Context-aware Job Scheduling for Cloud Computing Environments

Marcos D. Assunção, Marco A. S. Netto,
Fernando Koch, Silvia Bianchi

IBM Research
Sao Paulo, Brazil

***Abstract*—The more instrumented society is demanding smarter services to help coordinate daily activities and exceptional situations. Applications become sophisticated and context-aware as the pervasiveness of technology increases. In order to cope with resource limitations of mobile-based environments, it is a common practice to delegate processing intensive components to a Cloud Computing infrastructure. In this scenario, executions of server-based jobs are still dependent on the local variations of the end-user context. We claim that there is a need for an advanced model for smarter services that combines techniques of context awareness and adaptive job scheduling. This model aims at rationalising the resource utilisation in a Cloud Computing environment, while leading to significant improvement of quality of service. In this paper, we introduce such a model and describe its performance benefits through a combination of social and service simulations. We analyse the results by demonstrating gains in performance, quality of service, reduction of wasted jobs, and improvement of overall end-user experience.**

I. Introduction

Applications are becoming more sophisticated and context-aware in order to meet the ever increasing users' demand. Such application solutions range from the popular Apple Siri that delivers voice based assistance, IBM Shopping Personal Assistant that provides personalised shopping support to end-users, and customised personal assistants to provide support in emergency coordination scenarios. The latter solution is widespread in Central Operation setups, providing the infrastructure required for crowd coordination and monitoring.

To cope with resource limitations of mobile-based applications, it is a common practice to delegate processing intensive components to a Cloud Computing infrastructure [1], [2]. For instance, (a) Apple Siri runs its sophisticated voice recognition feature remotely; (b) emergency coordination assistants could execute the planning and scheduling processing on the Cloud; and (c) material composer assistants that prepare tailored content for meetings helping on offices' daily activities. Basically, in any of these solutions the process works by invoking server-based jobs that execute complex operations. Once a job completes, it returns the results to the mobile device for consumption.

In this scenario, the execution of a server-based job is still dependent on variations of local context. However, the solution is not always able to combine local context information to coordinate the server-based job execution. For example, a plan formed by emergency coordination assistants depends on how local events evolve; events that can become outdated even before their consumption. Similarly, the material generated by material composer assistants becomes irrelevant if the end-user decides not to participate in the meeting.

The question is: *How to rationalise the utilisation of Cloud resources by continuously adapting the execution of jobs in response to variations of the end-user's local context?*

For that, there is an opportunity to construct a model for smarter services that combines techniques of *Context Awareness* and *Adaptive Job Scheduling*. In this work, we introduce our research on advanced methods to adjust the priorities of server-based jobs in response or proactively to variations of the local context. Its application leads to significant improvement of quality of service by reducing the usage of servers' resources.

The key contributions of this work are towards a model to:

- Leverage user-context to optimise job scheduling in the Cloud;
- Provide delay-tolerant job execution required in mobile environments; and
- Reduce resource wastage by scheduling the jobs considering changes in user's context.

We analysed the results by demonstrating the gains in performance, quality of service, reduction of wasted jobs, and improvement of overall end-user experience. We describe the performance benefits of the proposed model through a combination of social and service simulations. We conclude that the impact is especially relevant in scenarios that involve a large number of personal assistants combined with a highly dynamic environment.

II. Background and Related Work

In this section, we provide background information on *context awareness* and *adaptive job scheduling*.

A. On Context Awareness

Context can be defined to be a user's physical, social, emotional or informational state [3]–[5]. It can be described as situations where the individual or machine is immersed. In this scenario, context awareness is the ability to sense and react to situation variations towards better operations. This approach has been used by software to provide better integration with the environment and reduce the need for user's input. As a general sense, there are three issues related to the implementation of context awareness: (i) *what information is required by the application and how to represent it?* (ii) *how the application collects this information?* (iii) *how to compute upon this information, the so-called context processing?*

We apply context awareness as a tool to automate adaptation of the processing of jobs in the Cloud Computing environment based upon variations of the local environment. For that, we exploit the concept of *window of opportunity*, as presented by Koch and Dighum [6], wherein the delivery of certain information is more relevant to the user. Thus, the context processing relates to the ability of sensing variation of local context and computing its *context distance* to a certain condition. This processing is trivial for some aspects of context, such as location and time.

