

# REM-Rocks: A Runtime Environment Migration Scheme for Rocks based Linux HPC Clusters

Tong Liu, Saeed Iqbal, Yung-Chin Fang, Onur Celebioglu, Victor Masheyakhi and Reza Rooholamini

Dell Inc.

{Tong\_Liu, Saeed\_Iqbal, Yung-Chin\_Fang, Onur\_Celebioglu, Victor\_Masheyakhi, Reza\_Rooholamini}@dell.com

Chokchai (Box) Leangsuksun

Louisiana Tech University  
box@latech.edu

**Abstract.** Commodity Beowulf clusters are now an established parallel and distributed computing paradigm due to their attractive price/performance. Beowulf clusters are increasingly adopted in downtime-sensitive environments requiring improved fault tolerance and high availability (HA). From the fault tolerance prospect, the traditional Beowulf cluster architecture consists of a single master node which potentially renders a single point of failure (SPOF) in the system. Hence to meet the minimal downtime requirements, HA enhancements to the cluster management system are critical. In this paper we propose such enhancements based on the commonly used Rocks management system, called Run Environment Migration Rocks (REM-Rocks). Our previous experiences [4][14] in HA-OSCAR release suggests a significant HA improvement. REM-Rocks is resilient to multitude levels of failures and provides mechanisms for graceful recovery to a standby master node. We also discuss the architecture and failover algorithm of REM-Rocks. Finally, we evaluate failover time under REM-Rocks.

## 1. Introduction

In the past decade, the raw computational power of commodity Beowulf clusters [1] has dramatically increased. The trend has propelled clusters as the architecture of choice for parallel and distributed computing in academic, research and industrial environments. The availability of mature and stable open source operating systems, such as Linux, have also helped this trend immensely. Currently, there are several open source high performance Linux based clustering solutions available, Rocks [2] and OSCAR [3] are those among the most popular open source tools to build clusters.

The contemporary Beowulf cluster architecture has a single master node which creates a single point of failure (SPOF) [4]. In an event of a master node failure, the whole cluster will be out of service. This scenario is unacceptable for many mission critical applications which guarantee their users certain quality of service (QoS). The

2 Tong Liu, Saeed Iqbal, Yung-Chin Fang, Onur Celebioglu, Victor Masheyakhi, Reza Rooholamini and Box Leangsuksun

SPOF is one of the main impediments to clusters being commonly deployed for applications requiring high availability. Fault tolerance improvement in Beowulf clusters is urgently required.

High Availability (HA) [6] implies that system resources and services must be operational with high probability. HA in servers can be achieved by adding redundant hardware subsystems. However, the servers are much more expensive. An economical solution is to use redundant machines made of commodity components with special software to transfer control between machines. Such techniques are gaining popularity and are in demand nowadays. The latter approach can be employed to eliminate the SPOF in clusters at a much lower cost. Furthermore, this class of HA solutions improve clusters survivability from failures in the system by triggering failover [4] to redundant nodes.

In this paper we developed and evaluated an economical high availability enhancement called "REM-ROCKS" to the traditional Beowulf architecture. Our early lessons learnt from dual-headed HA-OSCAR demonstrate a cost-effective solution. REM-Rocks enables two master nodes, called *primary master node* and *standby master node*. REM-Rocks executes a failure detection module on the standby node which can detect failures on the primary master node quickly and immediately initiate migration of services from the primary master node to the standby node.

This paper is organized as follows: Section 1 gives an introduction. Section 2 gives further details about the SPOF and other relevant background. Section 3 gives details of the REM-Rocks architecture and failure detection algorithm. Section 4 gives details of the hardware setup and implementation of the algorithm. Section 5 outlines some related projects and future enhancements. Section 6 gives a summary of the paper.

## 2. Backgrounds: SPOF in Rocks Beowulf Cluster

Rocks, developed by San Diego Supercomputer Center, is an open source package which has been widely utilized to build Beowulf cluster varying from a few to several hundred heterogeneous compute nodes. Installing Rocks is a simple process which requires minimal expertise in Linux system administration or cluster architecture. After providing simple cluster configuration data on front-end node, all the compute nodes start installation automatically. To allow adding software components more easily after initial cluster deployment, Rocks provides a roll CD scheme which customizes them to meet software distribution requirements. Four clusters have been reported built by Rocks in the Top500 list in November 2004 [5].

