# Load Reduction for Timely Applications in the Cloud

Franco, Antonio; Landfeldt, Björn; Körner, Ulf; Nyberg, Christian

*Published in:*
IEEE Networking Letters

*DOI:*
10.1109/LNET.2021.3085115

2021

# Load Reduction for Timely Applications in the Cloud

Antonio Franco, Björn Landfeldt, Ulf Körner, Christian Nyberg

***Abstract*—In many IoT applications, sensor data is sent remotely to be processed, but only the freshest result is of interest. In this paper we investigate a feedback mechanism that aborts the processing of stale data at the remote end to reduce the load and save costs. The process is approximated by an M/M/$\infty$ queueing system with a feedback loop. We find the exact expression of the average computational time saved and show that with the feedback loop in place the computation time per CPU can be cut up to 25%, making the technique very promising.**

***Index Terms*—Age of Information; Queuing theory; Timely computation; Cloud computing.**

## I. Introduction

A current trend in the telecommunication industry is to shift the computation of tasks from the user devices, to data centers in the edge/cloud, especially in the field of the Internet of Things (IoT). While this gives a series of advantages to developers and content providers (platform independence, no need to distribute patches to all the users etc.), it also poses several challenges, especially for time sensitive applications; an application of the latter type may want to have many replicas of the same task being computed in a parallel fashion, and then just retrieve the replica that was computed first, discarding the others.

While the aforementioned scheme is of advantage for the user, if no mechanism is put in place to prevent stale replicas to continue being computed, this may result in wasted clock time at the data center side. By reducing the computational time per task, fewer servers can handle the same tasks, hence the total load of the data center is reduced. In general, we consider timely applications, where the time between the user interaction and the relative feedback should not exceed a certain threshold (e.g. Industrial IoT [1], digital twins [2], cloud gaming [3], haptic applications for the tactile internet [4]).

Timeliness in such systems is often analyzed through the Age of Information (AoI) metric, that is, a measure of how old is the information stored at a receiver end [5]. Specifically, we want to explore possible ways to reduce the load at the server/data center side. Particularly, the time a job spends

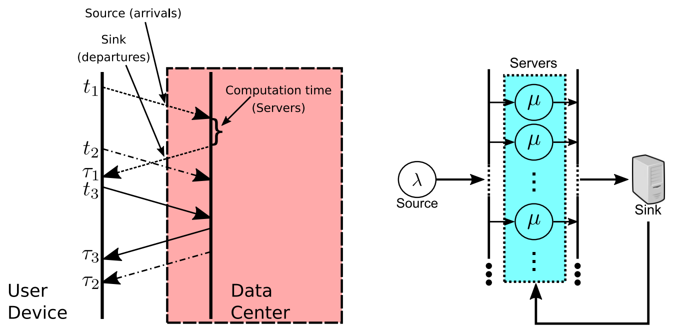
Manuscript submitted December 8, 2020 to IEEE Networking Letters. revised April 30, 2021; Accepted XXX, X 20XX. Date of publication XXX, X 20XX; date of current version XXX, X 20XX. The associate editor coordinating the review of this article and approving it for publication was XXXX.

The authors are with the Department of Electrical and Information Technology, Lund University, Box 118, 221 00 Lund, Sweden.

This work was supported by the "Excellence Center at Linköping-Lund in Information Technology (ELLIIT), Sweden"

Email: {antonio.franco, bjorn.landfeldt, ulf.korner, christian.nyberg}@eit.lth.se

Digital Object Identifier XX.XXXX/LNET.20XX.XXXXXXX



(a) (b)

Fig. 1. A user device offloading timely computations to a remote data center (a); the latter is approximated with an M/M/$\infty$ queuing system with a feedback loop (b). The user device sends the updates at times $t_i$, while receiving the corresponding results at times $\tau_i$.

in service, translates to clock time spent, e.g. at the edge. Since clock time is usually sold at a cost (by e.g. a cloud provider), the less service time is used, the better for the content provider/developer and the less the data center is loaded. Also, reducing the load at the cloud/edge side has an environmental benefit [6].

We then consider a system in which updates to be computed are sent by e.g. a device to a data center, which is loaded with a series of independent tasks, and therefore exhibit independent and exponentially distributed computation times. We also assume the interarrival times between updates to be independent and therefore exponentially distributed. The source can either be a user device, or it can be replicas of user tasks that are generated and distributed over several servers to decrease the expected finishing time for a computation. Any parallel computations that would finish after the first server, are aborted to save on computation time using the feedback loop (Fig. 1a). In accordance with previous work listed below, we approximate this system as an M/M/$\infty$ queuing system (Fig. 1b). We want to find the average computational time used by tasks. The aforementioned metric, if compared to the scenario without feedback loop, will give us a measure of the advantage of using this mechanism.