However, the representation of the situation encompasses more aspects of the environment. Context information in mobile setups encompasses information of eight dimensions, as presented by Graham and Kjeldskov [7]: (i) time; (ii) absolute location; (iii) relative location; (iv) objects present; (v) activity; (vi) social setting; (vii) environment, and; (viii) culture. The method to calculate in/out conditions for window of opportunity works based on answering questions like: *Are certain objects present? Is the user engaged in certain activities? Is the user part of or near a certain social setting? Do the environmental conditions enclose the situations a, b, and c?* The proximity can be derived from the conditions inferred based on these questions. For example, an application can infer that the user is in the window of opportunity if the user is physically near a position such as a meeting room (absolute location) and the application can access a file for the meeting (objects present) in a few seconds (activity).

The ability to estimate the application's relative position into the window of opportunity to deliver relevant information is essential to the operation of any well suited, context-aware solution. To that intend, we are leveraging in our model a Context Analyser able to infer context distances based on events received from the environment. The output of this computation is reasoned against the global context information (*i.e.* context information computed from all users and infrastructure in the system) and system's situation to adjust the status of current jobs.

B. On Adaptive Job Scheduling

Previous work leverages context information in mobile environments to improve Cloud service delivery. A framework for context-aware mobile services has been proposed by La and Kim [8]. The framework focuses on service provisioning based on context information. Depending on the user's context, a set of Cloud services can be provided or current services can be adapted dynamically. Similar to this approach, Sheng *et al.* [9] proposed a distributed, adaptive, and context-aware platform for personalised service provisioning. The platform aims to provide a more personalised service improving overall user experience. Other approaches propose to allocate a requested service to a specific service provider satisfying the quality of service constraints [10], [11]. Software engineering has also been considered to support context-aware applications [12].

Most of the context-aware Cloud services solutions focus on providing more personalised services, but very few address the problem of resource wastage.

Several job scheduling techniques have been proposed in order to reduce resource wastage [13], [14]. Boloor *et al.* proposed a request allocation algorithm for context-aware applications hosted in a distributed Cloud [15]. The goal is to meet the service level requirements to reduce the penalty charges in the Cloud. Nevertheless, this approach does not consider the window of opportunity during the job scheduling to deliver useful information to the end user.

Other approaches focus on methods to enhance the performance of local processing, bandwidth utilisation, memory consumption, power and connectivity based on context awareness techniques [6], [16]–[18]. Similarly, Capra *et al.* described an infrastructure to compose local adaptation based on context information [19].

In mobile environments the execution of jobs must adapt to variations of the user context. However, most solutions do not combine local context information to coordinate the job execution. Our work extends the vision of these solutions by adding the capability of central control and adjustment of server-based job processing to the system.

C. Illustrative Scenario

In Cloud Computing, user context information can be exploited to define priorities of jobs executed in the Cloud associated with applications running on the handheld user device.

Let us consider the illustrative scenario depicted in Figure 1. In this scenario, there is a user of a device D_1 changing context C_i in the following four dimensions:

- Time: refers to the time of day;
- Location: working area, demo room, and meeting room;
- Social settings: the set of users working in one or more applications that depend on one another;
- Object's present: applications App_1, App_2, App_3 and App_4.

The user starts in context C_0 where no application is executed on his/her device D_1. Thus, no job scheduling is required. Then the user moves to the working area and starts to use application App_1 (Context C_1), in which the final results must be ready when he/she moves to the meeting room. The associated remote jobs for App_1 are created with low priority.