Figure 1 shows the architecture of a traditional ROCKS Beowulf cluster. The main components of the cluster are a single master node, compute nodes, communication network switch and storage disks. The master node controls all job assignments to compute nodes. The master node receives requests from users, through submit

nodes, and distributes jobs to the specific compute nodes based on the decision created by a job scheduler and resource manager. In the event of a failure at the master node due to outages such as defected hard-drives or failed services, the master node is unable to perform its operations like scheduling and communication with the public network. For example, if the public network interface fails at the master node, job requests cannot be distributed due to a nonfunctional job scheduler daemon. In addition, running jobs may also crash. Such failures usually render catastrophic events and the whole cluster is out of service. Thus, a normal root cause of such down times is the single point of failure (SPOF) of the master node.

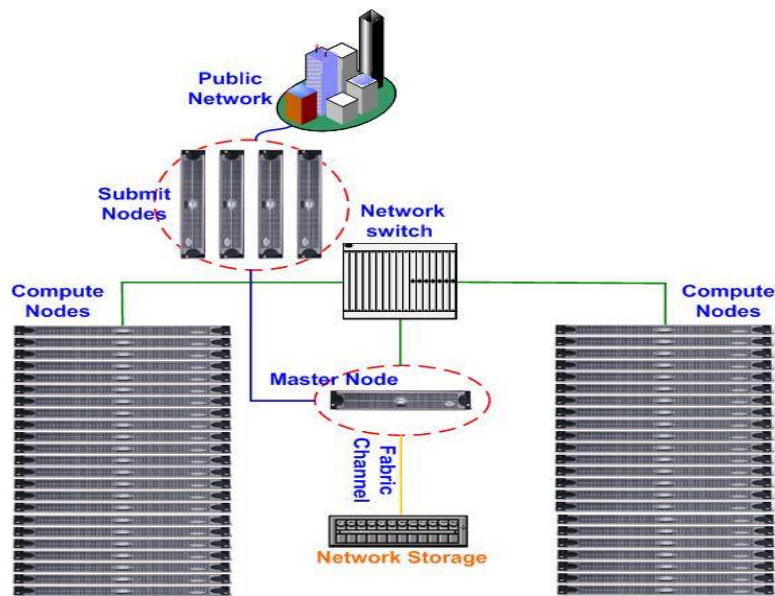


Fig. 1. Architecture of Rocks Beowulf Cluster

### 3. REM-Rocks: Architecture and Fault Tolerant Algorithm

In REM-Rocks similar to HA-OSCAR, the SPOF is eliminated by adding redundant master node. The multiple master nodes with proper software support ensure that in case of master node failure all of its functionality can be taken over by a standby master node. Figure 2 shows the REM-Rocks architecture with a *primary master node* and a *standby master node*. Both master nodes have network connections to the same private and public networks. In addition, they have access to the same network attached storage, so that either can take control of the cluster, after a failover or failback [8]. Either master node can provide us the same application data and environment settings which make system failure recovery almost transparent to users. In REM-Rocks, the standby master node is designed to monitor health of primary

4 Tong Liu, Saeed Iqbal, Yung-Chin Fang, Onur Celebioglu, Victor Masheyakhi, Reza Rooholamini and Box Leangsuksun

master node at all time. If the standby master node detects a failure at the primary master node, it triggers a failover operation. Furthermore, dual master nodes not only improve system availability but also help cluster maximize its performance by load balancing. The primary master node in a Beowulf cluster can become overloaded since many applications and daemons used to distribute jobs and manage cluster are executing on it. Introducing the secondary server can share some of loads with the primary master node. In our architecture, standby master node works as code compiling node and cluster monitoring node to take some load from primary master node. As an important feature, REM-Rocks support High Availability Network File System to ensure that both master nodes can access the same application data after failover or failback. Hence, no jobs resubmission is required after master node is swapped.

