HA-OSCAR Release 1.0: Unleashing HA-Beowulf

Chokchai Leangsuksun[1], Tong Liu[1], Stephen L. Scott[2],
Richard Libby[3], and Ibrahim Haddad[4]

[1]Computer Science Department, Louisiana Tech University, Ruston, LA 71272, USA
[2]Computer Science and Mathematics Division, ORNL, Oak Ridge, TN 37831, USA
[3]Enterprise Platforms Group, Intel Corporation
[4]Ericsson Corporate Unit of Research, 8400 Decarie Blvd, Montreal, QC H4P 2N2, Canada

[1]{box, tli001}@latech.edu
[2]scottsl@ornl.gov
[3]rml@hpc.intel.com
[4]ibrahim.haddad@ericsson.com

Abstract

March 2004 was a major milestone for the HA-OSCAR Working Group. It marked the announcement of the first public release of the HA-OSCAR software package. HA-OSCAR is an Open Source project that aims to provide the combined power of high availability and performance computing. HA-OSCAR enhances a Beowulf cluster system for mission critical grade applications with various high availability mechanisms such as component redundancy to eliminate this single point of failure, self-healing mechanism, failure detection and recovery mechanisms, in addition to supporting automatic failover and fail-back.
The first release (version 1.0) supports new high availability capabilities for Linux Beowulf clusters based on the OSCAR 3.0 release from the Open Cluster Group. In this release of HA-OSCAR, we provide an installation wizard graphical user interface and a web-based administration tool, which allows intuitive creation and configuration of a multi-head Beowulf cluster. In addition, we have included a default set of monitoring services to ensure that critical services, hardware components, and important cluster resources are always available at the control node. This release also featured new services that can be configured and added via a WebMin-based HA-OSCAR administration tool. In addition to a review of the new features of the current HA-OSCAR release, this paper will also discuss our experiments covering performance and availability, as well as our test results.

1. Introduction

High Availability Open Source Cluster Application Resource (HA-OSCAR) [1] is an Open Source project that aims toward non-stop services in the HPC environment through a combined power of High Availability and Performance Computing solutions. The HA-OSCAR project primary goal is to enhance a Beowulf cluster system for mission-critical applications and downtime-sensitive HPC infrastructures. The Open Cluster Group [2] recognized the HA-OSCAR project as an official working group, along with the current OSCAR and Thin-OSCAR working groups. HA-OSCAR introduces several enhancements and new features to OSCAR, mainly in the areas of availability, scalability, and security. The new features in the first official release (version 1.0) include head node redundancy, self-recovery for hardware, in addition to self-recovery from service and application outages.

[1] Research supported by Center for Entrepreneurship and Information Technology, Louisiana Tech University.
[2] Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

This paper provides an overview of the HA-OSCAR architecture and a systematic installation guide for researchers interested in experimenting with HA-OSCAR. Section 2 presents the HA-OSCAR architecture and its main components. Section 3 provides an overview of the first HA-OSCAR release, the supported distributions and different configurations. Section 4 provides a quick tutorial on HA-OSCAR cluster installation. Section 5 describes HA-OSCAR release 1.0 test environment and provides an analysis of performance impact and resource usages by HA-OSCAR failure detection and recovery mechanism. In Section 6, we present a summary and the HA-OSCAR roadmap.

2. HA-OSCAR Architecture

Figure-1 illustrates the architecture of HA-OSCAR. The first release of HA-OSCAR only supported a single network private interface failover by default installation. However, we plan to support dual private network interfaces and failover (Figure-1), in addition to external reliable storage in the second release. However, HA-OSCAR users can manually add and configure more NICs, switches, and external storage via HA-OSCAR Webmin [3].

