U.P.B. Sci. Bull., Series C, Vol. 86, Iss. 3, 2024 ISSN 2286-3540

A STUDY ON MONITORING HETEROGENEOUS CLUSTER INFRASTRUCTURES

Sergiu Weisz[1], Ioan-Teodor Țeugea[2], Maria-Elena Mihăilescu[3], Darius Mihai[4], Mihai Carabaș[5], Nicolae Țăpuș[6]

Monitoring solutions have been a mainstay of cloud infrastructures for the past two decades. Because of this our conceptions about infrastructure need to be re-evaluated. The monitoring paradigm has shifted, and new solutions have appeared to supplant existing software. This paper will analyze the existing cluster monitoring software available, present their advantages and drawbacks. Based on the analysis we will present an applied use case of cluster monitoring based on the tools presented.

1. Introduction

As the size of institutional IT clusters has increased, the likelihood of systems failure has turned from a rare occurrence into an expected cost of doing business. In order for the mission critical services to not be affected system issues have to be detected or prevented from appearing.

System administrators use monitoring tools to extract information about the state of nodes and applications. The status can be represented by metrics, such as request rate, memory usage, response time for applications, or resource

[1]PhD student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `sergiu.weisz@upb.ro`

[2]Masters student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `ioan_teodor.teugea@stud.acs.upb.ro`

[3]PhD student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `maria.mihailescu@upb.ro`

[4]PhD student, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `darius.mihai@upb.ro`

[5]Professor, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `mihai.carabas@upb.ro`

[6]Professor, Faculty of Automatic Control and Computer Science, National University of Science and Technology POLITEHNICA Bucharest, Romania, e-mail: `nicolae.tapus@upb.ro`

usage for hosts. Logs can also be an information source that can indicate issues with nodes or services.

There exists a breadth of monitoring and observability tools that can be deployed and configured. Each solution has advantages, such as optimized information lookup, visualization support, easy scalability and setup.

In this paper we will present the current state of the art of free and open source monitoring solutions. We will focus on their advantages, deployment use cases and disadvantages, taking into account scalability, visualization support, alerting support.

Based on the state of the art we will be presenting a use case for deploying monitoring solutions in a HPC cluster environment situated at the National University of Sciences and Technology Politehnica Bucharest (UNSTPB). The infrastructure description will include the services and types of nodes that we integrate and issues that we have had in deploying the monitoring tools.

2. **Related work**

In this chapter we will be enumerating the different cluster monitoring and alerting solutions used by systems administrators. We will focus on the use cases that they serve, their ability to visualize data in different forms such as graphs, lists and histograms. We will pay attention the ability of the solutions to integrate information provided from different types of systems and applications and display the information in custom dashboards. Community support is another aspect that we will focus on, which can signal the solution's health and future prospects.

We discuss cluster monitoring solutions which can gather information from as many types of systems as possible, and display it using complex visualization methods. The information that is displayed has to be in the form of graphs, because they can display at a glance the system state, and its evolution in a time period. A simple graph is sometimes not enough to display complex information, so we will also look at the other visualization types that can be offered, such as charts and histograms. The monitoring solutions have to be free and open source, and be hosted on-site for security and licensing issues.

Monitoring software has to collect data from the nodes and applications in the cluster. This can be done in a pull or push configuration. Pull configuration connects to each node and downloads information from them, while the push configuration receives the information from each nodes. The pull configuration is recommended in situations where all nodes can be accessed by the monitoring machines, and in systems where the consumers cannot send data to the monitoring systems. The push configuration is used when the client systems can access the central monitoring system, but not vice versa.

Scalability is an inportant factor when discussing desployed services, because clusters increase in size, and their monitoring needs increase too. A solution needs to be able to scale up without the need to pay increased fees, and with the ability to be set up without needing to be re-installed.

2.1. Zabbix

Zabbix [14] is a free and open source network monitoring solution which gathers data from system nodes or networking equipment deployed in the cluster. Zabbix uses a pull method of downloading metrics from systems. This means that an agent has to be developed for Zabbix, which will expose de information that will be downloaded by the server. As it is focused on networking monitoring, the visualization support for Zabbix is limited to 15 types of panels based on maps graphs and charts.

