# Leveraging Attention Scarcity to Improve the Overall User Experience of Cloud Services

Marco A. S. Netto, Marcos D. Assunção, Silvia Bianchi
IBM Research
Sao Paulo, Brazil

***Abstract*—Applications for mobile devices are increasingly relying on Cloud services to provide content and offload data processing tasks. Traditionally, web-based systems have been optimised to improve response time. Touch sensitive screens and the various sensors of mobile devices allow for better instrumentation, enabling providers to obtain more honest signals on how users utilise a service and learn about their behaviours. This work introduces an architecture that explores honest signals to determine a few user behaviours, *e.g.* the tendency to perform multiple tasks at a time, change focus, and expect fast response from a service. The architecture relies on a novel resource management strategy that considers such behaviours to prioritise service requests from users who demand fast response from a service. By comparing the proposed strategy with one that does not consider the signals, we show that the experience of users who demand faster response time can be improved without degrading the quality of service of those who often perform multiple activities. The proposed strategy also brings benefits to service providers as no additional resources are necessary to enhance overall user experience, which we modelled using Prospect Theory.**

## I. Introduction

Over the past few years, data centres and Cloud infrastructures have become essential to enhance the capabilities of mobile applications such as web search, speech recognition, games, search for points of interest, and access to social networks. Users of devices such as laptops, tablets, and smartphones interact with applications that increasingly rely on Cloud services used to provide content and offload data processing tasks, delivering an overall better user experience.

Web-based applications are often optimised to achieve a minimum percentile of requests that meet a given response time. In these scenarios, users are generally treated equally and information like intervals between requests is used as signals to infer user think time. Peak demands are likely to degrade the quality of service perceived by all users. The utilisation of tablets and other devices with touch sensitive screens and richer interfaces allow for obtaining more honest signals on user behaviour, enabling providers to better understand when results delivered by their Cloud services are consumed.

Figure 1 illustrates an example in which signals indicate when service results are consumed, where a user plays a game that runs locally on the device and accesses Cloud services such as calendar and train timetables. The user plays the game while waiting for results of a service request to arrive, and may not access the results immediately upon their transfer to the mobile device. The Cloud provider, if knowing that the user changed the application on focus, can prioritise other requests and process the request at hand in low priority. Considering this scenario, one of the questions in which we are interested is: *"Can these signals on service consumption be used to differentiate the processing of requests, optimise the use of Cloud resources, and deliver better user experience?"*

This work attempts to shed some light on answering this question, by exploring these signals to influence the manner user requests are processed by Cloud services. With the goal of improving overall user experience, the paper proposes a resource management strategy that takes into account how users consume the results of requests made to Cloud services. The strategy assumes that even though users deserve similar experience, the manner they consume results of service requests indicates that they have different quality of service requirements, which in turn dictate how resources are allocated to their requests in the Cloud. Although the proposed resource management strategy can be used for private setting services, we explore it for public Cloud services due to the large amount of users and their heterogeneous behaviours. Apart from increasing the overall user experience, understanding user behaviour can reduce resource needs due to a more efficient infrastructure management strategy.

The contributions of this paper are therefore:

- A resource management strategy that improves users' experience by using information on how users consume results of requests made to Cloud services;
- An architecture design to realise the proposed strategy in a real scenario; and
- Simulation results that demonstrate under what conditions the proposed strategy brings benefits.

## II. Background and Problem Description

Several solutions were provided to tackle challenges of resource management in various types of distributed systems, including high performance computing, clusters of computers [1], Grid computing [2], and more recently Clouds. When using Clouds as a means to offload tasks of mobile applications,

