



DEPLOYING MYSQL™ DATABASE IN SOLARIS™ CLUSTER ENVIRONMENTS FOR INCREASED HIGH AVAILABILITY

Ritu Kamboj, ISV Engineering

Sun BluePrints™ Online

Part No 820-6587-10
Revision 1.0, 11/3/08

Table of Contents

Advantages of Deploying MySQL Database with Solaris Cluster	1
Overview of Solaris Cluster	2
Solaris Cluster: Hardware Components	3
Solaris Cluster: Software Components	4
Deployment Options Using Solaris Zones	6
Installation and Configuration	8
Install Solaris Cluster and MySQL Data Service Software	8
Configure the Solaris Cluster Software	9
Configure the File System	10
Configure Solaris Cluster Resources for MySQL Data Service	11
Install and Configure MySQL Database	13
Conclusions	14
About the Author	14
Acknowledgements	14
References	14
Ordering Sun Documents	15
Accessing Sun Documentation Online	15

Deploying MySQL Database in Solaris Cluster Environments

MySQL™ database, an open source database, delivers high performance and reliability while keeping costs low by eliminating licensing fees. The Solaris™ Cluster product is an integrated hardware and software environment that can be used to create highly-available data services. This article explains how to deploy the MySQL database in a Solaris Cluster environment. The article addresses the following topics:

- “Advantages of Deploying MySQL Database with Solaris Cluster” on page 1 discusses the benefits provided by a Solaris Cluster deployment of the MySQL database.
- “Overview of Solaris Cluster” on page 2 provides a high-level description of the hardware and software components of the Solaris Cluster.
- “Installation and Configuration” on page 8 explains the procedure for deploying the MySQL database on a Solaris Cluster.

This article assumes that readers have a basic understanding of Solaris Cluster and MySQL database installation and administration.

Advantages of Deploying MySQL Database with Solaris Cluster

The primary advantage of deploying the MySQL database in a Solaris Cluster environment is high availability. The Solaris Cluster environment provides fault monitoring and failover capabilities not only for the MySQL software, but also for the entire infrastructure including servers, storage, interconnects, and the operating system. If any component of the entire infrastructure fails, that failure is isolated and managed independently with no impact on availability.

MySQL Master-Slave configurations, deployed outside of a Solaris Cluster environment, provide limited availability: if the master fails, then the slave can manually be assigned master status and take over operation. However, this process is not automatic but requires manual intervention by a system administrator. Solaris Cluster removes this limitation, as it automatically fails over in the case of a master node failure. In addition, Solaris Cluster provides high availability for slaves as well as for masters. By providing high availability for slaves, these slaves can be kept updated with the masters throughout database transactions, thereby supporting scalability of MySQL database services. The Solaris cluster also provides both failover and scalable Apache Web server instances, thereby offering larger high availability coverage of the SAMP stack. Thus, MySQL Master-Slave configurations deployed in Solaris Cluster environment become highly available in the true sense.

Solaris Cluster deployments provide additional benefits beyond high availability. The Solaris Cluster environment can simplify administration by enabling clustered systems to be managed as if they were a single system. Data services, such as the MySQL database, can be deployed in Solaris Containers, providing the benefits of consolidation (as provided by Solaris Containers) as well as high availability (as provided by Solaris Cluster). Finally, both the MySQL database and Solaris Cluster are free and open source software, helping to contain costs and provide a low-cost solution for highly available databases.

Overview of Solaris Cluster

Solaris Cluster is a platform for creating highly available and scalable services. The Solaris Cluster environment extends the Solaris Operating System into a cluster operating system. A cluster is a collection of loosely coupled computing nodes that provides a single client view of network services or applications, including databases, Web services, and file services.

Each cluster node is a standalone server that runs its own processes. These processes communicate with one another to form a single system that cooperatively provides applications, system resources, and data to users. Solaris Cluster 3.2 software can also run in Solaris Containers or Logical Domains, providing consolidation of applications and optimizing resource utilization.

A cluster offers several advantages over traditional single-server systems. These advantages include support for failover and scalable services, capacity for modular growth, and low entry price compared to traditional hardware fault-tolerant systems.

