

Resource Allocation for Delay Sensitive Applications in Mobile Cloud Computing

Omar Chakroun, Soumaya Cherkaoui

INTERLAB Research Laboratory, Université de Sherbrooke, Canada

{omar.chakroun, soumaya.cherkaoui}@usherbrooke.ca

Abstract—In this work, we propose an approach for optimising Mobile Cloud Computing (MCC) resources allocation using stochastic networks optimization based on Lyapunov optimization. The approach targets reducing users' requests experienced delay while enhancing resource availability. We use Lyapunov drift plus penalty optimisation to ensure design stability and to minimise the delay cost function. We perform extensive simulations under different charge conditions in order to prove the effectiveness of the approach in handling user requests with lower delays. Simulation results show that the approach enhances the experienced delay while ensuring a good utilization of resource pools.

Index Terms— MCC, Resources management, Lyapunov optimization.

I. INTRODUCTION

Resources management in Mobile cloud computing (MCC) is a challenging issue given the number of objectives to meet while considering relatively large infrastructures. These objectives include reducing delay, providing highly available platforms and reducing resources leasing cost including energy. Studies conducted by cloud providers showed that high delays in particular, are a primary source of dissatisfaction among mobile cloud users [1]. Therefore, lowering delay is an important issue in cloud resource management [2, 3, 4].

In this work, we propose an approach for resource management for Infrastructure-as-a-Service (IaaS) cloud offerings in MCC. The approach objective is to reduce the delay experienced by cloud users, while providing resource elasticity and ensuring system stability. The delay considered here is the delay incurred from the decision of admitting cloud user requests, to requests processing completion. Elasticity is defined as the capability to adapt the system behavior to incoming requests volume and characteristics in terms of needed resources [5, 6]. In our approach, we use a processors' overclocking technique to enhance processing resources elasticity without affecting equipment's lifecycle. System stability is a known problem faced by multiusers / multi-tenants platforms, where the more requests are received by the system, the slower it becomes [7]. In our approach, we use a stochastic optimization leveraging Lyapunov drift plus penalty optimization to achieve system stability. The approach aims to maximize the number of admitted requests while keeping the system stable within its capacity region, and minimizing a delay-penalty function.

Our contributions in this work can be summarized as follows:

(1) we propose an approach for MCC resource management which leverages Lyapunov optimization to ensure system stability while minimizing delay; (2) we use an overclocking technique to provide resource elasticity; and (3) we provide an

analytical modelling of the approach and perform simulation to test system stability and delay performances. Results show that the approach enhances the experienced delay while ensuring a good utilization of resource pools.

The remainder of this paper is organized as follows: Section II details the architectural design. Section III introduces the system model to ensure system stability and resources allocation optimization. Section IV presents the simulation environment and the results. Finally, Section V concludes the paper.

II. ARCHITECTURAL DESIGN AND APPROACH

The architectural deployment for a MCC system is built around three types of actors each having a number of resource pools (RP): (1) the *Mobile Device* (M) which usually has limited resources and under the condition of insufficient resources will offloads its requests to nearby cloud components, (2) the *Cloud Edge* (CE) which is a cloudlet-like actor as introduced by Satyanarayanan et al [8]; it includes virtualized resources, can handle a certain amount of tasks, is generally implanted at the edges of the cloud close to end-users, and (3) the cloud *Data Center* (DC) which hosts extensive virtualized resources, sometimes strewn over a large geographic area. It is worth noting that CE' resources are limited compared to DC but their role is to relieve the latter from a certain amount of tasks while ensuring lower request routing delays. Since CEs are closer to the end-user they experience better resources routing delay and can implement handoff mechanisms to facilitate VM migration following users' mobility.

A. Enhancing Resources Elasticity with Overclocking

We use an overclocking technique [9] both at CEs and DC. Using an overclocking technique at CEs allows processing requests faster and admitting more requests. Using overclocking at the DC allows limiting the impact of virtual machines (VMs) migration, on the performance of VMs coexisting on the same physical machine (PM). Overclocking in DC and CE can ensure some processing resources elasticity and reduce the delay needed to process a task. In order not to decrease physical components' lifetime, involve extra cooling systems, or increase voltage usage, we use overclocking up to a limit of 15% of the processor maximum clock speed. Overclocking can increase processor speed up to 20% without any change to components [9], but running a processor above its maximum speed can make the system consume more power. However, studies showed that the extra power involved is relatively low [9] (from 2W-6W/ processor per 200 MHz overclocking step up to 600 MHz [10]).

