Azerbaijan Journal of High Performance Computing, Vol 3, Issue 1, 2020, pp. 139-146
https://doi.org/10.32010/26166127.2020.3.1.139.146

The Hybrid Process and Data Monitoring Tools for High Performance Computing Systems

Farid Jafarov

Azerbaijan State Oil and Industry University, Baku, Azerbaijan, feridcefer@hotmail.com

*Correspondence: Farid Jafarov, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, feridcefer@hotmail.com

Abstract

Due to the growing demand for parallel programs and mathematical and industrial problems that require high performance computing, cluster systems are one of the essential parts of the computing world. Clusters have different features, and there are multiple topics can be studied while working in these systems. In this paper, the related issues about visualization procedure in clusters will be discussed, analyzed, and researched. There are quite similar visualization manners in a cluster environment, like the visualization of hardware usage, process flow, data flow, environmental data, etc. And the majority of contemporary monitoring tools were prepared with a particular target; for example, Nagios was developed to mainly monitor network and devices related to the work of the network. Hence, it is almost impossible to find a unique tool that can be used for most of the purposes mentioned above. This paper will mention the possibilities of having unique monitoring tools for most cluster systems.

Keyword: High performance computing, Clusters, Visualization tools, Daemons, Monitoring

1. Introduction

There is a growing demand for high performance computing in almost all areas in which computation speed is a crucial matter. To improve computing speed in many cases, scientists use different techniques such as parallel computation, in which HPC centers play an integral role. High Performance Computing Centers, namely, the computers connected, work together to solve a massive computational problem. Hence, in general, High-Performance Computing clusters are the connection of multiple nodes and the master (server) nodes, which handle most of the heavy lifting of the system. Nodes and servers are connected with fast local area networks into each other. The jobs are executed on these systems being parallel and skyrocket the execution time.

Obviously, in this type of messy environment, always some downsides occur. Thus, someone needs to see what is going on and where the problem comes from. Or else, while executing a program to stabilize and analyze the job, machine states, environmental issues (humidity, temperature, etc.). This is the time the visualization for HPC comes into place. This is not the new problem, so there are quite a few visualization tools that exist for that since

2000 and before. There are quite many monitoring tools like Ganglia, Nagios, HPC-toolkit, CluMON, Supermon, ClOver, Zabbix, and so on (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009; Adhianto, L., Banerjee, S., Fagan, M., et al., 2010).

System administrators often need to decide which visualization tool to use in their systems. In this place, it is important to define the advantages and disadvantages of these tools. For example, the administrator should know that if he needs to analyze the custom metrics of the CPU, he needs to write additional configuration scripts using C or python and use it in Ganglia. Or, the administrator who uses Nagios should be aware of how it is simple to add plug-ins in it. The administrator of HPC clusters in which there are thousands of cores should be careful about Zabbix, as large scale implementations of 10.000+ are not useful with this tool.

In general, CPU cycles, CPU idle, RAM use, Storage, I/O, GPU utilization, network flow, job states are in the most important key features list for the system administrators to see. Collecting and sending all the useful requirements to the server is the job of local daemons, which is located in the nodes. The daemons collect data from system logs, services hierarchy, hardware, and send them to listen to daemons, which is located in the server. Generally, the incoming data is huge, and taking into account that, some amount of data comes from dozens of nodes to analyze, save and show them to the administrator required a few tools or services to work on. Namely, the daemon analyzes the data and convert it into the format that can be saved into databases. Showing collected data is in the response of web application which reads data from databases and turns it into graphs, diagrams, charts, tables.

Sometimes, the combination of the visualization tools is used in the HPC clusters. It affects the workload of the systems over than its need. If the user's purpose is to analyze specific metrics and also apply some custom plug-ins, then they will need to use Ganglia and Nagios at the same time. In this situation, it might be asked, is it not possible to build the unique visualization tool that can be used for almost all purposes? Developing this type of imaginary tool is practically impossible. Because, by the simple view, the services used in this tool cannot be workable or error-free for all distributions. Besides, the development period will be longer than usual. Furthermore, as it needs to use more powerful daemons, it will affect the performance of the system much than other previous tools.