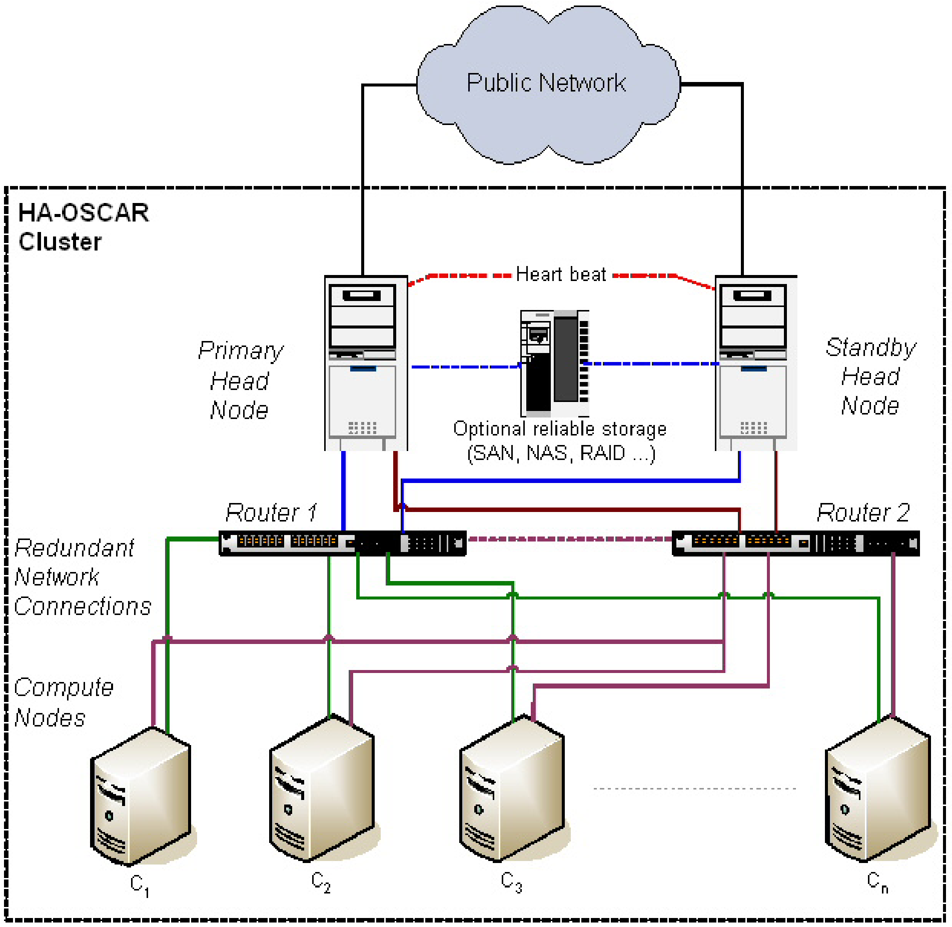


Figure-1: HA-OSCAR Architecture

HA-OSCAR consists of the following major system components:

1. *Primary server*: responsible for receiving and distributing the clients' requests to compute nodes. Each server has three NICs; one is connected to the Internet by a public

network address, and the other two are connected to a private LAN, which consists of a primary Ethernet LAN and a standby LAN for redundancy purposes. However, the first release only supported one private NIC.

2. *Standby primary server*: This server monitors the primary server and anticipates taking over when a failure in the primary server is detected.
3. *Multiple computer nodes*: These machines are dedicated to computation.
4. *Local LAN Switches*: The switches provide local connectivity among head and compute nodes.

We recommend that each head node have at least two NICs, eth0 and eth1. One of the NICs is a public interface to outside network and the other is a private interface to its local LAN and computing nodes. The configuration depends on how a user wants to connect eth0 and eth1 to either the public or the private network. Our illustrations assume eth0 is a private interface connected to a local private network, and eth1 is a public interface connected to a wider network or the Internet. Figure-2 illustrates a sample network configuration of HA-OSCAR head nodes.

