Modeling response times in the Google ROADEF/EURO Challenge

Paolo Cremonesi and Andrea Sansottera
Dipartimento di Elettronica e Informazione, Politecnico di Milano
Via Ponzio 34/5, 20133, Milan, Italy
paolo.cremonesi@polimi.it, sansottera@elet.polimi.it

ABSTRACT

In this paper, we extend the machine reassignment model proposed by Google for the ROADEF/EURO Challenge. The aim of the challenge is to develop algorithms for the efficient solutions of data-center consolidation problems. The problem stated in the challenge mainly focus on dependability requirements and does not take into account performance requirements (end-to-end response times). We extend the Google problem definition by modeling and constraining end-to-end response times. We provide experimental results to show the effectiveness of this extension.

1. INTRODUCTION

The aim of the Google ROADEF/EURO Challenge[1] is to improve the usage of a set of servers in a geographically distributed data-center.[2] Servers are arranged in local tightly coupled clusters (neighborhoods) and distributed over different sites (locations). A server has several resources (e.g., RAM and CPU), and runs processes which consume these resources. Processes are organized into single-tier services (e.g., mail service, data-base service) which are integrated into multi-tier applications. The allocation of processes to servers is constraint by three types of requirement: (i) *capacity constraints*: resource utilization cannot exceed maximum server capacity; (ii) *dependency constraints*: closed-coupled services must have processes running on the same server or in a cluster of tightly connected servers; (iii) *availability constraints*: conflicting or mutually fail–over processes cannot run on the same server; and (iv) *continuity constraints*: processes of mission critical services must be spread across different sites. The objective function is composed of multiple terms: (i) load costs, (ii) balance costs and (iii) movement costs. The problem described in the Google challenge has been partially addressed by other works in th past. Many works [1] [2] [3] do not consider response times and dependability constraints. Some works use optimization algorithms at run-time to optimize the actions taken by adaptive systems [4] [5]. Other works [6] [7] introduce response time constraints, assuming single tier-applications.

The problem described in the Google challenge has two limitations: (i) the problem does not take into account end-to-end response times, and (ii) when moving a process between servers, its resources consumption does not change. The implications of the first point are easy to understand: a solution to the optimization problem could lead to unacceptable increases in response times. Even if load costs penalizes high utilization of resources and hence high response times, response times are neither modeled nor constrained explicitly. The second point assumes that the number of visits to a process does not depend on the process allocation. This assumption is in conflict with the "dependency constraints", which bind the routing of request between closed–coupled services to processes running on the same cluster (i.e., neighborhood) of servers. This is better explain with Figure 1. Front-end service s_1 (composed of three load-balanced processes $p_1 \dots p_3$) processes and dispatches class 1 requests to back-end service s_2 (composed of two load-balanced processes p_4 and p_5). Because of a dependability constraint on class 1 requests, all these processes are required to run on the same cluster of servers (*neighborhood 1*). The two back-end processes manage also class 2 requests, which are not binded by any dependability constraints. If we move process p_4 to a different cluster of servers (*neighborhood 2*), that process cannot handle class 1 request any more, because of the dependability constraints. Therefore, all class 1 requests will be handled by back-end process p_5. This change of allocation has the effect of changing the routing of requests between processes and, in turns, their loads.

In this paper, we extend the machine reassignment model proposed by Google trying to address and solve the two above issues. To this purpose, we modify the definition of services by introducing routing probabilities and number of visits, and we develop a parametric queuing network model for the service response time estimation. Model parameters allow for the evaluation of response times as a function of the mapping between processes and machines.

2. PERFORMANCE MODEL

Let $\mathcal{S}$ be the set of services, $\mathcal{P}$ the set of processes, $\mathcal{M}$ the set of machines, $\mathcal{R}$ the set of resources, $\mathcal{N}$ the set of neighborhoods. We partition the set of resources $\mathcal{R}$ into two subsets, $\mathcal{R}^{\mathrm{S}}$ and $\mathcal{R}^{\mathrm{T}}$. Examples of resources in $\mathcal{R}^{\mathrm{T}}$ are processors. Examples of resources in $\mathcal{R}^{\mathrm{S}}$ are main memory and storage space. We model each pair $(m, r) \in \mathcal{M} \times \mathcal{R}^{\mathrm{T}}$ as a queueing station. Each service $s \in \mathcal{S}$ represents the entry point of a different workload class, which needs to be

[1] `http://challenge.roadef.org/2012/files/problem_definition_v1.pdf`

[2] The challenge is still in progress. Only 30 out of 81 teams successfully passed the first qualifications. The Politecnico di Milano team currently holds the 16-th place in the leaderboard, ranking 3-rd in its category (junior open–source).

