Evaluation of monitoring systems for metric collection in intelligent cloud scheduling

I.S. Kadochnikov[1,a], N.A. Balashov[1,b], A.V. Baranov[1,c], I.S. Pelevanyuk[1,d], N.A. Kutovskiy[1,2,e], V.V. Korenkov[1,2,f], A.V. Nechaevskiy[1,g]

[1] Joint Institute for Nuclear Research, 6 Joliot-Curie street, Dubna, Moscow region, 141980, Russia

[2] Plekhanov Russian University of Economics, 36 Stremyanny per., Moscow, 117997, Russia

E-mail: [a] kadivas@jinr.ru, [b] balashov@jinr.ru, [c] baranov@jinr.ru, [d] pelevanyuk@jinr.ru, [e] nikolay.kutovskiy@jinr.ru, [f] korenkov@jinr.ru, [g] nechav@jinr.ru

Infrastructure as a Service clouds have various uses: from application development and integration to parallel computing. Combined within the same cloud this leads to unbalanced and inefficient use of resources. Overcommitment with automated virtual machine migration can help improve efficiency. However, this approach requires real-time information about VM load distribution, as well as historical data to help guide migration strategies. The default monitoring system built into the OpenNebula cloud platform is limited in the possible collection and data processing options. Thus arises the problem of selecting the external monitoring system most suitable for the task. One important aspect to consider is performance, as in large clouds it can be the critical limiting factor. This study proposes a method to test and compare monitoring systems' performance in this context. The monitoring server is installed on a physical machine and set up to collect a small fixed set of metrics from virtual nodes. CPU, memory, disk and network use on the server are recorded for the duration of the test. Monitored virtual machines are started hourly in groups of 50 up to the total of 1000. This paper presents the test results for Ganglia, Icinga2, NetXMS, NMIS and Zabbix. Carrying out this experiment also allowed to assess more subjective properties of these systems, such as documentation quality and ease of use. Based on the performance results, as well the flexibility allowing to add custom metrics, Icinga2 was chosen as the load information collection system for the smart cloud scheduler project.

Keywords: cloud, monitoring, performance

This work was supported by RFBR grant #15-29-07027 "Development of a software package for intelligent scheduling and adaptive self-organization of virtual computing resources based on the LIT JINR cloud center."

Introduction

The Laboratory of Information Technologies of the Joint Institute for Nuclear Research has a cloud based on the OpenNebula platform [Baranov et al., 2016]. This Infrastructure as a Service cloud provides virtual machines to users with very varied computing demands, which creates irregular and inefficient use of the cloud resources. To solve this problem, an intelligent scheduler is being developed, which would automate VM migration and improve efficiency by enabling overcommitment [Balashov et al., 2016]. To apply a migration strategy it needs current and historical information about CPU and memory load on the virtual machines. It is proposed that this data collection is done not by a custom service, but by an existing monitoring system. The default OpenNebula monitoring module is limited in the possible collection and data processing options. Thus arises the problem of selecting the external monitoring system optimal for this project. Using taxonomy proposed by [Montes et al., 2013], this project is concerned with a specialized case of cloud-service-provider-side vision of infrastructure- and server-level monitoring.

Candidate systems

There are several criteria that we used to select candidate monitoring systems for further testing. It needs to be extensible, which would allow adding custom metrics to monitor VMs through the hypervisor. The stored metrics have to be available for use by the cloud scheduler. The system needs to be popular and active so that we can expect performance, stability and a practical architecture. And finally it has to be open-source and free. Table 1 lists the five selected candidates [Massie et al., 2004] [icinga.org] [netxms.org] [opmantek.com] [Tader, 2010]. Nagios, a very popular and flexible system, is not on the list, as Icinga2 is its fork and was tested instead as the likely superior candidate of the class.

