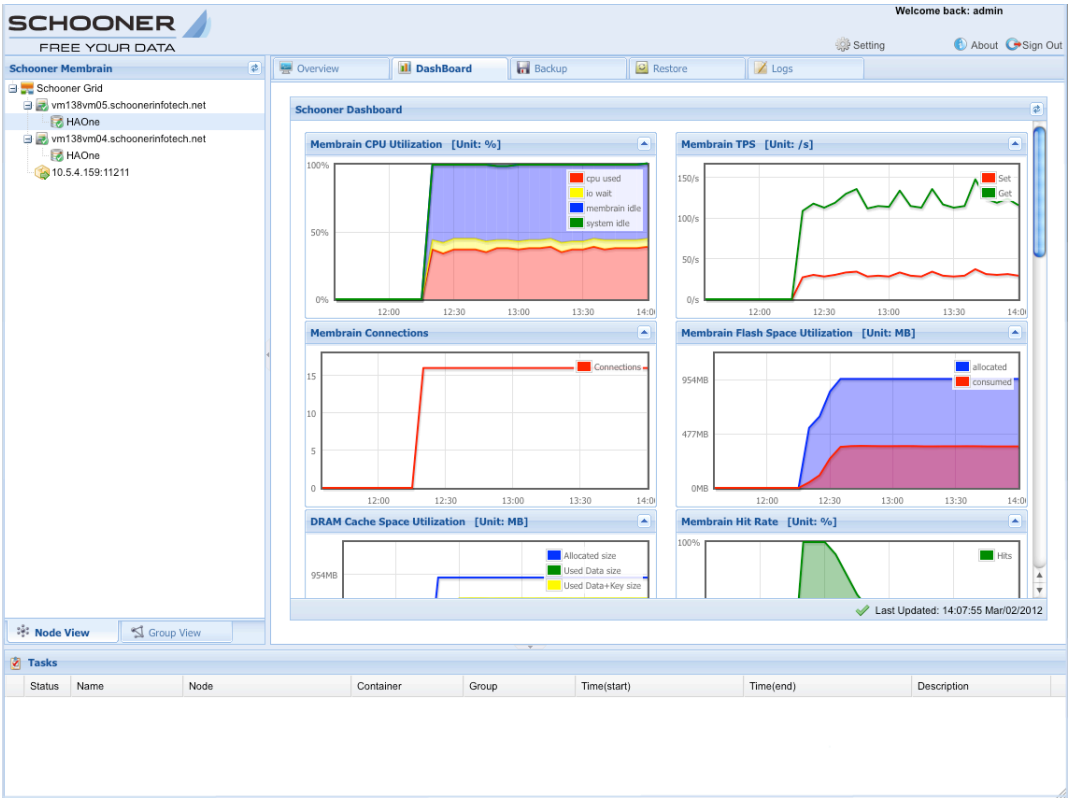
Application & Administration Guide

Schooner Membrain™

Version 4.0

Software that transforms standard x86 servers and flash memory into NoSQL and caching super-servers



SCHOONER
FREE YOUR DATA

Technical Support

Technical support for Schooner Information Technology products in North America is available from the following sources:

- Phone: (877) 888-5064 (toll free)
- Fax: (408) 736-4212
- Email: **support@schoonerinfotech.net**
- Website: **www.schoonerinfotech.com/support**

Documentation ID: S-M-v4.0-AG-01

Schooner Membrain™ Application & Administration Guide

Issued March 2012

Schooner Information Technology
501 Macara Ave., Suite 101
Sunnyvale, CA 94085, USA
Tel: (408) 888-1619
(877) 888-5064 (Toll free)
Fax: (408) 736-4212

Contents

Chapter 1. Introduction

Welcome

Thank you for purchasing Schooner Membrain™ and welcome to Schooner's new world of smart, fast, scalable, cost-effective data access.

Your Schooner Membrain™ server is compatible with your existing Memcached environment.