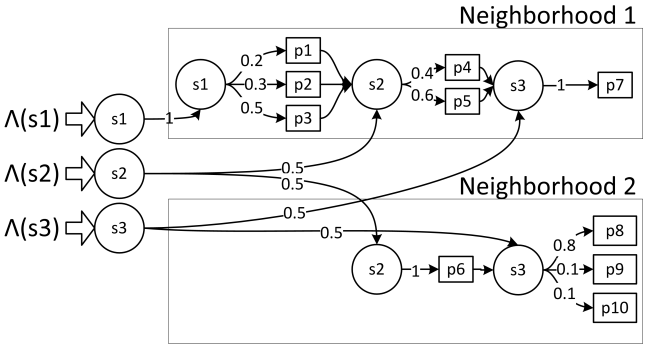

Figure 1: Example of possible routing probabilities with 3 services, 10 processes and 2 neighborhoods. Service s_1 depends on service s_2, which depends on service s_3.

processed by all the services on which s depends on, directly or indirectly. Any process p that belongs to s is functionally equivalent (this is implied by the dependency constraints). However, as in Google's proposal, processes of the same service might have different service times, even if running on the same server. Let $(\mathcal{S}, \mathcal{D})$ be the directed graph representing the dependency relation $\mathcal{D}$ on the set of services $\mathcal{S}$. Moreover, let $\mathcal{D}^+$ denote the transitive closure of $\mathcal{D}$. A job submitted to service $s \in \mathcal{S}$ visits all services $j \in \mathcal{S}$ such that $(s, j) \in \mathcal{D}^+$. We denote this set of services $\mathcal{D}_s$. We require that a job is serviced by processes assigned to machines that reside in the same neighborhood, since they have high-speed interconnections and shared storage.

We consider only transactional workloads. The arrival rate of class s workload is denoted as Λ_s. We denote the average number of visits performed by a job of class s to a service $j \in \mathcal{D}_s$ as v_{sj}. Once a job of class s enters the system, a neighborhood n must be selected. We assume the neighborhood choice is performed by an external load balancer and we denote the probability that a job of class s is routed to neighborhood n as $\alpha_{sn} > 0$.

Each tier (i.e., service) contains one or more processes running in parallel in the same neighborhood. For each tier, requests are dispatched to process p with probability $\beta_p > 0$ by means of a local load balancer.

We represent the assignment of processes to machines with a set of binary variables x_{pm}, which are equal to one if and only if process p is assigned to machine m. Moreover, the continuous variables $\alpha_{sn} \in [0, 1]$ and $\beta_p \in [0, 1]$ represent the routing probabilities defined above. We now describe how to compute the end-to-end response time τ_s for class s workload, i.e., the workload entering the system at tier s. The arrival rate at process p, which belongs to service s, is:

$$\lambda_p = \beta_p \sum_{n \in \mathcal{N}} \left(\sum_{m \in n} x_{pm} \right) \left(\Lambda_s \alpha_{sn} + \sum_{(s,j) \in \mathcal{D}^+} \Lambda_j \alpha_{jn} v_{js} \right) . \quad (1)$$

Let μ_{pr} be the service rate of process p at the resource $r \in \mathcal{R}^{\mathrm{T}}$ of a reference machine. Moreover, let f_{mr} be the speedup factor of the resource r of machine m with respect to the reference machine. According to the utilization law, the utilization of the resource r of machine m is

$$u_{mr} = \sum_{p \in \mathcal{P}} x_{pm} \frac{\lambda_p}{\mu_{pr} f_{mr}} , \quad (2)$$

where the binary variable x_{pm} is used to account only for the processes that are assigned to machine m. We model the resources in $\mathcal{R}^{\mathrm{T}}$ as M/M/1 queues and compute the expected response time of a process p executed at machine m as follows:

$$t_{pm} = \sum_{r \in \mathcal{R}^{\mathrm{T}}} \frac{1}{(1 - u_{mr})(\mu_{pr} f_{mr})} . \quad (3)$$

