EPJ Web of Conferences **214**, 07011 (2019) https://doi.org/10.1051/epjconf/201921407011
CHEP 2018

Advanced Scheduling in IaaS Clouds

Nikita Balashov[1,*], *Alexander* Baranov[1], *Sergey* Belov[1], *Ivan* Kadochnikov[1], *Vladimir* Korenkov[1,2], *Nikolay* Kutovskiy[1], *Andrey* Nechaevskiy[1], *Igor* Pelevanyuk[1]

[1]Joint Institute for Nuclear Research, Laboratory of Information Technologies, Dubna, Russia
[2]Plekhanov Russian University of Economics, Moscow, Russia

Abstract. The complexity of modern software libraries and applications makes it hard to predict possible workloads generated by the software that may lead to significant underutilization of hardware in Infrastructure-as-a-Service (IaaS) clouds. In this paper, we give a review of an approach aimed to deal with resource underutilization in cloud environments, including description of a developed software framework and an example algorithm. IaaS clouds following this universal approach can help increase overall cloud resource utilization independently of the variety of cloud applications.

1 Introduction

For the last decade, cloud technologies have developed significantly and continue gaining popularity in all areas that involve computations: from small to large business and academia. Of the three basic cloud models [1] – Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS) – the IaaS model is probably the most common in the High Energy Physics (HEP) domain. IaaS-based clouds serve a universal computing resource designed to accommodate lots of various independent user applications in a unified environment. The cloud of the Joint Institute for Nuclear Research (JINR) [2] is an example of such a service built specifically for a scientific community to support a wide range of activities: hosting of IT services as well as testing new systems and updates, software development and, of course, compute-intensive tasks, such as physical data analysis and modelling.

One of the main advantages of using cloud virtual machines (VMs) is that users can size them to better fit the amount of computing resources required for the task, thus reducing resource underutilization compared to traditional dedicated systems. Nevertheless, natively non-scalable applications still do underutilize allocated resources due to the dynamic nature of any computational process. As an example, Figure 1 illustrates how much JINR cloud resources are underutilized in normal running conditions by showing CPU load for one of the clusters.

In this article we will describe an approach and a software framework developed to further decrease underutilization of computing resources using such cloud features as live-migration of virtual machines and overcommitment of resources.

* Corresponding author: balashov@jinr.ru

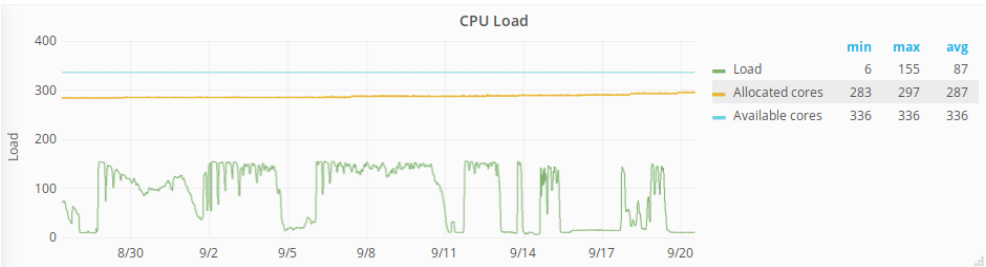

Fig. 1. CPU load in one of the JINR cloud clusters for the period of one month.

2 Dynamic relocation of resources

2.1 Overcommitment and live-migration of virtual machines

One of the most commonly used techniques to reduce underutilization of allocated resources is to "overcommit" VMs – allocate more virtual resources than physically available. This approach has two major drawbacks. First, as was mentioned before, the workloads generated by the VMs are dynamic, i.e. changing with time, therefore, applying this technique may lead to a hardware overload. This results in a number of negative effects, ranging from reduced performance of VMs to their total failure (depending on the hypervisor in use and its configuration). Second, it is possible that there would be periods when multiple VMs generate low load, still resulting in high underutilization of resources. To reduce these negative effects and further improve hardware utilization, an overcommitment technique can be combined with dynamic relocation of resources.