The goals of the Solaris Cluster software are:

- Reduce or eliminate system downtime because of software or hardware failure
- Ensure availability of data and applications to end users, regardless of the kind of failure that would normally take down a single-server system
- Increase application throughput by enabling services to scale to additional processors by adding nodes to the cluster
- Provide enhanced availability of the system by enabling maintenance to occur without shutting down the entire cluster

The following sections describe the hardware and software components of the Solaris Cluster platform in further detail.

Solaris Cluster: Hardware Components

A typical hardware configuration for a Solaris Cluster installation consists of the following components:

- Cluster nodes with local disks
- Multihost storage devices (shared disks)
- Cluster Interconnect (private interconnect)
- Public network interfaces

Figure 1 depicts a high level diagram of the hardware environment for a two node cluster. Each node is connected to local and shared disks, and there are redundant public and private interfaces.

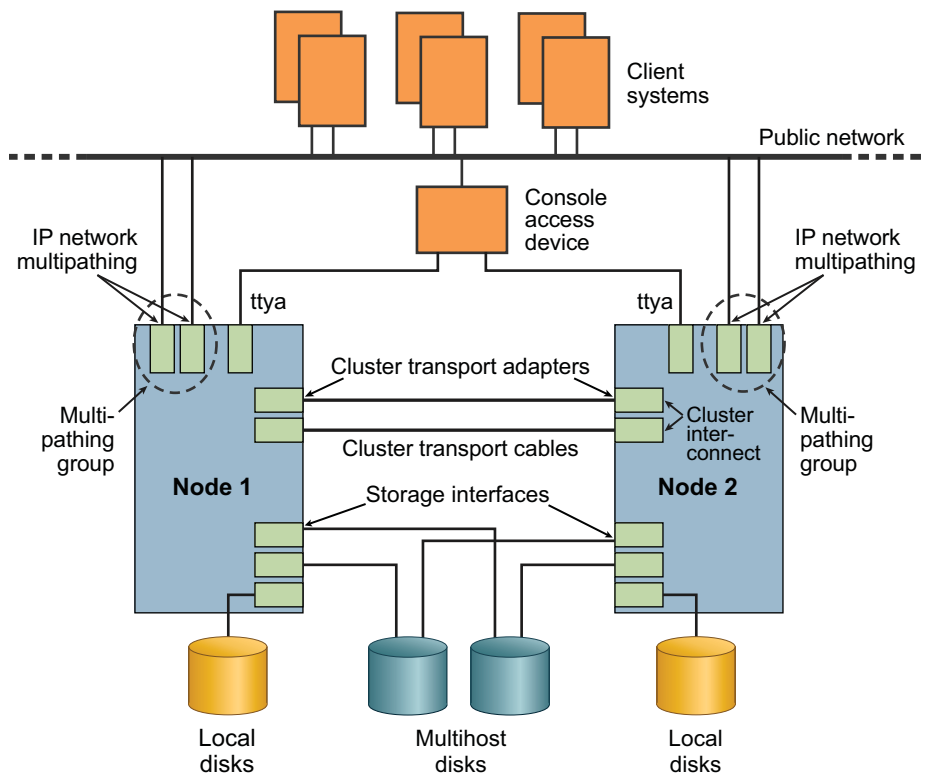


Figure 1. Solaris Cluster hardware components.

- *Cluster Nodes with Local Disks*

A cluster node is a machine (or Solaris Zone) that is running both Solaris Operating System and Solaris Cluster software. Solaris Cluster software supports up to sixteen nodes in a cluster on SPARC® systems and up to eight nodes in a cluster on x64 platforms.

All nodes in the cluster are grouped under a common name, referred to as the cluster name. The cluster name is used for accessing and managing the cluster.

Every node in the cluster is aware when another node joins or leaves the cluster. Additionally, every node in the cluster is aware of the resources that are running locally as well as the resources that are running on the other cluster nodes.

- *Multihost Storage Devices (Shared Disks)*

Multihost devices, disks that can be connected to more than one node at a time, provide highly available data storage in the Solaris Cluster environment. Local disks are connected to only a single node and, therefore, are not protected against node failure (they are not highly available). However, all disks, including local disks, are included in the global namespace and are configured as global devices. Therefore, the disks themselves are visible from all cluster nodes.