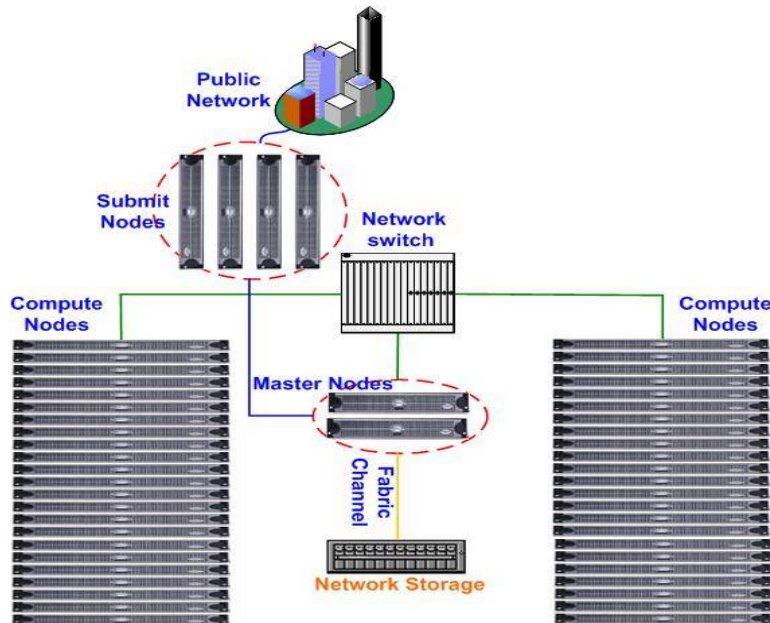


Fig. 2. Architecture of REM-Rocks Beowulf Cluster

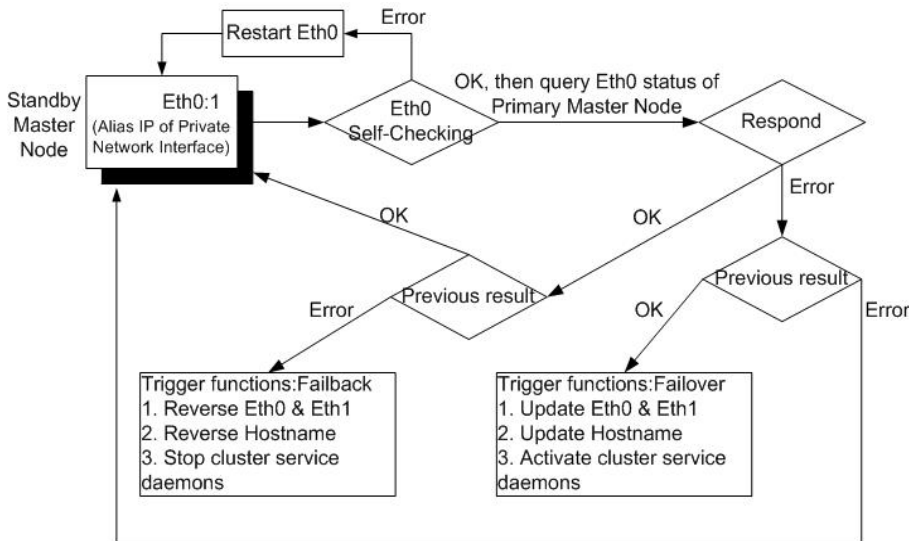
### 3.1. Failure Detection, Failover and Failback Algorithm

A key issue is an accurate failure detection that aims to minimize false failure detection. Existing failure detection algorithms like the Linux-HA [9], Kimberlite [10] and Linux FailSafe [11] have a tendency to report false failures. We have improved these detection algorithms and developed a scheme which can minimize or avoid false failure detections (The details are given below). In order to handle various kinds of failures that might occur at the master node, we employ a multi-level detection algorithm. To detect failures, a REM-Rocks daemon monitors a predefined list of services and devices. Any failure detected in these components is reported to the REM-Rocks management module. The management module triggers an

appropriate alert operation. To further understand the operation of our algorithm, let us consider the following example scenarios:

**Scenario 1: Network Failure**

Compute nodes in a Beowulf Cluster frequently communicate with the master node via the network. Obviously, if master node’s network goes down, all users will be unable to access the cluster and all current jobs will fail. Figure 3 shows a flow chart of our failover algorithm when a failure at the master node.



