Cloud Control Workshop

Rantzer, Anders; Westin, Eva

2014

Document Version:
Publisher's PDF, also known as Version of record

Workshop: Cloud control

LUND CENTER FOR CONTROL OF COMPLEX ENGINEERING SYSTEMS
LUND UNIVERSITY

Cloud control workshop

7-9 May 2014

Old Bishop's Palace at Biskopsgatan 1 in Lund

MAILING ADDRESS
Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND, SWEDEN

VISITING ADDRESS
Institutionen för Reglerteknik
Ole Römers väg 1
232 63 LUND

TELEPHONE
+46 46 222 87 87

FAX
+46 46 13 81 18

GENERIC E-MAIL ADDRESS
control@control.lth.se

WWW
www.lccc.lth.se

Printed: Media-Tryck, Lund, Sweden, Feb 2015

ISSN 0280-5316
ISRN LUTFD2/TFRT--7639--SE

Content

1. Introduction

LCCC workshops are organized in a 3-day format. About 20-25 speakers from academia and industry are invited for the workshop, selected for excellence and for an optimal coverage of the theme. The speakers are also encouraged to extend their stay beyond the workshop for further interaction with the local research environment. For each workshop, the research theme is chosen strategically to support the vision of a LCCC, usually with a cross-disciplinary perspective. An international scientific committee is responsible for the program.

1.1 WORKSHOP THEME

The Cloud Control Workshop was aimed to foster research in the multidisciplinary area of Cloud Control, leveraging expertise in areas such as distributed systems, control theory, autonomic computing, systems management, mathematical statistics, energy management, performance management, etc, to manage the cloud. The aim was to gather researchers from both academia and industry, and thereby promote new collaborations and ideas.

1.2 SCOPE

The workshop addressed challenges regarding the management and control of large-scale cloud infrastructures by bringing together the computer science community doing cloud management with the control community dealing with large-scale computing systems.

1.3 ORGANIZATION AND VENUE

The workshop was initiated by Maria Kihl (Dept. of Electrical and Information Technology, Lund University), Karl Erik Årzén (Dept. of Automatic Control, Lund University) and Erik Elmroth (Dept. of Computer Science, Umeå University).

The scientific committee consisted of Maria Kihl (chair), Karl Erik Årzén, Erik Elmroth, Tarek Abdelzaher (University of Illinois at Urbana-Champaign), Jie Liu (Microsoft Research), Bruno Sinopoli (Carnegie Mellon University), Vladimir Vlassov (Royal Institute of Technology), and Giovanni Toffetti (IBM Haifa Research Lab).

The local organization and interactions with workshop speakers and participants was handled by Eva Westin.

The workshop was held at the Bishop's Palace at Lund University, May 7-9 2014.

.

2. Group discussions

During the workshop, there were four group discussions on important issues in Cloud control. A short summary of each discussion follows.

2.1 OPEN PROBLEMS (FROM AN INDUSTRY PERSPECTIVE) IN CLOUD CONTROL

In this discussion, the focus was on research problems in Cloud control that are particularly interesting from an industry perspective.

The discussion was mainly focused on energy, which was believed as an important issue for industry. However, several of the industry representatives did not believe that turning off physical machines is a feasible solution for energy optimization. Instead, moving jobs to servers with low loads may be a better solution. Some types of jobs, for example big data jobs, can be placed wherever there is spare capacity. Also, real-time power optimization introduces one more complexity that may not be optimal for the system. Further, VMWare for example, has a solution for power reduction, but few customers use it. Also, there has been some work on virtualized batteries, which would be a way to make energy consumption sellable, that is, customers may buy a certain amount of power for their jobs.

Further, there was some discussion on application aware placement and infinite cloud architectures. For example, in Sweden, telecom operators are moving base station intelligence in to the cloud. Another example is cloud gaming applications. This will require new architectures with application-aware placement of components, which is a difficult challenge to solve.

2.2 CENTRALIZED VS. DECENTRALIZED CONTROL FOR LARGE-SCALE COMPUTING

This discussion was focused on issues related to centralized vs. decentralized control for large-scale computing. One general conclusion was that there is no single rule to apply on all of the problems. Also, when you want to design a system to either be centralized or decentralized, different aspects and tradeoffs need to be considered such as flexibility vs. agility and robustness, and scalability vs. optimality.

Generally speaking, the problems that we are dealing with in a cloud resource management domain are in principal so complex that centralized solutions will have their limitations no matter what. Instead, a feasible point of optimality should be found, and find decentralized solutions that benefits, for example, scalability and flexibility.

It was also mentioned that at some level of abstraction every system is naturally decentralized. The suggested solution should be a little bit of centralization and a lot of decentralization. From a design perspective, computation is better to be as decentralized as possible, while the data can be stored centrally.

2.3 ENERGY MANAGEMENT FOR DATA CENTERS

This discussion focused on several issues related to energy management for data centers. Much of the discussion was once more focused on the question of turning off and on machines. Also, it was discussed how exhaust energy and heat from data centers can be stored using batteries and water, respectively. The saved energy can be used to power surges in demand, and to flatten the energy usage curve. Additionally, there are

examples of where exhaust energy has been put to good use, heating adjacent homes, facilities, and waterbeds. However, taxation can make it unprofitable to sell exhaust energy.

It was agreed that data centers are not an efficient mean to generate heat and should not be considered a part of the energy infrastructure. Rather, effort should be directed towards reducing energy consumption through heat profiling and heat-aware placement. The discussion of VM placement and migration as a heat optimization parameter, and how challenging scaling down is, was reignited. Although it was agreed that migrating VMs would most likely require more energy in the proportion to the heat gained.

From a data center management perspective, the discussion came to evolve around the fact that different applications have different energy profiles that do not necessarily translate into what they pay for the service, depending on the type of workload, time of day, and the type en energy used. Thereafter the discussion was focused on how energy costs are transferred to the customer, and how application adminstrators are primarily concerned with overall performance. From where the session ended with a discussion on various energy saving schemes, ranging from energy aware hardware in the mobile industry to energy pricing profiles. It was proposed that energy should be provisioned and purchased separately by the application developer, but it was agreed that such a model would inhibit data-center flexibility and distract from the cloud paradigm.

2.4 RESOURCE OPTIMIZATION OF BURSTY WORKLOADS IN CLOUDS

This discussion was focused on topics related to resource optimization in clouds when there are bursty workloads with spikes. First of all, it is not easy to define a bursty workload. Load can be measured in several ways and requests can generate very different amount of processing. Also, bursts come at different time-scales and, therefore, it is not feasible to ask an "oracle" every millisecond. Interesting bursts are the ones that cannot be handled by "ordinary" control. Proactive (predictive) techniques should be used to detect spikes. Queuing theory could be used to analyze bursts, however few people use it and the question is how accurate M/M/Q models are. One input is that it is better to do something based on M/M/Q queues than just use ad-hoc methods.

Also, the topic of centralized vs. decentralized control came back. Centralized control should be used as much as possible, and then be complemented with decentralized control.

3. Panel Disscussion on ”Challenges for Cloud control

The workshop ended with a panel discussion on “Challenges for Cloud control”. The members of the panel were John Wilkes, Simon Tuffs, David Breitgand, and Geir Horn. A short summary of the discussion topics is give here.

Most of the discussion was focused on what challenges and problems the universities, and in particular PhD students should work on. A general conclusion was to always reach out to industry and solve real problems. One argument was that universities should work on fundamental problems that industry does not have time to solve. Availability is really important. Optimal solutions are not so important, better to have suboptimal solutions that are scalable and improve availability.

Another challenge is the lack of real data. Companies do not share data. One solution for this is to buy cloud capacity (which is very cheap) and run experiments directly on the cloud. However, this will not work for all research topics, for example scheduling. Another solution is to build your own cloud and bombard it with traffic, real or synthetic.

Part of the discussion was devoted to the notion of “Cloud”. One argument was that the term “Cloud” will disappear in a few years, instead everything will be called “Big Data” or something else. However, the fundamental challenges will remain so the future is bright.

Appendix A

PROGRAM

Wednesday, 7 May

09:30 Registration and coffee

10:00 Opening — Maria Kihl

10:15 Deploying complex applications to Google Cloud using Google Deployment Manager
Olia Kerzhner, Google
Cloud Control Systems - Real Time Analytics
Simon Tuffs, consultant
Energy and Resource Management Challenges in Data Centers
Tarek Abdelzaher, University of Illinois at Urbana Champaign

12:00 Lunch

13:00 Group Discussions
13:30 Group Presentation

14:00 Elasticity Manager for Elastic Key-Value Stores in the Cloud
Vladimir Vlassov, Royal Institute of Technology
Principles and Methods for Elastic Computing
Schahram Dustdar, Vienna University of Technology

15:00 Coffee

15:30 Exploring Autonomics for Clouds
Manish Parashar, Rutgers University
Thinking parallel: Multi-cores, virtual elasticity, and the application programmer
Geir Horn, University of Oslo, Norway
Simplified Cloud Control Using Dimension Reduction
Jianguo Yao, Shanghai Jiao Tong University

Thursday, 8 May

09:00 Real-time Performance Control of Elastic Virtualized Network Functions
Tommaso Cucinotta, Bell Labs, Alcatel-Lucent, Ireland
An Adaptive Utilisation Accelerator for Virtualized Environments
Giovanni Toffetti, IBM Haifa Research Lab

10:00 Coffee

10:30 Dynamic Power Management in Data Centers: Theory & Practice
Mor Harchol-Balter, Computer Science Department, Carnegie Mellon University
Reducing Power Delivery Costs for Cloud Data Centers
Bhuvan Urgaonkar, Penn State University
Brownout: Building More Robust Cloud Applications
Cristian Klein, Umeå University

12:00 Lunch

13:00 Group Discussions
13:30 Group Presentations

14:00 RT-Xen: Real-Time Virtualization for the Cloud
Chenyang Lu, Washington University in St. Louis
Service Level Agreement for Cloud Computing: Towards a Control-Theoretic Approach
Sara Bouchenak, University of Grenoble

15:00 Coffee

15:30 Performance-Energy Trade-off in Multi-Server Queueing Systems with Setup Delay
Samuli Aalto, Aalto University

17:30 Buses leave from Bangatan (green arrow on map)
18:30 Workshop Dinner at Turning Torso, Malmö

Friday, 9 May

09:00 LCCC activities in Cloud Control
Maria Kihl, Lund University
Capacity Management in IaaS Cloud
David Breitgand, IBM Research Haifa

10:00 Coffee

10:30 Deadline scheduling for big-data jobs and fault-tolerance of datacenter applications
Peter Bodik, Microsoft Research
Control issues in warehouse-scale datacenters
John Wilkes, Google
Application Performance Management in the Cloud using Learning, Optimization, and Control
Xiaoyun Zhu, VMware Inc.

12:00 Lunch

13:00 Modern Infrastructure: The Convergence of Network, Compute, and Data
Jason Hoffman, Ericsson
Event-based control: a way to reduce reconfiguration in autonomic computing
Nicholas Marchand, GIPSA lab, France
Guided tour through a cloud datacenter - The Umeå University approach to cloud resource management
Erik Elmroth, Umeå University

14:30 Coffee

15:00 Panel Discussion: Challenges in Cloud Control
15:30 Final Remarks and End of Workshop

Appendix B
PARTICIPANTS LCCC Focus Period Workshop on Cloud Control, May 2014

Aalto, Samuli	Aalto University	samuli.aalto@aalto.fi
Abdelzaher, Tarek	University of Illinois at U-C	zaher@illinois.edu
Ali Eldin Hassan, Ahmed	Umeå University	ahmeda@cs.umu.se
Årzén, Karl-Erik	Lund University	karlerik@control.lth.se
Åström Karl-Johan	Lund University	kja@control.lth.se
Berekmeri, Mihaly	GIPSA-lab, Grenoble	mihaly.berekmeri@gipsa-lab.grenoble-inp.fr
Bini, Enrico	Scuola Superiore St'Ana, Pisa	e.bini@sssup.it
Blomdell, Anders	Lund University	anders.blomdell@control.lth.se
Bodik, Peter	Microsoft	peterb@microsoft.com
Bouchenak, Sara	IMAG	sara.bouchenak@imag.fr
Breitgand, David	IBM	davidbr@il.ibm.com
Como, Giacomo	Lund University	giacomo.como@control.lth.se
Cucinotta Tommaso	Bell Labs, Alcatel-Lucent	tommaso.cucinotta@alcatel-lucent.com
Dellkrantz, Manfred	Lund University	manfred.dellkrantz@control.lth.se
Dustdar, Schahram	TU Wien	dustdar@infosys.tuwien.ac.at
Dürango, Jonas	Lund University	jonas.durango@control.lth.se
Eker, Johan	Lund University / Ericsson	johan.eker@control.lth.se
Elmroth, Erik	Umeå University	elmroth@cs.umu.se
Gambi, Alessio	Universitá della Svizzera Italiana	alessio.gambi@usi.ch
Gran, Ernst Gunnar	Simula Research Laboratory	ernstgr@simula.no
Harchol-Balter, Mor	Carnegie Mellon University	harchol@cs.cmu.edu
Hernandez, Francisco	Umeå University	francisco@cs.umu.se
Hoffman, Jason	Ericsson	jason.a.hoffman@ericsson.com
Horn, Geir	University of Oslo	geir.horn@mn.uio.no
Ibidunmoye, Olumuyiwa	Umeå University	muyi@cs.umu.se
Kerzhner, Olia	Google	olia@google.com
Kihl, Maria	Lund University	maria.kihl@eit.lth.se
Klein, Cristian	Umeå University	cristian.klein@cs.umu.se
Krzywda, Jakub	Umeå University	jakub@cs.umu.se
Landfeldt, Björn	Lund University	bjorn.landfeldt@eit.lth.se
Ljung, Peter	Sony Mobile	peter.ljung@sonymobile.com
Lorido-Botran, Tania	University of the Basque Country/ Umeå Univ.	tbotran@cs.umu.se
Lu, Chenyang	Washington University in St Louis	lu@cse.wustl.edu
Madjidian, Daria	Lund University	daria.madjidian@control.lth.se
Maggio, Martina	Lund University	martina.maggio@control.lth.se
Marchand, Nicolas	GIPSA-lab	nicolas.marchand@gipsa-lab.fr
Mehta, Amardeep	Umeå University	amardeep@cs.umu.se
Nilsson, Anders	Lund University	anders.nilsson@control.lth.se
Östberg, P-O	Umeå University	p-o@cs.umu.se

Papadopoulos, Alessandro	Lund University	alessandro.papadopoulos@control.lth.se
Parashar, Manish	Rutgers University	parashar@rutgers.edu
Rantzer, Anders	Lund University	anders.rantzer@control.lth.se
Robertsson, Anders	Lund University	anders.robertsson@control.lth.se
Robu, Bogdan	GIPSA-lab	bogdan.robu@gipsa-lab.grenoble-inp.fr
Sedaghat, Mina	Umeå University	mina@cs.umu.se
Skeie, Tor	Simula Research Laboratory/ Univ. of Oslo	tskeie@simula.no
Tärneberg, William	Lund University	william.tarneberg@eit.lth.se
Toffetti, Giovanni	IBM Haifa Research Lab	giovanni@il.ibm.com
Tomas, Luis	Umeå University	luis@cs.umu.se
Tordsson, Johan	Umeå University	tordsson@cs.umu.se
Truong, Hong-Linh	TU Wien	truong@dsg.tuwien.ac.at
Tuffs, Simon	consultant	simontuffs@gmail.com
Urgaonkar, Bhuvan	Penn State University	bhuvan@cse.psu.edu
Vlassov, Vladimir	KTH, Stockholm	vladv@kth.se
Wang, Cheng	Penn State University	cxw967@cse.psu.edu
Westin, Eva	Lund University	eva.westin@control.lth.se
Wittenmark, Björn	Lund University	bjorn.wittenmark@control.lth.se
Wilkes, John	Google	johnwilkes@google.com
Yao, Jianguo	Shanghai Jiao Tong University	jianguo.yao@sjtu.edu.cn
Zhu, Xiaoyun	VMware	xzhu@vmware.com

Appendix C

PRESENTATIONS

DEPLOYING COMPLEX APPLICATIONS TO GOOGLE CLOUD USING GOOGLE DEPLOYMENT MANAGER

Olia Kerzhner, Google

Deploying dynamic distributed applications in the cloud is hard. Google Deployment Manager is the new Google Cloud service that provides a simple templating mechanism that allows you to declaratively describe your solution, and then deploy it with a single command. Deployment Manager then provisions, scales, and monitors your solution. In this talk I will introduce Google Deployment Manager and demonstrate how it can be used to easily and repeatably declare and deploy dynamic systems in Google Cloud

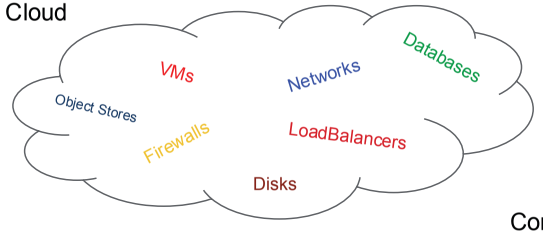
Google

Deploying complex applications to Google Cloud

Olia Kerzhner
olia@google.com

Google

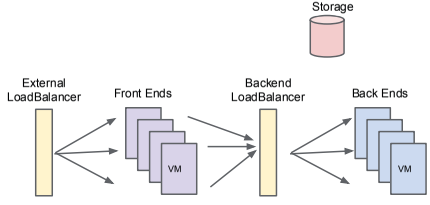
Cloud

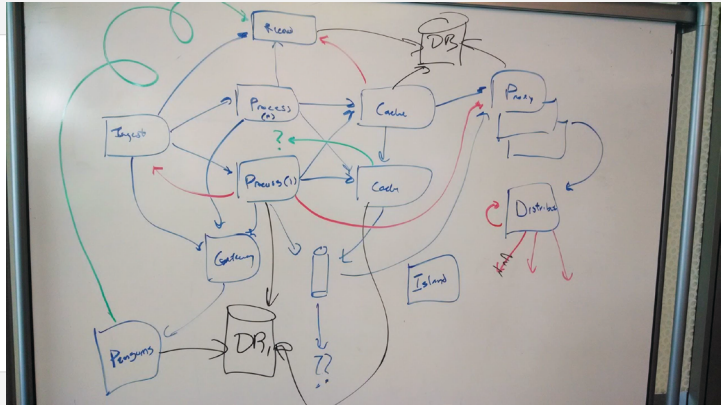


Control ..?

Google

Application stacks are complex





Google

Building on top of Google Compute Engine APIs

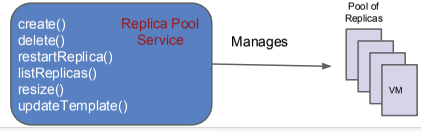
- GCE API are great for manipulating individual resources
- any minimally complex application requires users to write scripts or other code to deploy
- using scripts or manually creating resources results in "snowflake" deployments
- many open-source or proprietary tools try to solve this problem

Google

Introducing: Replica Pool API

Powerful new API to manage sets of homogenous VMs

- VMs are created based on a template
- declaratively specify what software to deploy
- VMs are monitored for health and are restarted on failed healthchecks
- the API supports the resize() operation that will add or subtract VMs to/from the set



Google

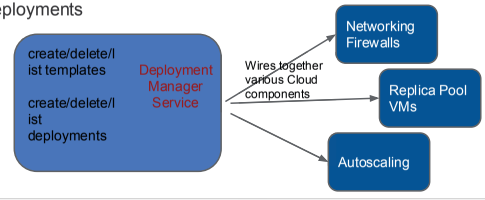
Managing more complexity

- need multiple sets of homogenous VMs: FrontEnds, BackEnds, Batch workers
- need load balancers, firewalls, networks, etc.
- need a way to tie the components together

Google

Introducing: Deployment Manager

- a single declarative JSON template to tie together different Cloud components to define an application stack
- templates are reusable, overrides are supported for customization of deployments



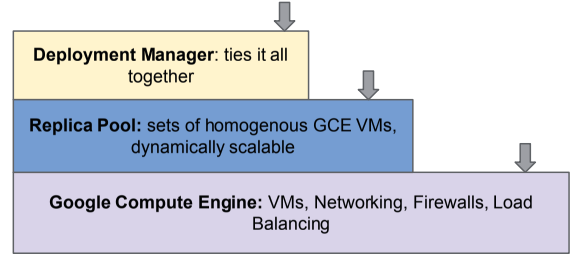
Google

Deployment Template

- a DECLARATIVE specification for how to deploy a set of Google Cloud resources together into an application stack
- defined in terms of "modules", each module corresponds to a Google Cloud API
- modules are connected together via module names and references

Google

Google Cloud Layered APIs



Google

Building Cloud-aware applications

What if you built your clustered application with the knowledge that it will run in the Cloud?

Google

Enabling Cloud-aware cluster setup

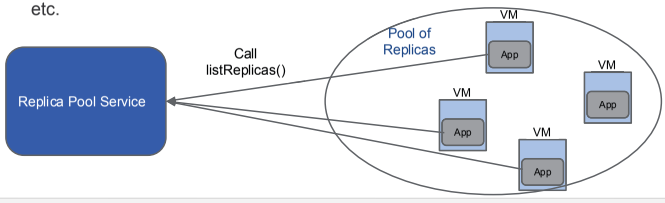
Traditional Cluster setup approaches:

- a hard-coded list of nodes supplied in the cluster on initialization
- connect peer-to-peer for discovery (complex configuration for peer connections)
- designating a master for set management
- brittle, complex, not scalable

Google

Replica Pool provides Set membership discovery

- there is now a REST endpoint that lists members (replicas) in a set
- adjusted dynamically as the set grows or shrinks
- Cloud-aware applications can use it to set up clusters, find peers, shard, etc.



Google

Summary

- **Replica Pool** is used to deploy scalable sets of homogenous VMs
- **Deployment Manager** is used to deploy and tie together multiple Replica Pools and other Cloud resources
- Setting up a cluster using Replica Pool allows for easy and robust cluster setup

Google

Give it a try!

- released to Limited Preview at the Google Cloud event in March
- documentation is public, usage requires whitelisting
- https://developers.google.com/deployment-manager
- https://developers.google.com/compute/docs/replica-pool

Try the sample templates, try writing your own. Send us your feedback!

CLOUD CONTROL SYSTEMS - REAL TIME ANALYTICS

Simon Tuffs, consultant

Event Driven, Service-Oriented Cloud Systems can be viewed as systems of multi-variable time-series data. This makes them amenable to analysis using discrete-time signal-processing and feedback control systems analytic techniques. Some illustrative examples from a real-world Cloud System are presented: Anomaly detection and classification using statistical tests, correlation, and causal inference based on service dependencies; Qualification of new code deployments into production, automated failure detection and recovery; and Auto-scaling challenges and remedies, with use of feed-forward predictors to survive outages, A general architecture for Cloud Systems Analytics is presented, together with a view of the interesting challenges ahead.

Cloud Control Systems - Real-Time Analytics

Dr. Simon Tuffs
Cloudstream

http://tinyurl.com/qfyuyn2

Background

- University of Oxford
  - Self Tuning Control Systems/GPC
- 25 years software industry: highlights
  - Iridium Ground Station (Motorola)
  - Spacestation Infrastructure (Boeing)
  - Cloud (Netflix)
- Theme: Software at Scale

This Talk

- Cloud Based Service Applications
  - Analytics & Control, how and why
- Cloud Applications As:
  - Multi-variable time-series systems
  - Amenable to signal processing
  - Resilient to failure using analytics
  - Operational using feedback control

Cloud Application Architectures

Cloud: Application Architectures

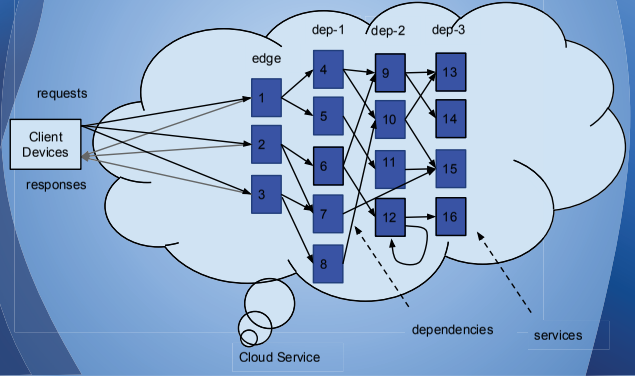
- Classes
  - Micro-service
    - Massive volume business operations (e.g. Netflix)
  - Big-Data
    - Terabytes of data, captured from streams into persistent stores.
    - Sparse compute intensive operations

Cloud: Application Architectures

- Cloud Application composed of services
- Services have dependencies (graph)
- Requests flow from the edge down
- Reponses flow back to edge
- Latencies accumulate forwards
- Errors propagate backwards
- Services are developed independently

Cloud: Application Architectures

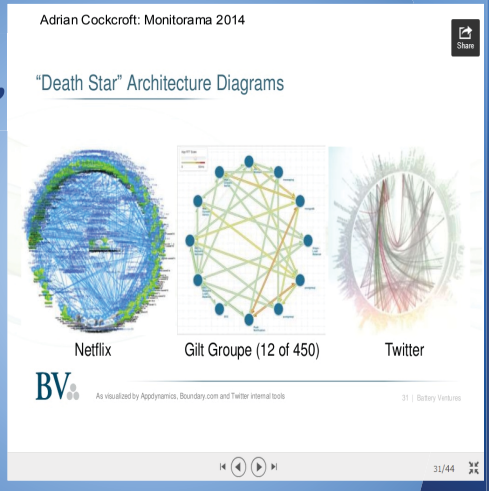
Services & Dependencies:



Cloud: Application Architectures

- But it isn't that simple...
- A real dependency graph...
- "Death Star Architecture"

"We are not alone"



Cloud: Application Architectures

- Complex beyond human comprehension
- Nonlinear
- Time-varying
- Partially predictable
- Potentially chaotic
- The worst kind of "system"

Analytics are not optional, they are essential

Cloud Applications: Analytics Classes

- Operational
  - Availability, fault detection, repair, peformance optimization
- Business Intelligence
  - how much money are we making?
  - how many customers did we just lose?
  - how can we make more money?

Cloud Applications: Monitoring

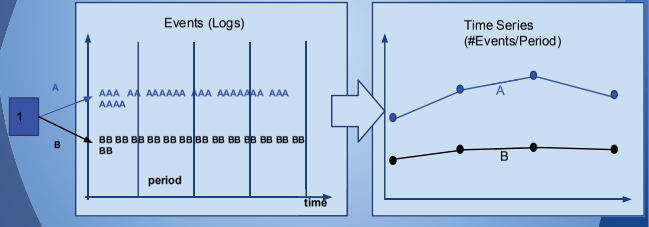
- To analyze you must monitor
- How do you handle billions of events?
- How do you transform them for analytics?

Cloud Applications: Monitoring

- Instrument services:
  - to expose internal details (e.g. type of errors, versus HTTP 503's)
- With significant request volume:
  - monitored events become statistically driven time-series
  - signal processing methods then apply

Cloud: Monitoring

- From Events To Time Series:

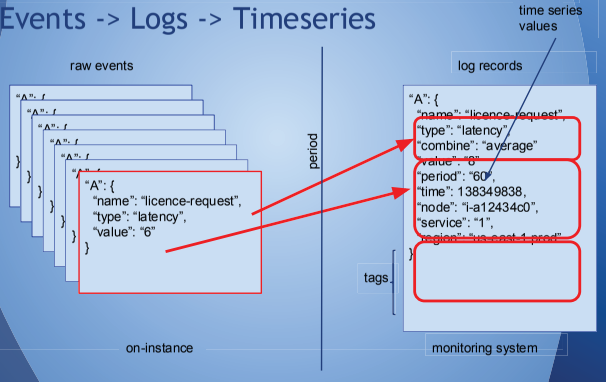


Cloud: Monitoring Architecture

- Convert events to time-series (coordinate transform)
  - bucket by period
  - classify & tag
  - store for query/retrieval
- Reduces dimension of data by many orders of magnitude
  - -> Real Time Analytics become feasible

Cloud: Monitoring Architecture

- Events -> Logs -> Timeseries



What to Monitor?

- "Assume that any metrics not being analyzed will turn out to be garbage"
  - Adrian Cockcroft, Architect Netflix Cloud
- Instrument to measure:
  - health (success, failure)
  - performance (load, cpu)
  - availability (timeouts, fallbacks)
  - resources (disk i/o, memory, handles),
  - sla's (latency)

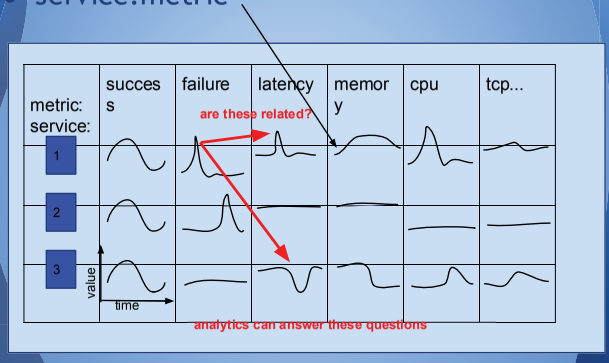
Visualization as an Analytic

Service Metric Visualization

- Classify metrics by type
- View services as rows of service:metrics
- Patterns start to emerge between visually.
- This scales to 100's of services and metrics (make the graphs small, human visual cortex sees patterns)

Cloud: Visualizing

- service:metric



Beyond Visualization: Computational Analytics

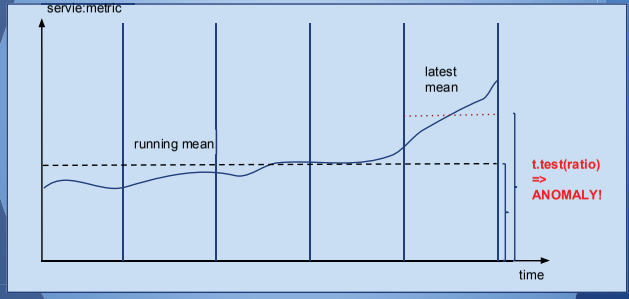
Anomaly Detection & Diagnosis

Anomaly Detection

- Look at a service:metric
- Is it behaving normally, or is it showing signs of distress?
- How can we automate this?
- Without lots of configuration?
- In a scale invariant way?
- Use a mean-shift analytic...

Analytics for Anomalies?

- mean? variance?

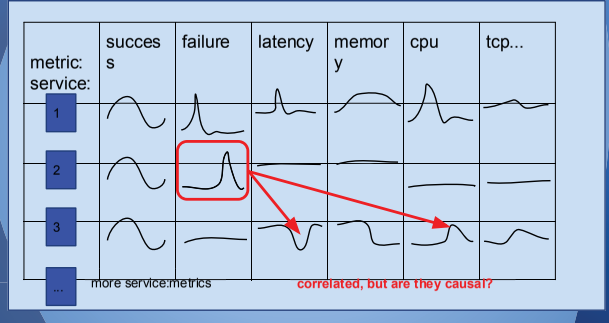


Analytics for Anomalies?

- You found an anomalous service:metric, now what?
- Correlate against *all other* service metrics
- This is fast (<0.1s for 400sm in R)

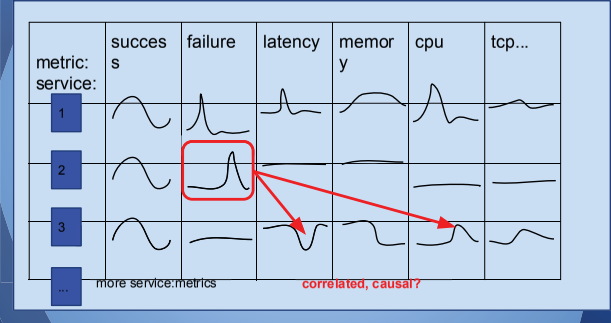
Correlate

- Pearson + mean removal



Filter

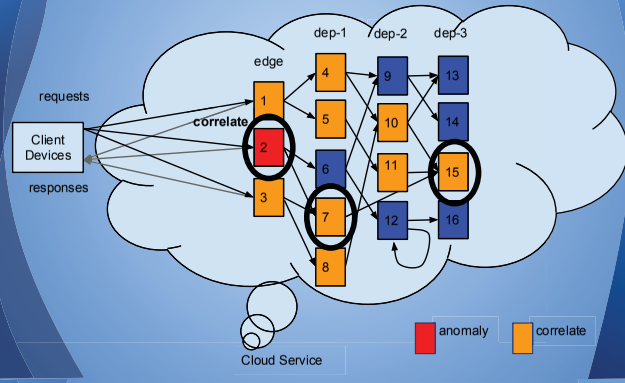
- Increase signal-to-noise:



Can we do more?

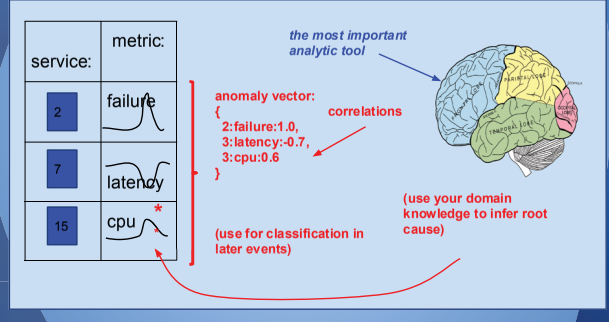
- Correlation x Dependency = Probable Cause

Anomaly -> Correlation -> Cause



Classify and Decide.

- Prune with dependency tree



Build a model

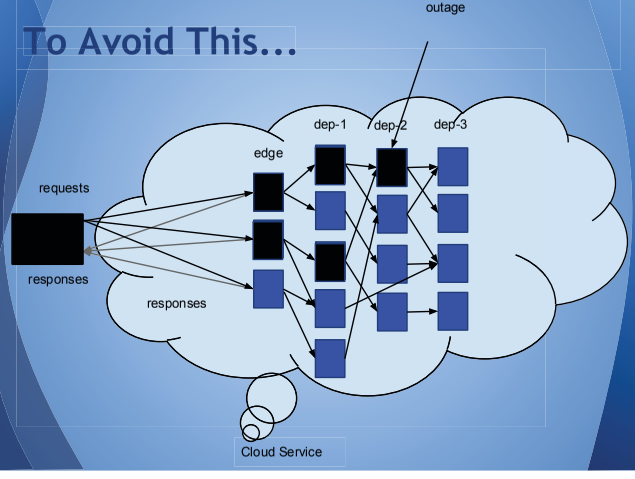
- Persist this pattern for future causal analysis
- Did we see this anomaly vector before?

Canary Analysis (deployment)

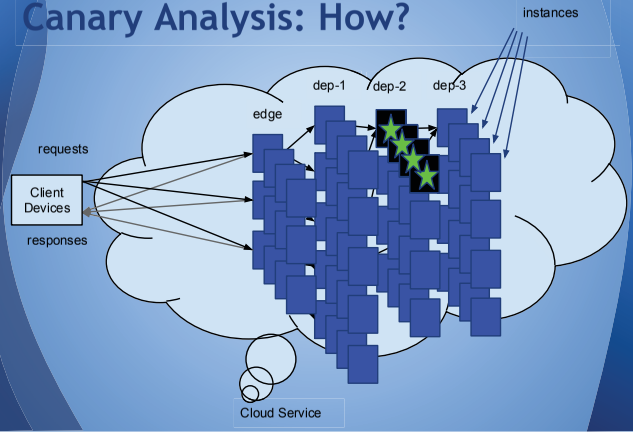
Canary Analysis Defined

- For a given service:
  - Deploy new code to limited #instances
  - Analyze against existing production code
  - Decide whether good or bad
  - Push forward (upgrade all service instances)
    - or roll back.

To Avoid This...



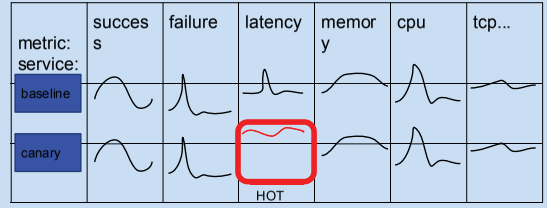
Canary Analysis: How?



Canary Analysis

- How does this work?
  - Service metric grid (again), 2 rows.
  - Compare canary to baseline, statistical tests.

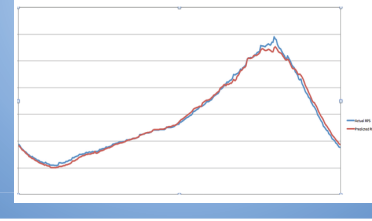
Automated Canary Analysis



Autoscaling

Load Based Autoscaling

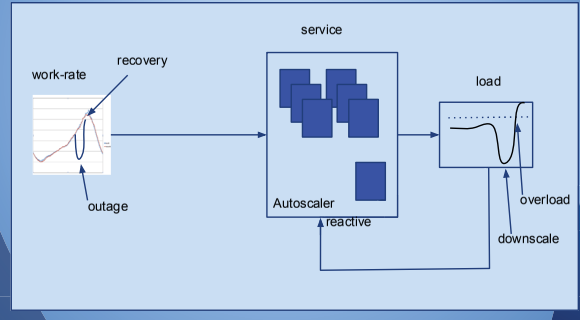
- increase #instances when load increases
- decrease #instances when load decreases
- works well...



Except when it doesn't..

- During an outage, load drops
- Instances are terminated
- Service becomes underprovisioned for return to normal request rate
- Overload occurs
- Other services suffer.
- Chaos.

Reactive Autoscaling

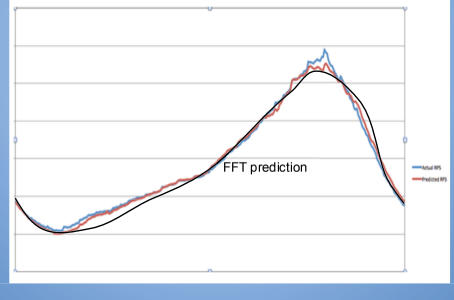


How do you avoid this?

- Use feedforward control
- Base on prediction of request rate
- Simple application of FFT low-pass filter.

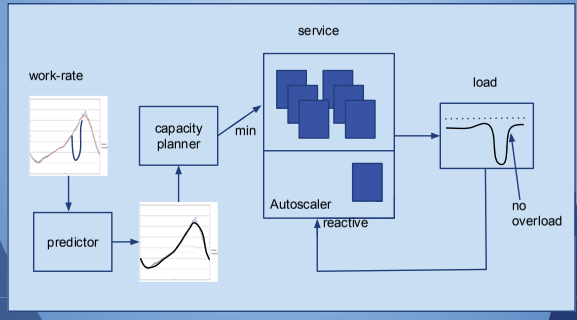
Scryer

- FFT based prediction



Netflix: Scryer

- Predictive+Reactive = Feedback Control



Real Time Analytics Engine

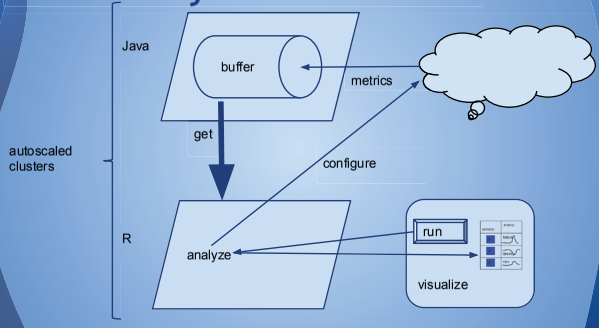
Analytics at Scale

- How do you do analytics at scale?
  - Do monitoring at scale
  - Do data-collection & buffering at scale
  - Run Analytics at scale
  - Use the Cloud to achieve scale.
- (But use a different Cloud).

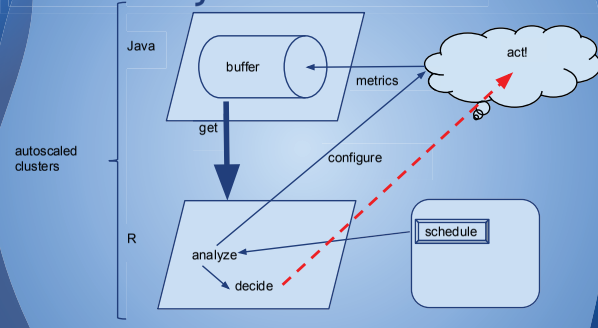
Analytics at Scale

- One possible architecture: Java and R engines in the Cloud
  - gathering data
  - running analytics
  - performing visualization
  - doing notification

Cloud Analytics: Interactive



Cloud Analytics: Automated



Analytics Challenges

Cloud Analytics: Big Challenges

- instance outlier detection at scale
- tuning queues & timeouts for services
- detection of overload/underprovision
- anomaly detection (prediction)
- behavior pattern classification
- automatic alert tuning
- “closing the loop”

“Cloudstream”

Cloudstream

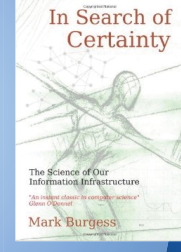
https://github.com/simontuffs/cloudstream/wiki

- Cloudstream Stack:
  - Netflix OSS, Open/CPU, iPython, Cloudsim, Amazon/Kinetics Netflix/Suro Storm/Spark
- Real-Time Analytics, in the Cloud, for the Cloud.
  - Currently building an application simulator
  - Design & train analytics

Questions?

Recommendation:

- Mark Burgess
  - In Search Of Certainty, 2013
  - Views information systems from a physics perspective, showing the non-deterministic complexity we are creating, and how hard it is to manage



Caution!

- Please seek a second opinion before spending years building a Ph.D. out of the following speculations & observations….

A Posteriori Observations

- Focus on $ not KWh for allocation
  - (they are isomporphic)
  - $ drive customer behavior the right direction
- Consider standardizing on "Model Predictive Controls" (e.g. GPC)
  - Superset all other linear methods, save time :)
- Most of my challenges do not close any control loops
  - other than estimation/modeling loops

A Posteriori Challenges

- Monitoring Validation
  - Our Cloud is down! Our Monitoring is down!
  - How can you tell?
- Avoid WOM (write-only monitoring)
  - how to aggregate useful data without losing information but still do analytics
- Causality
  - Infer dependency graph from data?
  - Cross-covariance for causation.

A Posteriori Challenges

- Develop Cloud invariants/assertions as "models of behavior"
  - increased latency => upstream errors
  - upstream errors => downstream request drop
  - increased cpu => increased latency
  - increased requests => increased (cpu, load)
  - parameterize & tune a behavioral model base on these invariants.

A Posteriori Challenges

- Machine learning (SVM, markov models)
  - Behavioral classification
  - Failure identification
- Evidence based learning
  - Bayesian networks for fault detection.
- Better predictors
  - Wavelets, basis functions.
- Modeling the Cloud
  - Dynamic Equilibrium
  - Transient Dynamics
  - "Kalman" Filtering

A Posteriori Challenges

- Auto-tune configuration parameters *(close the loop)*
  - 99.5% latency ⇔ errors => need to increase caller timeouts.
  - 99.5% latency ⇔ load => need to scale up if at the "knee".
  - 99.5% queue size ~ max-size => need to add worker threads
  - do this in production, across operating ranges

Thankyou!

ELASTICITY MANAGER FOR ELASTIC KEY-VALUE STORES IN THE CLOUD

Vladimir Vlassov, Royal Institute of Technology

The increasing spread of elastic Cloud services, together with the pay-as-you-go pricing model of Cloud computing, has led to the need of an elasticity controller. The controller automatically resizes an elastic service in response to changes in workload, in order to meet Service Level Objectives (SLOs) at a reduced cost. However, variable performance of Cloud Virtual Machines and nonlinearities in Cloud services, such as the diminishing reward of adding a service instance with increasing the scale, complicates the controller design. First, we briefly discuss challenges and some approaches to automation of elasticity of a cloud-based storage, and, then, present the design and evaluation of ElastMan, an elasticity controller for Cloud-based elastic key-value stores. ElastMan combines feedforward and feedback control. Feedforward control is used to respond to spikes in the workload by quickly resizing the service to meet SLOs at a minimal cost. Feedback control is used to correct modeling errors and to handle diurnal workload. To address nonlinearities, our design of ElastMan leverages the near-linear scalability of elastic Cloud services in order to build a scale-independent model of the service. We have implemented and evaluated ElastMan using the Voldemort key-value store running in an OpenStack Cloud environment.

Our evaluation shows the feasibility and effectiveness of our approach to automation of Cloud service elasticity.

Elasticity Manager for Elastic Key-Value Stores in the Cloud

Vladimir Vlassov vladv@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

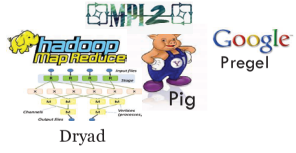
Joint work with: Ahmad Al-Shishtawy, SICS

Cloud Control Workshop, Lund University, 7-9 May 2014

Motivation

- Web 2.0 applications
  - WiKis, social networks, media distribution and sharing
- Data-intensive applications; big data

- Challenges
  - scalability, elasticity
    - Rapidly growing number of users and amount of user-generated data, data-intensive applications
  - load balancing, latency
    - Uneven load; users are geographically scattered
  - availability
    - Partial failures, very high load, load spikes
  - consistency guarantees
    - Data-centric applications and services



Cloud-Based Services and Applications

- **Clouds** provide the illusion of the infinite amount of resources
- Pay-as-you-go pricing model
  - High load: Allocate more resources to improve performance
  - Low load: Release resources to save money
- Enables **Cloud-based *Elastic* Services and Applications**

The Need for Elasticity

- Web services and applications frequently experience high workloads
  - A service can become popular in just an hour

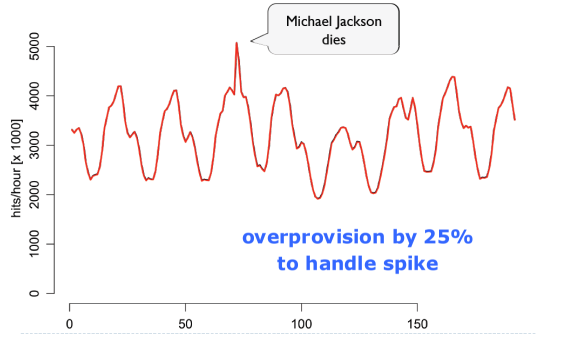
- The high level load does not last for long and keeping resources in the Cloud costs money

- This has led to ***Elastic Computing***
  - Ability of a system to grow and shrink at run-time in response to changes in workload
  - Cloud computing allows on-the-fly requesting and releasing VMs to scale/resize the service in order to meet SLOs at a minimal cost

Elasticity

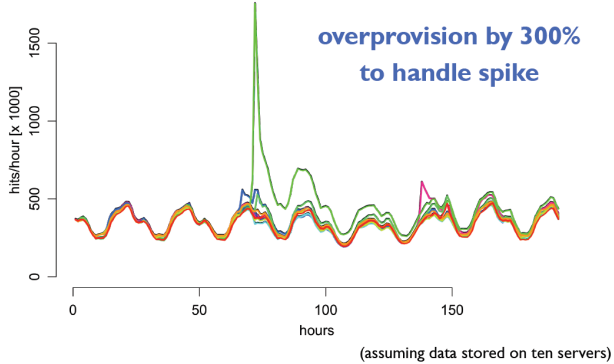
- In Physics, **Elasticity** is "*the property of a body or substance that enables it to resume its original shape or size when a distorting force is removed*" [The Free Dictionary]
- In Cloud computing, **Elasticity** is the ability to scale resource usage up and down [rapidly] according to [instantaneous] demand
  - The ability of a system to scale up and down (grow and shrink by requesting and releasing resources) in response to changes in its environment, workload, and QoS requirements

Wikipedia workload trace - June 2009



[From: "Solving the Scalability Dilemma with Clouds, Crowds, and Algorithms", Michael Franklin, UC Berkeley]

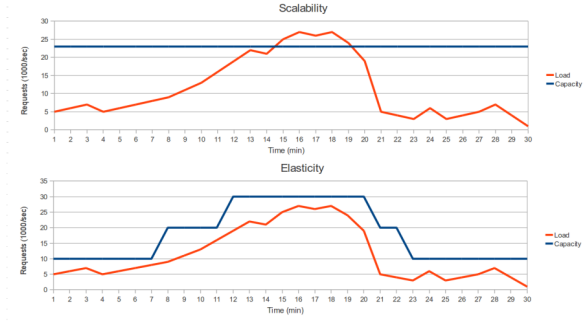
Over-provisioning a Storage System



(assuming data stored on ten servers)

[From: "Solving the Scalability Dilemma with Clouds, Crowds, and Algorithms", Michael Franklin, UC Berkeley]

Elasticity versus Static Provisioning



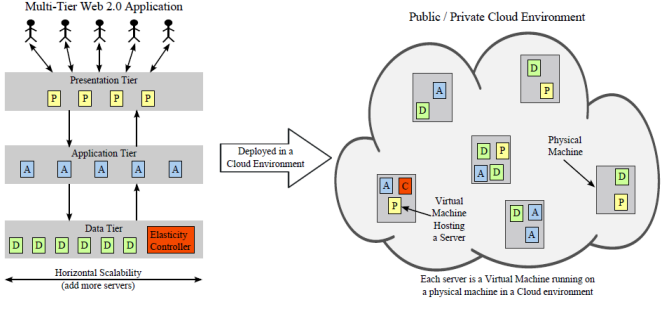
Outline

- Automation of elasticity
- Elasticity control for a cloud-based elastic storage
  - Requirements; challenges; approaches
- ElastMan: an elasticity manager for Cloud-based key-value stores
  - Feedforward control to handle workload spikes
  - Feedback control to handle gradual workload changes
- Evaluation of ElastMan in Voldemort key-value store
- Conclusions

Automation of Elasticity

- Elasticity can be controlled either manually (by the sys-admin) or automatically (by a autonomic manager)
- Automation of elasticity can be achieved by providing an **Elasticity Controller**
  - Helps to avoid SLO violations while keeping the cost low
  - Automatically adds/removes VMs (servers, service instances) in response to changes in some SLO metrics, e.g., access latency, caused by changes in workload
- Can be built using elements of Control Theory and/or Machine Learning techniques
  - Feedback-loop (a.k.a. closed-loop) control
  - Model Predictive Control (MPC)

Target System



Storage Services. Key-Value Stores

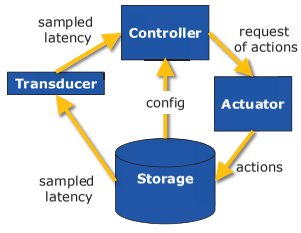
- Storage systems designed for horizontal scalability, such as key-value stores
  - minimum functionality: `get(key)` and `put(key, value)`
  - horizontal scalability, load balancing and replication
- Examples
  - Yahoo! PNUTS
  - Google BigTable
  - LinkedIn Voldemort
  - Apache Cassandra
  - UCB's SCADS
  - File systems, e.g., Hadoop Distributed File System

An Elasticity Controller for Cloud-Based Key-Value Stores

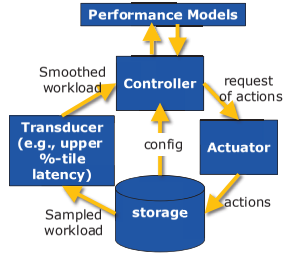
- Objective
  - To meet SLOs at a minimal cost by controlling the elasticity (size) of Cloud-based key-value stores by adding/removing resources (VMs, storage nodes)
- SLO Examples
  - Average read latency in one minute interval is less than 10ms
  - 99% of reads in one minute interval are performed in less than 10ms per read

Approaches to Automated Elasticity Control for a Cloud-Based Storage

- Closed-loop control
  e.g., [Lim2010][Mou2012]



- Model Preditive Control
  e.g., [Tru2011]



Requrements for Elastic Storage

- **Horizontal scalability**
  - The storage and I/O capacity scales (roughly linearly) with the number of the active servers.
- **Touch points**
  - Sensors to monitor workload and performance (e.g., read latency);
  - Actuators to add/remove resources and service instances
- **Load balancing (re-balancing)**
  - Distribute data across servers to effectively balance load;
  - Redistribute (rebalance) data in response to join/leave events
- **Replication**
  - For robust availability: enough to avoid interruptions on leave events

Challenges of Elasticity Control for a Cloud-Based Storage

- Actuator delays due to data movement (rebalancing)
- Interference with applications and sensor measurements
- Discrete storage units
- Nonlinearity due to diminishing reward of adding a storage unit with increasing scale
- 99th percentile of access latency is a relatively noisy signal
- VM performance is difficult to model and predict
- Highly dynamic workload that is composed of both gradual (diurnal) and sudden (spikes) variations

Touch-points

Sensors
- To monitor workload and performance (e.g., access latency)
- A controller input includes one or more system performance metrics (system outputs)
- Requirements to system metrics [Lim 2010]
  - Easy to measure accurately
  - Should expose the system(tier)-level behavior or performance
  - Should be reasonably stable
  - Should correlate to the measure of service level specified in SLO

Actuators
- To add/remove resources and service instances
  - Cloud APIs to request/release server instances (VMs)
  - Storage API to request handling joins and leaves and rebalancing

Sensors (1/3)

SLO metrics (what to sense/measure/monitor)
- Request latency
  - response time, service latency, download latency, etc.
  - End-user /client experience, QoS
- System-wide performance metrics
  - CPU utilization, memory usage, network utilization
- Cost
- Other metrics

Sensors (2/3)

- In many cases, a system performance metric strongly correlates with the overall request latency (response time)
- Performance counters (CPU/memory/network utilizations), can be used as sensors

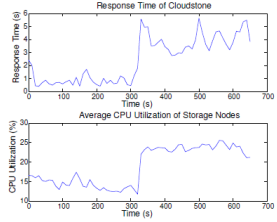


Figure 2: Cloudstone response time and average CPU utilization of the storage nodes, under a light load and a heavy load that is bottlenecked in the storage tier. CPU utilization in the storage tier correlates strongly with overall response time (the coefficient is .88), and is a more stable feedback signal.

