Adaptive Resource Management Made Real

Årzén, Karl-Erik; Romero Segovia, Vanessa; Schorr, Stefan; Fohler, Gerhard

2011

Adaptive Resource Management Made Real

Karl-Erik Årzén, **Vanessa Romero Segovia**
Lund University,Lund, Sweden

Stefan Schorr, **Gerhard Fohler**
Technische Universität Kaiserslautern,Germany

Abstract

The adaptive resource management framework implemented in the European ACTORS project is presented. A resource manager has been developed that collaborates with a new Linux scheduler providing support for hard constant bandwidth server reservations, in order to adapt applications to changes in resource availability and to adapt the resource allocation to changes in application requirements. The focus of this paper is the three demonstrators developed based on the framework. The demonstrators presented are an adaptive video quality demonstrator, a feedback control demonstrator, and a video decoding demonstrator. All of these execute under the control of the resource manager.

1 Introduction

Multimedia is an increasingly important application area for many soft real-time systems in, e.g., mobile communication. Dataflow modeling and programming [13] for these types of applications is also receiving renewed interest, to a large extent depending on its explicit support for parallelism, something which is urgently needed today with the current trend towards multi-core and many-core platforms.

Resource management is another increasingly important area especially for battery-driven devices. As the system complexity increases the use of threads and priorities as the sole abstraction mechanism for ensuring real-time properties easily becomes unmanageable. A more promising alternative is to use bandwidth server or reservation techniques, e.g., constant-bandwidth servers (CBS) [1], that enforce the abstraction of *virtual processors (VP)* in which an application is guaranteed a certain execution budget every server period that can or cannot be exceeded (soft/ hard CBS). In order to cater for dynamic changes in resource requirements and in the available resource amount it is furthermore necessary to have adaptive resource reservations and to adapt the applications to changing amounts of resources. A mechanism for achieving this type of adaptivity is to use feedback.

In the European FP7 project ACTORS (Adaptivity and Control of Resources for Embedded Systems), [17], an adaptive resource management framework for Linux-based multi-core platforms has been developed. The framework is primarily intended for soft real-time applications implemented in the CAL dataflow language, [8], which recently has been adopted as a part of the MPEG-4 RVC standard for specifying media codecs. However, the framework also supports legacy applications through the use of application wrappers. An early version of the framework was presented in [22]. The contribution of the current papers is to provide more details on the implementation and to present the demonstrators that are based on the framework: an adaptive video quality demonstrator, a feedback control demonstrator, and a video decoding demonstrator.

The only resource considered in the current version is CPU time. The framework is based on CPU reservations [16] implemented through hard Constant Bandwidth Servers (CBS), which in turn is based on Earliest Deadline First (EDF) scheduling. By exe-

cuting one or several threads within the control of a bandwidth server it is guaranteed that the threads may not execute longer than what is given by the budget of the server, each server period. The support for hard CBS is provided by the new Linux scheduler SCHED_EDF developed within ACTORS. SCHED_EDF is a partitioned hierarchical scheduler, where threads are grouped into VPs characterized by budgets and periods. The partitioned nature of the scheduler implies that each VP may only span one physical processor.

1.1 Resource Manager Objectives

The resource manager (RM) has two main objectives. The first objective is to be able to adapt the applications to changes in resource availability. A reason for such a change could be that other more important and more resource demanding applications need to execute on the same platform. Application adaptation is achieved by requiring that applications can execute at different discrete service levels. In the different levels the required amount of resources and the obtained quality-of-service (QoS) are different. At the highest service level, service level 0, the application is assumed to obtain its maximum QoS and consume the largest amount of resources. As the service level decreases the QoS and the resource consumption are reduced.

The second objective is to be able to adapt how the resources are distributed when the application requirements change. This is achieved by measuring how many resources each application really consumes and dynamically adjusting the resource allocation accordingly. The latter typically involves changing the VP parameters, e.g., the server budget, based on how much budget that is currently required.