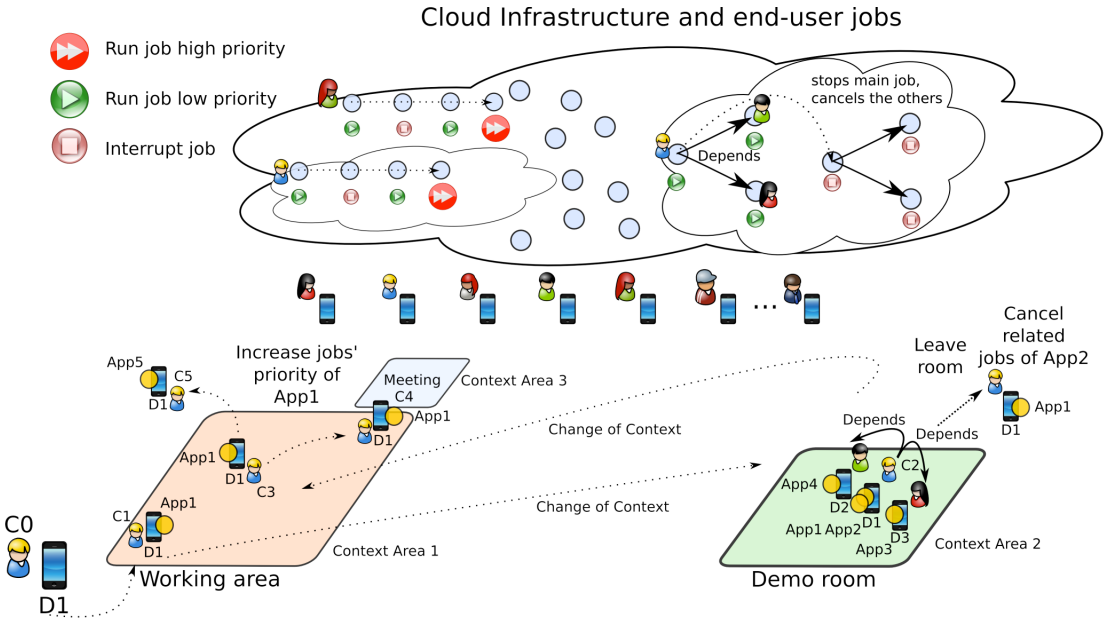


Fig. 1. Motivation scenario: user moves from context C_0 to context C_5 using a device D_1 and activating applications App_1, App_2, and App_5 over three context areas.

After that, the user moves to the demo room and starts to use application App_2 (Context C_2). At the same time, another user with device D_3 starts to use the application App_3 and user of D_2 starts to use the application App_4. The three users are involved in the same demo and the remote jobs associated with the three applications App_2, App_3 and App_4 depend on one another. The associated remote jobs for App_1 are interrupted.

If the user of D_1 leaves the demo room and users of D_2 and D_3 remain in the room, then the associated remote jobs for the dependent applications App_2, App_3 and App_4 are automatically interrupted as the applications are associated to the same demo.

Once the user of D_1 returns to the working area, the application App_1 restarts its execution with low priority (Context C_3) and depending on the application type of App_2, App_3 and App_4 are paused. If the user moves close to the meeting room, getting to Context C_4, jobs associated with App_1 receive high priority as the results are close to be required. Otherwise, if the user moves out the working area to perform a task that requires App_5, the jobs associated with App_1 are interrupted.

In this example, the Cloud provider utilises end-user local context to prioritise the remote jobs according to user needs. Existing solutions would give priority to jobs without considering the dynamic needs of the end-users.

As a realisation of the scenario illustrated above, let us consider a social setting where people of different nationalities try to communicate (*e.g.* Soccer World Cup and other big events). There is a need for "Universal Translators" enacted by jobs running on the Cloud and accessed by handheld devices that provide the interface and sensors for the local context. The translator jobs are started by a combination of both necessity (requirement to translate between language A, B, and C) and proximity (individuals communicating in the same room). When there is a variation of context and, for instance, the individual who speaks language C leaves the room, the system must be able to automatically react to this variation by sending an event to pause the jobs associated with the translations A/C and B/C on the Cloud.

III. Smarter Cloud Services Approach

We propose a system composed of: (i) a Cloud provider that contains a Resource Management System (RMS) able to create and prioritise end-user jobs based on information of the user local context and historical end-user context data; (ii) a set of computing resources to process end-user jobs; and (iii) handheld devices able to obtain the user context and communicate with the Cloud provider. The next two sections introduce the architecture and the algorithm to prioritise the jobs based on user local context.