**Fig. 3.** Failover and failback algorithm based on network failure

The REM-Rocks monitor daemon on standby master node periodically sends a health query message out and it assumes a failure happened on the network interface of primary master node if a successful response is not returned within a time limit. However, no response received can be due to a bad internal network interface on standby master node. To avoid this false alarm, we add a local network interface self-checking function prior to all operations. If the local self-checking fails and reports an error, the REM-Rocks management module will restart the local network interface. All actions are stopped until a successful local self-check is received. This will prevent an unnecessary failover operation from executing triggered by local failure.

Once an actual failure occurs, our solution will first check with the previous result. If the preceding return value is “OK”, that indicates a failure at the primary master node. REM-Rocks will start a failover function to convert critical network configurations on standby master node to primary master node’s settings and start cluster services accordingly. On the contrary, if the previous result is “Error”, REM-Rocks will not perform failover since it must have been already executed.

6 Tong Liu, Saeed Iqbal, Yung-Chin Fang, Onur Celebioglu, Victor Masheyakhi, Reza Rooholamini and Box Leangsuksun

Meanwhile, if the standby master node receives successful reply from the primary master node, REM-Rocks will generate return value as “OK”. This value has to be used to compare with the previous return value similar to what we stated in last paragraph. In the event of “OK” received right after “Error”, REM-Rocks will perform a failback operation on the standby master node and then change network configurations back to the original value and stop running cluster services. At this time, the primary master node takes back all its responsibilities for the whole cluster. If “OK” comes after “OK”, REM-Rocks will skip over and trigger no operation.

### Scenario 2: Cluster Service Failure

To enable the master node to survive a resource failure, we implement a cluster service failure detection module deployed on the primary master node. Its flow is illustrated in Figure 4. In the primary master node, the detection module keeps checking status of demons of the cluster programs and services selected by the administrator. If a failure response is returned, REM-Rocks will refresh the failed daemon by restarting. This feature furthers improves cluster fault resilience. If REM-Rocks is unable to restart the module, it will deactivate all the network interfaces and the cluster running services on primary master node. This will in turn prevent a response from the stopped services to the standby master node. Hence, the standby master node will initiate a failover operation occurring on the standby master node in scenario 1.