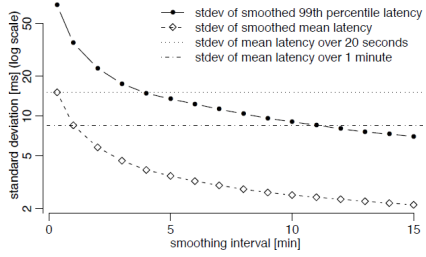
[Lim2010]

Sensors (3/3)

SLO requirements (what to regulate)
- Average values (means) of the SLO metrics
  - E.g., an average response time, an average CPU utilization
  - Classical closed-loop control
  - Example: HSC integral controller of HDFS-based storage [Lim2010]
- Upper quantiles of the SLO metrics
  - E.g. 99th percentile of latency
    "99% of all requests must be answered within 100ms"
  - Model Predictive Control
  - Example: SCADS Director for SCADS storage (UCB) [Tru2011]

Mean versus Upper Quatile

- Usually, upper percentiles of latency measurements are very "noisy"
- A noisy latency signal can cause oscillations in classical closed-loop control



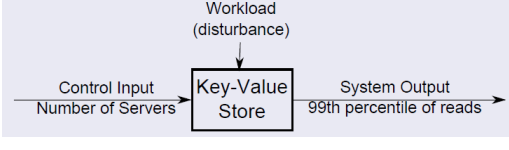
Standard deviation for the mean and 99th percentile of latency for increasing smoothing window sizes. [Tru2011]

ElastMan: Elasticity Manager for Cloud-Based Key-Value Stores

- Addresses the challenges
  - the variable performance of VMs,
  - dynamic workload (spikes, diurnal changes),
  - stringent performance requirements,
- Combines and leverages the advantages of feedback and feedforward control
- Once designed, the controller can operate for different sizes of the key-value store
- Evaluated with LinkedIn's Voldemort key-value store deployed in our private OpenStack Cloud

Modeling the Store (1/2)

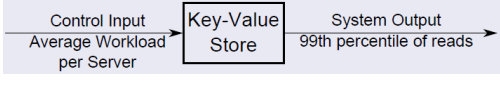
Typical approach



- Non linear Model
  - Adding 1 node to a 1 node system –> doubles the performance
  - Adding 1 node to a 100 nodes system –> only 1% improvement
- Workload treated as disturbance

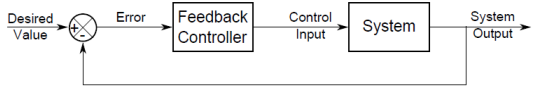
Modeling the Store (2/2)

ElastMan approach



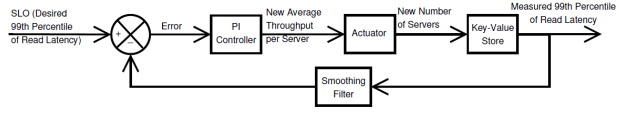
- Control the number of servers indirectly by controlling the average workload per server
- Relies on near linear scalability of key-value stores

Feedback Control [Hel2004]

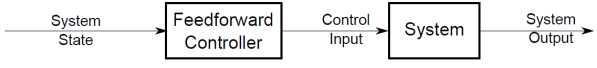


- The system's output (e.g., response time) is being monitored
- Controller calculates the control error
- Controller changes the control input (e.g., number of servers to add or remove) according to the amount and sign of the control error
- Advantage: controller can adapt to disturbance
- Disadvantages: oscillation, overshoot, possible instability

ElastMan Feedback Controller

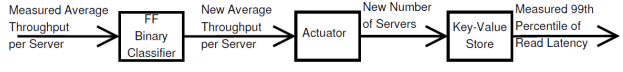


Feedforward Control [Hel2004]

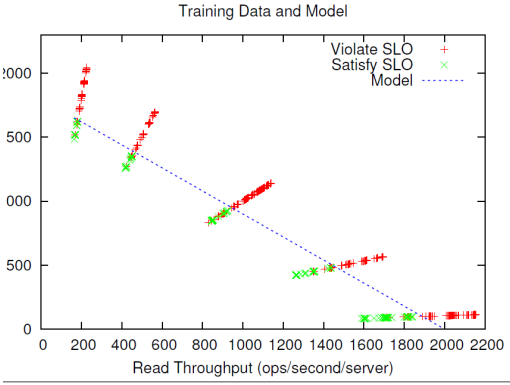


- The system's output is not monitored
- Other system states and variables are monitored
- Controller relies on a model of the system to calculate necessary
- change
- Advantages: faster and avoids oscillation and overshoot
- Disadvantages: sensitive to unexpected disturbances that are not modeled

ElastMan Feedforward Controller



ıary Classifier



ElastMan: Combining Feedback and Feedforward Control

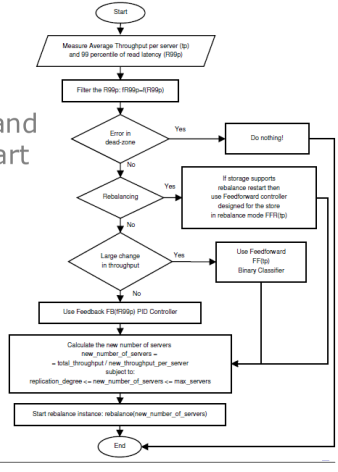
Feedback

- Classical PI controller
- Deals with diurnal workloads (e.g., slow day-night changes)
- Monitors 99th percentile of read latency
  - E.g., the read latency of 99% of reads in a 1 min interval is at most 10ms
- Can tolerate and adapt to modeling errors

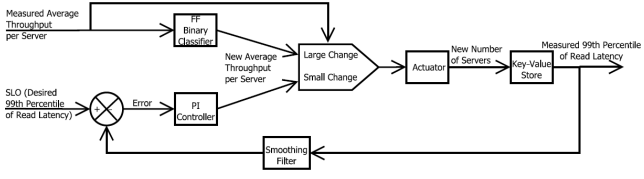
Feedforward

- A binary classifier using logistic regression
- Deals with workload spikes (large rapid changes)
- Monitors workload (intencity of reads and intencity of writes)
- Allows smoothing the noisy 99th percentile signal

ıed Feedback and rward Flow Chart



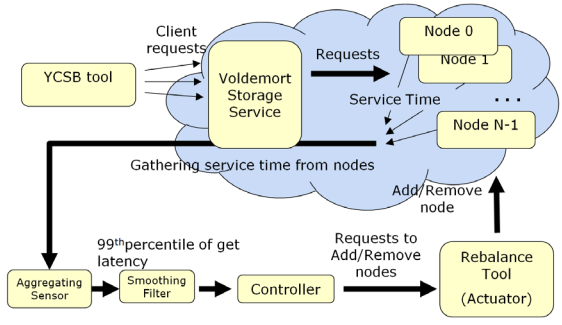
Combined Feedback and Feedforward Controller



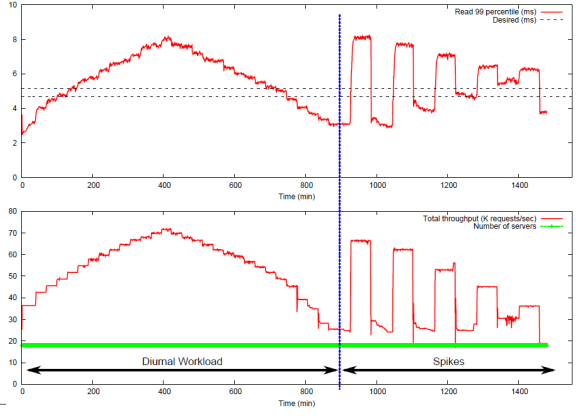
Evaluation

- Implemented a prototype of the ElastMan Elasticity Controller
- Evaluated with LinkedIn's Voldemort key-value store
- Deployed in our private OpenStack Cloud

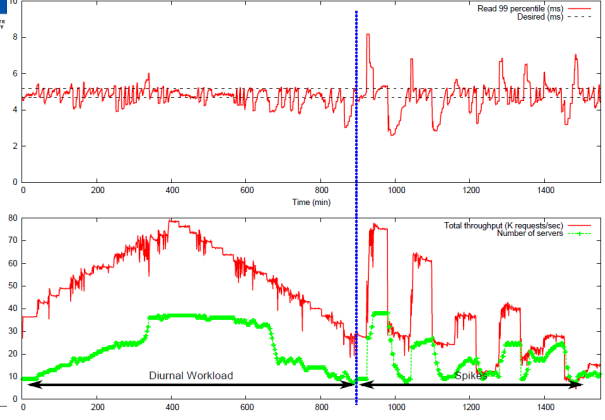
Voldemort Key-Value Store with the ElastMan Elasticlity Controller



Voldemort performance without ElastMan



Voldemort performance with ElastMan



Conclusions

- ElastMan addresses the challenges of the variable performance of Cloud VMs, dynamic workload, and stringent performance requirements
- ElastMan combines and leverages the advantages of both feedback and feedforward control
  - feedforward control quickly responds to rapid changes in workload
  - feedback controller handles diurnal workload and to correct modeling errors in the feedforward control
- Evaluation results show the feasibility of the ElastMan approach

ElastMan References

- Ahmad Al-Shishtawy, Vladimir Vlassov, ***ElastMan: Elasticity Manager for Elastic Key-Value Stores in the Cloud***, The ACM Cloud and Autonomic Computing Conference (CAC 2013), Miami, FL, USA, August 5-9, 2013.
- Ahmad Al-Shishtawy, Vladimir Vlassov, ***ElastMan: Autonomic Elasticity Manager for Cloud-Based Key-value Stores***,The 22nd ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC '13). ACM, New York, NY, USA, pp. 115-116.

- http://www.ict.kth.se/~ahmadas/ElastMan/

References

- **[Hel2004]** J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems. Wiley-IEEE Press, 2004.*
- **[Lim 2010]** Harold C. Lim, Shivnath Babu, and Jeffrey S. Chase, *Automated control for elastic storage*. ICAC '10, 2010
- **[Mou2012]** M. A. Moulavi, A. Al-Shishtawy, and V. Vlassov, *State-Space Feedback Control for Elastic Distributed Storage in a Cloud Environment*, ICAS 2012
- **[Tru2011]** Beth Trushkowsky, et al., *The SCADS director: scaling a distributed storage system under stringent performance requirements*. The 9th USENIX Conf on File and Storage Technologies (FAST'11), 2011

PRINCIPLES AND METHODS FOR ELASTIC COMPUTING

Schahram Dustdar, Vienna University of Technology

Elasticity is seen as one of the main characteristics of Cloud Computing today. Is elasticity simply scalability on steroids? In this talk I will discuss the main principles of elasticity, present a fresh look at this problem, and examine how to integrate people, software services, and things into one composite system, which can be modeled, programmed, and deployed on a large scale in an elastic way.

Principles and Methods for Elastic Computing -

Lund, 7 May 2014

Schahram Dustdar

Distributed Systems Group
TU Vienna

http://dsg.tuwien.ac.at/research/viecom/

Acknowledgements

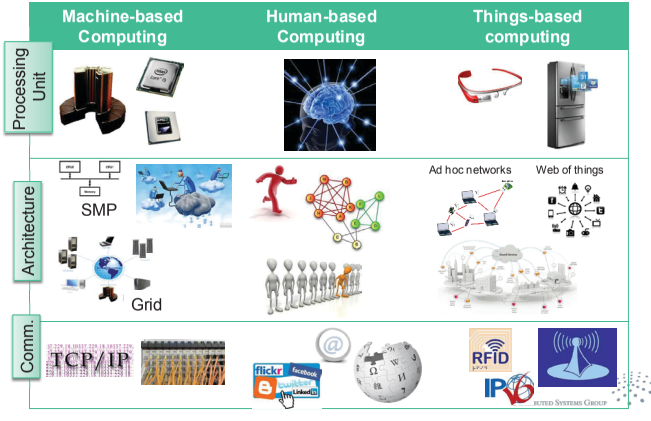
Includes some joint work with Hong-Linh Truong, Alessio Gambi, Muhammad Z.C. Candra, Georgiana Copil, Duc-Hung Le, Daniel Moldovan, Stefan Nastic, Mirela Riveri, Sanjin Sehic, Ognjen Scekic

NOTE: The content includes some ongoing work

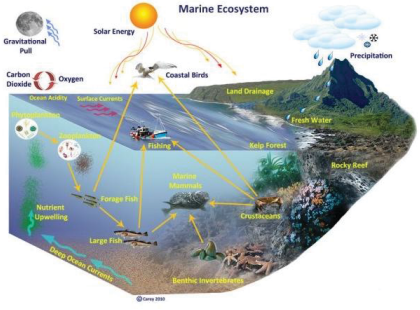
Smart Evolution – People, Services,Things

Computing Models

S. Dustdar, H. Truong, "Virtualizing Software and Humans for Elastic Processes in Multiple Clouds – a Service Management Perspective", in *International Journal of Next Generation Computing*, 2012



Think Ecosystems: People, Services, Things



Diverse users with complex networked dependencies and intrinsic adaptive behavior – has:

1. **Robustness mechanisms**: achieving stability in the presence of disruption
2. **Measures of health**: diversity, population trends, other key indicators

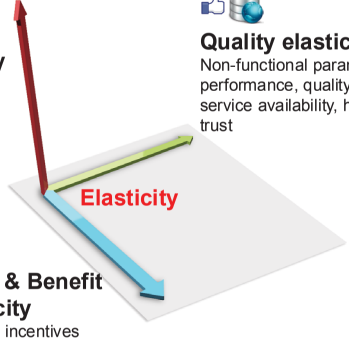
Our Approach

- **Unified service unit model** (Consumption, ownership, provisioning, price, function, etc.)
- **Connecting Data Centers to IoT**
  - From physically isolated verticals to virtual verticals
  - Software-defined elastic data centers and IoT ecosystems
  - SD units are described with well-defined API
  - Provisioning units for customized gateways
  - Dynamically composing units into runtime topologies
  - Runtime controlling and optimization via configuration policies (DevOps principle)
- **Human Augmentation**
  - Human computation capabilities under elastic service units
  - Programming human-based units together with software-based units

Elasticity ≠ Scaleability

Resource elasticity
Software / human-based computing elements, multiple clouds

Quality elasticity
Non-functional parameters e.g., performance, quality of data, service availability, human trust

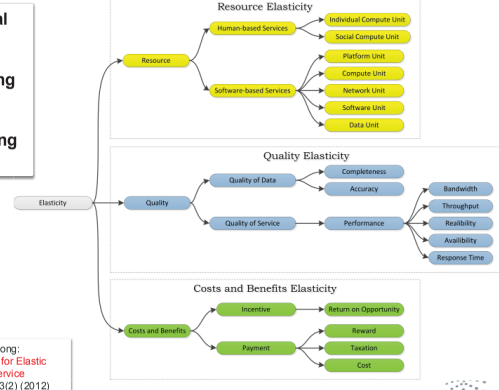


Costs & Benefit elasticity
rewards, incentives

Vienna Elastic Computing Model

dsg.tuwien.ac.at/research/viecom

- **Multi-dimensional Elasticity**
- **Service computing models**
- **Cloud provisioning models**



Schahram Dustdar, Hong Linh Truong: Virtualizing Software and Humans for Elastic Processes In Multiple Clouds- a Service Management Perspective. IJNGC 3(2) (2012)

Diverse types of elasticity requirements

- ***Application user***: "If the cost is greater than 800 Euro, there should be a scale-in action for keeping costs in acceptable limits"
- ***Software provider***: "Response time should be less than amount X varying with the number of users."
- ***Developer***: "The result from the data analytics algorithm must reach a certain data accuracy under a cost constraint. I don't care about how many resources should be used for executing this code."
- ***Cloud provider***: "When availability is higher than 99% for a period of time, and the cost is the same as for availability 80%, the cost should increase with 10%."

Data Center - Engineering Techniques

High Level Description of Elasticity Requirements

SYBL (Simple Yet Beautiful Language) for specifying elasticity requirements

SYBL-supported requirement levels

1. Cloud Service Level
2. Service Topology Level
3. Service Unit Level
4. Relationship Level
5. Programming/Code Level

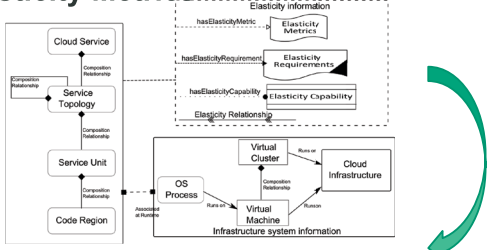
```
#SYBL.CloudServiceLevel
Cons1: CONSTRAINT responseTime < 5 ms
Cons2: CONSTRAINT responseTime < 10 ms
WHEN nbOfUsers > 10000
Str1: STRATEGY CASE fulfilled(Cons1) OR
fulfilled(Cons2): minimize(cost)

#SYBL.ServiceUnitLevel
Str2: STRATEGY CASE ioCost < 3 Euro :
maximize( dataFreshness )

#SYBL.CodeRegionLevel
Cons4: CONSTRAINT dataAccuracy>90%
AND cost<4 Euro
```

Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, Schahram Dustdar, **"SYBL: an Extensible Language for Controlling Elasticity in Cloud Applications"**, 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 14-16, 2013, Delft, Netherlands

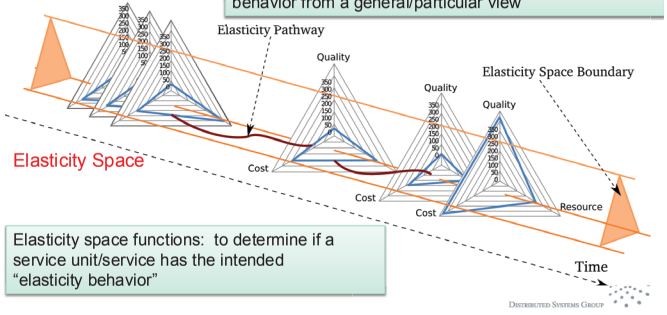
Mapping Services Structures to Elasticity Metrics



Elasticity Model for Cloud Services

Moldovan D., G. Copil, Truong H.-L., Dustdar S. (2013). **MELA: Monitoring and Analyzing Elasticity of Cloud Service. CloudCom 2013**

Elasticity Pathway functions: to characterize the elasticity behavior from a general/particular view



Elasticity space functions: to determine if a service unit/service has the intended "elasticity behavior"

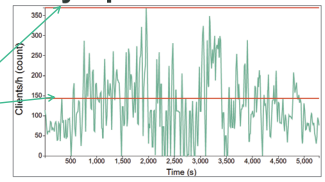
Multi-Level Elasticity Space

Service requirement

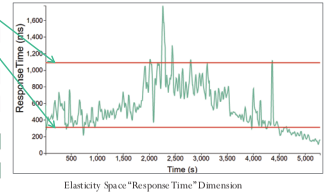
COST<= 0.0034$/client/h

2.5$ monthly subscription for each service client (sensor)

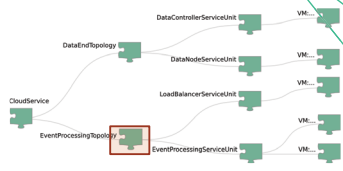
- Determined Elasticity Space Boundaries
  - Clients/h > 148
  - 300ms ≤ ResponseTime ≤ 1100 ms



Elasticity Space "Clients/h" Dimension

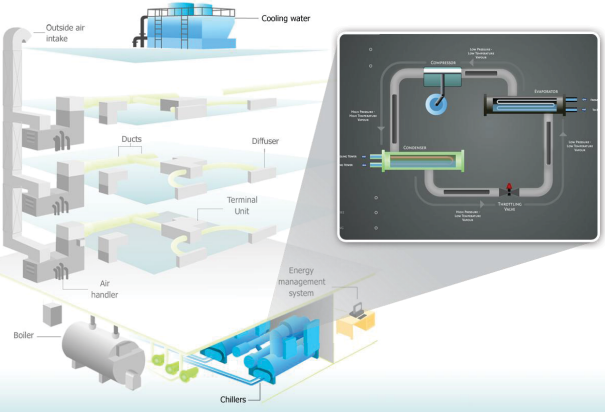


Elasticity Space "Response Time" Dimension

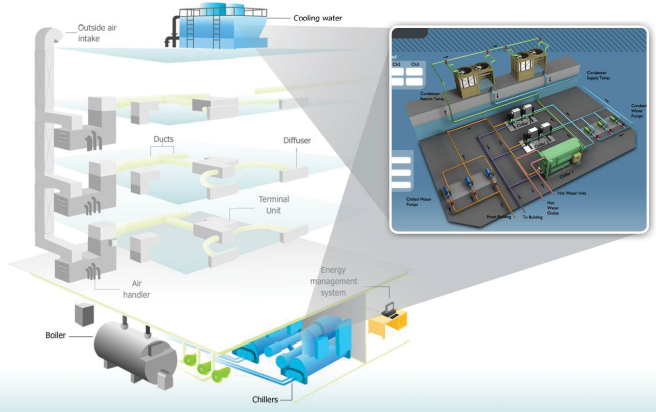


IoT- Engineering Techniques

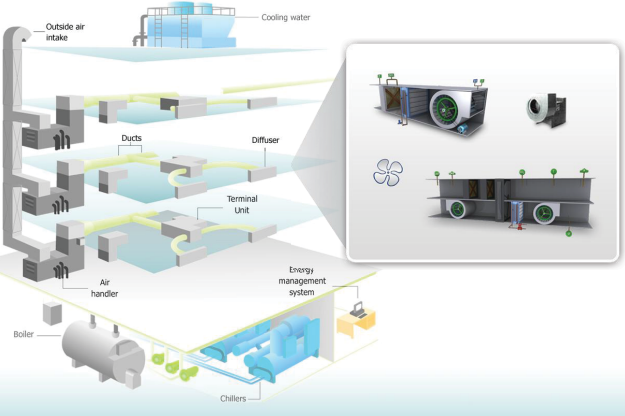
HVAC (Heating, Ventilation, Air Conditioning) Ecosystem



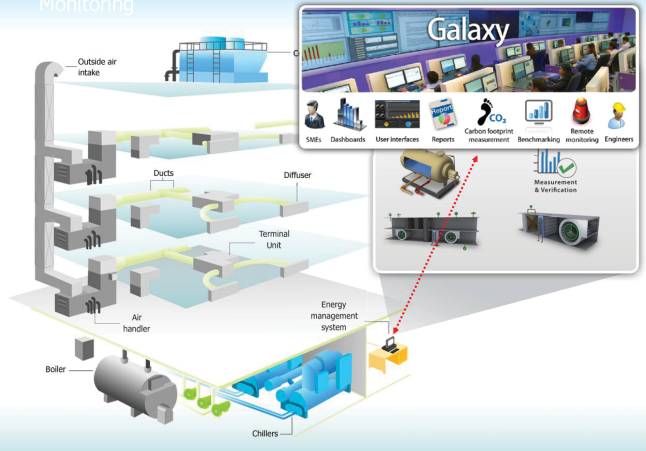
Water Ecosystem

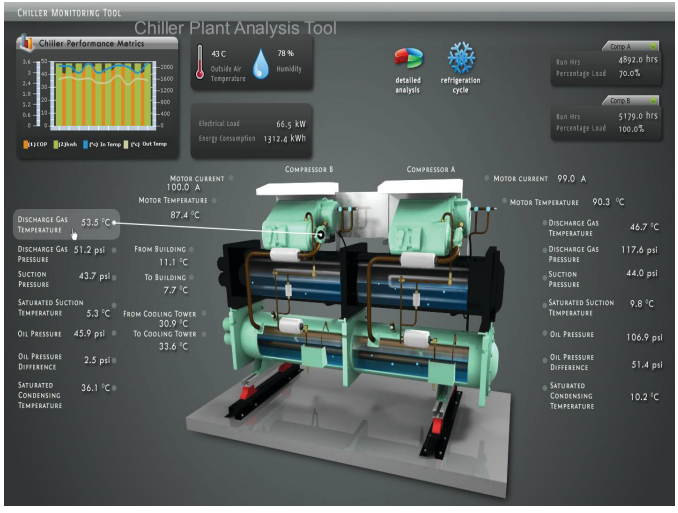


Air Ecosystem



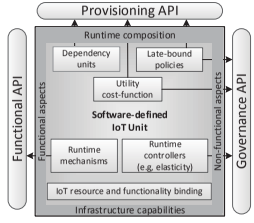
Monitoring





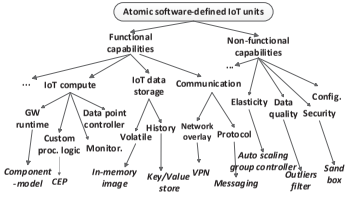
Software-defined IoT units

- Provide software-defined API for accessing, configuring and controlling units
- Support fine-grained internal configurations, e.g, adding functional capabilities like different communication protocols, at runtime.
- Can be composed at higher-level, via dependency units, creating virtual topologies (of multiple gateways) that can be (re)configured at runtime.
- Enable decoupled and managed configuration (via late-bound policies) to provision the units dynamically and on-demand.
- Have utility cost-functions that enable pricing the IoT resources as utilities.

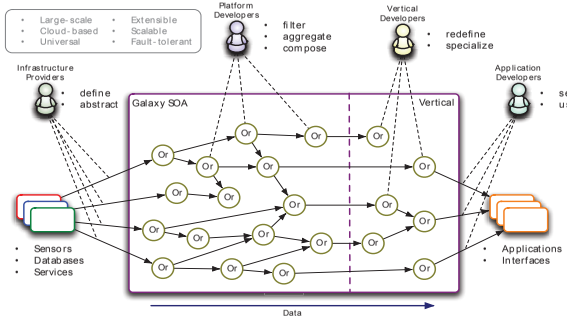


Ecosystem for software-defined IoT systems

- Create an ecosystem of software-defined IoT units for the creation of software-defined IoT systems.
- Distributing IoT units in a market-like fashion, e.g., via IoT AppStore.



The Programming Model (Origins)

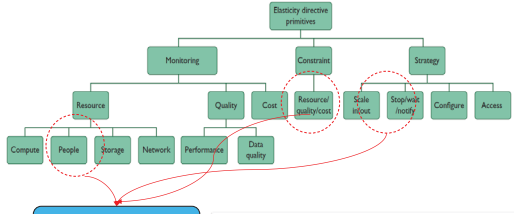


Sehic, S. Li, F., Nastic, S., Dustdar, S.. A Programming Model for Context-Aware Applications in Large-Scale Pervasive Systems, The 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2012), 8-10 October 2012, Barcelona, Spain.

Sehic, S., Nastic, S., Vögler, M., Li, F., Dustdar, S. Entity-Adaptation - A Programming Model for Development of Context-Aware Applications, 29th Symposium On Applied Computing (SAC 2014), Mobile Computing and Applications (MCA) track, 24-28 March 2014, Gyeongju, Republic of Korea

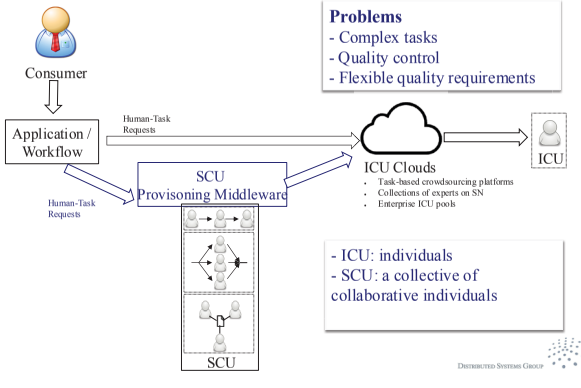
Human augmentation – Engineering Techniques

Specifying and controling elasticity of human-based services




```
#for a service unit analyzing chiller measurement
#SYBL.ServiceUnitLevel
Mon1 MONITORING accuracy = Quality.Accuracy
Cons1 CONSTRAINT accuracy < 0.7
Str1 STRATEGY CASE Violated(Cons1):
Notify(Incident.DEFAULT, ServiceUnitType.HBS)
```

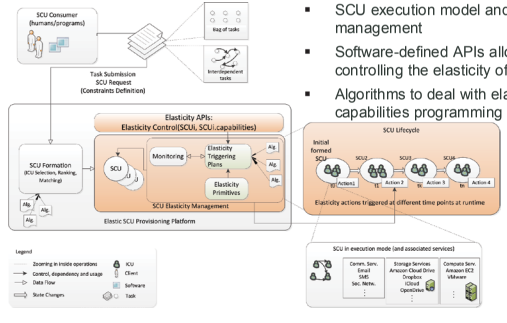
SCU for independent tasks



Problems
- Complex tasks
- Quality control
- Flexible quality requirements

- ICU: individuals
- SCU: a collective of collaborative individuals

Elasticity Capabilities and APIs



- SCU execution model and lifecycle management
- Software-defined APIs allow controlling the elasticity of SCU
- Algorithms to deal with elasticity capabilities programming

Mirela Riveni, Hong-Linh Truong, and Schahram Dustdar, **On the Elasticity of Social Compute Units**, CAISE 2014

Conclusions (1) – Engineering Elasticity

- The evolution of underlying systems and the utilization of different types of resources under different models for elasticity requires
  - Complex, open hybrid service unit provisioning frameworks
  - Different strategies for dealing with different types of tasks
  - Quality issues for software, data, and people in an integrated manner for different perspectives
- We are just at an early stage of developing techniques for engineering elastic applications wrt multi-dimensional elasticity

Conclusions (2) – Engineering Elasticity

Service engineering analytics of elastic systems

- Programming hybrid compute units for elastic processes
- Elasticity specifications and reasoning techniques
- Elasticity spaces analytics

Application domains

- "Social computer" and smart cities (FP 7 FET Smart Cities and PC3L)
- Computational science and engineering (FP 7 CELAR)

Thanks for your attention!

Prof. Dr. Schahram Dustdar

Distributed Systems Group
TU Wien

dsg.tuwien.ac.at

EXPLORING AUTONOMICS FOR CLOUDS

Manish Parashar, Rutgers University

A grand challenge for verification of control systems could be stated as follows: Given requirement Cloud computing has emerged as a dominant paradigm that is being widely adopted by enterprises. Clouds and Cloud federations are also rapidly joining high-performance computing system, clusters and Grids as viable platforms for scientific exploration and discovery. Clouds offer on-demand access to computing utilities, an abstraction of unlimited computing resources, customizable environments, and a pay-as-you-go business model. They also provide a potential for dynamic scale-up, scale-down and scale-out, and support IT outsourcing and automation. As a result, it is possible to create hybrid federated cloud infrastructures integrating private clouds, local data centers, and public clouds. However, developing and managing cloud applications/services to appropriately use the capacities and capabilities offered by these cloud federations can be challenging – for example, applications/services need to be managed according to pricing policy, quality of service requirements, budgets, etc. In this talk, I will explore autonomic application execution and management in federated Cloud infrastructures.

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Exploring Autonomics for Federated Clouds

Moustafa AbdelBaky, Javier Diaz-Montes, Mengsong Zou and **Manish Parashar**

The NSF Cloud and Autonomic Computing Center
Rutgers, The State University of New Jersey, USA
http://cometcloud.org

In collaboration with Omer Rana, Tom Beach, Ioan Petri, Cardiff University, UK

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

Moving towards the Cloud

- Cloud services provide an attractive platform for supporting the computational and data needs of academic and business application workflows
- Cloud paradigm:
  - Rent resources as cloud services on-demand and pay for what you use
  - Potential for scaling-up, scaling-down and scaling-out, as well as for IT outsourcing and automation
- Hybrid cloud services landscape spanning private clouds, public clouds, HEC centers, etc.
  - Heterogeneous offering with different QoS, pricing models, availability, capabilities, and capacities

Cloud Federations – Motivations

- Application workflow exhibit heterogeneous and dynamic workloads, and highly dynamic demands for resources
  - Various and dynamic QoS requirements
    - Throughput, budget, time
  - Often involve large amounts of data
    - Large size, heterogeneous nature, and geographic location
- Such workloads are hard to be efficiently supported using classic federation models
- Implications of the cloud paradigm
  - Rent required resources as cloud services on-demand and pay for what you use
  - Heterogeneous offering with different QoS and costs
- Provisioning and federating an appropriate mix of resources on-the-fly is essential and non-trivial

AUTONOMICS FOR CLOUD FEDERATIONS

RUTGERS

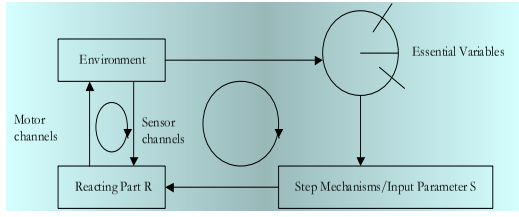
Integrating Biology and Information Technology: The Autonomic Computing Metaphor (~2004)

- Current paradigms, mechanisms, management tools are inadequate to handle the scale, complexity, dynamism and heterogeneity of emerging systems and applications
- Nature has evolved to cope with scale, complexity, heterogeneity, dynamism and unpredictability, lack of guarantees
  - self configuring, self adapting, self optimizing, self healing, self protecting, highly decentralized, heterogeneous architectures that work !!!
- Goal of autonomic computing is to enable self-managing systems/ applications that addresses these challenges using high level guidance
  - Separation of policy and mechanisms; Holistic; Automation

"Autonomic Computing: An Overview," M. Parashar, and S. Hariri, Hot Topics, Lecture Notes in Computer Science, Springer Verlag, Vol. 3566, pp. 247-259, 2005.

RUTGERS

Ashby's Ultrastable System (1920s)



RUTGERS

Integrating Biology and Information Technology: The Autonomic Computing Metaphor (~2004)

- Rich body of work on using autonomics for cloud/data-center management
  - Provisioning
  - Workload management
  - Power/energy management
  - Etc…
  - Using control theoretic approaches

"Autonomic Computing: An Overview," M. Parashar, and S. Hariri, Hot Topics, Lecture Notes in Computer Science, Springer Verlag, Vol. 3566, pp. 247-259, 2005.

RUTGERS

Autonomic Cloud/ACI Federation

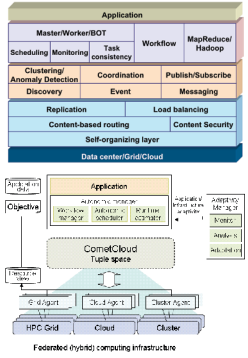
- Assemble a federated cloud/ACI on-the-fly integrating clouds, grids and HPC
  - Cloud-bursting: dynamic application scale-out/up to address dynamic workloads, spikes in demand, and other extreme requirements
  - Cloud-bridging: on-the-fly integration of different resource classes
- Provide policy-driven autonomic resource provisioning, scheduling and runtime adaptations
  - What and where to provision?
  - Policies encapsulate user's requirements (deadline, budget, etc.), resource constraints (failure, network, availability, etc.)
- Provide programming abstractions to support application workflows

RUTGERS

CometCloud – Federated Clouds for Science

- Enable applications on dynamically federated, hybrid infrastructure exposed using Cloud abstractions
  - **Services:** discovery, associative object store, messaging, coordination
  - **Cloud-bursting:** dynamic application scale-out/ up to address dynamic workloads, spikes in demand, and extreme requirements
  - **Cloud-bridging:** on-the-fly integration of different resource classes (public & private clouds, data-centers and HPC Grids)
- High-level programming abstractions & autonomic mechanisms
  - Cross-layer Autonomics: Application layer; Service layer; Infrastructure layer
- Diverse applications
  - Business intelligence, financial analytics, oil reservoir simulations, medical informatics, document management, etc.

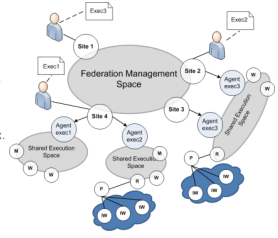
http://cometcloud.org



RUTGERS

On-Demand Elastic Federation using CometCloud

- Software defined ACI federations exposed using elastic on-demand Cloud abstractions
- Autonomic cross-layer federation management using user and provider policies and constraints
  - Separately defined; dynamically evolving
    - Specified based on availability, cost/ performance constraints, etc.
    - Assimilated (or removed) dynamically
    - Sites discover/coordinate with each others to:
      - Identify themselves / Verify identity (x. 509, public/private key,…)
      - Advertise their own resources capabilities, availabilities, constraints
      - Discover available resources

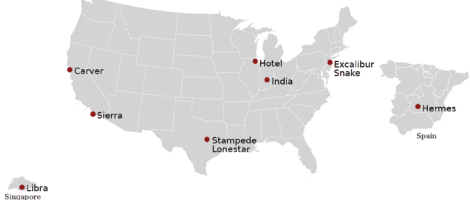


- Federated ACI testbed

RUTGERS

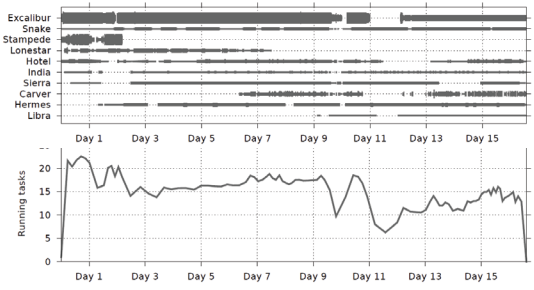
UberCloud Experiment

- 10 different resources from 3 countries federated using CometCloud
- 16 days, 12 hours, 59 minutes and 28 seconds of continuous execution
- 12,845 tasks processed, 2,897,390 CPU-hours consumed, 400 GB of data generated



RUTGERS

Summary of the experiment



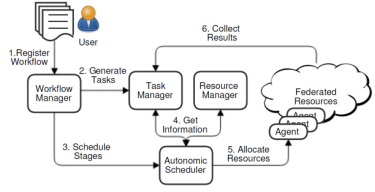
RUTGERS

DATA-DRIVEN WORKFLOWS [CLOUD'14] (WITH IBM)

RUTGERS

Enabling Data-Driven Workflows

- Enable the autonomic execution of complex workflows in software-defined multi-cloud environments
- Elastically compose appropriate cloud services and capabilities to ensure that the user's objectives are met



RUTGERS

Optimizing Resource Usage in Multi-Clouds

- Execute a data-driven workflow in a multi-cloud environment
- Different scheduling policies and objectives
  - Minimum Completion Time
    - Centralized storage vs Distributed storage
  - Deadline-based Policy
    - Performance optimization (Proc)
    - Data locality optimization (Data)
    - Performance and data optimization (ProcData)
    - Cost optimization (Cost)

RUTGERS

Experiment Setup

- Montage workflow
- Three heterogeneous and geographically distributed clouds

VM type†	#Cores	Memory	Max. VMs‡	Speedup
Alamo_Large	4	8 GB	2	3.55
Alamo_Medium	2	4 GB	4	2.77
Alamo_Small	1	2 GB	2	1.68
Sierra_Medium	2	4 GB	2	1
Sierra_Small	1	2 GB	3	0.71
Hotel_Small	1	2 GB	6	0.76

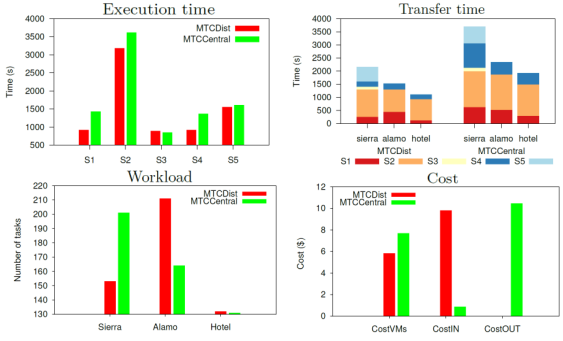
Note: † – Name of the site followed by the type of VM.
‡ – Maximum number of available VMs per type

Network (Down/Up)	Alamo	Sierra	Hotel
Alamo	-	10/0.9	15/15
Sierra	11/11	-	11/11
Hotel	18/18	12/1	-
Internal Network (Down/Up)	11/2.3	30/30	45/45

FutureGrid Resources
- Sierra – SDSC
- Alamo – TACC
- Hotel – U. Chicago

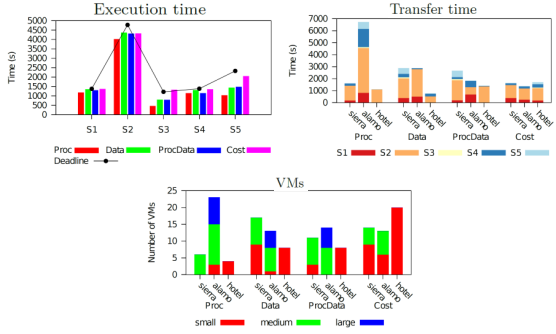
RUTGERS

Minimum Completion Time



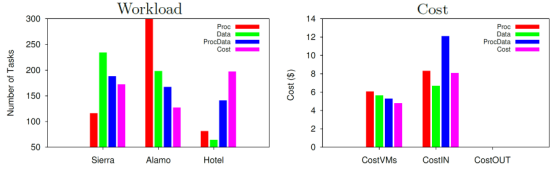
RUTGERS

Deadline-based Policies



RUTGERS

Deadline-based Policies (Cont.)



RUTGERS

FEDERATING RESOURCES USING SOCIAL MODELS [IC2E'14]

RUTGERS

Exchanging Resources in a Federated Cloud

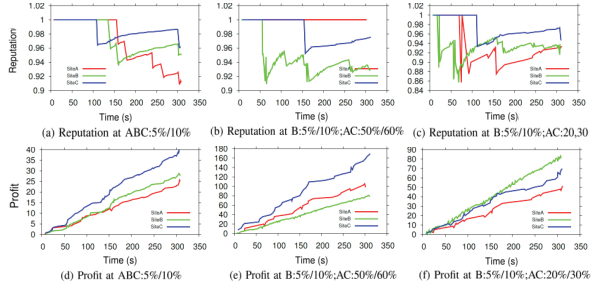
- Consider federation policies and determine their impact on the overall status of each site
- Market model for resource sharing
  - External task vs Local task
  - Heterogeneous tasks - different deadlines and costs
  - Each site decides how much benefit per task (% cost)
  - Federation policy = Auction criteria
- Federation infrastructure between Cardiff (UK) and Rutgers (USA)

RUTGERS

Profit and Reputation of Each Site

Task/Metric	Cost	TTC	Deadline
Red	10	9	12
Black	8	7	10
Blue	6	5	8

- Auction Criteria based on Price



(a) Reputation at ABC:5%/10% (b) Reputation at B:5%/10%;AC:50%/60% (c) Reputation at B:5%/10%;AC:20,30

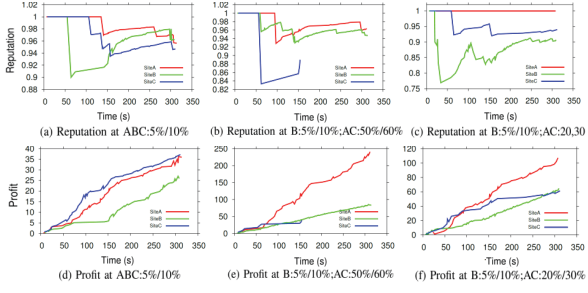
(d) Profit at ABC:5%/10% (e) Profit at B:5%/10%;AC:50%/60% (f) Profit at B:5%/10%;AC:20%/30%

RUTGERS

Profit and Reputation of Each Site II

Task/Metric	Cost	TTC	Deadline
Red	10	9	12
Black	8	7	10
Blue	6	5	8

- Auction Criteria based on Price and Reputation



(a) Reputation at ABC:5%/10% (b) Reputation at B:5%/10%;AC:50%/60% (c) Reputation at B:5%/10%;AC:20,30

(d) Profit at ABC:5%/10% (e) Profit at B:5%/10%;AC:50%/60% (f) Profit at B:5%/10%;AC:20%/30%

RUTGERS

HPC PLUS CLOUD FEDERATIONS [E-SCIENCE'10]

RUTGERS

Exploring Hybrid HPC-Grid/Cloud Usage Modes (eScience'09, ScienceCloud'10)

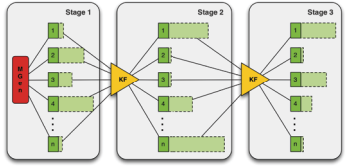
What are appropriate usage modes for hybrid infrastructure?