Multihost devices are used to store application data, application binaries, and configuration files. Global access is provided through a primary node that masters the disks. Multihost devices can tolerate single-node failures. If clients request data through one node and that node fails, the requests are switched over to use another node with a direct connection to the same disks.

- *Cluster Interconnect*

The cluster interconnect is the physical configuration of devices that are used to transfer cluster-private communications and data service communications between cluster nodes. In contrast to the public network, the cluster interconnect provides a private channel for internode communication.

Redundant interconnects enable operation to continue over the surviving interconnects while system administrators isolate failures and repair communication. The Solaris Cluster software detects, repairs, and automatically re-initiates communication over a repaired interconnect.

All nodes must be connected by the cluster interconnect through at least two redundant physically independent networks, or paths, to avoid a single point of failure. While two interconnects are required for redundancy, up to six can be used to spread traffic to avoid bottlenecks and improve redundancy and scalability. The Solaris Cluster interconnect uses Fast Ethernet, Gigabit Ethernet, InfiniBand, or the Scalable Coherent Interface (SCI, IEEE 1596-1992), enabling high-performance cluster-private communications.

- *Public Network Interfaces*

Public network interfaces are used by client systems to access data services on the cluster.

Solaris Cluster: Software Components

The Solaris Cluster system is designed to prevent data corruption and ensure data integrity while providing high availability and scalability of data services. Figure 2 shows a high level view of the software components that work together to create the Solaris Cluster software environment.

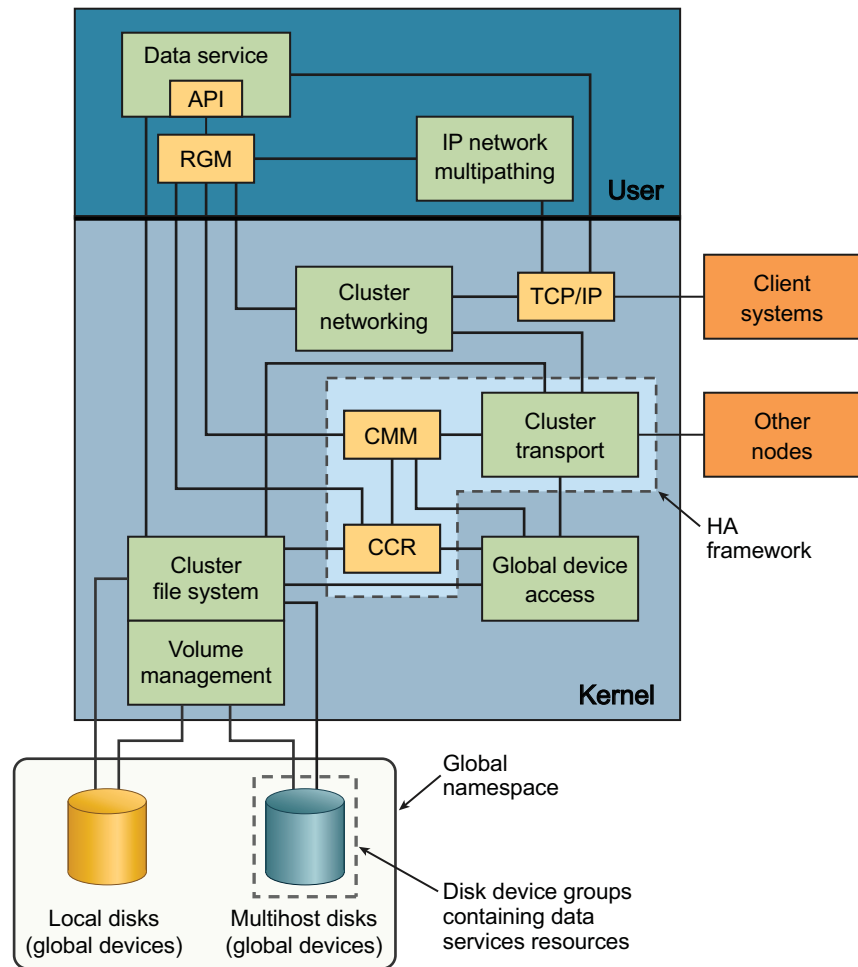


Figure 2. Solaris Cluster software components.