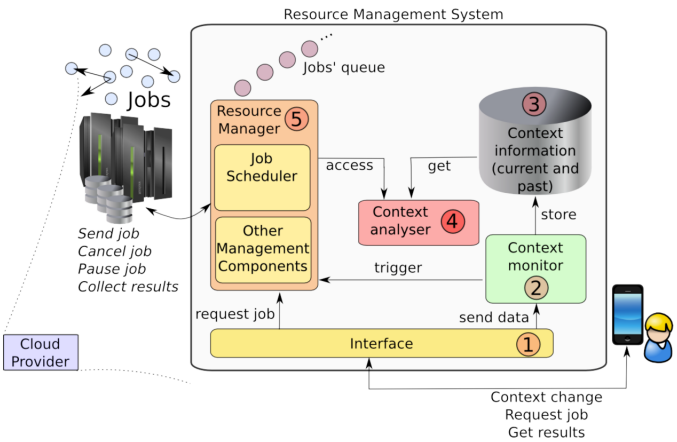


Fig. 2. Architecture of the resource management system.

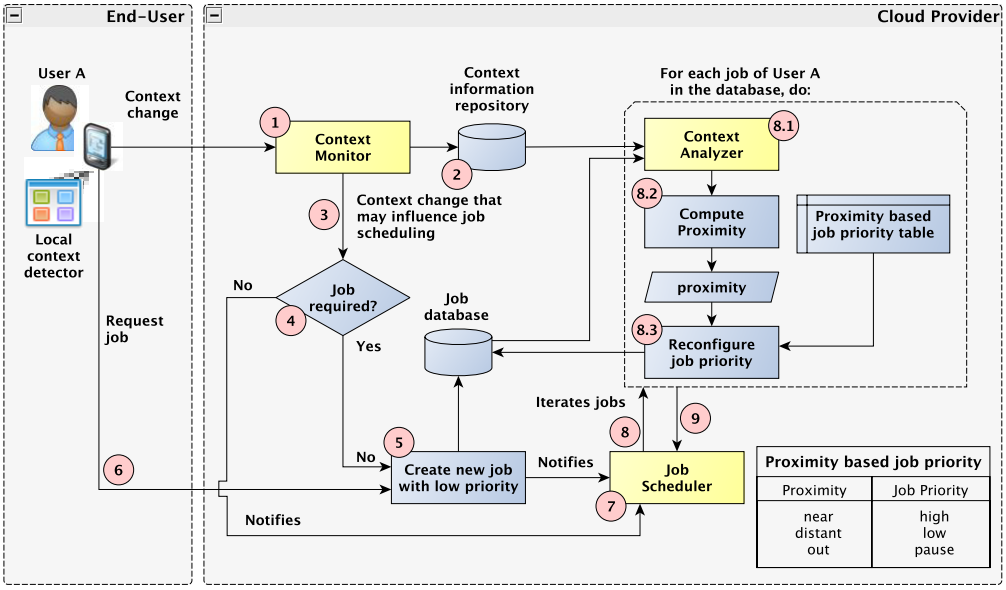


Fig. 3. Algorithm for scheduling jobs based on end-user local context.

A. Architecture

The proposed architecture comprises the five main components depicted in Figure 2, which are described as follows:

Interface: provides support to receive information through the communication structure (not presented), originated from variations of local context being captured and transmitted by the handheld devices. It is responsible for obtaining context change information, request job executions, and for getting job results.

Context Monitor: receives and interprets context information. This process (i) stores and indexes context information in the Repository of Context Information and (ii) triggers the analysis process in the Context Analyser module.

Repository of Context Information: a database that stores end-user context information – including when users change their contexts, user profiles [20] – and configuration parameters to translate context information into job priorities.

Context Analyser: works by analysing the received context information and sending recommendations to the Job Scheduler. The analysis involves (i) cross-relating the current context to historical data, other system's situational information, eventual server-based context information (*e.g.* personal information system stored on the Cloud provider), and existing jobs, forming the *global context*, and; (ii) using the computed information to calculate context proximity, such as to infer if the user is *near*, *distant*, or *out* of the window of opportunity for notification delivery.

Job Scheduler: receives recommendations for alterations of job priorities and processes them accordingly. Several existing priority-based algorithms can be used for job scheduling. For instance, the scheduler can use existing policies such as High-Priority Job First (HPJF) to sort and re-sort the jobs for execution. Moreover, jobs with low priority can be preempted to spare computing resources to high priority ones. Alternatively, more resources can be provisioned to execute high priority jobs. Different from traditional deadline-driven scheduling, our scheduler does not know the period a user will remain within a window of opportunity.

