

ANTONIO CARLOS SALZVEDEL FURTADO JUNIOR

AUTOMATIC ELASTICITY IN CLOUD COMPUTING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Almeida

CURITIBA

2011

ANTONIO CARLOS SALZVEDEL FURTADO JUNIOR

AUTOMATIC ELASTICITY IN CLOUD COMPUTING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Almeida

CURITIBA

2011

CONTENTS

ABSTRACT	ii
1 INTRODUCTION	1
2 VIRTUALIZATION DESIGN ADVISOR	4
2.1 Problem definition	5
2.2 Architecture	6
2.2.1 Cost estimator	7
2.3 Online refinement	9
BIBLIOGRAFIA	10

ABSTRACT

The concept of cloud computing has been gaining a lot of attention lately, both in the business world and in research papers. Its popularity resides at the low costs it propitiates, opposed to traditional ways of providing computational resources. The main advantage is its elasticity, in which they can be offered on-demand. On the other hand, new challenges arise because of this dynamism. A system needs to guarantee a proper scheduling of these resources, in order to fulfil an organization's demands. At the same time, it needs to meet its scheduling policies (minimize costs, heigh availability, and so on) and deal with peaks of demand.

The paper is dedicated to the study of an efficient algorithm.

CHAPTER 1

INTRODUCTION

Cloud computing has a big potential to change the Information Technology (IT) world. It was popularized as a business model by Amazon's Elastic Compute Cloud, which started selling virtual machines (VMs) in 2006. Over time, more cloud providers have appeared in the market, offering their computational resources (CPU, memory, and I/O bandwidth). This type of cloud, in which the IT infrastructure is deployed through virtual machines, is referred as Infrastructure-as-a-service (IaaS). One of the most appealing benefits of this paradigm, when associated to cloud providers, is the ability to cut costs. Companies may base their IT strategies on cloud-based resources, spending very little or no money managing their own IT infrastructure. They pay for these resources on-demand, in contrast to the traditional resource provisioning model, in which they would need to deal with both under- and over- utilization of their own resources. Also, the cloud providers may offer lower prices because they are benefited with the economy of scale.

The benefits mentioned earlier are only noticeable when using public clouds – clouds made available in a pay-as-you-go manner to the public by a cloud provider. Although the market has evolved around this type of cloud, organizations might build IaaS clouds using their own infrastructure, known as private clouds. Their aim is not sell capacity over the internet, but give local users an agile and flexible infrastructure to run service workloads in their administrative domains. These users are offered VMs, which are scheduled among a group of physical machines from their organization, which we call a cluster. This leads to a better utilization of resources, since services with little demands can be packed into the same machine, process known as server consolidation. Other benefits, such as the migration of VMs between hosts and the ability to dynamically change the amount of resources provided to it, which are enabled by the technology present in virtual machine monitors (VMMs), make it possible to deal with fluctuations in the workload. This

provides an elastic environment, which is good for a private cloud, and vital for the pay-on-demand model used in public clouds. It's also possible for an organization to create a hybrid cloud, which are useful to supplement a private cloud's infrastructure with external resources from a public one.

The machine virtualization proposed by the cloud concept is essential to achieve its benefits. Database management systems (DBMS), like other software systems, are also increasingly being run on virtualized environments with the same goal. In [1], it is discussed the virtualization design problem for relational database workloads, which can be defined as follows: *"Given N database workloads that will run on N database systems inside virtual machines, how should we allocate the available resources to the N virtual machines to get the best overall performance?"*. According to it, the problem of virtualization design may find a better solution when applied to relational database systems due to three factors. First, relational database workloads consist of SQL queries with constrained and highly specialized resource usage patterns. Second, queries are highly variable in the way they use resources – one query might heavily need CPU, while another needs I/O bandwidth instead. Thus, they could benefit from elasticity. Third, database systems already have a way of modeling their own performance, namely the query optimizer.