The information in this Guide is organized as follows:

- "Chapter 1: Introduction" discusses the system and network requirements of Schooner Membrain™ and its key features and benefits.
- "Chapter 2: Configure Your Membrain Environment" discusses the special considerations in optimizing Membrain for the Schooner system.
- "Chapter 3. Common Management Tasks" describes how to manage and monitor nodes, groups, and instances of Schooner Membrain™.
- "Chapter 4. System and Data Recovery" discusses the procedures for recovering system configuration and/or container data.
- "Chapter 5. Schooner CLI" describes how to use the Schooner CLI to configure and manage the system.
- "Chapter 6. Mirrored Group Network Setup Requirements" contains important information about the network environment that must be taken into consideration when configuring mirrored groups.
- "Chapter 7. Hot Keys and Hot Clients" discusses how hot keys and hot clients are handled in Schooner Membrain™.

Integrating Schooner Membrain™

Normally, integrating Schooner Membrain™ into your Memcached environment is simply a matter of adding the Schooner 's network interfaces to the configuration files of your Memcached clients. There should be no need to alter your application client code unless you require data persistence. In that case, see "Persistence" on page 4.

Schooner Administration Architecture

Web Administration

Schooner Membrain™ nodes can be administered via the Schooner Web-based graphical user interface (GUI). You can log into any node in the cluster from the GUI for administration as long as it is running the latest Schooner application software.

CLI Administration

Schooner Membrain™ nodes may also be administered via the Schooner Command Line Interface (CLI). Each CLI can manage the node on which you are logged onto.

Schooner Membrain™ High Availability

Mirrored Groups

Schooner Membrain™ supports replication of keys and data between a pair of mirrored nodes.

A "mirrored group" defines a high availability domain within a Schooner cluster:

- Replication of updates on one node to the other node.
- Automatic fail-over of Membrain access via VIP (virtual IP).
- Automatic recovery of a failed node.

When nodes are added to the system, they are assigned to a pool of non-replicating nodes, called the Independent Group. This group does not support replication.

In order to start replication between two nodes, you need to create a mirrored group via the CLI by specifying the two nodes you want to use. From this point on, all Membrain data will be replicated between the two nodes.

In the event of a failure of a mirrored node, the survivor node will take over the VIPs of the failed node. This allows Membrain clients to continue operation.

Upon recovery of the failed node, the failed node will sync all data from the survivor node. Once the recovery is complete, the revived node will take over its VIPs and operation should return to the normal state.

Backup

Schooner Membrain™ supports backup and restore of persistent Membrain data. For more information, refer to backup and restore procedures in Chapter 3.

Support for Binary Protocol

With support for Binary Protocol, Schooner Membrain™ is 100% memcapable. Schooner Membrain™ is able to simultaneously serve Memcached clients that use either ASCII or Binary Protocol. Binary Protocol enables extensibility (standard support for additional data in the protocol), enhanced multi-gets (with support for quiet operations), and enhanced CAS (compare and swap) support.

Support for Large-Sized Objects

This feature enables users to cache objects that are as large as 8 MB in size. Applications do not need to perform complex operations to split large objects into 1-MB chunks to store them on the Schooner Membrain™ server.

Support for 128 Schooner Dynamic Containers

This feature enables 128 containers to operate simultaneously on a single Schooner Membrain™ server. The enhanced user interface makes it easy to manage a large number of containers.

The feature allows for simplified consolidation of a large number of Membrain instances in a way that each instance can be simply mapped to a container. This is

also the first step in enabling cloud-computing providers to start leveraging the Schooner Membrain™ solution.

The sum of all container sizes must be less than or equal to 1 terabyte.

Flexible Container Addressing Schemes

This feature allows multiple containers on a Schooner Membrain™ server to be addressed using different IP addresses, but the same Port number. This feature makes it easier to integrate Schooner Membrain™ into a variety of customer networking environments where clients are configured to target servers based on IP addresses.

Chapter 2. Configuring Your Membrain Environment

Schooner Membrain™ is compatible with existing Memcached environments. This chapter discusses special considerations in configuring Membrain for a Schooner system.