Timeliness in an M/M/$\infty$ queuing system was studied in [7], [8]. The first work found the exact expression of the average AoI, while the latter found its entire distribution. Timely applications in the edge/cloud were studied in [9]–[12]. Particularly, in the latter, a feedback loop is inserted as a means of reducing the load at the server side, but only the percentage of aborted tasks is considered as a metric of the advantage of having feedback. As far as the authors are aware this paper

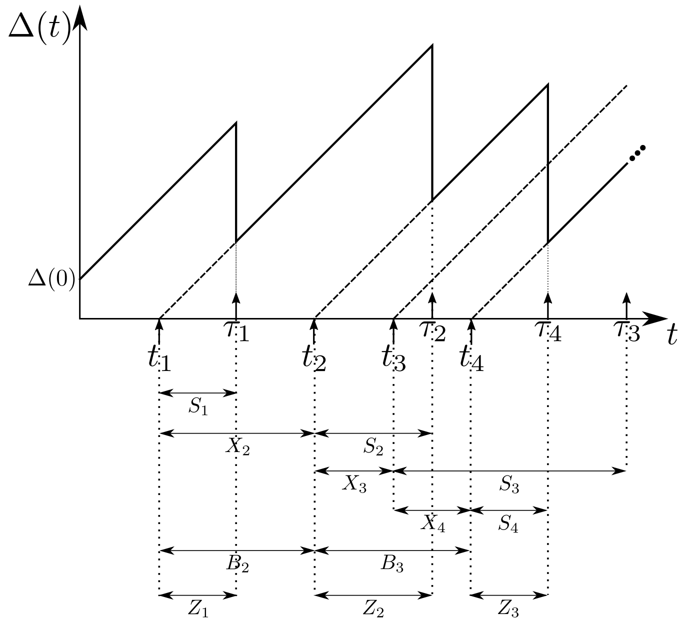


Fig. 2. A typical time period for an M/M/$\infty$ queueing system, where $\Delta(t)$ is the AoI at the sink side. Generation times are marked as $t_i$, while the corresponding departure times are marked as $\tau_i$.

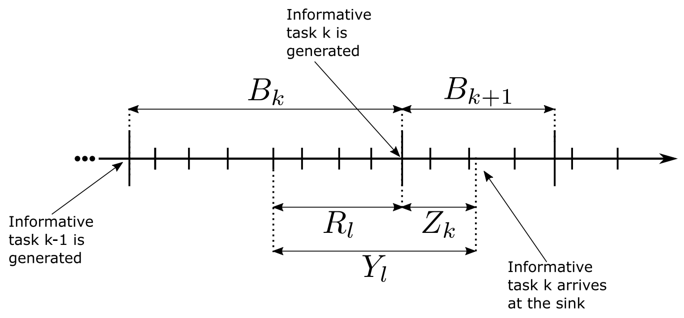
is the first addressing the percentage of computational time per task saved by aborting staler tasks in a massively parallel environment, thus quantifying the savings that can be achieved.

The remainder of this paper is organized as follows. In Section II the scenario is described in detail. In Section III the expression of the expected value for the service times per task is found. In Section IV the previous expression is compared with simulations and results analyzed. Finally in Section V conclusions and future work are discussed.

## II. Model description

The source generates pieces of information (i.e. tasks) with an average rate of $\lambda$ jobs per second. Each piece of information is timestamped with its generation time. The servers all serve jobs with an average rate of $\mu$ tasks per second, modeling a pool of independent jobs computed in a massively parallel environment. The sink acknowledges each received piece of information by broadcasting the latest information generation time-stamp to all the servers involved. Upon receiving the broadcast, each server checks whether it has a piece of information in service of the same age or older than the broadcasted generation time-stamp. If there is such a job in service, it is aborted to save computation time. The broadcast is assumed instantaneous; the latter assumption is based on the fact that a message passed between computational units in a data center takes a negligible amount of time with respect to the average computation time.We will call a task that is not discarded an informative task, while a task that is discarded, an obsolete task.

