Patience-Aware Scheduling for Cloud Services: Freeing Users from the Chains of Boredom*

Carlos Cardonha[1], Marcos D. Assunção[1], Marco A.S. Netto[1], Renato L.F. Cunha[1], and Carlos Queiroz[2]

[1] IBM Research Brazil
[2] IBM Research Australia

Abstract. Scheduling of service requests in Cloud computing has traditionally focused on the reduction of pre-service wait, generally termed as waiting time. Under certain conditions such as peak load, however, it is not always possible to give reasonable response times to all users. This work explores the fact that different users may have their own levels of tolerance or patience with response delays. We introduce scheduling strategies that produce better assignment plans by prioritising requests from users who expect to receive results earlier and by postponing servicing jobs from those who are more tolerant to response delays. Our analytical results show that the behaviour of users' patience plays a key role in the evaluation of scheduling techniques, and our computational evaluation demonstrates that, under peak load, the new algorithms typically provide better user experience than the traditional FIFO strategy.

1 Introduction

Traditionally, job schedulers do not take into account how users interact with services. They optimise system metrics, such as resource utilisation and energy consumption, and user metrics such as response time. However, understanding interactions between users and a service provider over time allows for custom optimisations that bring benefits for both parties.

In this article we propose scheduling strategies that take into account users' expectations regarding response time and their patience when interacting with Cloud services. Such strategies are relevant mainly to handle peak load conditions without the need to allocate additional resources for the service provider. Although elasticity is common in a Cloud setting, resources may not be available quickly enough and their allocation can incur additional costs that may be avoidable. The main contributions of this paper are: (i) a **Patience-Aware Scheduling (PAS)** strategy and an **Expectation-Aware Scheduling (EAS)** strategy for Cloud systems; (ii) Analytical comparisons between the **EAS** strategy and the traditional **First-In, First-Out (FIFO)** scheduling strategy; (iii) Evaluation of the proposed strategies and a discussion on when they bring benefits for users and service providers.

* Extended version of this paper is available at `arxiv.org`

2 Proposed Scheduling Strategies

This work considers Cloud services that support applications running on mobile devices and desktops, most of which are highly interactive and iterative. Service performance over time usually shapes the users' expectations on how it is likely to perform in the future. The provider stores information on how its service responded to user requests and uses this information to gauge her expectations and patience.

PAS and **EAS** utilise user expectation to schedule service requests. Both strategies share the following common goals: (i) minimise the number of users abandoning the service; (ii) maximise the users' level of happiness with the service; and (iii) perform such optimisations without adding new resources to the service. An incoming job request will be directly assigned if there are available resources in the service provider. Therefore, choosing among **FIFO**, **PAS**, and **EAS** becomes more crucial during peak load.

PAS has the goal of serving first users whose patience levels are the lowest when interacting with the Cloud service. When new requests arrive, the algorithm sorts the tasks in its waiting queue according to the *Patience* of their users (in ascending order), and when a new resource is freed, the request positioned in the head of the waiting list is assigned to it. An adequate estimate of how the user's happiness level and the user's tolerance curves behave is very important for the evaluation of the proposed strategies. In our implementation of **PAS** and in our computational evaluation, we employed the definition of Brown *et al.* [6], where patience is given by the ratio of the time a user expects to wait for results to the time the user actually waits for them.

EAS has the goal of serving first requests from users whose response time expectations are translated into "soft" deadlines that are positioned earlier in time. The difference between **EAS** and traditional deadline-based algorithms lies in the nature of the "buffer" adding to the minimum response time, as it changes over time and is related to users' patience. **EAS** sorts service requests in the waiting queue according to their users' expectation, which is the sum of arrival time and expected response time, where *arrival time* is the time at which the job arrived on the waiting queue and *expected response time* is the time that the service provider need to complete the task. **EAS**, then, schedules the job with the least *expectation* when a new resource is freed.

2.1 Analytical Investigation of the EAS Strategy

Let $\mathcal{U}$ be the set of users of a service provider. Let $\mathcal{T}$ denote the sequence of job requests being submitted, where each $t \in \mathcal{T}$ arrives at time $a(t) \in \mathbb{R}^+$ and has processing time $\Delta(t) \in \mathbb{R}$. Task t is submitted by user $u(t)$, who is expecting to wait an amount of time $w(u(t)) \in \mathbb{N}$ in addition to $\Delta(t)$, *i.e.*, $w(u)$ denotes u's tolerance with response delays. The service provider has a dispatching algorithm responsible for the assignment of each incoming task to one of its m processors.