Overclocking can be of particular interest in MCC if the gap between the needed and available processing capability does not exceed 10% of the total processor capabilities. We take advantage of this important aspect in our approach. Many proposed MCC resource management architectures, leave 10% of servers' resources for potential VM migration tasks [10, 11]. This 10% of servers' resources is unusable for user requests processing. Using overclocking, CEs for example, can handle some extra user requests without the incurrance of extra delays needed to route some requests to the DC. Furthermore, overclocking enhances requests processing delay at the CE and DC. Eq. 1 shows the relationship between processing delay, the gain in frequency due to overclocking f_{over} and the amount of processing cycles needed δ [1].

$$D_p(\delta, f) = \frac{\delta}{f + f_{over}} \quad (1)$$

B. Request Processing

Requests can be either processed on the mobile device side, or be offloaded to a nearby RP (CE or DC). In our approach, the choice of the RP to handle the admitted requests will favor the one which lowers the experienced delay, provided that the requests can be admitted. The choice of RP can be characterized by an optimization process as in Eq. 2.

$$\begin{aligned} & \text{Min}\{D_S(\delta, f_M); D_S(\delta, f_{CE}) + D_N(O, CE); D_S(\delta, f_{DC}) + D_N(O, DC) \\ & \text{s.t. } f_{CE} \leq f_{CE_N} + f_{CE_{OVER}}, f_{DC} \leq f_{DC_N} + f_{DC_{OVER}}, \\ & 0 \leq f_{OVER} \leq 0.15f_i, i \in \{CE, DC\} \end{aligned} \quad (2)$$

where $i \in \{M, CE, DC\}$, $j \in \{N, OVER\}$ denotes the different processing frequencies in the RPs (M, CE, DC) while activating overclocking (OVER) or using the nominal processor speed (N). The total delay DS denotes the sum of the processing delay DP and the queuing delay DQ on the resource pool side (M, CE, DC). DN denotes the request routing delay from request originator O to the RP that will handle its processing.

III. PROBLEM FORMULATION AND THEORETICAL ANALYSIS

We consider the architectural deployment of Fig. I, where each actor (M, CE, DC) comprises a RP of virtualized resources. We consider that RPs are managed using time slots. At each time slot (TS), the admission controller determines whether to admit or decline the new requests. We assume that the system state does not change during TS. The admitted requests are stored on the router buffer before being routed to the appropriate RP through the dispatcher. Each server on one or the other of the RPs has a set of resources W_j .

A. Problem Formulation

We consider an RP with S servers that host a set of N applications denoted by A . Each server j hosts a subset of applications. It allocates one VM per application. We define indicator function e_{ij} as follows; equal 1 if application a_i is hosted on server j , 0 otherwise. For simplicity, we consider that each application constitutes a single tier. We consider three main RPs as discussed earlier. At every TS new requests arrive for application a_i with a rate λ_i according to a random process

which is independent from the amount of unfinished work. We suppose that we have no knowledge of the statistics of these arrivals. Let $A_i(t)$ denote the new requests arriving for application a_i in TS t . Let $R_i(t)$ be the requests admitted by the admission controller and stored in the router buffer for application a_i . Let us denote the buffer by $W_i(t)$ and assume that requests which are not admitted are automatically rejected or stored in a buffer outside of the queuing system and will be held for future processing at the subsequent TS. Thus we have:

$$\forall i, t \quad 0 \leq R_i(t) \leq A_i(t).$$

Let us denote $R_{ij}(t)$ the requests for application a_i that are routed to RP $_j$ in TS t . Then the router queue dynamic can be characterized by (3).

$$W_i(t+1) = W_i(t) - \sum_j R_{ij}(t) + R_i(t) \quad (3)$$

Let $S_i(t)$ denote the set of servers capable of handling the application a_i at TS t thus the routing decision must satisfy the following constraints (4) and (5) at every TS.

$$R_{ij}(t) = 0 \text{ if } e_{ij} = 0 \quad (4)$$

$$0 \leq \sum_{j \in S_i(t)} e_{ij} R_{ij}(t) \leq W_i(t) \quad (5)$$

Let us denote the set of control action available at the RP $_j$ level at TS t under any control policy by $I_j(t)$ and let $D_j(t)$ the corresponding processing delay. Then the queuing dynamics of the requests of the application a_i on RP j follows (6) where $\mu_{ij}(I_j(t))$ denotes the service rate for the application a_i at RP j under the control decision $I_j(t)$.

$$U_{ij}(t+1) = \max[U_{ij}(t) - \mu_{ij}(I_j(t)), 0] + R_{ij}(t) \quad (6)$$

We assume that the expected value of the service rate can be known since, as discussed in section II, we can derive the experienced processing delay based on the processor frequency assignment. Thus at every TS t , the following decisions have to be made: 1) Admission control decision for all application a_i requests $R_i(t)$, 2) Routing decision for the admitted requests $R_{ij}(t)$, 3) Resource allocation decision $I_j(t)$ including resources distribution among VMs and activation of the overclocking.