ISBN 978-3-901882-53-1, 9th CNSM and Workshops ©2013 IFIP 

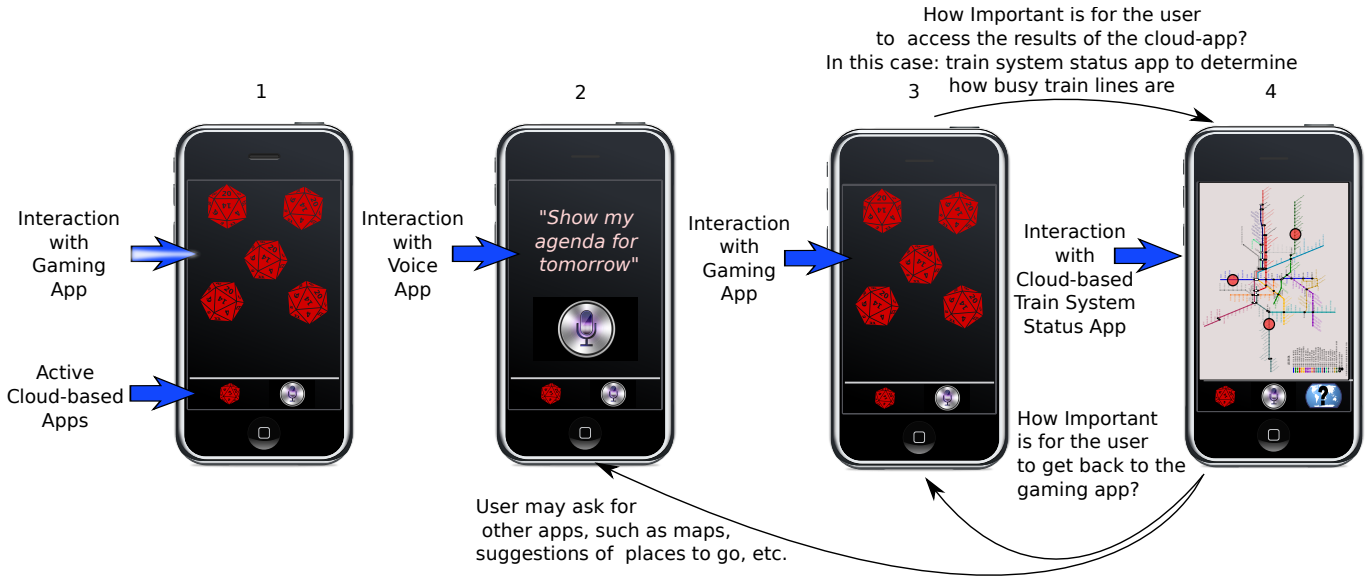



Fig. 1: Example of more honest signal on consumption of service results.

users are often unaware of how many resources their requests need [3], and the user's perception on quality of service depends on whether the results are ready when he intends to consume them. It is of the provider's interest to guarantee that the results be provided when the user expects them. In web applications, providing such guarantees was commonly interpreted as synonymous to reducing request response time.

In a society where human attention is increasingly becoming a scarce commodity, and where users perform multiple concurrent activities [4] on their mobile devices [5], response time might not be the sole metric to optimise the use of Cloud resources and improve user experience. With more honest signals on how users are interacting with their devices and considering how they consume results may help providers optimise the use of their resources.

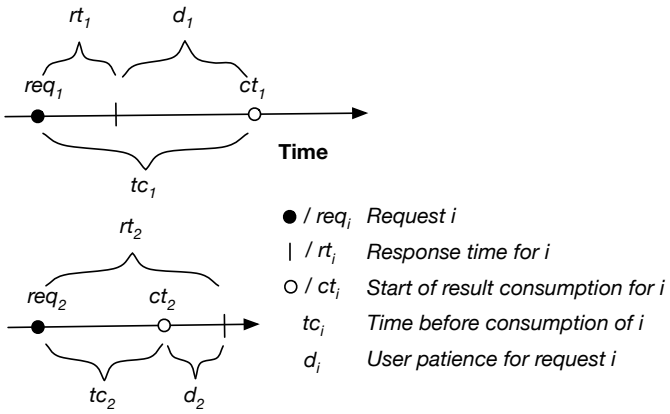



Fig. 2: Request times considered in this work.

Figure 2 depicts two service requests, $req_1$ and $req_2$ made by two distinct users over time. Each request, represented by a black circle, has its respective response time (*i.e.* $rt_1$ and $rt_2$). This work considers that by exploring honest signals enabled by touch sensitive interfaces, such as the change of application focus, scrolling and other touches on certain interface elements, it is possible to reveal details on when the user starts consuming the results of a service request. The figure shows that the time before consumption for $req_1$, *i.e.* $tc_1$, is greater than its response time. The second request, $req_2$ shows a different scenario where the user expects to consume the results $tc_2$ before all results are delivered to the device. The information on $tc_2$ can be determined, for instance, by the user starting to scroll down a page before it is completely loaded.