Configuring Your Memcached Clients

Configure your Memcached clients to use your Schooner Membrain™ listening addresses. These are the 's IP addresses on your application subnets, which you entered in Step 2 of the First Time Wizard (FTW).

In most cases, you can do this by simply adding Schooner Membrain™ network interfaces to the configuration file of your Memcached clients.

Persistence

Because the flash memory in your Schooner Membrain™ is non-volatile, the data stored there is persistent. However, to optimize performance, DRAM memory is used as a cache front-end. DRAM memory is volatile and will lose data in the event of a power loss or node failure.

If you are using your Schooner Membrain™ server as a standard cache, you may not need your data to be persistent. In that case, set Persistence to NO, and the cache (first DRAM, then flash) will start off empty on restart. However, it will take some "warm-up" time for the cache to fill. The "warm-up" time can be shortened if the cache data is persistent.

If you are using your Schooner Membrain™ server as a back-end storage medium, it is important that all data be persistent.

If you set Persistence to YES, beware that the persistent data in flash memory may be "stale" (out of date) upon restart after failure. This is because, upon restart, a data object that might have been updated in DRAM might not have been updated in flash. Therefore, if data in DRAM is lost, the persistent data retrieved from flash may be stale.

Ensuring Current, Persistent Data

To help provide precise control over data consistency, Schooner provides the SYNC API (which includes the sync and sync_all commands) in Schooner Membrain™ for persistent containers. These two commands are Schooner-specific additions to the Memcached protocol and follow the standard Memcached conventions. Both respect data integrity in the sense that successful completion guarantees that complete (not partial) objects have been stored.

Upon successful return, the sync command guarantees that a specific object has been stored on persistent storage, while sync_all guarantees that all objects that were successfully updated before the invocation of the sync_all command are committed to flash.

Keep the following points in mind when using these two commands:

- Any writes that are issued after (or concurrent with) the last `sync/sync_all` may or may not be visible after a recovery.
- If the container is running in eviction mode, it is possible that a committed object won't show up after a restart due to eviction. However, the system does guarantee that if an object is recovered, it will have a value no older than the last value before the most recent successful `sync/sync_all` command.

The use of the SYNC API is not mandatory for persistent containers. Even if you do not use the sync/sync_all commands, you should still be able to recover most of your objects since Schooner Membrain™ automatically syncs in the background. Typically only a small number of the most recent updates will not be recovered.

sync()

Synchronizes a specific item between DRAM cache and secondary storage.

Table 2-1: Attributes of the 'sync' Command

Attribute	Description
Syntax	`sync <key>\r\n`
Parameter	`<key>` the object key for the item to synchronize
Returns if successful	`SYNCED\r\n`
Errors	`NOT_FOUND\r\n` if the item with this key was not found. Other error conditions as found in Membrain protocol.

sync_all()

Synchronizes all unsynchronized items between DRAM cache and secondary storage.

Table 2-2: Attributes of the 'sync_all' Command

Attribute	Description
Syntax	`sync_all\r\n`
Parameters	None
Returns if successful	`SYNCED\r\n`
Errors	Error conditions as found in Membrain protocol.
Notes	An updated item is synchronized only if the update is completed before the server receives the `sync_all` request.

Recovery

On recovery, data is guaranteed to be consistent with the last sync call.

Multiple Membrain Containers

In Schooner Membrain™, multiple storage domains may be created to provide fine-grain control over storage resources. A "container" defines the policies controlling eviction (whether objects can be evicted or not), storage capacity, and persistence for a

given storage domain. Each Schooner Membrain™ server can support up to 128 dynamic containers. Containers are identified by a unique name.

Container Addressing Schemes

Schooner Membrain™ allows the creation of multiple containers on the same TCP port number but different IP addresses. With this feature, users can direct traffic from applications to different containers using the same TCP port number but different IP addresses, without having to make changes to the TCP port-Membrain mapping on the application side. By specifying a combination of an IP address and a port number, you can direct traffic from a Memcached client to a specific container.

Below are the container-addressing schemes that are supported.