The expected response time of a request for service s routed to neighborhood n is computed considering the routing probabilities β_p:

$$T_{sn} = \sum_{p \in s} \beta_p \sum_{m \in n} x_{pm} t_{pm}, \quad (4)$$

where the binary variable x_{pm} selects only the processes $p \in s$ assigned to a machine $m \in n$. The end-to-end response time of service s is determined by the routing probabilities α_{sn} and the visit counts v_{sj} for all services j which service s depends on:

$$\tau_s = \sum_{n \in \mathcal{N}} \alpha_{sn} \left(T_{sn} + \sum_{j \in \mathcal{S}:(s,j) \in \mathcal{D}^+} v_{sj} T_{jn} \right) . \quad (5)$$

An example with 3 services, 10 processes and 2 neighborhoods is shown in Figure 1. The dependency relation is $\mathcal{D} = \{(s_1, s_2), (s_2, s_3)\}$. The expected response times for service s_2 are $T_{s_2,n_1} = 0.4 t_{p_4,m_1} + 0.6 t_{p_5,m_7}$ and $T_{s_2,n_2} = t_{p_6,m_8}$, where m_1, m_7 and m_8 are the machines to which p_4, p_5 and p_6 are assigned, respectively. The end-to-end response time is $\tau_{s_2} = 0.5(T_{s_2,n_1} + v_{s_2,s_3} T_{s_3,n_1}) + 0.5(T_{s_2,n_2} + v_{s_2,s_3} T_{s_3,n_2})$. The objective function is the same of the Google's ROADEF/Euro Challenge. Moreover, we consider spread, conflict, dependency, capacity and transient capacity constraints, which we do not report for the sake of brevity. Of course, a set of constraints ensures that each process is assigned to one and only one machine:

$$\sum_{m \in \mathcal{M}} x_{pm} = 1 . \quad (6)$$

We impose constraints on the end-to-end response times $\tau_s \leq \tau_s^*$, where τ_s^* is a user defined threshold for service s. Finally, we need some constraints for the routing probabilities:

$$\sum_{n \in \mathcal{N}} \alpha_{sn} = 1 \quad (7)$$

$$\alpha_{sn} \leq \sum_{p \in s} \sum_{m \in n} x_{pm} \quad (8)$$

$$\sum_{p \in s} \beta_p \sum_{m \in n} x_{pm} = \min \left(1, \sum_{p \in s} \sum_{m \in n} x_{pm} \right) \quad (9)$$

Constraint (7) is a normalization condition. Constraint (8) ensures that $\alpha_{sn} = 0$ if there are no processes of s in neighborhood n. According to constraint (9), the routing probabilities for the processes of s located in n must sum to 1 if there is at least one process and to 0 otherwise. Some of the parameters in our model are not available in the data sets provided by Google. In particular, the external arrival rates

Instance	Initial	Best Challenge	Our Best	Initial resp. time			Final resp. time		
				min	avg	max	min	avg	max
A1-1	49,528,750	44,306,501	46,837,191	6e-4	1.81	26.14	4e-4	1.59	23.49
A1-2	1,061,649,570	777,532,896	963,060,081	2e-3	12.14	1879.79	2e-3	9.47	1051.33
A1-3	583,662,270	583,005,717	543,401,234	4e-3	19.20	1079.41	3e-3	18.33	1235.23
A1-4	632,499,600	252,728,589	378,911,830	3e-4	4.81	204.38	3e-4	3.75	176.73
A1-5	782,189,690	727,578,309	734,170,638	4e-4	1.23	287.74	4e-4	1.12	241.36
A2-1	391,189,190	198	243,956,086	1e-3	6.31	359.99	6e-4	4.83	282.23
A2-2	1,876,768,120	816,523,983	1,421,701,784	0.03	184.68	20998.3	0.02	147.47	16611.2
A2-3	2,272,487,840	1,306,868,761	1,804,099,143	0.11	99.74	1988.74	0.12	74.13	1308.66
A2-4	3,223,516,130	1,681,353,943	2,277,732,762	0.02	65.84	2446.17	0.01	57.63	2400.46
A2-5	787,355,300	336,170,182	473,741,548	0.04	130.34	12132.8	0.04	116.91	10429.5

Table 1: Objectives obtained by our heuristic compared with the initial objectives and the best know objectives for Google's Challenge model. Statistics on the end-to-end response times are also reported.