For a request $req_i$, we term as the user's *patience* the distance between response times and time to start consuming results $d_i$, *e.g.* $d_1 = tc_1 - rt_1$. As $d_i$ approaches 0, a user becomes less patient, eventually becoming *angry* if $d_i$ is negative. To characterise the user experience $ue$, we use an approach based on Prospect Theory, where the sense of losing an opportunity has higher impact in a user's satisfaction [6], as illustrated in Figure 3. As described in the evaluation section, $d < 0$ translates into worse $ue$ than positive $d$.

Considering a model where users' expectation on when to consume results is not heavily influenced by the average request response times, we want to: (i) minimise the distance between response times and time to start consuming results, *i.e.* $d_1$ and $d_2$ in Figure 2; and (ii) investigate the impact of minimising this distance on overall user experience.

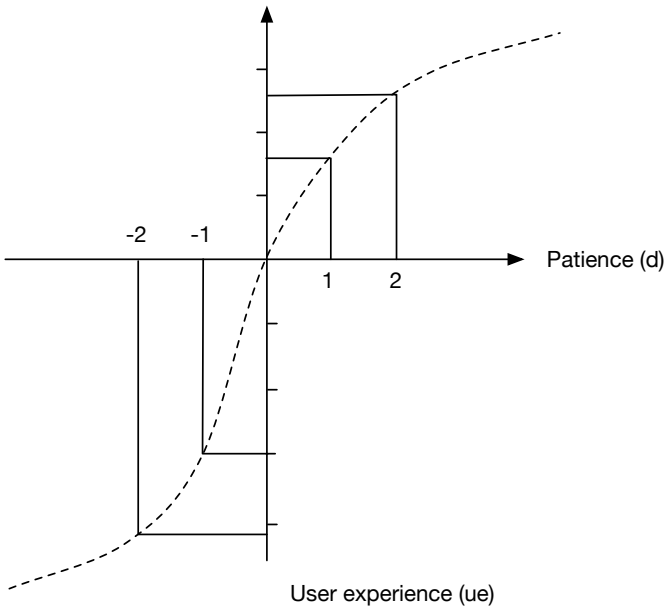



Fig. 3: Using Prospect Theory [6] to model user experience as a function of patience.

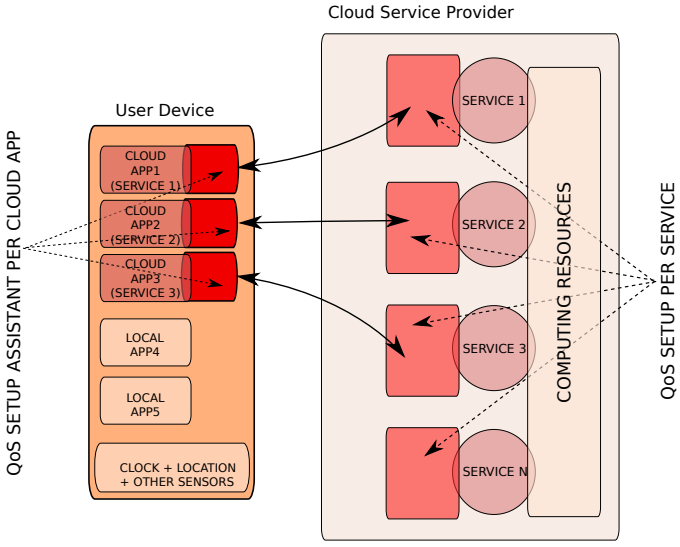



Fig. 4: Components to realise the adaptive QoS setup using users' service consumption behaviours.

## III. Adaptive QoS Architecture

This section describes the proposed architecture for adapting the quality-of-service of Cloud services according to user consumption behaviour. As shown in Figure 4, the architecture comprises:

- **User Device:** smartphones, tablets, or a device that runs applications that access services from a Cloud.
- **QoS Setup Assistant:** runs on the user device and assists the Cloud provider to identify whether an application accessing a remote service needs high QoS. This component uses information on Cloud services, local applications, clock time, geographical location, and other sensors to assist determining service consumption behaviour.
- **Cloud Service Provider:** offers services (*e.g.* web search, gaming, and data provisioning) that require computing resources provided by the Cloud service provider itself or hosted by a third party.
- **QoS Setup Service:** used by the Cloud Provider to set up QoS parameters for services.