Zabbix [19] is a more complex solution to set up, as opposed to modern solutions such as Prometheus and InfluxDB. It needs an existing web server and database to be set up, which will be connected to it. This added complexity also makes it harder to use for quick lookups and testing, because the data is stored in a specific format in the SQL database.

Zabbix offers an alerting system based on the metrics [21] that are stored by configuring actions in the web interface.

2.2. InfluxDB

InfluxDB [5] is a time series database that collects information from agents running on nodes. It uses the push model to receive metrics from nodes in the cluster; which can be an issue if the number of agents is large, as it can puts a large amount of pressure on the monitor node. The issue being that InfluxDB can offer a HA platform only for paying customers.

InfluxDB has proven to be a popular solution for many types of environments such as IoT [20] or High Energy Phisics [16], because it provides an SQL-like interface, and large programming languages support, which can perform lookups.

As opposed to other solutions such as Zabbix or Prometheus, InfluxDB's use case is for event logging as opposed to metrics gathering. For InfluxDB the focus is not on querying and joining metrics but on being able to look up time based events for logging purposes. InfluxDB supports alerts for specific triggers selected by the users, the same as other monitoring solutions.

InfluxDB does not support visualizations by itself, so it is most often packed together with Chronograf[3], a product of the same developers. It allows users to run test queries and create dashboards to display data from InfluxDB.

2.3. Prometheus

Prometheus [11] is a time series database that collects data retrieved from monitored systems such as nodes and applications. It uses the pull model to download information exposed by endpoints running on the same systems as the monitored solutions. Metrics data is the focus of the time series indexing. The metrics are downloaded from clients which have to respond on a specific port [23]. The issue with the pull model is that it requires routing between all the nodes and the monitoring system and an open firewall for the inter-node communication, which is hard to configure in some instances.

Prometheus offers limited visualization support through a web interface. It allows users to test different queries and see their output, but it doesn't integrate many visualization types, and it doesn't support dashboards.

High availability is available in Prometheus, because it can be deployed with multiple front end server that can be connected to the same database [17].

Prometheus can be integrated with AlertManager [10], an alert engine which can be installed along side Prometheus, and queries Prometheus for metrics information. It can send alerts based on conditions from different metrics. The alerts can be silenced, or repeated as long as the condition matches.

Prometheus benefits from a large community adoption rate because of its ease of use, performance per dollar in virtualized environment and agents ease of use [15].

2.4. Grafana

Grafana [4] is a tool that offers visualization and alerting tools for displaying information based on different data sources. It does not handle data storage, instead connecting to data sources by implementing their querying language.

In most use cases Grafana is found pulling data from time series databases [18], which are specialized organizing data based on the time that it was recorded. These types of databases are used because Grafana can be set up for graphing different metrics, system statuses and metrics which can be used to summarize the status of a system or an application.

Grafana offers a large variety of data visualization options to allow different types of events an metrics to be displayed. Information can be displayed as time series, charts, heat maps, histograms, gauges and more. They can be integrated in dashboards which can pull data from data sources. Grafana dashboards can be build from scratch by the users or they can be downloaded from community forums which export them as JSON files. Dashboards can be further customized through parameters specified by the user.

Grafana supports alerts based on triggers, but it supports fewer features than a more alert focused solution.

2.5. Nagios

Nagios [7] is an open source event monitoring system. It is both used as a stand alone product, used for monitoring service states, and as a base project used by other larger infrastructure projects such as Checkmk and Icinga. The NRPE Nagios agent is required in order for the Nagios service to access the state of a machine. The agent can be made to checked the status of different metrics that can be collected by it

Nagios is primarily used as a health check system for cluster infrastructures [22]. Additional plugins such as RRDtool [12] can be used to also display information obtained from Checkmk, but setting these up increase the setup complexity, and aren't in the stated goals of Nagios.

A web interface is provided by Nagios where the general state of hosts and services can be seen at a glance. Nagios configurations are file based, so changing the system requires is not as simple as for other systems.

3. **Monitoring Infrastructure**

Based on the state of the art for monitoring solutions we have had to make a choice of what system to use in the institutional deployment. The monitoring solution selected should be easy to set up, and configure. It should provide a flexible node configuration system that allows for splitting hosts into groups. We should be able to visualize the state of the different systems deployed, with further detailed panels available for deep dives. An alerting system should be setup which checks the node states and notifies administrators of outages and system failures.