B. How does it work?

A flowchart in Figure 3 illustrates the algorithm for scheduling jobs based on user local context. The algorithm consists of the following steps:

1. The local context detector notifies the Cloud provider upon a change in the local context.
2. The notification is intercepted by the Context Monitor and stored in a Context information repository.
3. The Context Monitor determines whether the context change triggers an event that may influence the current job schedule.
4. The context change may require a new job to be created.
5. Whenever a new job is required, it is created with low priority and added to the Job database.
6. Jobs can also be created upon explicit request from the user.
7. Regardless the need to create a new job, the Job Scheduler is notified of events that may affect the schedule.
8. Upon the receipt of such an event, the scheduler iterates the jobs of the user whose context changed. For each job, the scheduler:

8.1. Invokes the Context Analyser, which gathers information about the current user context from the repository.

8.2. Computes the user proximity to the *window of opportunity* in which the job is required.
8.3. Sets the priority of the job according to the Proximity based job priority table. An example of such a table is provided, in which we define three levels of proximity. However a more elaborated table with more levels of proximity could be used.

9. Once the priorities are re-defined, the Job Scheduler can schedule (or re-schedule) the jobs using a policy defined by the administrator.

In what follows, we demonstrate the gains in resource utilisation and quality of service obtained by applying the aforementioned proposed architecture to the illustrative problem scenario.

IV. Evaluation

The basis for the design of the context-aware job scheduling is predicated on the idea that by allowing the Cloud provider to analyse end-user local contexts, it is possible to avoid resource waste and provide users with better quality of service. The experimental results demonstrate that the principle is sound.

A. Environment Setup

We evaluated the context-aware job scheduling against the traditional approach where static priorities are used to schedule remote jobs. We developed an event driven-simulator that contains the two approaches (*i.e.* context-aware scheduling and traditional scheduling). Both approaches use HPJF to sort jobs for preemption and scheduling. The simulator receives a file containing information on user contexts and application execution time. We crafted three workloads where users create jobs according to their context. Each workload contains 20,000 users, and each user creates jobs in the Cloud according to a normal distribution. Job execution times are defined according to a normal distribution as well.

We consider the workload generated by a simulation that reproduces a simplified societal behaviour. For each individual, we consider four equally distributed random behaviours described in Figure 4(a).

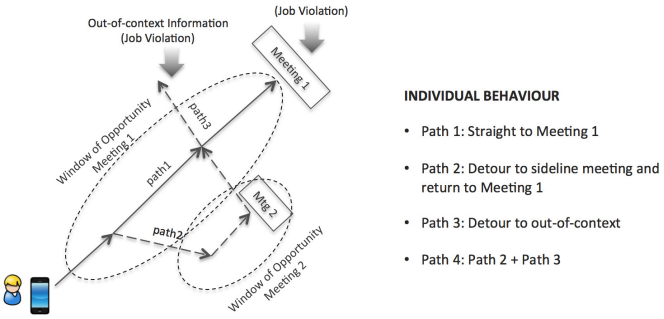
Once the user enters the window of opportunity, it generates an event to the Cloud Service requesting for the *start* of a related job. Conversely, when stepping out of that context, it issues an event to *pause* the job. In order to work properly, it is expected that the jobs' execution times are less than or equal to the size of the *window of opportunity*. Should a job take longer than that size (or not paused), the system will deliver the information outside the *window of opportunity*. This event will generate a *job violation*.

To replicate loads generated by a collective, we created a social simulation for a variable number of end-users during the day. In order to allow for reproducibility of the experiments, we depicted the workloads in Figure 4(b). We consider three variations of societal behaviours:

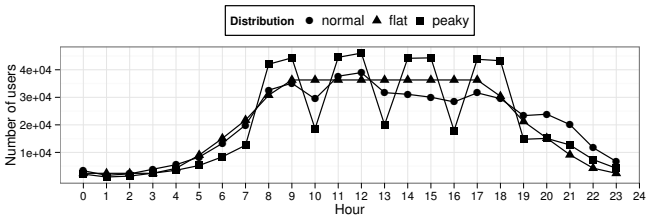
- *Normal day*: consists of small peaks of utilisation during the start, middle, and end of work hours reflecting the time when office workers check their e-mails and websites, for example. Outside these intervals, but still in work hours, this workload remains around the peak values, while outside the working hours it goes down significantly. This load is realistic and most seen in commercial production environments. This configuration is used to test the solution's behaviour handling normal situations.