As mentioned, DBMSs have particularities involving their workloads. Therefore, the application running inside a VM shouldn't be treated as a black box. Instead, it should exploit the database system cost model. As VMMs have parameters to control the share of physical resources, database systems have tuning parameters to manage their own performance. These two sets need to be simultaneously analyzed and tuned. In [2], it is proposed a *virtualization design advisor*. It works by setting the configuration parameters of a VM containing a DBMS. These parameters determine how the shared resources will be allocated to each DBMS instance. It uses information about their anticipated workloads to specify the parameters offline. Furthermore, runtime information collected after the deployment of the recommended configuration can be used to refine this recommendation and to handle fluctuations in the workload. It was not proposed to run this advisor in a cloud, rather it was implemented in a single physical machine, in which two DBMS

instances were deployed.

Our study proposes to implement this virtualization design advisor in a cloud environment. The advisor should be able to configure all the VMs deployed in a cluster, considering the resources of all physical machines. It is expected that this advisor provides automatic elasticity for the cloud. The rest of this paper is structured as follows. In Section 2, the *virtualization design advisor* is described. Section 3 is used to show how a cloud infrastructure is managed. In section 4, we propose an integration of the advisor within the cloud management system.

CHAPTER 2

VIRTUALIZATION DESIGN ADVISOR

In [2], it is considered a typical scenario of resource consolidation, in which several DMBS instances, each one of them running in a separate VM, share a common pool of physical resources. The mentioned paper addresses the problem of optimizing the performance of these instances by controlling the configurations of the VMs in which they run. These configurations determine how these resources will be allocated to each DMBS instance. It's also considered that the physical resources belong to one server, in which all the VMs run. The scenario is illustrated in 2.1.

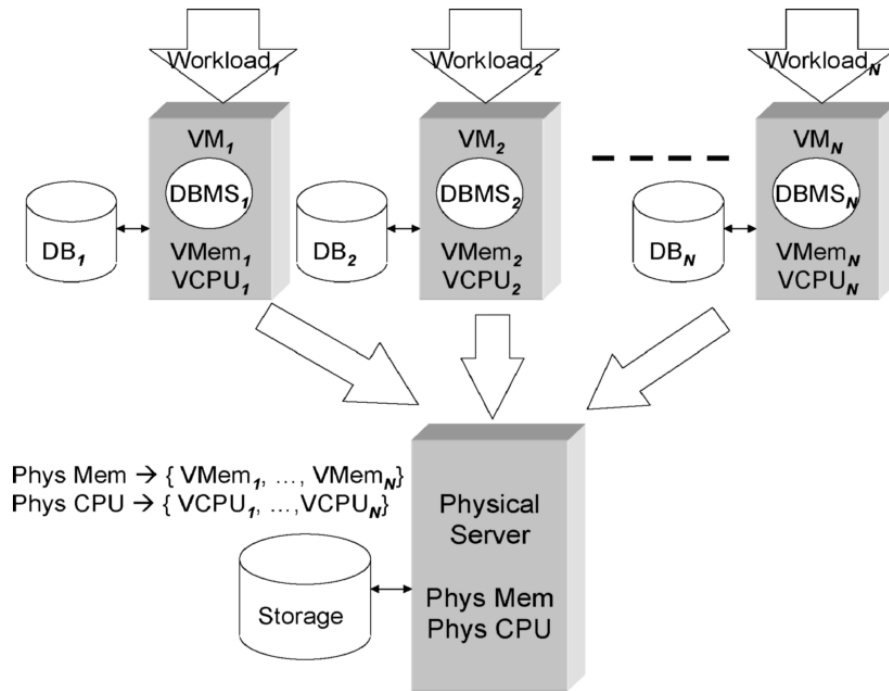


Figure 2.1: Resource Consolidation scenario

2.1 Problem definition

In order to model this problem, the author assumes that there are M types of resources, such as CPU capacity, memory, or I/O bandwidth. The notation used to represent the share of resources allocated to a VM running a workload W_i is $R_i = [r_{i1}, \dots, r_{iM}]$, $0 \leq r_{ij} \leq 1$. This workload represents the statements processed by the different DMBS instances at the same amount of time.

