Control Engineering Practice 52 (2016) 24–34

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Control-based load-balancing techniques: Analysis and performance evaluation via a randomized optimization approach[☆]

Alessandro Vittorio Papadopoulos[a,*], Cristian Klein[b], Martina Maggio[a], Jonas Dürango[a], Manfred Dellkrantz[a], Francisco Hernández-Rodriguez[b], Erik Elmroth[b], Karl-Erik Årzén[a]

[a] *Department of Automatic Control, Lund University, Lund, Sweden*
[b] *Umeå University, Umeå, Sweden*

ARTICLE INFO

Article history:
Received 28 October 2015
Received in revised form
5 February 2016
Accepted 29 March 2016
Available online 18 April 2016

Keywords:
Load-balancing
Randomized optimization
Cloud control

ABSTRACT

Cloud applications are often subject to unexpected events like flashcrowds and hardware failures. Users that expect a predictable behavior may abandon an unresponsive application when these events occur. Researchers and engineers addressed this problem on two separate fronts: first, they introduced replicas – copies of the application with the same functionality – for redundancy and scalability; second, they added a self-adaptive feature called *brownout* inside cloud applications to bound response times by modulating user experience. The presence of multiple replicas requires a dedicated component to direct incoming traffic: a load-balancer.

Existing load-balancing strategies based on response times interfere with the response time controller developed for brownout-compliant applications. In fact, the brownout approach bounds response times using a control action. Hence, the response time, that was used to aid load-balancing decision, is not a good indicator of how well a replica is performing.

To fix this issue, this paper reviews some proposal for brownout-aware load-balancing and provides a comprehensive experimental evaluation that compares them. To provide formal guarantees on the load-balancing performance, we use a randomized optimization approach and apply the scenario theory. We perform an extensive set of experiments on a real machine, extending the popular lighttpd web server and load-balancer, and obtaining a production-ready implementation. Experimental results show an improvement of the user experience over Shortest Queue First (SQF)—believed to be near-optimal in the non-adaptive case. The improved user experience is obtained preserving the response time predictability.

1. Introduction

Cloud computing has dramatically changed the management of computing infrastructures. On one hand, public infrastructure providers, such as Amazon EC2, allow service providers, such as Dropbox and Netflix, to deploy their services on large infrastructures with no upfront cost (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009), by simply leasing computing capacity in the form of Virtual Machine (VMs). On the other hand, the flexibility offered by cloud technologies, which allow VMs to be hosted by any Physical Machine (PM) (or server), favors the adoption of private clouds (Gulati, Shanmuganathan, Holler, & Ahmad, 2011). Therefore, also self-hosting service providers are converting their computing infrastructures into small clouds.

However, the scalability of these infrastructure is a serious concern and due to their ever-increasing complexity, and hardware failures are rather common. In fact, in cloud computing infrastructures failures are the norm rather than an exception (Barroso, Clidaras, & Hölzle, 2013; Guan & Fu, 2013). This is why internet-scale interactive applications, such as e-commerce websites, include replication early in their design (Hamilton, 2007). Replication improves scalability, i.e., more users can be served by adding more replicas, but also makes the application more resilient to failures. In case a replica fails, other replicas can take over.

In a replicated setup, a load-balancer is responsible for monitoring replicas' health and directing requests as appropriate. Indeed, this practice is well established and using it, applications can successfully deal with failures as long as the remaining computing

☆This work was partially supported by the Swedish Research Council (VR) for the projects "Cloud Control" and "Power and temperature control for large-scale computing infrastructures", and through the LCCC Linnaeus and ELLIIT Excellence Centers.

* Corresponding author.
E-mail addresses: alessandro.papadopoulos@control.lth.se (A.V. Papadopoulos), cklein@cs.umu.se (C. Klein), martina.maggio@control.lth.se (M. Maggio), jonas@control.lth.se (J. Dürango), manfred@control.lth.se (M. Dellkrantz), hernandf@cs.umu.se (F. Hernández-Rodriguez), elmroth@cs.umu.se (E. Elmroth), karlerik@control.lth.se (K.-E. Årzén).

http://dx.doi.org/10.1016/j.conengprac.2016.03.020
0967-0661/© 2016 Elsevier Ltd. All rights reserved.

capacity is sufficient (Hamilton, 2007).

However, failures in cloud infrastructures are often correlated in time and space (Gallet et al., 2010; Yigitbasi, Gallet, Kondo, Iosup, & Epema, 2010). Therefore, it may be economically inefficient for the service provider to provision enough spare capacity for dealing with all failures in a satisfactory manner. In case correlated failures occur, the service may *saturate*, i.e., it may no longer be able to serve users in a timely manner. This in turn leads to dissatisfied users, that can abandon the service. Note that the saturated service causes infrastructure overload, which by itself may trigger additional failures (Chuah et al., 2013), aggravating the initial situation. This strongly motivates the need for a solution to deal with rare, and cascading failures, that produce temporary capacity shortages on the underlying architecture.

A promising self-adaptation technique that solves this problem is *brownout* (Klein, Maggio, Årzén, & Hernández-Rodriguez, 2014; Maggio, Klein, & Årzén, 2014). In essence, a service is extended to serve requests in two modes: with mandatory content only, such as product description in an e-commerce website, and with both mandatory and optional content, such as the product information and recommendations of similar sales. Serving more requests with optional content, increases the revenue of the provider up to 50% (Fleder, Hosanagar, & Buja, 2010), but also greatly affects the capacity requirements of the service. The tradeoff between the two is a matter of quality of service, and it has been explored using control theory, where a controller was used to decide the percentage of requests to be served with optional content enabled (Klein, Maggio, et al., 2014; Maggio et al., 2014), in order to keep the response time below the user's tolerable waiting time (Nah, 2004). These studies were conducted in a single replica case and subsequently extended to the multiple replica, by adding a load-balancer (Dellkrantz, Maggio, Klein, Papadopoulos, & Hernández-Rodriguez, 2014; Klein, Papadopoulos, et al., 2014).

