Designing and testing VM allocation algorithms for the CIRCLE Cloud manager

Ádám Belákovics, Arnold Czémán, Imre Szeberényi

Abstract: CIRCLE is an IaaS based cloud manager [1] [2]. It provides an ideal way to effectively use virtualized systems in educational environments. BME (Budapest University of Technology and Economics) has been increasingly utilizing a CIRCLE-based cloud management system for years. However, increasing utilization imposes serious requirements on infrastructure and hence for system administrators. As the number of users increases, lack of service has a major impact on the efficiency of everyday university work. It is indispensable to define and observe the SLAs that ensure the long-term utilization and quality of the system. One such problematic part, regarding the quality of the system's service, is the distribution of physical resources among virtual machines. The problem of assigning virtual machines to physical machines is the main topic of this publication.

Keywords: CIRCLE SLA VM allocation policy CloudSim

Introduction

The CIRCLE cloud system operates with a finite set of servers. Therefore the proper allocation of resources is a serious issue. The system has to fulfil different requirements (lecture, workstation, webserver). To solve this problem, the measurement of the results on different metrics and the accurate knowledge of user needs are essential. Improving the scheduling can be achieved by the creation of an algorithm that enables better results in the Virtual Machine (VM) allocation (Selecting an available Physical Machine (PM) for the VM).

The current scheduler

Allocation of the VMs is performed by the scheduler using a simple priority-based algorithm. The PM with the lowest amount of CPU usage is selected.

- The amount of RAM that is allocated on the machines does not play a role in the selection process. This creates uneven RAM utilization among the PMs. In case of heavy loads the lack of this resource has repeatedly posed a problem in the system. The monitoring system enables queries on historical data. While analyzing the results of the queries, minimal deviation can be observed among the CPU utilization, the opposite is experienced in the case of RAM. Consequently, the algorithm performs well for the CPU utilization but neglects available RAM.

- Bulk launching: Another problem is that if many virtual machines are launched simultaneously, the scheduler assigns too many new VMs to a given PM. The reason for this is that the algorithm makes the decision based on the state of the monitor. However, this information may be out of date because the monitoring system's data is updated in every minute. Solving the problem of bulk launch is especially important because this is one of the most important use case of the system. The lab lessons, which are hosted in CIRCLE, follow this behavior.

Setting up the testing environment

Metrics

The aim of scheduling (in our system) is to provide an equal load on each physical machine. If the resources are allocated equally, the system is able to fulfill the SLA requirements (if the required amount of computing power and RAM does not exceed the available.) Therefore the main metric that we use is the mathematical deviation of the resources between the PMs. The goal of our algorithms is to minimize this value. For resource r the deviation function is the following:

$$D_n^2 = \frac{(p_1^r - \overline{p^r})^2 + (p_2^r - \overline{p^r})^2 + \cdots + (p_n^r - \overline{p^r})^2}{n}$$

where,
p_i^r is the utilization of resource r on the PM identified by i,
$\overline{p^r}$ is the mean value of utilization for a given resource,
n: number of PMs.

In all further reference D^2 will be used marked with d.

Basic algorithms

- **A**(random resource allocation): The PM selected for the new VM is random.
- **B**(CPU based load balancing): The new VM is placed on the PM with the lowest CPU utilization.
- **C**(CPU-RAM based load balancing): The new VM is placed on the PM with the lowest CPU and RAM utilization.
  - **C1**: The decision is based on the following value: $CPUfree(\%) + RAMfree(\%)$.
  - **C2**: The decision is based on the following value: $CPUfree(\%) * RAMfree(\%)$.

The simulation

For the simulation we used a simple python simulator program as we mentioned above, which creates PMs with the specified parameters, then it allocates VMs to these by the specified algorithm. For the reason of its simplicity, its usage is limited to point out the differences between the proposed algorithms. The input is a data set, which resembles our infrastructure. In this infrastructure the resources are heterogeneous. The output is the deviation and other data for the visualization.

Results

The python code generates graphs about the scheduling. This proved useful for understanding the test results. The deviation results are based on an average of several test runs. In all test runs the deviation is calculated after the 100th VM is scheduled.

