Письма в ЭЧАЯ. 2016. Т. 13, № 5(203). С. 957–961

OPTIMIZATION OF OVER-PROVISIONED CLOUDS

N. Balashov [a, 1], *A. Baranov* [a], *V. Korenkov* [a, b]

[a] Joint Institute for Nuclear Research, Dubna

[b] Plekhanov Russian University of Economics, Moscow

The functioning of modern applications in cloud-centers is characterized by a huge variety of computational workloads generated. This causes uneven workload distribution and as a result leads to ineffective utilization of cloud-centers' hardware. The proposed article addresses the possible ways to solve this issue and demonstrates that it is a matter of necessity to optimize cloud-centers' hardware utilization. As one of the possible ways to solve the problem of the inefficient resource utilization in heterogeneous cloud-environments an algorithm of dynamic re-allocation of virtual resources is suggested.

PACS: 89.20.Ff

Cloud technologies are rapidly gaining popularity both in commercial and scientific areas. The reason of this popularity growth is a high flexibility of the cloud architectures and that it also reduces the cost of ownership of datacenters built using cloud technologies. Spreading the cloud computing all over the world led to construction of large-scale datacenters containing thousands of computing nodes. Many large scientific organizations have already started deploying their own private cloud environments [1, 2] and transfer to them their information services and computing.

The Joint Institute for Nuclear Research (JINR) also has a cloud infrastructure deployed at the Laboratory of Information Technologies (LIT) [3]. It is based on the Infrastructure as a Service (IaaS) model — one of three basic cloud service models — and built on the open-source cloud-platform OpenNebula. The end-product of the IaaS-based cloud-service is a Virtual Machine (VM). Basically, the JINR cloud is a cluster of physical servers (also known as hosts), holding user's virtual machines. When a user requests a VM, the cloud scheduler picks a server that fits the VM's requirements, deploys this virtual machine and gives the user access to it.

These VMs are used for a variety of different activities, such as development or administration of information services, and they are also used to perform scientific computations (mostly physical analysis and modeling). All of these activities generate very different workloads. The table and Figs. 1 and 2 illustrate the JINR cloud resources utilization.

When user creates a virtual machine, it is really hard to define the precise amount of resources he would need and in most cases it is an issue hard to resolve and it is just

[1]E-mail: balashov@jinr.ru

JINR cloud utilization

Host	Hypervisor	Allocated CPU, cores × 100	Total CPU, cores × 100	Real CPU, cores × 100	Allocated memory, GB	Total memory, GB	Real memory, GB	Running VMs
1	KVM	300	400	181	6	7.7	4.1	2
2	KVM	300	400	0	3	3.7	2.2	2
3	KVM	500	400	207	6.5	7.7	6	4
4	KVM	300	400	103	4	7.7	4.8	3
5	KVM	0	400	0	0	7.7	0	0
6	KVM	500	400	83	5	7.7	3.9	3
7	OVZ	200	200	0	1	3.7	0.8	2
8	OVZ	400	400	6	3.5	3.8	0.6	4
9	OVZ	2500	2400	7	35.5	23.4	10.1	22
10	OVZ	800	400	32	9	7.7	3.3	4
11	OVZ	600	400	1	7	7.7	1.2	6
12	OVZ	400	400	99	6	7.7	2.3	4
13	OVZ	800	400	101	8	7.7	2	4
14	OVZ	1900	1200	103	35	35.2	4.6	4
15	OVZ	1400	1200	751	14	35.2	9.9	4
16	OVZ	1300	1200	15	23.9	35.2	3.8	5
17	OVZ	1100	1200	18	17	35.2	7.1	7
18	OVZ	400	400	0	6.5	7.7	1.2	4
	Total:	**13700**	**12200**	**1617**	**190.9**	**252.4**	**67.9**	**84**

unreasonable to spend much time on solving it, so the most popular strategy to decide on the amount of required resources is "as much as possible" strategy. This strategy inevitably leads to a high degree of underutilization of computing resources and together with the variety of workloads, produced by the VMs, leads to uneven workload distribution. Figures 1 and 2 illustrate a resource consumption by each host in the JINR cloud.