Despite the fact that load-balancing techniques have been widely studied (Barroso et al., 2013; Birdwell et al., 2003; Lin, Liu, Wierman, & Andrew, 2012; Lu et al., 2011; Nakrani & Tovey, 2004; Robertsson, Wittenmark, Kihl, & Andersson, 2004; Tang, Birdwell, Chiasson, Abdallah, & Hayat, 2008), state-of-the-art load-balancers forward requests based on metrics that cannot discriminate between a replica that is avoiding overload by not executing the optional code and a replica that is not subject to overload. This called for an analysis on the introduction of brownout-aware load-balancers (Dürango et al., 2014). The promising results of the analysis encouraged the implementation of brownout-aware load-balancers (Klein, Papadopoulos, et al., 2014). The implementation was compared with the state-of-the-art load-balancing algorithm that obtained the best performance in the simulation campaign, Shortest Queue First (SQF) (Gupta, Harchol Balter, Sigman, & Whitt, 2007).

Despite obtaining promising simulation results (Dürango et al., 2014) and good implementation performance (Klein, Papadopoulos, et al., 2014), the load-balancers were tested only in specific scenarios and no guarantees on the generality of the results were given. Moreover, since the events that can happen at the data center level are unpredictable and unknown, it is impossible to construct a set of scenarios that would make the validation extensive enough for a production environment. To address the problem of providing formal guarantees despite unpredictability, this paper applies the *scenario theory* (Calafiore & Campi, 2005; Campi & Garatti, 2011; Nemirovski & Shapiro, 2006) to obtain formal probabilistic guarantees about the behavior of the load-balancing strategies. The scenario theory has been recently used in many different situations especially when dealing with stochastic or uncertain systems (Calafiore & Campi, 2006; Campi, Garatti, & Prandini, 2009; Papadopoulos & Prandini, 2014), also in combination with robust optimization (Margellos, Goulart, & Lygeros, 2014). The application of the scenario theory allows one to extensively evaluate the performance of different techniques, providing some formal guarantees via the solution of a chance-constrained optimization problem.

In essence, the contribution of this paper is to provide probabilistic guarantees on the behavior of brownout-aware load-balancers and on the improvements they bring in comparison to brownout-unaware ones. The solution of this problem greatly extends the contribution of Dürango et al. (2014), that was only proposing time-based algorithms and evaluating them with a simulator, and of Klein, Papadopoulos, et al. (2014), that only used specific scenarios as case studies for the evaluation, therefore not being able to derive any guarantee on the generic behavior of the proposed load-balancers. Solving this problem required an extensive experimental campaign – this paper considers about 800 experiments for each load-balancing strategy against the 30 considered before. Also, the evaluation campaign conducted for this extension considers a more realistic experimental setting – both the workload characterization and the resource availability are here randomized. The adoption of the scenario theory and the required additional data provide probabilistic guarantees on the worst-case obtainable performance – these guarantees can not be obtained with the bare statistical analysis conducted in the past.

The solution proposed in this paper can be applied to a wider class of applications. Indeed, in self-adaptive software systems, there is a strong need for a different type of evaluation campaign, where all the sources or randomness in the execution are properly identified and all the possible configurations are potentially tested. This paper presents the first attempt of using the scenario theory to provide this analysis.

Results show that the cloud application can tolerate more replica failures and that the novel load-balancing algorithms improve the number of requests served with optional content, by up to 5%, the previous results of Klein, Papadopoulos, et al. (2014) on specific cases. In addition, the worst case scenario is improved by 15% with respect to SQF, with a very high probability that is quantified through the application of the scenario theory. This result is especially remarkable, since SQF is believed to be near-optimal, in the sense that it minimizes the average response time for non-brownout-enabled replicas (Gupta et al., 2007; Kuri & Kumar, 1995).

The rest of the paper is organized as follows. Section 2 discusses the state of the art, while Section 3 describes in more detail the brownout framework, better highlighting the contribution of this paper. Section 4 details the proposed control-theoretical load-balancers and some of their implementation aspects. Section 5 introduces the scenario theory and its application to test the different load-balancing strategies, highlighting the obtainable formal guarantees. Section 6 presents an extensive set of experiments that are evaluated with the scenario theory, and compared with previous results. Section 7 concludes the paper and sketches possible future work.

To make our results reproducible and foster further research on improved resilience through brownout, the source code for the load-balancer and to run the experiments has been released online.[1]

2. Related work

Load-balancers are standard components of internet-scale services (Wang, Pai, & Peterson, 2002), allowing applications to achieve scalability and resilience (Barroso et al., 2013; Hamilton, 2007; Lin & Kulkarni, 2013; Wolf & Yu, 2001). Many load-balancing policies have been proposed, aiming at different optimizations, spanning from equalizing processor load (Stankovic, 1985) to managing memory pools (Diao et al., 2005; Patterson, Gibson, Ginting, Stodolsky, & Zelenka, 1995), to specific optimizations for

[1] https://github.com/cloud-control/brownout-lb-lighttpd