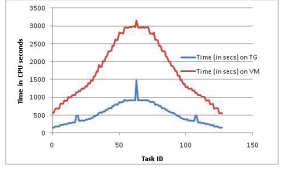
- Acceleration -- How can Clouds be used as accelerators to improve the application time to completion
  - To alleviate the impact of queue wait times
  - "Strategically Off load" appropriate tasks to Cloud resources
  - All while respecting budget constraints.
- Conservation – How Clouds can be used to conserve HPC Grid allocations, given appropriate runtime and budget constraints.
- Resilience – How Clouds can be used to handle:
  - General: Response to dynamic execution environments
  - Specific: Unanticipated HPC Grid downtime, inadequate allocations or unexpected Queue delays/QoS change

RUTGERS

Reservoir Characterization: EnKF-based History Matching

- Black Oil Reservoir Simulator
  - simulates the movement of oil and gas in subsurface formations
- Ensemble Kalman Filter
  - computes the Kalman gain matrix and updates the model parameters of the ensembles
- Heterogeneous workload, dynamic workflow
- Based on Cactus, PETSc





RUTGERS

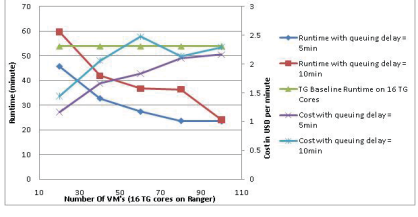
Using Clouds as Accelerators for HPC Grids

- Explore how Clouds (EC2) can be used as accelerators for HPC Grid (TG) workloads
  - 16 CPUs (Ranger)
  - Average queuing time for Ranger was set to 5 and 10 minutes
  - Number of EC2 VMs (m1.small) from 20 to 100 in steps of 20
  - VM start up time was about 160 seconds

RUTGERS

Using Clouds as Accelerators for HPC Grids I

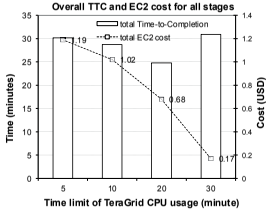
- Acceleration is more notable with more VMs - lower the TTC
- The reduction in TTC is roughly linear
  - Affected by complex interplay between the tasks in the workload and resource availability



RUTGERS

Exploring Conservation

- Application deadline 33 minutes (time using only TeraGrid)
- What if we have limited resources on TeraGrid? But we need to keep the same deadline
- Use Cloud to save HPC resources

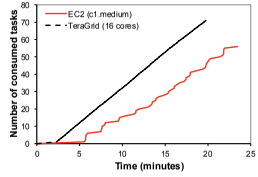


CPU usage limit (min)	5	10	20	30
Num of scheduled VMs (EC2)	7	6	4	1
Num of expected tasks consumed by EC2	111	92	54	14
Consumed tasks by EC2	109	89	49	16

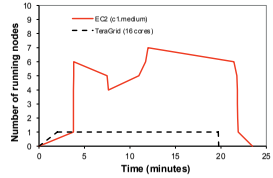
RUTGERS

Exploring Resilience

- Deadline 20 minutes
- Two EC2 instances are failed at around 8 minutes



(a) Number of consumed tasks



(b) Number of nodes

RUTGERS

Conclusions

- Complex application workflows necessiciate software defined federated platforms that integrated heterogeneous cloud services
- Provisioning and federating an appropriate mix of resources on-the-fly is essential and non-trivial
- Autonomics can provide the abstractions and mechanism to manage complexity
  - Separation + Integration + Automation
- However, there are implications
  - Added uncertainty
  - Correctness, predictability, repeatability
  - Validation
  - New formulations necessary….

RUTGERS

The CometCloud Team

Moustafa AbdelBaky
PhD student, Dept. of Electrical & Computer Engr.
Rutgers University
Email: moustafa@cac.rutgers.edu

Mengsong Zou
PhD Student, Dept. of Computer Science
Rutgers University
Email: mz228@cac.rutgers.edu

Javier Diaz-Montes, Ph.D.
Assistant Research Professor, Dept. of Electrical & Computer Engr.
Rutgers University
Email: javidiaz@rdi2.rutgers.edu

Manish Parashar, Ph.D.
Prof., Dept. of Electrical & Computer Engr.
Director, Center for Autonomic Computing
Rutgers University
Email: parashar@rutgers.edu

And many collaborators….

CometCloud: http://cometcloud.org

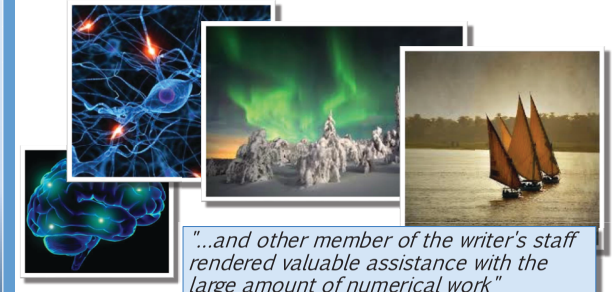
THINKING PARALLEL: MULTI-CORES, VIRTUAL ELASTICITY, AND THE APPLICATION PROGRAMMER

Geir Horn, University of Oslo, Norway

CPUs can offer a large number of cores and dedicated accelerators. Large data centres offers large number of virtual machines in the Cloud. At the exception of some highly optimised eScience applications, these resources are used to support high throughput computing of data parallel applications. Mixing true parallelism with the cloud is inherently complicated, and the application programmer cannot think parallel. This talk will discuss the issue and present some possible ways forward, hopefully stimulating an interesting discussion on future research directions.

Thinking parallel: Multi-cores, virtual elasticity, and the application programmer

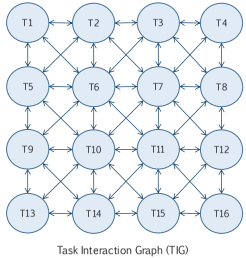
Geir Horn



"...and other member of the writer's staff rendered valuable assistance with the large amount of numerical work"
- H. E. Hurst[2]

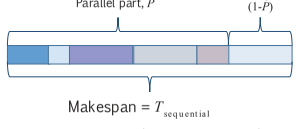
[1] Suzana Herculano-Houzel, 'The human brain in numbers: a linearly scaled-up primate brain', *Front. Hum. Neurosci.*, vol. 3, p. Article 31, Nov. 2009.
[2] Harold Edwin Hurst, 'Long-term storage capacity of reservoirs', *Transactions of the American Society of Civil Engineers*, pp. 770–799, 1951.

The basics



Task Interaction Graph (TIG)

Sequential scheduling (one processor)



Makespan = $T_{sequential}$

Parallel scheduling (three processors)

Makespan = $T_{parallel} + T_{serial}$

Speedup = $T_{sequential} \Big/ \left(T_{parallel} + T_{serial}\right)$

Speedup versus Scalability

- Ideal speedup = number of processors = $|N|$
- Amdahl's law[3]:

  Speedup = $\dfrac{1}{(1-P) + \frac{P}{|N|}}$
- Max speedup as $|N| \to \infty$ is

  $\dfrac{1}{1-P}$
- Example: With $P = 90\%$ the max speedup is 10
- Alternatively: Keep run time fixed, but increase problem size with the number of processors
- Hypothetical sequential run time

  $T_{sequential} = T_{serial} + |N| \times T_{parallel}$
- Gustafson-Barsis' law[4]: the scaled speedup is

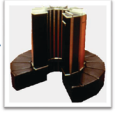
  $$SS = T_{sequential} \Big/ \left(T_{parallel} + T_{serial}\right)$$
  $$= \left(T_{serial} + |N| \times T_{parallel}\right) \Big/ \left(T_{parallel} + T_{serial}\right)$$
  $$= |N| - \alpha\left(|N| - 1\right)$$

  with $\alpha = T_{serial} \Big/ \left(T_{parallel} + T_{serial}\right)$

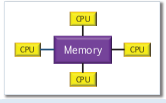
[3] Gene M. Amdahl, 'Validity of the single processor approach to achieving large scale computing capabilities', in *Proceedings of the AFIPS spring joint computer conference*, Conference location: Atlantic City, New Jersey, USA, 1967, pp. 483–485.
[4] John L. Gustafson, 'Reevaluating Amdahl's law', *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

Parallel *digital* computers

Burroughs' D825
4 processors
(1962)



Cray-1 (1976)
Cray X-MP (1980)

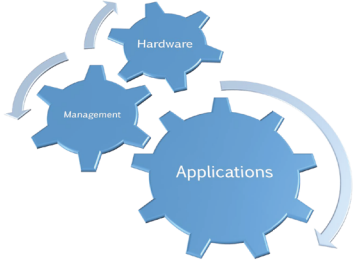


Shared memory

Distributed computing
- Massive Parallel Processing = Dedicated interconnect
- Cluster Computing = Ethernet + TCP/IP
- GRID computing = Internet
- Cloud = everything virtualised

Thinking parallel: Multi-cores, virtual elasticity, and the application programmer

Parallel computing†



Hardware
- The execution platform

Management software
- Operating System (OS)
- Platform control
- Management middleware
- Development support

Applications
- The essential value

† High Performance Computing, Many Task Computing, maybe High Throughput Computing, not Dedicated Computing

Thinking parallel: Multi-cores, virtual elasticity, and the application programmer

More beyond Moore

Gordon E. Moore's law[5]:
"...the number of transistors on a chip will double every 18 months"

Mostly process scaling with 0.7 every 24 months from 0.8 µm in 1992 until ~2000

- Physical distance problems with scaling
- Voltage supply scaling and increased leaks
- Clock frequency: maximum around 5GHz
- Power density wall and heat
- Design complexity: exponential design team growth

[5] Gordon E. Moore, 'Cramming More Components Onto Integrated Circuits', *Electronics*, pp. 114-117, Apr. 1965.
[6] Jeff Parkhurst, John Darringer, and Bill Grundmann, 'From single core to multi-core: preparing for a new exponential', in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design (ICCAD '06)*. Conference location: San Jose, California, USA, 2006, pp. 67-72.

Thinking parallel: Multi-cores, virtual elasticity, and the application programmer

...on heterogeneous cores

Intel & AMD x86 dual-cores (2005)

"Assuming that this trend will follow Moore's Law scaling, mainstream systems will contain over 10 processing cores by the end of the decade, yielding unprecedented theoretical peak performance"

Justin Rattner
Vice president and chief technology officer, Intel
(2005)[10]

simultaneous multithreading (2004)[8]

SUN UltraSparc T2 8 cores, 64 threads (2007)[9]

[7] J. M. Tendler *et al*, 'POWER4 system microarchitecture', *IBM Journal of Research and Development*, vol. 46, no. 1, pp. 5-25, Jan. 2002.
[8] B. Sinharoy *et al*, 'POWER5 system microarchitecture', *IBM Journal of Research and Development*, vol. 49, no. 4-5, pp. 505-521, Jul. 2005
[9] Umesh Gajanan Nawathe *et al*, 'An 8-Core 64-Thread 64b Power-Efficient SPARC SoC', in *Digest of Technical Papers from the IEEE International Solid-State Circuits Conference (ISSCC 2007)*. Conference location: San Francisco, CA, USA, 2007, pp. 5-7; 108-110.
[10] Justin Rattner, 'Multi-core to the masses', in *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT 2005)*. Conference location: Saint Louis, Missouri, USA, 2005, p. 3.

Hardware: The future

- Dedicated *accelerators*† or proven FPGA templates or design patterns
- Proven library of modules with predictable performance[15]
  - Parameterised interconnects
  - Standard adapters and interfaces
  - Reusable building blocks
- Asymmetric speedup ≥ symmetric speedup[16]
- "*Software will take a more prominent role in the multi-core era*"[6]
- "*...the programming model for heterogeneous architectures is much more complicated*"[17]

† "*Technology based on "manycore" will employ 100s to 1000s of CPU cores per chip (by2011)*"[14]

[14] John Shalf, 'The new landscape of parallel computer architecture', *Journal of Physics: Conference Series*, vol. 78, p. 012066, Jul. 2007.
[15] John A Darringer, 'Multi-core design automation challenges', in *Proceedings of the 44th annual Design Automation Conference (DAC '07)*, San Diego, California, USA, 2007, pp. 760–764.
[16] Mark D. Hill and Michael R. Marty, 'Amdahl's Law in the Multicore Era', *Computer*, vol. 41, no. 7, pp. 33–38, Jul. 2008.
[17] Geoffrey Blake *et al.* 'A survey of multicore processors', *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 26–37, Nov. 2009.

Operating Systems

- "No current OS is really multithreaded"[a]
- New approaches to operating systems
  - XtreemOS[b] =  + 
  - S(o)SS = Service Oriented Operating Systems[18]
  - Tessllation[19]
  - FUSE: OS support for easy HW accelerator integration[20]
- Native hypervisors + microkernels
- Real-time OS[d]

a Lutz Schubert, Universität Ulm
b http://www.xtreemos.eu/ and http://research.cs.wisc.edu/condor/
c http://www.soos-project.eu/
d QNX Neutrino RTOS (http://www.qnx.com/products/neutrino-rtos/neutrino-rtos.html) or INTEGRITY (http://www.ghs.com/products/rtos/integrity.html)

[18] Lutz Schubert *et al.* 'Service-oriented operating systems: future workspaces', *IEEE Wireless Communications*, vol. 16, no. 3, pp. 42–50, Jun. 2009.
[19] Krste Asanovic *et al.* 'A view of the parallel computing landscape', *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, Oct. 2009.
[20] Aws Ismail and Lesley Shannon, 'FUSE: Front-End User Framework for O/S Abstraction of Hardware Accelerators', in *Proceedings of the 19th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM 2011)*, Salt Lake City, Utha, USA, 2011, pp. 170–177.

High Throughput Mass Market

- Desktop
  - Legacy software "does the job"
  - Current software developed for single-core
  - Per-application performance is important (only) if the load consists of only a few applications or if there are performance-critical applications[33]
- Servers
  - Database and web servers are designed for high throughput
  - Idle time can be masked by multi-threading[33]
- Large parallel application part = more cores, otherwise more complex cores[16]

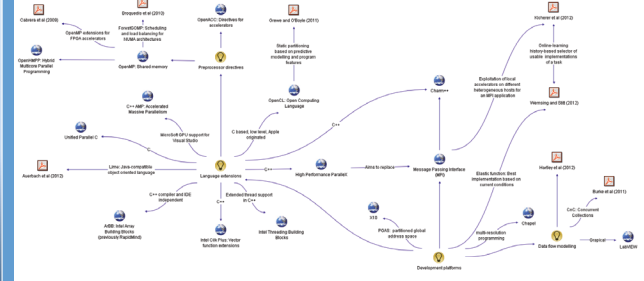
[30] Jacques A. Pienaar *et al.* 'MDR: performance model driven runtime for heterogeneous parallel platforms', in *Proceedings of the international conference on Supercomputing (ICS '11)*, Tucson, Arizona, USA, 2011, pp. 225–234.
[31] Anders Logg *et al.* 'Past and Future Perspectives on Scientific Software', in *Simula Research Laboratory*, vol. Part II, Aslak Tveito *et al.* (Eds.) Springer, 2010, pp. 321–362.
[32] Michela Becchi *et al.* 'Enabling Legacy Applications on Heterogeneous Platforms', in *Proceedings of the 2nd USENIX Workshop on Hot Topics in Parallelism (HotPar '2010)*, Berkeley, CA, USA, 2010.
[33] Angela C. Sodan *et al.* 'Parallelism via Multithreaded and Multicore CPUs', *Computer*, vol. 43, no. 3, pp. 24–32, Mar. 2010.

The future

- Pin-limits: no technology in sight![33]
- Three scenarios[35]
  - "Drop the ball" ⇒ Cloud computing
  - "Niche markets" ⇒ Multimedia, gaming...
  - "Scalable, dependable, software development"[13]
- Urgently needed: *"Parallel computing for all"* ™
  - Standardised, industry accepted development platforms
  - Education of programmers on these platforms
  - Requirements engineering for parallelism

[35] David Patterson, 'The trouble with multi-core', *IEEE Spectrum*, vol. 47, no. 7, pp. 28–32, 53–54, Jul. 2010.

Development support



http://www.open-mpi.org/
http://openmp.org
http://www.openhmpp.org
http://www.openacc-standard.org/
http://www.khronos.org/opencl/
http://upc.lbl.gov/
http://threadingbuildingblocks.org/
http://software.intel.com/en-us/articles/intel-cilk-plus/
http://software.intel.com/en-us/articles/intel-array-building-blocks/
http://msdn.microsoft.com/en-us/library/hh265137%28v=vs.110%29
http://charm.cs.uiuc.edu/research/charm/
http://stellar.cct.lsu.edu/tag/parallex/
http://www.ni.com/labview/
http://chapel.cray.com/
http://x10-lang.org/

Management software: The future

- Operating systems
  - Thin generic access layers
  - Automatic adaptive schedulers learning the characteristics of the execution platform and the application
- Optimised computation kernels
  - BLAS, LINPACK, Diffpack, FFT, Boost,...
  - Learning based implementation selectors[26] and elastic computing[27]
- Development support
  - Methodology: Thinking parallel, patterns[28]
  - Languages
    - Enhanced or new languages, e.g. Lime[29]
    - Model Driven Development and Runtime[30]
  - Integrated Development Environments
  - Cross – and just-in-time compilers[31]
- Tools to assist conversion of legacy software[32]

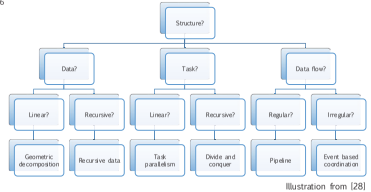


Illustration from [28]

Paradigm: The ACTOR model[36]

- Mathematical model of concurrent computation
  - Dynamic creation of actors
  - Asynchronous, unordered message passing
  - Concurrent computation
  - Dynamic topology – addresses in messages
- Service oriented
- No threads – no shared memory
- Legacy compliant

Virtual elasticity

- Migrating actors
- Location: Data ↔ Algorithms
- Scheduling of actors
  - Virtual threads
  - Bin packing
  - Communication aware
- Integrated interconnect

In a soldier's stance, I aimed my hand
At the mongrel dogs who teach
Fearing not that I'd become my enemy
In the instant that I preach
My pathway led by confusion boats
Mutiny from stern to bow
Ah, but I was so much older then
I'm younger than that now

Yes, my guard stood hard when abstract threats
Too noble to neglect
Deceived me into thinking
I had something to protect
Good and bad, I define these terms
Quite clear, no doubt, somehow
Ah, but I was so much older then
I'm younger than that now

Bob Dylan
My back pages

Geir Horn
Geir.Horn@mn.uio.no
+47 93 05 93 35

SIMPLIFIED CLOUD CONTROL USING DIMENSION REDUCTION

Jianguo Yao, Shanghai Jiao Tong University

Automated management of complex information technology applications such as cloud systems requires dynamic configuration of both application-level and system-level parameters. The existence of large number of tunable parameters makes it difficult to design a feedback controller that adjusts these parameters effectively in order to achieve the application-level performance targets. In this talk, we will summarize our recent work which introduces a new approach for simplified control architecture of large-scale complex systems based on dimension reduction techniques. It combines the online selection of critical control knobs through LASSO -- a powerful L1 -constrained fitting method, and Compressive Sensing (CS)-- a L1-optimization method, and the design of adaptive control of the identified knobs. We use evaluation results to demonstrate the effectiveness of this new approach.

Simplified Cloud Control Using Dimension Reduction

Jianguo Yao (SJTU), Xue Liu(McGill University),
Joint work with Xiaoyun Zhu (VMware)

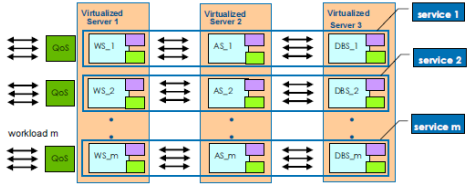
Outline

- Motivation
- Problems and Challenges
- Solution architecture
- Dimension Reduction Methods
  - LASSO/LARS
  - Compressive Sensing
- Three-module controller design
  - Dimension reduction
  - RLS-based model identification
  - Linear quadratic optimal controller
- Evaluation results
  - Simulation
  - Experiment
- Conclusion and future work

Motivation

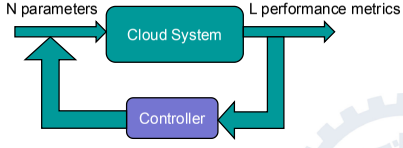
Increasingly complex cloud systems

- Cloud system and applications are getting more complex
- Large numbers of system-level and application-level parameters
  - System-level: no. of VMs, resource allocation (CPU, GPU, memory, network, I/O), workload/data placement, cache size, processor frequency
  - Application-level: no. of processes/threads, no. of database connections, time-out, keep-alive



Problems

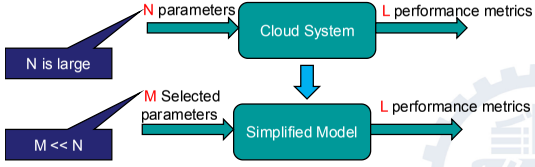
- Problem
  - How to adjust these parameters to achieve application-level SLOs?
  - Manual tuning is hard -> automation via feedback control



Cloud Control needs a model for cloud system to design the controller.

Challenges

- Challenges
  - Too many tunable parameters (inputs) -> which ones are the most critical?
  - Set of critical parameters may vary over time
  - A controller that tunes all the parameters is not efficient due to the high dimension



How to automatically build a simplified model with low dimension.

A motivating example A realistic case

- To adjust response time in Apache, we can insert a function of Sleep call.
- However, there are many functions in the Apache in which only a few can dramatically effect the response time.

Coefficients — More than 100

$$r(k+1) = \sum_{i=1}^{m} \lambda_i \, s_i(k) + e_1(k)$$

Response time — Sleep time — Noise

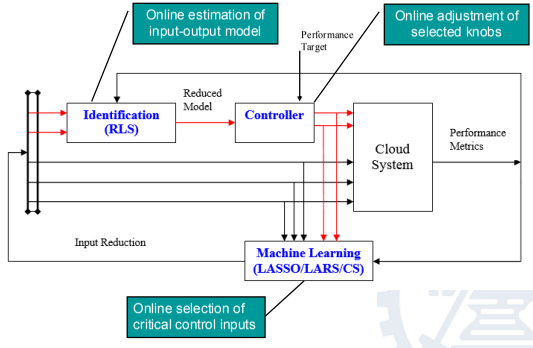
TABLE 1
Coefficients for functions.

Functions	Coefficient
apr_socket_recv	$\lambda_1 = 33.6382$
apr_protocol_get	$\lambda_2 = -0.1023$
apr_pool_get	$\lambda_3 = -0.0011$
apr_inherit_set	$\lambda_4 = -0.0698$
apr_inherit_unset	$\lambda_5 = 0.0199$
apr_sockaddr_equal	$\lambda_6 = 0.1180$
apr_mcast_join	$\lambda_7 = -0.0323$
apr_mcast_leave	$\lambda_8 = 0.0164$
apr_mcast_interface	$\lambda_9 = -0.0806$
apr_ipsubnet_test	$\lambda_{10} = -0.0202$

- Simplified model:

$$r(k+1) = \lambda_1 s_1(k) + e(k) \qquad \text{where } e(k) = \sum_{i=2}^{m} \lambda_i \, s_i(k) + e_1(k)$$

Simplified cloud control architecture



Dimension reduction using Lasso

- Lasso – Shrinkage and selection method for linear regression
  - R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *Journal of Royal Statistics Society,* 1996.
    http://statweb.stanford.edu/~tibs/lasso.html

System inputs: $u_1, u_2, \ldots, u_m$, zero mean and unit length

System output: y, zero mean

Find model: $\hat{y} = \sum_{j=1}^{m} \hat{x} u_j$ (coefficient)

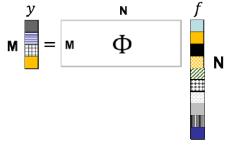
where $\min \; \| y - \hat{y} \|_2$

s.t. $\|\hat{x}\|_1 = \sum_{j=1}^{m} |\hat{x}_j| < t$ (Tuning parameter)

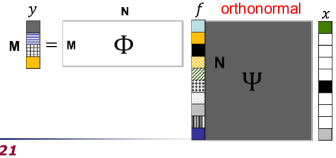
Dimension reduction using CS

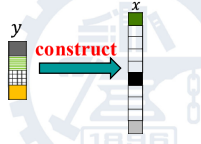
- Measurement data

$$y = \Phi f,\ M < N$$



- Applying Compressive Sensing (CS) when : signal x is compressible, sparse…

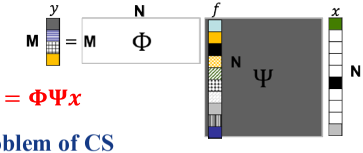




Recovery

- Recovery

$$y = \Phi f,\ f = \Psi x$$



- Solve for x, s.t. $y = \Phi\Psi x$
- Optimization problem of CS

M << N

$$\min_x \|x\|_1$$

$$\text{s.t. } \|y - \Phi\Psi x\|_2 \le \varepsilon$$

Tuning parameter

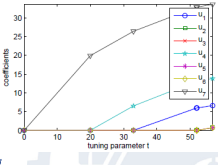
- Our problem

$$\hat{y} = \sum_{j=1}^{m} \hat{x}_j u_j$$

sparse

An Example Estimation of coefficients

- Least Angle Regression (LARS) method allows selection of s < m inputs to predict the output
  - A variant of LASSO, a simpler method for computation
  - B. Efron, I. Johnstone, T. Hastie, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, 2004.
- Input-output model
  - $y(k+1) = 1.0\ u_1(k) + 0.1\ u_2(k) + 0.1\ u_3(k) + 2.0\ u_4(k) + 0.1\ u_5(k) + 0.1\ u_6(k) + 5.0\ u_7(k) + e(k)$



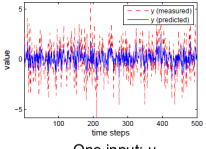
- Estimation using Compressive Sensing

$$\hat{x} = [0.9711\ \ 0.0006\ \ -0.0013\ \ 2.0068\ \ 0.0250\ \ 0.0007\ \ 4.9945\ ]^T$$

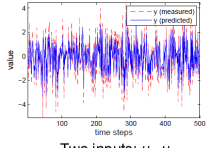
An Example Estimation of coefficients

- Selected input subsets v.s. n and their prediction performance

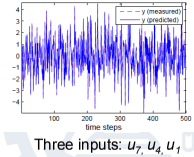
s	Selected inputs	Mean squared error	r^2 measure
1	u_7	360.01	0.73
2	u_7, u_4	131.76	0.90
3	u_7, u_4, u_1	2.09	0.998



One input: u_7



Two inputs: u_7, u_4



Three inputs: u_7, u_4, u_1

Online model estimation

- **Input-output model using selected inputs**

u_s : vector of s selected inputs

$$y(k+1) = X\,\phi(k) + e(k) \qquad \phi(k) = u_s(k).$$

$$\varepsilon(k+1) = y(k+1) - \hat{X}(k)\phi(k),$$

$$\hat{X}(k+1) = \hat{X}(k) + \frac{\varepsilon(k+1)\phi^T(k)P(k-1)}{\lambda + \phi^T(k)P(k-1)\phi(k)},$$

$$P^{-1}(k) = P^{-1}(k-1) + \left(1 + (\lambda - 1)\frac{\phi^T(k)P(k-1)\phi(k)}{[\phi^T(k)\phi(k)]^2}\right)\phi(k)\phi^T(k),$$

- **Online adaptation using recursive least squares (RLS) with exponential forgetting λ**
  - Estimation of X updated in every interval k

Linear quadratic optimal controller

- **Based on the estimated model with reduced dimension**

$$y(k+1) = X\,\phi(k) + e(k)$$

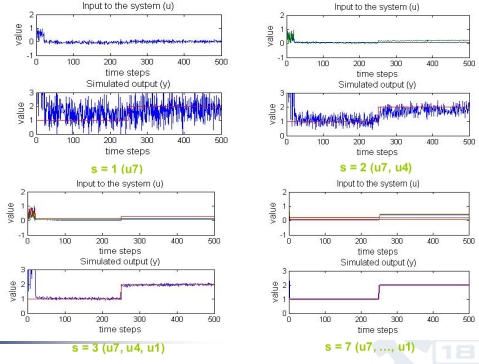
- **Minimizing quadratic cost function**

$$J = \| W(y(k+1) - y_{ref}(k+1)) \|^2 + (\| Q(u_s(k) - u_s(k-1)) \|^2)$$

- **Optimal solution:**

$$u_s^*(k) = ((W\hat{X}(k))^T W\hat{X}(k) + Q^T Q)^{-1}[(W\hat{X}(k))^T W\, y_{ref}(k+1) + Q^T Q u_s(k-1)]$$

Simulation Varying number of selected inputs



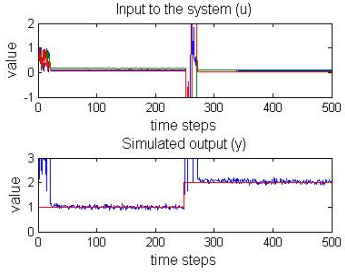
Simulation varying the system behavior

- **At interval k = 250, system model changes to**

$$y(k+1) = 0.1\,u_1(k) + 3.1\,u_2(k) + 14.1\,u_3(k) + 0.1\,u_4(k) + 2.1\,u_5(k) + 0.1\,u_6(k) + 0.1\,u_7(k) + e(k)$$

- **Controller detects the degradation of performance, and starts to collect 20 new samples of input-output data**
- **At interval k = 270, LARS selects a new set of inputs $\{u_2, u_3, u_5\}$**

Simulation varying the system behavior



Experiment A realistic case

- A testbed with iCore7-2600k 3.4GHz CPU and 16GB RAM. Each hosted VM is assigned dual cores and 2GB RAM. We choose Windows7 x64 operating system for both host OS and guest OSes.
- Simplified Model: $r(k+1) = \mathbf{\lambda}_1 s_1(k) + e(k)$
- We vary the tuning parameter "t" from 33 to 34.2
- Estimates of regression coefficients using LRAS/LASSO for the realistic case

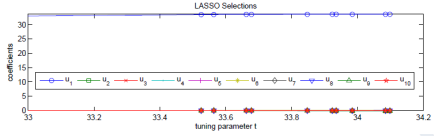
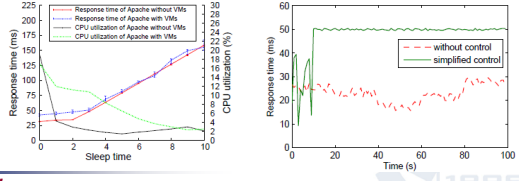


TABLE 1
Coefficients for functions.

Functions	Coefficient
apr_socket_recv	$\lambda_1 = 33.6382$
apr_protocol_get	$\lambda_2 = -0.1023$
apr_pool_get	$\lambda_3 = -0.0011$
apr_inherit_set	$\lambda_4 = -0.0698$
apr_inherit_unset	$\lambda_5 = 0.0199$
apr_sockaddr_equal	$\lambda_6 = 0.1180$
apr_mcast_join	$\lambda_7 = -0.0323$
apr_mcast_leave	$\lambda_8 = 0.0164$
apr_mcast_interface	$\lambda_9 = -0.0806$
apr_ipsubnet_test	$\lambda_{10} = -0.0202$

Experiment A realistic case

- Estimated model is piece-wise linear.
- Set 50.0 ms as the reference.
- The average response time of Apache has been controlled at 49.943 ms.
- Response time of Apache with/without reduced dimension simplified control.



Related work

- Prior work on applying control theory to computing systems
  - Control knobs determined in advance
- Xiong *et al.* ICDCS'11, ICPE'2013
  - Automated model-driven framework for application performance diagnosis in consolidated Cloud Environments
  - Robust provisioning of N-Tier cloud workloads: a multi-level control approach
- Diao *et al.* IM'03
  - Online discovery of critical metrics for a database system, used to construct a quantitative model
  - No online adaptation of the model or the metrics
- Classical model reduction in control theory
  - Reduces the dimensionality of the state space

Conclusions and future work

- **First framework for combined online selection of control knobs and dynamic tuning of the knobs**
- **Three-module controller achieves three design objectives**
  - Online selects a subset of control knobs that have the most significant impact on the controlled output
  - Dynamically tunes the selected control knobs effectively to maintain the system output at the desired value
  - Automatically detects the change in the most critical knobs and uses the new knobs to regulate the system output accordingly
- **Next step**
  - Apply the framework to a real problem in large-scale cloud systems management
    - Dynamic allocation of multiple resources
    - Application configuration management

Reference

- Jianguo Yao, Xue Liu, Xiaoyun Zhu and Haibing Guan, “Control of Large-Scale Systems Through Dimension Reduction”, IEEE Transactions on Service Computing, 2014. (Preprint)
- Jianguo Yao, Xue Liu, Xiaoyun Zhu, "Reduced Dimension Control Based on Online Recursive Principal Component Analysis", in Proceedings of the 2009 American Control Conference (ACC'2009), St. Louis, MO, USA, 2009.
- T. Abdelzher, J.A. Stankovic, C. Lu, R. Zhang, and Y. Lu, ”Feedback performance control is software services,” IEEE Control System Magazine, vol. 23, no. 3, pp.74-90, 2003.
- R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," Journal of Royal Statistics Society, 1996.
- R.G.Baraniuk, ”Compressive Sensing [Lecture Notes],” IEEE Signal Processing Magazine, vol.24, no.4, pp.118-121, 2007.

Questions?

Thanks!

Backup

Architecture

- Goal: Ensure FPS_{VM} = Target
- Insert sleep function $T_{sleep} = \frac{1000}{F} - T_{cpu} - T_{gpu}$

```
While(1)
{
        DrawShapes(&VGA_Buffer);
        SwapBuffer(); // Tell GPU to
display the buffered content.
}
```

Challenge: Uncertainties from execution time of CPU and GPU

Solution: Using feedback control to address the uncertainties

REAL-TIME PERFORMANCE CONTROL OF ELASTIC VIRTUALIZED NETWORK FUNCTIONS

Tommaso Cucinotta, Bell Labs, Alcatel-Lucent, Ireland

Controlling end-to-end service quality, and particularly real-time performance and reliability, of time-sensitive applications in cloud environments is overly challenging. The problem is even harder when trying to switch from the earlier world of network functions shipped as physical boxes, to the emerging paradigm of virtualized cloud-ready elastic network functions, where still the "5 9s" and tight sub-second real-time performance requirements as coming from precise SLAs have to be met. This talk provides an overview of past and ongoing research carried out at Bell Labs on these challenging issues.

Real-time Performance Control of Elastic Virtualized Network Functions

Tommaso Cucinotta
Bell Laboratories, Alcatel-Lucent
Dublin, Ireland

Introduction

Introduction

A new era of computing for ICT

- Wide availability of broadband connections
  ==> shift in computing paradigms towards distributed computing (**cloud computing**)
- More and more resources provided remotely
  - Not only *remote storage* and *batch processing*
  - But also *remote processing* for *interactive applications*
- Network operators are shifting provisioning of critical network services to virtualized network functions (through **private or hybrid cloud** provisioning models)

Examples

- **Virtual Reality** with heavyweight physics simulations
- Distributed editing of HD video (**film post-production**)

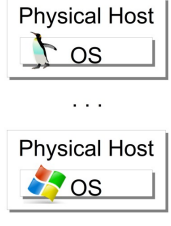
AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

Introduction

Virtualization technologies are key

- For **IaaS** providers (Cloud Computing)
- For **server consolidation**

Different virtualization technologies



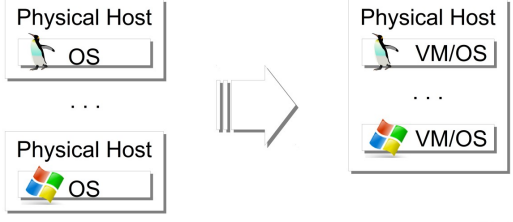
AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

Introduction

Virtualization technologies are key

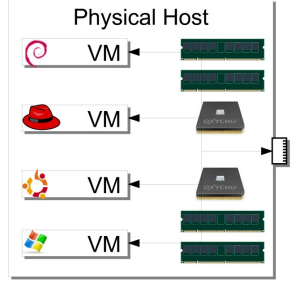
- For **IaaS** providers (Cloud Computing)
- For **server consolidation**

Different virtualization technologies



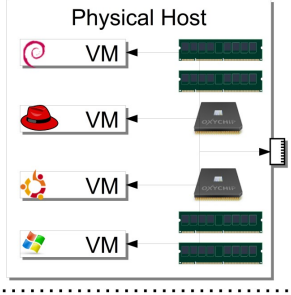
Need for Performance Isolation

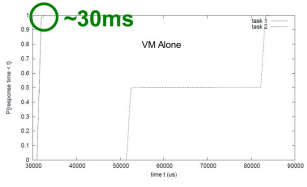
Resource sharing
→ **Temporal interference**



Need for Performance Isolation

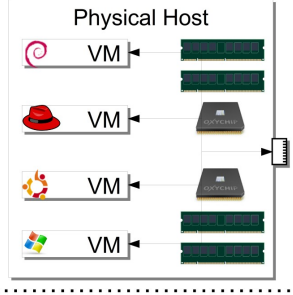
Resource sharing
→ **Temporal interference**

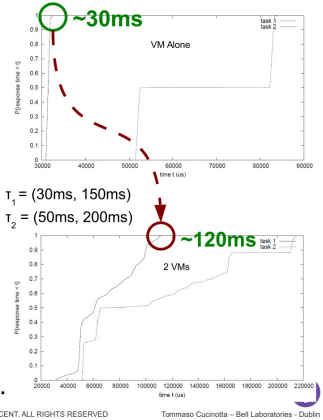




Need for Performance Isolation

Resource sharing
→ **Temporal interference**





Co-Scheduling Virtual Machines

Issues in deploying RT SW in VMs

- Scheduling and timing
  - **VM scheduling impacts on the vision of time by guest OSes**
    - Time granularity (for measuring time and setting timers)
    - Non-uniform progress-rate of applications
  - SMP-enabled guests
    - Spin-lock primitives assume release of locks within very short time-frames
      - What happens if the **lock-owner VM is descheduled** ?
- **Benchmarking**
  - A VM may be deployed on different HW (SOA scenario)
    - How to achieve predictable performance ?
  - VMs may be deployed on **General-Purpose HW** (with cache)
    - How to account for **HW-level interferences** ?

Co-Scheduling Virtual Machines

Issues in deploying RT SW Components in VMs

- **Temporal isolation** across VMs
  - Compute-bound and I/O-bound VMs
  - Shared host resources (e.g., network interrupt drivers)
  - Intensive I/O on virtualised peripherals (big-data)
- Proper management of **shared resources**: what MP **resource-sharing protocol** is appropriate ?
  - Proper management of **priority inversion**
  - Reduced overheads (limited number of preemptions)
  - Run-time schedulability analysis and **admission control**

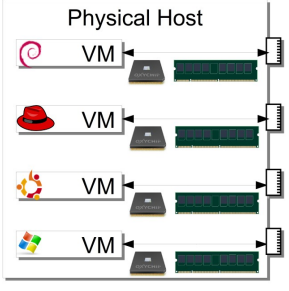
Possible Solutions

Hardware replication and static partitioning

- Computing
  - Multi-core (**1 core per VM**)
- Networking
  - Multiple network adapters (**1 network adapter per VM**)
  - Multi-queue adapters

Drawbacks

- Limitation of **flexibility**
- **Under-utilization** of resources



Possible Solutions

Another approach

- Let **multiple VMs use the same resources**
- Use proper **resource scheduling** strategies

For example

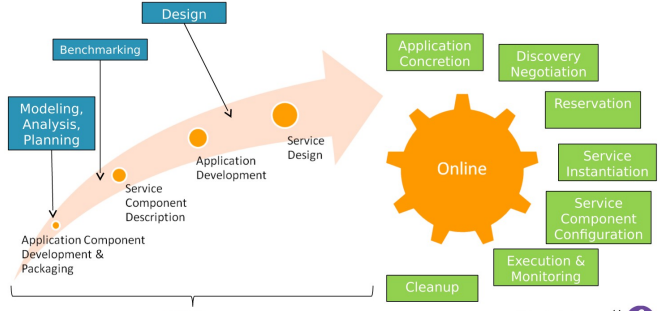
- Computing
  - Xen credit-based, SEDF schedulers, RT-Xen exts
- Networking
  - QoS-aware protocols (IntServ, MPLS)

Advantages

- Increased **flexibility**
- Increased **resource saturation** levels
- **Reduced** infrastructure **costs**

General IRMOS Approach

IRMOS Two-Phase Approach



Approach

Traditional (hard) real-time techniques are not appropriate

- lead to poor resource utilization
- imply high/unsustainable development costs

Soft real-time techniques are more appropriate

- **Stochastic models** for system/QoS evolution
- **Probabilistic guarantees** (as opposed to deterministic ones)

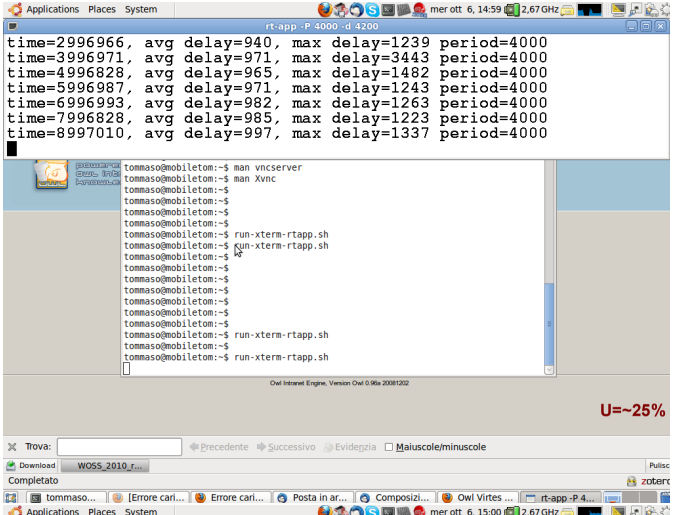
Pragmatic approach

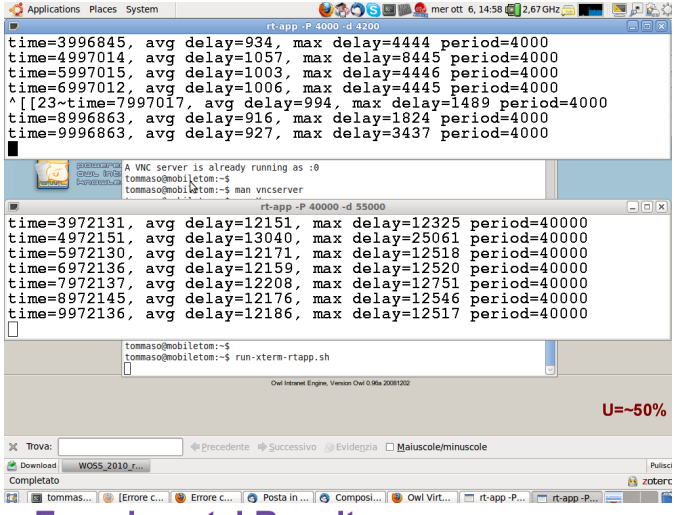
- Theory is always applied
  - on **real GPOS** (Linux)
  - with a **real Virtual Machine Monitor** (KVM)
  - on **real multimedia applications** (mplayer, vlc, ...)

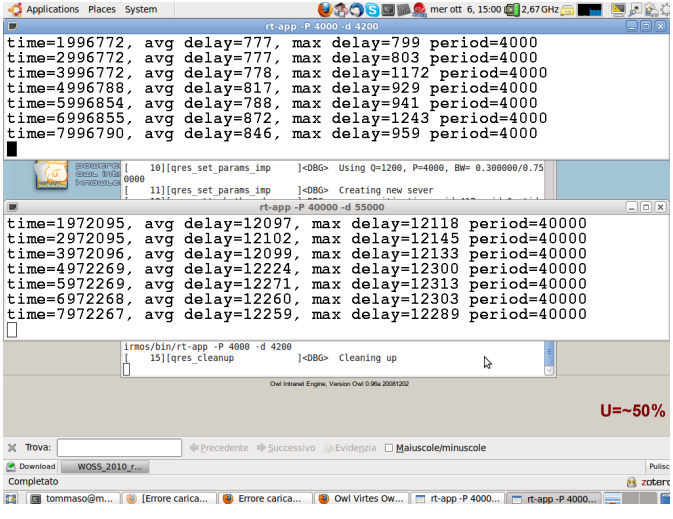
Approach

Basic Building blocks

- Linux / KVM enriched with our RT Scheduler(s)
- Each VMU is attached RT scheduling parameters
  (defining its temporal capsule)
- Improvements on the real-time virtualization performance
  - Modifications at the hypervisor level
  - Modifications at the kernel level
- Analysis of Virtualized RT applications by Hierarchical Real-Time Schedulability Analysis





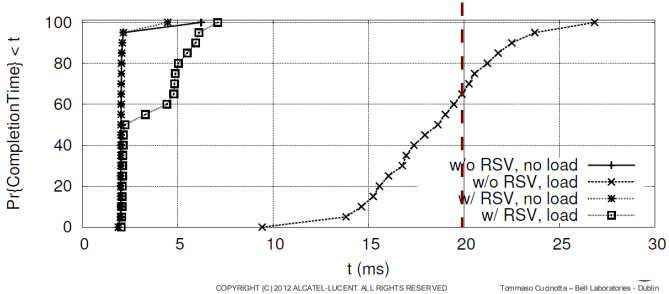


Experimental Results

(application-level benchmark)

Download time for a 100 KB file from Apache

- Periodic download requests every 20ms
- **Response-times** may be kept much more **stable** by **real-time scheduling**



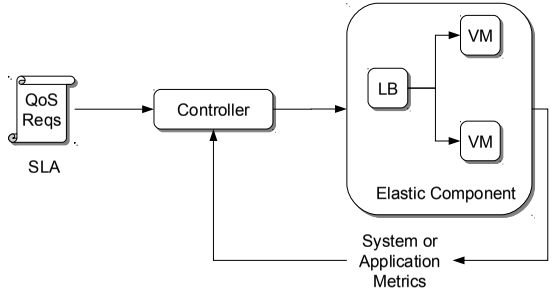
Controlling Elastic Virtualized Applications

Plethora of Cloud Providers, Tools and Frameworks

- Cloud IaaS
  - Amazon, Rackspace, Google Compute, ...
  - OpenNebula, OpenStack, CloudStack
  - CloudBand, ...
- Configuration Management (skip)
- **Monitoring and Orchestration**
  - Amazon AutoScaling, Heat+Ceilometer, Cloudify, CloudFoundry, Chef Recipes, ...

AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

Elasticity Loop



AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

But...

Adaptation logic built on unstable terrain!

AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

But...

Adaptation logic built on unstable terrain!

Can we make anything better?

Related Publications

- *"Elastic Admission Control for Federated Cloud Services,"* (to appear on) IEEE Transactions on Cloud Computing
- *"Data Centre Optimisation Enhanced by Software Defined Networking*," (to appear) in IEEE CLOUD 2014
- "Brokering SLAs for end-to-end QoS in Cloud Computing," CLOSER 2014, Barcelona
- "End-to-End Service Quality for Cloud Applications," GECON 2013, Zaragoza
- "Run-time Support for Real-Time Multimedia in the Cloud," REACTION 2013, Vancouver
- *"Admission Control for Elastic Cloud Services,"* IEEE CLOUD 2012, Hawaii
- *"Virtualised e-Learning with Real-Time Guarantees on the IRMOS Platform*," IEEE SOCA, December 2010 [best paper award]
- *"Hierarchical Multiprocessor CPU Reservations for the Linux Kernel,"* OSPERT 2009, Dublin

AT THE SPEED OF IDEAS™ Alcatel·Lucent
COPYRIGHT (C) 2012 ALCATEL-LUCENT. ALL RIGHTS RESERVED
Tommaso Cucinotta – Bell Laboratories - Dublin

Thanks for your attention

Questions ?

AN ADAPTIVE UTILISATION ACCELERATOR FOR VIRTUALIZED ENVIRONMENTS

Giovanni Toffetti, IBM Haifa Research Lab

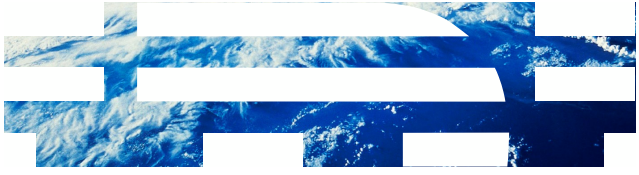
One of the key enablers of a cloud provider competitiveness is ability to over-commit shared infrastructure at ratios that are higher than those of other competitors, without compromising non-functional requirements, such as performance. A widely recognized impediment to achieving this goal is so called "Virtual Machines sprawl", a phenomenon referring to the situation when customers order Virtual Machines (VM) on the cloud, use them extensively and then leave them inactive for prolonged periods of time. Since a typical cloud provisioning system treats new VM provision requests according to the nominal virtual hardware specification, an often occurring situation is that the nominal resources of a cloud/pool become exhausted fast while the physical hosts utilization remains low.

We present IBM adaPtive UtiLiSation AcceleratoR (IBM PULSAR), a cloud resources scheduler that extends OpenStack Nova Filter Scheduler. IBM PULSAR recognises that effective safely attainable over-commit ratio varies with time due to workloads' variability and dynamically adapts the effective over-commit ratio to these changes.

An Adaptive Utilization Accelerator for Virtualized Environments

LCCC Workshop in Cloud Control

David Breitgand, Zvi Dubitzky, Amir Epstein, Oshrit Feder, Alex Glikson, Inbar Shapira and Giovanni Toffetti
Cloud Operating Systems Technologies – IBM Haifa Research Lab



What is the problem? VM sprawl in private clouds

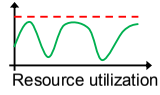
- VM sprawl
  - Proliferation of inactive / unused VMs in clouds
  - Stems from cloud provisioning model (and relative lack of control)
- In the absence of resource utilization models, VM provisioning is based on nominal resource demand

Looking at a VM:

2 VCPUs
4 GB RAM

Nominal resource demand

!=



Resource utilization

Looking at a DC:

My private cloud

100%

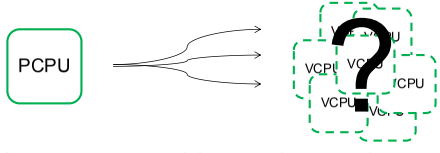
Fully utilized nominal capacity

Low Infrastructure utilization

VM provisioning based on nominal resource demand

Unsatisfied VM placement demand

Common solution: resource over-commit



Over-commit Ratio (OCR) = #VCPUs / #PCPUs

- OCR is a cloud-wide configuration parameter
- Default CPU OCR for Openstack: 16(!)