*User Device* and *Cloud Service Provider* are components that underly typical scenarios of users consuming Cloud services and resources. The components introduced here are *QoS Setup Assistant*, which runs on the user device, and the *QoS Setup Service*, executed on the Cloud provider. Figure 4 depicts one *QoS Setup Service* per Cloud service, which we believe is a more appropriate scenario where resources for Cloud services are allocated using resources from IaaS providers. This is the scenario considered here, but the architecture can adapt to PaaS providers offering and managing multiple services, where a *QoS Setup Service* would define the QoS parameters for multiple services.

### A. QoS Setup Assistant

*QoS Setup Assistant* is an optional component that determines a user's service consumption behaviour. A Cloud provider offering a search engine can identify when a user clicks on a hyperlink on a page if the mobile application using the service was developed in house. However, if the provider is not responsible for the development of the application or the OS, it is unlikely to identify whether a user is scrolling down the results page, identify whether the results page in on focus, or if the user changes the focus to a music player.

This component explores APIs exposed by the device's OS to deliver important contextual information to the *QoS Setup Service*, such as list of active applications, windows that are on focus, window sizes, GPS data, and click speed. This information can be obtained from mobile operating systems by instrumenting what views are displayed, when an application goes to background or returns to foreground execution, among other details. For ordinary operating systems, such as Microsoft Windows, GNU/Linux or Mac OS X this information is commonly exposed by their window managers. When information cannot be obtained from the device OS, the *client application* needs to incorporate the *QoS Setup Service* functionalities to collect information specific for the service it uses (see Figure 4). For instance, the client application

can inform the provider about the user's menu navigation behaviour.

In order to enhance privacy, the *QoS Setup Assistant* may calculate locally when resources are available, the expected time before consumption of results $tc$ and send only this information to the *QoS Setup Service* associated with the request. Note that, if a user is performing a single task, the $tc$ will be equal to the response time as the user requires prompt results. Moreover, the assistant can notify the user, via a configuration menu, the type of information from his/her device that can be published to the Cloud provider.

Another aspect to be considered is the frequency at which the collected information is reported to the Cloud Provider. The information can be sent at regular time intervals, and the provider can use historical information for inferring user behaviour. Another approach consists in dynamically determine when information about a user is required based on how other users interact with the service. In addition to determining the frequency of data reporting, it is important to evaluate the cost of collecting data from user devices, especially when considering mobile devices with limited battery life time. The cost of performing such data collecting is ongoing work, and this paper focuses primarily on the benefits of collecting such user behaviour data.

### B. QoS Setup Service

*QoS Setup Service* determines how users consume service results and how important certain services are to them. It considers two phases for each service: an initial setup where the user's context helps determining the required QoS; and how users interact with the service after the initial setup. This work focuses on the latter by introducing a resource management strategy termed as **Patience-aware Prioritisation (PaP)** that adjusts the QoS of users as follows:

1) *QoS Setup Service* obtains the expected time before consumption of results $tc$—*e.g.* reported by *QoS Setup Assistant*—and expected response time $rt$ and determines the user patience $d$.

2) Give priority to users who have low values of $d$; *i.e.* those who are "less patient".

To avoid starvation a user's patience decreases as his request waits to be served.

## IV. Evaluation

The proposed resource management strategy uses signals on service consumption to prioritise requests and improve overall user experience. Requests from users who require prompt results are processed before those from users who have more relaxed requirements—*i.e.* are more patient—without heavy compromise on the latter. The experimental results presented in this section demonstrate that the principle is sound. The proposed strategy, PaP, is compared against a baseline strategy, **Standard**, which processes requests as they arrive.

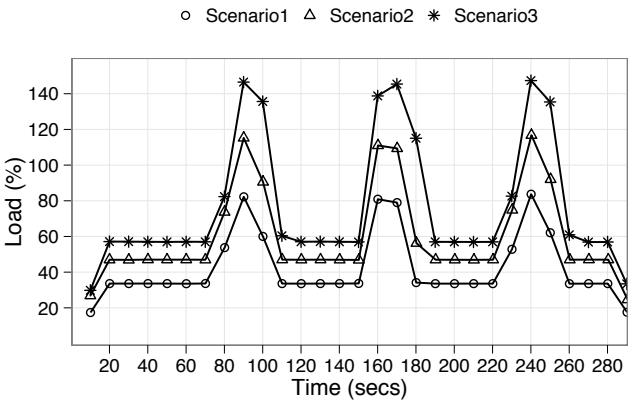



Fig. 5: Workloads with resource peak demands.