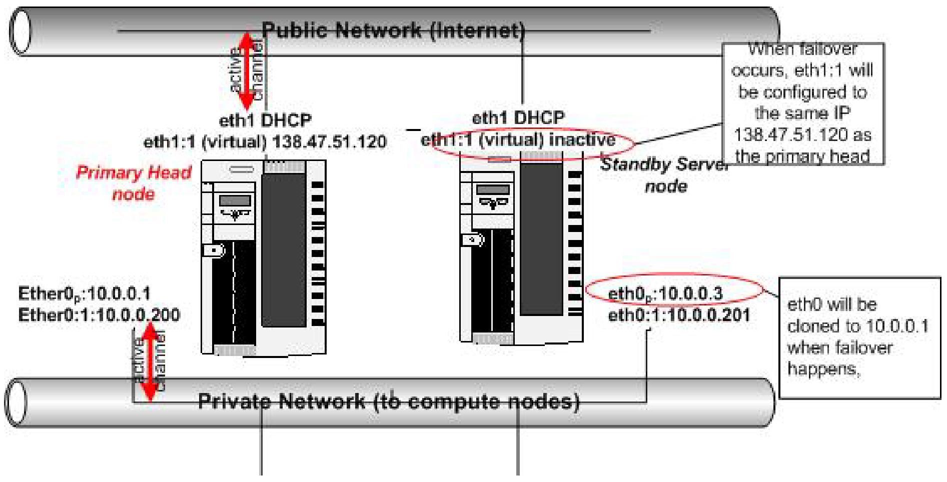

Public Network (Internet)
active channel
eth1 DHCP
eth1:1 (virtual) 138.47.51.120
Primary Head node
eth1 DHCP
eth1:1 (virtual) inactive
Standby Server node
When failover occurs, eth1:1 will be configured to the same IP 138.47.51.120 as the primary head
Ether0p:10.0.0.1
Ether0:1:10.0.0.200
eth0p:10.0.0.3
eth0:1:10.0.0.201
eth0 will be cloned to 10.0.0.1 when failover happens,
active channel
Private Network (to compute nodes)

Figure-2: Sample of HA-OSCAR head node network configuration

3. HA-OSCAR 1.0 beta release

In March 2004, the eXtreme Computing Research (XCR) group at Louisiana Tech University announced the first public release HA-OSCAR 1.0 [1]. HA-OSCAR 1.0 incorporated a self-healing mechanism, failure detection and recovery, automatic failover and fail-back, and support for Active/Hot standby mode for the head nodes.. The result is a cluster system that is able to achieve high availability. Component redundancy is adopted in HA-OSCAR cluster to eliminate single point of failures, especially at the head node. The 1.0 beta release supports new high-availability capabilities for Linux Beowulf clusters based on OSCAR 3.0 [2]. It provides an installation wizard GUI, and a web-based administration tool that allows users to create and configure a multi-head Beowulf cluster. A default set of monitoring services is included to ensure that critical services, hardware components and important resources are always available at the control node. New services can be configured and added via a WebMin-based [3]HA-OSCAR administration tool.

3.1 Supported Distributions

We successfully tested HA-OSCAR with several OSCAR versions: OSCAR 2.3, 2.3.1, and 3.0, based on Red Hat 9.0. Our test environment consisted of the following:

- Two dual Xeon 2.4 GHz head nodes, each has one GB of RAM, a 40 GB hard disk, and two NICs.
- Four dual Xeon 2.4 GHz compute nodes, each comes with 1 GB of RAM, a 40 GB hard disk, and two NICs
- One D-link 10/100 Mbps switch

3.2 System Requirements

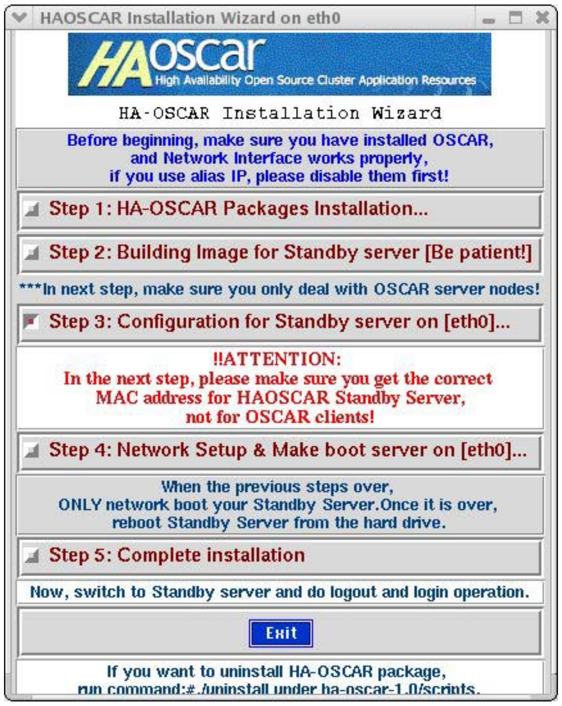
The HA-OSCAR installation is dependent on a cluster built with OSCAR [4]. The primary and standby head nodes should have homogeneous hardware; each head node must be equipped with at least two NICs. The network interface must support PXE boot. The standby server eth0 must be connected to the local switch where the primary server PXE daemon will be listening to the broadcast boot request. We recommended that both primary and standby server eth0s be connected to the local switch. In the upcoming release of HA-OSCAR, we plan to remove this requirement.