1.2 Related Work

The basic reasoning for the resource management in ACTORS is inspired by the MATRIX project [21, 20]. A number of architectures for end-to-end QoS for multimedia applications have been presented, e.g., [6, 11, 25, 23, 9], or see [3] for an overview. While architectures like [11] give an overall management system for end-to-end QoS, covering all aspects from user QoS policies to network handovers, in our work we focus on QoS management and resource adaptation in the application domain.

Comprehensive work on application-aware QoS adaptation has been reported in [11, 14]. Both separate between the adaptations on the system and application levels. While in [11] the application adjustment is actively controlled by a middle-ware control framework, in [14] this process is left to the application itself, based on requests from the underlying system. Classical control theory has been examined for QoS adaptation. [15] shows how an application can be controlled by a task control model. The method presented in [24] uses control theory to continuously adapt system behavior to varying resources. However, a continuous adaptation maximizes the global quality of the system but it also causes large complexity of the optimization problem. Instead, we propose adaptive QoS provision based on a finite number of quality levels.

The approach in this project has been to layer the adaptation on top of a bandwidth reservation system. Another approach is to integrate the adaptation with the bandwidth servers. One example of this is the variable-bandwidth servers proposed in [7]. A work that strongly relates to our work is the bandwidth adaptation performed within the Aquosa scheduler used in the European FRESCOR project, [2]. Also, here adaptation is layered on top of a CBS server. The main differences are that in FRESCOR the CBS servers used are of a soft nature and that only single core platforms are considered.

Resource reservations can be provided also using other techniques than bandwidth servers. One possibility is to use hypervisors, see e.g. [10] for an overview aimed at embedded applications, or to use resource management middleware or resource kernels, e.g., [18]. Resource reservations are also partly supported by the mainline Linux completely fair scheduler (CFS).

Adaptivity with respect to changes in requirements can also be provided using other techniques. One example is elastic task scheduling, e.g., [5], where tasks are treated as springs that can be compressed in order to maintain schedulability in spite of changes in

task rate. Another possibility is to support mode changes through different types of mode change protocols, e.g., [19]. A problem with this is that the task set parameters must be known both before and after the change, which is typically not the case for the types of applications considered in this paper.

This paper aims mainly at soft real-time application for which best-effort scheduling is sufficient. However, in [4] schedulability analysis for hard real-time applications modeled as DAGs and mapped to multiprocessor reservations of the type used in this paper is presented. Support for these types of hard real-time applications could quite easily be added to the current resource management framework.

2 Inputs and Outputs

The input to the RM can be split into static information made available either when a new application registers or at system start time, and dynamic information provided on-line to the RM. The static information from each application consists of the service level table and information about the threads of the application. An example of a service level table is shown in Table 1.

Table 1: Service level table for application A1

Application name	SL	QoS	BW [%]	TG [ms]	BWD [%]
A1	0	100	240	100	[60, 60, 60, 60]
	1	75	180	200	[45, 45, 45, 45]
	2	40	120	500	[30, 30, 30, 30]

In the table, SL denotes the service level index, BW the total amount of CPU required for a certain service level, i.e., on a quad-core platform the maximum value would be 400, TG the timing granularity, which indicates the time horizon over which the resources are needed and is currently used as the period of the VPs at the respective service level, and BWD informs the RM of how many VPs the application consists of and how the total bandwidth should be distributed among these. In addition to the service level table the application also informs the resource manager about the threads (thread IDs) that it consists of and how these threads should be grouped into VPs. In the table application A1 consists of four VPs. Each of the VPs contains at least one thread and execute within one physical core. The timing granularity, TG, is the same for all the VPs of an application, but may vary between the service levels, e.g., 100 ms at service level 0, 200 ms at service level 1, etc. The timing granularity is used as the period in the corresponding VPs. The budget of the VPs is given by the bandwidth multiplied by the period. For example, at service level 0 the budget of all the VPs of A1 will be 60 ms.

The static system-wide information consists of an importance value for each application or application class. The importance is used to determine the relative importance among the applications in the case of overload. A default value is given to applications that do not have any pre-defined value.