### A. Experiment Setup

The evaluation uses a built-in-house discrete-event simulator that contains a computing resource used by a service to serve multiple users' requests. To simplify the analysis, the length of requests is constant; they always demand the same processor time. We classified the users in two categories: *single-task users* who require prompt results from the Cloud provider, and *multi-task users* who engage in other concurrent activities and therefore do not require prompt results. The $tc$ of single-task and multi-task users is 2 and 10 seconds respectively. The modelled service can process 100 requests at a time. We also considered three scenarios with different load rates, each with three demand peaks as shown in Figure 5. Load over 100% indicates that certain requests wait for resources to be processed. The number of users varies depending on the load, ranging from 500 to 700 users; where each user makes from 10 to 30 server requests, thus generating up to 2100 requests in the third scenario.

We first evaluate the user patience $d$ as described in Section II, which expresses how long users start to consume results once they are available or how long they wait for results to become available. To improve the experience of single-task users who require prompt results, we explore the patience of multi-task users. In the experiments, the service time is 2 seconds, hence forcing the patience $d$ of single-task users to be zero.

### B. Result Analysis

Figure 6 shows the patience of multi-task users under each scenario. We observe that the lower the number of users the higher their patience for both strategies. This happens

because the provider can process requests faster and results stay longer in users' devices before they are consumed. PaP strategy reduces the users' patience in comparison to Standard by prioritising requests based on estimations of when users consume results. As the system load increases, the provider requires, in average, more time to answer to user requests. In this case, thus the PaP strategy makes more use of the patience of multi-task users.

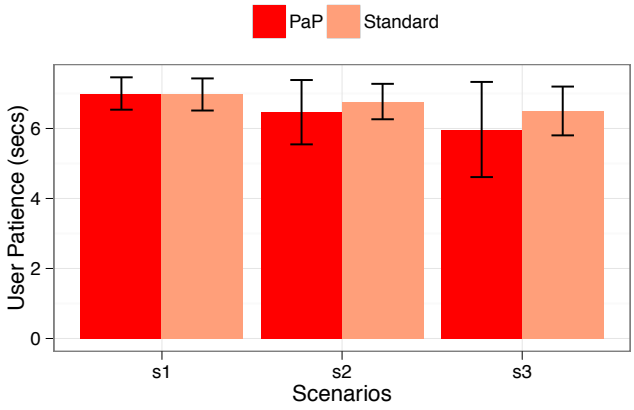



Fig. 6: Overall patience of multi-task users.

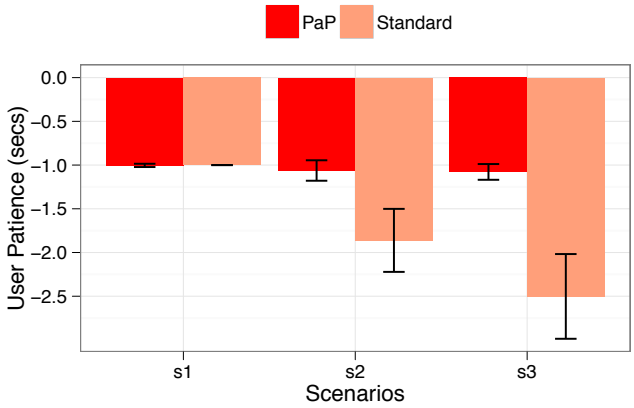


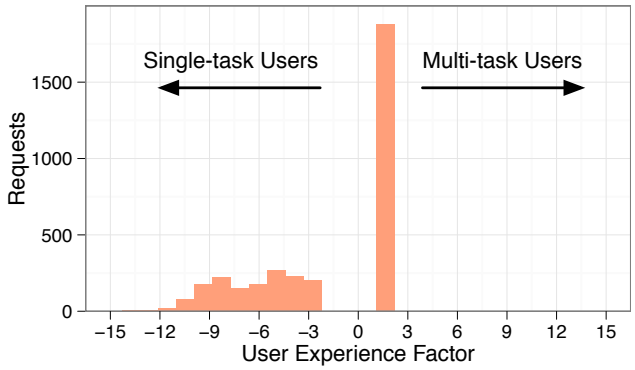

Fig. 7: Overall patience of single-task users.