4. Detailed Cluster Installation Procedure

To allow easy and seamless cluster installation and management, we have packaged HA-OSCAR in an easy-to-install package, combined with a GUI installation wizard. After downloading HA-OSCAR and uncompressing the package, you can start the installation procedure (as root) by typing the following command:

```
      % ./haoscar_install <interface>
```

The installation wizard will start (Figure-3).


HAOSCAR Installation Wizard on eth0

HA-OSCAR Installation Wizard

Before beginning, make sure you have installed OSCAR,
and Network Interface works properly,
if you use alias IP, please disable them first!

Step 1: HA-OSCAR Packages Installation...

Step 2: Building Image for Standby server [Be patient!]

***In next step, make sure you only deal with OSCAR server nodes!

Step 3: Configuration for Standby server on [eth0]...

!!ATTENTION:
In the next step, please make sure you get the correct
MAC address for HAOSCAR Standby Server,
not for OSCAR clients!

Step 4: Network Setup & Make boot server on [eth0]...

When the previous steps over,
ONLY network boot your Standby Server.Once it is over,
reboot Standby Server from the hard drive.

Step 5: Complete installation

Now, switch to Standby server and do logout and login operation.

Exit

If you want to uninstall HA-OSCAR package,
run command:#./uninstall under ha-oscar-1.0/scripts.

Figure-3: HA-OSCAR Installation Wizard

The interface directive is the private network interface for the primary head. The `<interface>` is normally "`eth0`", since the first release supports only one private network interface. The HA-OSCAR installation wizard will walk the user through a complete installation process that normally consists of five steps:

Step 1 Installing HA-OSCAR package (software staging on the primary head)
Step 2 Building a standby server image (cloning the primary server)
Step 3 Configuring initial standby server
Step 4 Setting up the network and creating a boot image on the standby server
Step 5 Completing the installation

The following sections describe the HA-OSCAR wizard installation process and provide visuals for the associated screens.

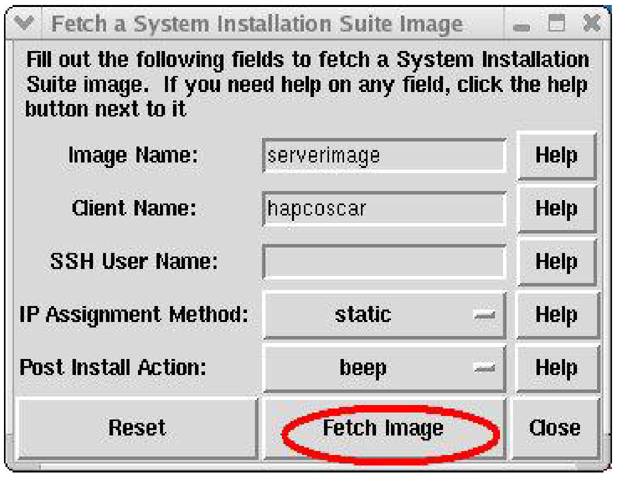
4.1 HA-OSCAR Package Installation

In `<step 1>`, the installation wizard will install all the required packages HA-OSCAR to the OSCAR cluster server, and prepare the environment for the remaining four (4) steps. This step will take less than one minute to complete. [Note: All times are based on timings from our test environment presented in section 3.1.]

4.2 Fetching (Cloning) Image for Standby Server

In <step 2>, the user is ready to build a standby server image from the primary node. When the user clicks the button <Building Image for Standby server>, the wizard pops up a new window in which the user can define a server image name. Normally, the user can leave the default value and just press the <Fetch image> button (Figure-4) to get an image for the standby server. This step will take several minutes.