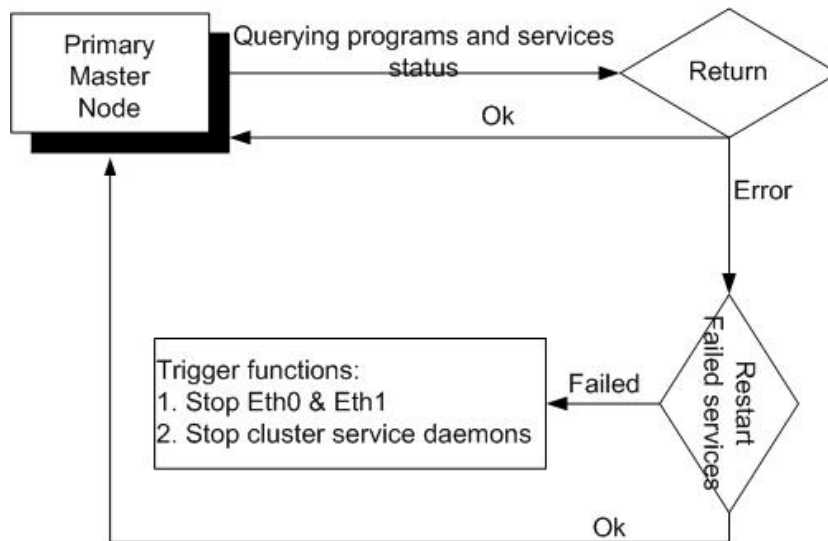


Fig. 4. Cluster service failure detection algorithm

## **4. Implementation and Experimental Evaluation**

### **4.1. Hardware Setup**

Our testing environment is based on 6 Dell PowerEdge PE2650s (two master nodes and four compute nodes) with the following specification on each: Each node has two IA32 based Intel Xeon processors running at 3.2 GHz (1 MB L2 cache, 533 MHz front side bus (FSB)) with 4GB memory (266MHz DDR SDRAM). Each node has two integrated Intel Gigabit Ethernet adapters. Hyper Threading was turned off during all experiments.

### **4.2. Software Installation**

First, we built our testing cluster with ROCKS 3.3. After system initialization, we installed REM-Rocks package and duplicated the original master node. One was chosen as the primary master node and the other was the standby master node. The primary master node held all job management responsibilities and the standby master node was set to monitor the running status of all the nodes. The Sun Grid Engine (SGE) is used as a resource manager.

### **4.3. Approach: Simulate Failure Scenarios**

To evaluate performance of the failover algorithm, our approach is to simulate failures on the experimental setup and measuring failure over time. Consider the following simulate failures:

#### **4.3.1. Simulated Network Failure**

In this scenario, we performed two test cases. First, we directly unplugged the network cable from the primary master node's private network port, so that the standby master node lost communication within the detection interval. Once the unsuccessful communication was reported, REM-Rocks's management module will trigger a failover operation immediately. Consequently, the standby master node took the control of the cluster network file system and all the compute nodes and resumed providing access for external users. Figure 5 shows time measurements of the corresponding failover operation.

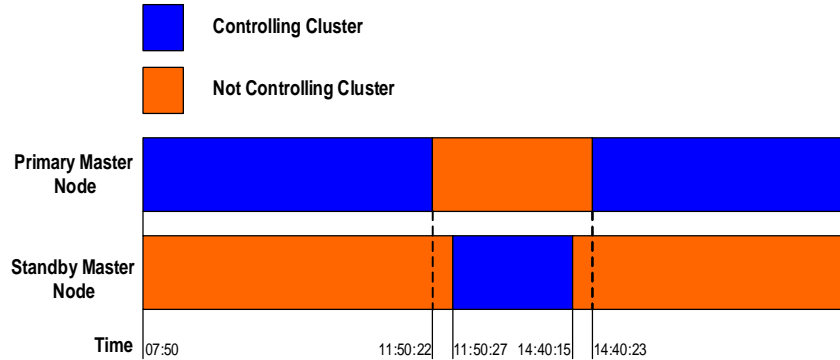


Fig. 5. Timing of failover due to the simulated network failure

We started this test from 7:50 am. Initially the primary master node worked correctly. At time 11:50:22, we pulled out the network cable from eth0 on the primary master node. After 5 seconds, the standby master node was functioning as the frontend node for our test cluster. At time 14:40:15, we plugged network cable back to eth0 of the primary master node which would resume working within 8 seconds at 14:40:23. This experimental result suggests that REM-Rocks can reduce system downtime from hours to a few seconds.

In order to verify the self-checking function of REM-Rocks, we manually disabled a network interface on the standby master node. Shortly, that network interface was successfully reactivated since REM-Rocks performed the self-checking function and activated it without producing an improper failover operation

#### 4.3.2. Simulated Cluster Service Failure

To simulate this failure, we planned to monitor a job scheduler management tool, SGE (Sun Grid Engine) [13]. First we added names of SGE server daemons, *sge\_qmaster* and *sge\_commd*, to the configuration file of REM-Rocks resource monitoring module. Then we run command: *service rcsge stop* to shut down SGE scheduler. After several seconds, we checked SGE running status and found the server daemons, *sge\_qmaster* and *sge\_commd*, resumed working since REM-Rocks restarted SGE scheduler automatically right after the failure was detected.