Consider now a policy η that takes control decisions denoted by $S^\eta(t)$, $R_i^\eta(t)$, $R_{ij}^\eta(t)$, $I_j^\eta(t)$, $D_j^\eta(t)$. These control decisions refer respectively to the set of servers, the admitted requests for application a_i , the requests for application a_i routed to RP $_j$, the resources allocation decision, and the processing delay, all of which applied under policy η for all i, j at TS t . This control policy has to satisfy the routing constraints (4), (5) and the resources allocation constraints (6) every TS for all i, j .

Let us denote the average expected rate of admitted requests for the application a_i under control policy η by r_i^η and the average expected processing delay under the same conditions by d_j^η .

$$r_i^\eta = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{R_i^\eta(\tau)\} \quad (7)$$

$$d_j^\eta = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E\{D_j^\eta(\tau)\} \quad (8)$$

Thus considering a collection of non-negative weights α_i, β , our objective is to design a control policy η that solves:

$$\begin{aligned} \text{Maximize: } & \sum_{i \in A} \alpha_i r_i^\eta - \beta d_j^\eta \\ \text{S.T: } & 0 \leq r_i^\eta \leq \lambda_i \end{aligned} \quad (9)$$

Thus the problem is a weighted linear combination of the sum throughput of the applications and the average delay in the RP.

B. Delay-Dependent Optimization based on Lyapunov

Let $V > 0$ be a control parameter that has to be chosen by the system administrator to ensure the desired tradeoff between the performance and the experienced delay. Let $W_i(t)$ and $U_{ij}(t)$ be the router and the RP queue backlog at TS t . As the backlog values evolve over time as in (3) and (6), the algorithm adapts to system solves the problem in (9) leveraging a sequence of optimization problems over time on three distinct steps.

Admission control:

For each application i , choose the number of new requests to be admitted as the solution of the following optimization.

$$\begin{aligned} \text{Maximize: } & R_i(t) [V \alpha_i - W_i(t)] \\ \text{ST: } & 0 \leq R_i(t) \leq A_i(t) \end{aligned} \quad (10)$$

Thus for this sub-problem we have a simple threshold solution such as if $W_i(t) > V \alpha_i$ then no request are admitted else if $W_i(t) \leq V \alpha_i$ then all requests are admitted and $R_i(t) = A_i(t)$.

Request routing: let denote j^* the server hosted in a RP that is having the smallest queue backlog and belonging to the set of servers thrown over multiple RPs that are able to process the requests for application a_i . Thus, a routing policy can redirect all requests for application a_i to such a server under the condition that $W_i(t) > U_{ij^*}(t)$.

Resources allocation: at each server j , choose the resource allocation $I_j(t)$ that solves the Lyapunov optimization process (11) where $d_{\max}$ denotes the maximum experienced delay if processors are at the lowest processing speed and requests are experiencing the highest request routing delays.

$$\begin{aligned} \text{Maximize: } & \sum_{i \in A} U_{ij}(t) E \{ \mu_{ij}(I_j(t)) \} - V \beta d_j(t) \\ \text{ST: } & d_j(t) \leq d_{\max} \end{aligned} \quad (11)$$

IV. RESULT OVERVIEW

A. Simulation Environment

Table I below summarizes the main simulation parameters. We simulated three RPs; (1) a mobile device (M) comprising 1 server, (2) a Cloud Edge (CE) hosting a set of 4 servers and (3) a data center (DC) hosting 100 servers. The model integrates a switching box that routes the request to the appropriate RP while using Lyapunov optimization. Servers' nominal processors speed ranges from 1.4GHz to 2.8GHz (without overclocking). The number of requests arriving to the system is randomized and ranges from 0 to 100 requests per TS.