A specific IP address and a specific TCP port number

Traffic to a particular IP address and TCP port is directed to a specific container. The IP address can be a real IP or VIP.

Example:

1. Container A: 10.10.10.1 (IP address) 11211 (TCP port number) – To Container A only.
2. Container B: 10.10.10.2 (IP address) 11211 (TCP port number) – To Container B only.

Up to 16 unique IP addresses supported per container

Example:

1. Container A: 10.10.10.1 (IP address) 11211 (TCP port number)
2. Container A: 10.10.10.2 (IP address) 11211 (TCP port number)
3. Container A: 10.10.10.3 (IP address) 11211 (TCP port number)

All traffic addressed to IP addresses 10.10.10.1/2/3 and TCP port number 11211 will be sent to Container A. In this scenario, a container can use up to 16 different IP addresses.

One TCP port number per container

Example:

1. Container A: 10.10.10.1 (IP address) 11211 (TCP port number)
2. ~~Container A: 10.10.10.1 (IP address) 22222 (TCP port number)~~

VIPs required for replication mode

If two nodes are configured as a replication pair, VIPs must be used. In this scenario, a container can have up to 16 VIPs, or 8 per node. The VIPs must be configured on both nodes and the configurations must be completed before they are assigned to containers.

Unique container names

Containers are identified by name on the Schooner Centralized Administrator or CLI. So their names must be unique.

Backward compatibility

In this scenario, a container is configured using a specific TCP port number and any IP address so that it can listen on all IP addresses and one specific TCP port. Each container uses one specific TCP port, and no port can be shared. Also, no IP address (including VIP) should be added to a container that is configured to listen on any IP address; otherwise, the application will not be backward compatible with its earlier versions.

Example:

1. Container A: None (IP address) TCP port number 11211
2. Container B: None (IP address) TCP port number 11212
3. ~~Container C: None (IP address) TCP port number 11212~~ (Port 11212 has already been used by Container B, so it cannot be assigned to any other container.)
4. ~~Container D: 10.1.1.4 (IP address) TCP port number 11212~~ (Container D is configured to listen on all IP addresses; no specific IP address should be added to it.)

Schooner Membrain™ Resource Allocation

This section describes the system resources required by Schooner Membrain™ and discusses how to assign them.

Schooner Membrain™ System Resources

Schooner Membrain™ requires exclusive control of a portion of the following system resources:

- CPU cores
- DRAM
- Secondary storage

The amount of system resources allocated to Schooner Membrain™ will dictate its maximum performance. For example, Schooner has verified that the following general hardware configuration will generate the maximum performance. Note that the actual performance will depend on the specific vendor platform and configuration.

- 2 x Intel Xeon, 4 cores, 2.6Ghz CPU
- 128GB DRAM
    - This amount of memory generally ensures high cache hit rates for most applications.
- 1TB solid-state secondary storage
    - Solid-state storage is required in order to maintain the maximum throughput.
    - Disk drives may also be used as secondary storage, but the maximum performance will be much less than if solid-state devices are used.
    - Note that 1TB of secondary storage is the maximum supported size.
- 10GE network interface or multiple bonded 1GE interfaces

- Used for both application and replication traffic.

Schooner Membrain™ System Resource Allocation

The performance and cache capacity of your Schooner Membrain™ installation will depend on the amount of CPU, DRAM, secondary storage and networking resources you assign. In this section, we show you how to allocate these resources appropriately for your applications.

Secondary Storage Allocation

Solid-state storage devices will yield maximum throughput. Hard disks may also be used but the performance will be lower than for solid-state devices. Membrain supports a maximum of 1TB secondary storage size.

Schooner Membrain™ treats secondary storage as a single file. This file can be mapped to physical storage in a number of ways:

- As a link to physical devices.
- As a file in a file system partition.
- As a file in a file system partition on a RAID array.