Figure 7 depicts the overall patience of single-task users. As aforementioned, the fact that the user patience is negative means that the user expected to consume the response before it arrives. By reducing the patience of multi-task users, single-tasks users enjoy better user experience because their requests are given priority. Note that the benefit of the PaP strategy increases as the number of users grows, mainly because more users compete to obtain CPU time to process requests. Prioritisation of single-task users' requests makes them wait less to obtain results. The increase of the standard deviation with the system load is a result of the request bursts in Scenarios 2 and 3.

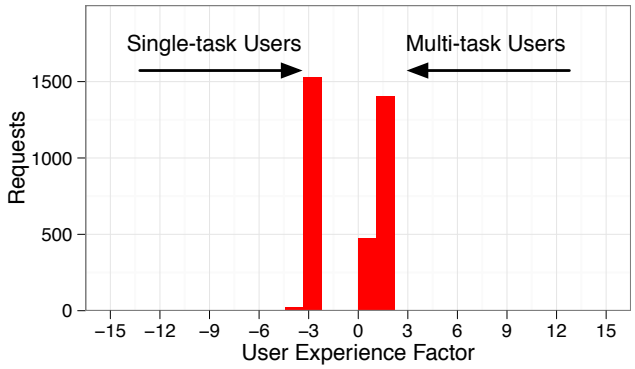
Figure 8 shows, for Scenario 3, the impact of the strategies on user experience ($ue$). We use prospect theory [6] to model user experience using Equation 1. The single-task users have user experience factor below 0 and multi-task users have user experience factor above 0. The figure illustrates that PaP improves experience of single-task users until the experience of multi-task users becomes affected negatively. Such a point is where both types of users start to compete for resources to have positive user experience.

$$ue = \begin{cases} log(d), & if \quad d > 0 \\ -exp(-d), & if \quad d < 0 \\ 0, & otherwise \end{cases} \tag{1}$$

Figure 8 (a) depicts that several requests have user experience factor between -12 and -3, whereas a great number of requests are around 2. With PaP, in Figure 8 (b), a considerable number of requests have their user experience reduced, having several of them migrated from 2 to 1 and 0. This helps bringing the requests with very low user experience close to -3. Similar

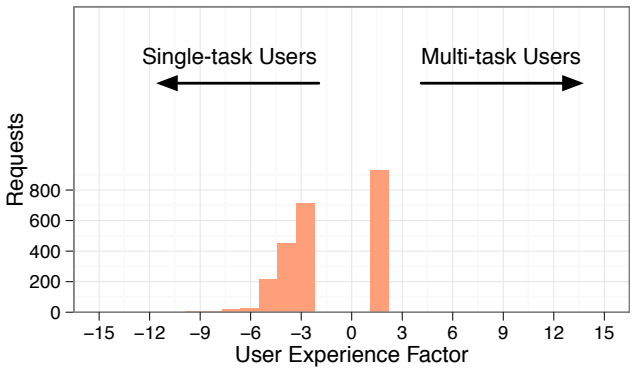



(a) Standard strategy.

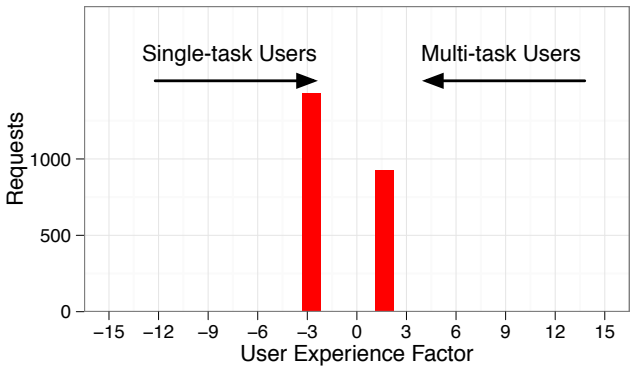



(b) PaP strategy.

Fig. 8: User experience for Scenario 3.




(a) Standard strategy.




(b) PaP strategy.

Fig. 9: User experience for Scenario 2.

behaviour happens in Scenario 2 (Figure 9), however with less impact due to lower load and peak demands.

