A Load-balancer for Brownout-compliant Cloud Applications

AUTHOR1 [*] **AUTHOR2** [*] **AUTHOR3** [**] **AUTHOR4** [*]

[*] *Department of Automatic Control, Lund University*
[**] *Department of Computer Science, Umeå University*

Abstract:

Keywords: Computer systems, Feedback loops, Model-based control, Multiprocessor systems, Probabilistic models, Queuing theory.

1. INTRODUCTION

Cloud computing is expected to be the driving force of future IT innovation **?**. Instead of forcing businesses to make costly and risky investment in IT infrastructure, public clouds, such as Amazon EC2, allow such businesses to rent computing infrastructure as they grow, with no upfront costs **?**. In fact, the model proved so successful, that companies are now converting their IT infrastructure to so-called private clouds, i.e., data-centers managed using cloud technologies, to improve IT efficiency **?**.

In cloud computing, three roles can be distinguished: the **Infrastructure Provider (IP)**, who owns and operates the cloud infrastructure, i.e., the physical hardware; the **Service Provider (SP)**, who owns and operates cloud applications running on cloud infrastructures; the **end-user**, who uses a cloud application through a network, commonly the Internet. Under the hood, cloud computing works by giving SPs the illusion of unrestricted (i.e., super-user) access to hardware, in the form of Virtual Machines (VMs). Besides simplifying certain tasks for SPs by hiding the complexity of the underlying infrastructure, VM also increase the flexibility of the IP by allowing an arbitrary mapping of physical hardware resources, such as CPU, memory and storage, to VMs.

A defining characteristic of clouds is **elasticity**. It is defined as the ability of the SP to rapidly and automatically acquire and release computing resources **?**. One can distinguish to complementary types of elasticity: vertical and horizontal. Vertical elasticity consists in adding or removing resources from an existing VM, for example, by adding a virtual core or more memory. Horizontal elasticity consists in adding a new VM to the pool of the SP. Vertical elasticity is simpler to implement, as SP's cloud application can immediately take advantage of the new resources, but is limited to the capacity of a single physical machine, since a VM cannot span multiple physical machines. In contrast, provided enough resources are available, horizontal elasticity is only limited to the total capacity of the data-center, however, it is more complex to implement: New replica of the application have to be launched inside the new VMs, replicas have to be kept synchronized and end-users have to be redirected to new VMs, using a specialized component called a **load-balancer**. These techniques are well-known and have already been successfully applied in practice.

However, an existing issue with clouds is their robustness to unexpected events. For example, flash-crowds are sudden increases of end-users, that may increase the required capacity by up to 5 times **?**. Similarly, hardware failures may temporarily reduce the capacity of the data-center, while the failure is repaired. Due to the large magnitude and short duration of such events, it may be economically too costly to provision enough hardware to properly deal with them. As a results, unexpected events may lead to data-center overload, which translates to unresponsive cloud applications, leading to dissatisfied end-users and revenue loss.

In previous work **?**, we have proposed a cloud application development paradigm called **brownout**. In essence, some code of the application is marked as optional, such that it can be deactivated as necessary. While the execution of such code is desirable, as it improves end-user experience, its deactivation is preferred to overloading the data-center. For example, online stores display personalized recommendations of similar products. No doubt, such recommendations improve the experience of the end-user and the revenue of the SP, however, due to their sophistication, they are also very resource demanding. Hence, the SP desires to serve as many recommendations as possible, however, she would prefer to stop serving them instead of having an unresponsive application. Our previous contribution not only consists in a general architecture for brownout-compliant applications, but also a controller to decide the probability of serving optional content as a function of the application's response time. While the approach proved to be a success for dealing with unexpected events, it is restricted to applications having a single replica, running inside a single VM.

In this article, we extend our previous work to multiple VMs. This brings two improvements to brownout-compliant cloud applications. First, horizontal elasticity can be used to take advantage of more resources than provided by a single physical machine. Second, applications can be made more resilient by deploying several replica. Achieving this requires a brownout-compliant load-balancer to decide how to divide end-users between the replicas. Existing, state-of-the-art load-balancers direct load based on the response-time of each replica, which leads to inefficient decisions for brownout-compliant applications, since such applications already keep their response-time at a given setpoint (at the expense of reducing the amount of optional content served). Thus, the challenge is to find a load-balancing architecture and algorithm that maximizes the amount of served optional content.

Our contribution is threefold:

(1) We propose a load-balancing architecture for brownout-compliant cloud applications and formalized a problem statement (Section 2).
(2) Using control theory, we design a load-balacing algorithm that maximizes the performance of brownout-compliant applications (Section 3).
(3) We evaluate the approach and show that it improves the number of served optional content when compared with existing load-balancing algorithms (Section 4).

2. PROBLEM STATEMENT

3. SOLUTION

4. EVALUATION

5. RELATED WORK

6. CONCLUSION

ACKNOWLEDGEMENTS

This work was partially supported by the Swedish Research Council (VR) under contract number C0590801 for the project Cloud Control and through the LCCC Linnaeus Center. Also, we received partial support from the ELLIIT Excellence Center.

REFERENCES