The dynamic information comes from the applications and from the SCHED_EDF scheduler. The applications provide a boolean measure, the *happiness* value, of whether the achieved quality corresponds to the expected quality for the current service level. The default value of the happiness is that the application is happy, i.e., an application that is not able to measure or calculate its obtained quality is considered to content with the resources it has received. SCHED_EDF measures the accumulated consumed budget for each VP and the accumulated number of server periods that the budget has been completely exhausted. The latter is used to indicate to what extent the bandwidth server "throttles" the execution of the application.

The outputs of the RM consists of the actual service level that the applications should execute at, and the VP parameters. These parameters consists of the server period, the server budget and the affinity of the VP. The period and budget are applied to the corresponding CBS server as well as to all the threads executing under the control of this server.

The overall structure of the RM is shown in Fig. 1.

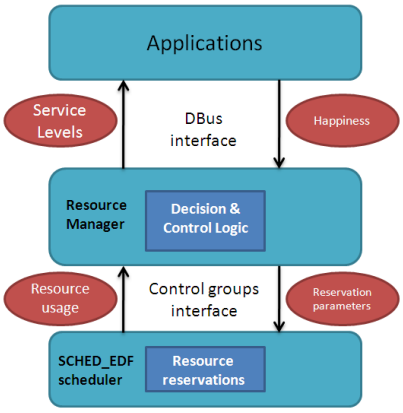

Figure 1: Overall structure of the ACTORS architecture.

3 Implementation

The RM is implemented in C++. It consists of two threads that execute within the same fixed-size reservation in one of the cores. The RM communicates with the applications through a DBus interface and with the underlying SCHED_EDF scheduler using the control groups API of Linux. The first thread handles incoming DBus messages containing, e.g., the service level table information which is sent when an application registers; happiness values for executing applications, and notifications that an application has terminated. The second thread periodically samples the VPs, measures the resource consumption, and invokes the bandwidth controllers.

The decision control logic is implemented as exchangeable classes. This makes it easy to experiment with alternative resource management policies. Currently the RM contains five different realizations of the logic. The control logic is invoked in the following four cases:

- When a new application registers.
- When an application terminates.
- When it is time to sample the VPs of an application. The sampling interval is 10 times the period of the VP.
- When the available amount of resources changes.

4 Control Logic

The five alternative realizations of the decision and control logic have several common features. Only one of them uses a heuristic for the service level assignment, all others formulate the problem as an ILP problem where the goal is to maximize the sum of the QoS weighted with the application importance, subject to the constraint that the sum of the total BW requirements must be smaller than total bandwidth available. The allocation of the VPs is a bin-packing problem solved with a first-fit heuristics with the objective to balance the load among the cores. In some of the realizations compression of the VP bandwidth is done in order to provide feasible solutions.

The bandwidth provided to each VP from the service level can be seen as a reserved amount of bandwidth, that the VP is guaranteed to obtain. However, as long as the VP is still happy the bandwidth controller may reduce the bandwidth of the VP. The aim of the controller is to keep the bandwidth assigned to the VP sufficiently close to the actual used bandwidth. How close it can be is decided by the Exhaustion Percentage (EP), which is defined as the percentage of the periods that the server budget is completely exhausted. If the EP is small the assigned bandwidth can be very close to the used bandwidth, whereas if the exhaustion percentage is large a larger distance must be kept in order to avoid unnecessary throttling.

The difference between the reserved and the allocated bandwidth can in most of the logic realizations only be reused by ordinary non SCHED_EDF, tasks. However, one of the logics allows a VP to use more bandwidth than what has been reserved for it as long as either there is bandwidth that is not currently used by any SCHED_EDF tasks or there are other SCHED_EDF tasks of less importance than the current one which are executing above their reserved bandwidth amount. In the latter case the VPs of the less important tasks are compressed. However, a VP may only be compressed as long as it is executing above its reserved bandwidth.