⇧ OCR => infrastructure utilization ⇧

⇧ OCR => risk of congestion ⇧

performance degradation ⇧

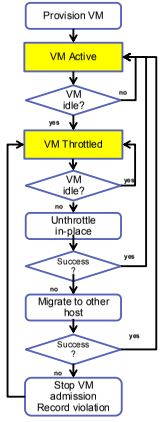
[OpenStack Operations Guide, 2014-03-06]

Our proposal – Adaptive over-commit

- RATIONALE:
  - There is **NO "right"** fixed OCR
  - VMs "activity" vs. "idleness" are application-dependent and vary over time
  - Need for automated solution
- GOALS/FEATURES:
  - Increase DC utilization
  - Minimize performance degradation
  - Transparent to VM tenants
  - No assumptions/forecast on VM resource consumption

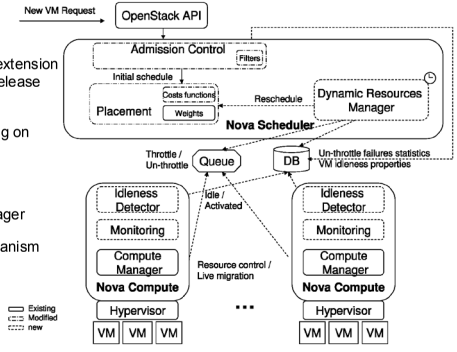
Pulsar high-level functioning

- **IBM Adaptive Utilization Accelerator for Virtualized Environments (PULSAR):**
  - Simple VM idleness detector (CPU util threshold)
  - Claim resources from idle VMs by 'throttling' them (reducing their resource reservation, **cgroups in KVM**)
  - Use **adjusted capacity** (considering throttling) to provision and place more VMs in the system



Implementation

- Full implementation as extension to OpenStack Havana release (soon IceHouse)
- Idleness detector running on each host
- Adjusted capacity filter
- Dynamic resource manager
- Admission control mechanism

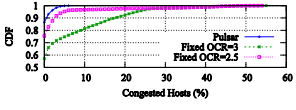


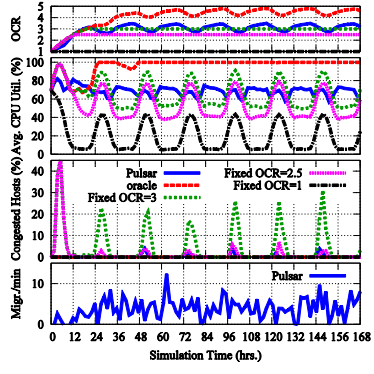
Pulsar evaluation

- Experiments
  - Smaller testbed
  - Full implementation
  - (Synthetic) trace-driven workload: boulders and sand model
  - Measure performance degradation
- Simulations
  - Large testbeds
  - Nova scheduler + testbed emulator (pymoc)
  - Synthetic and real datacenter trace-driven workload
  - Estimate performance degradation through host congestion
  - Compare with theoretical upper-bound "oracle" scheduler

Synthetic workload simulation

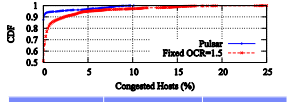
- Based on OpenStack code
- Medium-size scenario
  - 100 hosts, 24 PCPUs each (2400 total cores)
- 1 week synthetic workload using "boulders and sand" model
  - Boulders: long-living VMs with periodic demand pattern
  - Sand: short-lived VMs CPU-intensive jobs (dev-test, map-reduce) Markov-chain demand model
- Compare with fixed OCR (1,2.5,3) and Oracle (theoretical upper bound)



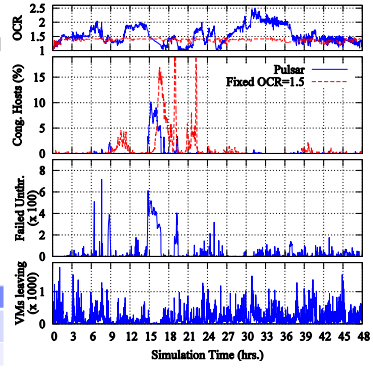


Trace-based simulation

- Pseudo-random sample from Google cluster traces [Wilkes 2011]
- 800 hosts, 48 cores each [38400 cores]
- Pulsar:
  - +20% admitted VMs
  - -50% congestion
- Limits of:
  - Reactive admission control
  - No VM preemption / priorities



10 runs avg.	Fixed OCR 1.5	PULSAR
Total admitted VMs	455k	548k
Avg. Congestion	7.27%	3.12%



Experimental evaluation

- Trace-driven experiment (trace generated with boulder/sand model)
  - Daytrader (DT) Web app [3VCPU, 30min period]
  - Sudoku solver (SD) [1VCPU, 90min avg lifetime, **5% probability of switching btw idle/active each minute**]
  - Very "active" workload, very low maximum achievable OCR [1.5 max from Oracle]
- Testbed
  - Openstack Controller node [Supermicro 8 Xeon E5420 2.5 GHz cores, 8GB RAM]
  - 2 Openstack Compute nodes [IBM System X3550 M3, 24 Xeon X5680 3.3 GHz cores, 28GB RAM]
- Runs (averaged over 20 executions):
  - R1: 4 DTs + 36 SDs (group A), OCR=1
  - R2: PULSAR with group A + SDs from a Poisson process with 2 minutes inter-arrival time (group B)
  - R3: fixed OCR=1.27 (average obtained by Pulsar) groups A+B

Run	DT avg RT (STD) [ms]	SD avg thr (STD) [Hz]	Host 1 avg util [%]	Host 2 avg util [%]
R1	28.8 (6.4)	61.2 (13)	57.8	62.3
R2	34.6 (9.7)	50.25 (14.1)	79.1	80.4
R3	41.6 (11.8)	46.95 (12.85)	85	84

Conclusion

From our evaluation:

- PULSAR is adaptive to changes in resource utilization
- It increases infrastructure utilization
- Limited host congestion
- Limited number of VM migrations
- Outperforming any fixed-OCR solution

Future work:

- Use improved idleness detector / load predictors
- Proactive admission control / VM priorities - preemption
- Larger experiments!

- **Questions?**

Backup slides

Related work

- Many papers on demand prediction for stable VM population [Breitgand 2012] [Chen 2011] [Meng 2010] [Gmach 2007]
  - We consider dynamic VM population, discrepancy between nominal and actual resource usage, adaptive over-commit
- [Gmach 2012] [Yanagisawa 2013] assume static VM population and no overcommit model, use past VM demand patterns to predict future demand
  - We left out prediction of future demand on purpose assuming dynamic VM population, albeit we could leverage this information
- [Carrera 2012] aim at fair placement decision by using a model of expected performance given a resource allocation for each workload
- [Blagodurov 2013] requires application performance monitoring instrumentation and knowledge of resource consumption profiles to classify applications as batch or interactive
- [Wuhib 2012] use average resource utilization over a sliding window to implement different placement policies (e.g., consolidation). The same solution can be applied for adaptive overcommit. However churn and high utilization variation cause number of required migrations to grow quickly

References

- [Wilkes 2011]: John Wilkes, “More Google cluster data”, Google research blog, Nov 2011
- [Breitgand 2012] D. Breitgand and A. Epstein, “Improving Consolidation of Virtual Machines with Risk-Aware Bandwidth Oversubscription in Compute Clouds,” in INFOCOM, 2012.
- [Chen 2011] M. Chen, H. Zhang, Y.-Y. Su, X. Wang, G. Jiang, and K. Yoshihira, “Effective VM Sizing in Virtualized Data Centers,” in IEEE/IFIP IM'11, Dublin, Ireland, May 2011.
- [Meng 2010] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, “Efficient Resource Provisioning in Compute Clouds via VM Multiplexing,” in The 7th IEEE/ACM International Conference on Autonomic Computing and Communications, Washington, DC, USA, Jun 2010.
- [Gmach 2007] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, “Workload Analysis and Demand Prediction of Enterprise Data Center Applications,” 2007 IEEE 10th International Symposium on Workload Characterization, pp. 171–180, Sep. 2007.
- [Gmach 2012] D. Gmach, J. Rolia, and L. Cherkasova, “Selling T-shirts and Time Shares in the Cloud,” Cloud and Grid Computing, 2012.
- [Yanagisawa 2013] H. Yanagisawa, T. Osogami, and R. Raymond, “Dependable Virtual Machine Allocation,” in Infocom, 2013, pp. 653–661.
- [Blagodurov 2013] S. Blagodurov, D. Gmach, M. Arlitt, Y. Chen, C. Hyser, and A. Fedorova,“Maximizing Server Utilization while Meeting Critical SLAs via Weight-Based Collocation Management,” in IFIP/IEEE IM'13, 2013.
- [Wuhib 2012] F. Wuhib, R. Stadler, and H. Lindgren, “Dynamic resource allocation with management objectives: Implementation for an OpenStack cloud,” in CNSM'12, 2012, pp. 309–315.

DYNAMIC POWER MANAGEMENT IN DATA CENTERS: THEORY & PRACTICE

Mor Harchol-Balter, Computer Science Department, Carnegie Mellon University

Energy costs for data centers continue to rise, already exceeding ten billion dollars yearly. Sadly much of this power is wasted. Servers are only busy 10-30% of the time, but they are often left on, while idle, utilizing 60% of more of peak power while in the idle state. The obvious solution is dynamic power management: turning servers off, or re-purposing them, when idle. The drawback is a prohibitive "setup cost" to get servers back on. The purpose of this talk is to understand the effect of the "setup cost" and whether dynamic power management makes sense.

We first turn to theory and study the effect of setup cost in an M/M/k queue. We present the first analysis of the M/M/k/setup queueing system. We do this by introducing a new technique for analyzing infinite, repeating, Markov chains, which we call Recursive Renewal Reward (RRR).

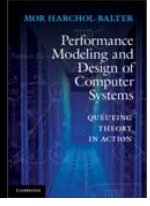
We then turn to implementation, where we implement and evaluate dynamic power management in a multi-tier data center with key-value store workload, reminiscent of Facebook or Amazon. We propose a new dynamic algorithm, AutoScale, which is ideally suited to the case of unpredictable, time-varying load, and we show that AutoScale dramatically reduces power in data centers.

Joint work with: Anshul Gandhi, Alan Scheller-Wolf, and Mike Kozuch

Power Management in Data Centers: Theory & Practice

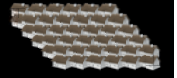
Mor Harchol-Balter
Computer Science Dept
Carnegie Mellon University

Anshul Gandhi, Sherwin Doroudi,
Alan Scheller-Wolf, Mike Kozuch



Power is Expensive

Annual U.S. data center energy consumption
||
100 Billion kWh or 7.4 Billion dollars
||
Electricity consumed by 9 million homes
||
As much CO_2 as all of Argentina



Sadly, most of this energy is wasted

[energystar.gov], [McKinsey & Co.], [Gartner]

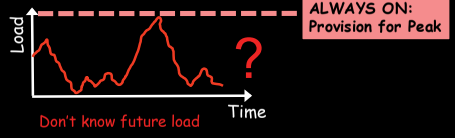
Most Power is Wasted

Servers only busy 5-30% time on average,
but they're left ON, wasting power. *[Gartner Report] [NYTimes]*

Setup time 260s 200W

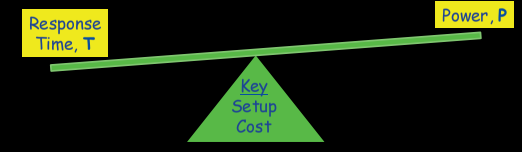
- BUSY server: 200 Watts
- IDLE server: 140 Watts
- OFF server: 0 Watts

Intel Xeon E5520
2 quad-core 2.27 GHz
16 GB memory

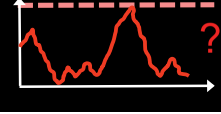


ALWAYS ON:
Provision for Peak

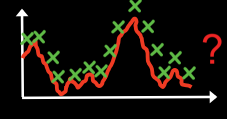
Talk Thesis



ALWAYS ON
+ Low response time
- Wastes power



ON/OFF
- High response time
+ Might save power

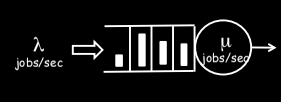


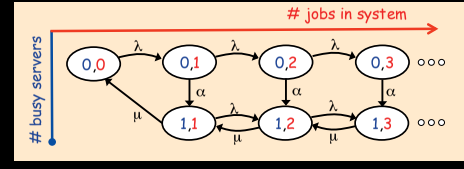
Outline

- Part I: Theory - M/M/k
  -- What is the effect of setup time?
- Part II: Systems Implementation
  Dynamic power management in practice

5

M/M/1/Setup

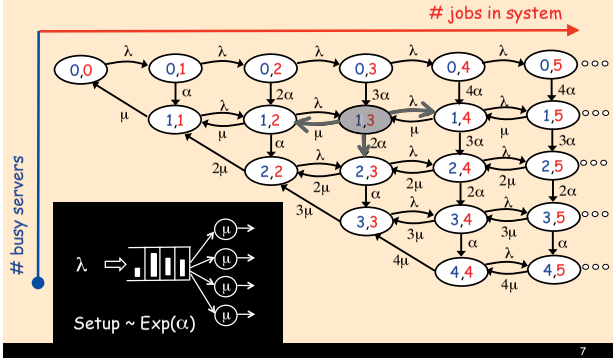




[Welch '64] $E[T^{M/M/1/Setup}] = E[T^{M/M/1}] + E[Setup]$

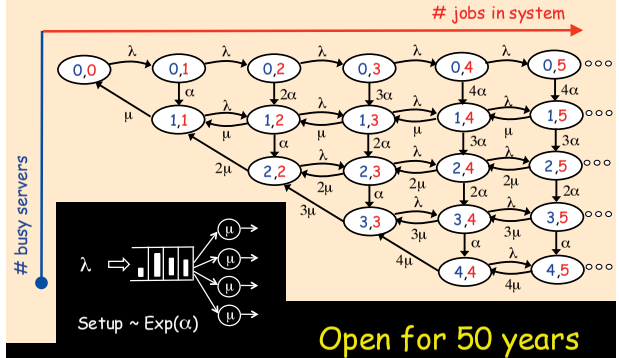
6

M/M/k/Setup (k=4)



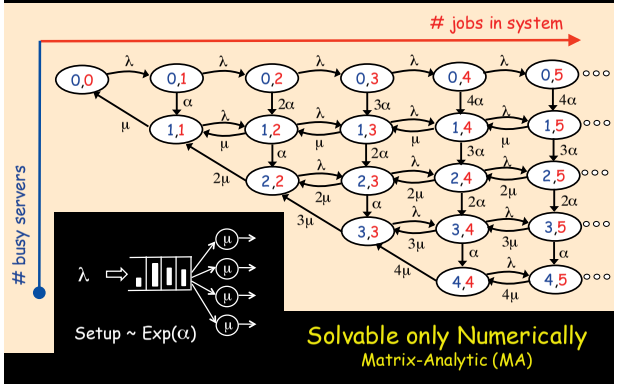
7

M/M/k/Setup (k=4)



Open for 50 years

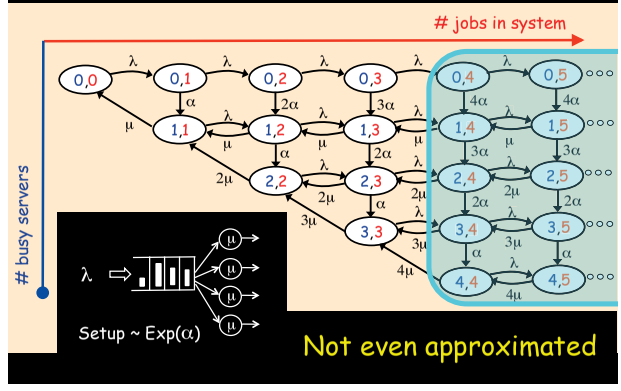
M/M/k/Setup (k=4)



Solvable only Numerically

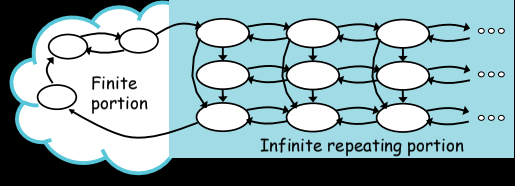
Matrix-Analytic (MA)

M/M/k/Setup (k=4)



Not even approximated

New Technique: RRR [Sigmetrics 13]



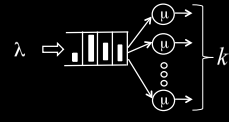
Recursive Renewal Reward (RRR)

- Exact. No iteration. No infinite sums.
- Yields transforms of response time & power.

Closed-form for all chains that are skip-free in horizontal direction and DAG in vertical direction.

11

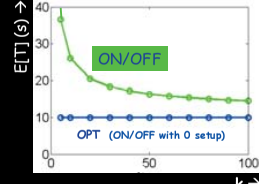
Results of Analysis

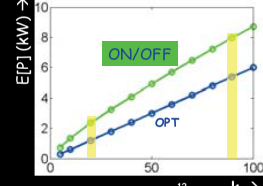


$E[\text{Job size}] = 10s$

$E[\text{Setup}] = 100s$

$\text{fix utilization} = \lambda/k\mu = 30\%$

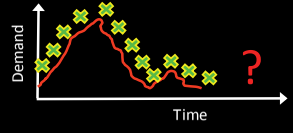




12

Outline

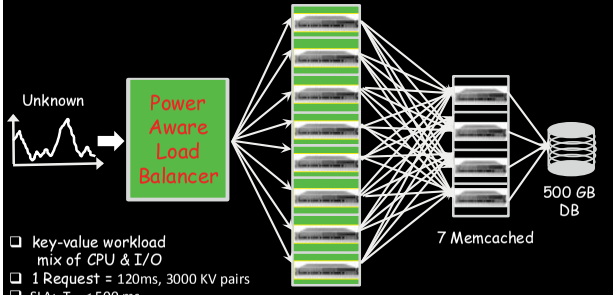
- Part I: Theory - M/M/k
  What is the effect of setup time?
  - -- Setup hurts a lot when k: small
  - -- But setup much less painful when k: large
  - -- ON/OFF allows us to achieve near optimal power
- Part II: Systems Implementation
  Dynamic power management in practice



- -- Arrivals: NOT Poisson
  Very unpredictable!
- -- Servers are time-sharing
- -- Job sizes highly variable
- -- Metric: T_{95} < 500 ms
- -- Setup time = 260 s

13

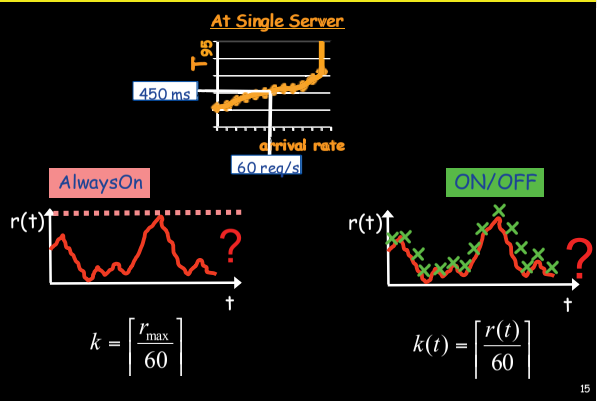
Our Data Center



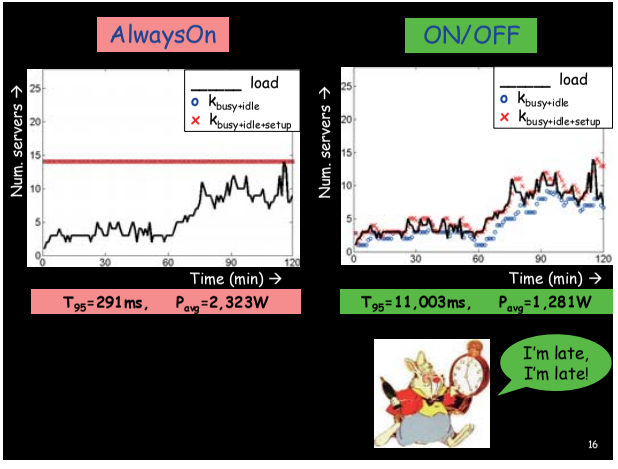
- key-value workload
  mix of CPU & I/O
- 1 Request = 120ms, 3000 KV pairs
- SLA: T_{95} < 500 ms
- Setup time: 260 s

14

Provisioning

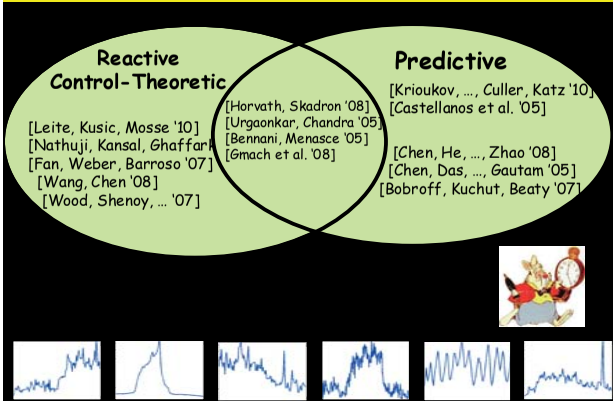


15



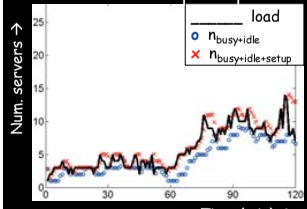
16

ON/OFF Variants



ON/OFF

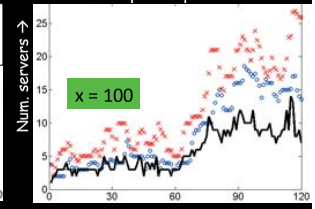
$$k(t) = \left\lceil \frac{r(t)}{60} \right\rceil$$



T_{95}=11,003ms, P_{avg}=1,281W

ON/OFF+padding

$$k(t) = \left\lceil \frac{r(t)}{60} \right\rceil \cdot (1 + x\%)$$



T_{95}=487ms, P_{avg}=2,218W

A Better Idea: AutoScale

Existing ON/OFF policies are too quick to turn servers off … then suffer huge setup lag.

Two new ideas

- Wait some time (t_{wait}) before turning idle server off
- "Un-balance" load: Pack jobs onto as few servers as possible w/o violating SLAs

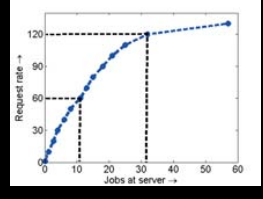
[Transactions on Computer Systems 2012]

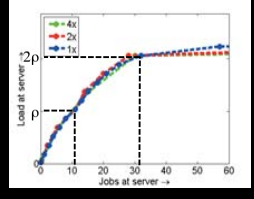
Scaling Up via AutoScale

Request rate is insufficient indicator of load. # jobs/server more robust indicator.

But not so obvious how to use # jobs/server …

10 jobs/server ⇔ load ρ
30 jobs/server ⇔ load 2ρ





[Transactions on Computer Systems 2012]

Why AutoScale works

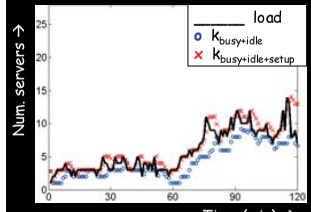
Theorem : As k → ∞, M/M/k with DelayedOff + Packing approaches square-root staffing.

$$k_{avg}^{AutoScale} \rightarrow k_{avg}^{OPT} + \sqrt{k_{avg}^{OPT} \log\left(k_{avg}^{OPT}\right)}$$

[Performance Evaluation, 2010 (b)]

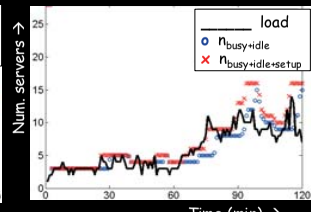
21

ON/OFF



T_{95}=11,003ms, P_{avg}=1,281W

AutoScale



T_{95}=491ms, P_{avg}=1,297W

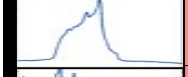
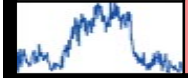
I'm late, I'm late!

Within 30% of OPT power on all our traces!

Facebook cluster-testing AS

22

Results

	AlwaysOn		ON/OFF		AutoScale	
	T_{95}	P_{avg}	T_{95}	P_{avg}	T_{95}	P_{avg}
	291 ms	2323 W	11,003 ms	1281 W	491 ms	1297 W
	271 ms	2205 W	3,802 ms	759 W	466 ms	1016 W
	289 ms	2363 W	4,227 ms	1,391 W	470 ms	1679 W
	377 ms	2263 W	> 1 min	849 W	556 ms	1412 W

[Transactions on Computer Systems 2012]

23

Conclusion

Dynamic power management ➔ Managing the setup cost

Part I: Effect of setup in M/M/k

- First analysis of M/M/k/setup and M/M/∞/setup
- Introduced RRR technique for analyzing repeating Markov chains
- Effect of setup cost is very high for small k, but diminishes as k increases

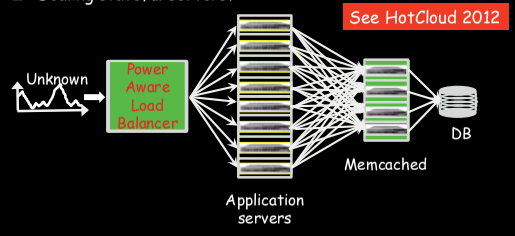
Part II: Managing the setup cost in data centers

- Non-Poisson arrival process; load unknown; unpredictable spikes
- Leaving servers AlwaysOn wastes power, but setup can be deadly.
- Lesson: Don't want to rush to turn servers off.
- Proposed AutoScale with Delayedoff, Packing routing & Non-linear Scaling.
- Demonstrated effectiveness of AutoScale in practice and theory.

24

Comments related to LCCC

- Scaling stateful servers?



- Tradeoffs between architectures:
  "Should we separate stateful from stateless?"

See Middleware 2012 – best of both

References

Anshul Gandhi, Sherwin Doroudi, Mor Harchol-Balter, Alan Scheller-Wolf. "Exact Analysis of the M/M/k/setup Class of Markov Chains via Recursive Renewal Reward." *ACM SIGMETRICS 2013 Conference*, June 2013.

Anshul Gandhi, Mor Harchol-Balter, R. Raghunathan, Mike Kozuch. "AutoScale: Dynamic, Robust Capacity Management for Multi-Tier Data Centers." *ACM Transactions on Computer Systems*, vol. 30, No. 4, Article 14, 2012, pp. 1-26.

Anshul Gandhi, Timothy Zhu, Mor Harchol-Balter, Mike Kozuch, "SOFTScale: Stealing Opportunistically for Transient Scaling." *Middleware* 2012.

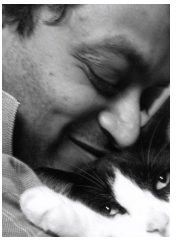
Timothy Zhu, Anshul Gandhi, Mor Harchol-Balter, Mike Kozuch. "Saving Cash by Using Less Cache." HotCloud 2012.

Anshul Gandhi, Mor Harchol-Balter, and Ivo Adan. "Server farms with setup costs." *Performance Evaluation*, vol. 67, no. 11, 2010, pp. 1123-1138.

Anshul Gandhi, Varun Gupta, Mor Harchol-Balter, and Michael Kozuch. "Optimality Analysis of Energy-Performance Trade-off for Server Farm Management." *Performance Evaluation* vol. 67, no. 11, 2010, pp. 1155-1171.

REDUCING POWER DELIVERY COSTS FOR CLOUD DATA CENTERS

Bhuvan Urgaonkar, Penn State University

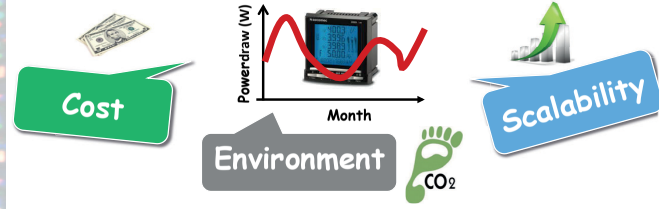


Power-related costs are significant and growing components of overall data center expenditure (both capital and operational). In this talk, I will provide a brief overview of the sources and complexities of these costs, and of our work on using batteries and several IT knobs for reducing these costs. I will then provide a more detailed description of our use of stochastic control techniques for optimizing the data center's monthly electric utility bill. Finally, I will discuss complementary related work, open problems, and future directions.

Optimizing Peak Power-Related Costs in Cloud Data Centers

Bhuvan Urgaonkar
Department of Computer Science and Engineering
Penn State University
Collaborators: C. Wang, G. Kesidis

Data Centers Consume Lots of Power!



f treated as a country, *fifth* in the world for electricity use
ouble in next 5 years, imposing a peak load of over 20 GW on the grid

Monthly Cost of 10MW Data Center

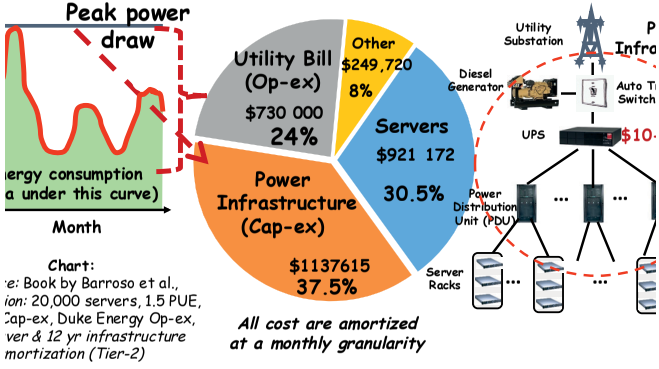
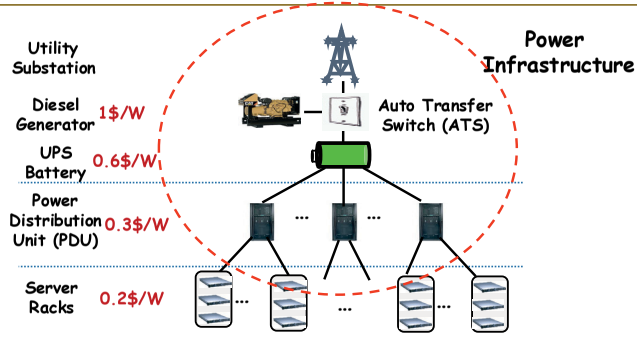


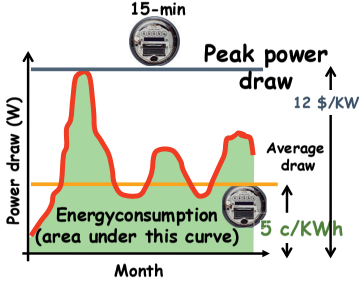
Chart:
e: Book by Barroso et al.,
ion: 20,000 servers, 1.5 PUE,
Cap-ex, Duke Energy Op-ex,
ver & 12 yr infrastructure
mortization (Tier-2)

All cost are amortized at a monthly granularity

Provisioned Peak Power Impact on Cap-ex



Consumed Peak Draw Contribution to Op-Ex (Explicit Peak-based Tariff)

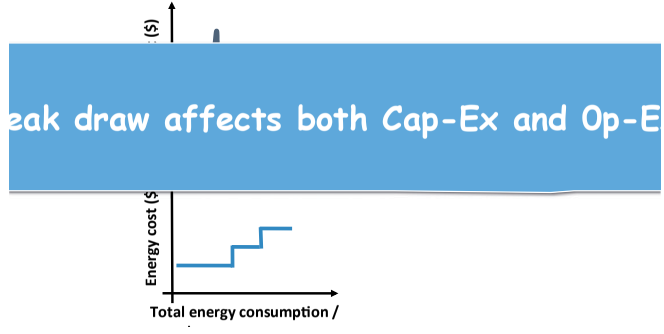


Duke Utility Tariffs
(12 $/KW, 5 c/KWh)

Note: Tariff rates collected from Duke Energy Utility.

Consumed Peak Draw Contribution to Op-Ex (Implicit)

eal-time pricing with high "coincident" peak charges



Optimizing Cap-Ex and Op-Ex

Cap-Ex optimization: How much capacity to provision for the next several years?

- An offline problem

Optimizing Cap-Ex and Op-Ex

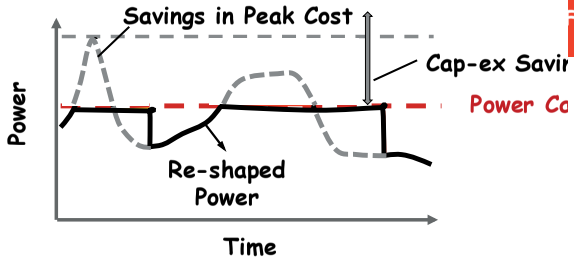
Cap-Ex optimization: How much capacity to provision for the next several years?

- An offline problem

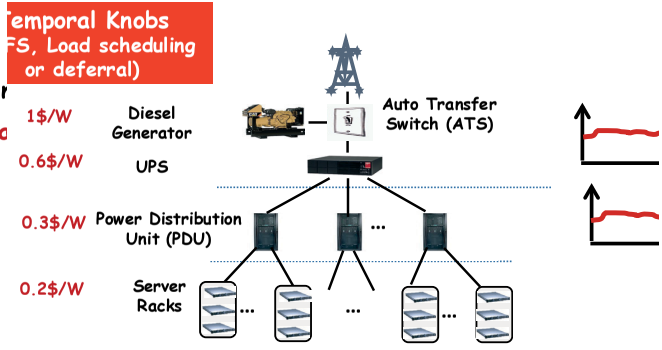
Op-ex: How much peak to admit for ***this billing cycle***?

- An online control problem
  - Control windows may be in the minutes (or even seconds)
- Complementary problem: how to operate cost-effectively within a specified power capacity (as determined by cap-ex optimization)

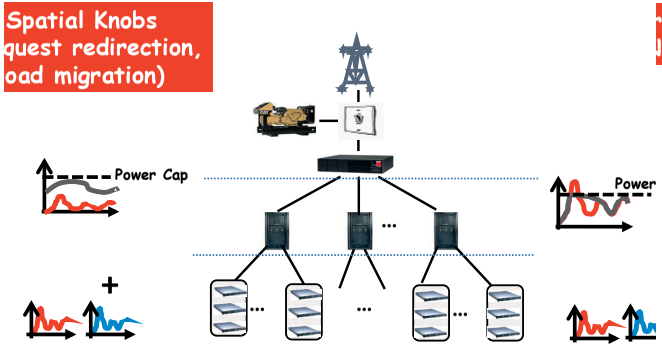
Demand Response: An Important Set of Techniques for Optimizing Power Costs



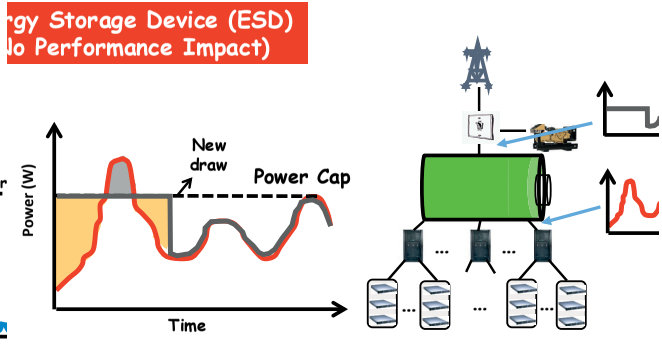
Demand Response Knobs in a Data Center



Demand Response Knobs in a Data Center



Demand Response Knobs in a Data Center



Overview of our Work

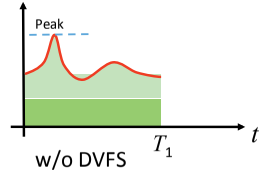
This talk
- Op-ex optimization using IT control knobs for a peak-based utility pricing scheme

Other work (happy to discuss offline)
- Cap-ex improvements via provisioning of batteries and local generation sources
- Op-ex optimization:
  - Real-time utility pricing schemes
  - Control of batteries and local generation sources

A Simple Model for IT-based DR

Despite their diversity, IT knobs can be viewed as *dropping* and/or *delaying* some power demand at the cost of performance degradation / revenue loss.

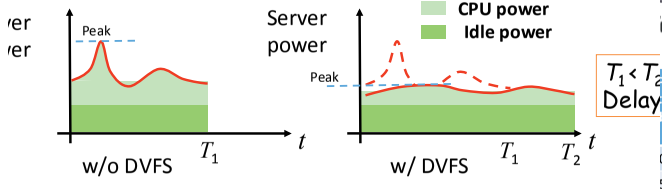
Example: DVFS/Scheduling



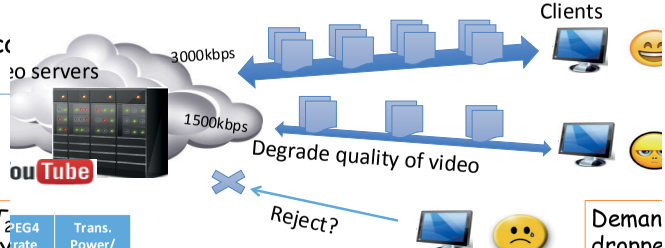
A Simple Model for IT-based DR

Despite their diversity, IT knobs can be viewed as *dropping* and/or *delaying* some power demand at the cost of performance degradation / revenue loss.

Example: DVFS/Scheduling



Example 1: MPEG Video Server

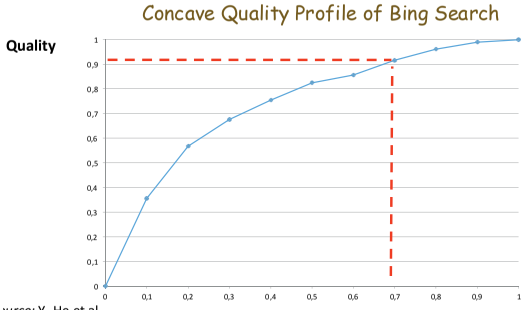


EG4 rate bps)	Trans. Power/ stream (watt)
000	3.0
500	1.5

Source: Y. Sharrab et al., MMSys'13 and Wikipedia

E.g., Source on revenue impact: "Video stre quality impacts viewer behavior," Krishna a Sitaraman, IEEE/ACM TON, 2013

Example 2: Search Engine



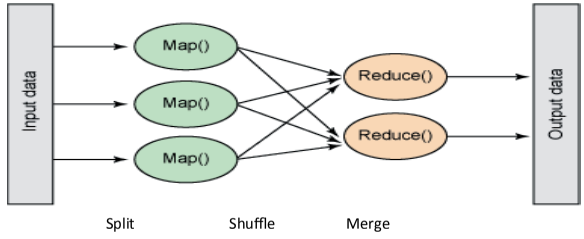
Source: Y. He et al., *SOCC'12*

bing

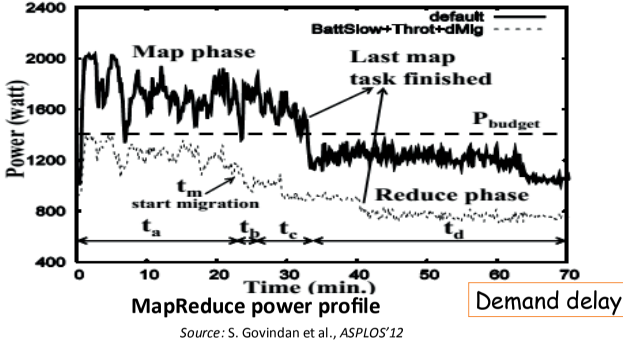
Degrade quality of query

Demand dropped

Example 3: Delay-tolerant, Batch



Example 3: Delay-tolerant, Batch



Source: S. Govindan et al., *ASPLOS'12*

Demand delay

Op-ex Optimization Problem

How to use IT-based dropping or delaying of power demand to optimize op-ex vs. performance/revenue loss trade-off?

Much Related Work for Real-time Pricing

al-time pricing

	Adversarial power demands	Stochastically known power demands
sing IT-based DR	Z. Liu et al., Sigmetrics'13, robust optimization, avoid coincident peak	
sing batteries		R. Urgaonkar et al., Sigm'11, Lyapunov optimization, distance from optimal inversely prop. to battery size P. Van de Ven et al., Energy'11, residential energy storage, MDP

21

Much Related Work for Real-time Pricing

al-time pricing

	Adversarial power demands	Stochastically known power demands
sing IT-based DR	Z. Liu et al., Sigmetrics'13, robust optimization, avoid coincident peak	**Current work (SDP formulation)**
sing batteries		R. Urgaonkar et al., Sigm'11, Lyapunov optimization, distance from optimal inversely prop. to battery size P. Van de Ven et al., Energy'11, residential energy storage, MDP

22

But Less for Peak-based Pricing

k-based pricing

	Adversarial power demands	Stochastically known power demands
sing IT-based DR	Current work: CR=2 for time-varying energy prices; CR=2-1/T for fixed energy prices	Current work (SDP formulation, gSBB heuristic)
sing batteries for DR	A. Bar-Noy et al., WEA'08, threshold-based, CR of H_n (=7.84 if 30-min time-slot)	Current work (SDP formulation, gSBB heuristic)

23

But Less for Peak-based Pricing

k-based pricing

	Adversarial power demands	Stochastically known power demands
sing IT-based DR	**Current work: CR=2 for time-varying energy prices; CR=2-1/T for fixed energy prices**	**Current work (SDP formulation, gSBB heuristic)**
sing batteries for DR	A. Bar-Noy et al., WEA'08, threshold-based, CR of H_n (=7.84 if 30-min time-slot)	**Current work (SDP formulation, gSBB heuristic)**

24

Online "Dropping" of Power Demand

Lets begin by assuming that the "knob" available to the data center is that of dropping part of the power demand

- Dropped demand never returns

Recall examples of a video streaming server and a search engine

Demand Response to Optimize Peak-based Utility Bill

How to determine the peak demand to admit in an *online* fashion?

Offline Formulation for Dropping Demand

Demand dropping

- $l_{drop}(x)$: Dropping demand v.s. Performance/Revenue loss
- Discretized optimization horizon T: A billing cycle (typically a mont
- Known demand time series $\{p_t\}_{t=1}^{T}$

Peak cost | Energy cost | Dropping cost

$$\min_{\{A_t\},\{D_t\}} \beta y_{max} + \sum_{t=1}^{T} \{\alpha_t a_t + l_{drop}(d_t)\}$$

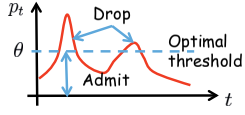
$$s.t. \quad p_t - a_t - d_t = 0, \forall t \quad \text{New demand either admitted or dropp}$$

$$y_{max} \geq a_t, \forall t \quad \text{Peak of admitted demand}$$

Online Control: ON_{Drop}

No information about future demand

Peak charge + time-varying energy price

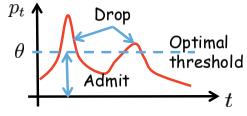


Online Control: ON_{Drop}

No information about future demand

Peak charge + time-varying energy price $l_{drop}(x) = k_{drop}x$

emma. The optimal solution has a demand dropping threshold θ of the follo
rm: If we denote as $\hat{p}_t$ the t-th largest demand value in $\{p_t\}_{t=1}^T$ and as $\hat{\alpha}_t$
rresponding energy price, then $\theta = \hat{p}_n$
here $\beta - \sum_{t=1}^{n-1}(k_{drop} - \hat{\alpha}_t) \geq 0$ and $\beta - \sum_{t=1}^{n}(k_{drop} - \hat{\alpha}_t) \leq 0$.

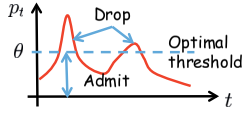


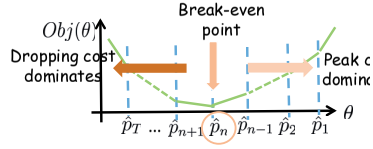
Online Control: ON_{Drop}

No information about future demand

Peak charge + time-varying energy price $l_{drop}(x) = k_{drop}x$

emma. The optimal solution has a demand dropping threshold θ of the follo
rm: If we denote as $\hat{p}_t$ the t-th largest demand value in $\{p_t\}_{t=1}^T$ and as $\hat{\alpha}_t$
rresponding energy price, then $\theta = \hat{p}_n$
here $\beta - \sum_{t=1}^{n-1}(k_{drop} - \hat{\alpha}_t) \geq 0$ and $\beta - \sum_{t=1}^{n}(k_{drop} - \hat{\alpha}_t) \leq 0$.





Online Control: ON_{Drop}

Decision-making. Admit , drop .

$\theta_0 = 0$

At time t, sort $p_1, p_2, ..., p_t$ into $\hat{p}_1, \hat{p}_2, ..., \hat{p}_t$ such that $\hat{p}_1 \geq \hat{p}_2 \geq ... \geq \hat{p}_t$.

Update θ_t as follows: Find index n such that $\beta - \sum_{t=1}^{n-1}(k_{drop} - \hat{\alpha}_t) \geq 0$ and $\beta - \sum_{t=1}^{n}(k_{drop} - \hat{\alpha}_t) \leq 0$; set $\theta_t = \hat{p}_n$.

Decision-making. Admit $\min\{p_t, \theta_t\}$, drop $[p_t - \theta_t]^+$.

*): The CR of ON_{Drop} can be improved if θ_0 can be trained using historical data.

Theorem. ON_{Drop} offers a competitive ratio of 2 under peak-based pricing.

Stochastic Control for Dropping Demand

In many cases, workloads can be predicted
- Often via Markovian models

Can develop a SDP that leverages such predictive models

Offline formulation:

$$\min_{\{\alpha_t\},\{D_t\}} E\left\{\beta y_{max} + \sum_{t=1}^{T}(\alpha_t a_t + l_{drop}(d_t))\right\}$$

Unconventional state space due to *sum + max*

Stochastic dynamic programming?

Sum + Max

Sol: Track peak-so-far by state y_t

$$y_{t+1} = \max\{y_t, a_t\}$$

Stochastic Control for Dropping Demand

SDP_{Drop} optimality rules:

$$V_t(y_t, p_{[t-1]}, \alpha_{[t-1]}) = \min_{\{A_t\},\{D_t\}} E\{\alpha_t a_t + l_{drop}(d_t) + V_{t+1}(y_{t+1}, p_{[t]}, \alpha_{[t]}) \mid P_{[t-1]} = p_{[t-1]}, \Lambda_{[t-1]} = \alpha_{[t-1]}\}$$

$$s.t. \quad y_{t+1} = \max\{y_t, a_t\}$$
$$p_t - a_t - d_t = 0$$

Lemma. Under stage-independent demand SDP_{Drop} has the following threshold-based optimal control policy :

$$(a_t^*, d_t^*) = \begin{cases} (\phi_t p_t, p_t - \phi_t p_t), & \text{if} \phi_t \le 1 \\ (p_t, 0), & \text{if} \phi_t > 1 \end{cases}$$

Making the model a bit more complex

What if dropping alone does not capture DR behavior?

Recall example of MapReduce …

Offline Problem Formulation

Demand delaying $l_{delay}(x,t)$: Delay up to τ time slots

Demand dropping $l_{drop}(x)$

Peak cost | Energy cost | Dropping cost | Dela

$$\min_{\{A_t\},\{D_t\}} \beta y_{max} + \sum_t \{\alpha_t \sum_{i\in h^+(t)} a_{i,t} + \sum_{i\in h^+(t)} l_{drop}(d_{i,t}) + \sum_{i\in h(t)} l_{delay}(a_{i,t}, t -$$

$$s.t. \quad p_t - a_{t,t} - d_{t,t} = r_{t,t+1}, \forall t$$
$$r_{i,t} - a_{i,t} - d_{i,t} = r_{i,t+1}, i \in h(t), \forall t$$
$$r_{t-\tau,t} - a_{t-\tau,t} - d_{t-\tau,t} = 0, \forall t$$
$$r_{i,T+1} = 0, i \in h(t)$$
$$y_{max} \ge \sum_{i\in h^+(t)} a_{i,t}, \forall t$$

New demand either admitted or dropp

Pending demand either admitted or dr

Delayed for τ time slots: Admit immed

No more delay at the end of billing cyc

Peak of admitted demand

Stochastic Control

SDP formulation

- Track all pending demand if $l_{delay}(x,t)$ is non-linear w.r.t. t
- Curse of dimensionality: $O(RL_p^{2(\tau+2)}L_a T)$

$$t = (r_{t-\tau,t}, r_{t-\tau+1,t}, \ldots, r_{t-1,t}, y_t)$$

$$(s_t, p_{[t-1]}, \alpha_{[t-1]}) = \min_{\{A_t\},\{D_t\}} E\{\alpha_t a_t^+ + l_{drop}(d_{t,t}) + \sum_{i\in h(t)} l_{delay}(a_{i,t}) +$$
$$V_{t+1}(s_{t+1}, p_{[t]}, \alpha_{[t]}) \mid P_{[t-1]} = p_{[t-1]}, \Lambda_{[t-1]} = \alpha_{[t-1]}]$$

Stochastic Control: Curse of Dimensionalit Stochastic Control: Linear delay cost

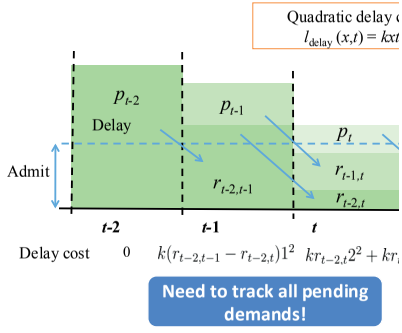
te vector

$t-\tau,t, r_{t-\tau+1,t}, \ldots, r_{t-1,t}, y_t, p_t)$

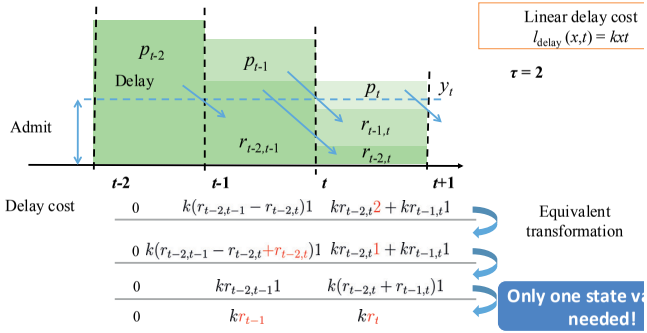
Delay / time slots	Num. of states
0	L_p^2
1	L_p^3
2	L_p^4
τ	$L_p^{2+\tau}$

Curse of dimensionality

L_p: Discretization level



Need to track all pending demands!