Each request is characterised by the needed processing delay. Servers are assumed to be able to complete a request in a maximum delay ranging between 2 and 5 TS depending on the processing frequencies. Each RP has an internal router where requests are stored temporarily waiting for resources to be available. The simulation duration is 1000 TS.

TABLE I – GLOBAL SIMULATION PARAMETERS

Parameter	Value
Number of servers	M(1), CE(4), DC(100)
Routers queues length	M(1), CE(4), DC(100)
Number of request arriving to the system	0-100 per TS
Servers processing frequency	1.4 – 2.8 GHz
Processing capabilities per server	2 - 5 TS per request
Overclocking maximum rate	15% (up to 3.2GHz total)
Requests routing delays in TS	M(0), CE(0.5), DC (0.7)
Simulation duration	1000 TS

B. Results Overview and Analysis

We are interested in assessing the gain in the experienced delay from request arrivals to the system, to their completion on one of the RPs. We compare three approaches; (1) the standard approach where requests are routed proportionally to one or the other of the RPs, (2) Lyapunov optimization approach where requests are routed to the RP that is ensuring the lowest delays while admitting the maximum number of requests, and (3) Lyapunov optimization and overclocking approach combination where requests are routed to the RP that is ensuring the lowest delays, and admitting the maximum number of request while RPs servers are implementing the overclocking technique. It is worth noting that overclocking technique is activated on the CE and the DC sides only.

Fig. I shows the experienced delay from request generation to its processing completion for the three approaches. Fig. I (A) shows the case where we use the standard approach. The figure shows that without the adequate request routing technique, the system will experience poor performances. With the standard approach, the RPs only admit around 230 request each for the total simulation duration (see Fig. II(A)) while ensuring 8 TS, 4.5 TS and 4.7 TS delays for respectively the M, CE and DC. On the other hand, under Lyapunov optimization, RPs will ensure higher delays while admitting more than 2.2k requests (see Fig. II(B)) for the total simulation duration. CE and M can handle up to 350 requests and 230 requests respectively while experiencing higher delays. This is explained by the fact that they are handling more requests, and the waiting delays in the routers increase because servers are more utilized. Combining Lyapunov optimization and overclocking technique ensures lower delays since servers can be run to 115% of their capabilities. We notice that the delay experienced by the mobile slightly decreases since its server is less utilized and most of the requests are routed to the DC and CE. For the CE and DC, on the other hand, we notice a delay enhancement by around 12.5%.

Fig. II shows the cumulative number of requests processed in the different RPs for the three distinct approaches. Under Lyapunov optimization, the number of admitted and processed requests is higher than the standard approach. Meanwhile, combining overclocking and Lyapunov optimization ensures better performances in term of processed requests while ensuring better delays.

