The Octotron Approach: Towards Autonomous and Reliable Operation of Supercomputers

Alexander Antonov, Dmitry Nikitenko, Pavel Shvets, Sergey Sobolev, Konstantin Stefanov, Vadim Voevodin, Vladimir Voevodin, Sergey Zhumatiy

Research Computing Center, Lomonosov Moscow State University, Moscow, Russia
{asa,dan,shpavel,sergeys,cstef,vadim,voevodin,serg}@parallel.ru

1 Introduction: Background and Motivation

To guarantee the highest productivity of the supercomputer, its maintenance requires constant monitoring of the performance of all critical components, discovering all types of failures, automatic decision-making in case of a failure, and immediate operator notification about the supercomputer's current status. Until all of these requirements are met, neither the efficient operation of the supercomputer nor the safety of its hardware can be guaranteed. The overall complexity of the maintenance problem grows extremely fast simultaneously with the growth of state-of-the-art supercomputer complexity.

The work to ensure reliable supercomputer operation has been going on for a while, and a broad range of materials and methods has been accumulated [1]. In global practices, the resilience of a supercomputer is primarily viewed in the context of ensuring reliable application execution. Based primarily on checkpointing and message logging [2], these mechanisms aim to provide a transparent application resiliency and do not address components safety and notification issues. Maintaining reliable operation, monitoring and self-control of a supercomputer can be done using proprietary solutions: HP has OpenView, IBM has xCAT extensions, T-Platforms has Clustrx Safe, etc. This way is costly and vendor-dependent. Iaso by NUDT [3] is very promising but not available publicly. So a popular solution is to install and configure one of the freely distributed monitoring system (Ganglia, Zabbix, Nagios, …) accompanied by a set of self-written scripts to respond to specific subsets of potentially dangerous situations. These solutions are quite installation-dependent: being fine-tuned to the specific computer, they cannot be transferred easily onto another computer and hardly provide continuity in maintaining a set of supercomputers.

These problems leaded us to develop a system named Octotron with the following key requirements: to discover all types of failure in supercomputers as well as their root causes and relationships; to react automatically to any failure; to accumulate and improve supercomputers maintaining experience by proposing a unified methods for failure and reaction description; to be hardware and software independent and highly scalable. Considering these requirements, an architecture was proposed for the Octotron system based on a formal model of supercomputer operation.

2 The Octotron System: Model-based Approach

Octotron represents the supercomputer model in the form of a graph (Fig. 1). Vertices in the graph correspond to physical or logical components of the supercomputer that need to be monitored: computing nodes, UPS modules, job queues, software components, licenses, etc. The graph edges correspond to the relationships between components, e.g., "consists of," "provides power to," "connected with Infiniband." Each vertex in the graph is associated with a set of attributes which describe component's characteristics: processor temperature, amount of memory, number of jobs in a queue, etc. The Octotron system updates attribute values through the supercomputer's own monitoring systems or directly via external interfaces on the components.

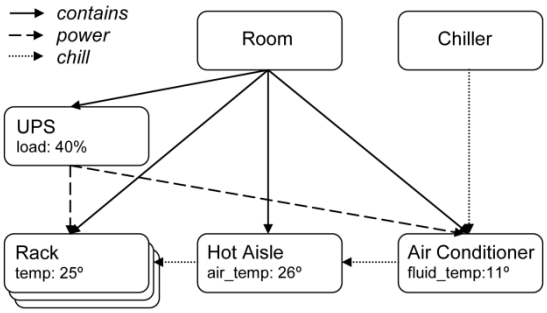

Fig. 1. General idea of the Octotron supercomputer model:components as vertices, relationships as edges, component's characteristics as attributes

The model is accompanied by a set of rules and reactions. The rules are required to discover a failure of the supercomputer's component, and the reactions describe what actions need to be taken once a rule is triggered. For instance, a rule which compares CPU temperature of computing node to a specified threshold may set "overheat_alarm" attribute of the corresponding vertex. After that, a reaction which depends on this attribute will be triggered for powering off the node and administrator notification.

Model as a graph is stored in the Neo4j database. Python language is used to describe the model. The special API was created that simplifies definition of model objects and links. Rules and reactions are described as Python functions, they are able to access, set and modify attributes. Octotron itself is written in Java and works together with Jython interpreter for Python integration. Octotron is not tied to any specific monitoring system. It can work with any source through simple import modules.

As it was stated in the requirements, model makes it possible to control all of the supercomputer's components and relations between them with unified principles. Component and failure description in formal way improves the maintenance experience accumulation and sharing. In case of a component failure, only necessary, linked or dependent components could be selected for reaction triggering; this minimizes the failure impact on supercomputer in general. All these features are already implemented in the current Octotron version.

Lately Octotron will be expanded to track relations between components and to analyze root causes of failures. Octotron alarm statistics will be used for failure prediction.

3 Evaluation of Octotron: MSU Experience

The Octotron system is currently used to control “Lomonosov” and “Chebyshev” supercomputers in Moscow State University. Their models reflect power supply system, cooling system, management components, computing components, shared file systems, and networks. Relationships reflected are “contains,” “chill,” “connected via Ethernet,” “connected via service network,” “connected via Infiniband,” “includes”, “provides power to,” etc. Concerning model scale, for example, the graph for the “Chebyshev” supercomputer (60 TFlops, 5,000 CPU cores) contains 10,228 vertices, 24,698 edges, and 205,044 attributes.

The main supplier of operational data for supercomputers’ components is the monitoring system based on collectd. The hardware infrastructure provides data via SNMP and modbus. Other health data sources include FLEXlm license manager, batch systems, file systems monitoring, etc.

About 160 rules are used to control the operation of MSU supercomputers. Some of them are: failures in the operation of two or three chillers; substantial increase in the error rate on network interfaces; number of user sessions at the host is below threshold; number of blocked nodes is above threshold; time is out of sync on the nodes; load average on a node without user jobs exceeds threshold; modes of two network-connected ports do not match; GSM modem account balance is close to the deactivation limit. In reality, the most frequent cases detected by Octotron are load average exceeding on nodes, short-time CPU overheat, Infiniband and Ethernet ports and nodes disabling, weak load of some supercomputers’ partitions in short time intervals.

Octotron is available under an open MIT license [4].

Acknowledgements. This material is based upon work supported by the Ministry of Education and Science of the Russian Federation (Agreement N14.607.21.0006, unique identifier RFMEFI60714X0006).

References

1. Snir, M, et al.: Addressing Failures in Exascale Computing. International Journal of High Performance Computing Applications (2014.).
2. Cappello F., Geist F., Gropp W., Kale S., Kramer B., Snir M.: Toward Exascale Resilience: 2014 update. Supercomputing Frontiers and Innovations, Vol 1, No 1 (2014), 5-28.
3. Kai LU,Xiaoping WANG,Gen LI, et al. Iaso: an autonomous fault-tolerant management system for supercomputers[J]. Front. Comput. Sci., 2014, 8(3): 378-390.
4. Octotron framework: modeling and monitoring of complex computer systems, https://github.com/srcc-msu/octotron