This is an important step in cloning a standby server image from a primary one. For a stringent downtime requirement, we recommend a separate image server for image repository and recovery purpose.



Fill out the following fields to fetch a System Installation Suite image. If you need help on any field, click the help button next to it

Image Name:	serverimage	Help
Client Name:	hapcoscar	Help
SSH User Name:		Help
IP Assignment Method:	static	Help
Post Install Action:	beep	Help

Figure-4: Fetching/Cloning Server Image

This step will take between 10 and 15 minutes. Once the step finishes, a successful status window will pop up, and then you may click on the `<Close>` button.

4.3 Configuring the Standby Server

<step 3>requires the user to enter an alias public IP address. Once the user proceeds by clicking on <step 3>, HA-OSCAR will pop up `<the Standby server initial network Configuration screen>` shown in Figure-5. Users normally use this public IP address as a virtual entry point to access the head node. When the failover occurs, the standby

server will be assigned to this address so users can continue accessing the cluster as if nothing has happened. The following are normal procedures within this step.

1. Input alias public IP for the interface, i.e. eth1. When the failover occurs, the standby server will automatically clone the cluster public IP on the designated network interface, i.e. eth1.
2. Determine whether HA-OSACAR selected standby local IP address has not been reserved for another node (e.g. 10.0.200.200). If this IP address has been occupied, please select the new one. Otherwise, a user can keep this default IP.
3. The last two items `<Subnet Mask>` and `<Default Gateway>` should not be modified.
4. When ready, Click `<Add Standby Server>`.

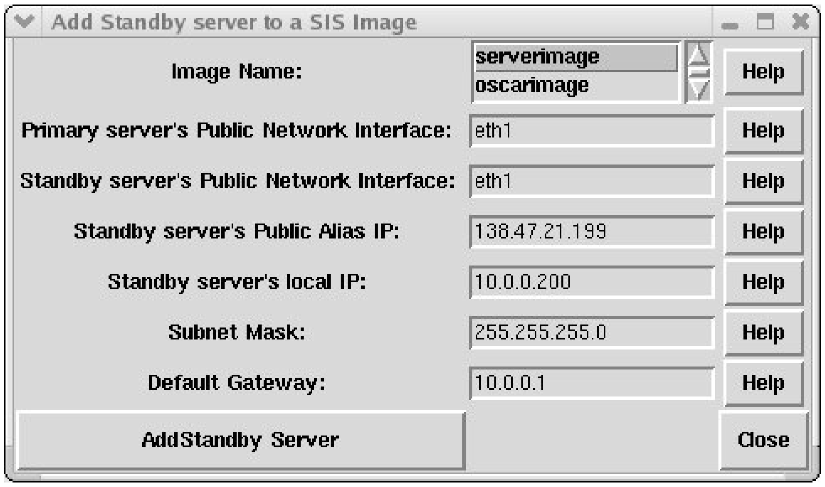

Add Standby server to a SIS Image

Image Name:	serverimage / oscarimage	Help
Primary server's Public Network Interface:	eth1	Help
Standby server's Public Network Interface:	eth1	Help
Standby server's Public Alias IP:	138.47.21.199	Help
Standby server's local IP:	10.0.0.200	Help
Subnet Mask:	255.255.255.0	Help
Default Gateway:	10.0.0.1	Help