Table 1. Examined systems

Name	Storage	License	Extensions	Latest release
Ganglia	RRD, graphite	BSD	plugin API, gmetric push	3.7.2 14.07.2016
Icinga2	graphite, influx, spool	GPL	plugin scripts	2.5.4 30.08.2016
NetXMS	SQL	GPL and LGPL	push API, scripts	2.0.6 02.09.2016
NMIS	RRD	GPL	SNMP only	8.6.0 14.10.2016
Zabbix	SQL	GPL	SNMP only	3.2.1 30.09.2016

Performance testing process

Two physical machines are used for this benchmark. The monitoring server that collects and stores data is installed on one node, the server, and configured to gather information from OpenVZ containers running on the second node, the host. Thus all systems are in identical conditions during the test with regards to physical resources, the network and data flow. Each OpenVZ container runs a monitoring agent that reports to the server and represents a cloud host running VMs and sending their status to the scheduler.

As the set of metrics required by the scheduler is small and uniform, the monitoring systems are configured to collect a fixed list of values from the nodes describing CPU and memory use. The collection interval is set at 5 minutes for all systems. CPU, memory, network and disk load are recorded into CSV files using a simple script on both the server and the host for the duration of the test. On mi-

nute 30 of the test, the host begins launching OpenVZ containers hourly in groups of 50 until 1000 are running. The system and the recording script runs for a day after starting all containers to check for possible instabilities or memory leaks.

After the test the containers are stopped, and the server OS is reinstalled. Afterwards, the next monitoring server is installed on the server and the virtual image is prepared with the agent.

Agent resource usage

During testing it was noticed that the Icinga2 and NetXMS servers lost connection to their agents and marked many nodes as failed. The extent of this problem is illustrated by Figure 1 which shows the aggregation of node status changes from the logging table in the NetXMS database. Further analysis revealed this to be caused by high CPU load on the host machine produced by the agents running on guests, which is shown on Figure 2. This problem makes performance results for Icinga2 and NetXMS less reliable by lowering the amount of actual monitoring data received and processed by the server, as can be seen on Figures 9 and 10.

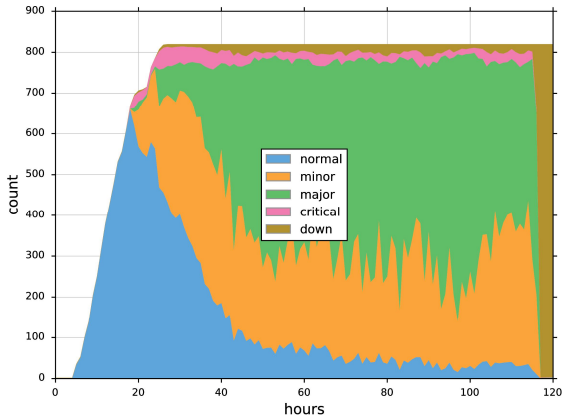

Fig. 1. NetXMS agents by status

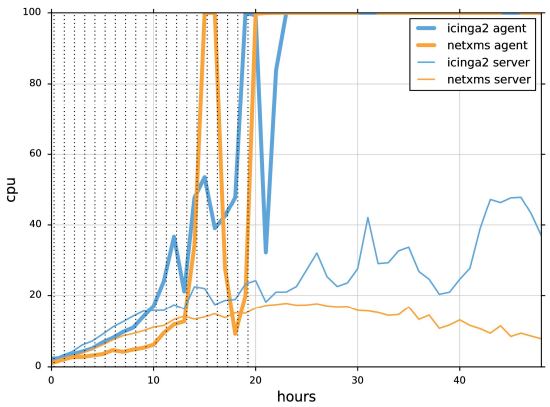

Fig. 2. CPU load for resource intensive agents

As you can see on Figure 3, Ganglia, NMIS and Zabbix agents produce less total CPU load on the host than the respective servers do on the server machine, so their agents are much lighter and they do not exhibit this CPU overload problem. NMIS does not provide its own agent and is focused on SNMP, so NetSNMP was used as the agent in this test. Both Icinga2 and NetXMS can use SNMP to collect metrics, which gives a possible solution to this CPU overload. The protocol change could have an effect on server performance, but it is unlikely to be significant.

In this benchmark only the CPU limit was encountered. For example on Figure 4 it is apparent that there was enough memory for all tests. On bigger scales memory or other system resources may interfere with the test, demanding changes to the benchmarking method. A possible improvement is to use more hosts to run monitored containers, though this would require some basic synchronization between the hosts.