Live-migration technology makes it possible to transfer running virtual machines between different servers with minimal downtime and preserve all running processes' state, so that the migration process is barely noticeable and has minimal impact on VM functionality. This technique allows us to offload some virtual machines when hardware overload occurs (or prior to it when load reaches the defined thresholds). And it also can be used to dynamically track cloud workloads and relocate VMs to keep resource utilization higher even under dynamically changing workload conditions.

There are a number of different algorithms that manage dynamic workloads within cloud environments using overcommitment and live-migration technologies [3-5]. In the next section we describe a simple heuristic algorithm which we implemented based on the JINR cloud workloads profile.

2.2 Example VM relocation algorithm

Analysis of the JINR cloud workloads showed that VMs can be split into two classes: low-active and highly active ones, which we will refer to as "cold" and "hot" VMs respectively. Based on this, the algorithm takes 4 main stages:

First stage: VM classification. In the case of the JINR cloud, classification is based on VM CPU load and memory usage only, assuming that other major characteristics, such as disk and network I/O, can be safely ignored. Figure 2 shows an example of such a classification for one of the JINR cloud clusters.

Second stage: server classification. In this stage we define the number of physical resources required to accommodate the VMs of each class and assign each server the corresponding class.

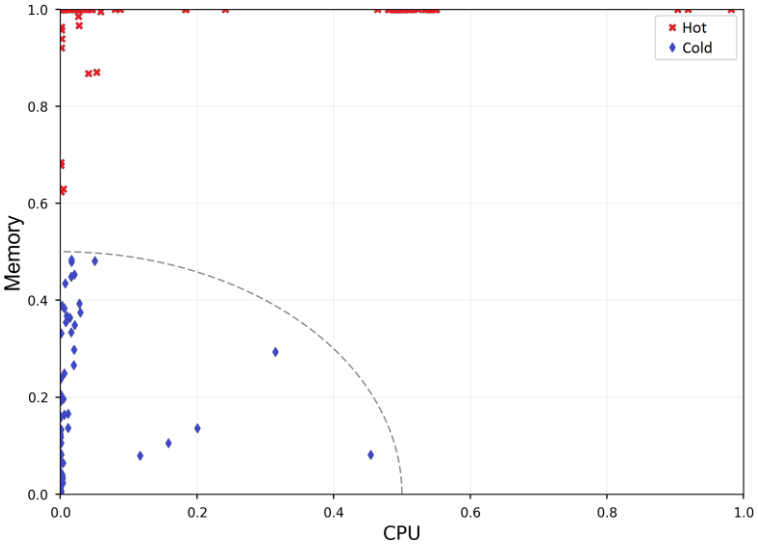

Fig. 2. Example of the VM classification in the JINR cloud. The X-axis shows the real CPU usage of the VM as a fraction of the total allocated CPU for the VM, the Y-axis shows the real memory usage by the VM as a fraction of the total allocated memory for the VM.

Third stage: build the migration map. Choosing which VMs to migrate to which servers can be treated as a bin packing problem. A modification of a Best-Fit Decreasing (BFD) algorithm may be used to handle the task [6].

Fourth stage: VM migration. The migration process itself consumes computing resources: during every migration, snapshots of memory (and in some cases the VM data disks, depending on the migration scheme) must be transferred over the network between two servers. If only a small number of VMs have to be migrated, all the migrations can be performed all at once. But when a large number of migrations are required, spreading them in time and doing a small number of simultaneous migrations becomes reasonable.

By giving "cold" servers higher overcommitment thresholds and lower (or zero) values to "hot" server thresholds, we make overcommitment more reasonable and safe. Cycling through the four main stages forms a simple heuristic algorithm, the detailed information on which can be found in one of our previous publications [7]. A schematic representation of this process is illustrated in Figure 3.

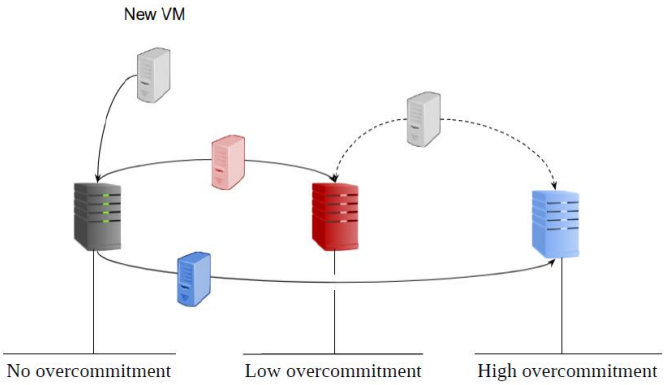

Fig. 3. Overview of the dynamic relocation algorithm.