In the 2nd section, it will be discussed different visualization tools that are used by a large community. The architecture of those tools, the daemons which they use, etc. details will be discussed separately. In the end, there is a table that represents positives and drawbacks using those mentioned tools. In the 3rd section, an approach to achieve a unique solution to the problem will be given and explained. The important monitoring functionalities will be mentioned in bullet form, and later on, will be described in detail. The 4th and at the same time, the final section, will demonstrate the conclusion of the work.

2. Related work

In the contemporary world, most of the monitoring tools exist, and the majority of them focus on the particular dimension of measurement. For example, some part of tools mainly chooses job state analyzing as the main target to measure, but others are used to monitor hardware-specific metrics. There are also the monitoring tools that take care of scalability while monitoring most.

Ganglia is a scalable distributed monitoring system that was created for large localized clusters, namely high-performance computing systems or even widely distributed Grids (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009; Massie, M. L., Chun, B. N., & Culler, D. E., 2004; Sacerdoti, F. D., Katz, M. J., Massie, M. L., & Culler, D. E., 2003). Ganglia differs from other monitoring tools by the architectural approach that it is used in (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009; Massie, M. L., Chun, B. N., & Culler, D. E., 2004; Sacerdoti, F. D., Katz, M. J., Massie, M. L., & Culler, D. E., 2003). Gmond (Ganglia monitoring daemon) and gmetad (Ganglia meta daemon) are the main daemons used in Ganglia. Ganglia uses Round Robin Databases (RRD tool) as the data storage (Massie, M. L., Chun, B. N., & Culler, D. E., 2004; Massie, M., Li, B., Nicholes, B., Vuksan, V., et al., 2012). Gmond should be installed in each node, and the task of gmond is to collect metric data from different hardware devices. Gmond sent the collected data into Gmetad using XML streams, which is sent by TCP connections (Massie, M., Li, B., Nicholes, B., Vuksan, V., et al., 2012). The job of gmetad is to process the incoming data and convert it into the custom formats, which can be readable for RRD tool (Massie, M. L., Chun, B. N., & Culler, D. E., 2004; Massie, M., Li, B., Nicholes, B., Vuksan, V., et al., 2012). From this point on, the scripts which were coded using PHP, fetch data from RRDs and convert it into human-readable graphs and diagrams. Although RRD tools are very useful and system efficient, using write/read periodically might lead to performance bottlenecks (Massie, M., Li, B., Nicholes, B., Vuksan, V., et al., 2012). Especially, its archiving techniques make too many updates causing unnecessary disk I/O. Unlike other modern visualization tools, the ganglia alert system is not well prepared for the needs of administrators. Besides, having the individual view for each cluster might look fancy, but when the number of nodes comes over thousands to get the general view of cluster health to become a trouble.

Nagios is the monitoring tool for cluster systems and networks (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009; Burgess, C., 2005). The power of Nagios comes from its plug-in based architecture. Nagios itself knows nothing about the system hardware or software. Nagios shows the incoming data from plug-ins. There are currently many ready plug-ins available for community use, including IMAP, HTTP, FTP, POP3, SSH, CPU Load, DHCP, Disk Usage, Unix/Linux, Memory Usage, Current Users, Netware Servers, Windows, Routers and Switches, and others. Since there are no restrictions or criteria, different database software tools can be used (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009; Burgess, C., 2005). The Nagios itself only works on server and ping data from the plug-ins which work on different hosts(nodes) (Burgess, C., 2005). Thus, the simple web interface shows the information in different graphs and charts to administrators. Additionally, Nagios has a notification system; whenever something goes wrong on the system, its alerting system sends emails, SMS, or push notifications to the defined users (Montaldo, D., Mocskos, E., & Slezak, D. F., 2009). Again, one of the drawbacks of the Nagios is its lack of general view of cluster healthiness and dependency over plug-ins.