The following chapter presents the infrastructure deployed at UNSTPB, which satisfies the above mentioned requirements, using free and open source software. We will be detailing the configurations made, the software solutions chosen and we will display the resulting visualizations.

3.1. **Cluster infrastructure overview**

The UNSTPB cluster is made up of an OpenStack [9] private cloud infrastructure that servers the compute needs of the teaching and research staff. The private cloud runs on physical machines distributed in three data centers which serve as both compute and service nodes collocated, installed through the `kolla-ansible` [6] deployment tool. All the OpenStack services and physical nodes must be monitored to make sure that the machines are using the resources efficiently, and the services are running correctly.

The cluster also hosts a SLURM [13] grid that runs compute-intensive workloads for users who want to run multi-machine OpenMPI [8] jobs or GPU accelerated applications. The SLURM service status and the node resource usage must be monitored, to make sure that we are able to serve the users' needs.

Both the grid and cloud software use a CEPH [1] storage cluster which is deployed in a single data center location. The storage system must be monitored to ensure system uptime and quality of operations. Because the storage is running on hard disks we must also monitor the health of each disk, to prevent disk failures which can result in data loss.

3.2. Implementing the runtime data collection service

The OpenStack infrastructure hosts a large number of services and the nodes, which requires a scalable method to regularly monitor host and service health. Logs are not enough to inspect the health of a service, because they are a way to display certain errors, but they do not display a simple to parse message. Metrics can be used to parse at a glance the general state of services and systems. They can be used to determine both the operational parameters in which services and systems are expected to run, and also deviations from the expected behaviour. Logs would only allow us to see when a system actually fails, not the iterative process of its failure.

For the collection of statistical data, we chose to use the Prometheus service. Prometheus works by connecting to a list of endpoints configured to export runtime parameters.

We chose to use Prometheus because it runs using the pull model, where it connects to services, instead of the push model, where services connect directly to a central node. The advantage of using the pull method is that we do not have to provide a central node with a large number of resources dedicated to it. Another advantage of the Prometheus service is that it provides an easy query interface where we can perform complex queries to obtain information in the form of time series. The Prometheus community also provides a large array of tools and agents to monitor different services and hardware, so that we would not need to implement agents specific to our own use case.

The disadvantage of using Prometheus is that we have to ensure connectivity between the system on which Prometheus is running and all the stations from which we want to collect information. Thus it is necessary to configure network routes, entries in firewall systems and configurations on nodes that allow connection between nodes. All these aspects add complexity to the system and make it harder to maintain, but we can solve these aspects by automating the generation of configurations for nodes, thus removing the human element from configuring endpoints.

Prometheus provides a graphical interface in the form of a web page where we can run queries and view the results. Figure 1 displays the Prometheus service GUI querying the number of virtual machines running in the OpenStack infrastructure. The web interface can be used for test queries, but in order to have broader visualization support, one has to run another service which will serve information from Prometheus.

Each configured service has a different endpoint to which it publishes the information and uses a different way to aggregate it because each service works differently. For each type of service we need to configure an application (called an exporter) that will collect the information, aggregate it and publish it. Each exporter has a different configuration mode and receives different parameters.

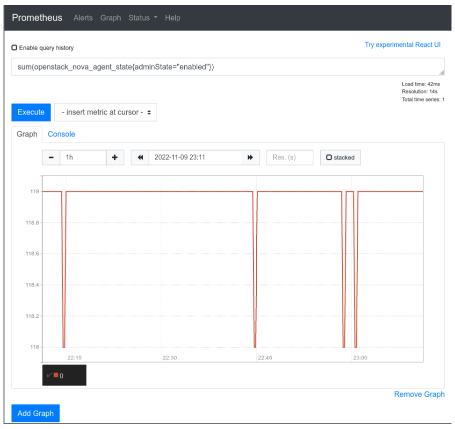
Prometheus is configured to connect to the exporter using the HTTP protocol and download the exposed information. Listing 1 shows a Prometheus job which will download service information from the OpenStack infrastructure exporter.

```
- job_name: 'openstack_exporter'
  static_configs:
  - targets: [
      'os-controller.grid.pub.ro:9198',
  ]
```

LISTING 1. Prometheus exporter configuration

Since it was necessary to configure the Prometheus service for more than 100 physical nodes, each hosting several services, we developed a set of scripts that generate Prometheus configuration files based on a set of services and node groups that are associated with the. This script reduces the number of lines, and implicitly the number of mistakes, that can be added to the Prometheus configuration file by a systems administrator.