2.3 Software framework

To implement the described algorithm, a software framework [8] was developed by the JINR cloud team. The framework acts as a meta-scheduler complementing the built-in cloud scheduler, leaving the latter the responsibility of allocating the initial VM. Figure 4 shows an overview of the framework and includes the following main components:

- A monitoring system gathers required metrics and stores them in a time-series database. It uses Icinga 2 for monitoring, InfluxDB for storing and Grafana for visualizing data.
- A custom Linux daemon that runs the scheduling algorithm.
- A library which implements basic cloud operations and interaction with the monitoring system. Currently, it implements only OpenNebula XML-RPC interface.
- An additional relational database for storing extra meta-information needed by the algorithms, such as host and VM classes.

The framework is completely built on open-source systems, so it can be modified freely to suit any needs.

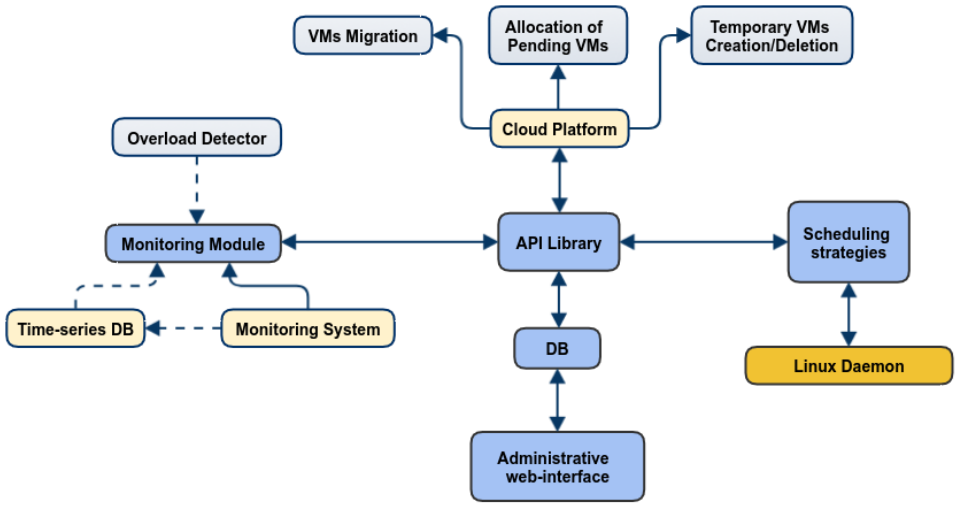

Fig. 4. Main components of the software framework.

3 Conclusions

While clouds provide a convenient environment for solving a wide variety of computational tasks, this universality may lead to lower hardware utilization when used for a large number of independent activities. In this paper we gave an example of how cloud-native features can be used to improve computational efficiency of the whole infrastructure. The same technologies can be used to tackle more “exotic” problems, like reducing local hardware overheating by dynamically redistributing workloads across a data center.

The developed system and methods are intended to smooth out the drop in efficiency and give the community a framework for further developments and improvements of cloud scheduling approaches.

The work was conducted under financial support of the RFBR project 15-29-07027

References

1. P. Mell, T. Grance, ACM, **53**, 50 (2016)
2. A.V. Baranov, N.A. Balashov, N.A. Kutovskiy, R.N. Semenov, PPNL **13**, 672 (2016)
3. A. Mosa, N. W. Paton, J. Cloud Comput., **5**. (2016)
4. A. Ashraf, I. Porres, IJPEDS, **33**, 103-120 (2017)
5. A. Beloglazov, R. Buyya, Concurr. Comput. : Pract. Exper., **24**, (2012)
6. S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. (1990)
7. N. Balashov N. et al., International scientific journal «Modern Information Technologies and IT-Education», **14,** 1 (2018)
8. SmartScheduler Framework source-code: https://github.com/jinr-lit