Both the inter generation times – described by the r.v. $X$ – and the service times – described by the r.v. $S$ – follow an exponential distribution with an average rate of $\lambda$ and $\mu$ tasks per second respectively. The source sends updates about



Informative task k is generated
$B_k$
$B_{k+1}$
Informative task k-1 is generated
$R_l$
$Z_k$
$Y_l$
Informative task k arrives at the sink

Fig. 3. Timeline for obsolete tasks between informative tasks.

a single information stream i.e. there is only one class of tasks. Each server in the model serves tasks by drawing service times from the same exponential distribution with rate $\mu$ tasks per second.

It is also worth mentioning that for the remainder of the paper the Probability Density Function (PDF) of a random variable $X$ will be expressed as $f_X(x)$, and its Cumulative Distribution Function (CDF) as $F_X(x) = \Pr\{X \leq x\}$. For clarity, we will sometimes abuse the probability notation and write, e.g. $\Pr\{X = x\} \triangleq f_X(x)$, even if $X$ is a continuous random variable. Also, we will indicate a multivariate random variable of dimension $b - a$ as $\mathbf{X}_a^b$, where one or both the extremes could be infinite, and its outcome as $\mathbf{x}_a^b$. Finally, unless stated otherwise, all the random variables have non negative support.

In Fig. 2 a typical time period is shown, along with the AoI $\Delta(t)$, i.e. the difference between the current time, and the time of generation of the last informative task received. Generation times are marked as $t_i$, while the corresponding departure times (i.e. the times when the sink receives the task) are marked as $\tau_i$. Task 1 is generated at $t_1$, and arrives at the sink after a time $S_1$, at the instant $\tau_1$. The AoI will then jump to the service time experienced by task 1. Then it will continue to grow with slope 1, until task 2 arrives at the sink, where it again jumps to its service time $S_2$. Notice that, since task 3 is generated before task 4, but arrives after the latter, it is discarded, i.e., is an obsolete task. The time between the arrival of two informative tasks to the M/M/$\infty$ queueing system we call the *effective inter-generation time*, described by the random variable $B$. Also, the service time experienced by an informative task we call the *effective service time*, and the random variable that describes it is $Z$.

## III. Average computational time per task

First, we note that the joint PDF of $n$ inter-arrivals $f_{\mathbf{X}_1^n}(t)$ is the product of the PDFs of $n$ independent and identically distributed (i.i.d.) exponential random variables, i.e.:

$$f_{\mathbf{X}_1^n}(\mathbf{x}_1^n) = \prod_{k=1}^{n} f_X(x_k) = \lambda^n \mathrm{e}^{-\lambda \sum_{k=1}^{n} x_k}.$$

The random variable describing the inter-arrival times after task $i$, is a vector, with non negative support, indicated as $\mathbf{X}_{i+1}^{\infty} = \{X_{i+1}, X_{i+2}, \ldots\}$.

We refer to Fig. 3. We suppose that the $i$-th generated task is informative and renders $n$ tasks obsolete. It is the $k$-th informative task, and the $i$-th task from the start of our system. The spacing between informative task $k-1$ and informative task $k$ (i.e. task $i$) is described by the random variable $B_k$. We concentrate on task $i-l-1$, with $l<n$. Its service time $s_{i-l-1}$ ends after the service time of the next informative task $s_i$. Being $s_i$ the realization of the random variable describing the service time for informative tasks $Z$, we will refer to its underlying random variable as $Z_k$. When, after $l$ arrivals plus $s_i$, the next informative task arrives, task $i-l-1$ is aborted. This means that the computation time between its arrival and the arrival at the sink of the next informative task, is the computational time effectively used by the task. We describe this quantity with the random variable $Y_l$. $Y_l$ is the sum of the r.v. describing the next $l$ arrivals $R_l$, and the service time of the next informative task $Z_k$. Our goal is to calculate the average server occupancy per task in our scheme, described by the random variable $Y$; in order to calculate the expected value of $Y$, we first need to calculate:

$$\mathrm{E}\left[Y_l\right]=\mathrm{E}\left[R_l\right]+\mathrm{E}\left[Z_l\right], \tag{1}$$

where we renamed $Z_k$ to $Z_l$, being it the r.v. describing the service time for an informative task $i$ that has rendered at least $l$ previous tasks obsolete, so, under our steady state assumptions, is independent of the task number.