Performance comparison

The final results to compare performance of the tested monitoring system is provided in Figures 5-10.

Fig. 3. Hypervisor and server CPU, lightweight agents

Fig. 4. Server and hypervisor memory use in %

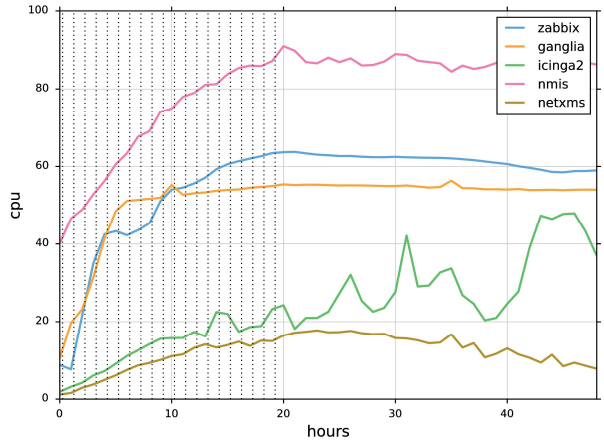

Fig. 5. CPU load on server in %

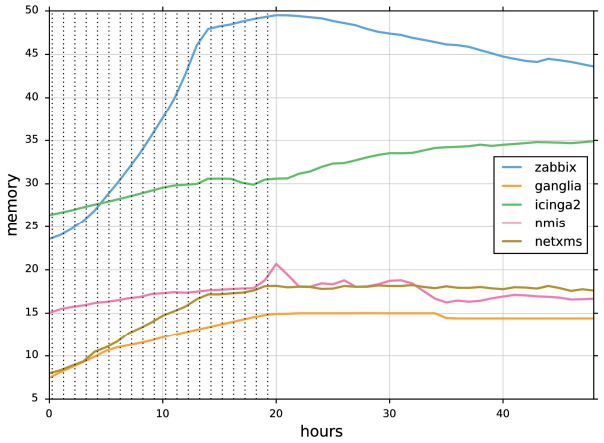

Fig. 6. Memory use on server in %

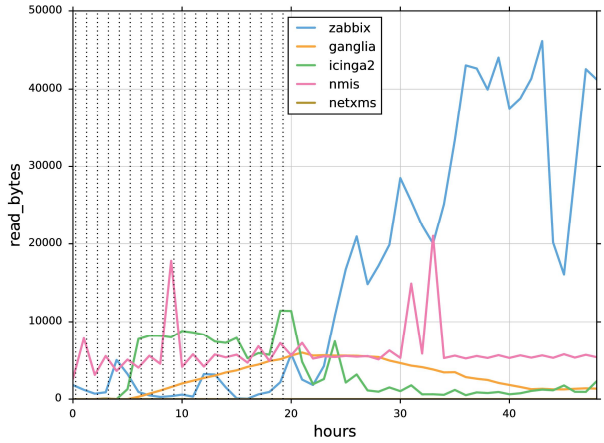

Fig. 7. Disk reads on server in bytes/sec

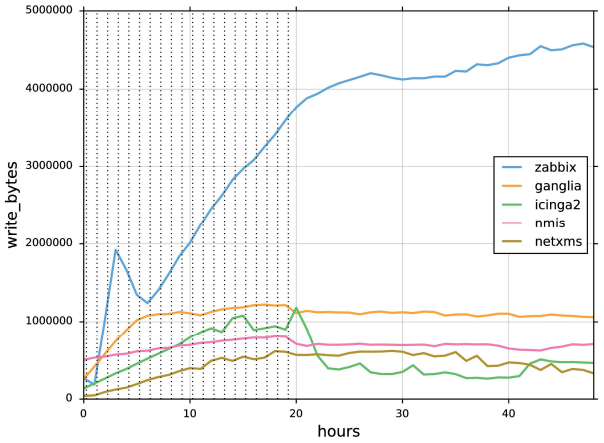

Fig. 8. Disk writes on server in bytes/sec

Conclusions

A small number of systems were evaluated in this study with a limited set of criteria related to a specific task. There are more general surveys of cloud monitoring systems [Fatema et al., 2014].