Multi-task users are not affected because the priority of their requests increases as they wait for resources, and after some time they compete with single-task users, thus having the same priority. We believe PaP is suitable to peak demand scenarios where adjusting the Cloud infrastructure to handle the surge takes time; additional VMs are required and booting them up takes a while; a period during which PaP can be enforced. Another aspect to be highlighted is that the improvements of PaP over the Standard strategy does not require additional resources.

## V. Related Work

Existing work leveraged mobile users' contextual information to improve Cloud service delivery [7], [8]. La and Kim [7] proposed a framework where Cloud services are provided or adapted to a user's context. Sheng *et al.* [8] proposed a platform for personalised service provisioning, which aims to provide more personalised services that improve overall user experience. Software engineering methods also considered support for context-aware applications [9]. Other approaches directed service requests to specific service providers that satisfy certain quality of service constraints [10], [11]. By identifying similarities amongst QoS of active and training users, CloudRank [12] ranks the personalised QoS for a set of Cloud services.

Whereas most of these context-aware [13] solutions focus on personalising Cloud services, they pay little attention to Cloud resource management. For the latter, previous work provides techniques to adjust resource allocation and guarantee QoS [14], [15] by, for instance: adjusting resource allocation under performance interference [15]; collecting information from the mobile device and reporting it to the provider to adjust the service [16]; using information on the power consumed by a mobile device to decide on the QoS requirements and whether a service should run locally or on the Cloud [17].

Previous work also tackled several challenges inherent to job scheduling [18], [19]. For instance, Boloor *et al.* proposed a request allocation algorithm for context-aware applications hosted in a distributed Cloud [20]. Buyya *et al.* [21] discuss challenges of autonomic resource provisioning based on users QoS requirements. Other techniques were introduced to improve resource management in Clouds and guarantee QoS, such as provide means for performance isolation across multiple VMs [22]. Previous work also highlighted some of the sins of Cloud computing research [23], and the heterogeneity of Cloud workloads [24].

Other approaches focus on methods to enhance the performance of local processing, bandwidth utilisation, memory consumption, power and connectivity by inferring local context [25], [26], [27], [28], [29]. Our previous work proposes scheduling Cloud jobs according to variations of the user context, such as time, location and social settings [30]. We aim to build on such efforts to improve the provision of Cloud services and the overall user experience when using the services.

## VI. Conclusions

This paper motivated the importance of understanding how users consume Cloud services and how this knowledge can be leveraged to enhance user experience and optimise the use of Cloud resources. Sensors to understand consumption behaviour are becoming more pervasive at both hardware and software levels. We proposed an architecture and resource management strategy to prioritise user requests based on information on when users need the results from the Cloud.

We find that it is possible to improve the experience of users that require prompt results from the Cloud without heavily affecting users who are involved in multiple activities at a time, thus not demanding quick answers from the Cloud. The strategy tries to leverage as much as possible the patience of multi-task users to enhance the user experience of single-task users. To achieve this goal multi-task users have their priority increased over time, so when they wait long enough for resources, they start to complete with single-task users.

This work illustrates how traditional resource management can leverage more honest signals provided by smartphones, tablets, and software systems of such devices. A strategy that employs user consumption behaviour to provision Cloud services was presented. Initial evaluation shows that it suits scenarios with peak demand where the Cloud infrastructure has not yet been adjusted accordingly; additional VMs may be required at additional cost and provisioning them takes time. Ongoing work is on prototyping the QoS Setup Assistant to obtain signals to determine service consumption behaviour.

## REFERENCES


[1] R. Buyya, Ed., *High Performance Cluster Computing: Programming and Applications*, ser. 0-13-013785-5. NJ, USA: Prentice Hall PTR, 1999, vol. 2.

[2] I. Foster and C. Kesselman, *The Grid2: Blueprint for a New Computing Infrastructure*, 2nd ed., ser. ISBN: 1558609334, I. Foster and C. Kesselman, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[3] C. B. Lee and A. Snavely, "On the user - scheduler dialogue: Studies of user-provided runtime estimates and utility functions," *International Journal of High Performance Computing Applications*, vol. 20, no. 4, pp. 495–506, 2006.

[4] R. Benbunan-Fich, R. F. Adler, and T. Mavlanova, "Measuring multitasking behavior with activity-based metrics," *ACM Transactions on Computer-Human Interaction*, vol. 18, no. 2, pp. 7:1–7:22, Jul. 2011.