In order to calculate $\mathrm{E}\left[R_l\right]$, we define the event $E_1(i)$ that the task $i$ is informative and the event $E_2(l)$, meaning that a task $i$ has rendered at least $l$ previous tasks obsolete. Note that, as found in [7, Appendix A], $E_1(i)$ is independent of the task number $i$, so is $E_2(l)$. Additionally, we define $\mathcal{E}(l)=E_1(i)\cap E_2(l)$, that is, the event where informative task $i$ has rendered at least $l$ previous tasks obsolete. We first write the joint distribution of the $l$ inter-arrivals before task $i$ given that the next informative task is task $i$ as:

$$\begin{aligned}&f_{\mathbf{X}_{i-l+1}^{i}\mid\mathcal{E}(l)}\left(\mathbf{x}_{i-l+1}^{i}\right)\\&\quad=\frac{f_{\mathbf{X}_{i-l+1}^{i}}\left(\mathbf{x}_{i-l+1}^{i}\right)}{\Pr\left\{\mathcal{E}(l)\right\}}\Pr\left\{\mathcal{E}(l)\mid\mathbf{X}_{i-l+1}^{i}=\mathbf{x}_{i-l+1}^{i}\right\},\end{aligned} \tag{2}$$

where:

$$\begin{aligned}&\Pr\left\{E_2(l)\mid S_i=s_i,\mathbf{X}_{i-l+1}^{i}=\mathbf{x}_{i-l+1}^{i}\right\}\\&\quad=\Pr\left\{\bigcap_{l_1=0}^{l-1}\left(S_{i-l_1-1}>s_i+\sum_{k=0}^{l_1}x_{i-k}\right)\right\}\\&\quad=\mathrm{e}^{-\mu l s_i-\mu\sum_{k=1}^{l}k x_{i-l+k}}.\end{aligned} \tag{3}$$

From [7, Eq. (5)] we know:

$$\begin{aligned}&\Pr\left\{E_1(i)\mid S_i=t,\mathbf{X}_{i+1}^{\infty}=\mathbf{x}_{i+1}^{\infty}\right\}\\&\quad=\mathbb{1}\left\{t<x_{i+1}\right\}+\sum_{r=1}^{\infty}\left(\mathrm{e}^{-\mu\left(rt-\sum_{k=1}^{r}(r-k+1)x_{i+k}\right)}\right.\\&\quad\left.\times\,\mathbb{1}\left\{\sum_{k=1}^{r}x_{i+k}<t<\sum_{k=1}^{r+1}x_{i+k}\right\}\right).\end{aligned} \tag{4}$$

where $\mathbb{1}\{E\}$ is the indicator function, defined as:

$$\mathbb{1}\{E\}=\begin{cases}1 & ,E\text{ is true}\\0 & ,E\text{ is false}\end{cases}.$$

Noticing that (3) and (4) are conditionally independent given $S_i$, $\mathbf{X}_{i-l+1}^{i}\cap\mathbf{X}_{i+1}^{\infty}\equiv\varnothing$ and that all the interarrival times are i.i.d. we can write:

$$\begin{aligned}&\Pr\left\{\mathcal{E}(l)\mid\mathbf{X}_{i-l+1}^{i}=\mathbf{x}_{i-l+1}^{i}\right\}\\&\quad=\int_0^{+\infty}\underbrace{\int_0^{+\infty}\cdots\int_0^{+\infty}}_{|\mathbf{x}_{i+1}^{\infty}|\text{ times}}\Pr\left\{E_2(l)\mid S_i=s_i,\mathbf{X}_{i-l+1}^{i}=\mathbf{x}_{i-l+1}^{i}\right\}\\&\quad\times\Pr\left\{E_1(i)\mid S_i=s_i,\mathbf{X}_{i+1}^{\infty}=\mathbf{x}_{i+1}^{\infty}\right\}f_{\mathbf{X}_{i+1}^{\infty}}\left(\mathbf{x}_{i+1}^{\infty}\right)f_S(s_i)\\&\quad\mathrm{d}\mathbf{x}_{i+1}^{\infty}\mathrm{d}s_i=\rho^{-(\rho+l+1)}\mathrm{e}^{\rho-\mu\sum_{k=1}^{l}kx_{i-l+k}}\gamma\left(\rho+l+1,\rho\right).\end{aligned} \tag{5}$$

where $|\mathbf{x}|$ represents the cardinality of the (possibly infinite) set $\mathbf{x}$, $\rho=\lambda/\mu$ is the server load, and $\gamma(s,x)$ is the lower incomplete gamma function:

$$\gamma(s,x)=\int_0^x t^{s-1}\,\mathrm{e}^{-t}\,\mathrm{d}t.$$

By averaging the previous over the $l$ inter-arrivals:

$$\begin{aligned}\Pr\left\{\mathcal{E}(l)\right\}&=\underbrace{\int_0^{+\infty}\cdots\int_0^{+\infty}}_{l\text{ times}}\Pr\left\{\mathcal{E}(l)\mid\mathbf{X}_{i-l+1}^{i}=\mathbf{x}_{i-l+1}^{i}\right\}\\&\quad\times f_{\mathbf{X}_{i-l+1}^{i}}\left(\mathbf{x}_{i-l+1}^{i}\right)\mathrm{d}\mathbf{x}_{i-l+1}^{i}\\&\quad=\frac{\rho^{-(\rho+1)}\mathrm{e}^{\rho}\gamma\left(\rho+l+1,\rho\right)\Gamma(\rho+1)}{\Gamma\left(\rho+l+1\right)}.\end{aligned} \tag{6}$$

By substituting the previous and (5) in (2) we obtain:

$$\begin{aligned}f_{\mathbf{X}_{i-l+1}^{i}\mid\mathcal{E}(l)}\left(\mathbf{x}_{i-l+1}^{i}\right)&=\frac{\mu^l\Gamma\left(\rho+l+1\right)}{\Gamma(\rho+1)}\mathrm{e}^{-\sum_{k=1}^{l}(\lambda+k\mu)x_{i-l+k}}\\&=\prod_{k=1}^{l}\left[\frac{1}{\lambda+k\mu}\mathrm{e}^{-(\lambda+k\mu)x_{i-l+k}}\right].\end{aligned}$$

We notice that the previous joint distribution is the product of all the marginals, thus all the involved random variables are independent. Then, the expected value of the sum will be the sum of the expected value of each marginal, that is, an exponential with rate $\lambda+k\mu$; if we call $A_l$ the r.v. representing the sum of the next $l$ inter-arrivals, we have:

$$\begin{aligned}\mathrm{E}\left[R_l\right]&=\mathrm{E}\left[A_l\mid\mathcal{E}(l)\right]=\mathrm{E}\left[\sum_{k=1}^{l}X_{i-l+k}\middle|\,\mathcal{E}(l)\right]\\&=\sum_{k=1}^{l}\mathrm{E}\left[X_{i-l+k}\mid\mathcal{E}(l)\right]=\sum_{k=1}^{l}\frac{1}{\lambda+k\mu}\\&=\frac{1}{\mu}\left[\psi(\rho+l+1)-\psi(\rho+1)\right],\end{aligned} \tag{7}$$

where we took advantage of the recurrence relation of the digamma function $\psi(x)$ defined as the derivative of the logarithm of the gamma function. Notice how in (7) $\mathrm{E}\left[A_l\mid\mathcal{E}(0)\right]=0$, as expected, being it the aborted service time for an informative task.

In order to find $\mathrm{E}\left[Z_l\right]$, we use a reasoning similar to (5), but we instead average over all the arrivals; i.e.:

$$\Pr\left\{\mathcal{E}(l)\mid S_i=s_i\right\}$$

$$= \underbrace{\int_0^{+\infty} \cdots \int_0^{+\infty}}_{|\mathbf{x}_{i-l+1}^{\infty}| \text{ times}} \Pr\left\{E_2(l) | S_i = s_i, \mathbf{X}_{i-l+1}^{i} = \mathbf{x}_{i-l+1}^{i}\right\}$$
$$\times \Pr\left\{E_1(i) | S_i = s_i, \mathbf{X}_{i+1}^{\infty} = \mathbf{x}_{i+1}^{\infty}\right\} f_{\mathbf{X}_{i-l+1}^{\infty}}\left(\mathbf{x}_{i-l+1}^{\infty}\right)$$
$$\mathrm{d}\mathbf{x}_{i-l+1}^{\infty} = \frac{\rho^l \Gamma(\rho+1)}{\Gamma(\rho+l+1)} \mathrm{e}^{\rho - [\lambda + l\mu] s_i - \rho \mathrm{e}^{-\mu s_i}}.$$

We then use Bayes along with (6), in order to find the PDF of $Z_l$, i.e.:

$$f_{Z_l}(s_i) = \frac{\mu \rho^{\rho+l+1}}{\gamma(\rho+l+1,\rho)} \mathrm{e}^{-[\lambda+(l+1)\mu]s_i - \rho \mathrm{e}^{-\mu s_i}}.$$