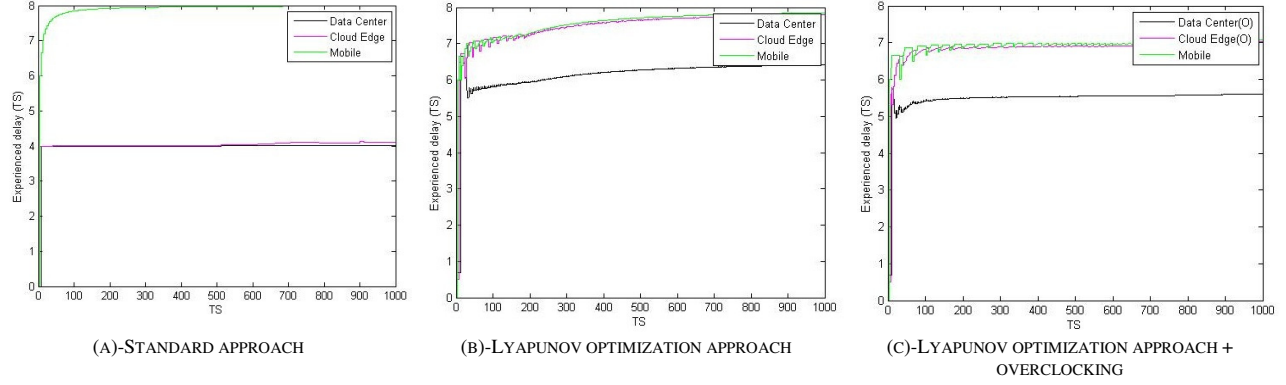


FIGURE I— EXPERIENCED DELAY IN THE DIFFERENT RPs OVER SIMULATION TIME

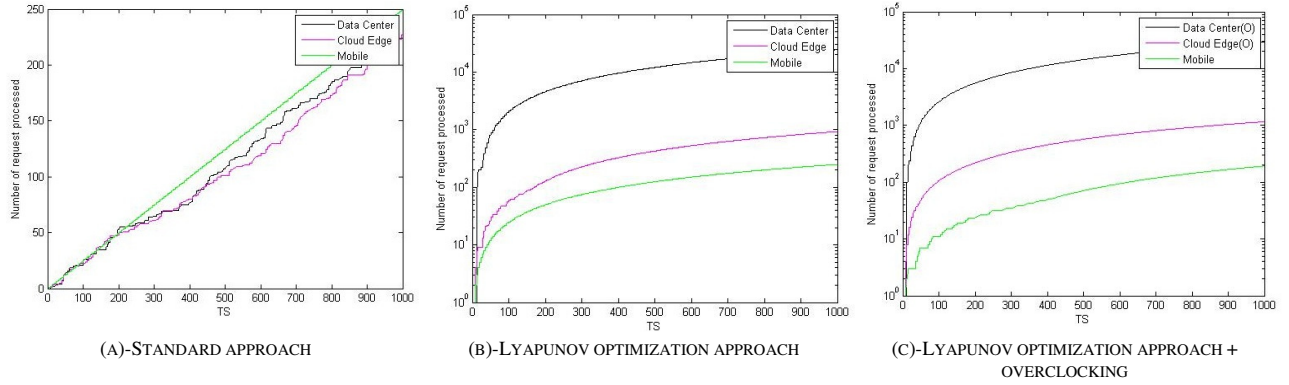


FIGURE II— CUMULATIVE NUMBER OF REQUESTS PROCESSED IN RPs

Combining overclocking and Lyapunov optimization allows handling the processing of more than 13k requests while deactivating overclocking allows the system to overall handle only around 17% of the requests processed by combining both techniques.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a resource management approach based on Lyapunov optimization and overclocking in order to reduce the delay experienced by cloud users while enhancing elasticity and ensuring system stability. We performed a performance study while considering resource pools processing requests at the mobile, the cloud edge and the data centers. Results of the study present the impact of the approach on lowering the overall experienced delay while ensuring better resources utilization. Simulations show that, even at high servers' charge and high request volumes, the approach ensures system stability while leading to a gain up to 12.5% in the experienced delay compared to the Lyapunov optimization alone, while allowing the processing of higher numbers of requests.

REFERENCES

- [1] D. A. Patterson and J. L. Hennessy. 1990. *Computer Architecture: a Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *SIGCOMM Comput. Commun. Rev.* 39, 1 (December 2008), 68-73. DOI=<http://dx.doi.org/10.1145/1496091.1496103>
- [3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1-122, 2010.
- [4] X. Xu and H. Yu, "A Game Theory Approach to Fair and Efficient Resource Allocation in Cloud Computing," *Mathematical Problems in Engineering*, vol. 2014, Article ID 915878, 14 pages, 2014. doi:10.1155/2014/915878.
- [5] N. R. Herbst, S. Kounev, R. Reussner, "Elasticity in Cloud Computing: What It Is, and What It Is Not," *Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013)*, San Jose, CA, June 24-28.
- [6] R. Buyya, J. Broberg, A. M. Goscinski, *Cloud Computing Principles and Paradigms*, John Wiley and Sons, 2011, ISBN 978-0-470-88799-8
- [7] http://www.xbitlabs.com/articles/cpu/display/power-consumption-overclocking_10.html#sect
- [8] M. Satyanarayanan, P. Bahl, R. Caceres and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," in *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14-23, Oct.-Dec. 2009. doi: 10.1109/MPRV.2009.82.
- [9] H. B. Jang, et al. "Leveraging Process Variation for Performance and Energy: In the Perspective of Overclocking." *IEEE Transactions on Computers* 63.5 (2014): 1316-1322.
- [10] <http://www.choixpc.com/overcloc.htm>
- [11] I. Takouna, R. Rojas-Cessa, K. Sachs, C. Meinel, "Communication-Aware and Energy-Efficient Scheduling for Parallel Applications in Virtualized Data Centers," in *Utility and Cloud Computing (UCC)*, 2013 IEEE/ACM 6th International Conference on, vol., no., pp.251-255, 9-12 Dec. 2013 doi: 10.1109/UCC.2013.50
- [12] S. Bagchi, *Emerging Research in Cloud Distributed Computing Systems*, 2015.