Only one state v needed!

Scalable Approx. for SDP

SDP_{Lin}
- Linear approximation for $l_{delay}()$
- $O(RL_p^5 L_a T)$

$$s_t = (r_t, y_t)$$

$$V_t(s_t, p_{[t-1]}, \alpha_{[t-1]}) = \min_{\{A_t\},\{D_t\}} E\{\alpha_t a_t + l_{drop}(d_t) + l_{delay}(r_t) + V_{t+1}(s_{t+1}, p_{[t]}, \mid P_{[t-1]} = p_{[t-1]}, \Lambda_{[t-1]} = \alpha_{[t-1]}\}$$

$$s.t. \quad y_{t+1} = \max\{y_t, a_t\}$$
$$r_{t+1} = (p_t - d_t) - a_t - r_t$$

What if SDP does not scale?

A scalable technique based on a "gSBB" model for power demand

gSBB-based Control

Raw demand is modeled as "generalized stochastically bounded bursty control loop" curve

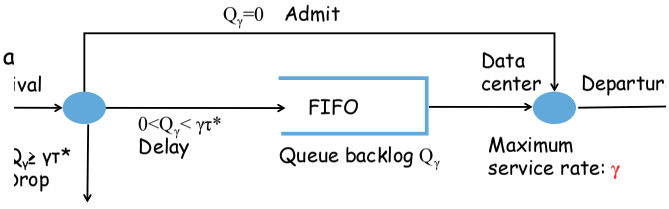
$$\{(\gamma, \phi(\gamma\tau^*)) \mid \gamma > \mu\}$$

A queue whose arrivals are the "raw" demands and is served at rate γ will have backlog Q_γ such that

$$Pr(Q_\gamma \geq \gamma\tau^*) \leq \phi(\gamma\tau^*)$$

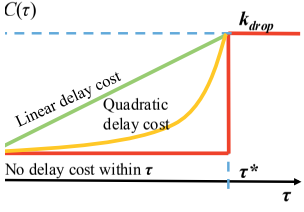
gSBB-based Control

How to obtain γ ?

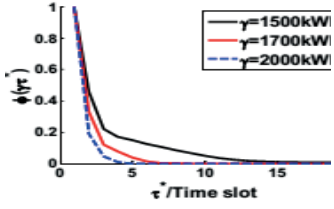


gSBB-based Control

Objective $\min_{\gamma>\mu} T\mu \int_0^\infty C(\tau) dF_\gamma(\tau) + \beta\gamma - \phi(\gamma\tau)$



Examples of C(τ)



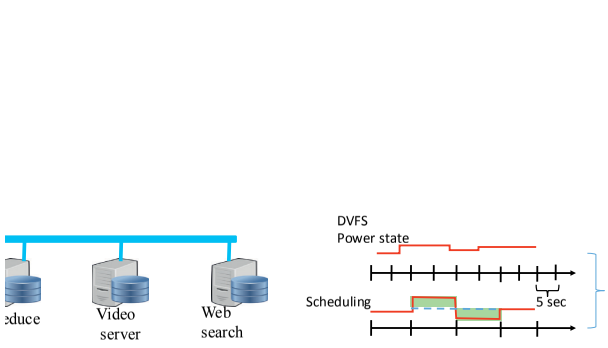
Example of $\phi(\gamma\tau^*)$

Selected Simulation Results

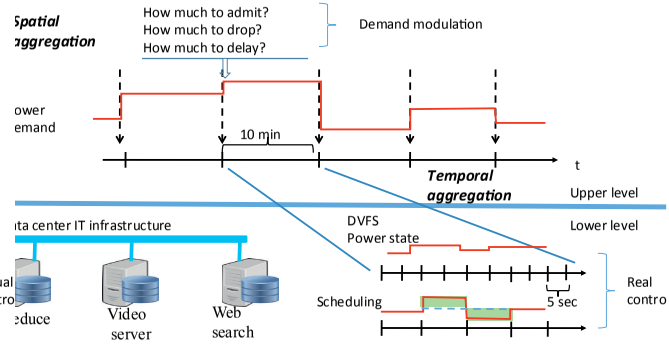
Cost benefits of demand response via abstract demand dropping and delaying

From abstract control to real control: A case study

A Hierarchical Demand Response Framework



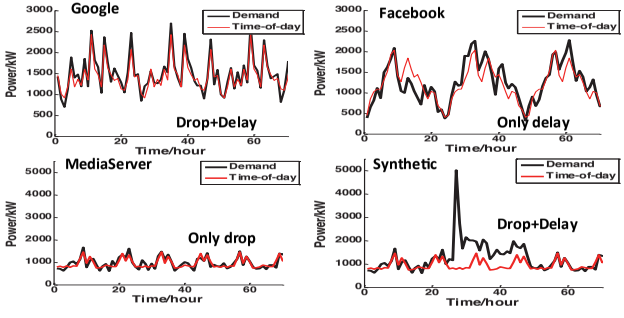
A Hierarchical Demand Response Framework



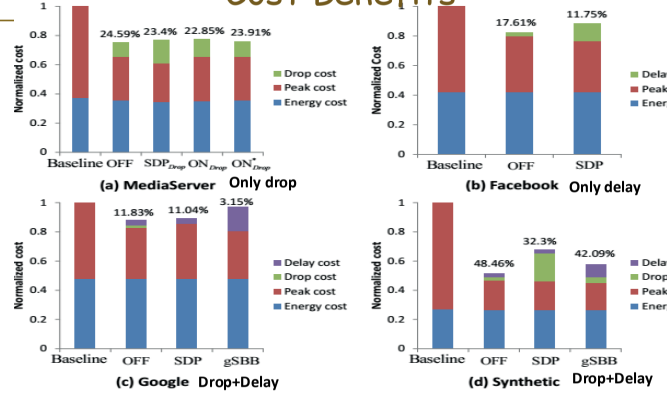
races

- Google, Facebook, MediaServer, Synthetic

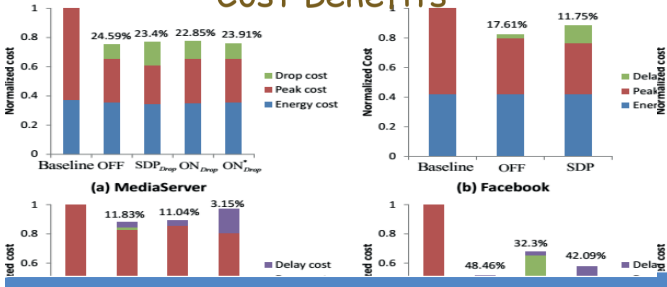
eak-based pricing



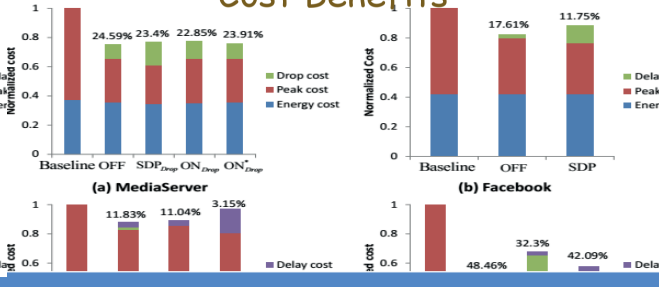
Cost Benefits



Cost Benefits



Cost Benefits

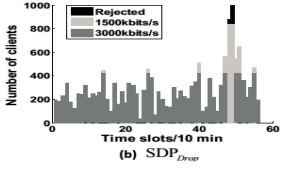


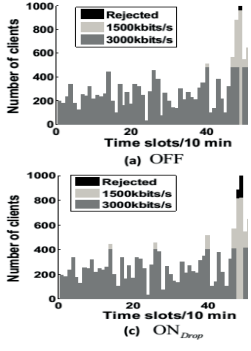
Under peak-based pricing, our approaches provide significa cost savings for real-world workloads w/o losing much "rav demand.

SDP and ON_{Drop} achieve near-optimal cost-saving except fo *Synthetic* (flash crowd); gSBB is able to handle workload unpredictability.

A Case Study: Media Server

ınslation.
Degrade QoS (trans. bitrate) to meet dropping ision as much as possible;
Otherwise, reject some requests to compensate the target dropping power that is unmet.





Conclusions and Open Problems

Peak power draw significantly impacts both cap-ex and op-ex

Algorithms and empirical case studies from our work on such optimization using IT knobs

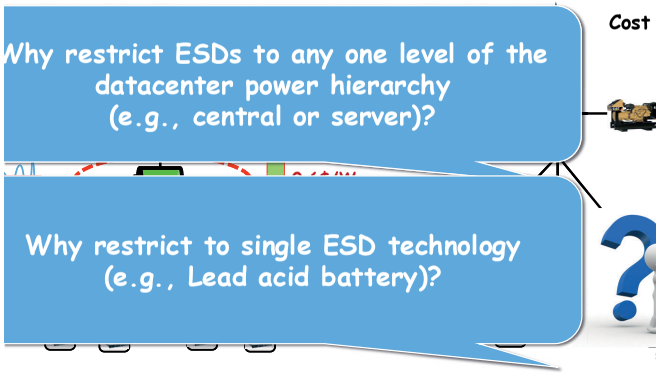
- Key idea: Abstract myriad IT knobs as dropping or delaying power demand at the cost of performance/revenue loss
- Results for both adversarial inputs and stochastically known inputs

Plenty of scope for more work (both theoretical and empirical) o op-ex optimization for peak-based pricing schemes, e.g.,:

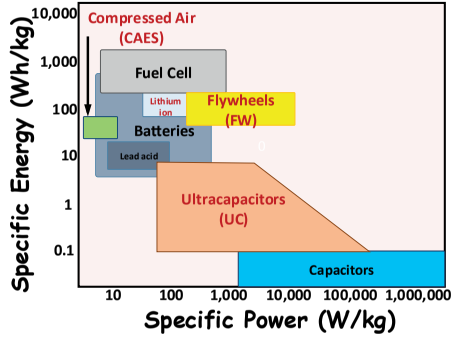
- Competitive analysis for real-time pricing using batteries for DR
- Competitive analysis for peak-based pricing using IT knobs and/or batteries when using both "dropping" and "delaying" of power demand

More details at: http://www.cse.psu.edu/~bhuvan

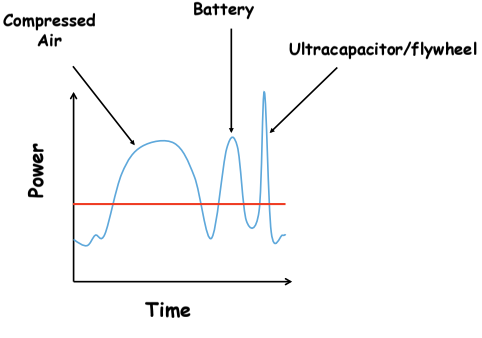
ESDs in Current Datacenters



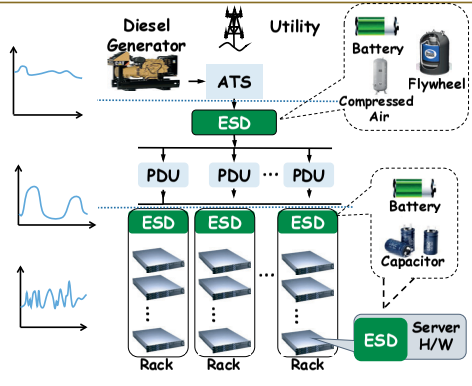
Ragone Plot



Hybrid ESD solution may be desirable



Multi-level Multi-technology ESDs



BROWNOUT: BUILDING MORE ROBUST CLOUD APPLICATIONS

Cristian Klein, Umeå University

Resource allocation in clouds is mostly done assuming hard requirements, applications either receive the requested resources or fail. Given the dynamic nature of workloads and the risk of cascading failures, guaranteeing on-demand allocations requires large spare capacity. Hence, one cannot have a system that is both reliable and efficient.

To solve this issue, we introduce brownout, a new paradigm to improve the robustness of replicated cloud applications. Brownout applications contain some optional code that can be dynamically deactivated as needed. Although this idea is simple and fairly non-intrusive to application code, properly supporting it required changes in several components. First, at the replica level, we synthesize a replica controller to decide when to execute the optional code and when to skip it. Second, we propose a resource manager to decide allocations among multiple brownout applications in a fair manner. Third, we propose two novel load-balancing algorithms, specifically designed for brownout replicas, to maximize the amount of optional content served. We theoretically prove properties of the overall system using control and game theory.

To show the practical applicability, we implemented brownout versions of RUBiS and RUBBoS with less than 170 lines of code. The load-balancing algorithms were implemented on top of lighttpd with less than 180 lines of code. Experiments show that brownout may enable considerable improvements in withstanding flash-crowds or hardware failures. Brownout opens up more flexibility in cloud resource management, which is why we encourage further research by publishing all source code.

How I Learned to Stop Worrying and Love Capacity Shortages

Cristian Klein
Umeå University

2014-05-08, LCCC Workshop on Cloud Control
Lund, Sweden

Cloud Computing Definition

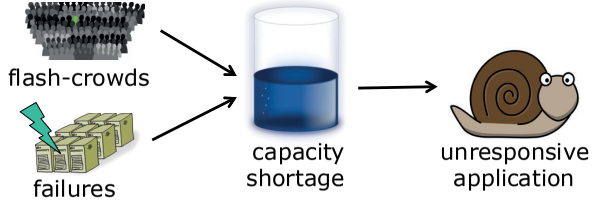
Rapid provisioning (and release) from a **shared pool** of resources

- 3 deployment models
  - Public cloud ("our stuff")
  - **Private cloud ("my stuff")**
  - Hybrid cloud (combine the two above)
- 3 service models (what is leased)
  - Software-as-a-Service (SaaS)
  - Platform-as-a-Service (PaaS)
  - **Infrastructure-as-a-Service (IaaS)**

Infrastructure-as-a-Service (IaaS)

- Data-center management
  - Lease CPU, memory, storage
  - Allocate **capacity**
  - Packed as **Virtual Machine (VM)**
- 3 stakeholders
  - Infrastructure Provider (IP)
  - Service/Application Provider (SP)
  - End-user

Problem: Unexpected Events

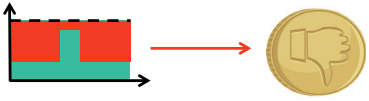


- 82% of end-users give up on a lost payment transaction*
- 25% of end-users leave if load time > 4s**
- 1% reduced sale per 100ms load time**
- 20% reduced income if 0.5s longer load time***

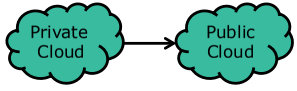
* JupiterResearch ** Amazon ***Google

State-of-Practice

- Large spare capacity
  - May be economically impractical

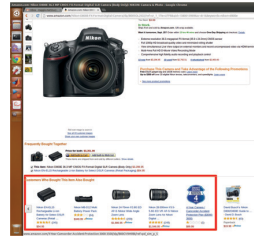


- Cloud bursting
  - Lease capacity from a public cloud
  - Does not really solve the problem



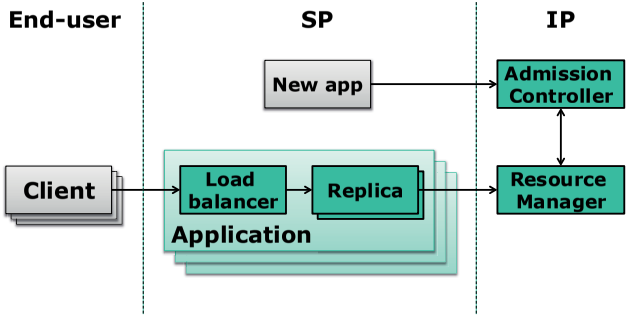
Brownout: Idea

- Disable **optional** content
  - Minimally intrusive
- E.g. recommendations
  - 50% increase in sales *
- Challenge
  - Maximize optional content
  - Avoid high response times

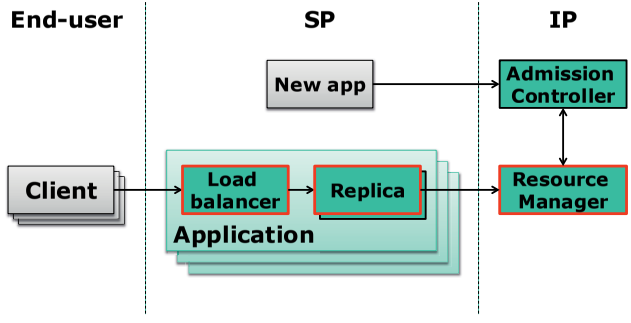


* D. Fleder et al., "Recommender systems and their effects on consumers: the fragmentation debate," in Electronic Commerce, 2010. DOI: 10.1145/1807342. 1807378.

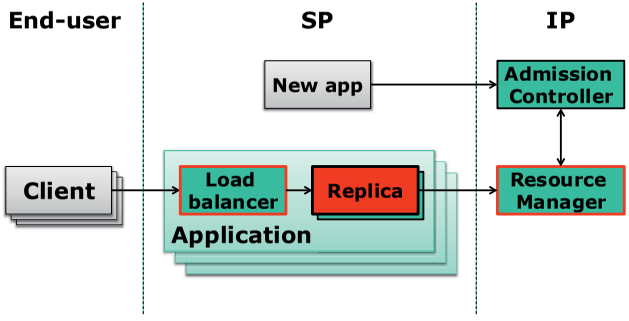
Cloud Architecture



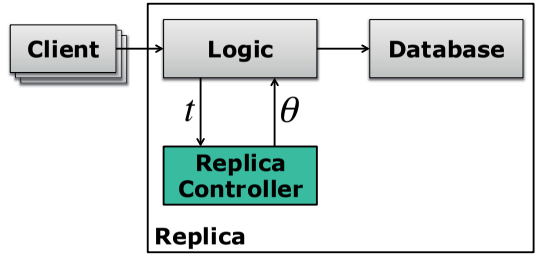
Cloud Architecture



Cloud Architecture



Brownout: Inside a Replica



t = response times

θ = probability of serving optional content (**dimmer**)

C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén,
"Brownout: building more robust cloud applications", ICSE, 2014

Replica Controller (1)

- Need to adapt to changes
  - Number of users
  - Available capacity
- Not all requests take the same time
  - E.g., cached in memory, disk
- Need to reject disturbances
  - E.g., NTP daemon firing up, cron jobs

M. Maggio, C. Klein, K-E Årzén,
"Control strategies for predictable brownouts in cloud computing", IFAC, 2014

Replica Controller (2)

- Start from a simple model

$$t^{k+1} = \alpha^k \cdot \Theta^k + \delta t^k$$

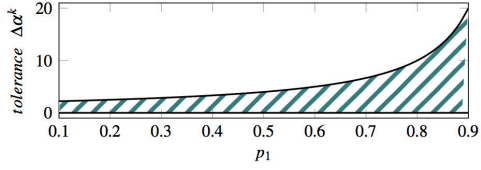
- Adaptive PI controller

$$\Theta^{k+1} = \Theta^k + \frac{1 - p_1}{\alpha} \cdot e^{k+1}$$

- α estimated using RLS

Robustness to Model Uncertainties

$$\alpha = \tilde{\alpha} \cdot \Delta\alpha$$



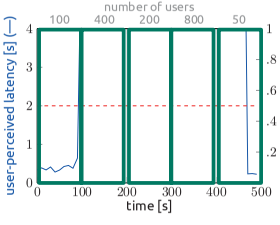
Evaluation

- RUBiS: eBay-like prototype
  - Added a recommender
- RUBBoS: Slashdot-like prototype
  - Added a recommender
  - Marked comments as optional
- Effort in lines of code:

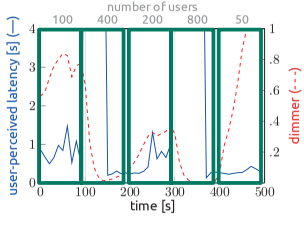
Modification	RUBiS	RUBBoS
Recommender	37	22
Dimmer	3	6
Reporting response time to controller	5	5
Controller	120	120
Total	*165*	*153*

Results: RUBiS, flash-crowd

Non-adaptive vs. self-adaptive



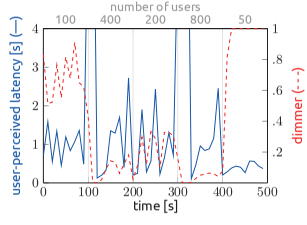
non-adaptive



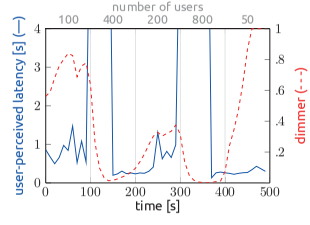
self-adaptive
pole 0.9

Results: RUBiS, flash-crowd

Self-adaptive: different pole values

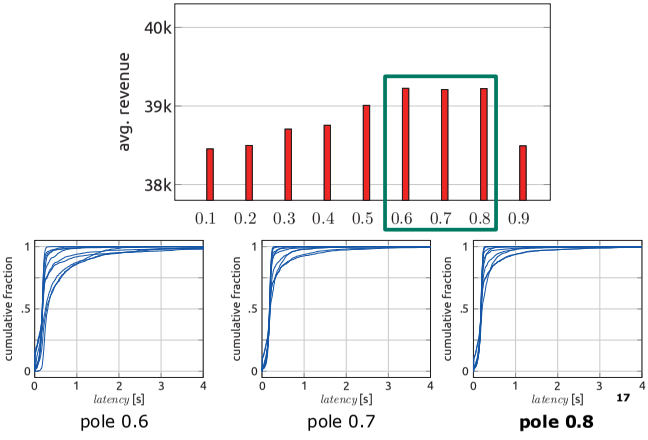


self-adaptive
pole 0.5

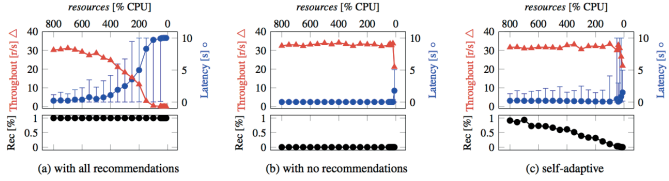


self-adaptive
pole 0.9

Results: revenue = 1×#requests + 0.5×#optional

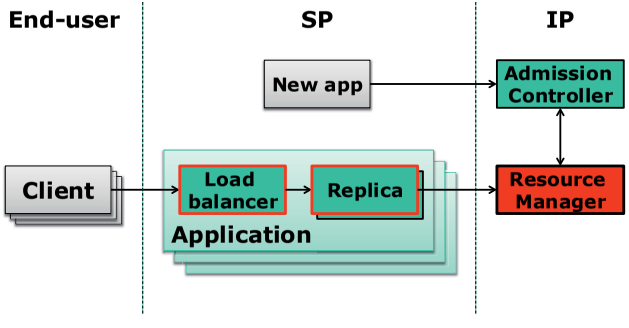


Results: Stress Test



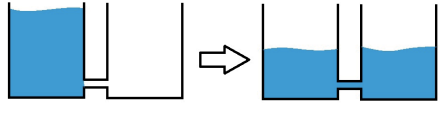
Can we do better?

Cloud Architecture

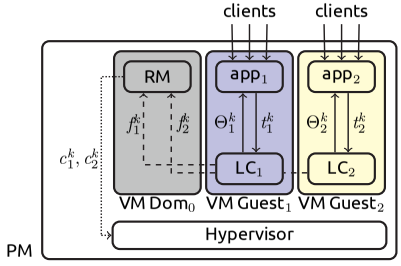


Goal

- Give capacity to application that "struggles"
- **Fairly** balances capacity among applications



Zoom: Inside a Physical Machine



C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén,
"Resource management for service level aware cloud applications", REACTION, 2013

Details

- Application sends **matching values**

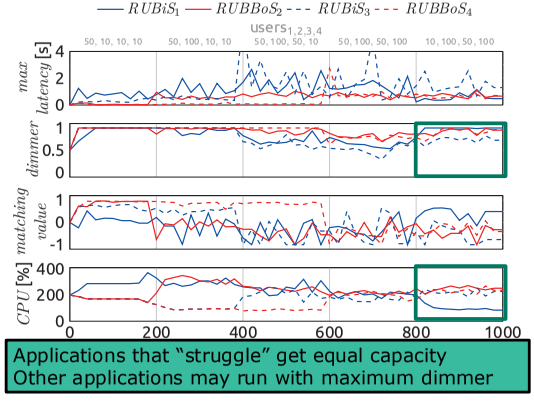
$$f_i^k = 1 - t_i^k / \bar{t}_i$$

- Resource manager computes **capacities**

$$c_i^{k+1} = c_i^k - \varepsilon_{rm}\left(f_i^k - c_i^k \cdot \sum_p f_p^k\right)$$

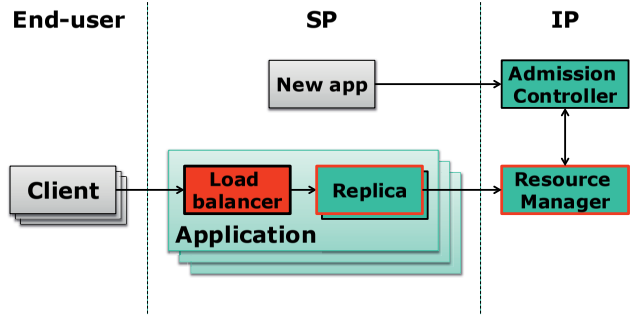
- Proven to **converge** and be **fair** using game theory

Results: 4 Applications



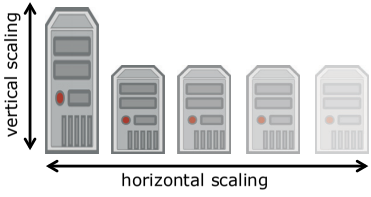
Applications that "struggle" get equal capacity
Other applications may run with maximum dimmer

Cloud Architecture



Why Replication?

- Scale beyond a physical machine

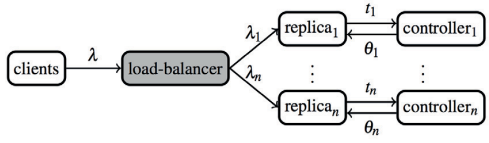


- Resilience, **hide** infrastructure failures

25

Why Replication and Brownout?

- Hide auto-scaling mishaps
- Hide failures leading to **capacity shortage**

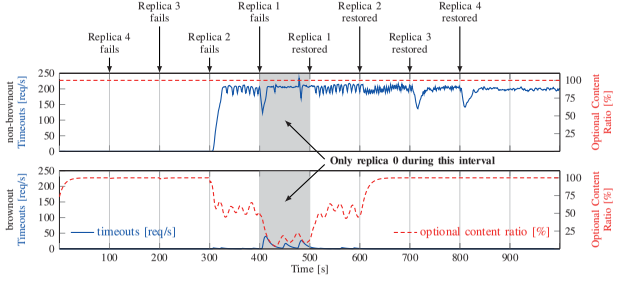


- **Goal:** Maximize optional content served

26

Results: Replication and Brownout

Brownout-unaware load-balancing: shortest queue first



27

Maximizing Optional Content

	Weight-based (periodic)	Queue-based (event)
Brownout-unaware	WRR	SQF
Variation-based	**PIBH**	**PIBH+**
Equality-based	**EPBH**	**EPBH+**
Optimization-based	**COBLB**	

Tested using simulations
- SQF best brownout-unaware method
- Brownout-aware methods better
- Somewhat slow to react

Tested using experiments (lighttpd)

J. Dürango et al. "Control-theoretical load-balancing for cloud applications with brownout", (submitted to CDC 2014)
C. Klein et al. "Improving Cloud Service Resilience using Brownout-Aware Load-Balancing", (submitted to SRDS 2014)

28

Acknowledgments

Umeå University


Erik Elmroth, PhD, Professor


Francisco Hernandez, PhD, Assistant Professor


Johan Tordsson, PhD, Assistant Professor


Luis Tomás, PhD, Post-doc

Lund University


Karl-Erik Årzén, PhD, Professor


Martina Maggio, PhD, Post-doc


Alessandro Vittorio Papadopoulos, PhD, Post-doc

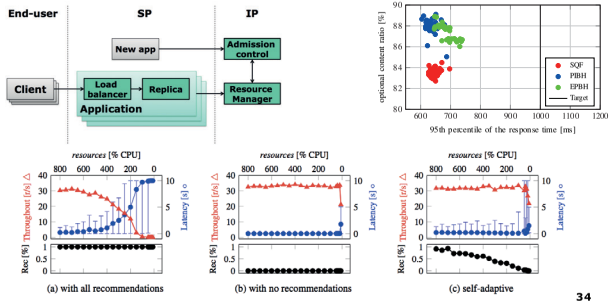

Manfred Dellkrantz, PhD Student


Jonas Dürango, PhD Student

Thank you for your attention!

Cristian Klein

How I Learned to Stop Worrying and Love Capacity Shortages



https://github.com/cloud-control

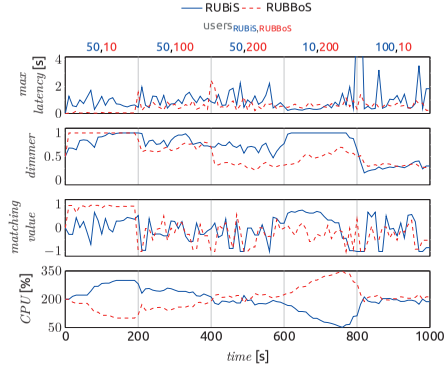
References

1. C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén, "Brownout: building more robust cloud applications", ICSE, 2014
2. M. Maggio, C. Klein, K-E Årzén, "Control strategies for predictable brownouts in cloud computing", IFAC, 2014
3. C. Klein, M. Maggio, F. Hernández-Rodriguez, K-E Årzén, "Resource management for service level aware cloud applications", REACTION, 2013
4. J. Dürango, M. Dellkrantz, M. Maggio, C. Klein, A. V. Papadopoulos, F. Hernández-Rodriguez, E. Elmroth, K-E Årzén, "Control-theoretical load-balancing for cloud applications with brownout", (submitted to CDC 2014)
5. C. Klein, A. V. Papadopoulos, M. Dellkrantz, J. Dürango, M. Maggio, K-E Årzén, F. Hernández-Rodriguez, E. Elmroth, "Improving Cloud Service Resilience using Brownout-Aware Load-Balancing", (submitted to SRDS 2014)
6. L. Tomás, C. Klein, J. Tordsson, F. Hernández-Rodriguez, "The straw that broke the camel's back: safe cloud overbooking with application brownout", (submitted to CAC 2014)

Appendix

Cloud Computing

Results: 2 Applications



RT-XEN: REAL-TIME VIRTUALIZATION FOR THE CLOUD

Chenyang Lu, Washington University in St. Louis

Recent years have witnessed increasing demand of running real-time applications in the cloud. However, existing virtualization platforms cannot provide real-time performance guarantees to virtual machines. This talk will introduce RT-Xen, a real-time virtual machine scheduling framework in the Xen hypervisor. Built based on compositional real-time scheduling theory, RT-Xen realizes a suite of real-time schedulers spanning the design space including global and partitioned multi-core scheduling, fixed and dynamic priority, and different budget management schemes. Our experimental study shows RT-Xen schedulers deliver significant improvement in real-time performance over Xen's existing credit scheduler and explores the tradeoff in real-time scheduler design in virtualized platforms. RT-Xen has been released as open-source software at https://sites.google.com/site/realtimexen/. Work is underway to incorporate RT-Xen in the Xen distribution and to integrate RT-Xen with the OpenStack cloud management system. RT-Xen represents a promising step toward real-time cloud computing for latency-sensitive applications.

CPSL
Cyber-Physical Systems Laboratory

RT-Xen: Real-Time Virtualization for the Cloud

Chenyang Lu
Cyber-Physical Systems Laboratory
Department of Computer Science and Engineering

Washington University in St. Louis

Real-Time Virtualization

- Cars are becoming real-time mini-clouds!
  - Consolidate 100 ECUs → 10 multicore processors.
  - Integrate multiple vendors' systems → common platforms.
  - Must preserve real-time guarantees on a virtualized platform!
- Internet of Things → Cyber-Physical Systems
  - Smart manufacturing, smart transportation, smart grid.
  - Internet-scale sensing and control → real-time cloud computing.
- Cloud gaming
  - Xbox One: cloud offloading computation of environmental elements
  - Sony acquired Gaikai, an open cloud gaming platform.

Virtualization is *not* real-time today

- Existing hypervisors provide no guarantee on latency
  - Xen: credit scheduler, [credit, cap]
  - VMware ESXi: [reservation, share, limitation]
  - Microsoft Hyper-V: [reserve, weight, limit]
- Public clouds lack service level agreement on latency
  - EC2, Compute Engine, Azure: #VCPUs

Current platforms provision CPU resources, not real-time performance!

Challenges

- Support real-time applications in a virtualized environment.
  - Latency *guarantees* to tasks running in virtual machines (VMs).
  - Real-time performance *isolation* between VMs.
- Real-time performance provisioning at different levels
  - Virtualization within a host
  - Communication and I/O
  - Cloud resource management

RT-Xen

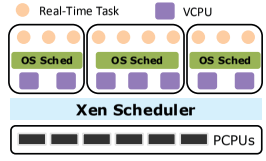
- Real-time hypervisor based on Xen
  - Real-time VM scheduling
  - Real-time communication
- Build on compositional scheduling theory
  - VMs specify resource interfaces
  - Real-time guarantees to tasks in VMs
- Open source
  - Xen patch in progress

RT-Xen Real-Time Virtualization

- RT-OpenStack: cloud management based on RT-Xen

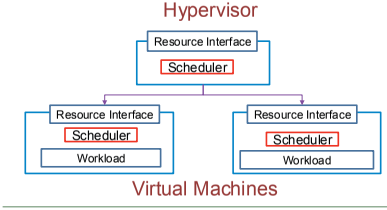
Xen Virtualization Architecture

- Xen: type-1, baremetal hypervisor
  - Domain-0: drivers, tool stack to control VMs.
  - Guest Domain: para-virtualized or fully virtualized OS.
- Xen scheduler
  - Guest OS runs on VCPUs.
  - Xen schedules VCPUs on PCPUs.
  - Credit scheduler: round-robin with proportional share.



Compositional Scheduling

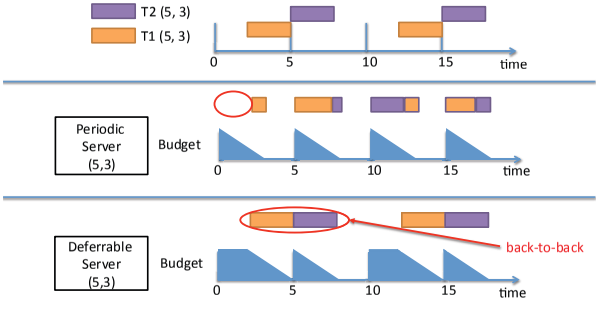
- Analytical real-time guarantees to tasks running in VMs.
- VM resource interfaces
  - Hides task-specific information
  - Multicore: <period, budget, #VCPU>
  - Computed based on compositional scheduling analysis



Real-Time Scheduling Policies

- Priority schemes
  - Static priority: Rate Monotonic
  - Dynamic priority: Earliest Deadline First (EDF)
- Multi-core
  - Global scheduling: allow VCPU migration across cores
  - Partitioned scheduling: bound VCPUs to cores

Scheduling VM as "Server"



RT-Xen: Real-Time Scheduling in Xen

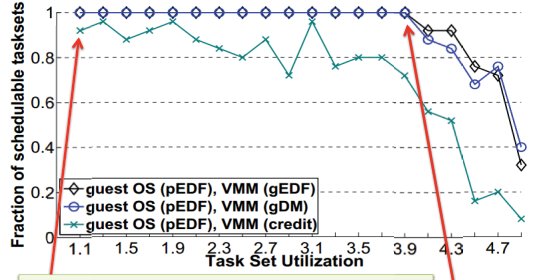
- Single-core RT-Xen 1.0
- Single-core enhanced RT-Xen 1.1
- Multi-core scheduling RT-Xen 2.0
  - RT-global
  - RT-partition

	Fixed Priority (RM)		Capacity Reclaiming Periodic	Work Conserving Periodic	Polling
	Deferrable	Periodic	Periodic	Deferrable	Sporadic
Global Scheduling	Deferrable	Periodic	Periodic	Deferrable	Partitioned Scheduling
		Dynamic Priority (EDF)			

Experimental Setup

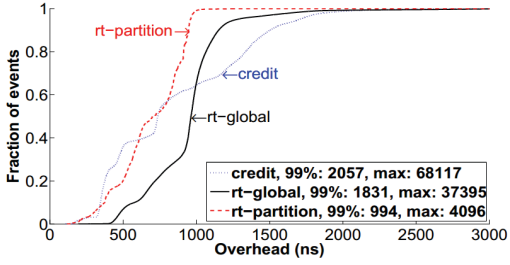
- Hardware: Intel i7 processor, 6 cores, 3.33 GHz
  - Allocate 1 VCPU for Domain-0, pinned to PCPU 0
  - All guest VMs use the remaining cores
- Software
  - Xen 4.3 patched with RT-Xen
  - Guest OS: Linux patched with LITMUS
- Workload
  - Period tasks: synthetic, ARINC 653 avionics workload (RT-Xen 1.1)
  - Allocate tasks → VMs

RT-Xen 2.0: Credit Scheduler



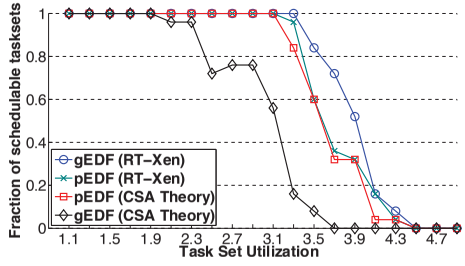
- Credit misses deadlines at 22%/CPU utilization.
- RT-Xen delivers real-time performance at 78%/CPU utilization.

RT-Xen 2.0: Scheduling Overhead



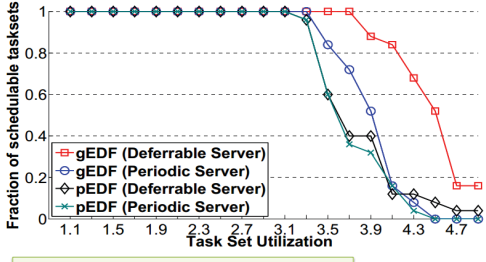
- rt-global has extra overhead due to global lock.
- Credit has poor max overhead due to load balancing.

RT-Xen 2.0: Theory vs. Experiments



- gEDF > pEDF empirically, thanks to work-conserving global scheduling.
- gEDF < pEDF theoretically due to pessimistic analysis.

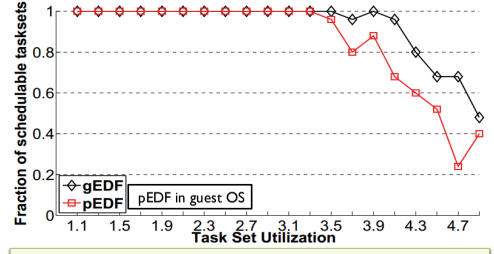
RT-Xen 2.0: Deferrable vs. Periodic



Work-conserving wins empirically!

- Deferable Server (DS) > Periodic Server.
- gEDF+DS → best real-time performance.

RT-Xen 2.0: How about Cache?



- gEDF > pEDF for cache intensive workload.
- Benefit of global scheduling dominates migration cost.
- Shared cache mitigates cache penalty due to migration.

Conclusion

- Diverse applications demand real-time virtualization and cloud.
  - Embedded real-time systems
  - Internet-scale cyber-physical systems
  - Latency-sensitive cloud applications
- RT-Xen provides real-time performance and guarantees
  - Efficient implementation of diverse real-time scheduling policies.
  - Leverage compositional scheduling theory → analytical guarantee.
  - Resource interface → systematic resource allocation for latency bounds.
- On-going
  - Working on RT-Xen patch for Xen core distribution.
  - RT-OpenStack: integration with OpenStack on the way.

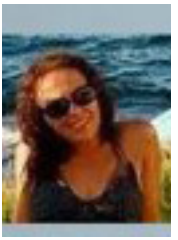
Check out RT-Xen

https://sites.google.com/site/realtimexen/

- **RT-Xen 1.0**: S. Xi, J. Wilson, C. Lu, and C.D. Gill, RT-Xen: Towards Real-Time Hypervisor Scheduling in Xen, ACM International Conferences on Embedded Software (EMSOFT), 2011.
- **RT-Xen 1.1:** J. Lee, S. Xi, S. Chen, L.T.X. Phan, C. Gill, I. Lee, C. Lu and O. Sokolsky Realizing Compositional Scheduling through Virtualization, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012
- **RT-Xen 2.0**: S. Xi, C. Lu, C. Gill, M. Xu, L.T.X. Phan, I. Lee, and O. Sokolsky, Real-Time Multi-Core Virtual Machine Scheduling in Xen, Washington University Technical Report, WUCSE-2013-109 2013
- **inter-domain communication**: S. Xi, C. Li, C. Lu, and C. Gill, Prioritizing Local Inter-Domain Communication in Xen, ACM/IEEE International Symposium on Quality of Service (IWQoS), 2013.

SERVICE LEVEL AGREEMENT FOR CLOUD COMPUTING: TOWARDS A CONTROL-THEORETIC APPROACH

Sara Bouchenak, University of Grenoble



Cloud Computing is a paradigm for enabling remote, on-demand access to a set of configurable computing resources. This model aims to provide hardware and software services to customers, while minimizing human efforts in terms of service installation, configuration and maintenance, for both cloud provider and cloud customer. A cloud may have the form of an Infrastructure as a Service (IaaS), a Platform as a Service (PaaS) or a Software as a Service (SaaS). However, cloud's ad-hoc management in terms of quality-of-service and service level agreement (SLA) poses significant challenges to the performance, availability, energy consumption and economical costs of the cloud. We believe that a differentiating element between Cloud Computing environments will be the quality-of-service and the service level agreement (SLA) provided by the cloud. In this talk, we will discuss the definition and implementation of a novel cloud model: SLAaaS (SLA aware Service). The SLAaaS model enriches the general paradigm of Cloud Computing, and enables systematic and transparent integration of service levels and SLA to the cloud. SLAaaS is orthogonal to IaaS, PaaS and SaaS clouds and may apply to any of them. Both the cloud provider and cloud customer points of view are taken into account. From cloud provider's point of view, we present autonomic SLA management to handle performance, availability, energy and cost issues in the cloud. An innovative approach combines control theory techniques with distributed algorithms and language support in order to build autonomic elastic clouds. Novel models, control laws, distributed algorithms and languages will be proposed for automated provisioning, configuration and deployment of cloud services to meet SLA requirements, while tackling scalability and dynamics issues. On the other hand from cloud customer's point of view, we discuss SLA governance. It allows cloud customers to be part of the loop and to ba automatically notified about the state of the cloud, such as SLA violation and cloud energy consumption. The former provides more transparecy about SLA guaranties, and the latter aims to raise customers' awareness about cloud's energy footprint.

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:

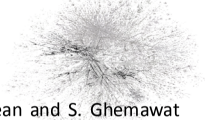
Vast amounts of data generated daily

- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams – Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

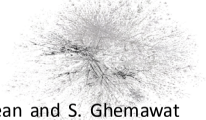
1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:
Vast amounts of data generated daily
- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

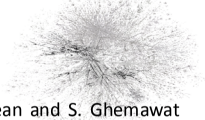
1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:
Vast amounts of data generated daily
- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

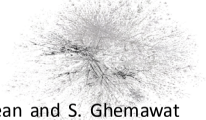
1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:
Vast amounts of data generated daily
- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:

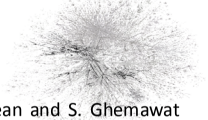
Vast amounts of data generated daily

- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams – Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

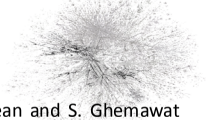
Big Data, Big Problems

Problem:
Vast amounts of data generated daily
- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams – Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:

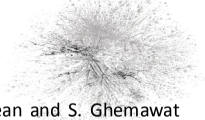
Vast amounts of data generated daily

- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams - Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

A Control Approach for Performance of Big Data Systems

Mihaly Berekmeri, Damian Serrano, Sara Bouchenak, Nicolas Marchand, Bogdan Robu

GIPSA - LIG - Grenoble University, France

The structure of the presentation

1. **Introduction**
   - Big Data MapReduce
   - State of the art
2. **Experimental setup**
   - Sensors / Actuators
   - MRBS
3. **Control**
   - Our model
   - Control architecture
   - Control examples
4. **Conclusions and Future Work**

Big Data, Big Problems

Problem:

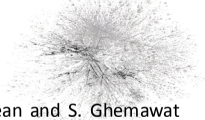
Vast amounts of data generated daily

- Facebook:
  - 1.11 x 10^9 active users, 50% log in daily
  - 3.2 x 10^9 likes and comments/day
  - > 100 clusters (largest has > 100PB, 200 million files)
- CERN's LHC: Up to 1 PB/s during experiments

How do we store it? How do we process it?

Ben Chams – Fotolia

MapReduce



Programming model introduced by J. Dean and S. Ghemawat (Google) in 2004 as a PaaS paradigm -> large scale distributed data processing on clusters of commodity computers

Automatic features: data partitioning and replication, task scheduling, fault tolerance

Used by the biggest companies :

Amazon, eBay, Facebook, LinkedIn, Twitter, Yahoo, Microsoft...

Wide range of applications :

log analysis, data mining, web search engines, scientific computing, business intelligence,...

PERFORMANCE-ENERGY TRADE-OFF IN MULTI-SERVER QUEUEING SYSTEMS WITH SETUP DELAY

Samuli Aalto, Aalto University

In this talk we review some recent results related to the performance-energy trade-off in multi-server queueing systems, where the servers have multiple energy states. In addition to the normal BUSY and IDLE states, a server can be switched OFF to save energy. However, switching the server again on results in a SETUP delay which consumes additional energy and deteriorates performance. For a single server system, we consider optimal strategies to switch the server off and on. Multi-server systems may have a central queue, or they may consist of parallel servers with their own queues. Switching servers off and on in a reasonable way is the main objective. In a system with parallel servers, one should also consider the dispatching (a.k.a. task assignment) problem: an arriving job is to be routed to one of the parallel servers. Here we present a size- and energy-aware MDP approach to solve the problem.

Aalto University
School of Electrical
Engineering

Performance-Energy Trade-off in Multi-Server Queueing Systems with Setup Delay

Samuli Aalto
Aalto University, Finland

LCCC Cloud Computing Workshop
7–9 May 2014
Lund, Sweden

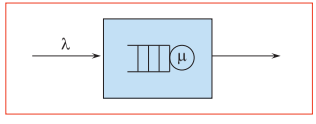
Co-operation with

Esa Hyytiä, Pasi Lassila, Misikir Gebrehiwot
(Aalto University)

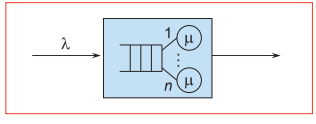
Rhonda Righter
(UC Berkeley)

Queueing models

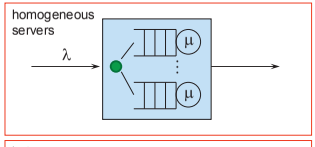
- Single-server queue (M/G/1)

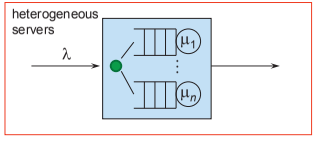


- Multi-server queue (M/M/*n*)



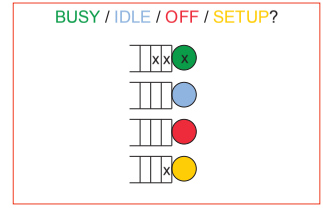
- Parallel queues





Performance-energy trade-off

- Energy saved by switching the server off when idle
- However, performance impaired, if switching the server back on takes time (setup delay)



Cost model

- Performance:

  E[*T*] = mean delay per job (in seconds)

  E[*X*] = mean number of jobs = $\lambda \cdot E[T]$

- Energy:

  E[*E*] = mean energy per job (in joules)

  E[*P*] = mean power consumed = $\lambda \cdot E[E]$

- Definition: delay = response time

Power consumption levels:

$$0 = P_{\text{off}} < P_{\text{idle}} \leq P_{\text{setup}} = P_{\text{busy}}$$

Objective function

- Energy-Response-time-Weighted-Sum (ERWS):

  $E[T] + E[E]/\beta$

  e.g. Wierman & al. (2009)

- Energy-Response-time-Product (ERP):

  $E[T] \cdot E[E]$

  e.g. Gandhi & al. (2010b)

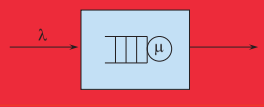
- General form:

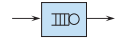
  $w_1 \cdot E[T]^{t1} \cdot E[E]^{e1} + w_2 \cdot E[T]^{t2} \cdot E[E]^{e2}$

  by Maccio & Down (2013)

  - ERWS: $w_1 = 1, t_1 = 1, e_1 = 0$ $w_2 = 1/\beta, t_2 = 0, e_2 = 1$
  - ERP: $w_1 = 1, t_1 = 1, e_1 = 1$ $w_2 = 0, t_2 = 0, e_2 = 0$

Part I
Single-server queue with setup delays





Optimal switching on/off policy

Maccio & Down (2013)

- M/G/1-FIFO
  - Setup delay *D* generally distributed with mean $1/\gamma$
- Control parameters:
  - Delayed switch-off for an exponential time with mean $1/\alpha$
  - Server switched on after *k* new job arrivals
- Objective function: Gen. form

$$w_1 \mathrm{E}[T]^{t_1}\, \mathrm{E}[E]^{e_1} + w_2 \mathrm{E}[T]^{t_2}\, \mathrm{E}[E]^{e_2}$$

- Policies:
  - NEVEROFF: $\alpha = 0$
  - DELAYEDOFF: $0 < \alpha < \infty$
  - INSTANTOFF: $\alpha = \infty$
- Theorem: For ERWS objective function optimal policy is either NEVEROFF or INSTANTOFF
  - Similar result in Gandhi & al. (2010b) for ERP objective function

Optimal switching on/off policy

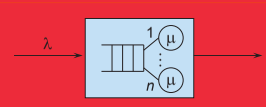
Gebrehiwot & al. (2014)

- M/G/1-FIFO
  - Setup delay *D* generally distributed with mean $1/\gamma$
- Control parameters:
  - Delayed switch-off for a gen. distributed time with mean $1/\alpha$
  - Server switched on after *k* new job arrivals
- Objective function: Gen. form

$$w_1 \mathrm{E}[T]^{t1}\,\mathrm{E}[E]^{e1} + w_2 \mathrm{E}[T]^{t2}\,\mathrm{E}[E]^{e2}$$

- Policies:
  - NEVEROFF: $\alpha = 0$
  - DELAYEDOFF: $0 < \alpha < \infty$
  - INSTANTOFF: $\alpha = \infty$
- Theorem:
  For gen. objective function optimal policy is either NEVEROFF or INSTANTOFF
  - NEVEROFF is better if P_{idle} is sufficiently small compared to P_{setup}

Part II
Multi-server queue with setup delays



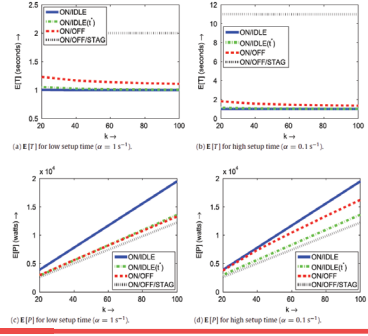
Analysis of server farms with setup delays

Gandhi & al. (2010a)

- M/M/*n*
  - Setup delay *D* exponentially distributed
- Objective function:
  Separately E[*T*] and E[*P*]
- Policies:
  - ON/IDLE = NEVEROFF
  - ON/OFF = INSTANTOFF
  - ON/OFF/STAG = INSTANTOFF with "staggered bootup"
- Mixed policy:
  - ON/IDLE(*t*)
    switching idle server off only if nr of busy and idle servers > *t*
- Conclusions:
  - "Under high loads, turning servers off can result in higher power consumption and far higher response times."
  - "As the size of the server farm is increased, the advantages of turning servers off increase."