5 Tools

In ACTORS a number of tools have been developed. These includes a GUI that shows the internal workings of the RM, an application wrapper for legacy applications and an external load generator that mimics a CPU-intensive multi-core application. The GUI window is shown in Fig. 2. The top left corner con-

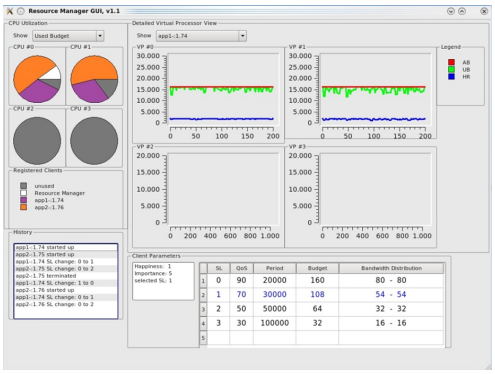

Figure 2: The GUI of the resource manager.

tains pie diagrams that show the allocations in each core, in this case the platform is a dual-core laptop. The plots show the used budget or bandwidth, the assigned budget or bandwidth, and the exhaustion percentage for the selected application. Below the plots the information in the service level table is shown as well as an event log.

6 Examples

In ACTORS three demonstrators were developed involving the RM: a feedback control demonstrator, a video quality adaptation demonstrator, and a video decoder demonstrator.

The feedback control demonstrator includes two feedback control applications implemented in CAL: one inverted pendulum controller and one ball and beam controller. The inverted pendulum controller consists of a pendulum held by an ABB IRB 2400 industrial robot. The objective of the controller is to automatically swing-up the pendulum and then balance the pendulum in its upward position. The aim of the ball and beam controller is to control the position of a ball rolling on a tilting beam. The inverted pendulum process is shown in Fig. 3 and the ball and beam process is shown in Fig. 4. Both applications

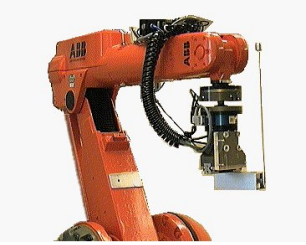

Figure 3: An inverted pendulum actuated by an industrial robot.

Figure 4: The ball and beam process. The control is based on sensors measuring the ball position and beam angle.

execute under the control of the RM. The different service levels correspond to different sampling periods of the controllers, where a higher service level implies faster sampling, higher bandwidth requirements, and better QoS, i.e., control performance.

The video quality adaptation demonstrator consists of a video player client executing under the control of the RM. The video player can either be implemented in CAL or be a legacy media player. The video stream is received over the network from a video server. When the available resources for the decoding decrease and it needs to lower its service level it issues a command to the video server to adapt the video stream by skipping frames, in the case of MPEG-2 streams [12], or by skipping macro block coefficients in the case of MPEG-4 streams.

The third demonstrator consists of a CAL MPEG4-

SP decoder connected to an Axis network camera that streams MPEG4-SP frames. The decoder has two partitions, three service levels, and can report its happiness value to the resource manager. When the decoder is required to switch to a lower service level it configures the camera to reduce the frames per second (fps) and resolution in order to reduce the resources required to decode the video frames. The happiness indicates if the resulting frame rate of the displayed video corresponds to what can be expected at the current service level.

In order to evaluate the behaviour of the decoder at different service levels a second application is introduced. This application corresponds to a CAL periodic pipeline with two partitions and three service levels. The importance value of the decoder and the pipeline applications correspond to 1 and 10 respectively, which implies that the pipeline application is more important than the decoder application.

Table 2 shows the service level information provided for the two applications during registration.

Table 2: Service level table of the decoder and pipeline applications

Application name	SL	QoS	BW [%]	TG [ms]	BWD [%]
Decoder	0	100	120	100	[60, 60]
	1	80	100	33	[50, 50]
	2	60	40	100	[20, 20]
Pipeline	0	100	80	20	[40, 40]
	1	90	54	30	[27, 27]
	2	70	32	50	[16, 16]

Figure 5 shows the used bandwidth UB (green), the assigned bandwidth AB (red) and the exhaustion percentage EP (blue) signals of the two virtual processors VP0 and VP1 of the decoder application. For this example the exhaustion percentage set point EP_{SP} was set to 15%.