The Solaris Cluster uses the Resource Group Manager (RGM), Cluster Membership Monitor (CMM), Cluster Configuration Repository (CCR), disk fencing, and quorum devices to help prevent data corruption and ensure data integrity

- *Resource Group Manager*
The Resource Group Manager (RGM) controls data services as resources and is responsible for availability management decisions of data services. The RGM stops and starts resource groups on selected nodes or in selected zones in response to cluster membership changes.
- *Cluster Membership Monitor*
Cluster Membership Monitor (CMM) ensures data integrity by ensuring that an unhealthy node continues to remain out of a cluster until it is repaired.

The CMM performs the following tasks:

- Accounting for a change in cluster membership, such as a node joining or leaving the cluster

- Ensuring that an unhealthy node leaves the cluster
- Ensuring that an unhealthy node remains inactive until it is repaired
- Preventing the cluster from partitioning itself into subsets of nodes
- *Cluster Configuration Repository*

The Cluster Configuration Repository (CCR) is a private, cluster-wide, distributed database for storing information that pertains to the configuration and state of the cluster. To avoid corrupting configuration data, each node must be aware of the current state of the cluster resources. The CCR ensures that all nodes have a consistent view of the cluster. The CCR is updated when error or recovery situations occur or when the general status of the cluster changes.
- *Disk Fencing*

A major issue for clusters is a failure that causes the cluster to become partitioned (called *split brain*). When this situation occurs, not all nodes can communicate, so individual nodes or subsets of nodes might try to form individual or subset clusters. Each subset or partition might “believe” it has sole access and ownership to the multihost disks. Attempts by multiple nodes to write to the disks can result in data corruption.

Failure fencing limits node access to multihost disks by preventing access to the disks. When a node leaves the cluster (it either fails or becomes partitioned), failure fencing ensures that the node can no longer access the disks. Only current member nodes have access to the disks, ensuring data integrity.
- *Quorum Device*

A quorum device is a shared storage device that is shared by two or more nodes or a quorum server that contributes votes used to establish a quorum. The cluster can operate only when a quorum of votes is available. The quorum device is used when a cluster becomes partitioned into separate sets of nodes to establish which set of nodes constitutes the new cluster.

Deployment Options Using Solaris Zones

The MySQL failover data service in Solaris Cluster 3.2 is supported to run in Solaris Zones. The Solaris Zones partitioning technology provides isolated and secure environments within a single instance of the Solaris Operating System (Solaris OS). This isolation prevents processes that are running in one zone from affecting processes running in other zones. Every Solaris system contains one global zone, which is the default zone for the system. When Sun Cluster software is run on the Solaris 10 OS, any number of non-global zones can be created on a physical node. By default, all processes run in the global zone if no non-global zones are created on the system.

Note – See *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones* for more information on Solaris Zones.

In the Solaris Cluster environment, a zone can be used as an RGM node hosting data services. This zone is treated by the RGM like any other node in the cluster. Data services can be configured to run in a zone, and these data services will fail over automatically to other nodes or zones in the cluster.

The MySQL database is supported in the following configurations on Solaris Clusters:

- Default Solaris 10 OS deployment
The default Solaris 10 OS deployment, also referred to as global zone deployment, does not make use of any non-global zones.
- Non-global zone deployment
In this deployment option, one or more non-global zones are configured. Each Solaris Zone (the global zone and any non-global zones) is considered to be like any other node in the cluster.
- Non-global failover zone deployment
In this deployment option, non-global zones are treated as failover resources of the resource group. These resources can failover from one node or zone to another like any other failover resource, and are managed through the Sun Cluster HA for Solaris Containers data service agent. This agent manages each zone as a resource of the resource group.

Note – The examples in this paper use the default Solaris OS deployment, which does not make use of any non-global zones. For more information on deploying the MySQL database using Solaris Zones, see the *Sun Cluster Data Service for MySQL Guide for Solaris OS*.

Installation and Configuration

To function as a cluster member, a node must have the following software installed:

- Solaris Operating System
- Solaris Cluster software
- Data service application (MySQL database in this example)
- Volume management

The following sections provide details on installing and configuring the Solaris Cluster software and MySQL data service.