We see that there are 84 active VMs running and 131 computing cores with around 184 GB of RAM allocated for them. The real usage is much less than allocated: CPU utilization sums up to around 20 cores and only around 60 GB of memory is utilized or only 11.8% of allocated CPU and 35.5% of allocated memory is really utilized overall. So, the reasonable and obvious question is: can we somehow optimize this?

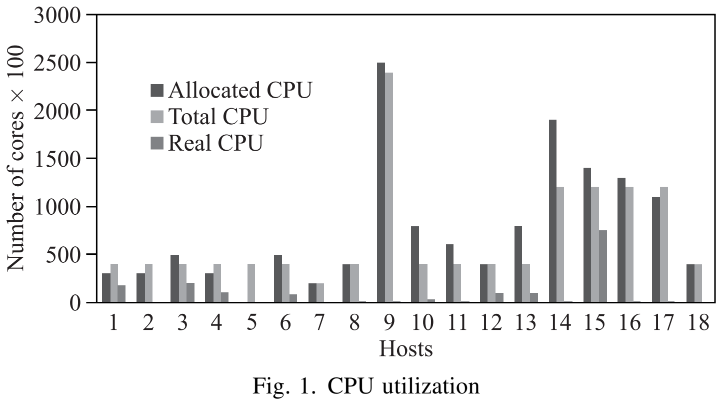

Fig. 1. CPU utilization

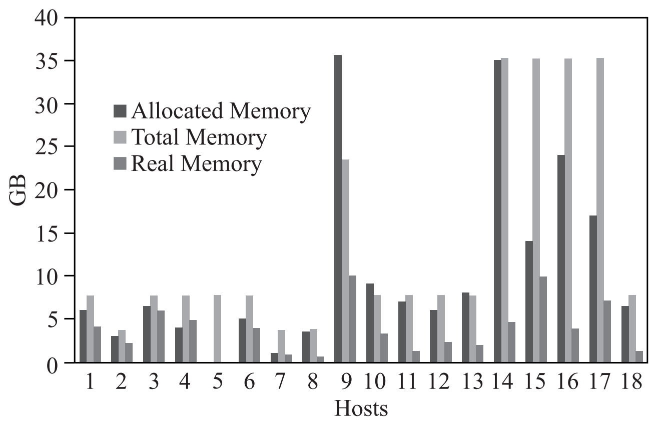

Fig. 2. Memory utilization

A standard way is to overcommit. Basically, it is when we allocate more virtual resources than we actually have, but due to low utilization it keeps working.

As an example, we have only 12 cores on host 14 in the table, but we allocate 19 cores and the overall performance does not degrade because the summarized workload over all VMs on this host equals nearly one core or around 10%. We have no memory overcommitment here. In case of host 10 we overcommit even more CPUs and slightly memory — we try to not overcommit memory much, because in case of memory overload it may start swapping which leads to the significant degradation of performance for all of the VMs on a server. Again what we see here is that the real utilization gets even less with a higher degree of overcommitment than on host 14.

The obvious benefit of overcommitment is a better hardware utilization. However, apparently the more we overcommit, the less reliable gets the service and the quality of service degrades. One more issue with overcommitment in our cloud is that built-in OpenNebula scheduler is not capable of managing overcommitment, so it should be done manually by a system administrator. Doing it manually can be a very time-consuming operation, since we do not want to overload the infrastructure and have to pick a less provisioned host.

To address all of these issues, we decided to implement a custom scheduler, which would automatically deal with overcommitment and also dynamically optimize VM-placement depending on the current workload.

There are some developments of schedulers that allow one to re-allocate VMs dynamically and are able to manage overcommitment, for example, OpenStack Neat [4] for OpenStack platform and Snooze [5] for OpenNebula. However, these systems are aimed at optimizing the energy consumption more than on optimizing the performance of the datacenter, while the scheduler proposed in this paper is aimed at increasing the hardware utilization efficiency with a minimal quality of service degradation based on historical performance metrics gathered from each of the VMs.

Our scheduler relies on the method of dynamic VM re-allocation described below.