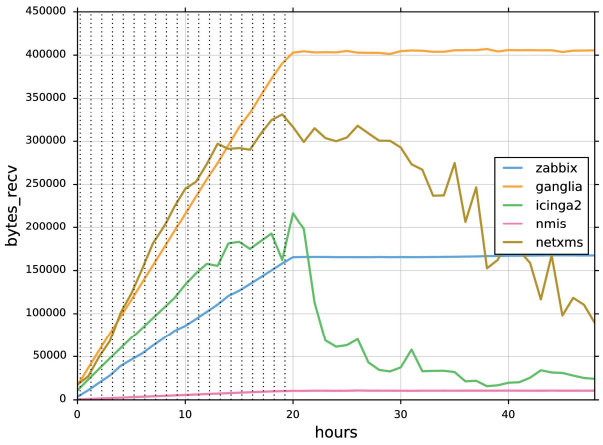

Fig. 9. Server network bytes/sec recieved

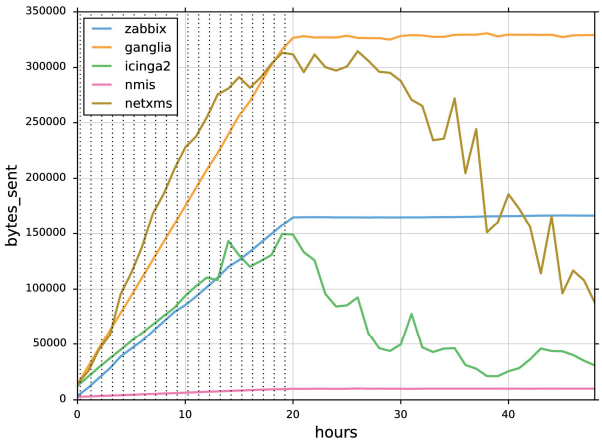

Fig. 10. Server network bytes/sec sent

The method discussed in this paper compares monitoring systems' performance in application to a specific task. Suitability can depend on the job at hand, and performance may be improved by fine-tuning settings of the monitoring server, the database it uses, the operating system and hardware, as well as network architecture and capacity. This study did not aim to isolate and quantify these effects, default settings were used wherever possible for fairness.Beyond the performance results, this project provided hands-on experience in working with the examined systems and allowed to take into account other criteria when selecting the monitoring system to use. One of the most important features is extensibility. For example, when using a plugin module to report custom data to Ganglia, the list of metrics is provided on service start-up. This restricts application of Ganglia to collecting VM load from the hypervisor, as the machines migrate and thus the reported metrics need to change often.

The performance results and other considerations allowed us to select Icinga2 as the system to be used in the future to collect cloud data for the scheduler.

References

Balashov N., Baranov A., Korenkov V. Optimization of over-provisioned clouds // Phys. Part. Nuclei Lett. 2016. Vol. 13, № 5. P. 609–612.

Baranov A.V. et al. JINR cloud infrastructure evolution // Phys. Part. Nuclei Lett. 2016. Vol. 13, № 5. P. 672 675.

Fatema K. et al. A survey of Cloud monitoring tools: Taxonomy, capabilities and objectives // Journal of Parallel and Distributed Computing. 2014. Vol. 74, № 10. P. 2918–2933.

Icinga - Open Source Monitoring [Electronic resource]. URL: https://www.icinga.org/ (accessed: 07.11.2016).

Massie M.L., Chun B.N., Culler D.E. The ganglia distributed monitoring system: design, implementation, and experience // Parallel Computing. 2004. Vol. 30, № 7. P. 817–840.

Montes J. et al. GMonE: A complete approach to cloud monitoring // Future Generation Computer Systems. 2013. Vol. 29, № 8. P. 2026 2040.

Network Management System. Free Software Tools. NMIS [Electronic resource]. URL: https://opmantek.com/network-management-system-nmis/ (accessed: 07.11.2016).

NetXMS [Electronic resource]. URL: https://www.netxms.org/ (accessed: 07.11.2016).

Tader P. Server Monitoring with Zabbix // Linux J. 2010. Vol. 2010, № 195.