An increasing number of technological and scientific research has increased the demand on HPC clusters, which lead to the development of exascale systems. The exascale systems capable of executing a quintillion (10^18) floating-point operations in a second. At the same time, it enables for science and business more opportunities than ever. The storage capacity, processors' architecture, and other hardware details of these systems are too complex to monitor with current visualization tools. The efficient monitoring and management of extreme-scale HPC environments mandate the utilization of automation and the discovery of bottlenecks in dynamically changing production environments. Unlike the traditional visualization tools, the newly developed visualization tools for exascale systems monitor the environmental features such as cooling, humidity, etc. which might influence the performance of the system. The alerting system is one of the key features of traditional visualization tools. These systems work well with a small number of computers; however, when the numbers pass thousands, the limitless number of alerts might confuse system administrators. Hence, finding out the actual problem might be harder or else impossible. So, the designers of contemporary monitoring tools have tried to minimize the duplication of alerts for common problems using machine learning algorithms. This strategy made alerting systems stricter than the previous ones to show the actual problem. OMNI (Operations Monitoring and Notification Infrastructure) is one of the powerful visualization tools that was developed for efficient monitoring of exascale HPC systems (Bautista, E., Romanus, M., Davis, T., Whitney, C., & Kubaska, T., 2019. Integrating Kubernetes, Prometheus, and Grafana platforms, the power of the device doubled (Sukhija, N., & Bautista, E., 2019). These platforms make OMNI available for dynamic deployment, proactive alert management, data visualizations, and predictive analytics to avoid outages. Scalability, management of multiple alerts, and an opportunity to see practical metrics in a single view and other well-prepared properties push OMNI in front of Nagios, Ganglia, Zabbix, Spiceworks, etc. (Sukhija, N., & Bautista, E., 2019). However, in this stage, it needs to notify for the small HPC clusters deploying traditional tools is useful and practical for their light-weight resource use, rather than OMNI (Sukhija, N., & Bautista, E., 2019).

Table 1. Advantages and disadvantages of Ganglia, Nagios, and OMNI with respect to the special features.

Features	Ganglia	Nagios	OMNI
Advanced web UI for visualization of hardware	+	+	+
Completely free to use	+	-	+
Open-source (or prepared by the configuration of open-source software tools)	+	+	+
Automatic device discovery	-	-	+
High level of robustness against system failures	+	-	+
Modern alert system	-	+	+
Smart alert system	-	-	+
Works out of the main cluster manager architecture	+	+	+

High-level job state/resource monitoring	-	-	-
Advanced web UI for JOB state/resource visualization	-	-	-
Handy for small-sized clusters	+	+	-
Active development team support	-	+	+
Monitor network flow & devices in high scrutiny	-	+	-
Active community support	+	+	-

Table 1 demonstrates the positive and negative sides of the tools discussed above. As can be seen from the table, all of the visualization tools have modern WEB UI, which meets almost all requirements for users and system administrators. Furthermore, all of them are open-source, which means the users can modify and their custom functionalities into them with solid knowledge. "+" sign in front of the feature illustrates that the tool has this capability. "-" defines the tool mentioned in the top of the column does not have the feature mentioned in a row. It can be clearly seen that none of the tools above can monitor job/process state monitoring. Besides, the features like automatic device discovery, smart alert system are only available in OMNI. However, neither OMNI nor Ganglia can monitor network flow and devices as detailed as Nagios. Above, it is not handy to write all features and compare them into the table; only strong points were mentioned and compared.