To configure an exporter on a node, we can follow the example of the `node_exporter` exporter. It is installed using the distribution's package manager. A systemd unit file is created for the exported, which can then be started on the node.



```
sum(openstack_nova_agent_state{adminState="enabled"})
```

Load time: 42ms
Resolution: 14s
Total time series: 1

2022-11-09 23:11

FIGURE 1. Viewing OpenStack compute node in Prometheus

Because the Prometheus service ingests a large amount of data, we have chosen to implement a data retention policy. This policy is intended to automate the removal of old monitoring data. The system has been configured to retain data collected up to one year. We chose this policy because it allows us to analyze historical data, which would allow us to identify gradual changes to the system without causing uncontrolled disk space growth. This policy should be revisited each time a new service is integrated, as the metric download rate may become higher than the old metric deletion rate.

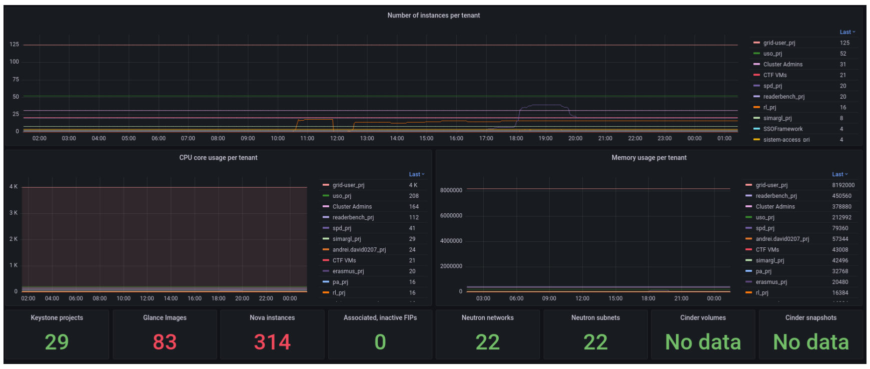
We have integrated the following information using different exporters:

- OpenStack service status and statistics;
- Compute node resource usage;
- Web service status, such as GitLab, login services, e-learning platforms and storage systems;
- SSL certificate lifetime for managed web services;
- Data center generator power and fuel status;
- ICMP prober for checking latency between systems;
- SLURM grid metrics;
- Storage statistics such as bandwidth usage and IO time;
- GPU usage metrics.

3.3. Investigating the health of services using dashboard

For advanced investigation of systems health, we chose to use the Grafana service to generate graphs and dashboards from which we can visualize complex images based on time series. We have chosen to use Grafana because of its extensive visualization support, because different services have different visualization needs. Grafana offers us the query functionality of the Prometheus service, so we can use the Prometheus Query Language queries that we have used so far directly from the Prometheus dashboard to easily view the information.

Figure illustrates dashboard elements for the OpenStack system. At a glance we can see the following information: the number of virtual machine images are installed, the number of virtual machines started, the number of virtual machines started per tenant, the number of CPUs used per tenant, the amount of RAM used by each tenant.



Number of instances per tenant

	Last
grid-user_prj	125
uso_prj	52
Cluster Admins	31
CTF VMs	21
spd_prj	20
readerbench_prj	20
rl_prj	16
simargl_prj	8
SSOFramework	4
sistem-access_pri	4

CPU core usage per tenant

	Last
grid-user_prj	4 K
uso_prj	208
Cluster Admins	164
readerbench_prj	112
spd_prj	41
simargl_prj	29
andrei.david0207_prj	24
CTF VMs	21
erasmus_prj	20
pa_prj	16
rl_prj	16

Memory usage per tenant

	Last
grid-user_prj	8192000
readerbench_prj	450560
Cluster Admins	378880
uso_prj	212992
spd_prj	79360
andrei.david0207_prj	57344
CTF VMs	43008
simargl_prj	42496
pa_prj	32768
erasmus_prj	20480
rl_prj	16384

Keystone projects	Glance Images	Nova instances	Associated, inactive FIPs	Neutron networks	Neutron subnets	Cinder volumes	Cinder snapshots
29	83	314	0	22	22	No data	No data

Figure 2. Displaying OpenStack resource usage using Grafana

Grafana is configured to access information based on the time frame we request access to. Since there have been cases where there are too many systems for which we collect the information, we have chosen to add to each metric in Prometheus a tag that defines the type of service provided, so that we can filter the metrics in queries based on the type of service we want to investigate. We can see this in the dashboard showing latency in Figure .