At time $t = 0$ the decoder application registers with the resource manager, since there is no other application executing on the system, the resource manager assigns the highest service level 0 to the application, which corresponds to an initial assigned

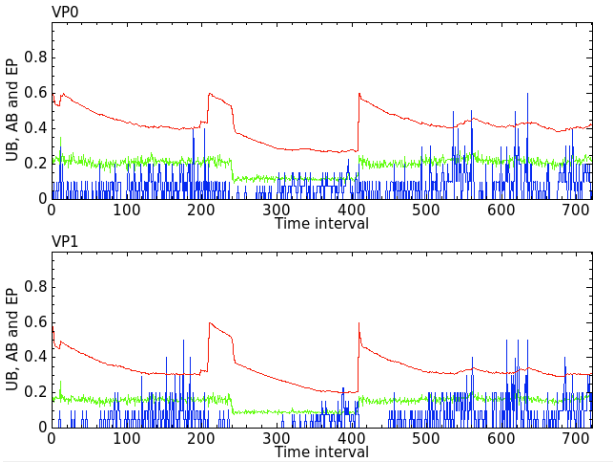

Figure 5: CAL SP decoder application.

bandwidth AB equal to 0.6. After registration the bandwidth controllers adapt the assigned bandwidth AB in each of the VPs trying to keep the EP below 0.15. If the EP is greater than 0.15 the bandwidth controllers increment the AB. The decoder application becomes unhappy at time $t = 10$ and $t = 210$ which causes the bandwidth controllers to increment the allocated bandwidth until the application is happy again. The periodic pipeline application registers with the resource manager at time $t = 240$. Since this application has higher importance than the decoder, the resource manager assigns service level 0 to the pipeline application and reduces the service level of the decoder application from 0 to 1. The initial assigned bandwidth of the decoder application at the new service level equals 0.5, which later on is decreased by the bandwidth controllers. Around time $t = 410$, the pipeline application unregisters, this increases the amount of free CPU resources, and triggers a new service level assignment for the decoder application, which in this case increases from service level 1 to service level 0.

7 Conclusions

Adaptive resource management in combination with reservation-based scheduling provides a flexible, and

yet predictable, platform for execution of a large class of real-time applications. In the ACTORS EC project a C++ based framework has been developed. Applications can either be implemented in the CAL dataflow language or consist of legacy software. The underlying scheduler is the new Linux scheduler SCHED_EDF.

The approach has been evaluated on different types of applications, from media streaming to feedback control. Although the current functionality of the resource manager is fully adequate for the applications where it has been used, there are several possible extensions:

- Support for power management. With relatively small means it would be possible to extend the resource manager to also support powering down cores, either using discrete power saving modes or using dynamic voltage and frequency scaling. Using power saving mode requires that the resource manager dynamically can pack the virtual processors onto as few physical processors as possible, making it possible to turn off the rest. The functionality for this, i.e., to be able to dynamically migrate virtual processors and their threads is already available. Using dynamic scaling techniques implies the possibility to dynamically modify the amount of available bandwidth, something that also is very straightforward.

- Model-free adaptation. The service level tables can be seen as models provided by the applications to the resource manager. Based on these, quite detailed models, the resource manager then performs the service level allocation. This can be viewed as a feedforward approach. The feedback then dynamically adjusts the size of the associated reservations. An interesting approach would be to instead have a completely feedback-based resource manager.

- Support for other resources than CPU time is clearly necessary if the framework should be applicable also to hard real-time applications.

7.1 Acknowledgments

The work has been done with partial support from the EC FP7 project ACTORS (Contract IST-216586).

References

[1] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the Real-Time Systems Symposium (RTSS), Madrid, Spain, 1998.*

[2] Luca Abeni, Tommaso Cucinotta, Giuseppe Lipari, Luca Marzario, and Luigi Palopoli. Qos management through adaptive reservations. *Real-Time Systems*, 29(2-3):131–155, 2005.

[3] C. Aurrecoechea, A. Campbell, and L. Hauw. A survey of QoS architectures. *Multimedia Systems*, 6:138–151, 1998.

[4] Giorgio Buttazzo, Enrico Bini, and Yifan Wu. Partitioning parallel applications on multiprocessor reservations. In *Proceedings of the 2010 22nd Euromicro Conference on Real-Time Systems*, ECRTS '10, pages 24–33, Washington, DC, USA, 2010. IEEE Computer Society.