- *Flat day*: consists of a flat number of end-users during the work hours. This load is unrealistic in real-world environments that involve end-users, but can reflect some automated environments. This configuration is useful to create a common denominator for a "perfect world" analysis.

- *Peaky day*: consists of tipping workload peaks during the working hour. This configuration is realistic and reflects the situation where impacting news reach the outside world, causing office workers to access their e-mail and websites often. The configuration is used to test the solution's behaviour handling stress situations.



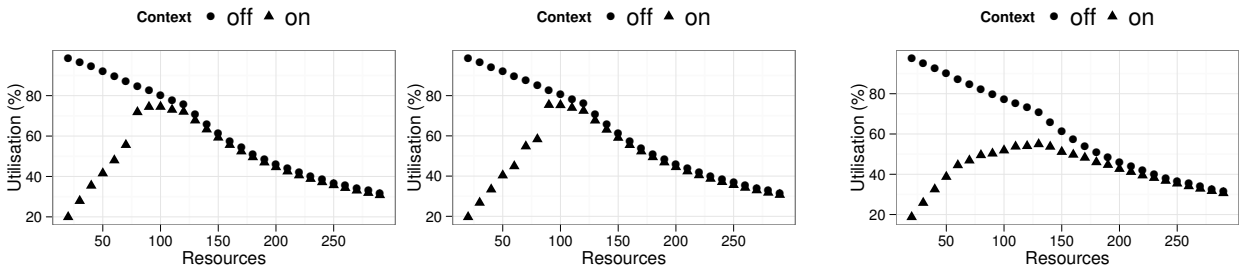
(a) Individual behaviour.



(b) Societal behaviours

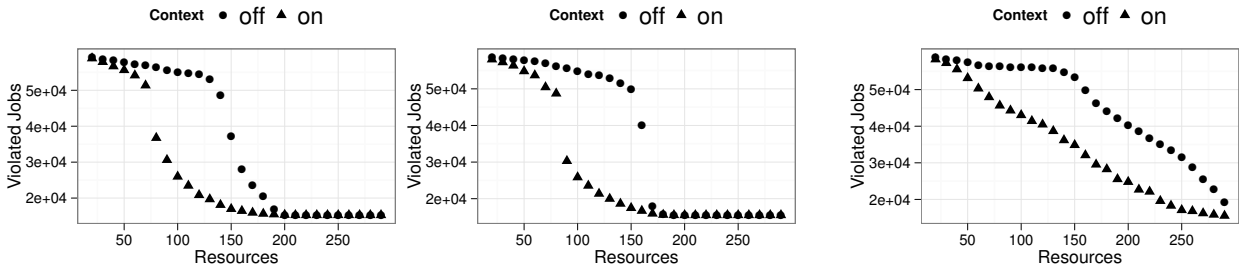
Fig. 4. Workload definition.

The objective is to reproduce the Cloud service performance whilst handling the workload generated by these three societal behaviours. The perfect system should process the jobs while the end-user is inside the window of opportunity and refraining of delivering their results when already outside it. However, due to variations of overall system workload, *e.g.* either generated by the numerous end-user's requests incoming from peak time or lack of processing resources, the system may fail to generate the jobs' results in time. This leads to unexpected *job violation* events.



(a) Normal (b) Flat (c) Peaky

Fig. 5. System utilisation as a function of the number of resources and scheduling policy with and without context.



(a) Normal (b) Flat (c) Peaky

Fig. 6. Number of violated jobs without context and with context as a function of the number of resources.

We demonstrate that the proposed solution is able to balance the amount of resources and curb the effects of peak loads by adjusting the processing prioritisation in response to the aforementioned events. The simulations' results are detailed below.

As metrics, we measured:

- **System utilisation:** Ratio of the amount of work processed and total computing capacity—number of computing resources;
- **Number of violated jobs:** Jobs that were completed after the user needed the results, for the traditional policy, and jobs that were cancelled because the results were not ready when necessary, for the context-aware scheduling policy.

In our analysis we varied the number of resources in the Cloud infrastructure to evaluate different system loads. We used the range from 20 to 300 computing resources.