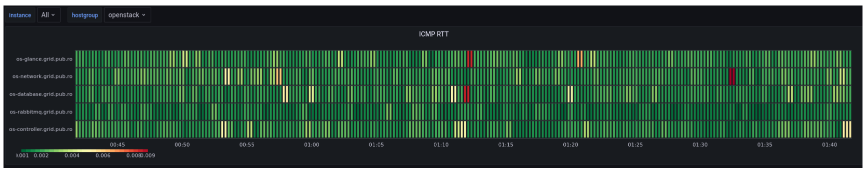


Figure 3. Displaying services latency using Grafana

We observe in Figure the variable `hostgroup` that defines the group of instances we want to query. In the example we showed, the query was addressed for the OpenStack systems.

The advantage of using Grafana for graphs is that it has a very flexible suite of visualization modes, which allows us to present information in many different ways, from regular graphs and numerical summaries to heatmaps and histogram graphs. We can integrate other dashboards in Grafana by creating our own or using the existing dashboards published by the community.

We have integrated into Grafana dashboards for viewing the following elements:

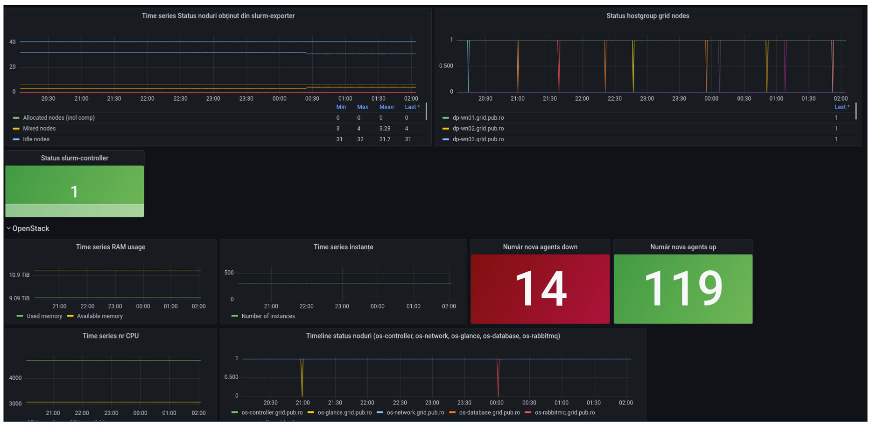
- OpenStack infrastructure status;
- Connection latency between nodes;
- Web services status;
- Power generator status;
- Resource consumption of host nodes;
- CEPH storage status;
- Grid services status and usage;
- GPU usage per node;

It is difficult to keep track of all the dashboards because a large amount of metrics is collected from many different types of services. To make it easier to check resources at a glance, we've implemented an overview dashboard that we refer to often. We can see a fragment of it in Figure .

3.4. Monitoring disk health

UNSTPB runs a large amount of hard disks for different use cases. Hard disks wear out over time, which requires monitoring for errors that might appear due to hard disk failure.

We needed to have a method to check the disk health and prevent a situation where they might fail during high demand loads. To inspect the disk state at a given time, we can use the S.M.A.R.T. metrics exposed by the disk, which gives us insight into disk wear. The S.M.A.R.T. metrics that we are



	Min	Max	Mean	Last *
Allocated nodes (incl comp)	0	0	0	0
Mixed nodes	3	4	3.28	4
Idle nodes	31	32	31.7	31

	Last *
dp-wn01.grid.pub.ro	1
dp-wn02.grid.pub.ro	1
dp-wn03.grid.pub.ro	1

FIGURE 4. Cluster services overview configured in Grafana

interested in are the read error rate, the running temperature of the disk, and how many errors occur when transmitting data from the disk to the system. They can indicate the first signs that a disk might fail in the near future.

