SALSA: QoS-aware load balancing for autonomous service brokering

Bas Boone, Sofie Van Hoecke, Gregory Van Seghbroeck

Supervisor(s): Bart Dhoedt, Filip De Turck

I. Introduction

Nowadays, many new applications are constructed through integration of already available service components. The approach is made possible through the Service Oriented Architecture (SOA) and "Software as a Service" paradigm, typically using web service technologies to publish, discover and integrate service components. This technology also allows to replicate web services on new servers to scale in response to the needed demands. Instead of hard coding service calls in the customer's source code, brokers provide dynamic service selection to automatically select and seamlessly link the services in order to meet the business system requirement, optimize response times or reduce the costs. By using web service brokers, customers only have to interact with the service broker, hiding the complexity of selecting the appropriate service.

In a commercial application typically a Service Level Agreement (SLAs) can be mediated between the customers and the service providers defining the functional and non-functional requirements such as the levels of availability, performance and billing. Often, a service provider also wants to service a class of customers on a best effort basis. In the case of performance, the SLA usually specifies constraints on the response time. If no special precautions are taken, unexpected request patterns can drive a web server into overload, leading to poor performance since the server is unable to keep up with the demands. Service providers can solve this problem by over-dimensioning their resources and provide dedicated servers for premium customers to meet their SLAs. If we want to avoid using dedicated servers, intelligent autonomous service brokering is needed.

B. Boone is with the Information Technology Department, Ghent University (UGent), Gent, Belgium. E-mail: bas.boone@intec.ugent.be .

II. SALSA

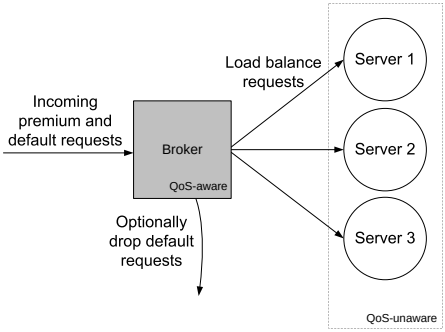

Figure 1. Objective of the Simulated Annealing Load Spreading Algorithm

We have examined two requirements for service brokers: on one hand, the broker should be able to autonomously guarantee constraints on the response time for premium clients by fulfilling a n-percentile on the response time, i.e. the value for which at most n% of the response times are fulfilled in less than that value. On the other hand, brokering should be transparent for the actual servers executing the service: they should not need to be changed.

In order to fulfill these requirements, the Simulated Annealing Load Spreading Algo-

rithm (SALSA) presented here can load balance requests, and selectively drop some requests from the default users to reduce the web servers' load in order to guarantee SLA to premium customers and provide best effort to default customers (see Figure 1).

The web servers are modeled as M/M/1 systems using standard queueing theory [2], and the broker is modeled as a statistical switch: it forwards requests randomly to a server with a given probability, and it drops default requests with a certain probability. The SALSA algorithm determines the best possible values for these probabilities, given the arrival rate of default and premium requests, and the processing intensity of the different web services. In order to do this, we define a score function which considers average waiting times, overloaded servers, exceeded waiting time thresholds for premium clients, and the number of dropped default requests. This score function is optimized for the switching probabilities using Simulated Annealing [3], a generic optimization heuristic.

III. Evaluation

We implemented the SALSA algorithm and performed simulations to test the results, using a theoretically derived optimality criterion. These tests confirmed that SALSA is able to find optimal solutions. The results were compared with those of Weighted Round Robin (WRR), nowadays the most commonly used load balancing algorithm, and it was found that SALSA was able to guarantee the threshold requirement for higher arrival rates than WRR.

We also implemented a testbed evaluation. Our test setup consisted of a load generator, two web servers and a service broker implemented using Apache Synapse. Using this setup, we compared the performance of SALSA to WRR, with different arrival rate patterns. Some results are shown in Table 1, where λ_d and λ_p represent default resp. premium arrival rates. It can be seen from this table that WRR slightly outperforms SALSA in underloaded circumstances. This is normal, since SALSA requires more processing, and has to slightly overestimate arrival rates in order to make statistical guarantees. However, in situations with higher arrival rates, SALSA is able to guarantee a 95-percentile waiting time, where WRR is not.

λ_d	λ_p	Algorithm	Crossing threshold (%)
40	20	SALSA WRR	1,66 0,71
10	40	SALSA WRR	2,28 0,59
50	50	SALSA WRR	0,47 28,70
40-80	20-40	SALSA WRR	2,16 17,15

Table 1. Comparing SALSA to WRR.

IV. Conclusions

By using the SALSA algorithm, requiring slightly more processing than weighted round-robin, brokers can guarantee a n-th percentile response time to their premium users, while providing best effort to the default customers. As service-oriented architectures have largely distributed topologies, SOA broker architectures can benefit from our SALSA algorithm as the service providers can be QoS unaware, released from mediating SLAs, and don't have to be a priori over-dimensioned.

References

[1] B. A. Shirazi, A. R. Hurson, K. M. Kavi, Eds., *Scheduling and load-Balancing in Parallel and Distributed Systems*, IEEE CS Press, 1995.

[2] D. Gross, C. Harris, *Fundamentals of Queueing Theory, 3rd ed, 1998.*

[3] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, *Optimization by Simulated Annealing*, Science, Number 4598, 220, 4598:671-680, 1983.