The Octotron Approach:
Towards Autonomous and Reliable Operation of Supercomputers

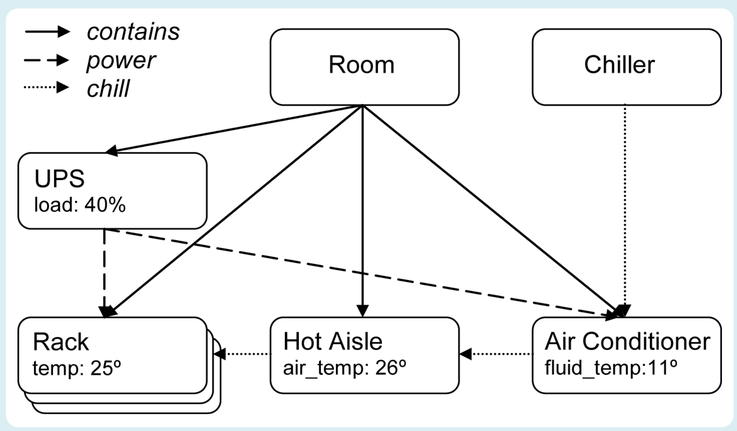
Alexander Antonov, Dmitry Nikitenko, Pavel Shvets, Sergey Sobolev,
Konstantin Stefanov, Vadim Voevodin, Vladimir Voevodin, Sergey Zhumatiy
{asa,dan,shpavel,sergeys,cstef,vadim,voevodin,serg}@parallel.ru

Research Computing Center, M.V.Lomonosov Moscow State University, Moscow, Russia

Project Goals

- to discover **all types of failures** 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**.

Main Idea of the Octotron System



Octotron represents the supercomputer model in the form of a **graph**:

- **vertices** – physical or logical components:
  compute nodes, UPS modules, job queues, software components, licenses …

- **edges** – relationships between components:
  “consists of”, “provides power to”, “connected with Infiniband” …

- each vertex has a set of **attributes**:
  processor temperature, amount of memory, number of jobs in a queue …
  - **trigger** is a special attribute showing that a failure occurred.

The model includes **rules** and **reactions**:
- **rules** – calculate attributes based on other attributes;
- **reactions** – what to do if a trigger is on.

Model is described in Python language using a special adapter module for Jython interpreter.

Model-based Approach Benefits

1. Model makes it possible to control all of the supercomputer’s components and relations between them with unified principles [done];
2. Component and failure description in the formal way improves the maintenance experience accumulation and sharing [done];
3. 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 [done];
4. Tracking relations between components failures and analyzing root causes of failures [in progress];
5. Failure prediction based on Octotron’s alarm statistics [in progress].

Octotron Evaluation at MSU

The Octotron system is currently used to control **“Lomonosov” and “Chebyshev”** supercomputers at MSU. Their **models** reflect:
- power supply system;
- cooling system;
- management components;
- computing components;
- shared file systems;
- networks.

Supercomputers **health data sources** we use include:
collectd, SLURM, SNMP (poll + traps), Lustre, modbus …

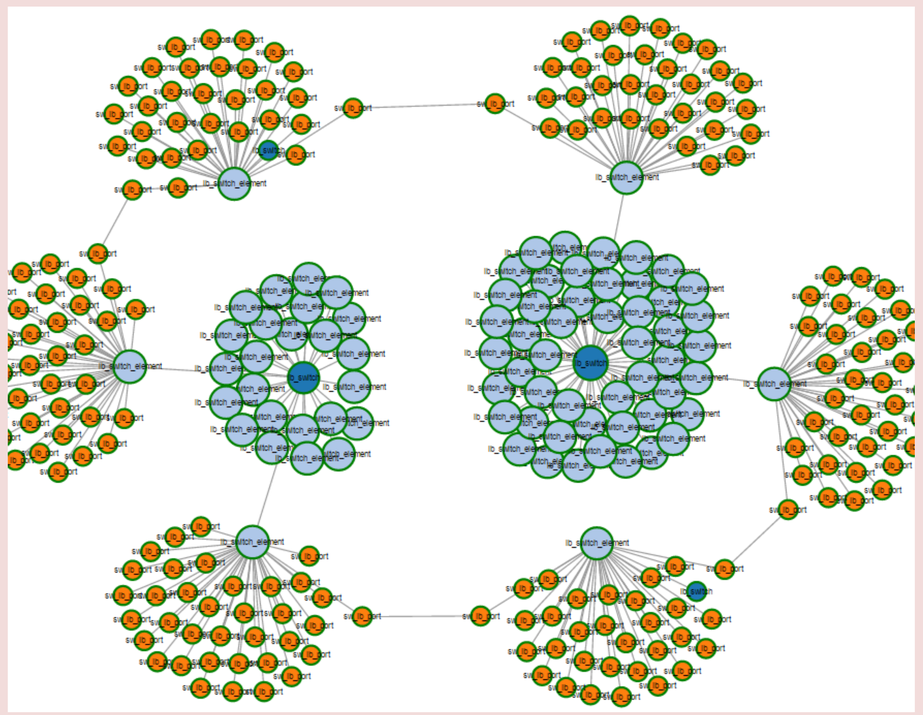
Examples of **failures to detect** (i.e. Octotron’s rules) are:
- errors in the operation of two or three chillers;
- substantial increase of the error rate on network interfaces;
- number of user sessions at the host is below threshold;
- number of blocked nodes is above threshold;
- time is out of 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;
- **… more than 160 rules!**

Reactions to failures are:
- writing to log;
- SMS/e-mail notification;
- powering off the components;
- moving nodes out of computations.

In reality, **the most frequent cases** detected by Octotron are:
- load average exceeding on nodes;
- short-time CPU overheat;
- Infiniband and Ethernet ports disabling;
- unavailable nodes;
- low usage of some supercomputers’ partitions/queues in short time intervals.

Visual Model Verification

Visual model verification is especially valuable for complex models like InifniBand interconnect.



Octotron is available under an open MIT license:
https://github.com/srcc-msu/octotron

This research is supported by the Ministry of Education and Science of the Russian Federation (Agreement N14.607.21.0006, unique identifier RFMEFI60714X0006)