Finally, we can directly calculate the expected value of $Z_l$ from the previous, i.e.:

$$\begin{aligned} \mathrm{E}[Z_l] &= \frac{\mu \rho^{\rho+l+1}}{\gamma(\rho+l+1,\rho)} \int_0^{\infty} s_i \mathrm{e}^{-(\lambda+\mu)s_i - \rho \mathrm{e}^{-\mu s_i}} \mathrm{d}s_i \\ &= -\frac{\rho^{\rho+l+1}}{\mu(\rho+l+1)^2 \gamma(\rho+l+1,\rho)} \int_0^1 \ln(y) y^{\rho+l} \mathrm{e}^{-\rho y} \mathrm{d}y \\ &= \frac{\rho^{\rho+l+1}}{\mu(\rho+l+1)^2 \gamma(\rho+l+1,\rho)} \\ &\times {}_2F_2\left(\rho+l+1, \rho+l+1; \rho+l+2, \rho+l+2; -\rho\right), \quad (8) \end{aligned}$$

where, in order to solve the last integral, we used the recurrence integral relation of the generalized hyper-geometric function [13, Theorem 38].

In order to de-condition $\mathrm{E}[Y_l]$, we need to find the probability of being exactly $l$ tasks from the next informative task $i$, represented by the event $E_3(l)$. We notice that the event $E_3(l > 0) \equiv \mathcal{E}(l)$. So, simply, by using (6) for $l > 0$:

$$\Pr\{E_3(l>0)\} = \frac{\rho^{-(\rho+1)} \mathrm{e}^{\rho} \gamma(\rho+l+1,\rho)\Gamma(\rho+1)}{\Gamma(\rho+l+1)}. \quad (9)$$

For $l = 0$ we proceed similarly as in (5):

$$\begin{aligned} \Pr\{E_3(0)\} &= \int_0^{+\infty}\int_0^{+\infty}\underbrace{\int_0^{+\infty}\cdots\int_0^{+\infty}}_{|\mathbf{x}_{i+1}^{\infty}| \text{ times}} \Pr\{S_{i-1} < x_i + s_i\} \\ &\times \Pr\left\{E_1(i) | S_i = s_i, \mathbf{X}_{i+1}^{\infty} = \mathbf{x}_{i+1}^{\infty}\right\} f_{\mathbf{X}_{i+1}^{\infty}}\left(\mathbf{x}_{i+1}^{\infty}\right) \\ &\times f_X(x_i) f_S(s_i) \mathrm{d}\mathbf{x}_{i+1}^{\infty} \mathrm{d}s_i \mathrm{d}\mathbf{x}_i \\ &= \rho^{-(\rho+1)} \mathrm{e}^{\rho} \gamma(\rho+1,\rho) \left(1 - \frac{1}{\lambda+\mu}\right). \end{aligned}$$

By combining the previous with (9):

$$\begin{aligned} &\Pr\{E_3(l)\} \\ &= \frac{\rho^{-(\rho+1)} \mathrm{e}^{\rho} \gamma(\rho+l+1,\rho)\Gamma(\rho+1)}{\Gamma(\rho+l+1)} \left(1 - \frac{\delta_{l,0}}{\lambda+\mu}\right), \end{aligned} \quad (10)$$

where $\delta_{i,j}$ is the Kronecker delta defined as:

$$\delta_{i,j} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}.$$

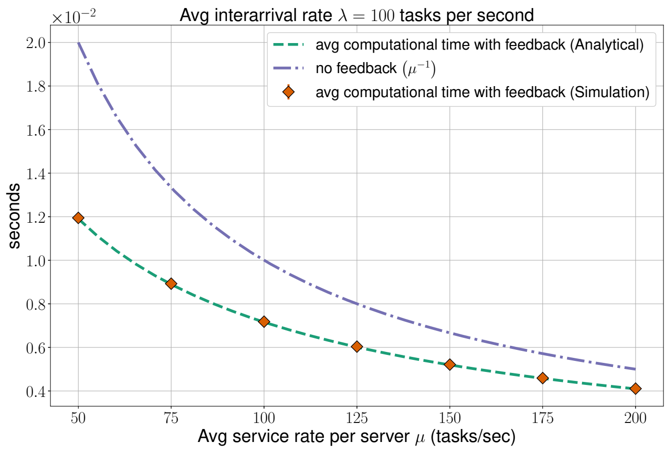


Fig. 4. Average computational time per task with and without feedback loop (11); Simulation vs analytical.