Let us denote by $s(t) \in \mathbb{R}^+$ the time at which task t starts to be processed. The response time for task t is given by $r(t) = (s(t) - a(t)) + \Delta(t)$, and $e(u(t), t) =$

$r(t) - (\Delta(t) + w(u(t)))$ denotes the amount of time by which the response time differs from $u(t)$'s original expectation.

We denote user u's level of happiness by $h(u) \in [0, 1]$, a linear scale where $h(u) = 0$ and $h(u) = 1$ indicates that u is absolutely discontent and happy, respectively. We assume that u stops sending requests as soon as $h(u)$ is below some critical value $c(u)$ in $[0, 1]$. User u is active if $h(u) > c(u)$. The impact that $e(u, t)$ has on $h(u)$ is formulated by function $i : \mathcal{U} \times \mathbb{R} \to \mathbb{R}$, and the impact that $e(u, t)$ has on $w(u)$ is described by some function $j : \mathcal{U} \times \mathcal{T} \times \mathbb{R} \to \mathbb{R}$. If we assume that $i(u, e(u, t))$ and $j(u, t, e(u, t))$ are addictive factors, then, after the computation of some task t, the happiness level of user $u(t)$ will be given by $h(u) + i(u, e(u, t))$, while $u(t)$'s patience level becomes $w(u(t)) + j(u(t), t, e(u(t), t))$.

Optimisation Criteria. Let Z denote the closed interval $[0, 1] \subset \mathbb{R}$. We say that a vector $s \in Z^{|\mathcal{U}|}$ denotes a service provider's *user happiness state* if $s_x = h(u_x)$ $\forall u_x \in \mathcal{U}$, $1 \le x \le \mathcal{U}$. In order to evaluate and compare different scheduling strategies, we have to define a cost function $c : Z^{|\mathcal{U}|} \to \mathbb{R}$. The definition of a proper cost function depends on the optimisation criteria one wants to establish. We will consider two optimisation goals. The first one is the *maximisation of the overall happiness of users*, where service providers should try to reach states $s \in Z^{|\mathcal{U}|}$ of maximal L^1-norm. The other criteria consists of the *maximisation of active users*, where service providers try to keep as many active users as possible. Formally, a state $s \in Z^{|\mathcal{U}|}$ satisfying this second goal is associated to a vector $s' \in Z^{|\mathcal{U}|}$ such that $s'_x = s_x$ if $s_x \ge c(u_x)$, $s'_x = 0$ otherwise, and $||s'||_0$ is maximal.

Batch Requests. We consider initially how scheduling strategies affect the user happiness states when we take into account a single batch of job requests. We assume here that each user submits a single request, and therefore we do not investigate variations of $w(u)$. The optimisation criteria in this section will be the L^1-norm of the user happiness state vector. Let us consider the family of scenarios where each task in $\mathcal{T}$ consumes time Δ, and let $t_x, t_y \in \mathcal{T}$ be such that $x + m < y$ and $a(t_x) + w(u_x) > a(t_y) + w(u_y)$.

If **FIFO** is employed, the scheduling plan P will have each request t serviced according to arriving time $a(t)$. In particular, t_x will be processed before t_y according to P and in different moments in time (i.e., they will not be serviced in parallel).

For the same sequence $\mathcal{T}$, because $a(t_x) + w(u_x) > a(t_y) + w(u_y)$, **EAS** would invert the order in which tasks t_x and t_y are processed, so let us consider the plan P' that is almost equal to P, having only the positions of t_x and t_y exchanged. Because all the tasks consume the same amount of time, it is clear that we can transform plan P into plan P^* that would be generated by **EAS** if we apply the same exchange technique sequentially until every pair of requests is positioned accordingly.

Let s and s' be the user happiness state vectors of p after the execution of plans P and P', respectively, and let f_x and f_y be the times at which t_x and t_y have their processing tasks finished according to plan P, respectively (*i.e.*, $f_x < f_y$). Let us refer to $e(t_x)$ and $e(t_y)$ as $e^1(t_x)$ and $e^1(t_y)$ for **FIFO**, respectively, and as $e^2(t_x)$ and $e^2(t_y)$ for **EAS**, respectively.

Finally, let $q_{x,y} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be the function parameterized by $e(t_x)$ and $e(t_y)$ denoting the sum of the changes in the happiness levels of users u_x and u_y after tasks t_x and t_y have been serviced, respectively. It is clear that $q_{x,y}$ depends on the behaviour of i.