Λ_s, the service rates μ_{pr}, the speedup factors f_{mr}, the visit counts v_{sj} and the response time thresholds τ_s^* are unknown. We generate the external arrival rates and the visit counts according to the exponential distribution. Moreover, the speedup factor can be determined as the ratio of the capacities, i.e. $f_{mr} = C_{mr}/C_{\tilde{m}r}$, where $\tilde{m}$ is an arbitrarily chosen reference machine. Given these parameters, if we fix the routing probabilities of the initial solution, the service rates μ_{pr} are uniquely determined by (1) and (2). Hence, we randomly generate α_{sn} for the initial solution and determine the initial β_p making the assumption that the response times of each service are balanced within neighborhoods. Determining the initial β_p requires the solution of linear systems. The end-to-end response time thresholds τ_s^* are set 20% higher than the end-to-end response times of the initial solution.

3. RESULTS

We solved problems based on the first ten instances provided by Google for the ROADEF/Euro challenge. These instances have from 4 to 100 machines, 100 to 1000 processes, 79 to 1000 services and up 12 different resources. To solve the proposed optimization model, we implemented a Variable Neighborhood Search heuristic. Our heuristic was run with a time limit of 15 minutes on a computer with a Core i7 640M processor. Results are reported in Table 1. For each instance, we detail the initial objective value, the best known objective value for the original Google's model and the best objective value found for the model proposed in this paper. The table also reports the minimum, average and maximum response times for the initial configuration (i.e., the initial mapping between processes and servers) and for the best configuration found from the consolidation model proposed in this paper.

3.1 Discussion

Before discussing the results, we should remember that our machine reassignment model differs from the Google model for two aspects: (i) our model allows for the rerouting (i.e., balancing) of requests between processes handling the same workload; and (ii) our model requires the end-to-end response time to increase of no more than 20% after relocation. We can generally observe that our model is able to significantly reduce the objective cost function for all of the instances, without affecting response times. The only exceptions to this behavior are for instances A1-3 and A2-1. With instance A1-3, we observe (i) a cost function lower than the best optimum known for the Google model, and (ii) an increment of the maximum response time. Both aspects are due to the rerouting of requests, that allows for the creation of "thin" processes (e.g., processes with little load) which can be fine-allocated in order to better saturate servers, with the risk of larger response times and the benefit of a better allocation of processes. In instance A2-1, the result found for our model has a much higher cost than the best solution for the Google's model. This result is not surprising since the presence of response time constraints severely limits the space of feasible solutions.

4. REFERENCES

[1] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Trans. on Services Computing*, vol. 3, no. 4, pp. 266–278, 2010.

[2] F. Hermenier, S. Demassey, and X. Lorca, "Bin repacking scheduling in virtualized datacenters," in *Proc. of the 17th int. conf. on Principles and practice of constraint programming*, ser. CP'11, 2011, pp. 27–41.

[3] J. Anselmi, E. Amaldi, and P. Cremonesi, "Service consolidation with end-to-end response time constraints," in *Software Engineering and Advanced Applications, 2008. SEAA'08. 34th Euromicro Conference.* IEEE, 2008, pp. 345–352.

[4] T. Nowicki, M. Squillante, and C. Wu, "Fundamentals of dynamic decentralized optimization in autonomic computing systems," *Self-star Properties in Complex Information Systems*, pp. 366–366, 2005.

[5] J. Rolia, A. Andrzejak, and M. F. Arlitt, "Automating enterprise application placement in resource utilities," in *DSOM*, ser. LNCS, M. Brunner and A. Keller, Eds., vol. 2867. Springer, 2003, pp. 118–129.

[6] J. Anselmi, P. Cremonesi, and E. Amaldi, "On the consolidation of data-centers with performance constraints," *Architectures for Adaptive Software Systems*, pp. 163–176, 2009.

[7] K. Dhyani, S. Gualandi, and P. Cremonesi, "A constraint programming approach for the service consolidation problem," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 97–101, 2010.