First, we give each server some rank with the maximum rank being the best and zero-rank being the worst. In case of a heterogeneous environment like ours at JINR, where we have different servers in terms of hardware, we can use server parameters like number of cores and amount of memory to rank them. In homogeneous clouds with similar servers we can rank them randomly. The goal is to form logical levels based on ranks and we can have several servers on the same level.

Then each VM also gets some rank, but these ranks would change in time, unlike server ranks, and each VM-rank corresponds to particular host-rank.

As you can see in Fig. 3, when user creates a VM, it gets the highest rank and is deployed on the host with the maximum rank. Then the scheduler starts analyzing VM's usage and if it does not show any significant workload for some period of time, its rank is decreased and it gets live-migrated to the host with the lower rank. If, instead, the usage grows, the VM's rank increases and, again, it gets migrated to the higher-rank host. Finally, we have low-loaded VMs stacked up on the low-rank hosts releasing higher-rank servers for VMs with higher resource utilization. The lower the rank the host has, the more stacked it gets, and consequently less reliable.

The development of the scheduler is split up into four phases:

1) implementing and deploying system that would gather VMs performance-metrics;

2) development of interactive cloud model based on metrics gathered on the previous stage;

3) implementation of algorithms for dynamic re-allocation of VMs and testing them on the interactive model;

4) deploying scheduler on the production infrastructure.

The most crucial activities are:

• defining hosts and VMs ranking criteria;

• defining live-migration possibility;

• defining safe VM consolidation thresholds.

The proposed approach is an attempt to solve problems of the inefficient usage of the cloud environment resources by re-allocating them between virtual machines using the accumulated historical metrics of their utilization.

This work was supported by the RFBR, grant 15-29-07027.

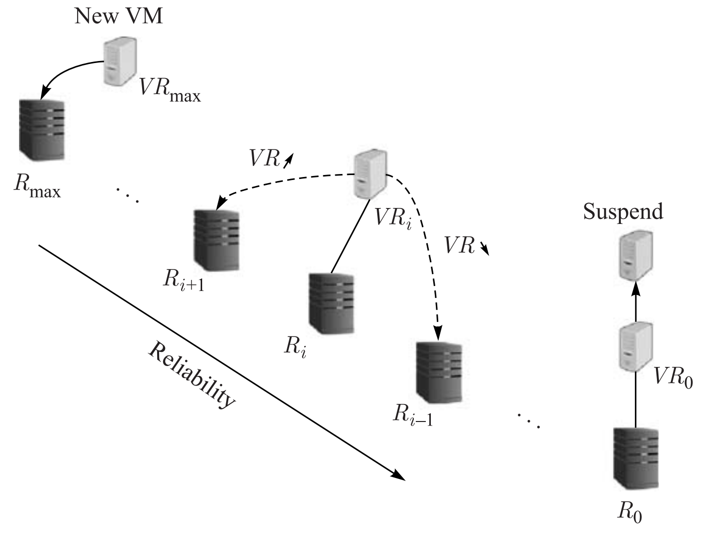

Fig. 3. Dynamic VMs re-allocation method

REFERENCES

1. *Meinhard H.* Virtualization, Clouds and IaaS at CERN // Proc. of the 6th Intern. Workshop on Virtualization Technologies in Distributed Computing, VTDC'12. ACM New York, NY, USA, 2012. P. 27–28.
2. *Hao Wu et al.* Automatic Cloud Bursting under FermiCloud // Proc. of the Intern. Conf. on Parallel and Distributed Systems (ICPADS). 2013. P. 681–686.
3. *Baranov A. V. et al.* Cloud Infrastructure at JINR // Comp. Res. Modeling. 2015. V. 7, No. 3. P. 463–467.
4. *Beloglazov A., Buyya R.* OpenStack Neat: A Framework for Dynamic and Energy-Efficient Consolidation of Virtual Machines in OpenStack Clouds // Concurrency and Computation: Practice and Experience (CCPE). V. 27, No. 5. P. 1310–1333.
5. *Feller E., Rilling L., Morin C.* Snooze: A Scalable and Autonomic Virtual Machine Management Framework for Private Clouds // Proc. of the 12th IEEE/ACM Intern. Symp. on Cluster, Cloud and Grid Computing (CCGrid). 2012. P. 482–489.