Proposition 1. *If $q_{x,y}$ is always the same $\forall x, y \in \mathcal{U}$, is monotonic, and respects exactly one of the following scenarios, then it is possible to decide if either* **EAS** *or* **FIFO** *yields a plan resulting in a user happiness state s with maximal $||s||_1$:*

– $f_{x,y}(a, b) \geq f_{x,y}(c, d)$ *whenever* $|a| + |b| \geq |c| + |d|$*; or*
– $f_{x,y}(a, b) \leq f_{x,y}(c, d)$ *whenever* $|a| + |b| \geq |c| + |d|$*; or*
– $f_{x,y}(a, b) = f_{x,y}(c, d)$ *whenever* $|a| + |b| \geq |c| + |d|$*.*

Proof. Simple inspection shows that $a(t_x)+\Delta+w(u_x)$, $a(t_y)+\Delta+w(u_y)$, f_x, and f_y can appear in six different relative ordering schemes (e.g., $a_y + \Delta + w(u_y) < a_x+\Delta+w(u_x) < f_x < f_y$ is one of them)[1]. Moreover, one can also see that $e^1(t_x)+e^1(t_y) = e^2(t_x) + e^2(t_y)$ and that $max(e^1(t_x), e^1(t_y)) > max(e^2(t_x), e^2(t_y))$ in each of these cases. Therefore, we have $|e^1(t_x)| + |e^1(t_y)| \geq |e^2(t_x)| + |e^2(t_y)|$.

Based on these observations and on our hypothesis, we have the following situations:

– if $f_{x,y}(a, b) \geq f_{x,y}(c, d)$ whenever $|a| + |b| \geq |c| + |d|$, then $c(s) \geq c(s')$;
– if $f_{x,y}(a, b) \leq f_{x,y}(c, d)$ whenever $|a| + |b| \geq |c| + |d|$, then $c(s) \leq c(s')$; and
– if $f_{x,y}(a, b) = f_{x,y}(c, d)$ whenever $|a| + |b| \geq |c| + |d|$, then $c(s) = c(s')$.

Therefore, P' is better than, equal to, or worse than P if $f_{x,y}$ has the first, the second, or the third property, respectively.

Finally, if we assume that $f_{x,y}$ is always the same $\forall x, y$ in $\mathcal{U}$, the resulting user happiness state associated to P^* is better than, equal to, or worse than P if $f_{x,y}$ has the first, the second, or the third property, respectively. □

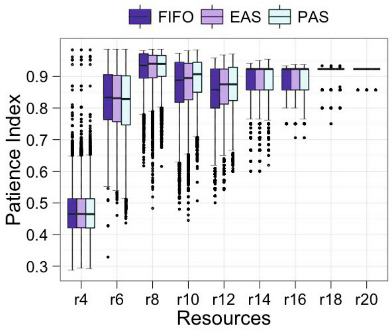
Other propositions comparing the proposed and **FIFO** strategies are presented in the extended version of the paper [7].

3 Evaluation

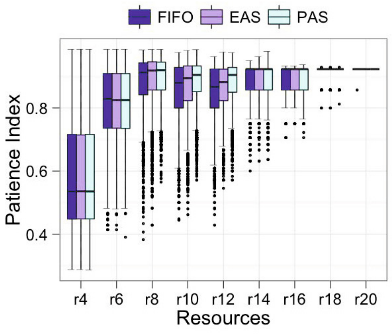
A discrete event simulator was used to evaluate the performance of the scheduling strategies. To model the load of a Cloud service, we crafted three types of workloads with variable numbers of users over a 24-hour period: normal day, flat day, and peaky day. More detail on the workloads is given in the extended version of this work [7].

For each workload we vary the number of resources used by the Cloud service, thus allowing for evaluating the system under different stress levels. When using the system, a user makes a request and waits for its results before making a new

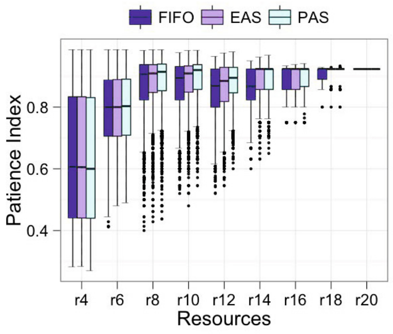
[1] Recall that $a(t_x) + \Delta + w(u_x)$ is already defined as greater than $a(t_y) + \Delta + w(u_y)$.



(a) Flat.



(b) Normal.

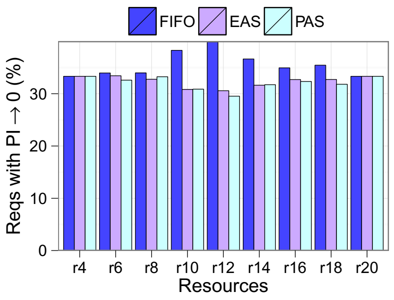


(c) Peaky.