- **A**(random resource allocation): Average deviation: $d(CPU) = 0.0362, d(RAM) = 0.0597$
- **B**(CPU based load balancing): Average deviation: $d(CPU) = 0.0010, d(RAM) = 0.0490$ For the CPU the deviation is lower by an order of magnitude.
- **C**(CPU-RAM based load balancing): Average deviation for C1: $d(CPU) = 0.0127, d(RAM) = 0.0124$ for C2: $d(CPU) = 0.0132, d(RAM) = 0.0087$

Summary

	A	B	C_1	C_2
d(CPU)	0.0362	0.0010	0.0127	0.0132
d(RAM)	0.0597	0.0490	0.0124	0.0087

- The results for algorithm **B** (which is a current scheduler) indicates the issue introduced above. The RAM deviation is high, nearly as high as for algorithm **A**.

- **C1** and **C2** algorithms perform relatively well with both metrics, by examining the data another problem arises. The average deviation of the PMs allocated resources is constantly increasing. It doesn't differentiate between the values of separate resource usage. Based on algorithm C_1 a VM with a CPU usage of 80% and RAM usage of 20% gets the same rank as one with both values at 50%. This unfortunately makes this algorithm unsuitable for long-term scheduling.

The new algorithm

From the results we can conclude that it is necessary to have an algorithm that makes its decision by two variables but does not allow the long-term divergence seen above.

Prioritized load balancing

C3 (CPU-RAM based prioritized load balancing): The selected machine will be the one with the highest $MIN(CPUfree(\%), RAMfree(\%))$ value. This algorithm will be able to solve the divergence problem. It selects the critical resource and makes its decision by the utilization value.

Correction for heterogeneous systems

If the above algorithm is used in a heterogeneous infrastructure (with varying hardware capabilities), it can make the CPU and RAM utilization rates meaningless. Therefore a correction value is needed. This will be the {CPU, RAM}_weight.

$$PM^{i}_{cpu\_weight} = \frac{PM^{i}_{cpu\_cores}}{\sum_{j=1}^{N} PM^{j}_{cpu\_cores}},$$

where:
$PM^{i}_{cpu\_weight}$: cpu_weight of the PM identified by i,
$PM^{i}_{cpu\_cores}$: number of processing cores of the PM identified by i.

For RAM the same correction value can be calculated.

Prioritized load balancing with weights

D(CPU-RAM based prioritized load balancing with weights): The machine with the highest $MIN(CPUfree(\%) * cpu\_weight, RAMfree(\%) * ram\_weight)$ value will be selected.

Tests

The algorithm D was tested using the previously mentioned python program. The following results can be observed: $d(CPU) = 0.0272$, $d(RAM) = 0.0092$. Based on the deviation values, the algorithm does not performs exceptionally well, the deviation for CPU is of the same magnitude as the algorithm **B** for RAM. On the other hand $d(RAM)$ is much lower, which is a promising result. The reason for these results lies in the characteristics of the input data. In the current input PM data (which reflects the resources of the cloud infrastructure at the university), RAM is the scarcer resource. If we change the characteristics of the input data the algorithm will favor the resource which has the lower available amount for a PM. However this is not a problem, since the algorithm designed to focus on the critical variable. Further examination of the data also shows that the D algorithm does not produce the divergence seen at C_1 and C_2, which makes it suitable for long-term scheduling. This fulfils the requirements.

Future of the research

Although the simple test environment was suitable for the algorithms to be compared, it does not mean that the new algorithm would not have problems. The algorithm was extended to handle bulk launching as well, but our current testing environment is not able to validate its effectiveness. First we need to improve our simulation environment. CloudSim [3] is a software package developed by the University of Melbourne at CLOUDS, which allows the simulation of complex cloud infrastructures. This would allow for more accurate testing and validation of algorithms [4]. In addition, there are several publications about this topic [5] [6], and by examining them we are looking to expand our knowledge, to achieve better results solving the SLA problems of our cloud infrastructure.

Acknowledgements

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013).

References

[1] Guba Sándor (2012), Cloud rendszer fejlesztése oktatási környezethez, BME

[2] Guba Sándor (2014), Oktatási felhő kialakítása, BME

[3] Calheiros, R. N., Ranjan, R. , Beloglazov, A. , De Rose, C. A. and Buyya, R. (2011), CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract. Exper., 41: 23-50. doi:10.1002/spe.995

[4] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Generation Computer Systems, Volume 28, Issue 5, 2012, Pages 755-768, ISSN 0167-739X

[5] Zoltán Ádám Mann (2015), Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms, Journal ACM Computing Surveys (CSUR) Volume 48 Issue 1, September 2015 Article No. 11

[6] J. Hu, J. Gu, G. Sun, and T. Zhao. (2010) A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89-96, IEEE