Each workload has a cost, which depends on the resource share allocated to the VM in which it runs. It is used $Cost(W_i, R_i)$ to represent the cost of running the workload W_i under resource allocation R_i . Considering that there are N workloads, the goal is to choose r_{ij} , $1 \leq i \leq N$, $1 \leq j \leq M$ such that

$$\sum_{i=1}^N Cost(W_i, R_i)$$

is minimized.

This problem is generalized to satisfy Quality of Service (QoS) requirements. One of these requirements is to specify the maximum increase in cost that is allowed for a workload under the recommended resource allocation. It was defined a *cost degradation* as

$$Degradation(W_i, R_i) = \frac{Cost(W_i, R_i)}{Cost(W_i, [1, \dots, 1])}$$

, where $[1, \dots, 1]$ represents the resource allocation in which all the available resources are allocated to W_i . It can be specified a *degradation limit* L_i ($L_i \geq 1$), such that

$$Degradation(W_i, R_i) \leq L_i$$

for all i . This limit is set per workload, so it does not need information about other workloads that it will be sharing the physical server with.

The other QoS requirement introduced is the ability to specify relative priorities among the different workloads. A *benefit gain factor* G_i ($G_i \geq 1$) can be used to indicate how important is to improve the performance of W_i . Each unit of improvement is considered

to worth G_i cost units. When this parameter is applied to the problem, it may cause a workload to get more resources than its fair share. In order to incorporate it to our problem, the cost equation is modified to minimize the following

$$\sum_{i=1}^N G_i * Cost(W_i, R_i)$$

2.2 Architecture

The process of determining the allocation of resources to each VM is not immediate, nor static. The proposed advisor follows a sequence of steps. Initially, it makes resource allocations based on the workload descriptions and performance goals, which is performed offline, i.e., the VMs are not running yet. Then two following steps are performed online. First, it adjusts its recommendations based on workload costs to correct for any cost estimation errors made during the initial phase. Second, it uses continuing monitoring information to dynamically detect changes in the workloads. This last step is important because a workload cannot be considered static, its resource needs may change during execution. This approach prevents the advisor from allocating resources to DBMS instances that will obtain little benefit from them.

Since the advisor has a considerable number of sub-processes, it makes sense to organize it in modules. An overview of this advisor in a modular way is given in 2.2. This paper intends to give a brief explanation of each module.

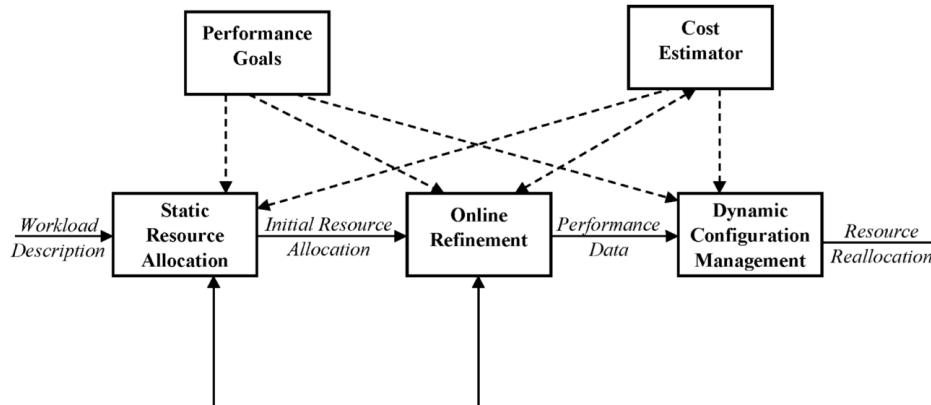


Figure 2.2: Advisor overview

2.2.1 Cost estimator

Given a workload W_i , the cost estimator will estimate $Cost(W_i, R_i)$. The strategy used to implement this module is to leverage the cost models built into database systems for query optimization. The query optimizer cost model can be described as $Cost_{DB}(W_i, P_i, D_i)$, where W_i is a SQL workload, $P_i = [p_{i1}, \dots, p_{iL}]$ is a vector of parameters that are used to list both the available computing resources and DBMS configuration parameters relevant to the cost model, and D_i is the database instance.