Here are some guidelines for optimizing the secondary storage:

- For maximum performance, use 8 SSDs or a pair of PCI express flash cards.
- Software RAID generally performs faster than hardware RAID.
- When using SSDs, use RAID5 if using more than 2 SSDs.
- When using PCI express flash cards, use a pair of cards with RAID10.
- When configuring RAID5 using md software RAID, use a chunk size that is large enough to minimize the number of writes per object. A typical value is 256KB.
- Use the xfs file system for maximum performance, create the file system with these settings:

```
-d sunit=512,swidth=3584 -i attr=2
```

- Mount the file system with these settings:

```
mount -o noatime, nodiratime, sunit=512, swidth=3584,
attr2, largeio, prjquota
```

- When using hard disks for storage, use non-persistent mode with eviction for maximum performance. Schooner strongly recommends the use of flash storage.

DRAM Cache Allocation

The amount of DRAM to assign to the Schooner Membrain™ cache depends on the total amount of container storage you wish to support.

In general, a ratio if 8:1 secondary storage to DRAM is required to sustain maximum performance. Recall that Membrain supports a maximum of 1TB secondary storage. This would require 128GB DRAM for best cache performance. A 32GB/256GB DRAM/secondary store configuration would work for smaller applications.

If your server has a limited amount of DRAM, you may increase the secondary storage to DRAM ratio, but the maximum performance will be limited due to higher cache miss rates.

Due to its extremely high application traffic capacity, Schooner Membrain™ may be required to support large numbers of network connections. Schooner Membrain™ supports a maximum of 400K simultaneous network connections.

Support for large numbers of connections coupled with very large objects (greater than 100KB) requires transient DRAM buffering. This can lead to swapping and therefore large performance losses. Reduce the DRAM allocation for Schooner Membrain™ should this occur.

CPU Core Allocation

Schooner Membrain™ utilizes a proprietary, lightweight threading mechanism to ensure the fastest response to system and IO events. This requires exclusive usage of a set of CPU cores by Membrain.

In general, Schooner recommends that 75% of available CPU cores be assigned to Membrain. As with DRAM, you may assign few CPU resources at the expense of maximum performance.

Network Interface Allocation

Schooner Membrain™ supports two types of network traffic:

- Membrain application traffic.
- Inter-node replication traffic.

Application traffic can be supported by multiple 1GE network interfaces, a set of bonded 1GE interfaces or a 10GE network interface. The number and type of application network interfaces assigned is dependent on your application requirements. Schooner Membrain™ can fully saturate a 10GE network interface with responses to Membrain application requests.

Schooner recommends the highest network bandwidth available for replication traffic. Schooner Membrain™ supports synchronous replication and the lower the replication latency, the higher the overall system performance. A 10GE network interface or a bonded set of 1GE interfaces may be used for replication.

Typical Schooner Membrain™ Configuration

During initialization, you will be asked to assign CPU, DRAM and secondary storage to Schooner Membrain™. The amount of each system resource allocated depends on the performance and capacity needs of your application.

A typical, high-performance, high-capacity installation might use the following Membrain configuration:

- 6 of 8 CPU cores (75%)
- 128GB DRAM
- 1TB secondary storage
- 10GE network interface or bonded 1GE interfaces

Small Schooner Membrain™ Configuration

For evaluation systems, you may use a small Membrain configuration:

- CPU core
- 4 GB DRAM
- 32 GB secondary storage
- 1GE network interface

Storage System Persistence

Storage system persistence guarantees can vary among devices. For example, some storage devices and controllers support high-speed caches in order to improve performance. If the devices do not provide a guaranteed way to ensure that cached objects are written to persistent storage, the system must execute the operations required to guarantee persistence.

In particular, certain solid-state storage devices support on-board caches but require execution of explicit sync commands in order to force cached objects to persistent storage.

The Schooner Membrain™ CLI supports a command, "`storage_sync`", that tells the Membrain server to issue the appropriate device cache flush commands whenever a "`sync`" or "`sync_all`" command is issued.

Chapter 3. The Schooner Administrator

Starting the Administrator

Point your browser to http://server_ip/admin

Login using the default credentials:

- User: “admin”
- Password: “admin”

Screen Layout

The layout of the Administrator screen is shown in Figure 3-1.