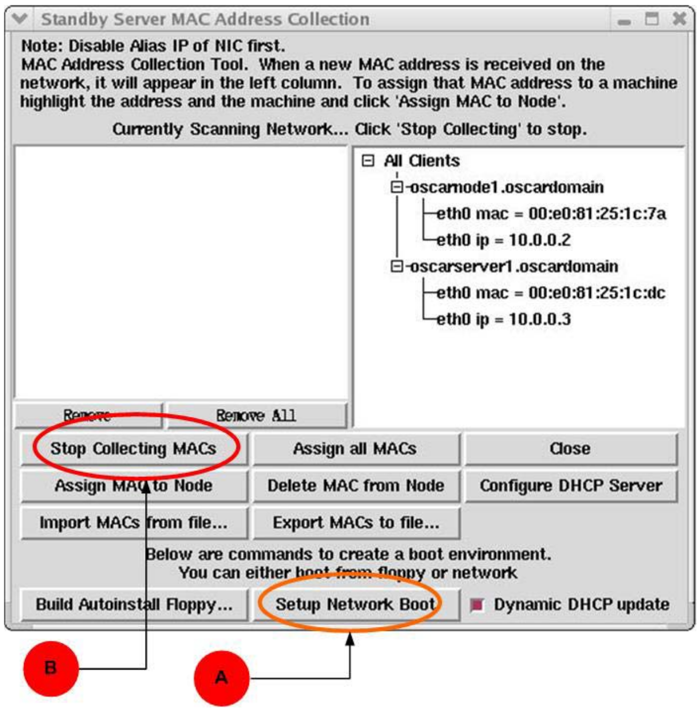
Figure-5: Standby Server Initial Network Configuration

This step will last for less than a minute. Once the successful status window pops up, the user can click on the `<Close>` button.

4.4 Retrieve standby server MAC address (for PXE boot) and build the image on its local drive

In `<step 4>`, the user will retrieve the standby server's MAC address for PXE booting before building its images on the local drive. One of the standby server network interfaces, eth0, is typically connected to the private LAN and reserved for broadcasting its MAC address during its network boot. When the primary server is ready to build the standby server image, it starts cloning its images with the collected addresses. Consequently, the standby server will get the image by network booting the standby server via PXE (or floppy) from the primary server. When the standby server is successfully cloned, the server will be rebooted from its hard disk. This marks a completion of the standby server installation.

We will explain the steps to assign standby server MAC address and build a local image on the standby server. First, the user should proceed to `<step 4>` (Figure-3). HA-OSCAR will display *`<The standby server MAC address configuration screen>`* as shown in Figure-6.



Standby Server MAC Address Collection

Note: Disable Alias IP of NIC first.
MAC Address Collection Tool. When a new MAC address is received on the network, it will appear in the left column. To assign that MAC address to a machine highlight the address and the machine and click 'Assign MAC to Node'.

Currently Scanning Network... Click 'Stop Collecting' to stop.

```
⊟ All Clients
  ⊟-oscarnode1.oscardomain
     ├eth0 mac = 00:e0:81:25:1c:7a
     └eth0 ip = 10.0.0.2
  ⊟-oscarserver1.oscardomain
     ├eth0 mac = 00:e0:81:25:1c:dc
     └eth0 ip = 10.0.0.3
```

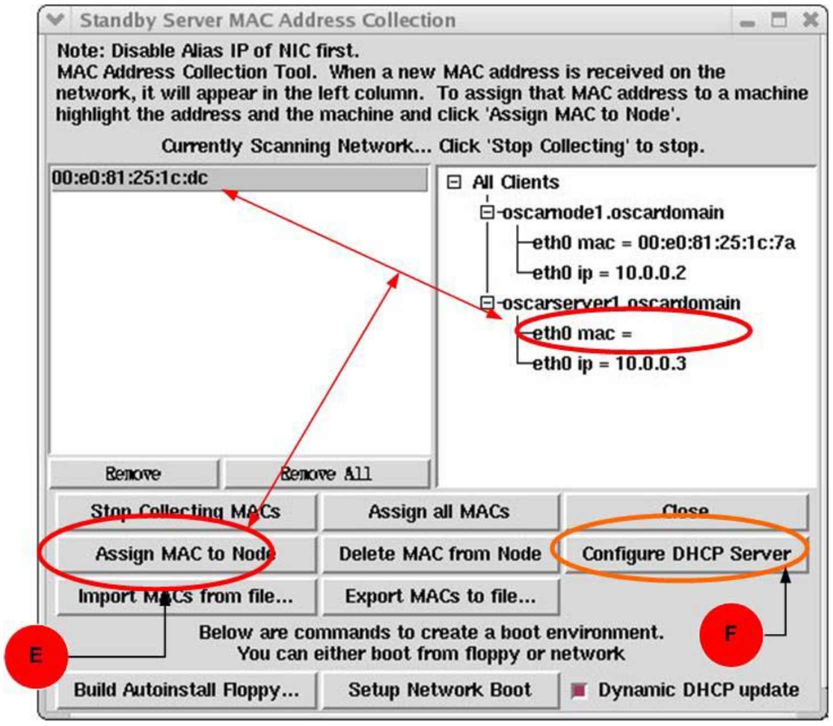
Below are commands to create a boot environment.
You can either boot from floppy or network