To interpret the S.M.A.R.T. characteristics we use the values recommended by the manufacturers for the condition of the discs. An example of metrics exposed by S.M.A.R.T can be viewed in Listing **??**.

TABLE 1. S.M.A.R.T metrics exposed for CEPH disks

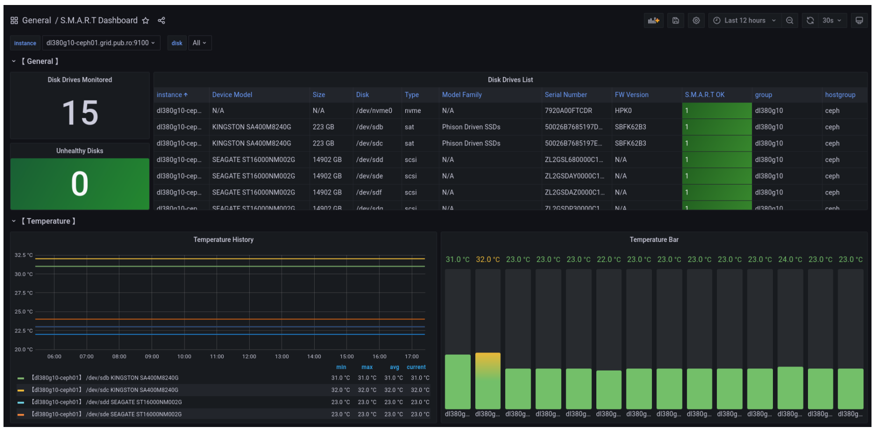
ATTRIBUTE NAME	**VALUE**	**WORST**	**THRESH**
Raw_Read_Error_Rate	100	100	000
Power_On_Hours	100	100	000
Power_Cycle_Count	100	100	000
CRC_Error_Count	100	100	000
SSD_Life_Left	099	099	000

The columns of interest to us are:

(1) `VALUE`, the last value read;
(2) `WORST`, the largest value read;
(3) `THRESH`, the value from which the disk can be considered defective.

To expose the metrics in an easy-to-read environment, we used a script that collects the S.M.A.R.T. and exposes them using the already installed `node_exporter` service.

The Prometheus service, installed on the Grafana instance automatically, pulls the metrics. They are used to output generate Grafana dashboard through which we can view the state of the disks, where disks suspect of failure can be highlighted through the S.M.A.R.T. metrics.



Disk Drives List

instance	Device Model	Size	Disk	Type	Model Family	Serial Number	FW Version	S.M.A.R.T OK	group	hostgroup
dl380g10-cep...	N/A	N/A	/dev/nvme0	nvme	N/A	7920A00FTCDR	HPK0	1	dl380g10	ceph
dl380g10-cep...	KINGSTON SA400M8240G	223 GB	/dev/sdb	sat	Phison Driven SSDs	50026B7685197D...	SBFK62B3	1	dl380g10	ceph
dl380g10-cep...	KINGSTON SA400M8240G	223 GB	/dev/sdc	sat	Phison Driven SSDs	50026B7685197E...	SBFK62B3	1	dl380g10	ceph
dl380g10-cep...	SEAGATE ST16000NM002G	14902 GB	/dev/sdd	scsi	N/A	ZL2GSL680000C1...	N/A	1	dl380g10	ceph
dl380g10-cep...	SEAGATE ST16000NM002G	14902 GB	/dev/sde	scsi	N/A	ZL2GSDAY0000C1...	N/A	1	dl380g10	ceph
dl380g10-cep...	SEAGATE ST16000NM002G	14902 GB	/dev/sdf	scsi	N/A	ZL2GSDAZ0000C1...	N/A	1	dl380g10	ceph
dl380g10-cep	SEAGATE ST16000NM002G	14902 GB	/dev/sdg	scsi	N/A	ZL2GSDP30000C1	N/A	1	dl380g10	ceph

	min	max	avg	current
【dl380g10-ceph01】 /dev/sdb KINGSTON SA400M8240G	31.0 °C	31.0 °C	31.0 °C	31.0 °C
【dl380g10-ceph01】 /dev/sdc KINGSTON SA400M8240G	32.0 °C	32.0 °C	32.0 °C	32.0 °C
【dl380g10-ceph01】 /dev/sdd SEAGATE ST16000NM002G	23.0 °C	23.0 °C	23.0 °C	23.0 °C
【dl380g10-ceph01】 /dev/sde SEAGATE ST16000NM002G	23.0 °C	23.0 °C	23.0 °C	23.0 °C