[5] Giorgio C. Buttazzo, Marco Caccamo, and Luca Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51:289–302, 2002.

[6] G. Coulson, A. Campbell, and D. Hutchison. A quality of service architecture. *ACM SIGCOMM Computer Communication Review*, 24:6–27, 1994.

[7] S.S. Craciunas, C.M. Kirsch, H. Payer, H. Röck, and A. Sokolova. Programmable temporal isolation through variable-bandwidth servers. In *Proc. Symposium on Industrial Embedded Systems (SIES)*, pages 171–180. IEEE, 2009.

[8] J. Eker and J. Janneck. CAL Language Report. Technical Report ERL Technical Memo UCB/ERL M03/48, University of California at Berkeley, December 2003.

[9] M. Garca Valls, A. Alonso, J. Ruiz, and A. Groba. An architecture of a quality of service resource manager middleware for flexible embedded multimedia systems. In *Lecture Notes in Computer Science*, volume 2596, 2003.

[10] Gernot Heiser. The role of virtualization in embedded systems. In *Proceedings of the 1st workshop on Isolation and integration in embedded systems*, IIES '08, pages 11–16, New York, NY, USA, 2008. ACM.

[11] A. Kassler, A. Schorr, C. Niedermeier, R. Schmid, and A. Schrader. MASA - a scalable qos framework. In *Proceedings of Internet and Multimedia Systems and Applications (IMSA), Honolulu, USA*, 2003.

[12] Anand Kotra and Gerhard Fohler. Demo: Resource aware real-time stream adaptation for MPEG-2 transport streams in constrained bandwidth networks. In *The IEEE International Conference on Multimedia and Expo (ICME) 2010*.

[13] Edward A. Lee and David G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Trans. Comput.*, 36(1):24–35, 1987.

[14] B. Li and K. Nahrstedt. A control-based middleware framework for quality-of- service adaptations. *IEEE Journal on Selected Areas in Communications*, 1999.

[15] B. Li and K. Nahrstedt. Impact of control theory on QoS adaptation in distributed middleware systems. In *Proceedings of the American Control Conference*, 2001.

[16] C. W. Mercer, S. Savage, and H. Tokuda. Processor capacity reserves: Operating system support for multimedia applications. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems, 1994*.

[17] The ACTORS project webpage. http://www.actors-project.eu. URL, 2010.

[18] R. Rajkumar, K. Juvva, A. Molano, and S. Oikawa. Resource kernels: A resource-centric approach to real-time systems. In *In Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, 1998.

[19] Jorge Real and Alfons Crespo. Mode change protocols for real-time systems: A survey and a new proposal. *Real-Time Syst.*, 26:161–197, March 2004.

[20] Larisa Rizvanovic and Gerhard Fohler. The MATRIX: A framework for real-time resource management for video streaming in networks of heterogenous devices. In *The International Conference on Consumer Electronics 2007*.

[21] Larisa Rizvanovic, Damir Isovic, and Gerhard Fohler. Integrated global and local quality-of-service adaptation in distributed, heterogeneous systems. In *The 2007 IFIP International Conference on Embedded and Ubiquitous Computing*.

[22] Vanessa Romero Segovia, Karl-Erik Årzén, Stefan Schorr, Raphael Guerra, Gerhard Fohler, Johan Eker, and Harald Gustafsson. Adaptive resource management framework for mobile terminals - the ACTORS approach. In *Proceedings of the First International Workshop on Adaptive Resource Management (WARM), Stockholm, Sweden*, 2010.

[23] M. Shankar, M. De Miguel, and J.W.S. Liu. An end-to-end QoS management architecture. In *Real-Time Technology and Applications Symposium (RTAS)*, 1999.

[24] J.A Stankovic, T. Abdelzaher, M. Marleya, G. Tao, and S. Son. Feedback control scheduling in distributed real-time systems. In *Proceedings of the Real-Time Systems Symposium (RTSS)*, 2001.

[25] L. Xichen, C. Xiaomei, and W. Huaimin. The design of qos management framework based on corba a/v stream architecture. In *High Performance Computing in the Asia-Pacific Region*, 2000.