3. Our approach

The continuing development of scientific and industrial problems which require longer execution time and use massive data to achieve successful results in the end. In the contemporary world, to solve the majority of those problems, the supercomputers or high-performance computing clusters are used. Certainly, the users (or system administrators) need to see detailed monitoring information about the status of jobs and also usage of hardware details of cluster nodes. Today's monitoring tools can monitor hardware usage of nodes in high scrutiny. However, finding a visualization tool that can monitor process and data flow in the cluster is not easy. There are multiple reasons that monitoring tools developers and designers avoid creating tools for job monitoring. Maybe the strongest point is the dissimilarity of architectures used in cluster systems. Namely, the daemons can track the job and data flow in the X cluster manager tool, cannot be used to do the same job with the Y cluster manager tool. Hence, the majority of tools cannot monitor the process and data flow of clusters while executing the process. Furthermore, the monitoring tools which can be used for visualizing hardware usage, process, and data flow is approximately impossible to find.

Thus, it can be seen in the points mentioned above, the cluster world needs one unique tool which can monitor hardware details, process, and data flow, respectively. Firstly, it is not a straightforward problem to handle with just one paper. There are multiple challenges to achieve this type of tool. Firstly, the daemons should be chosen for each purpose. Namely, if it is wanted to get hardware details of nodes in the cluster, then there should be at least one daemon in for each node that collects hardware usage specifications from logs, etc. places and send them into listener daemons in server. A similar situation should happen for monitoring of process flow and data flow in cluster systems. It is important to have multiple high robust daemons to achieve this visualization tool. In every node, there should be services that finalize the job of monitoring both from the node and server-side. The required daemons will be listed above in general:

a) The daemons for collect metrics data from hardware details

b) The daemons for process flow tracking and monitoring

c) The daemons for data flow visualization

d) The daemons for monitoring network flow and devices

Certainly, to achieve high robust and scalable visualization tool like one mentioned above will not be possible with only using a few services. The ones mentioned above are the key daemons of the ideal monitoring tool.

As other traditional monitoring tools, there is always one important feature that should be considered first while designing the visualization tool. It is the availability of monitoring hardware resource usage and device states of nodes in the cluster. The gmond daemon was used in Ganglia is a great example of this type of daemon. Gmond can collect any type of low-level hardware detail like CPU idle, CPU usage, etc. by default. Hence, there should be at least one strong daemon in this monitoring environment to cope with the problems of metrics collection from hardware sections. Furthermore, this daemon should be prepared with low-level programming languages, namely with C. In this case, adding and removing custom scripts to get monitored data from exact devices in high scrutiny will be possible for system administrators or specific end users.

As one of the main targets of this imaginary visualization tool to design visualization tools for the collection process flow, this daemon should be taken into account in the first place. Before diving deeper in detail, it needs to be mentioned that to collect process (job) flow information from nodes is not an easy as collect hardware monitoring details. There are quite a few constraints that prevent achieving comfort in the visualization of process flow. For instance, the differences in cluster manager tools, architectural problems occurred by operating systems, etc. are just a few examples of dozens of constraints. First, all of the problems should be considered before applying this daemon into visualization tool architecture. Since the process flow is highly dynamic and is able to generate big data in a few minutes, the dangerous sides of those kinds of daemons should be counted beforehand. For example, the slurmd daemon collects and tracks process flow information in high scrutiny from each node in the cluster (Jette, M., & Grondona, M., 2003). Even it separates incoming data with respect to racks, computers, and targeted jobs. Slurmd daemon will be great to use for this purpose. However, the main downside of this daemon is that it can only work within SLURM architecture (Jette, M., & Grondona, M., 2003). Namely, using slurmd for this purpose decreases the portability of the monitoring tool, dramatically.

With regards to the monitoring of job data within the cluster is maybe the hardest side of the job. Since the data often comes into big data because of the dynamic architecture of parallel programs. The data flow between nodes of cluster and server will create massive headless data streaming in cluster systems. The daemons which should monitor and track these data flows should count dozens of specifications, including resource usage of RAM and other important hardware details. As it is known well, the data which is created while executing programs is volatile and processed over RAM. And reading that required information to get data flow in detail requires the possibility of speaking with the lowest level operating system architecture.

Additionally, the data flow should be authenticated to be aware of how streaming happens between nodes or even in nodes' local memory. This part is as important as the previous section because data means everything for the parallel programs. Most of the jobs depend on it, and if the corrupted data or any other problems occur during the execution process, it may lead to uncontrollable behavior and results.