Analysis of server farms with setup delays

Gandhi & al. (2010a)



(a) E[T] for low setup time ($\alpha = 1\,s^{-1}$). (b) E[T] for high setup time ($\alpha = 0.1\,s^{-1}$).

(c) E[P] for low setup time ($\alpha = 1\,s^{-1}$). (d) E[P] for high setup time ($\alpha = 0.1\,s^{-1}$).

Optimization of server farms with setup delay

Gandhi & al. (2010b)

- M/M/*n*
  - Setup delay deterministic
  - Additional sleep states *S* with

$$0 = P_{\text{off}} < P_{\text{sleep}} < P_{\text{idle}}$$

and deterministic (setup) delays

$$0 = d_{\text{idle}} < d_{\text{sleep}} < d_{\text{off}}$$

- Objective function: ERP

$$\mathrm{E}[T] \cdot \mathrm{E}[P]$$

- Policies:
  - NEVEROFF
  - INSTANTOFF
  - SLEEP(*S*)
  - Probabilistic and other
- Theorem:
  For $n = 1$, optimal static control is either NEVEROFF, INSTANTOFF or SLEEP(*S*)
- Robust policy:
  - DELAYEDOFF with MRB (Most Recent Busy)

Optimization of server farms with setup delay

Gandhi & al. (2010b)

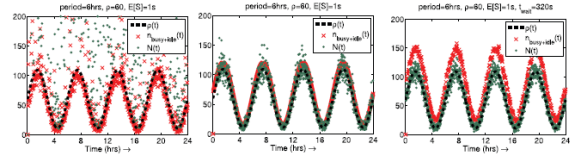
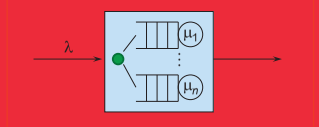


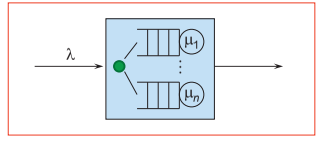
Fig. 4. Dynamic capacity provisioning capabilities of INSTANTOFF, LOOKAHEAD and DELAYEDOFF. The dashed line denotes the load at time t, $\rho(t)$, the crosses denotes the number of servers that are busy or idle at time t, $n_{\text{busy+idle}}(t)$, and the dots represent the number of jobs in the system at time t, $N(t)$.

Part III
Parallel queues with setup delays



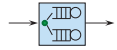
Dispatching problem

- Dispatching
  = Task assignment = Routing
  - Random job arrivals with random service requirements
  - Dispatching decision made upon the arrival



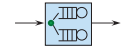
- Our setting: M/G/.
  - Poisson arrivals
  - generally distributed job sizes
  - heterogeneous servers with FIFO queueing discipline (NEVEROFF or INSTANTOFF)

Static dispatching policies



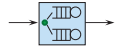
- RND = Bernoulli splitting
  - choose the queue pure randomly
  - no size nor state information needed
- SITA = Size Interval Task Assignment
  - choose the queue with similar jobs
  - based on the size of the arriving job, but no state information needed
  - Harchol-Balter et al. (1999)

MDP approach



- Any static policy (RND, SITA) results in parallel M/G/1 queues
- Fix the static policy and determine relative values for all these parallel M/G/1 queues
- Dispatch the arriving job to the queue that minimizes the mean additional costs
- As the result, you get a better dynamic dispatching policy
- This is called First Policy Iteration (FPI) in the MDP theory
- Applicable for the ERWS objective function

Relative values

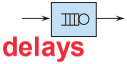


- Definition: For a fixed policy resulting in a stable system, the value function $v(x)$ gives the expected difference in the infinite horizon cumulative costs between
  - the system initially in state x, and
  - the system initially in equilibrium

- Definition: For a fixed policy resulting in a stable system, the relative value $v(x) - v(0)$ gives the expected difference in the infinite horizon cumulative costs between
  - the system initially in state x, and
  - the system initially in state 0

Size-aware M/G/1 queue without setup delays

Hyytiä et al. (2012)



- State description:

$$u = \Delta_1 + \ldots + \Delta_n$$

  - Δ_i = remaining service time of job i
  - u = backlog = unfinished work

- Mean values:

$$E[T] = E[S] + \frac{\lambda E[S^2]}{2(1-\rho)}$$

$$E[P] = (1-\rho)P_{\text{idle}} + \rho \cdot P_{\text{busy}}$$

- Result: Size-aware relative values

$$v_T(u) - v_T(0) = \frac{\lambda u^2}{2(1-\rho)}$$

$$v_P(u) - v_P(0) = u \cdot (P_{\text{busy}} - P_{\text{idle}})$$

Size-aware M/G/1 queue with setup delays

Hyytiä et al. (2014a)

- State description:

$$u = \Delta_0 + \Delta_1 + \ldots + \Delta_n$$

  - Δ_i = remaining service time of job i
  - Δ_0 = remaining setup delay
  - u = virtual backlog

- Assume: Deterministic setup delay d and

$$P_{\text{setup}} = P_{\text{busy}}$$

- Mean values:

$$E[T] = E[S] + \frac{\lambda E[S^2]}{2(1-\rho)} + \frac{d(2+\lambda d)}{2(1+\lambda d)}$$

$$E[P] = \frac{\rho+\lambda d}{1+\lambda d} \cdot P_{\text{busy}}$$

- Result: Size-aware relative values

$$v_T(u) - v_T(0) = \frac{\lambda u^2}{2(1-\rho)} - \frac{\lambda u d(2+\lambda d)}{2(1-\rho)(1+\lambda d)}$$

$$v_P(u) - v_P(0) = \frac{u}{1+\lambda d} \cdot P_{\text{busy}}$$

FPI policy

Hyytiä et al. (2012, 2014a)

- For NEVEROFF servers:

  Dispatch the job with service time x to queue i minimizing the mean additional costs:

$$a_T(u,x,i) = u + x + v_T(u+x,i) - v_T(u,i)$$

$$a_P(u,x,i) = v_P(u+x,i) - v_P(u,i)$$

- For INSTANTOFF servers:

  Dispatch the job with service time x to queue i minimizing the mean additional costs:

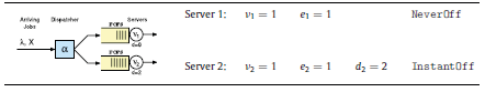
$$a_T(u,x,i) = u + x + d_i \cdot 1(u=0) + v_T(u+x+d_i \cdot 1(u=0),i) - v_T(u,i)$$

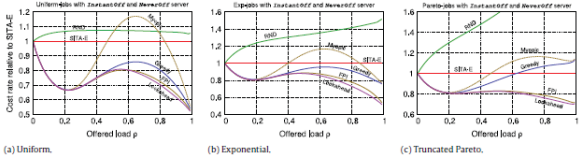
$$a_P(u,x,i) = v_P(u+x+d_i \cdot 1(u=0),i) - v_P(u,i)$$

Numerical results

Hyytiä et al. (2014a)

Table 2
Two-server system.



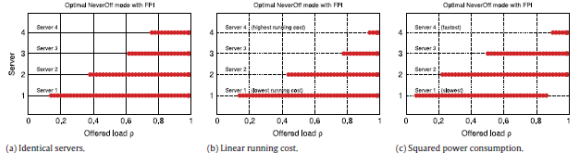


Numerical results

Hyytiä et al. (2014a)

Table 3
Four-server systems.

Parameter		(a) Identical	(b) Linear e	(c) Squared e
Service rates	$\nu_1, \ldots, \nu_4$:	1, 1, 1, 1	1, 1, 1, 1	1, 2, 3, 4
Running costs	$e_1, \ldots, e_4$:	1, 1, 1, 1	1, 2, 3, 4	1, 2, 9, 16
Switching delay	$d_1, \ldots, d_4$:	1, 1, 1, 1	1, 1, 1, 1	1, 1, 1, 1

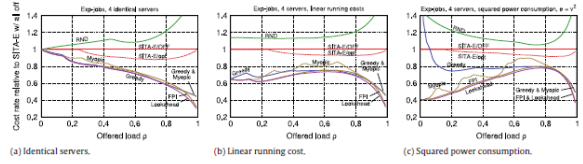


Numerical results

Hyytiä et al. (2014a)

Table 3
Four-server systems.

Parameter		(a) Identical	(b) Linear e	(c) Squared e
Service rates	$\nu_1, \ldots, \nu_4$:	1, 1, 1, 1	1, 1, 1, 1	1, 2, 3, 4
Running costs	$e_1, \ldots, e_4$:	1, 1, 1, 1	1, 2, 3, 4	1, 2, 9, 16
Switching delay	$d_1, \ldots, d_4$:	1, 1, 1, 1	1, 1, 1, 1	1, 1, 1, 1



(a) Identical servers. (b) Linear running cost. (c) Squared power consumption.

Other queueing disciplines

Hyytiä et al. (2014b)

- LIFO in the M/G/. setting with setup delays
- PS in the M/D/. setting with setup delays

- But it is another story …

References

- Harchol-Balter, Crovella & Murta (1999)
  On Choosing a Task Assignment Policy for a Distributed Server System, *JPDC*
- Wierman, Andrew & Tang (2009)
  Power-aware speed scaling in processor sharing systems, in *IEEE INFOCOM*
- Gandhi, Harchol-Balter & Adan (2010a)
  Server farms with setup costs, *PEVA*
- Gandhi, Gupta, Harchol-Balter & Kozuch (2010b)
  Optimality analysis of energy-performance trade-off for server farm management, *PEVA*
- Hyytiä, Penttinen & Aalto (2012)
  Size- and state-aware dispatching problem with queue-specific job sizes, *EJOR*
- Maccio & Down (2013)
  On optimal policies for energy-aware servers, in *MASCOTS*
- Hyytiä, Righter & Aalto (2014a)
  Task assignment in a heterogeneous server farm with switching delays and general energy-aware cost structure, to appear in *PEVA*
- Hyytiä, Righter & Aalto (2014b)
  Energy-aware job assignment in server farms with setup delays under LCFS and PS, accepted to *ITC*
- Gebrehiwot, Lassila & Aalto (2014)
  Energy-aware queueing models and controls for server farms, ongoing work

The End

LCCC ACTIVITIES IN CLOUD CONTROL

Maria Kihl, Lund University

Cloud Control is a large framework project with researchers from both LCCC and the Cloud research group at Umeå University. The main objective with the project is to solve a range of cloud management probelms with a control theoretic approach. In this presentation,

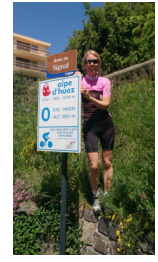
I will give an overview of some of the current work in Cloud Control within LCCC.

LCCC activities in Cloud Control

MARIA KIHL

Maria Kihl

- PhD in Teletraffic Systems, 1999
- Associate Professor at Dept. of Electrical and Information Technology since 2004.
- Internet related research projects, often with industry collaboration.
- Joint work with Dept. of Automatic Control since 2002.



Main research interests: performance modelling, analysis and control of internetworked systems.

Implementation in Networked and Embedded Systems

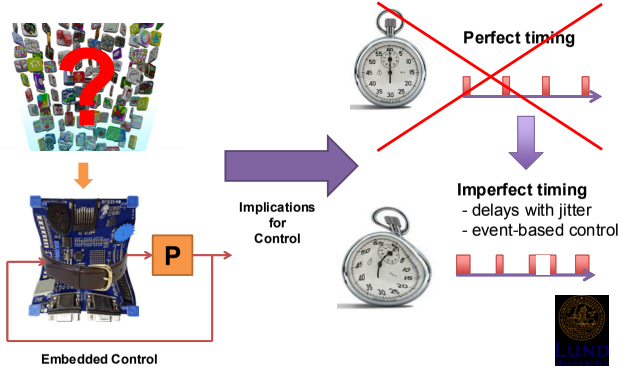
Maria Kihl
Karl Erik Årzén
Anders Robertsson
Bo Bernhardsson
Johan Eker (LU, Ericsson AB)
Martina Maggio
Anton Cervin
Alessandro Papadopoulos
Manfred Dellkrantz (PhD student)
Jonas Dürango (PhD student)
William Tärneberg (PhD student)
Yang Xu (PhD student)
Mikael Lindberg (PhD stud)
Payam Amani (PhD stud)

Background

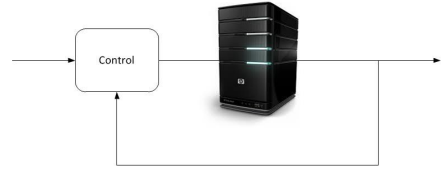
- Application complexity and uncertainty increases for both embedded system and server infrastructures.
- Strict demands on both resource efficiency (e.g. power) and service performance.

Dynamic resource management and control

Networked Embedded Control

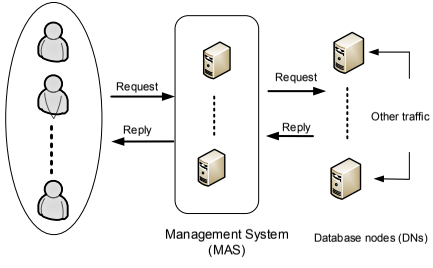


Resource optimization and control of server systems with latency constraints



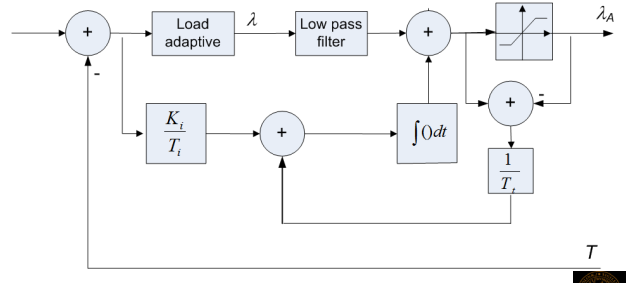
- Performance models
- Admission control
- Prediction based capacity optimization

Example: Mobile Service Support Systems developed by Ericsson AB

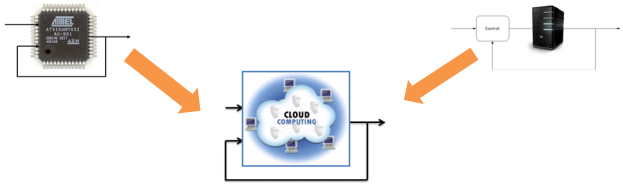


Load control of database nodes with unknown high priority background traffic.

Example: Load adaptive controller for mobile service support systems

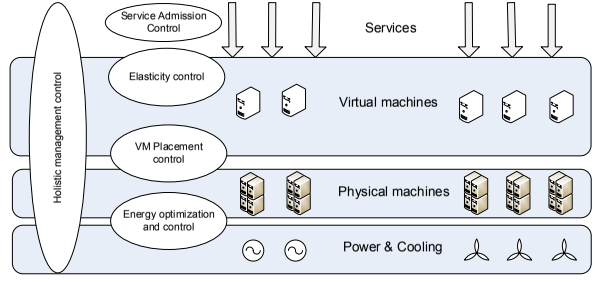


Cloud Control



- Joint work using our competences in both embedded control and server systems
- A control theoretic approach to a range of cloud management problems.

Challenges for Cloud Control



Some ongoing work

- Brownout
  - Martina Maggio, Alessandro Papadopoulos, Jonas Dürango, Manfred Dellkrantz
- VM startup time compensation
  - Manfred Dellkrantz
- Omnipresent clouds
  - William Tärneberg
- Modeling and autoscaling
  - Jonas Dürango

Brownout

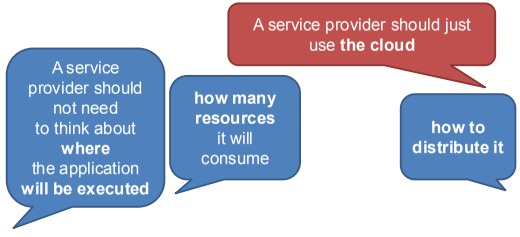
(Martina Maggio, Alessandro Papadopoulos, Jonas Dürango, Manfred Dellkrantz)

- Cloud applications need to cope with many unexpected events:
  - Flash crowds
  - Hardware failures
  - Unexpected performance degradations

Objective: Applications that **withstand variations** and have **similar behaviors** in **similar conditions**

Motivation: The need for predictability

- Tools like **cloud operating systems** and **virtualization** greatly simplify the development, management and deployment of software applications



Application performance challenge

However, in reality, another type of problems arise when an application is deployed in the cloud...

Applications do not behave always in the same way

Due to a varying amount of physical resources assigned to an application

Use resource allocation mechanisms inspired from embedded systems .

VM startup time compensation
(Manfred Dellkrantz)

Virtual machines take time to start up.

Controller saying, “Give me *m* VMs!” will have to wait for control signal to have effect.

↓

Dead time

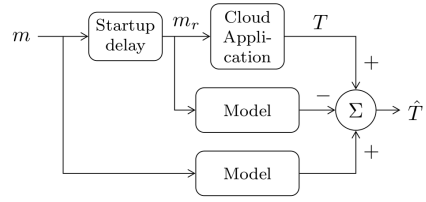
VM Startup Time Compensation

- VM startup time (“dead time” in control lingo) is a major challenge.
  - Controller reacts several times before its first action has an effect
  - Dead time-unaware controller gives overshoot (unnecessary costs)

Dead time compensation

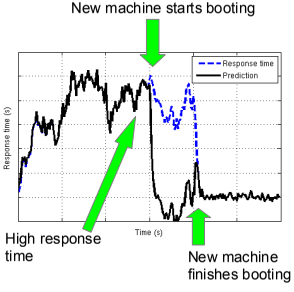
- Dead time compensation [Smith, 1957] calculates what the output would have been without dead time.
- Allows you to do faster control, still maintaining stability and avoiding oscillations.
- Requires a model!

Dead time compensator



- m – wanted # machines, m_r – actual # machines
- T – response time
- $\hat{T}$ – compensated response time

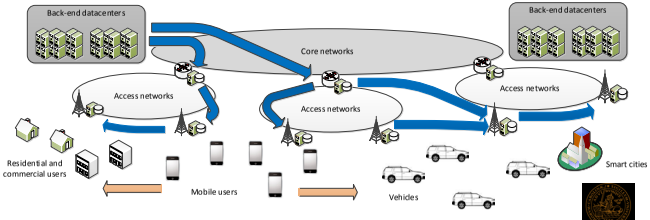
Proof-of-concept



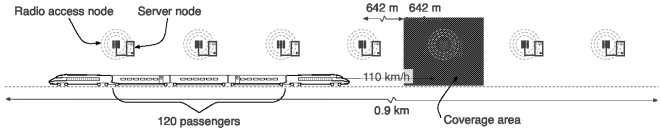
- Open-loop proof-of-concept
- Cloud infrastructure modelled as queuing system.
- Nonlinear continuous flow model [Tipper, 1990]
- Step up in number of VMs (*m*)

Omnipresent clouds
(William Tärneberg)

In the omnipresent cloud paradigm, cloud compute resources are migrated or located to the capillaries of the mobile networks.

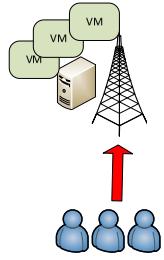


Scenario



- Cellular network corresponding to 4G/LTE.
- First mobility model corresponding to a train.
  - Easily extended to a highway with cars.
- Users request a web-like cloud service according to a stochastic process.
- VMs can be migrated between cells with a certain delay.

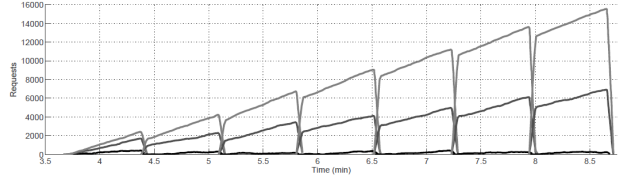
Cloud capacity moves with users



Cloud control challenges

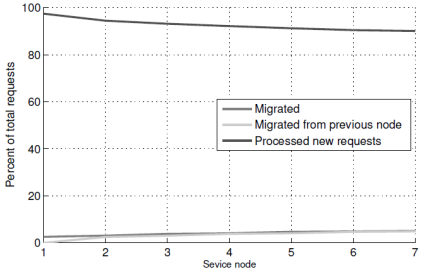
- Lack of good system models
  - Architecture models
  - Service models
  - Mobility models
  - Workload models
- Workload and Capacity demands in both time and space.
- Auto-scaling: When and Where?
- VM migration: Needs to be performed fast enough dependent on the mobility of the users.
- Heterogeneous nodes must be taken into account.

Load displacement example



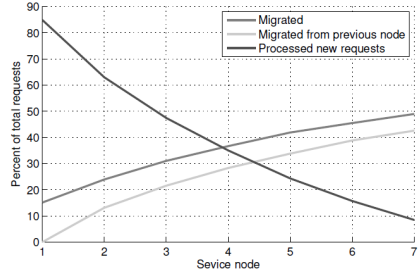
- The workload willl move between cloud servers, dependent on the mobility of the users.
- An underprovisioned system will spread in both time and space.

Performance during normal load



During normal load, the system migrates VMs according to the mobility and handles requests with low response times.

Performance during overload

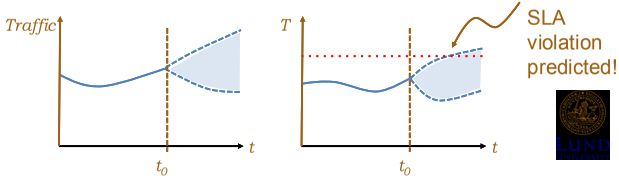


The system needs to cope with the mobility of users, or it will become overloaded. One major challenge is the delay for VM migration.

Modeling and autoscaling
(Jonas Dürango)

Computer system models suitable for control are usually tricky to find.

Understanding of dominant system dynamics is vital for decision making (Scale up? Down?)



Modeling and autoscaling

Use system identification to find models for different purposes:

- Simulation (model-based feedback control)
  - Better understanding of system dynamics -> allows better tuning of controllers.
- Prediction (proactive control)
  - Possibility to anticipate potential SLA violations

Summary

- Within LCCC, we have established a cross-disciplinary research environment with competence in control theory, real-time systems, and internetworked systems.
- Ongoing work on different aspects of Cloud Control.

Bring theory to practice

CAPACITY MANAGEMENT IN IAAS CLOUD

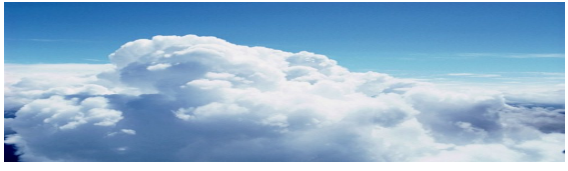
David Breitgand, IBM Research Haifa

One of the promises of elastic cloud computing is relieving its customers from capacity planning by adding just the right amount of resources just in time when elastic applications need them. While realizing this vision might indeed exempt the customer from the complex and effort consuming capacity management task, the cloud provider still needs to execute on capacity planning to strike the right balance between SLA commitments and cost efficiency. The cost efficiency is intimately related to statistical multiplexing of workloads in the cloud, allowing over-committing cloud resources. Naturally, over-committing implies risk of resource congestion.

Therefore, there is a tradeoff between improving resource utilization by increasing an over-commit ratio and exposing the infrastructure provider and customers to the risk of resource congestion. In this talk I am going to explore a number of approaches to managing this trade-off.

Capacity Management in IaaS Cloud

David Breitgand, IBM Haifa Research Lab



Based on joint works with many collaborators

Speaker Background

- Distributed Computing, Hebrew University, Israel
- Networking, Technion, Israel
- Applied Mathematics, Polytechnic University, Novosibirsk, Russia

- Technical Lead of Cloud Operating System Technology Group @IBM Research -- Haifa
- Technical Lead of Software Defined Manufacturing Group @IBM Research -- Haifa
- 10 years with IBM research, working on algorithms, performance analysis, load balancing, root cause analysis, data center optimization, SLA management, SAN performance management, Cloud computing, VM migration optimization, capacity management.

- **Overarching research theme:** studying, modeling, and optimizing tradeoffs between the costs of management and benefits accrued

Outline

- Introduction
- Problems
- Results
- Future: where does this become really hard and should we go there?
- Q&A

Cloud success == three things:

Business Model

Agility

Simplicity

Do I need [long term] Capacity Management in an IaaS Cloud?

Capacity [long term] Management on a IaaS Cloud?

If you are a **customer** of a public cloud

Capacity [long term] Management on a IaaS Cloud?

YES

If you are a **provider** of a public cloud

Why? And what [long term] "capacity management" means?

- IaaS commoditize → IaaS providers operate under perfect competition
- Provider needs to be competitive → Pressure to lower prices
- Pressure to lower prices → Pressure "to do more with less"

- Capacity management is required to achieve that

- The most obvious link to this workshop: cost efficient elasticity

How to do more with less?

- By increasing *over-commit ratio* to improve *cost efficiency*

- But if I over-commit too much, I will *degrade* cost-efficiency, won't I?

What does "cost-efficiency" means?

- Cost efficiency [first usage in 1970]
  - "Cost effective describes something that is of a good value, where the benefits and usage are worth at least what is paid for them"
- Cost efficiency is a relative concept describing relationship between ***at least*** two options
- Example 1:
  - Use of PSTN line versus VoIP for long distance call
- Example 2:
  - Static resource provisioning versus on-demand provisioning from shared multiplexed pool

The Gist of "capacity management" for provider

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it
- Service in IaaS:
  - Provide VM, storage, networks (i.e., resource collections)
- Trade-off:
  - Risk of resource congestion vs. cost of using successfully acquired resources

There is a problem, though

- How do you know the value a customer assigns to using the service???

If you expected an answer…

Second Price Auctions?

- Vickrey-Clark-Groves
- Heterogeneous goods
- Dominant strategy is to report the true value for the goods

- Beautiful game-theoretic mechanism
- Rarely used in practice
- Complex implementation
- Close to zero seller revenue

Zaman, Sharrukh, "Combinatorial Auction-Based Virtual Machine Provisioning And Allocation In Clouds" (2013). *Wayne State University Dissertations*. Paper 720. Recent Ph.D. thesis in designing Combinatorial Auctions for VM allocation

So, we have the input problem

- A provider optimizing capacity has a problem with the quality of the input

- Capacity is planned in practice solely based on the observed historical demand, error rate, and a forecast

- Forecast quality quickly deteriorates with:
  - Prediction horizon going further into the future
  - Historic data going back into the past

- Performance of forecasting often depends on fine tuning

SLAs

- A mechanism used absence of the real input on consumer valuation of the service
- Essentially a guess
- Essentially a declaration of intentions
- Perceived as due diligence by customers seeking to avoid risk
- Essentially a means to differentiate and compete under perfect competition conditions

Do we have SLAs today in an IaaS Cloud?

rackspace HOSTING:

"Finally, Real SLAs for Cloud Computing
The SLA adopted for Cloud Servers™ is just a[s] aggressive as the one Rackspace provides for traditional hos[ted] servers. It provides remedies for any downtime event caused by the network, data center infrastructure, the physical host server, or the migration of

607 words

amazon web services:

AWS will use commercially reasonable efforts to make [Amazon EC2 available with an Annual Uptime Percentage (defined below) of at least **99.95%** during the Service Year. In the event Amazon EC2 does not meet the Annual Uptime Percentage commitment, you will be eligible to receive a Service Credit as described below.

1500 words

GOGRID

10,000% Guaranteed, **100% Uptime** Service Level Agreement

supplements the Terms of Service and together such documents, and other [documents referenced] in the Terms of Service, form a binding agreement (the "**Agreement**") between [you and] **GoGrid** and Customer".
....

3700 words

"A "10,000% Service Credit" is a credit equivalent to **one hundred (100) times Customer's fees** for the impacted Service feature for the duration of the Failure. (For example, where applicable: a Failure lasting seven (7) hours would result in credit of seven hundred (700) hours of free service for the feature in question"

What do we **really** get? (assorted examples)

rackspace HOSTING

amazon web services

"Annu[al Uptime Percentage] of the pe[riod] Amaz[on EC2] **been** **is stil[l]** **the se[rvice]**

"We guaranty that **our data center network** will be available **100%** of the time in any given monthly billing period, excluding scheduled maintenance.

We guaranty that **data center HVAC and power** will be functioning **100%** of the time in any given monthly billing period, excluding scheduled maintenance. Infrastructure downtime exists when Cloud Servers downtime occurs as a result of power or heat problems.

We guaranty the functio[ning] hypervisor. If a cloud complete **within**

GOGRID

"The **minimum period of Failure eligible for a credit is 15 minutes**, and **shorter periods will not be aggregated**. The **maximum credit for any single Failure is one month's Service fees**. In the event that multiple periods of Failure overlap in time, credits will not be aggregated, and Customer **will receive credit only for the longest such period of Failure**".

Current Cloud SLA Practices Summary

- Inflated promises
- Standard SLA with "one size fits all" availability SLO
- Obfuscation of provider commitments:
  - Being non-specific about maximum maintenance time per billing period
  - Being non-specific about maintenance head-up warning
  - Unavailability periods aggregation trickery
  - Requiring customers to understand failures on the vendor side
  - Being non-specific about availability tests to verify SLA compliance
  - Placing the onus of damage proof on the customer
- Refund policies that do not compensate the loss of value to the business due to unavailability

Impact of downtime on business

- Loss of profits
- Impact on stock price
- Loss of cash flow from debtors
- Loss of customers (lifetime value of each)
- Market share loss of product
- Cost of fixing / replacing equipment
- Cost of fixing / replacing software
- Salaries paid to staff unable to undertake productive work
- Salaries paid to staff to recover work backlog and maintain deadlines
- Cost of re-creation and recovery of lost data
- Interest value on deferred billings
- Additional cost of credit through reduced credit rating
- Fines and penalties for non-compliance
- Liability claims
- Additional cost of advertising, PR and marketing to reassure customers and prospects to retain market share
- Additional cost of working; administrative costs; travel, etc.
- Cloud: "we will give you a free service next month"

To remind us about our problem

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it

- SLAs are an approximation of the customer valuation (being arbitrarily far from the true consumer valuation)

- **Takeaway:** SLA construction is intimately related to capacity management
- Short term (e.g., feedback loop control) resource management is done w.r.t. SLAs that might be suboptimal

- **In this talk: we don't discuss deriving optimal SLAs**
- The focus will be on an easier problem: make SLAs **more specific** and optimize capacity w.r.t. them

…and there is also another input problem

- Find minimal physical resources configuration to provide service to the customer, where the value of using the service worthy at least of what the customer pays for it

- Environment is very dynamic
- There are long running services and short running services, tenants come and go, hardware changes, failures happen, disasters happen…
- The environment changes as capacity optimization cycle completes…

…and, yeah, there is one more "small" input problem

- Administrators don't like your tool monitoring their production environment:
  - Too much storage overhead
  - Too much network overhead
  - Too much disturbance to services normal operation

IBM

So….

- No input on true customer valuations
- No stable VM populations
- More often than not no meaningful SLAs
- Scarce resources for capacity management

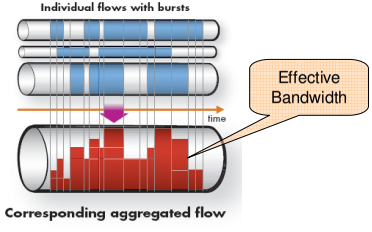
IBM

Before moving forward with theory: **a disclaimer**

- **A distance between theory and practice is shorter in theory than in practice**
- Human administrator is the bottleneck for anything “smart”
- **In practice:** capacity management helps **slowing down** procurement cycle, but nobody [that I know] is looking for squeezing the last drop of optimality and/or accuracy

IBM

Cloud is cost-efficient thanks to Statistical Multiplexing



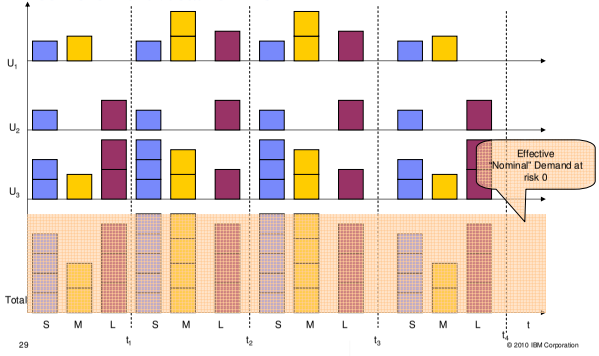
Over-commit: the total capacity of the shared resource is allowed to be *much* smaller than the maximum total demand

IBM

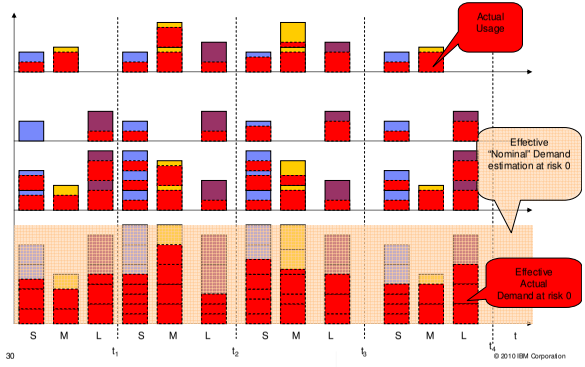
Resource Demand

- Can be expressed
  - In terms of “nominal allocations”
  - In terms of “actual utilization”

Over-Commit in IaaS: "Pool Level": birth/death of VMs



Over-commit in IaaS: "Host Level": actual utilization per VM



Highlights

Over-commit on Nominal Demand

- David Breitgand, Zvi Dubizky, Alex Glikson, Amir Epstein, Inbar Shapira, "SLA-aware Resource Over-Commit in an IaaS Cloud", CNSM'12, Las Vegas, USA

Over-Commit on Actual Demand

- David Breitgand and Amir Epstein, "Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds", INFOCOM'12, March 25-30.

Challenges in Respecting Nominal Allocations cost-efficiently

- No statistics about actual usage of a VM are available
- High dynamics with VMs arriving and departing: no constant VM population
- A resource should be provided as an indivisible "whole" (e.g., bare metal cloud server)

Standard SLA clause

- Standard availability clause in IaaS SLA:
Percentile of the billing period when **pinging** VM succeeds

p_a

Extended Availability SLA

- Assume virtual hardware discrete types $a_1, a_2, \ldots, a_n$
- For simplicity, assume a single elasticity range for all services: R = [R_{min}, R_{max}]
- Each service S_i is guaranteed to successfully assume configuration Gi = < $a_1, a_2, \ldots, a_n$ > subject to <elasticity range>, with probability <p> computed over <billing period>
- $p \leq p_a$, where p_a is the standard availability clause probability
- Rationale: guarantee on assuming desired configuration
- Interpretation: guarantee on probability to launch a VM of any type, subject to elasticity range

SLA-aware cloud over-commit (CNSM'12)

- Let ***n*** be the total number of workloads (services)
- Let ***i*** be number of VM discrete types
- Let $Y_{i,j}$ be the random variable representing number of VM instances of type ***i*** used by workload ***j***
- $X_i = \sum_{j=1}^{n} Y_{j,i}$ is the total number of VM instances of type *i* in the cloud
- Definition 1: **Nominal Demand** is $\overline{X} = (X_1, X_2, \ldots, X_l)$
- Definition 2: **Effective Nominal Demand**
  - Minimal vector $\overline{D}$, such that

$$Pr(\bigwedge_i (X_i \leq D_i)) \geq p$$

- Union bound: $Pr(\vee_i (X_i > D_i)) \leq \sum_{i=1}^{l} Pr(X_i > D_i)$

p_i

Critical observation

- We do not know distributions of $Y_{i,j}$
- Small contributions to X_i
- For large clouds effect of dependencies diminishes
- Can be treated as independent
- Central Limit Theorem: X_i asymptotically converges to normal distribution
- Should also be identically distributed, but we ignore that in this work

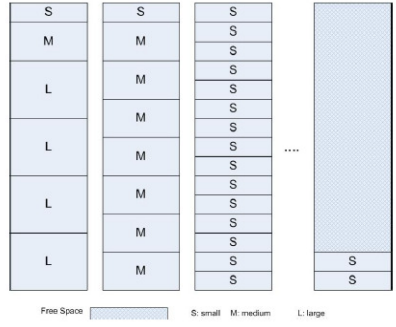
Not so fast ☺

- The variables are discrete
- Normal distribution is truncated normal

$$D_i = \lceil \mu_i' + Z_{p_i}\sigma_i' \rceil$$

$$\mu_i' = \mu_i + \sigma_i \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1-\Phi(-\frac{\mu_i}{\sigma_i})} \quad \sigma_i' = \sigma_i \sqrt{1 - \frac{\phi(-\frac{\mu_i}{\sigma_i})}{1-\Phi(-\frac{\mu_i}{\sigma_i})}\left(\frac{\phi(-\frac{\mu_i}{\sigma_i})}{1-\Phi(-\frac{\mu_i}{\sigma_i})} + \frac{\mu_i}{\sigma_i}\right)}$$

Illustration: Sample VM slots reservation



Main Virtues

- Computationally and storage efficient
- Extremely simple
- Simple placement
- Explicit calculation of capacity based on risk perceived as tolerable → easy to transform into a policy
- Fast: easy to do "what-if?" analysis to extract **providers'** true valuations of resource congestion

- **A framework is more important than a specific algorithm**

Things missed by nominal strategy

- Nominal capacity allocation strategy treats VMs/resources as indivisible "wholes"
- When should we expect it to produce best results?
- **Gives best results when most of the time actual resource utilization of most of VMs on all of resource types is close to nominal discrete configuration of the VMs**

- **If this is not the case, nominal strategy protects quality of experience (performance), but might be wasteful on resources**

IBM

Actual Utilization Over-Commit Strategy to the rescue!

- David Breitgand and Amir Epstein, "Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds", INFOCOM'12, March 25-30.

IBM

Stochastic Bin Packing Problem (SBP)

- $S=\{X_1,\ldots, X_n\}$ – Set of items
- X_i – random variable representing the size (bandwidth demand) of item i
- p – overflow probability
- Goal: Partition the set S into the smallest number of subsets (bins) $S_1,\ldots,S_k$ such that

$$\Pr[\sum_{i:X_i \in S_j} X_i > 1] \le p \quad \text{for} \quad 1 \le j \le k$$

p represents an SLA-stipulated value

IBM

Related Work – Bin Packing

- The problem is NP-hard
- Bin packing is hard to approximate to a factor better than 3/2 unless P=NP.
- First Fit Decreasing (FFD) has asymptotic approximation ratio of 11/9 and (absolute) approximation ratio of 3/2.
- MFFD algorithm has asymptotic approximation ratio of 71/60.
- AFPTAS exists.
- Online bin packing
  - First Fit (FF) has competitive ratio of 17/10.
  - Best upper and lower bounds are 1.58899 and 1.54014, respectively.

IBM

Related Work – Stochastic Bin Packing

- $O\left(\sqrt{\frac{\log p^{-1}}{\log\log p^{-1}}}\right)$ -approximation for SBP with Bernoulli variables [Kleinberg et. al 1997]
- SBP with Poisson, Exponential and Bernoulli variables [Goel and Indik 1999]
  - PTAS exists for Poisson and exponential distributions.
  - Quasi-PTAS exists for Bernoulli variables.
  - These results relax bin capacity and overflow probability constraints by a factor $1+\varepsilon$.
- $(1+\sqrt{2})(1+\varepsilon)$ - competitive algorithm for SBP with normal variables [Wang et. al 2011]

Our Results (Breitgand and Epstein INFOCOM'12)

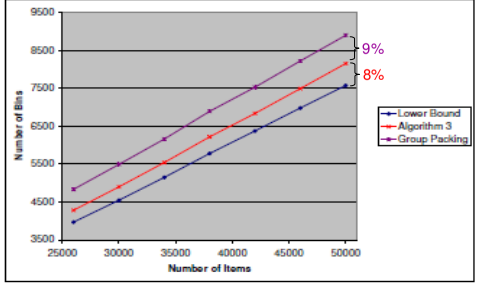
- 2-approximation algorithm for SBP with normal variables
- (2+ε)-competitive algorithm for online SBP with normal variables
  - Best known

Intuition

- Collocating "bursty" items (VMs) together **reduces** effective size
- Normality assumption: relatively small number of VMs per host
- For large hosts (e.g., large IBM Power machines), normality assumption can be dropped

Online Algorithms

- Large synthetic instances based on scaled real workloads



Real Instance

p	Algorithm 3 (Online)	Algorithm 1 (Approx.)	Group Packing	FFD	FF	Algorithm 2 (L.B)
0.1	164	146	595	332	334	144
0.01	215	195	785	519	522	192
0.001	263	243	881	656	662	237

Summary of the methodology

- Make SLA meaningful – starting point
- Calculate effective capacity with respect to observed demand and target resource congestion probability
- Calculate placement for effective capacity
- Continuously update effective capacity
- Recalculate placement only when a significant (affecting target congestion probability) is spotted

Conclusions & Future Research: where does this become really hard?

- Placement Constraints: cannot use simple BP anymore
  - David Breitgand and Amir Epstein, "SLA-aware placement of multi-virtual machine elastic services in compute clouds", IFIP/IEEE Integrated Network Management (IM'11), pp. 161-168, Dublin, Ireland
  - Extends: B. Urgaonkar, A. L. Rosenberg, and P. J. Shenoy, "Application placement on a cluster of servers," *Int. J. Found. Comput. Sci.*, vol. 18, no. 5, pp. 1023–1041, 2007.
  - Elastic Services Placement Problem (ESPP) generalizes GAP, but does not admit constant factor approximation $m^{1/2-\epsilon}$ for any constant $\epsilon > 0$.
- Performance degradation due to non-virtualized resources
  - L2 cache? Bus? Not visible metrics
- **Personal appreciation 1**: these two problems are the core ones impeding progress on **systematically** improving cost-efficiency in IaaS
- **Personal appreciation 2**: solution is likely in apps collaborating with infrastructure provider

DEADLINE SCHEDULING FOR BIG-DATA JOBS AND FAULT-TOLERANCE OF DATACENTER APPLICATIONS

Peter Bodik, Microsoft Research

I will describe two projects in the space of resource allocation in large-scale datacenters: Jockey -- scheduling big-data jobs to meet latency deadlines and application placement in datacenters to survive large-scale hardware failures.

Many big-data jobs, running in Hadoop MapReduce or Microsoft's Cosmos, require completion by a certain deadline. Missing a deadline might lead to reduced productivity of data analysts, stale content presented by a search engine, or even a financial penalty. However, today's cluster schedulers do not support specifying a deadline for a job and provide no guarantees on job completion. I will describe Jockey, a framework for providing deadline guarantees for big-data jobs. Offline, Jockey uses past executions of a job to build a model of the job and then, during job execution, uses the model in a control loop to adjust job resources to meet the specified deadline.

In the second half of the talk, I will talk about improving fault tolerance of applications deployed in datacenters. Datacenter networks have been designed to tolerate failures of network equipment and provide sufficient bandwidth. In practice, however, failures and maintenance of networking and power equipment often make tens to thousands of servers unavailable, and network congestion can increase service latency. Unfortunately, there exists an inherent tradeoff between achieving high fault tolerance for applications deployed in a datacenter and reducing bandwidth usage in network core. Spreading servers across fault domains improves fault tolerance, but requires additional bandwidth, while deploying servers together reduces bandwidth usage, but also decreases fault tolerance. We present a detailed analysis of a large-scale Web application and its communication patterns. Based on that, we propose and evaluate a novel optimization framework that achieves both high fault tolerance and significantly reduces bandwidth usage in the network core by exploiting the skewness in the observed communication patterns.

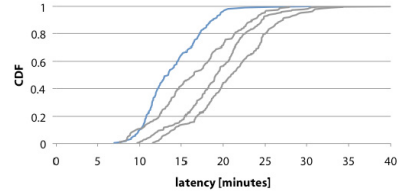
Big-data job deadlines
+
Fault-tolerant resource allocation

Peter Bodik
Microsoft Research

Why care about deadlines for big-data jobs?

Important big-data jobs have to finish on time
- missed deadline = delayed updates on site, financial penalty, productivity loss

Current clusters
- can't specify deadline in current schedulers
- users don't know how resources map to latency
- noise



Jockey: meeting deadlines for big-data jobs

Cosmos 101
- big-data platform in Microsoft
- job = SQL query + user C# code
- job compiled/optimized using SQL-like optimizer to a DAG of stages/vertices
- big jobs have 100s of stages, 1M vertices

Jockey
- input: single job with a deadline, past job runs
- offline: builds a job model
- run time: control loop adjusts allocation

Job model = past job runs + simulation

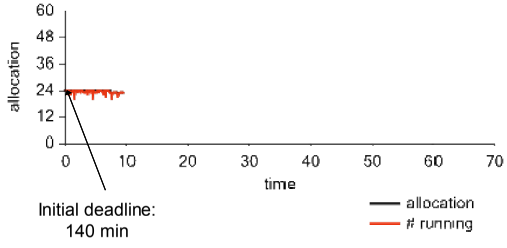
Job model
- input: current progress, allocation
- output: remaining time to completion

Example
- deadline = 30 min
- after 10 min, completed 50%
- will set allocation to 30 tokens

	10 tokens	20 tokens	30 tokens
10%	60 min	40 min	25 min
20%	59 min	39 min	24 min
30%	58 min	37 min	22 min
40%	56 min	36 min	21 min
50%	54 min	34 min	20 min

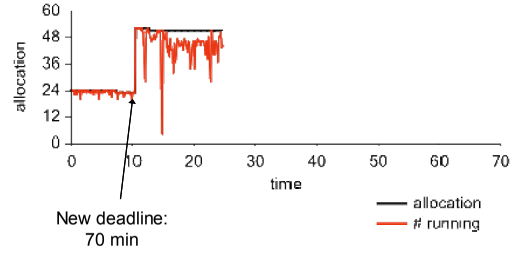
Issues
- in practice need to trade off between many jobs

Jockey in Action



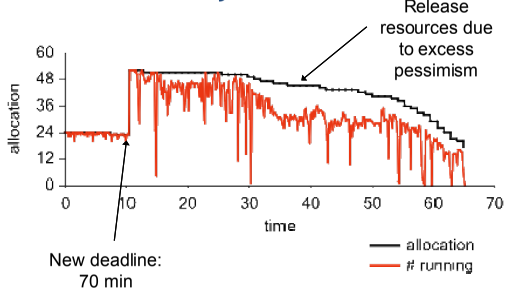
5

Jockey in Action



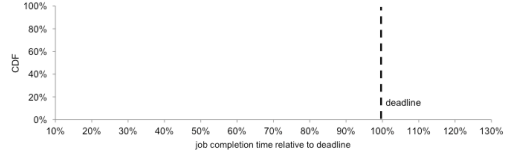
6

Jockey in Action



7

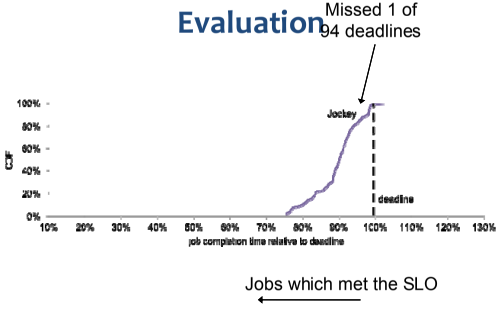
Evaluation



Jobs which met the SLO

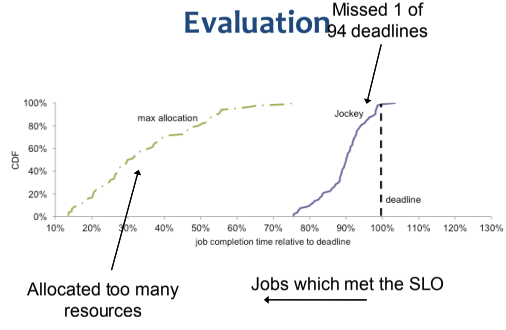
8

Evaluation



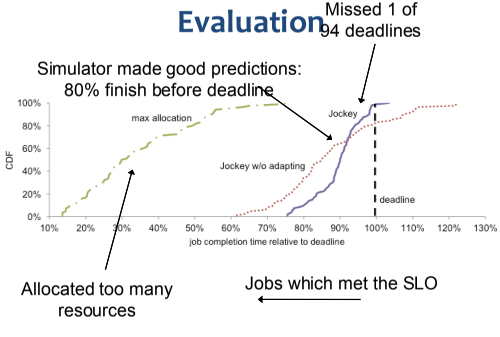
9

Evaluation



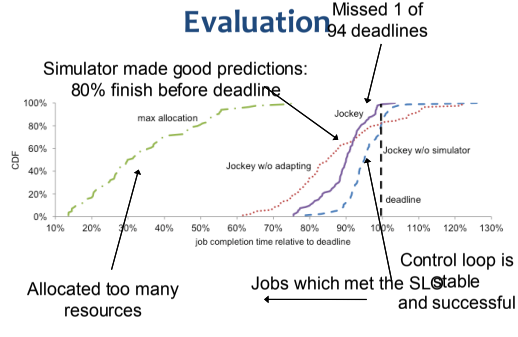
10

Evaluation



11

Evaluation



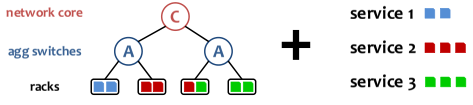
12

What's missing?