Install Solaris Cluster and MySQL Data Service Software

Before starting Solaris Cluster installation, ensure the following hardware prerequisites are met:

1. The cluster node has two physical interfaces for use as private interconnect.

Issue the following command to determine how many NICs are present on the system:

```
# dladm show-dev
```

Make sure that the NICs that will be used for the cluster are wired up. The link field in the output of `dladm` command should be set to `up`. These NICs do not have to be plumbed (that is, `ifconfig -a` should not list them); Solaris Cluster will plumb the NICs.

2. A 512 MB local disk partition is defined as `/globaldevices` in the `/etc/vfstab` file:

```
/dev/dsk/c3t1d0s0 /dev/rdisk/c3t1d0s0 /globaldevices ufs 1 no logging
```

3. The cluster node has at least two shared disks (not partitions, but disks) available. One shared disk will be used as a quorum device and another shared disk will be used for the shared cluster file system.

There is no need to use a dedicated disk as a quorum device. The quorum functionality can be leveraged on any shared disk without conflicting with the usage of that disk for a cluster file system or other application. However, in this particular deployment, two shared disks are used.

4. All current Solaris OS patches have been installed.

Solaris Cluster and MySQL Data Service Installation

It is assumed that the reader is familiar with the installation procedure. Full details on this procedure are beyond the scope of this document, but can be found in the *Sun Cluster Software Installation Guide for Solaris OS*.

1. On both nodes in the cluster, run the installer command to invoke the Solaris Cluster installer GUI.

2. Follow the on-screen instructions to install the Sun Cluster and Sun Cluster HA for MySQL.
 - a. When prompted, choose to install *Sun Cluster Core 3.2 core software* and *Sun Cluster HA for MySQL*.
 - b. When prompted, choose *Configure later before the installation*.

Configure the Solaris Cluster Software

Perform this procedure from one node of the cluster to configure the Solaris Cluster software on all nodes in the cluster.

1. Invoke the `scinstall` command:

```
# /usr/cluster/bin/scinstall
```

2. From the main menu, pick the “Option 1” to “Create a new cluster or add a cluster node.”
3. From the new cluster and Cluster Node menu, pick the “Option 1” to “Create a new cluster.”
4. From the Typical or Custom Mode, pick the “Option 1” for typical mode.
5. From the Cluster Nodes menu, type the node name.
6. Provide the first and the second private adapter name.
7. Type “no” for “disable automatic quorum device selection.”
8. Type “yes” for “create the new cluster.”
9. Type “no” for “Interrupt cluster creation for sccheck errors.”

At this point, the `scinstall` utility configures all cluster nodes and reboots the cluster. The cluster is established when all nodes have successfully booted into the cluster. The Solaris Cluster installation output is logged in the `/var/cluster/logs/install/scinstall.log.N` file.

Post-Configuration Checks

After the Solaris Cluster software is installed and running, perform the following post-installation checks to verify operation.

1. Solaris Cluster automatically selects one shared disk to be used as quorum device. Invoke the following `clquorum` command to determine which disk is used as the quorum device:

```
# ./clquorum list
```

2. Invoke the following command to display the current node status:

```
# ./clnode status
```

Configure the File System

Either a Cluster File System or a failover file system using a ZFS pool can be used. A failover file system has increased performance over a Cluster File System as all nodes do not have to commit. However, as only one node sees the file system at a time, there is a slight increase in failover time.

This section describes both of these methods in detail.

Use a Cluster File System:

1. Become superuser on any node of the cluster.
2. Issue the `newfs` command on UFS file system, where `raw-disk-device` is a raw disk device or a slice from the Volume Manager.

```
# newfs raw-disk-device
```

3. On each node create a mount point for the Cluster File System, and add an entry to the `/etc/vfstab` for the mount point.
4. Mount the Cluster File System.

More detailed information on creating Cluster File Systems can be found in the *Sun Cluster Software Installation Guide for Solaris OS*, Chapter 6 “Creating Cluster File Systems.”