Finally, the average service time per task $\mathrm{E}[Y]$ in our system, by using (7) and (8) in (1), and (10) to average over all $l$s, will be:

$$\begin{aligned} \mathrm{E}[Y] &= \sum_{l=0}^{\infty} \mathrm{E}[Y_l] \Pr\{E_3(l)\} \\ &= \sum_{l=0}^{\infty} \left[ \frac{\rho^{\rho+l+1}}{\mu(\rho+l+1)^2 \gamma(\rho+l+1,\rho)} \right. \\ &\times {}_2F_2\left(\rho+l+1, \rho+l+1; \rho+l+2, \rho+l+2; -\rho\right) \\ &+ \left. \frac{1}{\mu}\left[\psi(\rho+l+1) - \psi(\rho+1)\right] \right] \\ &\times \frac{\rho^{-(\rho+1)} \mathrm{e}^{\rho} \gamma(\rho+l+1,\rho)\Gamma(\rho+1)}{\Gamma(\rho+l+1)} \left(1 - \frac{\delta_{l,0}}{\lambda+\mu}\right). \end{aligned} \quad (11)$$

## IV. Numerical Results

TABLE I
Parameters used in the numerical study.

| Symbol | Description |
|---|---|
| $\mu$ | Tasks processed by a computational unit in the remote data center per second without a feedback loop in place (average) |
| $\lambda$ | Tasks sent by the device to the remote data center per second (average) |
| $\mathrm{E}[Y]$ | Average time (in seconds) a task spends in computation with the feedback in place |

We conducted simulation studies using OMNeT++ [14]. All plots involving simulations are allowed for a sufficient warm-up period before taking measurements. Confidence intervals are too tight to show at 95% confidence. All the plots make use of a black and white printer-friendly and accessible color scheme [15]. The parameters used in the numerical study are described in Table I.

In Fig. 4 we plotted (11) – i.e. the average service time per task $\mathrm{E}[Y]$ – along with the expected service time for tasks without a feedback loop – i.e. $\mu^{-1}$ seconds – for a broad range of average service rates; we fixed the interarrival rate $\lambda$ = 100 tasks per second and let the service rate vary between

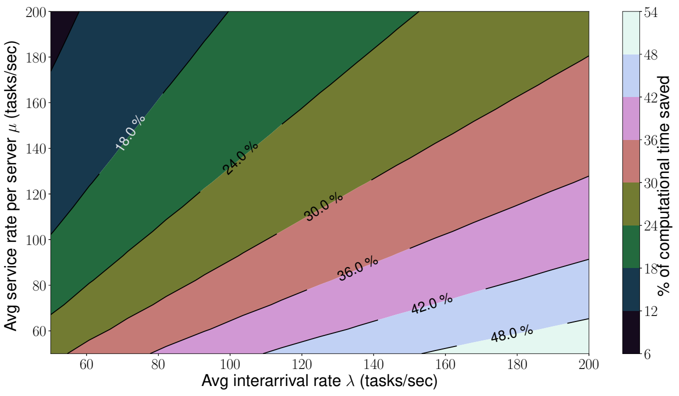


Fig. 5. Contour plot, where the isolevel lines are the percentage of computational time per task saved with the feedback loop in place (12).

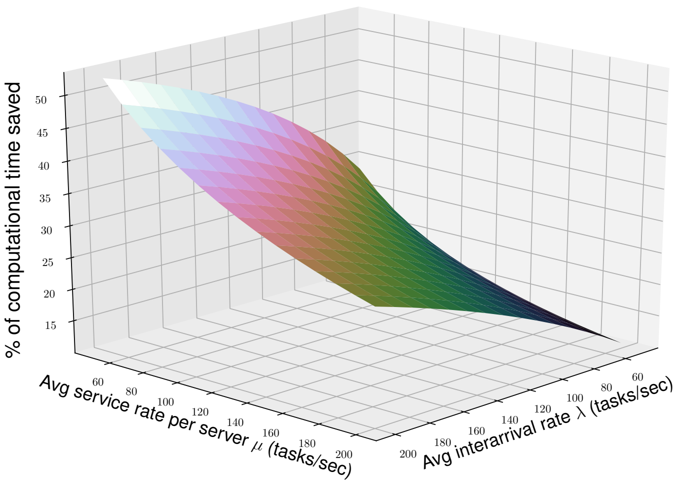


Fig. 6. Percentage of computational time per task saved with the feedback loop in place (12), isometric view.