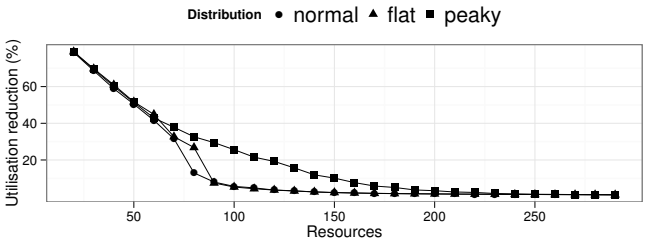
B. Results and Analysis

We start our analysis on the benefits of the context-aware job scheduling by showing results on system utilisation. Figure 5 presents how the system utilisation varies as a function of the number of resources and the use of context. When no context is considered, the system utilisation is inversely proportional to the increase in number of resources. This happens because a system with a large number of resources can handle the load more easily.

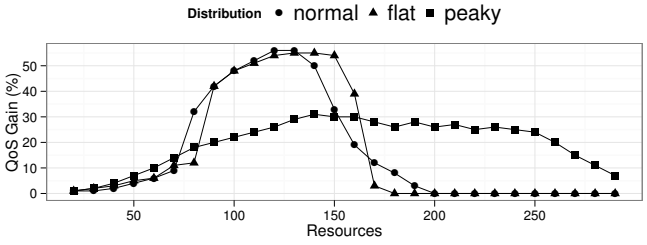
When context is used, the system utilisation starts low and increases steadily with the number of resources until it reaches a certain threshold where the utilisation is the same as when no context is used. After this threshold, which in our case is around 100 resources, both policies present the same behaviour. This occurs because with few resources, jobs wait longer to be executed and hence there is not enough time to complete their execution. If jobs are cancelled when their results are no longer required, more capacity is released, resulting in lower utilisation, which is the case of the context-aware scheduling policy. However, when the system has more resources, fewer jobs are cancelled and the utilisation increases. As the number of resources increases, a balance is established between number of cancelled jobs and system utilisation. Note that in context-aware scheduling, the jobs are cancelled when they are violated. The main conclusion for system utilisation is that context helps by cancelling jobs that are no longer necessary and filling the system with jobs whose results are more important to the end-user.

Figure 6 presents results on the number of violated jobs when varying the number of resources. This metric assesses the quality of service offered by the Cloud to the end-user. The results show that more jobs are affected when context is not used. Under extreme cases of resource scarcity or abundance (*i.e.* low or high resource capacity), the average quality of

service offered to the user is the same for both policies, except under a peaky load where the system is under-utilised. Overall, the use of context provides better quality of service when compared to approaches that do not consider context.



(a) Gain in system utilisation.



(b) Gain in Quality of Service.

Fig. 7. Benefits of context-aware job scheduling as a function of load distribution and number of resources.

The benefits of context-aware job scheduling can be summarised in Figures 7 (a) and (b), which present the system utilisation reduction and the quality of service (QoS) gains respectively. The QoS gain can be defined as the difference between the percentage of violated jobs with and and without the use of context. It is interesting to note that for both metrics, the peak load has more benefits compared to the other loads. This is because the peaky load stresses the system. This is more clearly noticed for the quality of service metric, where even when increasing the number of resources in the system, context-aware scheduling still provides gains when compared to traditional scheduling. Even when jobs are cancelled by the system, the average end-user experience is improved because users will not be notified of completion of jobs whose results are no longer required.

V. Final Remarks and Further Work

This work leverages variations of end-user local context to optimise the job scheduling in Cloud Computing environments. The research revolved around the following research question: *How to rationalise the utilisation of Cloud Computing resource by continuously adapting the execution of jobs in response to variations of the end-user's local context?*

We developed a model for smarter services that combines techniques of context awareness and adaptive job scheduling. The proposed model works by adjusting the priorities of the server-based jobs in response or pro-actively to variations of the end-user local context. It aims at providing delay-tolerant job execution required in mobile environment, while reducing the resource wastage by properly scheduling jobs in the Cloud.

We analysed the results by demonstrating gains in performance, quality of service, reduction of wasted jobs, and improvement of overall end-user experience. We considered workloads that reproduce a simplified societal behaviour. We created a social simulation for variable number of end-users during the day in three configurations: (i) *flat day*, useful to create a common denominator for a perfect world analysis; (ii) *normal day*, which reflects the workload most seen in commercial production environments, and; (iii) *peaky day*, which mirrors tipping workload during working hours. The objective was to reproduce the Cloud service performance while handling the workload generated by these three societal behaviours.