Failover File system using ZFS:

This section describes the creation and configuration of a failover file system using a ZFS pool. On the first node, perform these steps as root user:

1. Create a ZFS storage pool and a ZFS in that pool
 - a. Choose a suitable shared disk among those available. A shared disks will have two entries, when listed by typing the `cldevice list -v` command.
 - b. Execute the `zpool` command. In this example, volume `mysql3` and the specified shared disk name are used for illustration:

```
# zpool create mysql3 clt20030003BA13E6A1d0
```

- c. Create `sqlvol` in the ZFS pool `mysql3` and set mount point:

```
# cd /
# zfs create mysql3/sqlvol
# zfs set mountpoint=/global/mysql mysql3/sqlvol
```

2. Create a failover resource group:

```
# clsresourcegroup create mysql-rg
```

3. Register the `HASStoragePlus` resource type, if not already registered:

```
# clsresourcetype register HASStoragePlus
```

4. Create a HAStoragePlus resource named sql-stor in the resource group for the local ZFS:

```
# clresource create -g mysql-rg -t HAStoragePlus -p Zpools=mysql3
```

5. Bring the resource group that contains the HAStoragePlus resource online:

```
# clresourcegroup online -M mysql-rg
```

For more information, see “Enabling Highly Available Local File Systems” in the *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*.

Configure Solaris Cluster Resources for MySQL Data Service

Configuring the MySQL data service involves the following steps:

- Register a resource type
- Create resource groups
- Add resources to resource groups
- Bring resources online

Note – In previous releases of the Solaris Cluster software, the `scrgadm` command was used to manage resource types, resource groups, and resources. In the Solaris Cluster 3.2 release, object-oriented commands (`clresourcetype`, `clresourcegroup`, etc.) are used to perform these tasks.

Register a Resource Type

A resource type provides specification of common properties and callback methods that apply to all resources of the given type. Two resource types, HAStoragePlus and Generic Data Service (GDS), are used in this example:

- SUNW.HAStoragePlus describes a resource type which allows for specifying dependencies between data service resources and device groups, cluster (global) and local file systems. This enables data services to be brought online only after their dependent device groups and file systems are guaranteed to be available. HAStoragePlus also provides support for mounting, unmounting and checks of file systems.
- The Generic Data Service (GDS) is a mechanism that enables applications to be made highly available or scalable by plugging them into the Solaris Cluster Resource Group Manager (RGM) framework.

A resource type must be registered before resources of that type can be created.

1. Register the SUNW.HAStoragePlus and SUNW.gds resource types on both nodes by issuing the following commands on one node in the cluster:

```
# clresourcetype register SUNW.gds
# clresourcetype register SUNW.HAStoragePlus
```

Create a Resource Group

A resource group contains a set of resources, all of which are brought online or offline together on a given node or zone or set of nodes or zones. An empty resource group must be created before resources can be placed in it.

There are two types of resource groups:

- Failover
- Scalable

Since the MySQL data service is a failover data service, a failover resource group must be created. A failover resource group contains the following type of resources:

- Network address resources, which are instances of the built-in resource types *LogicalHostName* and *SharedAddress*
- Failover resources, which are MySQL resources to be failed over.

1. Use the following command to create a failover resource group named `mysql-rg`:

```
# clresourcegroup create mysql-rg
```

Add Resources to the Resource Group

Once the resource group is created, add resources to this resource group:

1. Create a resource for the MySQL Logical Hostname. The following command creates a resource named `mysql-resource` containing the logical hostname:

```
# clsreslogicalhostname create -g mysql-rg mysql-resource
```

To verify logical hostname resource is online, run the `ifconfig -a` command to confirm the virtual IP address is configured on the network interface.

2. Create a resource for the MySQL disk storage using the following command:

```
# clresource create mysql-has-resource -g mysql-rg  
-t SUNW.HAStoragePlus  
-x FileSystemMountPoints={MySQL mount points}
```

Bring Resources Online

After the resources are created, use the following command to bring the resource group online.