In addition to resumable cluster services, we conducted another test case for verifying REM-Rocks failover function based on an unrecoverable resource failure. In this scenario, we deleted all SGE files so that REM-Rocks reported an error after an attempt to restart *sge\_qmaster* and *sge\_commd*. Followed by the error, eth0 and eth1 on the primary master node were disabled which caused the standby master node to trigger a failover operation subsequently. When we restored all those files back to the same directory on the primary master node, REM-Rocks successfully started the SGE daemons and activate eth0 and eth1 which in turn make the standby master node rebuilt connection with the primary master node and return the system control back.

## 5. Related and future work

There are several High Availability Linux Cluster related works which are dedicated to providing failover mechanisms for common applications such as Samba, Apache, Databases, etc.

Kimberlite is an open source cluster technology developed by Mission Critical Linux. It provides data integrity and application availability which is suitable for NFS servers, web servers and database applications. SteelEye's LifeKeeper for Linux is a software solution that allows applications to failover to other servers in the cluster. Similar to Kimberlite, it only provides application availability for common web servers and database clusters. Linux-HA has a widely used package called 'Heartbeat', which performs a failover by using an alias IP address takeover on Linux systems. Its default configuration is applicable only for 2-node clusters and supports web servers, Mail servers, File servers, Database servers, DHCP servers, etc. However, all these solutions may not be suitable to the large-scale Linux Beowulf cluster.

HA-OSCAR [4] is an open source software solution constructed based on OSCAR package which provides system, service monitoring and alert management capability for Linux Beowulf cluster systems. It runs MON [7] on one master node to check the other master node health periodically. If there is no response for service access request or ICMP ECHO\_REQUEST from a node in a specified time, its service monitor will consider the node is dead and report the failure to its alert management module. Consequently, user customized operations will be triggered, such as failover or sending email to the system administrator. When the outage node is recovered, the alert management module will get notified and trigger failback operation accordingly.

In order to make a Linux Beowulf cluster a fully fault-tolerant system, we are considering an automatic job checkpointing and restarting mechanism to enable a transparent system recovery when a failover takes place on the master node. In addition, building active-active master nodes architecture is also in demand. If two or more master nodes can handle user requests, the cluster performance will be further improved.

## 6. Conclusion

We have developed and evaluated an economical HA architecture for Beowulf clusters. REM-Rocks can be used to improve the availability of the traditional Beowulf Cluster. Normally, a system outage occurs when there are failures on the master node. However, our experiments indicate that REM-Rocks ensures a successful failover operation to enable the standby master node functioned as the master node. Our solution improves cluster's total up time and lessen the load on the single master node of the traditional Beowulf cluster.

10 Tong Liu, Saeed Iqbal, Yung-Chin Fang, Onur Celebioglu, Victor Masheyakhi, Reza Rooholamini and Box Leangsuksun

## 7. References

- [1] Beowulf Cluster, <http://www.beowulf.org>.
- [2] NPACI Rocks, <http://www.rocksclusters.org>.
- [3] OSCAR, <http://oscar.openclustergroup.org>.
- [4] C. Leangsuksun, Lixin Shen, Tong Liu, H. Song, S. Scott, Availability Prediction and Modeling of High Availability OSCAR Cluster, IEEE International Conference on Cluster Computing (Cluster 2003), Hong Kong, December 2-4, 2003.
- [5] <http://www.top500.org>.
- [6] Evan Marcus, Hal Stern, Blueprints for High Availability: Designing Resilient Distributed Systems, *John Wiley & Sons*, 1 edition, Page 15-25, January 31, 2000.
- [7] MON, <http://www.kernel.org/software/mon>.
- [8] Failback, <http://www.faqs.org/docs/evms/x2912.html>.
- [9] Linux-HA, <http://www.linux-ha.org>.
- [10] Kimberlite, <http://oss.missioncriticallinux.com/projects/kimberlite>.
- [11] Failsafe <http://www.sgi.com/products/software/failsafe>.
- [12] PMB, <http://www.pallas.com/e/products/pmb>.
- [13] SGE, <http://gridengine.sunsource.net>.
- [14] HA-OSCAR, <http://xcr.cenit.latech.edu/ha-oscar>