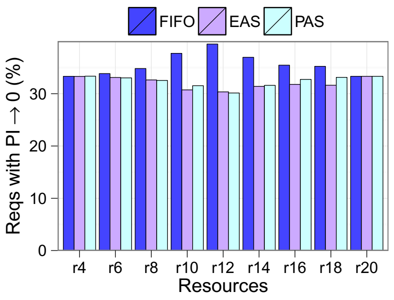
Fig. 1. Patience index under different workloads

request, with a think time between receiving results and making another request uniformly distributed between 0 and 100 seconds. To facilitate the analysis and comparison among the techniques, the length of jobs is constant (10 seconds).

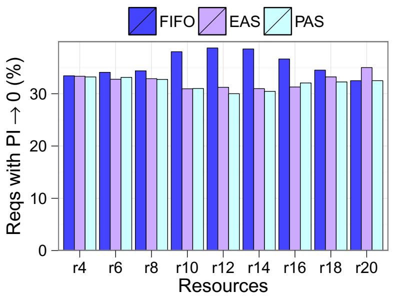
Previous interactions with the service are used to build a user's expectation on how the service should respond, and how quickly a request should be processed. The model that defines a user's expectation on the response time of a request uses two moving averages, (i) an Exponential Weighted Moving Average (EWMA) of the previous 20 response times, with $\alpha = 0.8$; and (ii) an average of the past 4 response times, used to eliminate outliers. When a request completes, if the response time is 30% below the average of the past 4 response times, then the EWMA is not updated, though the value is considered in future iterations. In essence, this model states that the user expects the service to behave similar to previous interactions, with a higher weight to more recent requests. Even though changes in response time affect the user's perception of the service, she disregards large deviations in service quality; unless they become common. As we believe that in real conditions, users would not correctly average their past response times (*i.e.* they may not recall past experiences well) we add a tolerance of 20% to the estimate of response time provided by the model.



(a) Flat.



(b) Normal.



(c) Peaky.

Fig. 2. Percentage of requests whose patience index tends to 0

Users patience thresholds—*i.e.* the maximum response time that she considers acceptable—is randomly selected between 40 seconds and 60 seconds. The provider tracks how it served previous requests made by a user and users the same model described above to compute an estimate of what it believes the user's expectation to be. 60 seconds is also what the provider considers to be the maximum acceptable response time that satisfies the service users. However, for **EAS** and **PAS**, if a request's response is above 60 seconds, the EWMA is updated with 40 seconds, which may give the user priority the next time she submits a request. It is a way the scheduler finds to penalise itself for yielding a response time too far from what it believes the user's expectation to be.

Figure 1 depicts the Patience Indexes (as defined in Section 2) of requests when below 1.0 for flat, normal, and peaky workloads. The lower the values the more unhappy the users. We observe that for high and low system load (*i.e.* r4–6 and r16–20), all strategies perform similarly, whereas for the other loads **PAS** and **EAS** produce higher Patience Indexes than **FIFO**. Under high loads, most requests are completed after the expected response time, thus not allowing the scheduler to exchange the order of the requests in the waiting queue in subsequent task submissions. On the other hand, a very light system contains a short (or empty) waiting queue; hence not having requests to be sorted.

The impact of the scheduling strategies becomes evident when the system is almost fully loaded, *i.e.* when the waiting queue is not empty and there are requests that can quickly be assigned to resources. In this scenario, requests with longer response time expectations can give room to tasks from impatient users. The **FIFO** strategy does not explore the possibility of modifying the order of requests considering user patience.

Figure 2 presents the percentage of requests that were served considerably later than the expected response time, that is, when their Patience Index tends to zero. Such requests represent the stage where users' level of happiness is decreasing considerably. The percentage was normalised by the total number of requests for each resource setting for all strategies. The behaviour of this metric follows the patience indexes, but it highlights the impact of the proposed strategies have on users with very low patience levels.

4 Related Work

Commonly used algorithms in resource management include First-In First-Out, priority-based, deadline-driven, some hybrids using backfilling techniques [18], among others [5,10]. Besides priority and deadline, other factors have been considered, such as fairness [9], energy-consumption [16], and context-awareness [2]. Moreover, utility functions were used to model how the importance of results to users varies over time [4,14] and attention scarcity was leveraged to determine priority of service requests in the Cloud [15].