- multiple jobs/deadlines, multiple pipelines
- better representation of job state

FAULT-TOLERANT RESOURCE ALLOCATION

How to allocate services to physical machines?



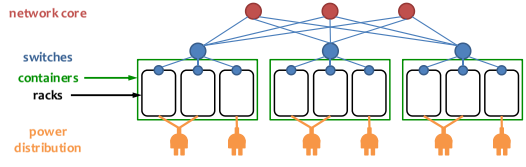
Three important metrics considered together

- FT: service fault tolerance
- BW: bandwidth usage
- #M: # machine moves to reach target allocation

SIGCOMM 2012, Surviving Failures in Bandwidth-Constrained Datacenters
Peter Bodik, Ishai Menache, Mosharaf Chowdhury, Pradeepkumar Mani, David A. Maltz, and Ion Stoica

15

FT: Improving fault tolerance of software services



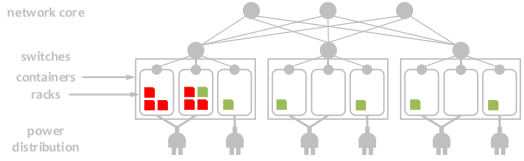
Complex fault domains: networking, power, cooling

Worst-case survival = fraction of service available during single worst-case failure

- corresponds to service throughput during failure

16

FT: Service allocation impacts worst-case survival

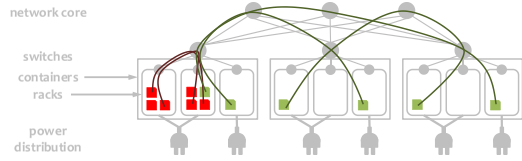


Worst-case survival:

- red service: 0% – same container, power
- green service: 67% – different containers, power

17

BW: Reduce bandwidth usage on constrained links



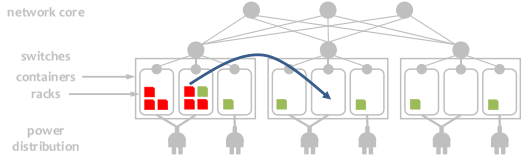
BW = bandwidth usage in the core

Goal

- reduce cost of infrastructure
- consider other service location constraints

18

#M: Need incremental allocation algorithms

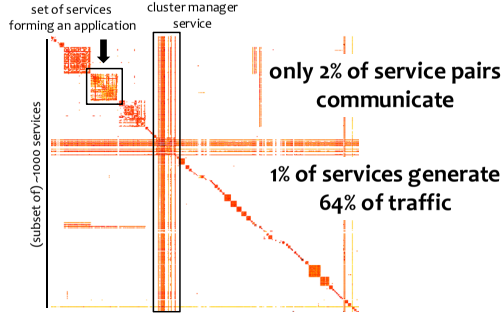


High cost of machine move

- need to deploy potentially TB of data
- warm up caches
- could take tens of min, impact network

19

Service communication matrix is very sparse and skewed



20

Formulate as convex optimization

Spread machines across all fault domains

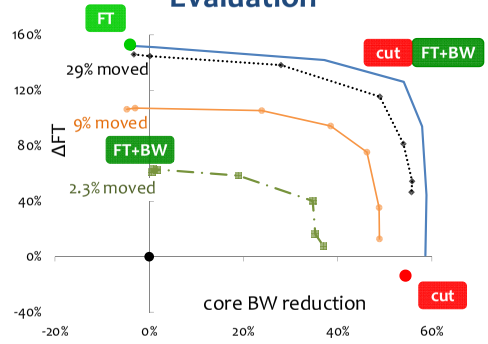
$$\min \alpha BW + \sum_{s} c_s \sum_{f} w_f \cdot z_{s,f}^2$$

- c_s: service weight
- w_f: fault domain weight
- $z_{s,f}$: number of machines of service s in domain f

Advantages of convex cost function

- local actions (machine swaps) lead to improvement of global metric
- directly considers #M

Evaluation



Potential future work

Deadline scheduling

- multiple deadline jobs (with different penalties for missing deadline)
- dependencies across jobs
- maximize total *utility*
- adapt to new jobs arriving, reduces capacity, …

Resource allocation

- consider *structure* of the applications, eg, data partitions and replications
- dependencies across services

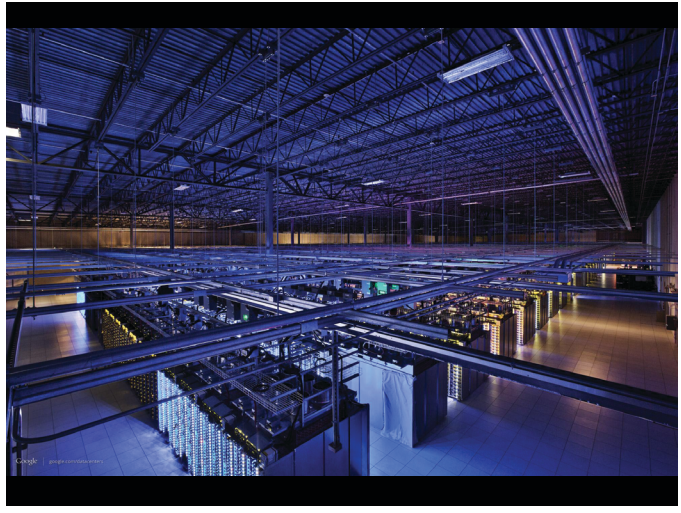
CONTROL ISSUES IN WAREHOUSE-SCALE DATACENTERS

John Wilkes, Google

Google's compute and storage clusters are managed by a raft of different software systems that attempt to achieve multiple, partially conflicting, goals simultaneously. I will present an overview of some of these systems, and highlight some of the challenges that we're facing along the way, on the grounds that many challenges also represent good research opportunities.

Control issues in warehouse-scale datacenters

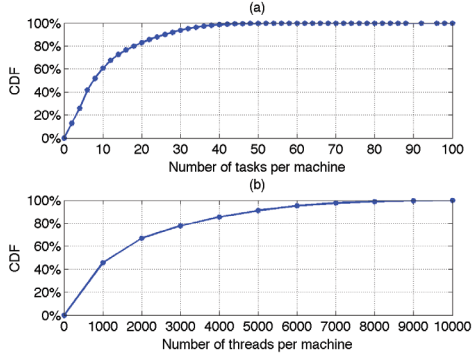
john wilkes
Cloud Control Workshop, Lund, Sweden
May 2014



The problem

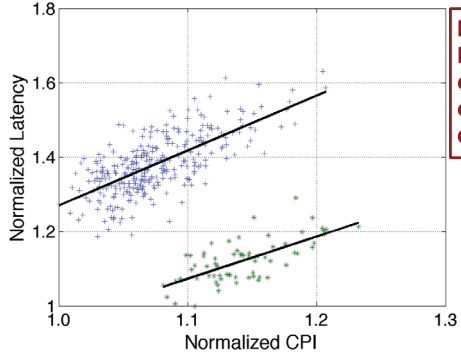
From: **CPI²: CPU performance isolation for shared compute clusters**. EuroSys'13.

high utilization => resource sharing



The problem

resource sharing => interference



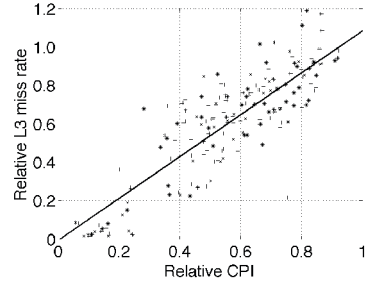
Interference happens tens of thousands of times per day

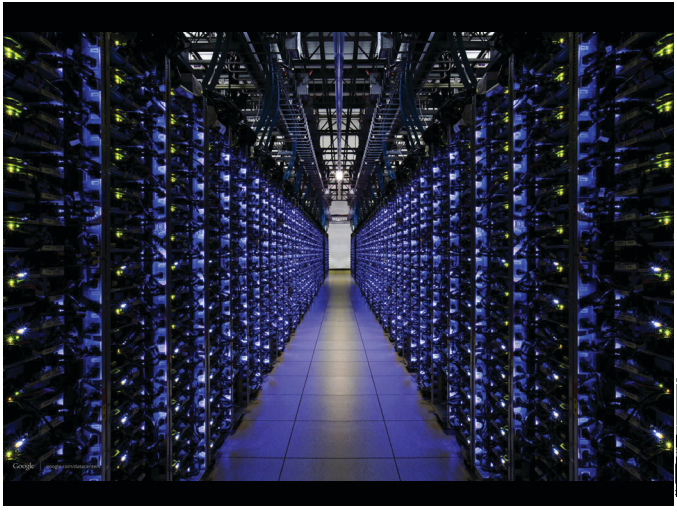
Our solution: CPI^2
a simple control system

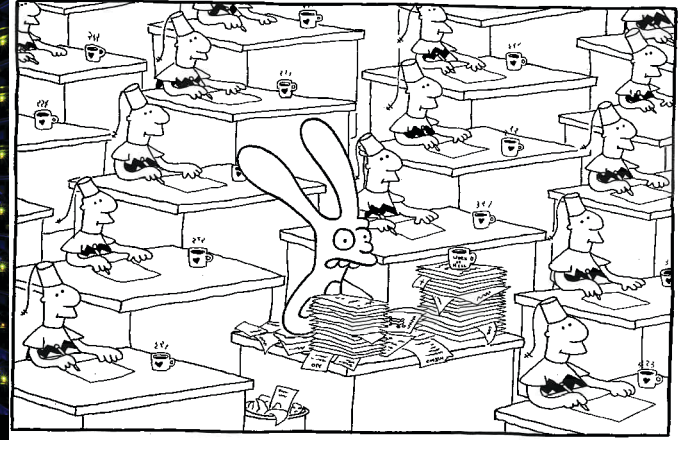
1. Monitor *Cycles Per Instruction* (**CPI**)
2. Learn anomalous behaviors
3. Identify a likely antagonist
4. Throttle it to shield victims

Why use CPI?

- It's cheap: < 0.1% CPU overhead, invisible to users
- It's stable (across time and space)
- It correlates well with L3 cache miss rate



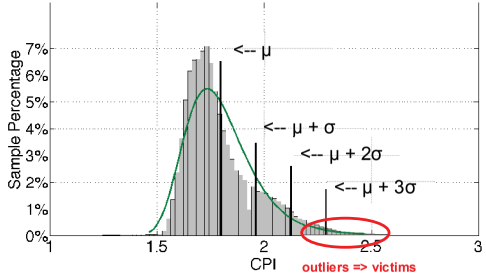




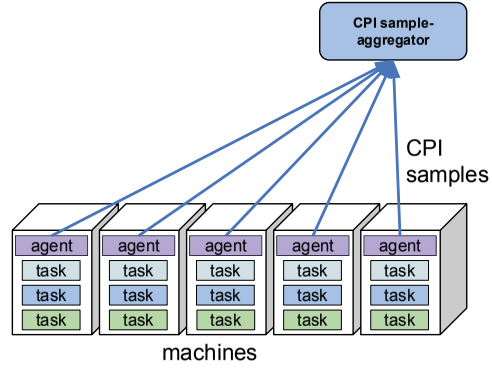
Gathering CPI

Build a CPI profile for a job

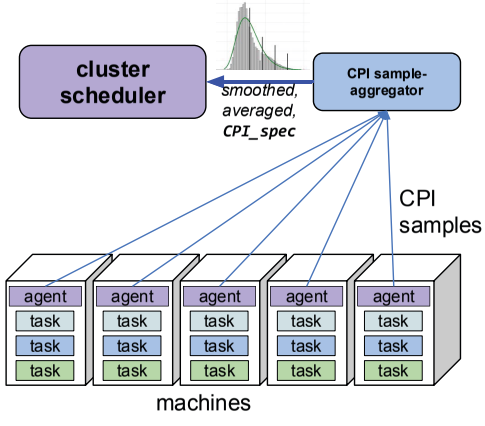
- per-cluster, per-platform
- mean (μ) & stddev (σ)



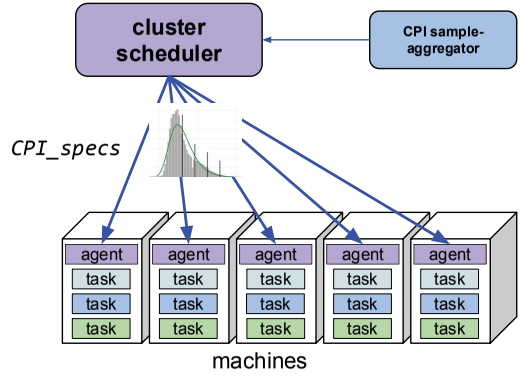
Gathering CPI



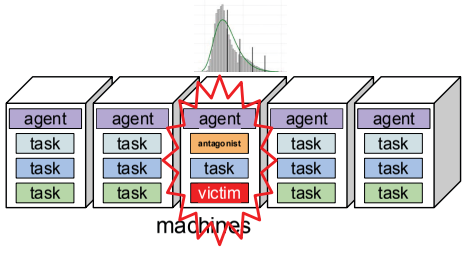
Gathering CPI



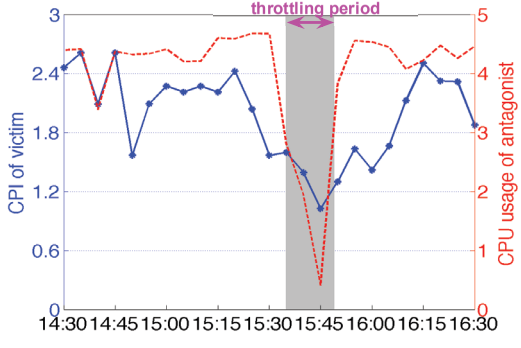
Using CPI to detect an anomaly



Using CPI to detect an anomaly



A motivating example



Now what?

Goal: reduce the effect of the antagonist

Let's **throttle** the antagonist!

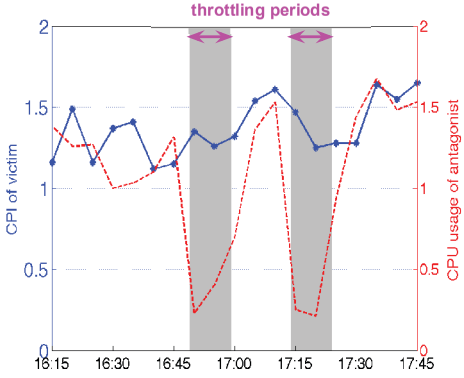
- CPU hard-capping: 0.1 core for 5 minutes

Restrictions:

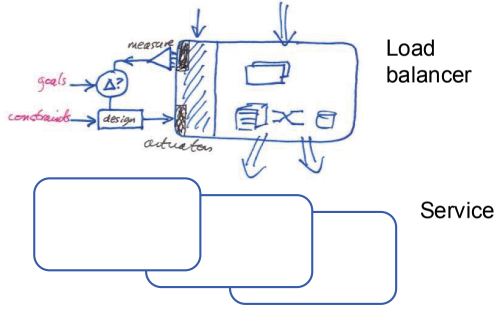
- only throttle batch jobs
- only help "important" victims

What could *possibly* go wrong?

A not so good example



Maybe batch-only was a bad idea?
After all: LS tasks have *load balancing*



Maybe batch-only was a bad idea?
After all: LS tasks have *load balancing*

A control system to achieve:

- failure tolerance (of server, of cluster)
- equal load (e.g., qps)
- equal performance (e.g., latency)

Overload
What does your system do?

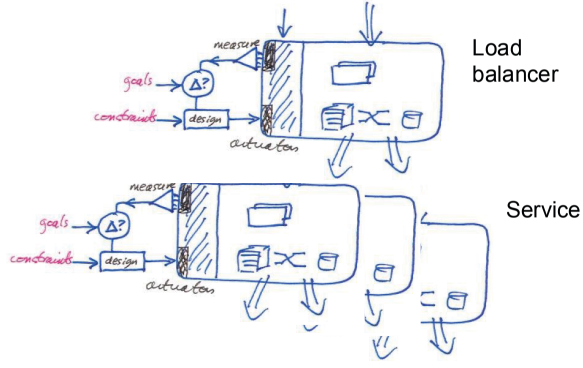
Tip: don't send all traffic to the first place on your list

Maybe batch-only was a bad idea?
After all: LS tasks have *load balancing*

Cascading failures

1. Overload-induced outage
   - busy cluster => oops
2. No worries! Shunt load elsewhere!
   - busy cluster => much oops (repeat)
   - e.g., Gmail outage, 2009-02-24

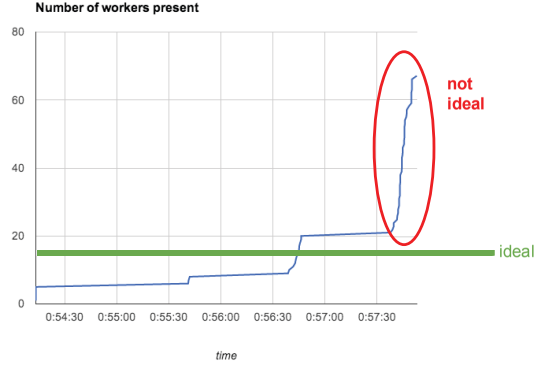
Maybe batch-only was a bad idea?
After all: LS tasks have *load balancing*



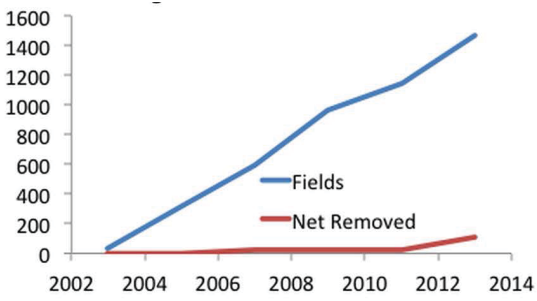
Interacting control loops

1. Load-placement
   - few-second response times
2. Number-of-workers
   - few tens-of-seconds response times
3. Add a little signalling delay ...

Auto-scaling to meet a job deadline



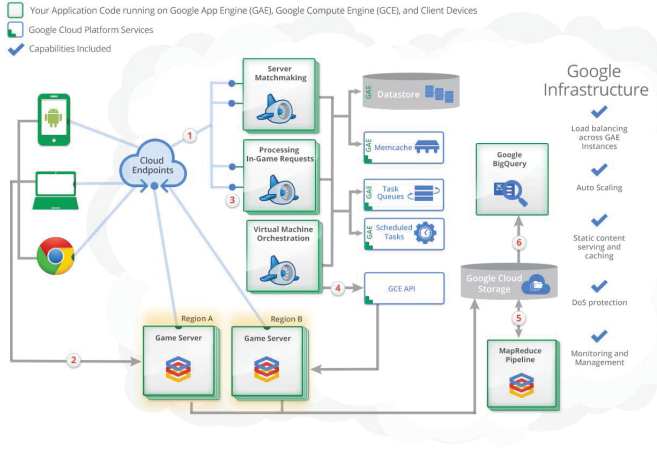
No worries!
Just add a few more knobs ...



Upload malformed configuration
What does your system do?

Tip: don't just stop working

Dedicated Server Gaming Solution on the Google Cloud Platform



GMail circa 2008

your browser

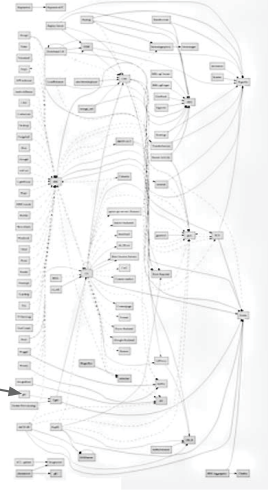
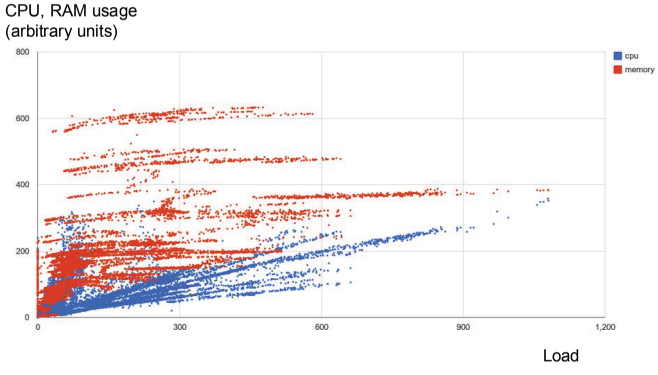


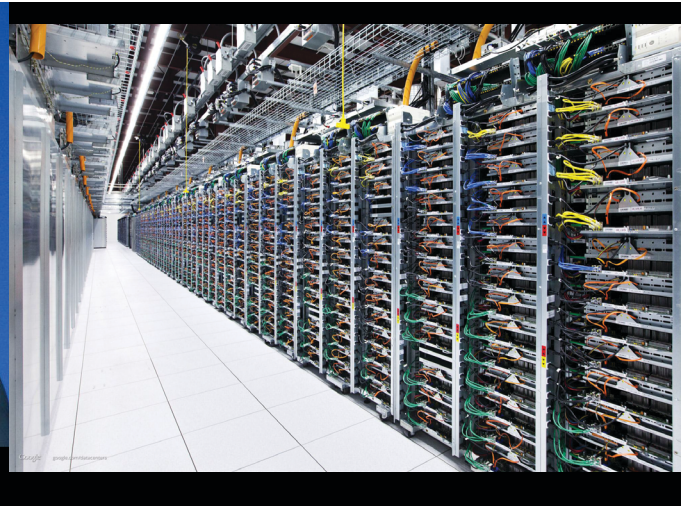
Image source: Hareesh Nagarajan

Model building is hard



Is it doing what it should be doing?

Maybe **more monitoring** would help?



It's 3am and your pager goes off

-- are we in trouble?

-- are we about to get into trouble?

➔ **what should you do about it?**

Delegation is hard

be careful what you ask for

Summary

Control systems do not run in isolation

1. Do no harm
2. Make things better
3. **Assume the world is out to get you**

   "any sufficiently advanced incompetence is indistinguishable from malice"

   -- Grey's Law

APPLICATION PERFORMANCE MANAGEMENT IN THE CLOUD USING LEARNING, OPTIMIZATION, AND CONTROL

Xiaoyun Zhu, VMware Inc.

Many businesses and organizations are increasingly relying on cloud based infrastructures and platforms to deliver their business-critical applications. In the meantime, a recent study shows that 79% of companies are concerned about the hidden costs of cloud services for their applications, citing "poor end-user experience due to performance bottlenecks" as their top management concern in relationship to cloud services.Existing practices in application performance management rely heavily on white-box modeling or heuristics-based, manual diagnostic approaches to find potential bottlenecks and remediation steps. However, the scalability and adaptivity of such approaches remain severely constrained, especially in a highly-dynamic, consolidated cloud environment. These challenges present unique opportunities in applying statistical learning, control, and optimization based techniques to developing model-based, automated application performance management frameworks. There has been a large body of research in this area in the last several years, but many problems remain. In this talk, I'll highlight some of the performance and resource management techniques we have developed within VMware, along with related technical challenges, and discuss open research problems, in hope to attract more innovative ideas and solutions from a larger community of researchers and developers.

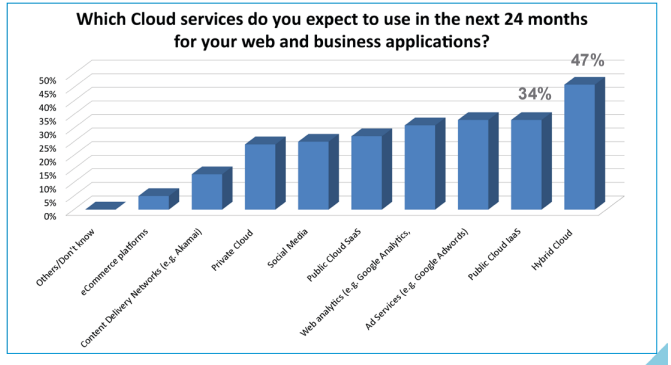
Application Performance Management in the Cloud using Learning, Optimization, and Control

Xiaoyun Zhu

May 9, 2014

vmware

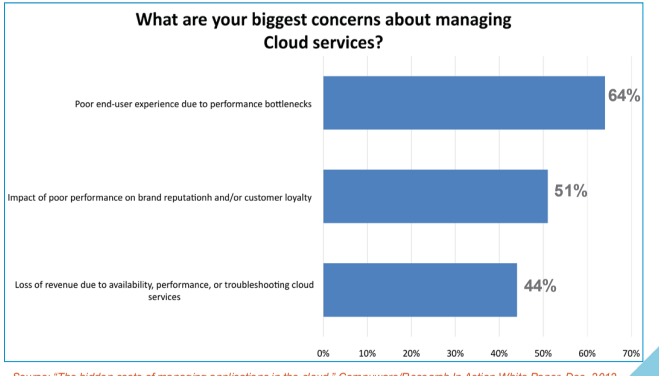
Rising adoption of cloud-based services



Source: "The hidden costs of managing applications in the cloud," Compuware/Research In Action White Paper, Dec. 2012, based on survey results from 468 CIOs in Americas, Europe, and Asia.

vmware

Application performance – a real concern



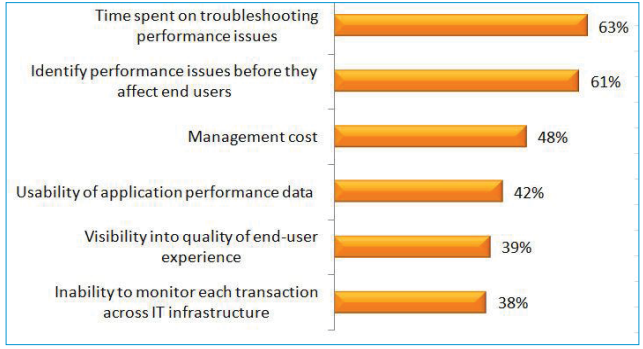
Source: "The hidden costs of managing applications in the cloud," Compuware/Research In Action White Paper, Dec. 2012, based on survey results from 468 CIOs in Americas, Europe, and Asia.

vmware

Application performance management is hard

vmware

Challenges in managing application performance



- On average, **46.2 hours** spend in "**war-room**" scenarios each month

Source: Improving the usability of APM data: Essential capabilities and benefits. TRAC Research, June 2012, based on survey data from 400 IT organizations worldwide

vmware

5

APM-related problems we're working on

- Real-time performance monitoring
  - Infrastructure-level vs. application-level monitoring
- Automated performance modeling
  - Knowledge-driven vs. data-driven
  - Linear vs. nonlinear models
  - Offline vs. online modeling
- Computer-assisted performance troubleshooting
  - Correlation & model based problem localization
- Service level remediation via auto-scaling
  - Horizontal vs. vertical scaling

vmware

6

Infrastructure-level performance monitoring

Physical host metrics

- System-level stats collected by the hypervisor
  - *e.g.*, `esxtop` – CPU, memory, disk, network, interrupt
- CPU stats
  - *%USED, %RUN, %RDY, %SYS, %OVRLP, %CSTP, %WAIT, %IDLE, %SWPWT*
- ~100s-1000s metrics per host!

VM metrics

- Resource usage stats collected by the guest OS
  - *e.g.*, `dstat, iostat`
- ~10s metrics per VM

- Widely available on most platforms
- Available at a time scale of seconds to minutes

vmware

Application-level performance monitoring

Metrics reflecting end user experience

- Response times
- Throughput (or errors such as timed out requests)

VMware Hyperic monitoring tool

- Agents deployed in VMs
- Auto-discovers types of applications running
- Plugins to extract application-related performance stats
- Stats available at a time scale of minutes
- Stats aggregated in Hyperic server
- Supports over 80 different application components
- Extensible framework to allow customized plugins

vmware

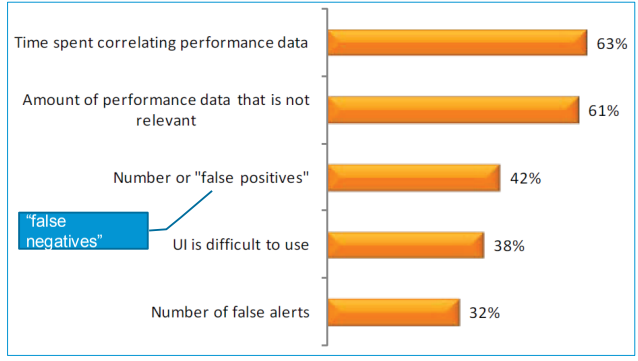
APM-generated big data

- "APM tools were part of the huge explosion in metric collection, generating thousands of KPIs per application."
- "83% of respondents agreed that metric data collection has grown >300% in the last 4 years alone."
- "88% of companies are only able to analyze less than half of the metric data they collect… 45% analyze less than a quarter of the data."
- "77% of respondents cannot effectively correlate business, customer experience, and IT metrics."

Source: "APM-generated big data boom." Netuitive & APMDigest, July 2012, based on survey of US & UK IT professionals.

vmware

Challenges in usability of performance data

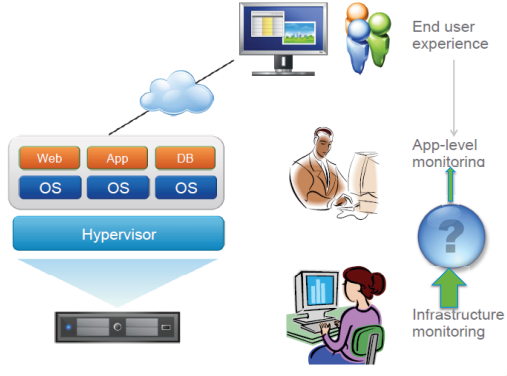


Source: Improving the usability of APM data: Essential capabilities and benefits. TRAC Research, June 2012, based on survey data from 400 IT organizations worldwide

vmware

The Semantic Gap challenge

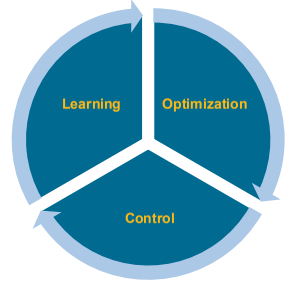
Correlating performance data from different sources



vmware

Better IT analytics for APM automation

Three-pronged approach



vmware

Semantic gap filled by performance models
Leaning-based approach

Traditional models harder to apply

- **First-principle models**: Only exist for special cases (e.g., flow models)
- **Queuing models**: More suitable for aggregate/average behavior
- **Architectural models**: Require domain knowledge, harder to automate

Empirical models via statistical learning

- Data driven, easier to automate and scale
- **Offline modeling** usually insufficient
  - Time-varying workloads
  - Changing system/software configurations
- **Online modeling** (models updated on demand)
  - Need to be low overhead and adaptive

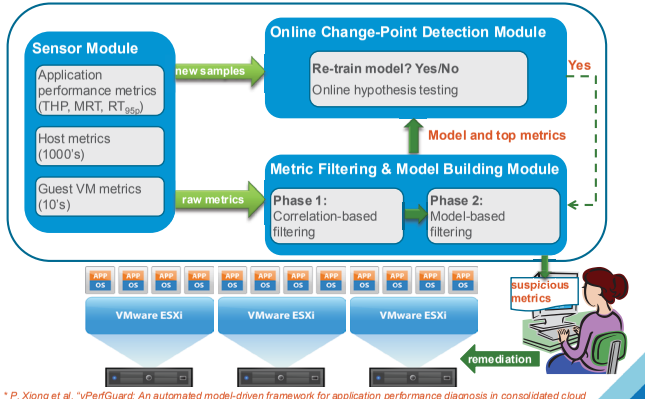
vmware 13

Three key questions

- **Q1: Which variables go into the model?**
  - Which system resources or parameters affect application performance the most?
  - Correlation-based analysis to provide hints
- **Q2:** What kind of model should we use?
  - Nonlinear models - better accuracy in general
  - Linear regression models - cheaper to compute and easier to interpret
- **Q3:** How do we know our model is (still) accurate?
  - Online change-point detection

vmware 14

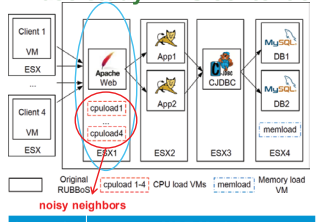
Correlation and model based metric selection



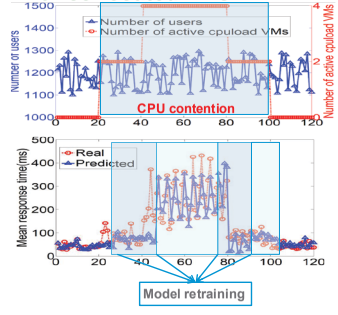
* P. Xiong et al. "vPerfGuard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments." ICPE 2013.

vmware 15

Case study: CPU contention with co-located VMs



Intervals	MRT Model
27 – 45	MRT = 1.13 H_ESX1_CPU_Util + 1.97 H_ESX4_Mem_Active – 89.7
46 – 74	MRT = 752.8 H_ESX1_CPULoad_1MinAvg – 562.9
75 – 89	MRT = 12.5 H_ESX1_Web_vCPU_Ready – 25.0
90 – 102	MRT = -7.70 H_ESX1_vCPU_Idle + 410.3



Note: **All models** during the contention period show **CPU** on **ESX1** as the top metric affecting application latency!

vmware 16

Performance remediation via auto-scaling

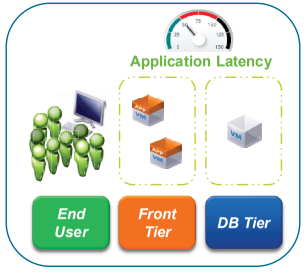
vmware

Challenges to ensure application performance

- Enterprise applications are distributed or multi-tiered
- App-level performance depends on access to many resources
  - HW: CPU, memory, cache, network, storage
  - SW: threads, connection pool, locks
- Time-varying application behavior
- Dynamic and bursty workload demands
- Performance interference among co-hosted applications

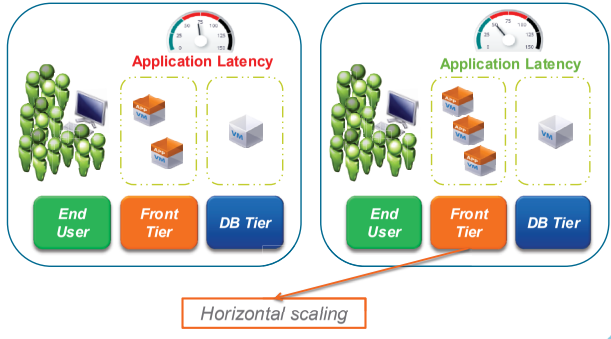
vmware

Auto-Scaling to maintain application SLO



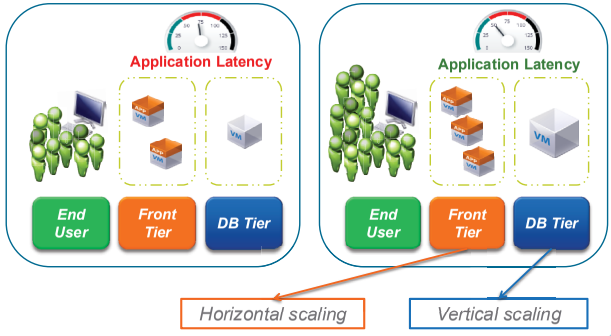
vmware

Auto-Scaling to Maintain Application SLO



vmware

Auto-Scaling to Maintain Application SLO



vmware

Horizontal scaling of applications

Academic research

- Muse: Managing energy and server resources in hosting centers (SOSP'01)
- A hybrid reinforcement learning approach to autonomic resource allocation (ICAC'05)
- A lot of recent work scaling clusters of VMs

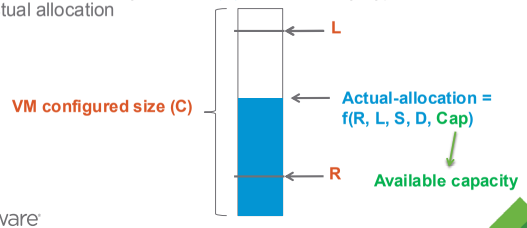
Commercial systems

- **Amazon Web Services**: http://aws.amazon.com/autoscaling/
- **RightScale**: http://www.rightscale.com
- **Rule-based:** User-set thresholds/alerts on resource utilization or load metrics
- **Learning-based:** Ongoing work at VMware

vmware

Vertical scaling of resource containers
Method 1: Dynamic resource control settings

- Available on various virtualization platforms
- For shared CPU, memory, disk I/O*, network I/O*:
  - ***Reservation (R)**** – minimum guaranteed amount of resources
  - ***Limit (L)*** – upper bound on resource consumption (non-work-conserving)
  - ***Shares (S)*** – relative priority during resource contention
- VM's CPU/memory *demand (D)*: estimated by hypervisor, critical to actual allocation



vmware

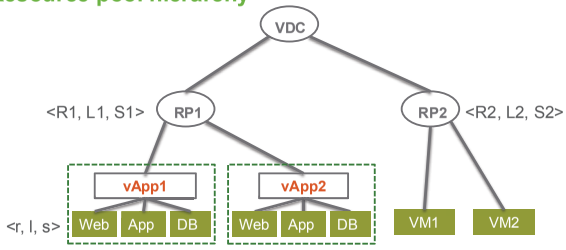
Vertical scaling of resource containers
Related work (not exhaustive)

- **Tuning resource *limits* (aka. *caps*)**
  - Adaptive control of virtualized resources in utility computing environments (Eurosys'07)
  - Autonomic resource management in virtualized data centers using fuzzy-logic-based applications (Cluster Computing Journal 2008)
  - Memory overbooking and dynamic control for Xen virtual machines in consolidated environment (IM'09, memory limit)
  - Vertical scaling of prioritized VMs provisioning (CGC'12)
  - Agile: Elastic distributed resources scaling for infrastructure-as-a-service (ICAC'13)
- **Tuning resource *shares* (aka. *weights*)**
  - *Maximizing server utilization while meeting critical SLAs via weight-based collocation management (IM'13)*
- **Tuning resource *reservations* (aka. *min*)**
  - Application-driven dynamic vertical scaling of virtual machines in resource pools (NOMS'14)

vmware

DRS (Distributed Resource Scheduler)

Resource pool hierarchy



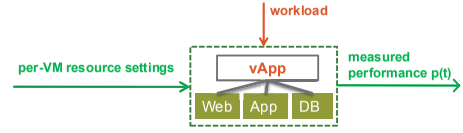
- Capacity of an RP divvied hierarchically based on resource settings
- Sibling RPs share capacity of the VDC
- Sibling VMs share capacity of the parent RP

* VMware distributed resource management: Design, implementation, and lessons learned, VMware Technical Journal, April 2012.

vmware

Powerful knobs, hard to use

- How do VM-level settings impact application performance?
- How to set RP-level settings to protect high priority applications within the RP?
- Fully reserved (R=L=C) for critical applications
  - Leads to lower consolidation ratio due to admission control
- Others left at default (R=0, L=C) until performance problem arises
  - Increases reservation for the bottleneck resource (which one? by how much?)

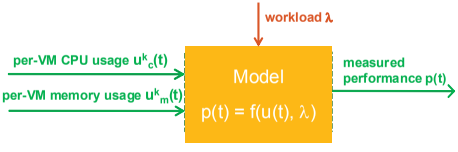


vmware

Performance model learned for each vApp

Maps VM-level resource allocations to app-level performance

- Captures multiple tiers and multiple resource types
- Choose a linear regression model (easy to compute)
- Workload indirectly captured in model parameters
- Model parameters updated online in each interval (tracks nonlinearity)



vmware

Rule-based vs. model-based feedback control

Rule-based	Model-based
often involves no analytical model	requires an analytical model
driven by intuition and domain knowledge	driven by quantitative relationships
hard to control multiple knobs at the same time	captures interactions between multiple metrics
no concern of dynamics	considers dynamics and transient responses
threshold and heuristics based	standard control methods as building blocks
no systematic consideration of stability	systematically handles tradeoff between stability & performance

vmware

Use optimization to handle design tradeoff

- An example cost function

$$J(\mathbf{u}(t+1)) = (p(t+1) - p_{SLO})^2 + \beta \| \mathbf{u}(t+1) - \mathbf{u}(t) \|^2$$

performance cost

control cost

Tradeoff between performance and stability

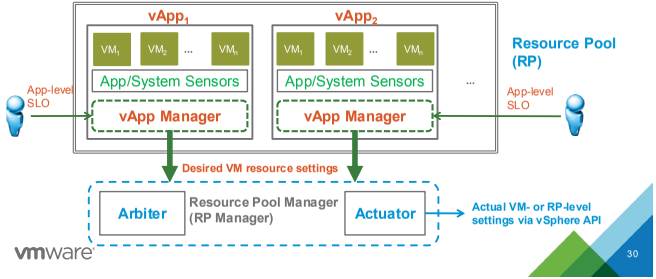
- Solve for optimal resource allocations

$$u^*(t+1) = g(p(t), p_{SLO}, \mathbf{u}(t), \lambda, \beta)$$

vmware

AppRM: Model-based vertical scaling

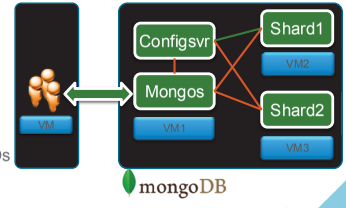
- Auto-tunes VM-level and RP-level resource control settings to meet application SLOs
  - For each application, **vApp Manager** translates its SLO into ***desired*** resource control settings at individual VM level
  - For each resource pool, **RP Manager** computes the ***actual*** VM- and RP-level resource settings to satisfy all critical applications



vmware

Performance evaluation

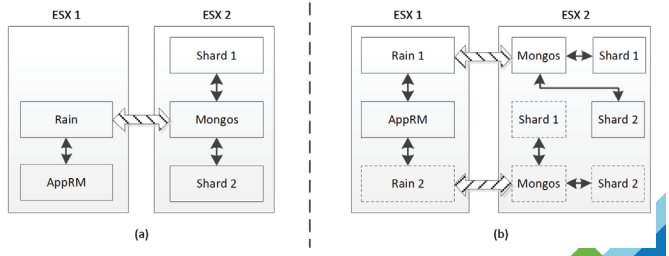
- Application
  - **MongoDB** – distributed data processing application with sharding
  - **Rain** – workload generation tool to generate dynamic workload
- Workload
  - Number of clients
  - Read/write mix
- Evaluation questions
  - Can the vApp Manager meet individual application SLO?
  - Can the RP Manager meet SLOs of multiple applications?



vmware

Testbed setup

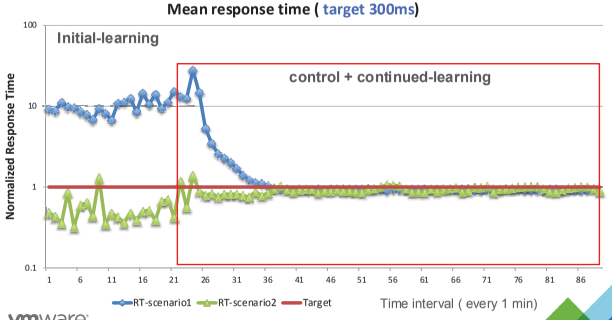
- Two ESX 5.0 GA hosts
- ESX2 (12 cores, 96 GB) to emulate the capacity of a VDC
- Three VMs per MongoDB instance (2 vCPUs, 4 GB)
- One VM per instance of Rain, one VM for AppRM



vmware

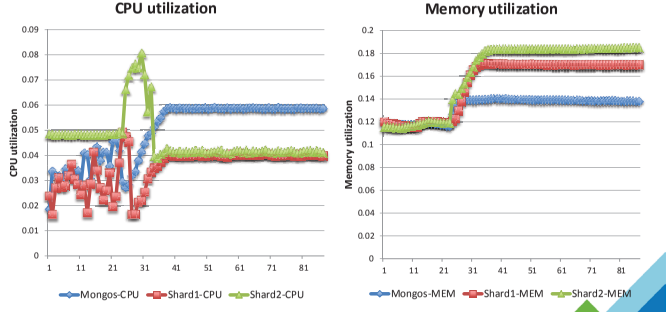
Result: Meeting mean response time target

- Scenario1 - Initial settings: R = 0, Limit = 512 (MHz, MB)
- Scenario2 – Initial settings: R = 0, L = unlimited (cpu, mem)



Resource utilization (under-provisioned case)

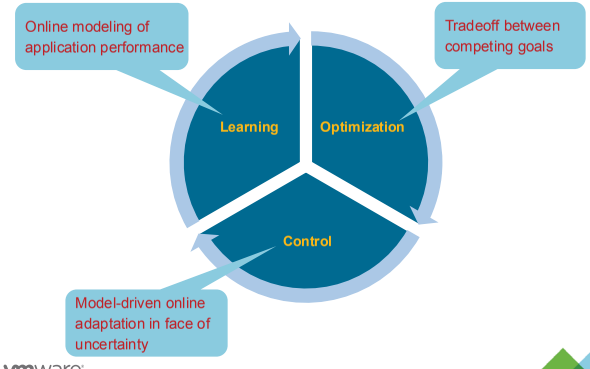
- Target response time = 300 ms
- Initial setting R = 0, L = 512 MHz/MB (under-provisioned)



Vertical scaling of resource containers
Method 2: Runtime reconfiguration of VM sizes

- Configured size for a VM
  - *#vCPUs*
  - *Memory size*
  - *#virtual disks, disk sizes*
  - *#vNICs*
- ESX allows over-commitment of CPU and memory
  - Sum(VM-size) >= host-capacity
- CPU/memory Hot-add supported by most recent OS's
  - Can be used to scale up a VM at runtime – work-in-progress
  - Need application support to leverage additional resources
- CPU/memory Hot-remove unsupported by most OS's
  - Requires VM reboot (undesirable)

Recap:
APM automation requires better analytics



References

- *X. Zhu, et al. "What does control theory bring to systems research?" ACM SIGOPS Operating Systems Review, 43(1), January 2009.*
- P. Padala *et al*. "Automated control of multiple virtualized resources." *Eurosys 2009.*
- A. Gulati *et al. "*Cloud scale resource management: Challenges and techniques." *HotCloud* 2011.
- A. Gulati *et al.* "VMware distributed resource management: Design, implementation, and lessons learned." *VMware Technical Journal*, Vol. 1(1), April 2012.
- P. Xiong *et al*. "vPerfGuard: An automated model-driven framework for application performance diagnosis in consolidated cloud environments." *ICPE 2013.*
- A. Gulati , "Towards proactive resource management in virtualized datacenters," *RESoLVE 2013.*
- L. Lu, *et al.*, "Application-Driven dynamic vertical scaling of virtual machines in resource pools." to appear at *NOMS 2014*.

MODERN INFRASTRUCTURE: THE CONVERGENCE OF NETWORK, COMPUTE, AND DATA

Jason Hoffman, Ericsson

The three pillars of our industry are network, compute, and data. All trends come down to the convergence of these. The convergence of network and compute resulted in the "the network is the computer"; the convergence of network and data spawned the entire networked storage industry and now we believe we're in the technology push where we are converging compute and data. In this talk, we'll cover the philosophical basis, the overall architecture, and the deep details of a holistic datacenter implementation.

SOME THOUGHTS

Johan Eker
Principal Researcher

Ericsson

Some slides and ideas courtesy of Jason Hoffman, Head of the Ericsson Cloud System and Platforms

ERICSSON, WHO?