The author identifies two problems in using only query optimizer cost models. The first problem is the difficulty of comparing cost estimates produced by different DBMSes. They may have different cost models. Even if they share the same notion of cost, they normalize costs differently. This problem is addressed partially by the advisor. Although it only considers that the DBMSes have the same notion of cost, it proposes a renormalization step, in order to make $Cost_{DB}(W_i, P_i, D_i)$ from different DBMSes comparable. For our purpose this step is not considered for implementation, since the support of multiple DBMSes is out of scope of this paper.

The second problem is that the query optimizer cost estimates depends on P_i , while the virtualization design advisor is given a candidate resource allocation R_i . The mapping of these two parameters is done through a calibration step. This step determines a set of DBMS cost model configuration parameters according to the different possible candidate resource allocations. It is supposed to be performed per DBMS on the physical machine before running the virtualization design advisor. Once the appropriate configuration parameters P_i are determined for every possible R_i , the DBMS cost model is used to generate $Cost_{DB}$.

The calibration step is performed on *descriptive parameters*, which are used to characterize the execution environment. The approach to the *prescriptive parameters*, which control the configuration of the DBMS itself, is to leave it for user definition. For instance, the 2.1 show some descriptive parameters used in PostgreSQL, while 2.2 show the prescriptive ones.

The calibration follows the basic methodology for each parameter $p_{ij} \in P_i$:

Parameter	Description
random_page_cost	Cost of non-sequential page I/O
cpu_tuple_cost	CPU cost of processing one tuple
effective_page_size	size of file system's page size

Table 2.1: Descriptive parameters

Parameter	Description
shared_buffers	shared bufferpool size
work_mem	amount of memory used by each sort and hashing operator.

Table 2.2: Prescriptive parameters

- (1) Define a calibration query Q and a calibration database D , such that $Cost_{DB}(Q, P_i, D)$ is independent for all descriptive parameters in P_i , except for p_{ij} ;
- (2) Choose a resource allocation R_i , instantiate D , and run Q under that resource allocation, and measure the execution time T_Q ;
- (3) This step refers to the renormalization of the $Cost_{DB}$ provided by the DBMS. As mentioned earlier in this section, we are not going into the details of this step;
- (4) Repeat the two preceding steps for a variety of R_i , associating with each parameter p_{ij} that describes that resource. For instance, query optimizer parameters that describe CPU, I/O and memory are independent of each other and can be calibrated independently;
- (5) Perform regression analysis on the set of (R_i, p_{ik}) value pairs to determine a calibration function Cal_{ij} that maps resource allocations to p_{ik} values.

During the described methodology, calibration queries should be carefully chosen. They need to be dependent only on the parameter that is being calibrated. If it is not possible to isolate one parameter, a system of k equations is solved to determine the values for the k parameters.

2.3 Online refinement

The initial allocation is based on the calibrated query optimizer cost model, as described earlier. This enables the advisor to make recommendations based on an informed cost model. However, this model may have inaccuracies that lead to sub-optimal recommendations. The *online refinement* is based on the observation the actual times of different workloads in the different virtual machines. It uses these observations to refine resource allocation recommendations. Then the advisor is rerun with the new cost models, so we can obtain an improved resource allocation for different workloads. These optimizations are performed until the allocations stabilize. It's important to notice that the goal of the *online refinement* is not deal with dynamic changes in the workload, which are dealt by another module, but to correct cost models. Thus, it's assumed that the workload is not going to change during this process.

BIBLIOGRAFIA

- [1] A.A. Soror, A. Abounnaga, and K. Salem. Database virtualization: A new frontier for database tuning and physical design. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 388 –394, april 2007.
- [2] Ahmed A. Soror, Umar Farooq Minhas, Ashraf Abounnaga, Kenneth Salem, Peter Kokosielis, and Sunil Kamath. Automatic virtual machine configuration for database workloads. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 953–966, New York, NY, USA, 2008. ACM.

ANTONIO CARLOS SALZVEDEL FURTADO JUNIOR

AUTOMATIC ELASTICITY IN CLOUD COMPUTING

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Departamento de Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Eduardo Almeida

CURITIBA

2011