FIGURE 5. Disk health check Grafana dashboard

4. Conclusion and Further Work

We have researched the optimal monitoring system for managing metrics for a private cloud infrastructure which also host institutional and e-learning systems.

Each viable monitoring solution has been analyzed and we have deployed Prometheus to download metrics from nodes, storage systems and services. Grafana was used to display the metrics in a new dashboards that allow administrators to see at a glance the cluster state.

We have deployed an alert system for notifying administrator when services fail, SSL certificates expire, or when disks show signs of degradation. The new system has allowed us to pre-empt disk failure incidents by replacing disks and rebuilding storage arrays in time and notice anomalous events in service behaviours leading to bug fixes.

In the future we aim to move the monitoring setup from virtual machines to a container-based environment with automated setup, so that it can be easier to install and replicate for further use.

REFERENCES

[1] Ceph website. https://ceph.io/. Accessed: 2023-10-04.
[2] Checkmk website. https://www.checkmk.com. Accessed: 2023-10-04.
[3] Chronograf website. https://www.influxdata.com/time-series-platform/chronograf/. Accessed: 2023-10-04.
[4] Grafana website. https://grafana.com/. Accessed: 2023-10-04.
[5] Influxdb website. https://www.influxdata.com/. Accessed: 2023-10-04.
[6] Kolla-ansible website. https://docs.openstack.org/kolla-ansible/latest/. Accessed: 2023-10-04.
[7] Nagios website. https://www.nagios.com. Accessed: 2023-10-04.
[8] Openmpi website. https://www.open-mpi.org/. Accessed: 2023-10-04.
[9] Openstack website. https://www.openstack.org/. Accessed: 2023-10-04.

[10] Prometheus alertmanager website. https://prometheus.io/docs/alerting/latest/alertmanager/. Accessed: 2023-10-04.

[11] Prometheus website. https://prometheus.io/. Accessed: 2023-10-04.

[12] Rrdtool website. https://oss.oetiker.ch/rrdtool/. Accessed: 2023-10-04.

[13] Slurm website. https://slurm.schedmd.com/documentation.html. Accessed: 2023-10-04.

[14] Zabbix website. https://www.zabbix.com/. Accessed: 2023-10-04.

[15] C. Anglano, M. Canonico, and M. Guazzone. Prometheus: A flexible toolkit for the experimentation with virtualized infrastructures. *Concurrency and Computation: Practice and Experience*, 30(11):e4400, 2018.

[16] T. Beermann, A. Alekseev, D. Baberis, S. Crépé-Renaudin, J. Elmsheuser, I. Glushkov, M. Svatos, A. Vartapetian, P. Vokac, and H. Wolters. Implementation of atlas distributed computing monitoring dashboards using influxdb and grafana. In *EPJ Web of Conferences*, volume 245, page 03031. EDP Sciences, 2020.

[17] M. Brattstrom and P. Morreale. Scalable agentless cloud network monitoring. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 171–176. IEEE, 2017.

[18] M. Chakraborty and A. P. Kundan. Grafana. In *Monitoring Cloud-Native Applications: Lead Agile Operations Confidently Using Open Source Software*, pages 187–240. Springer, 2021.

[19] A. Dalle Vacche. *Mastering Zabbix*. Packt Publishing Ltd, 2015.

[20] M. Nasar and M. A. Kausar. Suitability of influxdb database for iot applications. *International Journal of Innovative Technology and Exploring Engineering*, 8(10):1850–1857, 2019.

[21] R. Olups. *Zabbix Network Monitoring*. Packt Publishing Ltd, 2016.

[22] J. Renita and N. E. Elizabeth. Network's server monitoring and analysis using nagios. In *2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 1904–1909. IEEE, 2017.

[23] J. Turnbull. *Monitoring with Prometheus*. Turnbull Press, 2018.