Our main findings are that:

- The impact of applying the proposed method is expressly relevant in scenarios that involve a large number of personal assistants merged with highly dynamic environments. This is the case, in our simulations, of processing the workload of the "peaky day" scenarios using normal and busy intra-day configurations.
- The proposed model provides significant positive performance impact in situations where the Cloud Computing service is under intensive resource demand. That is, by being able to adjust the priority of incoming jobs in relation to variations of local contexts, the system improves the overall resource utilisation delivering better performance and, consequently, better quality of service.

One must bear in mind that the positive results in the Cloud Computing environment come in exchange to higher resource utilisation in the mobile application environment. That is, in order to apply the proposed methods the mobile applications must be equipped with modules to sense the variations of local context and continuously communicate with the remote server. Intuitively, it will imply in more utilisation of communication channel, processing, and power supply. This situation cannot be overlooked and must be taken in consideration while planning for the overall system performance. However, we claim that advances in mobile computing technology mitigate this problem and the benefits of the proposed solution are positive in most of the application scenarios.

Finally, we foresee a number of future developments for the proposed model. We will be analysing the resource impact of variations in the simple social simulation, such as alternative behaviours like users leaving and returning to the in-context area. Intuitively, we believe this differentiated behaviour will affect solely the memory resources, but that must be validated with a new set of simulation configurations. We will also be improving aspects of the Cloud Computing simulation by adding new parameters in relation to communication and memory resource allocation. We intend to extend the Context Evaluation module to consider different domains of context evaluation, not considered in this study.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Comp. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[2] H. P. Borges, J. N. de Souza, B. Schulze, and A. R. Mury, "Automatic generation of platforms in cloud computing," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS'12)*, 2012, pp. 1311–1318.

[3] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," College of Computing, Georgia Institute of Technology, Tech. Rep. GIT-GVU-99-22, June 1999.

[4] A. Schmidt, "Ubiquitous computing - computing in context," Ph.D. dissertation, Computing Department Lancaster University, UK, Nov 2002.

[5] J. Anhalt, A. Smailagic, D. Siewiorek, F. Gemperle, D. Salber, S. Weber, J. Beck, and J. Jennings, "Toward context-aware computing: experiences and lessons," *Intelligent Systems, IEEE*, vol. 16, no. 3, pp. 38–46, 2001.

[6] 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 (PAAMS10)*, Salamanca, May 2010.

[7] C. Graham and J. Kjeldskov, "Indexical representations for context-aware mobile devices," in *Proceedings IADIS International Conference on e-Society*, 2003, pp. 3–6.

[8] H. J. La and S. D. Kim, "A conceptual framework for provisioning context-aware mobile cloud services," in *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 466–473.

[9] Q. Z. Sheng, B. Benatallah, and Z. Maamar, "User-centric services provisioning in wireless environments." *Commun. ACM*, vol. 51, no. 11, pp. 130–135, 2008.

[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 Military Communications Conference (MILCOM'11)*, nov. 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*, ser. ARM '10. New York, NY, USA: ACM, 2010, pp. 32–38.

[12] 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 ACM Symposium on Applied Computing (SAC'09)*, 2009, pp. 456–460.

[13] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn, "Parallel job scheduling - a status report," in *Proceedings of the International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'04)*, 2004.

[14] 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 IEEE International Symposium on High Performance Distributed Computing (HPDC'01)*, 2001.

[15] 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 2011 IEEE International Conference on Services Computing*, ser. SCC '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 88–95.

[16] Y. Xiao, P. Hui, P. Savolainen, and A. Ylä-Jääski, "Cascap: cloud-assisted context-aware power management for mobile devices," in *Proceedings of the second international workshop on Mobile cloud computing and services (MCS'11)*. New York, NY, USA: ACM, 2011, pp. 13–18.

[17] 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.

[18] 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'12)*, 2012.

[19] L. Capra, W. Emmerich, and C. Mascolo, "Carisma: Context-aware reflective middleware system for mobile applications," *IEEE Trans. Softw. Eng.*, vol. 29, no. 10, pp. 929–945, Oct. 2003.

[20] J. Simoes, P. Weik, and T. Magedanz, "The human side of the future internet," in *Future Internet Assembly*, 2010, pp. 183–192.