Figure-6: Standby Server MAC Address Configuration

The following procedures must be performed in <step 4>:

a. Click on the <Setup Network Boot> button
b. Click on the <Start Collect MAC address> button to instruct the primary server to collect the standby server's MAC address.
c. Switch to the standby server terminal, configure its boot sequence to start with the network boot, and reboot the standby system. *Please make sure the standby server eth0 is connected to the local switch where the primary server PXE daemon will be listening to the broadcast boot request. Otherwise, the primary server will not be able to collect the standby MAC address in the next step.*
d. Toggle back to the primary server screen; Figure-6, the top left panel should be populated with the standby server's MAC address.
e. Assign the MAC address to the standby server network interface (e.g. eth0).
f. Click on the <Configure DHCP Server> button.
g. Toggle back to the standby sever terminal and reboot it; the user is now ready to build a local image on the standby server. This step takes 30 minutes to one hour; please be patient.

Once the standby server image is completed, please make sure to set the server boot device starting with its local hard disk and reboot the system.



Note: Disable Alias IP of NIC first.
MAC Address Collection Tool. When a new MAC address is received on the network, it will appear in the left column. To assign that MAC address to a machine highlight the address and the machine and click 'Assign MAC to Node'.

Currently Scanning Network... Click 'Stop Collecting' to stop.

```
00:e0:81:25:1c:dc
```

```
All Clients
 -oscarnode1.oscardomain
    eth0 mac = 00:e0:81:25:1c:7a
    eth0 ip = 10.0.0.2
 -oscarserver1.oscardomain
    eth0 mac =
    eth0 ip = 10.0.0.3
```

Below are commands to create a boot environment.
You can either boot from floppy or network

Figure-7: Standby Server MAC Address Configuration after MAC Address Collection

4.5 Completing the HA-OSCAR Installation

Upon the completion of above mentioned steps, all the packages are installed completely on the cluster. The cluster should be ready to use or test. HA-OSCAR also provides a web-based management to customize the HA-OSCAR configuration. Web-based management supports the capability to enable new outage monitor/detection modules and failover capabilities. The next section will elaborate on the first release test environment and HA-OSCAR resource usage measurement.

5. HA-OSCAR 1.0 release test environment

HA-OSCAR 1.0 system verification was rigorously carried out based on our targeted requirements and test plan document. We divided our beta test into 3 categories:

- Installation and configuration,
- Failure recovery and failover, and
- Integration and regression

We set up two HA-OSCAR clusters with the OSCAR release 3.0 and RedHat 9.0. Each of our system test clusters consists of two Pentium 4 server head nodes with 1 GB of RAM, 40 GB hard disk, with at least 2GB of free disk space, and two network interface cards. The test cluster included four Pentium 4 client nodes, each with 512 MB of RAM and a 40 GB hard drive. Each client node was equipped with at least one network interface card.
After successful installation, we measured CPU usage and network traffic load that were incurred by Mon [9] based HA-OSCAR monitoring and recovery mechanisms. We recorded 3-5 seconds for a manual failover time based on the 2 second polling interval, then measured HA-OSCAR resource usages.
We conducted eight sets of HA-OSCAR resource usage experiments based on different polling intervals; ranging from 1 sec to 60 seconds. During each 30-minute experiment duration, we observed an average 0.9% CPU usage required by HA-OSCAR monitoring daemons. TCPtrace [6] was used to measure HA-OSCAR network usage from each experiment. Figure-8 shows network traffic load from each 30-minute duration by various

HA-OSCAR polling interval sizes. We also found that the impact and packet size were very minimal.