50 and 200 tasks per second. We find that (11) agrees with the simulations.

In order to highlight the benefit of the feedback loop we also did a contour plot of the percentage of computational time per task saved with the feedback loop in place (Fig. 5) i.e.:

$$\texttt{\% saved} = (1 - \mu \mathrm{E}[Y]) \cdot 100 \ . \tag{12}$$

The rates $\lambda$ and $\mu$ both span between 50 and 200 tasks per second. The corresponding isometric view is presented in Fig. 6. As we can see from the figures, substantial advantages in terms of saved computational time can be reached when the service rate per server is low in comparison to the interarrival rate. By combining the results in [8] and Fig. 5, it is possible to lower the requirements on the speed of the computational units (i.e. purchase cheaper CPUs), while mantaining the same specifications on timeliness; particularly, at high loads, it is possible to scale down the data center significantly (a saving of at least 25% per CPU).

## V. Conclusion and future work

In this paper we studied a mechanism to reduce the load when timely updates have to be processed in a remote location. We examined a situation in which a user is sending updates to be processed by a server/data center, where computation of parallel tasks can be assumed independent. Specifically, we assumed that only the freshest update is of interest to the user, thus rendering the computation of staler tasks only a burden to the data center.

We investigated the properties of such a system by studying an M/M/$\infty$ system in which the sink has a feedback loop to the infinite servers, and immediately broadcast the timestamp of generation of the update that triggered the computation that was just completed, thus causing the immediate stop of jobs relative to staler updates.

We found the exact expression of the percentage of computational time per task saved with the feedback loop in place, as opposed to the average service time in an M/M/$\infty$ system, and shown its advantages. Future work will involve studying the effect of an eventual delay in the feedback loop, and extend the analysis to a G/G/$\infty$ system.

## References

[1] S. Chen, Y. Zheng, K. Wang, and W. Lu, "Delay guaranteed energy-efficient computation offloading for industrial IoT in fog computing," in *Proc. ICC 2019 - IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6.

[2] A. El Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE Multimedia*, vol. 25, no. 2, pp. 87–92, 2018.

[3] Z. Wen and H. Hsiao, "QoE-driven performance analysis of cloud gaming services," in *Proc. 2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2014, pp. 1–6.

[4] M. Dohler, T. Mahmoodi, M. A. Lema, M. Condoluci, F. Sardis, K. Antonakoglou, and H. Aghvami, "Internet of skills, where robotics meets AI, 5G and the tactile internet," in *Proc. 2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–5.

[5] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. 2012 IEEE INFOCOM*, March 2012, pp. 2731–2735.

[6] M. A. Inamdar and H. V. Kumaraswamy, "Energy efficient 5G networks: Techniques and challenges," in *Proc. 2020 International Conference on Smart Electronics and Communication (ICOSEC)*, 2020, pp. 1317–1322.

[7] C. Kam, S. Kompella, G. D. Nguyen, and A. Ephremides, "Effect of message transmission path diversity on status age," *IEEE Trans. Inform. Theory*, vol. 62, no. 3, pp. 1360–1374, March 2016.

[8] Y. Inoue, "The probability distribution of the AoI in queues with infinitely many servers," in *Proc. IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 297–302.

[9] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely-throughput optimal coded computing over cloud networks," in *Proc. of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 301–310. [Online]. Available: https://doi.org/10.1145/3323679.3326528

[10] L. Liu, X. Qin, Y. Tao, and Z. Zhang, "Timely updates in MEC-assisted status update systems: Joint task generation and computation offloading scheme," *China Communications*, vol. 17, no. 8, pp. 168–186, 2020.

[11] B. Buyukates and S. Ulukus, "Timely distributed computation with stragglers," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5273–5282, 2020.

[12] A. Franco, E. Fitzgerald, B. Landfeldt, and U. Körner, "Reliability, timeliness and load reduction at the edge for cloud gaming," in *Proc. 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC) (IPCCC 2019)*, London, United Kingdom (Great Britain), Oct. 2019.

[13] E. D. Rainville, *Special Functions*. New York: The Macmillan Co., 1960.

[14] A. Varga, "The OMNET++ discrete event simulation system," in *Proc. ESM'01*, 2001.

[15] Cynthia Brewer, Mark Harrower and The Pennsylvania State University. (2013) ColorBrewer. http://colorbrewer2.org.