OPERATOR CLOUD SEGMENTS

Operator Public Cloud	Operator IT Private Cloud	Operator Telecom Private Cloud
Virtualized Compute + WAN resources sold as a service to Enterprises aka "Managed Cloud"	Virtualized IT functions (OSS, BSS, SDP, ERP, CRM etc)	Virtualized telecom functions (e.g. IMS)

openstack™ CLOUD SOFTWARE

NFV

Network
Functions
Virtualization

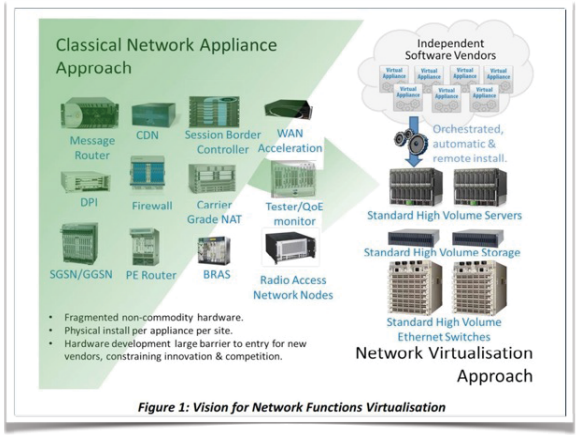


Figure 1: Vision for Network Functions Virtualisation

500 BILLION DEVICES

500 BILLION DEVICES

Doing what?

500 BILLION DEVICES

Collecting data. Acting on data.

DATA.
Source. Observer. Student.

DATA FROM HUMANS
Photos. Videos. Text. Audio.

DATA FROM HUMANS
Google. FB. Enterprise File & Mail.

DATA FROM MACHINES
Servers, Phones, Wearables, Sensors, Cars.

DATA FROM NATURE

Is the highest resolution data available.

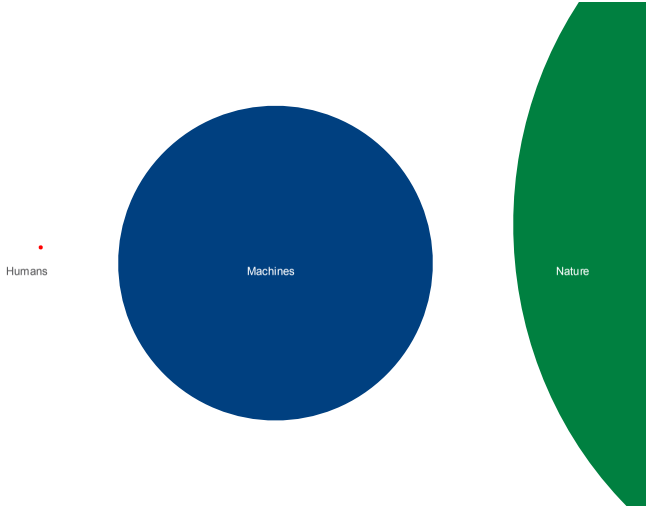
10,000,000 GENOMES IS 20EB

Keep all the data

STORAGE INDUSTRY SHIPPED 16EB

In 2012 (according to IDC)

A SINGLE IDEA CAN CONSUME AN ENTIRE INDUSTRY



WHAT WE'VE BEEN DOING

Humans Source, Humans Observe, Humans Learn
Nature Source, Nature Observe, Nature Learns
Nature Source, Humans Observe, Humans Learn
Nature Source, Machine Observes, Humans Learn

WHAT'S NEW

Humans Source, Machines Observe, Humans Learn
Humans Source, Machines Observe, Machines Learn
Machines Source, Machines Observe, Machines Learn
Nature Source, Machines Observe, Machines Learn

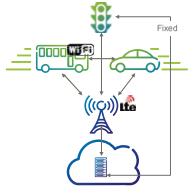
500 BILLION DEVICES

Collecting Data.
Learning.
Acting.

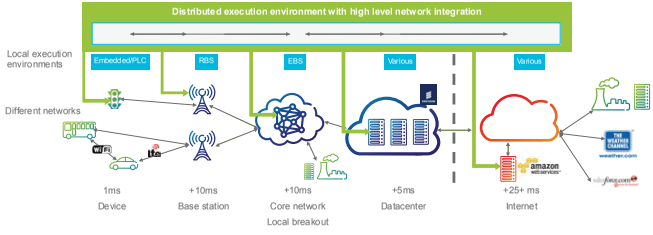
MISSION CRITICAL CLOUD

Timing

High availability



EVERYTHING IS PROGRAMMABLE



THIS IS THE HARDWARE

Distributed. Heterogenous. Dynamic.

HOW TO PROGRAM?

HOW TO PROGRAM?

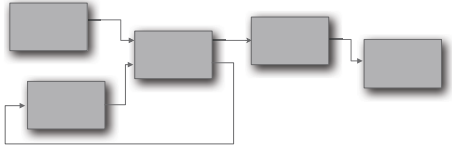
Ease of use. Resource. Timing. Resilient.

HOW TO MANAGE?

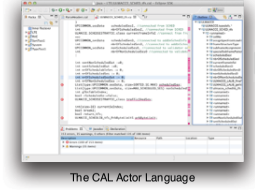
Distributed. Network, Computed & Data integration

AN APPLICATION AS A GRAPH

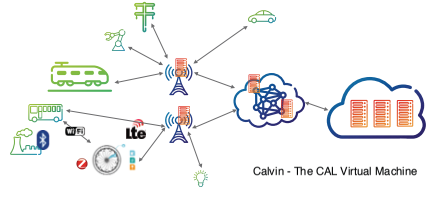
- Nodes called actors
- Message passing
  - Actors only interact via ports & FIFO connections
- Scheduling decoupled from algorithm
- Programming in CAL, Erlang or X



CALTOOPIA.ORG



The CAL Actor Language



Available at GitHub

Calvin - The CAL Virtual Machine

THE END

Everything is programmable

The data collected is used to control things

The cloud is integrated with devices & network

EVENT-BASED CONTROL: A WAY TO REDUCE RECONFIGURATION IN AUTONOMIC COMPUTING

Nicholas Marchand, GIPSA lab, France

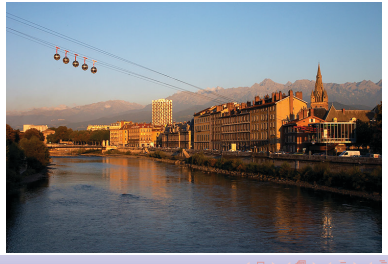
My talk will focus on a new control techniques called event-based control. This approach recently developed differs from classical control in the sense that the control value is updated only when needed. Reducing the number of control update often means for systems in the computer science domain a reduction of system re-configuration. The talk will focus on the practical use of this new technique to control the service time of an Hadoop MapReduce cluster. A comparison between classical control and event-based control will be given.

Event based control:

A way to reduce reconfigurations in autonomic computing ?

N. Marchand

gipsa, Control Systems Department, Grenoble, FRANCE



Outline

Motivation

to put control theory in computer science

- Dealing with the **dynamics**: **time is crucial**
- Mathematical tools to "control" a system
- By "control", we mean being able to
  - define a control objective
  - define control actions accordingly
  - **guarantee** performances of the controlled system
    - despite errors
    - despite perturbations
    - ➳ Facing everything that is unknown
    - ➳ Guarantee stability
  - Many other area of control theory are relevant to computer science
    - Fault tolerant control, fault detection, supervision, etc.
- Nowadays control theory is everywhere...
  - automotive, robotics, energy (grids, production, etc.), microelectronics (DVFS), etc.
- ...except maybe in computer science

Challenging difficulties

Let's start with the hardest things

- Languages difficulties

Words	Computer science	Control theory
Autonomic or Autonomous	Controlled	Uncontrolled
Time response	Queuing + processing time	time needed to reach x% of the final value
Parameter	variables you can change	constants
Cloud	Set of interconnected computers	Look at Lund's sky
Control	Parametrization	$\frac{dx}{dt} = f(x,u)$
...	...	...

- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

gipsa-lab

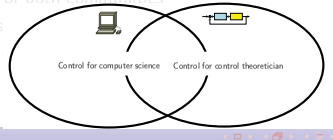
Challenging difficulties

Let's start with the hardest things

- Languages difficulties

Words	Computer science	Control theory
Autonomic or Autonomous	Controlled	Uncontrolled
Time response	Queuing + processing time	time needed to reach x% of the final value
Parameter	variables you can change	constants
Cloud	Set of interconnected computers	Look at Lund's sky
Control	Parametrization	$\frac{dx}{dt} = f(x,u)$
...	...	...

- Interest of both communities
- No phys
- "Let's d



gipsa-lab

Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
  - Computer science community wants control already operative
  - Control community don't care about computer sciences
    - Except in Lund !
    - IFAC technical commitee on computer for control but none on control for computers (one on on mining..., see IFAC website)
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

gipsa-lab

Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
  - Computer science community wants control already operative
  - Control community don't care about computer sciences
    - Except in Lund !
    - IFAC technical commitee on computer for control but none on control for computers (one on on mining..., see IFAC website)
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

gipsa-lab

Challenging difficulties

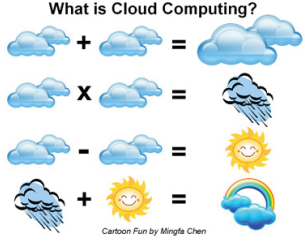
Let's start with the hardest things

- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
  - Building models is critical and unusual
    - How do i put the system in the control theory normal form $\frac{dx}{dt} x = f(x,u)$?
    - Frequency, poles, etc. are sometimes clear
    - Control, outputs, sensors, etc. can disappear with a system update
    - Evolution of a system can be discontinuous (robustness issue)
    - No "tiredness", only crashes
  - Model must capture main behavior BUT
    - if too precise ☞ too complex
    - if too complex ☞ inefficient for control (unrobust)
    - Model for control is not classical modeling
  - Requires much more interaction than usual sciences
- "Let's do things in cloud " (Sara Bouchenak from LIG-lab)

Challenging difficulties

Let's start with the hardest things

- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud **computing**" (Sara Bouchenak from LIG-lab)



Challenging difficulties

Let's start with the hardest things

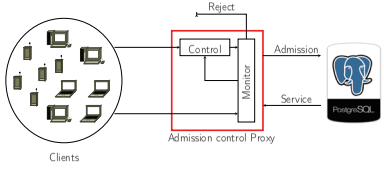
- Languages difficulties
- Interest of both communities
- No physics behind algorithms, applications, services, etc.
- "Let's do things in cloud **control**" (Sara Bouchenak from LIG-lab)

Menu

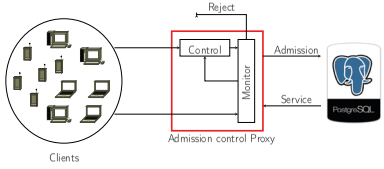
- *Starter*: An short example of how bad computer system can be for control theory:
  - Admission control for PostgreSQL database server
- *Main dish*:
  - Cloud control needs to
    - react to spikes (high frequency)
    - reconfigure as less as possible (low frequency)
    - Antinomic !
  - Focus on Event-Based control
    - More on event based PID
    - Short presentation of extensions
    - Assure SLA compliance in Hadoop Mapreduce
- *Dessert*: What need to be more efficient

Hope it will be not too indigestible !

A nonlinear system example



A nonlinear system example



- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} &= (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} &= -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} &= -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

- N: number of concurrent request on the server
- α: abandon rate
- T_o: Throughput of served requests
- T_i: Throughput of incoming requests
- MPL: Multi Processing Level, it is the control variable
- a, b, c and Δ: parameters

A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} &= (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} &= -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} &= -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

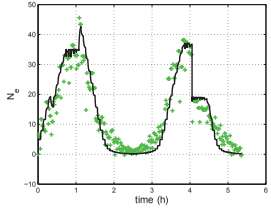
✓ Quite a pretty model
- few variables/parameters
- easily identifiable
- fits well

A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} &= (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} &= -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} &= -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model
- few variables/parameters
- easily identifiable
- fits well

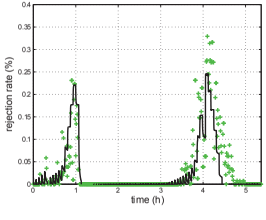


A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} &= (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} &= -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}(1-\frac{T_o}{T_i})\right] \\ \frac{dT_o}{dt} &= -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model
- few variables/parameters
- easily identifiable
- fits well

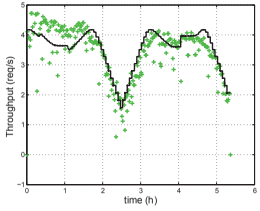


A nonlinear system example

- Model gives when saturated:

$$\begin{cases} \frac{dN}{dt} &= (1-\alpha)\cdot T_i - T_o \\ \frac{d\alpha}{dt} &= -\frac{1}{\Delta}\left[\alpha - \frac{N}{MPL}\left(1-\frac{T_o}{T_i}\right)\right] \\ \frac{dT_o}{dt} &= -\frac{1}{\Delta}\left[T_o - \frac{N}{aN^2+bN+c}\right] \end{cases}$$

✓ Quite a pretty model
- few variables/parameters
- easily identifiable
- fits well



✗ Not an easy model
- Model is highly nonlinear
- Not in the standard form for control theory
- Control is the level saturation of an exogenous input

gipsa-lab

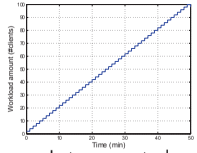
LCCC workshop on Cloud Control

N. Marchand

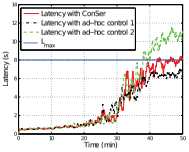
Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

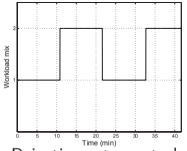
A nonlinear system example

- Controlled thanks nonlinear control theory (Lyapunov)
- Stability is guaranteed for any value of the parameters
  - No danger of miss-identification

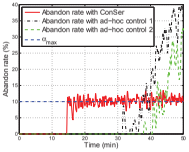


Latency control





Rejection rate control

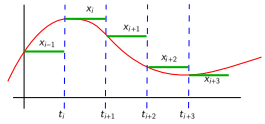


Menu

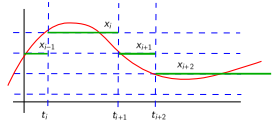
- *Main dish*:
  - Cloud control needs to
    - react to spikes (high frequency)
    - reconfigure as less as possible (low frequency)
    - Antinomic !
  - Focus on Event-Based control
    - More on event based PID
    - Short presentation of extensions
    - Assure SLA compliance in Hadoop Mapreduce

Event-based sampling vs. periodic sampling

- Periodic sampling
  - Sampling periodically on time
  - Analogical to Riemann's integral
  - Well known theory (Shannon, etc.)

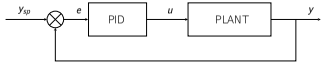


- Event-based sampling
  - Sampling on level's
  - At first glance close to Lebesgues integral
  - Different extension :
    - Outside event (event-triggered)
    - State/output dependent sampling (self-triggered)
  - Should reduce transmission/computation
  - Few theory



PID controller

Classical work



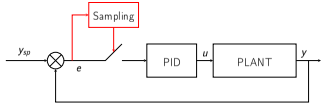
- Classical PID
  - In the frequency domain:

$$U(s) = K\left(E(s) + \frac{1}{T_i s}E(s) + T_d s E(s)\right)$$

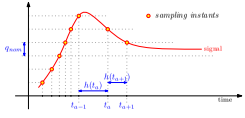
  - Discrete time version (h_{nom}: sampling period, N tunes the filter):

$$\begin{aligned} u_p(t_k) &= Ke(t_k) \\ u_i(t_{k+1}) &= u_i(t_k) + K_i h_{nom} e(t_k) \\ u_d(t_k) &= \frac{T_d}{T_d + N h_{nom}} u_d(t_{k-1}) + \frac{K T_d N}{T_d + N h_{nom}}(e(t_k) - e(t_{k-1})) \\ u &= u_p + u_i + u_d \end{aligned}$$

PID controller



- Idea:
  - do not update the control if y is close to y_{sp}, typically if $||e(t_a) - e(t_{a-1})|| \leq q_{nom}$
- No need to respect Shannon
- Bad behavior of the integral part
- q_{nom} linked to precision and noise



What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling $\Rightarrow$ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + (t_{i+1} - t_i) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{3}}$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling $\Rightarrow$ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + (t_{i+1} - t_i) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{3}}$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 13 / 26

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

What can happen (1/3) ?

- Simple integrator $\frac{dx}{dt} = u$
- Level-crossing sampling $\Rightarrow$ update the control when $e(x) = 0$
- Trajectory : $x_{i+1} = x_i + (t_{i+1} - t_i) \cdot u$
- Sampling instants t_i
- Sampling set: $T_{e,k,x_0} := \{t_i\} \supset \{0\}$

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $T_{e,k,x_0} := \{j \cdot (1 - \exp(-1)), j \in \mathbb{N}\}$
   - closed-loop system is globally asymptotically stable
   - the solution is defined on $[0, \infty[$
   - the sampling set depends upon x_0
2. $k(x) = -x^{\frac{1}{3}}$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{\sqrt{i}(i+1)}$
   - closed-loop system is globally asymptotically stable
   - Zeno phenomenon: the solution is defined only on $[0, 1.86[$

N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 13 / 26

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

What can happen (2/3) ?

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{i+1}$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = 0$ when $\lim t_i = \infty$
   - Infinitely fast sampling at infinity
2. $k(x) = -x^3$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \exp(2i) \cdot [1 - \exp(-1)]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$
   - Infinitely slow sampling at infinity

What can happen (2/3) ?

1. $k(x) = -x$, $e(x) = 0$ when $|x| = \frac{1}{\kappa}$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \frac{1}{i+1}$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = 0$ when $\lim t_i = \infty$
   - Infinitely fast sampling at infinity
2. $k(x) = -x^3$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$
   - $x_0 = 1 \Rightarrow t_{i+1} - t_i = \exp(2i) \cdot [1 - \exp(-1)]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$
   - Infinitely slow sampling at infinity

What can happen (3/3) ?

- **Unstable system**: $\frac{dx}{dt} = (x+u)^3$
- Solution is: $x_{i+1} = \frac{x_i + u}{\sqrt{1 - 2(t_{i+1} - t_i) \cdot (x_i + u)^2}} - u$

2. $k(x) = -2x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$ and initial condition $x_0 = 1$
   - $t_{i+1} - t_i = \frac{\exp(2i)}{2} \cdot \left[1 - \frac{1}{(2 - \exp(-1))^2}\right]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$

What can happen (3/3) ?

- **Unstable system**: $\frac{dx}{dt} = (x+u)^3$
- Solution is: $x_{i+1} = \frac{x_i + u}{\sqrt{1 - 2(t_{i+1} - t_i) \cdot (x_i + u)^2}} - u$

2. $k(x) = -2x$, $e(x) = 0$ when $|x| = \exp(-\kappa)$, $\kappa \in \mathbb{Z}$ and initial condition $x_0 = 1$
   - $t_{i+1} - t_i = \frac{\exp(2i)}{2} \cdot \left[1 - \frac{1}{(2 - \exp(-1))^2}\right]$
   - closed-loop system is globally asymptotically stable
   - $\lim t_{i+1} - t_i = \infty$ when $\lim t_i = \infty$
   - Shannon's condition is inconsistent
   - the solution is defined on $[0, \infty[$

PID controller

- We focus on the integral part
  - What happens one waits too long before updating the control ?
  - The integral part grows because $h(t_a)$ grows:

$$u_i(t_a) = u_i(t_{a-1}) + K_i \underbrace{h(t_a)}_{\text{big}} \underbrace{e(t_a)}_{\text{small}}$$

  - Strong overshoot when the control is updated (similar to saturated PID without antiwindup)
- **Solution**: replace the product $h \cdot e$ by a bounded function he:

$$u_i(t_a) = u_i(t_{a-1}) + K_i \underbrace{he(t_a)}_{\text{limited}}$$

  - Saturation, Exponential forgetting factor, Hybrid, etc.

PID controller

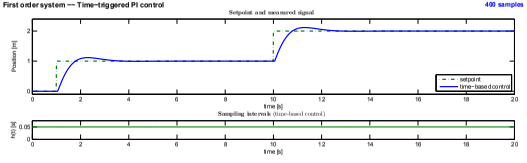
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Periodic sampling



gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

PID controller

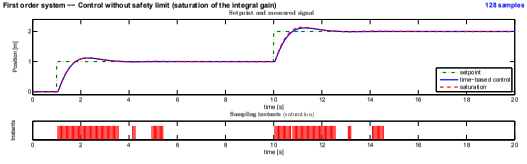
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Evend based PID with saturated $h \cdot e$



gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

PID controller

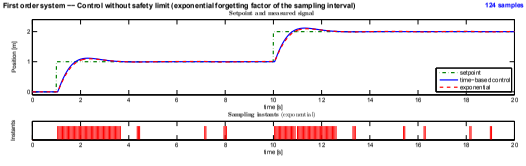
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

Evend based PID with exponential forgetting



gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

PID controller

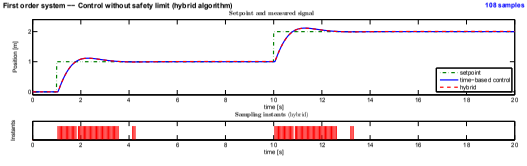
Simulation result

- **First order system**:

$$H(s) = \frac{G}{1+\tau \cdot s} \quad \text{where } G = 1 \text{ and } \tau = 1$$

- **PID controller**: $K_p = 1.83$, $T_i = 0.457$ and sampling rate $0.05\,s$

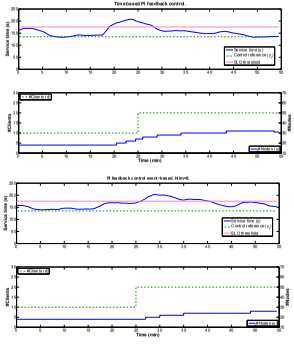
Evend based PID with hybrid $h \cdot e$



MapReduce control

Simple PI controller

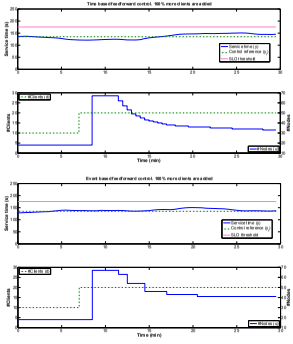
- Time based (8 updates) vs. Event based (4 updates)



MapReduce control

Simple PI controller with feedforward

- Time based (18 updates) vs. Event based (6 updates)



More food for people in control theory

- Event-based control is really recent
- Now exist :
  - Almost basic linear control where carried in an event-based framework (PID, LQR, etc.)
  - Sontag's general formula has been extended
  - A lot of strategies based on Lyapunov theory exist
  - Many practical implementations even on noisy and unstable systems
  - Early results are appearing for time-delayed systems
- Remain to clarify
  - Real number of control updates
  - All what it brings (in good and bad) is not clear
  - Frequency analysis is less convenient

Conclusion

- People always adopt control theory ...
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 21 / 26

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 21 / 26

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 21 / 26

Conclusion

- People always adopt control theory ... under constraint
  - Ecological constraints: car industry (in the 90's)
  - Nuclear plant: from the beginning (and one must be sure it works)
  - Cost constraints: Petrol industries (in late 50's)
  - Energy constraints: Embedded systems (in the 00's)
  - Crash risks: Smart Grids (nowadays)
- In all cases it was (is) a question of money
- Is it a question of money in cloud computing ?
- Before adopting control theory, intuitive control was the strategy
- Theory is the only way (control theory, game theory, queuing theory, etc.)
  - to face safely complexity
  - to guarantee results (even in unknown/unpredictable environment)
  - to have flexibility

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity etc

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity etc

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity etc

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Better sort things by speed
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that handles computer science problems
- From both side:
  - Spend more time together
  - Mix techniques from both side
- Some inspiring fields
  - Embedded systems
    - deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...
  - Electrical grids
    - centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

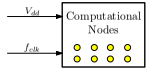
What need to be improved

- From the computer science side:
  - Classification of problems in big classes
  - Standardisation of inputs/outputs/variables for each class
  - Co-design / Control aware software
  - Patience (to explain and to get results)
- From the control theory side:
  - More interest
  - Building a theory that better handles computer science problems
  - Adaptive methods/model free
  - Large scale interconnected systems
- From both side:
  - Spend more time together
- Some inspiring fields
  - Embedded systems (deadline problems, energy optimization, re-allocation, heterogeneous MPSoC, ...)
  - Electrical grids (centralized/decentralized, providers/consumers, cascading failure, heterogeneity, etc.)

Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

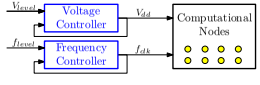
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

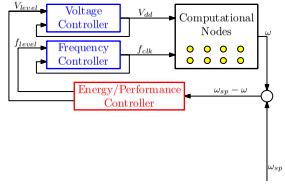
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

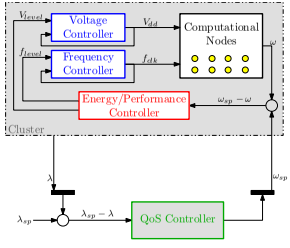
1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



gipsa-lab

LCCC workshop on Cloud Control

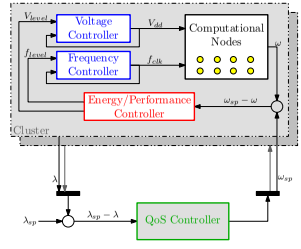
N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)

1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 24 / 26

gipsa-lab

LCCC workshop on Cloud Control

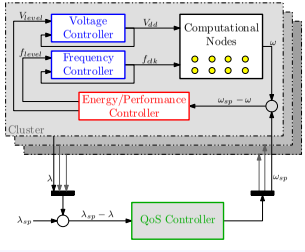
N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)
Also the approach used in smart grids

1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 24 / 26

gipsa-lab

LCCC workshop on Cloud Control

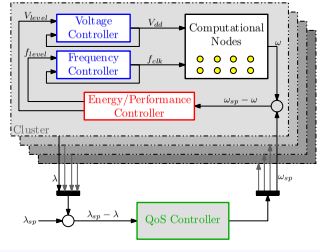
N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Feedback loops become essential to handle variability

Three nested loops are used (to dynamically manage energy on chips)
Also the approach used in smart grids

1. Control of the voltage and the frequency
2. Control of the energy-performance tradeoff
3. Control of the applicative Quality of Service (QoS)



N. Marchand (gipsa) LCCC workshop on Cloud Control 22/03/2012 24 / 26

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

Grenoble Workshop on Autonomic Computing and Control

- Date: 27 may 2014
- Location: Grenoble
- Organisation: Eric Rutten, INRIA and Stéphane Mocanu, Gipsa-lab
- Confirmed speakers:
  - Karl-Erik ARZEN (Lund, Sweden)
  - Alberto LEVA (Milano, Italy)
  - Ada DIACONESCU (Telecom Paris-Tech, France)
  - Suzanne LESECQ (CEA LETI)
  - Didier DONSEZ (LIG)
  - Bogdan ROBU (GIPSA)
  - Eric RUTTEN (INRIA)

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Confirmed lecturers:
  - Didier HENRION
  - Andrew TEEL
  - Laurent PRALY
  - Mirko FIACCHINI
  - Luca ZACCARIAN

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Co

gipsa-lab

LCCC workshop on Cloud Control

N. Marchand

Introduction
Motivation
Outline
NL example
EB-PID
Formulation
Cases
Simulations
MapReduce control
EB-Control
Conclusion

35th International Summer School of Automatic Control

- Date: September, 8-12, 2014
- Location: Grenoble
- Focus: Modern Tools for Nonlinear Contral
- Co

GUIDED TOUR THROUGH A CLOUD DATACENTER – THE UMEÅ UNIVERSITY APPROACH TO CLOUD RESOURCE MANAGEMENT

Erik Elmroth, Umeå University

By taking a holistic approach to cloud resource management, we aim to transform today's static and energy consuming cloud data centers into self-managed, dynamic, and dependable infrastructures, constantly delivering expected quality of service with acceptable operation costs and carbon footprint for large-scale services with varying capacity demands. The presentation will provide the birds-eye's view of our efforts as well as several glimpses of selected completed, ongoing, and planned research efforts. These efforts address fundamental and inter-twined self-management challenges assuming that there during execution are stochastic variations in capacity need and resource availability, as well as changes in system response and operation costs. Sample challenges include how much capacity to allocate at any time for an elastic application, where to allocate that capacity, if to admit an elastic service with unknown lifetime and future capacity demands, how to optimize the various management tools' concerted actions, etc, while taking into account the need for differentiated quality of service and the scalability requirements of the management tools themselves. For further reading about cloud resource management research at Umeå University, Sweden, please visit www.cloudresearch.org.

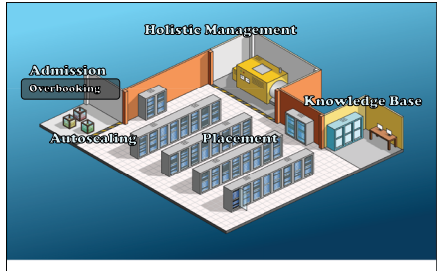
Guided Tour through a Cloud Datacenter

- The Umeå University approach to cloud resource management

Erik Elmroth
Umeå University
elmroth@cs.umu.se

LCCC Focus Period / 4th Cloud Control Workshop
Lund University,
May 7-9, 2014

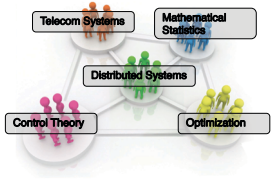
www.cloudresearch.org



Guided Tour

14

Interdisciplinary collaborations

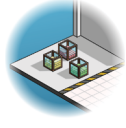


Images from http://artsandhumanities.pressible.org

11

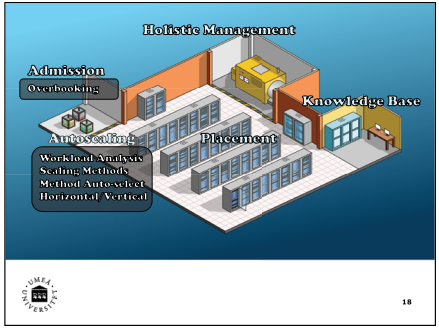
Admission Control

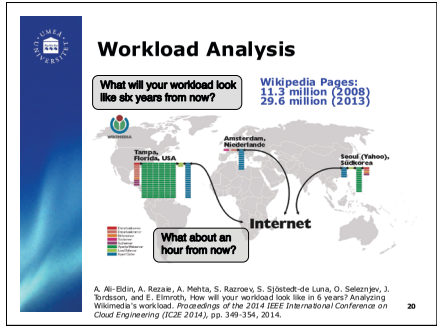
- Determines load, revenue, and risks
- Risk theory
  - Utility versus Violations
  - Overbooking
  - Long term effects

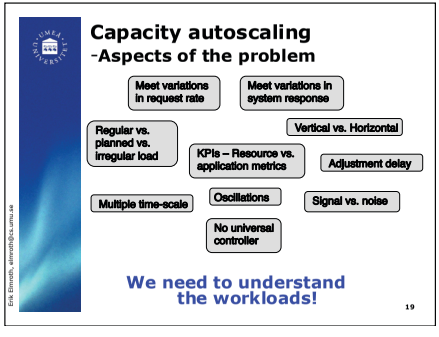


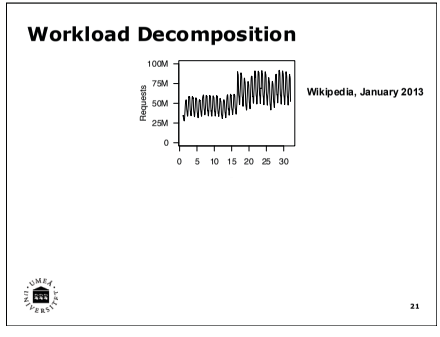
L. Tomás and J. Tordsson, Cloudy with a Chance of Load Spikes: Admission Control with Fuzzy Risk Assessments, Proc of 6th IEEE/ACM International Conference on Utility and Cloud Computing, 2013

15

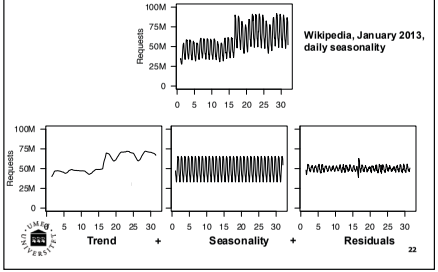

Holistic Management
Admission
Overbooking
Knowledge Base
Autoscaling
Placement
Workload Analysis
Scaling Methods
Method Autoscaler
Horizontal Vertical
18


Workload Analysis
What will your workload look like six years from now?
Wikipedia Pages:
11.3 million (2008)
29.6 million (2013)
Tampa, Florida, USA
Amsterdam, Niederlande
Seoul (Yahoo), Südkorea
Internet
What about an hour from now?
A. Ali-Eldin, A. Rezaie, A. Mehta, S. Razroev, S. Sjöstedt-de Luna, O. Seleznjev, J. Tordsson, and E. Elmroth, How will your workload look like in 6 years? Analyzing Wikimedia's workload. Proceedings of the 2014 IEEE International Conference on Cloud Engineering (IC2E 2014), pp. 349-354, 2014.
20


Capacity autoscaling
-Aspects of the problem
Meet variations in request rate
Meet variations in system response
Regular vs. planned vs. irregular load
Vertical vs. Horizontal
KPIs – Resource vs. application metrics
Adjustment delay
Multiple time-scale
Oscillations
Signal vs. noise
No universal controller
We need to understand the workloads!
Erik Elmroth, elmroth@cs.umu.se
19

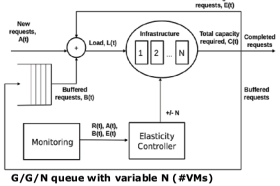

Workload Decomposition
100M
75M
50M
25M
0
Requests
0 5 10 15 20 25 30
Wikipedia, January 2013
21

Workload Decomposition



22

Capacity Autoscaling
Sample control theoretic model



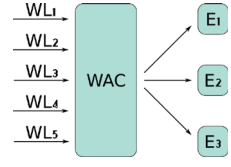
G/G/N queue with variable N (#VMs)

A. Ali-Eldin, M. Kihl, J. Tordsson, and E. Elmroth. Efficient Provisioning of Bursty Scientific Workloads on the Cloud Using Adaptive Elasticity Control, In Proceedings of the 3rd Workshop on Scientific Cloud Computing (ScienceCloud 2012), *ACM New York*, pp. 31-40, 2012.

A. Ali-Eldin, J. Tordsson, and E. Elmroth. An Adaptive Hybrid Elasticity Controller for Cloud Infrastructures, The 13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012), IEEE, pp. 204-212, 2012.

28

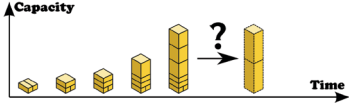
Several Autoscaling Methods + Auto-selection



A. Ali-Eldin, J. Tordsson, M. Kihl, and E. Elmroth. Workload Classification for Efficient Cloud Infrastructure Elasticity Control, *submitted*, 2013.

30

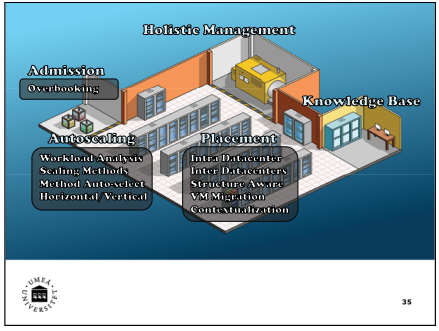
Scaling and Repacking

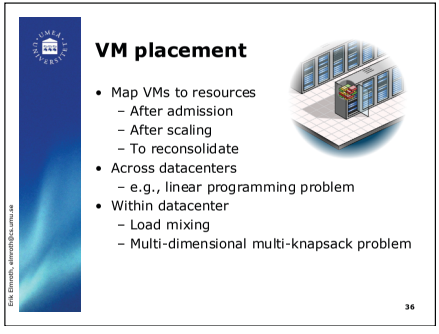


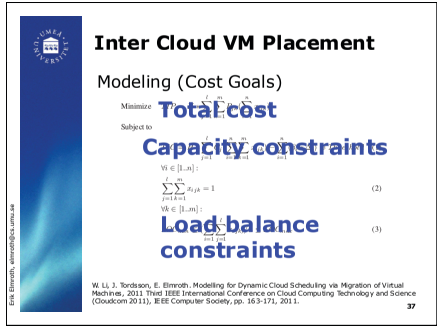
M. Sedaghat, F. Hernandez-Rodriguez, and E. Elmroth. A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud auto scaling, *Proc. ACM Cloud and Autonomic Computing Conference (CAC)*, ACM, 2013

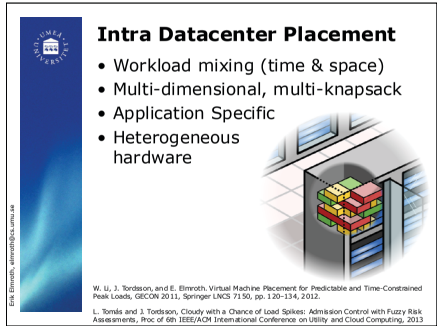
33

Erik Elmroth, elmroth@cs.umu.se


Holistic Management
Admission
Overbooking
Knowledge Base
Autoscaling
Placement
Workload Analysis
Scaling Methods
Method Autoselect
Horizontal Vertical
Intra Datacenter
Inter Datacenters
Structure Aware
VM Migration
Consolidation
35

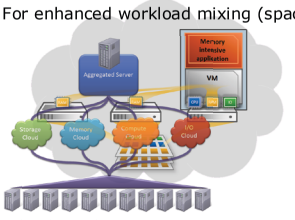

VM placement
Map VMs to resources
After admission
After scaling
To reconsolidate
Across datacenters
e.g., linear programming problem
Within datacenter
Load mixing
Multi-dimensional multi-knapsack problem
Erik Elmroth, elmroth@cs.umu.se
36


Inter Cloud VM Placement
Modeling (Cost Goals)
Minimize
Total cost
Subject to
Capacity constraints
Load balance constraints
(2)
(3)
W. Li, J. Tordsson, E. Elmroth. Modelling for Dynamic Cloud Scheduling via Migration of Virtual Machines, 2011 Third IEEE International Conference on Cloud Computing Technology and Science (Cloudcom 2011), IEEE Computer Society, pp. 163-171, 2011.
Erik Elmroth, elmroth@cs.umu.se
37


Intra Datacenter Placement
Workload mixing (time & space)
Multi-dimensional, multi-knapsack
Application Specific
Heterogeneous hardware
W. Li, J. Tordsson, and E. Elmroth. Virtual Machine Placement for Predictable and Time-Constrained Peak Loads, GECON 2011, Springer LNCS 7150, pp. 120-134, 2012.
L. Tomás and J. Tordsson, Cloudy with a Chance of Load Spikes: Admission Control with Fuzzy Risk Assessments, Proc of 6th IEEE/ACM International Conference on Utility and Cloud Computing, 2013
Erik Elmroth, elmroth@cs.umu.se

Relaxed box model virtualization

For enhanced workload mixing (space)

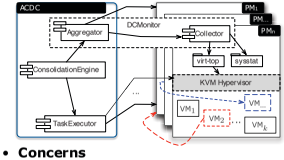


P. Svärd, J. Tordsson, B. Hudzia, E. Elmroth. Hecatonchire: Enabling Multi-Host VMs by Resource Aggregation and Pooling. *Submitted*, 2014

44

Erik Elmroth, elmroth@cs.umu.se

Datacenter Reconsolidation

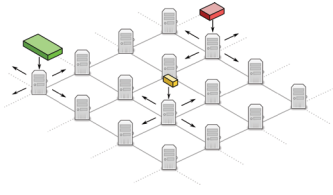


- **Concerns**
  - Optimal solution most likely infeasible
    - Gradual improvement
  - Heuristic approach

50

Erik Elmroth, elmroth@cs.umu.se

Decentralized Placement



M. Sedaghat, F. Hernández, and E. Elmroth, Peer to peer resource management for cloud data centers, *Submitted*, 2013

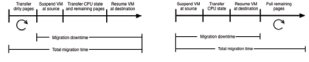
48

Erik Elmroth, elmroth@cs.umu.se

Live VM migration (without service interruption)

Pre-copy migration

Post-copy migration

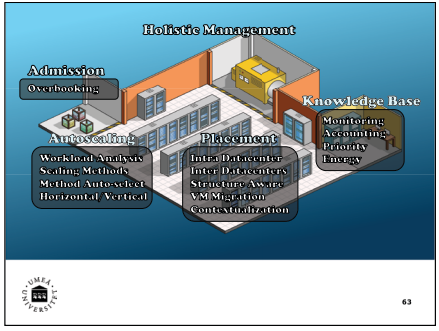


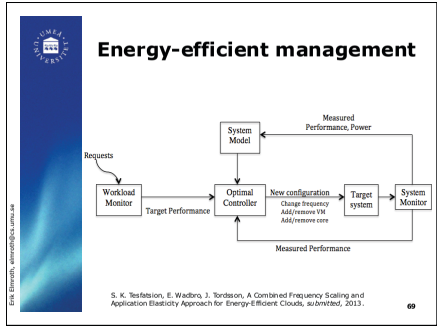
	Pre	Post	Hybrid
Continuous service		(✔)	✔
Resource usage		✔	✔
Robustness	✔		
Predictability		✔	✔
Transparency	✔	✔	✔

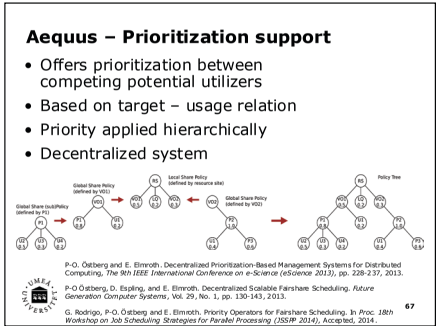
P. Svärd, J. Tordsson, E. Elmroth, B. Hudzia. The Noble Art of Live VM Migration - Principles and Performance, *submitted*, 2013.

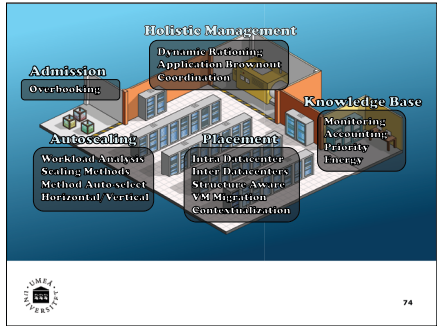
55

Erik Elmroth, elmroth@cs.umu.se


Holistic Management
Admission
Overbooking
Knowledge Base
Monitoring
Accounting
Priority
Energy
Autoscaling
Workload Analysis
Scaling Methods
Method Auto-select
Horizontal Vertical
Placement
Intra Datacenter
Inter Datacenters
Structure Aware
VM Migration
Contextualization
63


Energy-efficient management
Measured
Performance, Power
System
Model
Requests
Workload
Monitor
Target Performance
Optimal
Controller
New configuration
Change frequency
Add/remove VM
Add/remove core
Target
system
System
Monitor
Measured Performance
S. K. Tesfatsion, E. Wadbro, J. Tordsson, A Combined Frequency Scaling and Application Elasticity Approach for Energy-Efficient Clouds, submitted, 2013.
69


Aequus – Prioritization support
• Offers prioritization between competing potential utilizers
• Based on target – usage relation
• Priority applied hierarchically
• Decentralized system
P-O. Östberg and E. Elmroth. Decentralized Prioritization-Based Management Systems for Distributed Computing, The 9th IEEE International Conference on e-Science (eScience 2013), pp. 228-237, 2013.
P-O Östberg, D. Espling, and E. Elmroth, Decentralized Scalable Fairshare Scheduling, Future Generation Computer Systems, Vol. 29, No. 1, pp. 130-143, 2013.
G. Rodrigo, P-O. Östberg and E. Elmroth. Priority Operators for Fairshare Scheduling. In Proc. 18th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2014), Accepted, 2014.
67


Holistic Management
Dynamic Rationing
Application Brownout
Coordination
Admission
Overbooking
Knowledge Base
Monitoring
Accounting
Priority
Energy
Autoscaling
Workload Analysis
Scaling Methods
Method Auto-select
Horizontal Vertical
Placement
Intra Datacenter
Inter Datacenters
Structure Aware
VM Migration
Contextualization
74

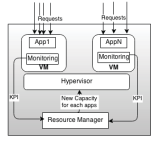
Dynamic Resource Rationing

Where to cut when resources are insufficient?

Two approaches
1. Strict QoS-level adherence
2. Overall cost-benefit with QoS-level weights

- Constrained optimization
- Substantial dependency on KPI-type (e.g. latency vs. throughput)
- System feedback on KPI and dimmer effect

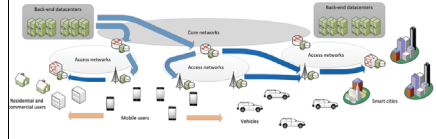
System Architecture



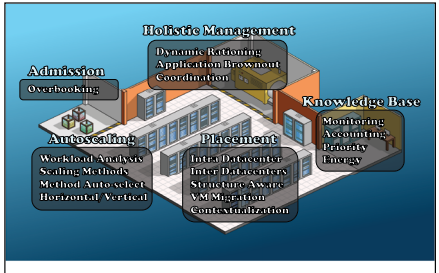
Erik Elmroth, elmroth@cs.umu.se

75

Managing the infinite (or telco) cloud



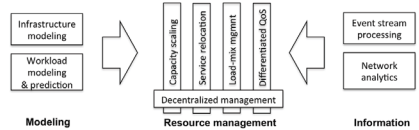
84

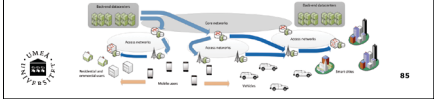


www.cloudresearch.org

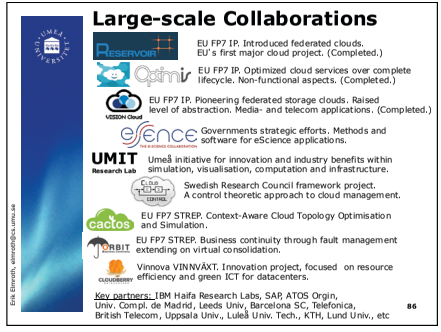
83

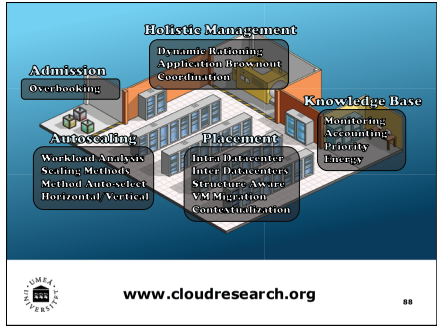
Managing the infinite (or telco) cloud

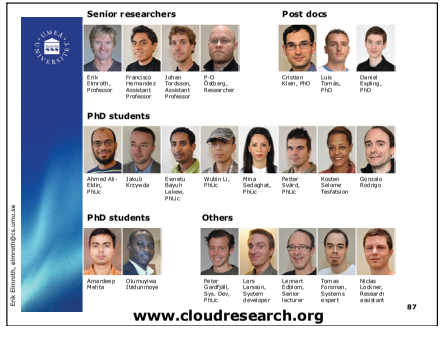




85


Large-scale Collaborations
RESERVOIR
EU FP7 IP. Introduced federated clouds. EU's first major cloud project. (Completed.)
Optimis
EU FP7 IP. Optimized cloud services over complete lifecycle. Non-functional aspects. (Completed.)
VISION Cloud
EU FP7 IP. Pioneering federated storage clouds. Raised level of abstraction. Media- and telecom applications. (Completed.)
eSence
Governments strategic efforts. Methods and software for eScience applications.
UMIT Research Lab
Umeå initiative for innovation and industry benefits within simulation, visualisation, computation and infrastructure.
Swedish Research Council framework project. A control theoretic approach to cloud management.
cactos
EU FP7 STREP. Context-Aware Cloud Topology Optimisation and Simulation.
ORBIT
EU FP7 STREP. Business continuity through fault management extending on virtual consolidation.
CLOUDBERRY
Vinnova VINNVÄXT. Innovation project, focused on resource efficiency and green ICT for datacenters.
Key partners: IBM Haifa Research Labs, SAP, ATOS Orgin, Univ. Compl. de Madrid, Leeds Univ, Barcelona SC, Telefonica, British Telecom, Uppsala Univ., Luleå Univ. Tech., KTH, Lund Univ., etc
86
Erik Elmroth, elmroth@cs.umu.se


Holistic Management
Dynamic Rationing
Application Brownout
Coordination
Admission
Overbooking
Knowledge Base
Monitoring
Accounting
Priority
Energy
Autoscaling
Workload Analysis
Scaling Methods
Method Autoselect
Horizontal/Vertical
Placement
Intra Datacenter
Inter Datacenters
Structure Aware
VM Migration
Contextualization
www.cloudresearch.org
88


Senior researchers
Erik Elmroth, Professor
Francisco Hernandez, Assistant Professor
Johan Tordsson, Assistant Professor
P-O Östberg, Researcher
Post docs
Cristian Klein, PhD
Luis Tomás, PhD
Daniel Espling, PhD
PhD students
Ahmed Ali-Eldin, PhUc
Jakub Krzywda
Ewnetu Bayuh Lakew, PhUc
Wubin Li, PhUc
Mina Sedaghat, PhUc
Petter Svärd, PhUc
Kristen Selome Tesfatsion
Gonzalo Rodrigo
PhD students
Amardeep Mehta
Olumuyiwa Ibidunmoye
Others
Peter Gardfjäll, Proj. Dev, PhUc
Lars Larsson, System developer
Lennart Edblom, Senior lecturer
Tomas Forsman, Systems expert
Niclas Lodmer, Research assistant
www.cloudresearch.org
87
Erik Elmroth, elmroth@cs.umu.se

Printed by Media-Tryck, Lund University, Sweden | Photo: Charlotte Carlberg Bärg

LUND
UNIVERSITY

Department of Automatic Control
Box 118, 221 00 Lund, Sweden
www.lccc.lth.se

ISRN LUTFD2/TFRT--7639--SE
ISSN 0280-5316