1. The following command enables the resource group `mysql-rg`:

```
# clresourcegroup online -M mysql-rg
```

Install and Configure MySQL Database

In order to configure the MySQL database, it must be enabled for cluster usage and then registered. This involves the following steps:

1. Create an admin password for the MySQL database server admin user.
2. Edit the `mysql-config` file and add the following information:

```
# Where is mysql installed (BASEDIR)
MYSQL_BASE=

# Mysql admin-user for localhost (Should be root)
MYSQL_USER=

# Password for mysql admin user
MYSQL_PASSWD=

# Configured logicalhost
MYSQL_HOST=

# Specify a username for a faultmonitor user
FMUSER=

# Pick a password for that faultmonitor user
FMPASS=

# Socket name for mysqld ( Should be /tmp/.sock )
MYSQL_SOCKET=

# FOR SC3.1 ONLY, Specify the physical hostname for the
# physical NIC that this logicalhostname belongs to for every node in
# the cluster this Resourcegroup can located on.
# IE: The logicalhost lh1 belongs to hme1 for physical-node phys-1 and
# hme3 for physical-node phys-2. The hostname for hme1 is phys-1-hme1
# and for hme3 on phys-2 it is phys-2-hme3.
# IE: MYSQL_NIC_HOSTNAME="phys-1-hme1 phys-2-hme3"
MYSQL_NIC_HOSTNAME=""
```

3. Run the `mysql-register` script.
4. Edit the `ha-mysql-config` file, using the comments in that file as a guide, to create and register the MySQL database as a failover data service:

```
# Where is mysql installed (BASEDIR)
MYSQL_BASE=

# Mysql admin-user for localhost (Should be root)
MYSQL_USER=

# Password for mysql admin user
MYSQL_PASSWD=

# Configured logicalhost
MYSQL_HOST=

# Specify a username for a faultmonitor user
FMUSER=
```

```
# Pick a password for that faultmonitor user
FMPASS=

# Socket name for mysqld ( Should be /tmp/.sock )
MYSQL_SOCKET=

# FOR SC3.1 ONLY, Specify the physical hostname for the
# physical NIC that this logicalhostname belongs to for every node in
# the cluster this Resourcegroup can located on.
# IE: The logicalhost lh1 belongs to hme1 for physical-node phys-1 and
# hme3 for physical-node phys-2. The hostname for hme1 is phys-1-hme1
# and for hme3 on phys-2 it is phys-2-hme3.
# IE: MYSQL_NIC_HOSTNAME="phys-1-hme1 phys-2-hme3"
MYSQL_NIC_HOSTNAME=""
```

5. Enable each resource. If multiple instances were created, repeat for each instance:

```
# clresource enable mysql-has-resource
```

Conclusions

The Solaris Cluster HA agent for MySQL provides a mechanism for fault monitoring and automatic failover of the MySQL database service. It delivers improved service levels for MySQL applications by providing continuous network, data, and service availability, thereby making recovery from failure transparent to clients. Other applications that are frequently employed by MySQL customers — including application servers, Web servers, and file and print services such as SAMBA — can be consolidated using Solaris Containers in a clustered environment, thereby improving resource utilization.

About the Author

Ritu Kamboj is a Staff Engineer in ISV Engineering's Open Source Team at Sun Microsystems. She has over 12 years of experience in software development with expertise on database design, performance and high availability. She has worked extensively on Sybase, Oracle and MySQL databases. Recently her primary focus has been making MySQL run best on the Solaris platform.

Acknowledgements

The author would like to recognize Gia-Khanh Nguyen and Detlet Ulherr from the Sun Cluster group for their contributions to this article.

References

Sun Cluster 3.2 2/08 Documentation Center:

<http://docs.sun.com/app/docs/doc/820-2562/xiwa?a=view>

Sun Cluster 3.2 2/08: Sun Cluster Software Installation Guide for Solaris OS:

<http://docs.sun.com/app/docs/doc/820-2555>

Sun Cluster Data Service for MySQL Guide for Solaris OS:

<http://docs.sun.com/app/docs/doc/819-3059>

System Administration Guide: Solaris Containers-Resource Management and Solaris Zones:

<http://docs.sun.com/app/docs/doc/817-1592>

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The docs.sun.com Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is

<http://docs.sun.com/>

To reference Sun BluePrints Online articles, visit the Sun BluePrints Online Web site at:

<http://www.sun.com/blueprints/online.html>

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

© 2008 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, BluePrints, MySQL, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Information subject to change without notice.



Printed in USA 11/08