User behaviour has been explored for optimising resource management in the context of Web caching and page pre-fetching [1,3,8,11]. The goal is to understand how users access web pages, investigate their tolerance level on delays, and pre-fetch or modify page content to enhance user experience. Techniques in this area focus mostly on web content and minimising response time of user requests.

Service research has also investigated the impact of delays on users' behaviour. For instance, Taylor [17] described the concept of delays and surveyed passengers affected by delayed flights to test their hypotheses. Brown et al. [6] and Gans et al. [12] investigated the impact of service delays in call centres. In behavioural economics, Kahneman and Tversky [13] introduced prospect theory to model how people make choices in situations that involve risk or uncertainty.

5 Conclusions

We presented **PAS** and **EAS** that use estimates on users' level of tolerance or patience to define the order in which resources are assigned to requests. Our analysis identified that it is not trivial to choose between **EAS** and **FIFO** as the quality of their schedules depends strongly on users' happiness with a service and tolerance to delays. Our computational evaluation shows that both **PAS** and **EAS** perform better than **FIFO** under peak load scenarios, and that **PAS** is slightly better than **EAS**.

References

1. Alt, F., Sahami Shirazi, A., Schmidt, A., Atterer, R.: Bridging waiting times on web pages. In: 14th Int. Conf. on Human-computer Interaction with Mobile Devices and Services (MobileHCI 2012), pp. 305–308. ACM, New York (2012)
2. Assunção, M.D., et al.: Context-aware job scheduling for cloud computing environments. In: 5th IEEE Int. Conf. on Utility and Cloud Computing, UCC (2012)
3. Atterer, R., Wnuk, M., Schmidt, A.: Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In: 15th Int. Conf. on World Wide Web (WWW 2006), pp. 203–212. ACM, New York (2006)
4. AuYoung, A., et al.: Service contracts and aggregate utility functions. In: 15th IEEE Int. Symp. on High Performance Distributed Computing, HPDC 2006 (2006)
5. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing 61(6), 810–837 (2001)
6. Brown, L., Gans, N., Mandelbaum, A., Sakov, A., Shen, H., Zeltyn, S., Zhao, L.: Statistical analysis of a telephone call center: A queueing-science perspective. Journal of the American Statistical Association 100, 36–50 (2005)
7. Cardonha, C., et al.: Patience-aware scheduling for cloud services: Freeing users from the chains of boredom. arXiv preprint cs/1308.4166 (2013)
8. Cunha, C.R., Jaccoud, C.F.B.: Determining www user's next access and its application to pre-fetching. In: 2nd IEEE Symp. on Computers and Communications (ISCC 1997), Washington, DC, USA, p. 6 (1997)
9. Doulamis, N.D., Doulamis, A.D., Varvarigos, E.A., Varvarigou, T.A.: Fair scheduling algorithms in grids. IEEE Transactions on Parallel and Distributed Systems 18(11), 1630–1648 (2007)
10. Feitelson, D.G., Rudolph, L., Schwiegelshohn, U., Sevcik, K.C., Wong, P.: Theory and practice in parallel job scheduling. In: Feitelson, D.G., Rudolph, L. (eds.) IPPS-WS 1997 and JSSPP 1997. LNCS, vol. 1291, pp. 1–34. Springer, Heidelberg (1997)
11. Galletta, D.F., Henry, R.M., McCoy, S., Polak, P.: Web site delays: How tolerant are users? Journal of the Association for Information Systems 5(1), 1–28 (2004)
12. Gans, N., Koole, G., Mandelbaum, A.: Telephone call centers: Tutorial, review, and research prospects. Manufacturing & Service Operations Management 5(2), 79–141 (2003)
13. Kahneman, D., Tversky, A.: Prospect theory: An analysis of decision under risk. Econometrica: Journal of the Econometric Society, 263–291 (1979)
14. Precise and Realistic Utility Functions for User-Centric Performance Analysis of Schedulers (2007)
15. Netto, M.A.S., Assunção, M.D., Bianchi, S.: Leveraging attention scarcity to improve the overall user experience of cloud services. In: Proceedings of the IFIP 9th International Conference on Network and Service Management, CNSM 2013 (2013)
16. Pineau, J.F., Robert, Y., Vivien, F.: Energy-aware scheduling of bag-of-tasks applications on master–worker platforms. Concurrency and Computation: Practice and Experience 23(2), 145–157 (2011)
17. Taylor, S.: Waiting for service: the relationship between delays and evaluations of service. The Journal of Marketing, 56–69 (1994)
18. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Transactions on Parallel and Distributed Systems 18(6), 789–803 (2007)