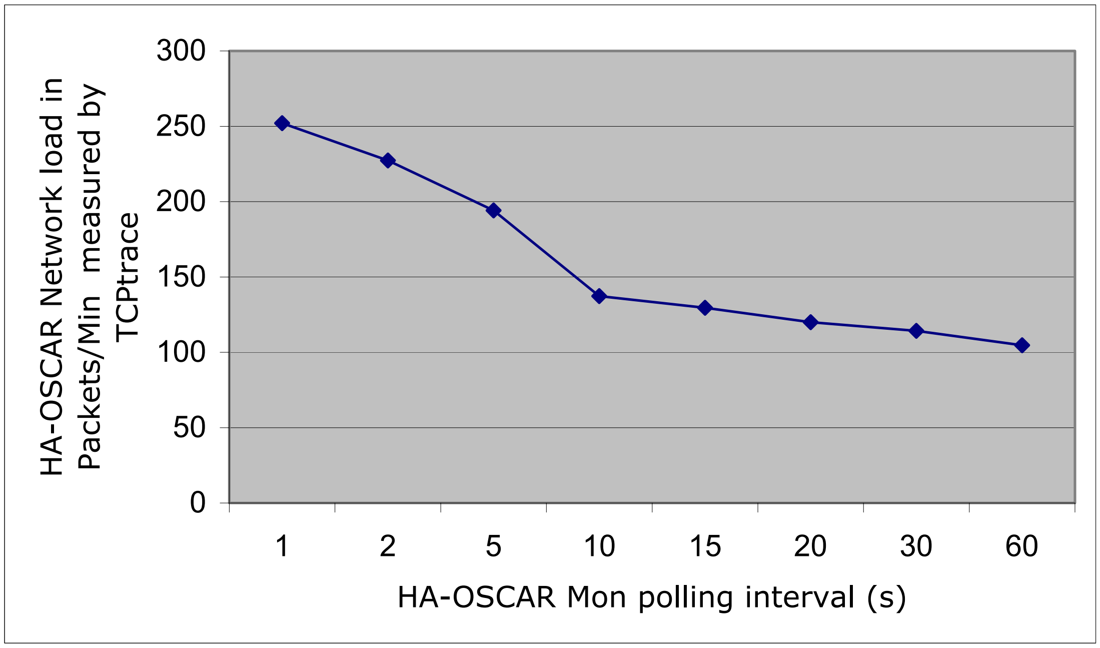


Figure-8: Comparison of network usages for HA-OSCAR different polling sizes (1-60 secs)

6. Summary and Roadmap

HA-OSCAR is the first known field-grade HA Beowulf cluster release. It is intended to provide an Open Source clustering stack for critical HPC/IT infrastructure. The 1.0 release was publicly released to the Open Source community in March 2004. The architecture supports an active-hot/standby dual-head node failover solution for Beowulf clusters. HA-OSCAR is available from the project Web site [1], where we also provide an HA-OSCAR community discussion board, in addition to a Subversion code repository and a bug reporting system.

Based on our test environment, we found that the current failover time was around 3-5 seconds, and resource consumption by the HA-OSCAR monitoring mechanism were minimal; 4% of CPU usage and negligible network traffic.

The n+1, active-active architecture is presently under consideration, as the single active head node of the active-hot_standby architecture does not take advantage of the computation power of the standby machine. We also plan to further HA-OSCAR fault tolerance with the various emerging technology checkpoint/restart MPI suites [8].

7. Reference:

[1] HA-OSCAR Project Web Site, http://xcr.cenit.latech.edu/ha-oscar

[2] Open Cluster Group, http://www.openclustergroup.org

[3] Webmin Web Site, http://www.webmin.com

[4] OSCAR Installation Manual, http://oscar.sourceforge.net

[5] J. Duell. "The design and implementation of Berkeley lab's Linux checkpoint/restart", Lawrence Berkeley National Laboratory, 2000

[6] TCP Trace Web Site, http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html

[7] HA-OSCAR Working Group, http://cenit.latech.edu/oscar
[8] Saurabh Agarwal, Gyu S. Choi, Chita R. Das, Andy B. Yoo, and Shailabh Nagar, “Co-ordinated Coscheduling in Time-Sharing Clusters through a Generic Framework,'' Proc. International Conference on Cluster Computing, Dec. 2003.
[9] http:// www.kernel.org/software/mon/