Network and Storage Failure Tolerance on High Performance Computing System Using Application Migration Approach

Husen Rusdiansyah and Heru Suhartanto
Universitas Indonesia

Introduction

The Use of High-Performance Computing (HPC) in running resource-hungry applications is increasing. Weather forecasting, drug discovery and development aircraft design are examples of resource-hungry applications that has an important role for life. Clusfter HPC consists of many nodes used to run applications in parallel. Obstacle encountered in HPC environments, there is a possibility of failure in a node which cause the termination of the process. Such failures may be caused by the storage failure or network failure in a node. This work examines the migration of an application process on a node that is identified will have storage failure or network failure into another nodes so that applications can be run to completion.

The case studies used in this work is molecular simulations using Gromacs. Storage failure identification is performed by Nearest-Neighbor Classifier using Self-Monitoring, Analysis and Reporting Technology (SMART) dataset while the network failure is identified by the Ping command. Process migration of molecular simulations is conducted using checkpoint/restart technique available in Gromacs. The experimental results show the process migration of molecular simulation can be done with the same output compared to normal simulation results.

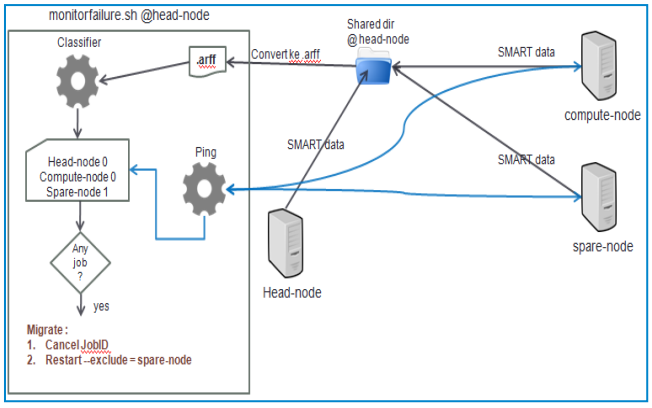


Figure. 2. How it Works

How it works ?

Figure 2 describes how the system works. A network failure identification, conversion of SMART data to .arff format and storage failure identification are controlled by head-node through a bash script called monitorfailure.sh whereas SMART data acquisition is performed in each nodes (head-node, compute-node, spare-node).

As can be seen in Figure 2, each nodes in a cluster stores SMART data in shared directory located in head-node periodically. Prior to be processed by nearest-neighbor classifier, monitorfailure.sh script converts SMART data to .arff format that recognized by WEKA. Once the conversion is completed, nearest-neighbor classifier uses .arff format to identifies storage failure of each nodes. Every nodes with value 0 (zero) identified as a node having good storage condition whereas every nodes with value 1 (one) identified as failed node. Once the storage failure identification is completed, the monitorfailure.sh then execute network failure identifier. Based on the output of network failure identifier, if there is a non-responsive node then monitorfailure.sh will updates the value of that node in classifyresult.txt.

Architecture

Figure 1 is the architecture of a system. It describes communication occurred between nodes in a cluster in general. It also describes all parts involved in each nodes when application is running in a cluster.

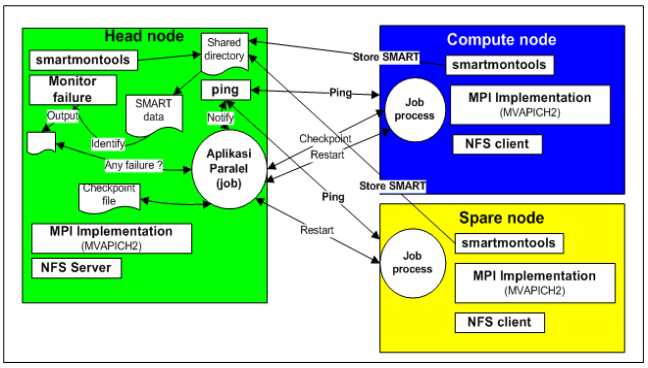
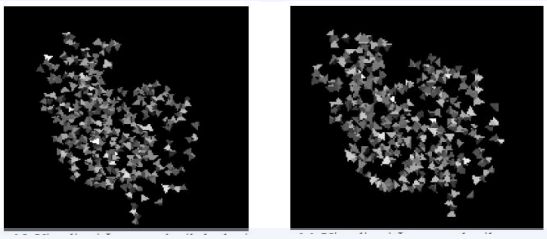


Figure. 1. System Architecture

As can be seen in Figure 1, the core of the system is located in head-node. The parallel application is executed in a cluster using SLURM submit job script supported by DMTCP package with specified checkpoint interval. A shared directory, as the name implies, is a directory located in head-node that shared using NFS across all nodes in a cluster. It stores all application files including SMART data and checkpoint files. Monitor failure is a crucial part in head-node. It is a bash script that executed periodically to identify storage failure based on SMART data, identify network failure using PING command and restart jobs, if any, based on checkpoint files with new nodes allocation. Besides its crucial role in a cluster, head-node together with other nodes in a cluster stores SMART data in shared directory periodically.



(a) (b)

Figure. 3. (a). Visualized Lysozyme in normal execution, **(b).** Visualized Lysozyme with migration process

Table. 1. Cluster Specifications

Processor	Intel core i7-2600 3.40 GHz (head)	Intel core i7-2600 3.40 GHz (compute)	Intel core i7-2600 3.40 GHz (spare)
#core	4	4	4
#Thread	8	8	8
RAM	6 GB	6 GB	6 GB
Hard Disk	1 TB	1 TB	1 TB
Software	• Gromacs 5.1.2 • Slurm-15.08.10 • MVAPICH2-2.2b • NFS • DMTCP-2.4.4 • SSH client&server • Mysql server 5.5.49 • WEKA3-7-13 • smartmontools	• Gromacs 5.1.2 • Slurm-15.08.10 • MVAPICH2-2.2b • NFS • DMTCP-2.4.4 • SSH client&server • smartmontools	• Gromacs 5.1.2 • Slurm-15.08.10 • MVAPICH2-2.2b • NFS • DMTCP-2.4.4 • SSH client&server • smartmontools

Conclusion

This poster presents our cluster system to tolerate network failure and storage failure through application migration using checkpoint/restart technique available in DMTCP and Gromacs.