Last but not least, as the network is essential for clusters like the CPU, etc. it is necessary to be able to what is going on with network flow and devices. Besides, all the mentioned daemons and services above need to strong network connection and speed to fulfill the user requirements and purposes. Hence, to visualize network flow, the device and status of connections require another separate daemon or services which should collect data of network status, connection states, and usage of the general network in cluster systems. For example, Nagios is one of the best tools which is used to visualize almost everything related to the network from devices to protocols. However, Nagios uses a different approach than daemons, and they are not using low-level coding languages. Thus, it is required to have the daemon to monitor the network and collect its data in the central server.

Finally, managing all those mentioned daemons should happen in the monitoring servers. It happens with other daemons that work only on the servers. Managing this type of incoming data and saving them in local databases requires many jobs. Even separating one of the nodes would be great because the daemons are used in nodes for monitoring and will generate sending tons of data from large scale clusters. Therefore, all of these issues should be counted and analyzed beforehand.

4. Conclusion

As can be seen from all the above topics, the visualization procedure of high-performance computing clusters does not only mean to collect and share data between daemons. It is more than that. While monitoring data, dozens of obstacles could arise.

With regards to creating a unique visualization tool for monitoring process flow, data flow, and hardware usage monitoring at the same time, neither easy nor impossible. After all, it is possible to use the experience of monitoring tools which was made beforehand. There are tools for monitoring hardware, and also there are tools for visualization of process, job states. Hence, there are already written daemons for all of the desired features for the unique monitoring tool. It is possible to make a unique tool with those daemons (or some new ones could be prepared by purpose). It will be the long term and full of hardships period to achieve the desired result.

Besides, collecting data from multiple sources using and saving them for future use is another primary task that should be challenged in this imaginary visualization tool. Fortunately, today's multiple database tools can cope with tons of real-time data, without having too much trouble.

Finally, all prospects point out the possibility of having one unique tool in the cluster visualization sphere. Surely, this will make the life of cluster administrators and users easier than ever because they use at least 2-3 monitoring tools in their system to achieve similar results, which may be provided with only one tool.

References

Adhianto, L., Banerjee, S., Fagan, M., et al. (2010). HPCToolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience, 22*(6), 685-701.

Bautista, E., Romanus, M., Davis, T., Whitney, C., & Kubaska, T. (2019, August). Collecting, monitoring, and analyzing facility and systems data at the national energy research scientific computing center. In *Proceedings of the 48th International Conference on Parallel Processing: Workshops* (pp. 1-9).

Burgess, C. (2005). The Nagios Book. http://www.xmarks.com/site/www.nagiosbook.org/PRERELEASE_The_Nagios_Book.pdf (07 November 2012)

Massie, M. L., Chun, B. N., & Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing, 30(7)*, 817-840.

Massie, M., Li, B., Nicholes, B., Vuksan, V., et al. (2012). *Monitoring with Ganglia: tracking dynamic host and application metrics at scale.* " O'Reilly Media, Inc.".

Montaldo, D., Mocskos, E., & Slezak, D. F. (2009) Clover: Efficient Monitoring of HPC Clusters.

Sacerdoti, F. D., Katz, M. J., Massie, M. L., & Culler, D. E. (2003, December). Wide area cluster monitoring with ganglia. In *2003 Proceedings IEEE International Conference on Cluster Computing* (p. 289). IEEE

Sukhija, N., & Bautista, E. (2019, August). Towards a Framework for Monitoring and Analyzing High Performance Computing Environments Using Kubernetes and Prometheus. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (pp. 257-262). IEEE.

Yoo, A. B., Jette, M. A., & Grondona, M. (2003, June). Slurm: Simple linux utility for resource management. In *Workshop on Job Scheduling Strategies for Parallel Processing* (pp. 44-60). Springer, Berlin, Heidelberg.

Submitted 22.01.2020
Accepted 29.05.2020