[5] L. M. Carrier, N. A. Cheever, L. D. Rosen, S. Benitez, and J. Chang, "Multitasking across generations: Multitasking choices and difficulty ratings in three generations of americans," *Computers in Human Behavior*, vol. 25, no. 2, pp. 483–489, Mar. 2009.

[6] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica: Journal of the Econometric Society*, pp. 263–291, 1979.

[7] H. J. La and S. D. Kim, "A conceptual framework for provisioning context-aware mobile cloud services," in *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD)*, 2010.

[8] Q. Z. Sheng *et al.*, "User-centric services provisioning in wireless environments," *Communications of the ACM*, vol. 51, no. 11, pp. 130–135, 2008.

[9] F. C. Delicato, I. L. A. Santos, P. F. Pires, A. L. S. Oliveira, T. V. Batista, and L. Pirmez, "Using aspects and dynamic composition to provide context-aware adaptation for mobile applications," in *Proceedings of the 2009 ACM symposium on Applied Computing (SAC)*, 2009, pp. 456–460.

[10] H. Song, C. S. Bae, J. W. Lee, and C.-H. Youn, "Utility adaptive service brokering mechanism for personal cloud service," in *Proceedings of the Military Communications Conference (MILCOM)*, 2011, pp. 1622–1627.

[11] P. Papakos, L. Capra, and D. S. Rosenblum, "Volare: context-aware adaptive cloud service discovery for mobile systems," in *Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware (ARM)*, 2010.

[12] Z. Zheng, X. Wu, Y. Zhang, M. R. Lyu, and J. Wang, "QoS ranking prediction for cloud services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, p. 1, 2012.

[13] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 4, pp. 263–277, Jun. 2007.

[14] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "QoS-aware clouds," in *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD)*, 2010.

[15] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for QoS-aware clouds," in *Proceedings of the 5th European conference on Computer systems (EuroSys)*, 2010.

[16] P. Zhang and Z. Yan, "A qos-aware system for mobile cloud computing," in *Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Sep. 2011, pp. 518–522.

[17] Y. Ye *et al.*, "A framework for QoS and power management in a service cloud environment with mobile devices," in *Proceedings of the Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE)*, 2010.

[18] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, "Parallel job scheduling – a status report," in *Job Scheduling Strategies for Parallel Processing (JSSPP)*, 2004.

[19] A. Takefusa, S. Matsuoka, H. Casanova, and F. Berman, "A study of deadline scheduling for client-server systems on the computational grid," in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, 2001.

[20] K. Boloor, R. Chirkova, T. Salo, and Y. Viniotis, "Management of SOA-based context-aware applications hosted in a distributed cloud subject to percentile constraints," in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2011.

[21] R. Buyya, R. N. Calheiros, and X. Li, "Autonomic cloud

computing: Open challenges and architectural elements," *Proceedings of the Third International Conference on Emerging Applications of Information Technology*, pp. 3–10, 2012.

[22] M. Silva, K. D. Ryu, and D. D. Silva, "VM performance isolation to support QoS in cloud," in *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2012, pp. 1144–1151.

[23] M. Schwarzkopf, D. G. Murray, and S. Hand, "The seven deadly sins of cloud computing research," in *USENIX HotCloud*, 2012.

[24] C. Reiss *et al.*, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing (SoCC)*, 2012.

[25] F. Koch and F. Dighum, "Enhanced deliberation behaviour for BDI-agents in mobile services," in *Proceedings of the 8th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS)*, Salamanca, May 2010.

[26] Y. Xiao, P. Hui, P. Savolainen, and A. Ylä-Jääski, "Cascap: Cloud-assisted context-aware power management for mobile devices," in *Proceedings of the 2nd International Workshop on Mobile Cloud Computing and services (MCS)*, 2011.

[27] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[28] B. Y. L. Kimura, H. C. Guardia, and E. dos Santos Moreira, "Disruption-tolerant sessions for seamless mobility," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2012.

[29] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications," *IEEE Transactions on Software Engineer*, vol. 29, no. 10, pp. 929–945, 2003.

[30] M. D. Assunção, M. A. S. Netto, F. Koch, and S. Bianchi, "Context-aware job scheduling for cloud computing environments," in *Proceedings of the IEEE Fifth International Conference on Utility and Cloud Computing (UCC)*, 2012.