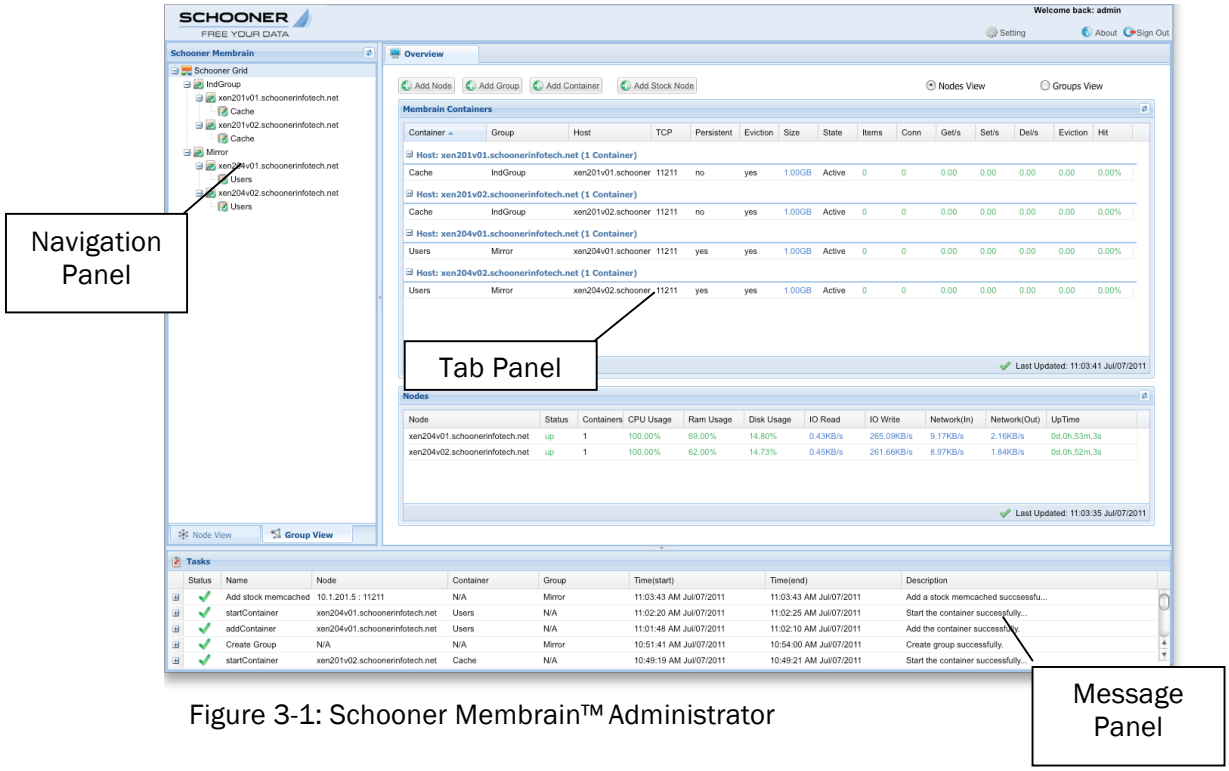


Figure 3-1: Schooner Membrain™ Administrator

Navigation Panel

The *Navigation Panel*, on the left side of the console, is where you choose the entity that you wish to manage depending on the view (node or group) that is presented:

- The grid (all servers joined in a Schooner Membrain™ network).
- A server.
- A Membrain or Memcached instance.
- A Schooner Membrain™ mirror group.

Tab Panel

The *Tab Panel* displays all of the information and functions related to the entity selected in the *Navigation Panel.* Some of the tabs are common across entities.

Message Panel

The *Message Panel* at the bottom of the screen displays status messages from actions you have started, such as adding a node, creating a Schooner Membrain instance, etc.

The Administrator issues commands asynchronously; you may perform more than one task at the same time. The state of each task will be displayed in the *Message Panel*. Detailed information about each task can be found by expanding the message line.

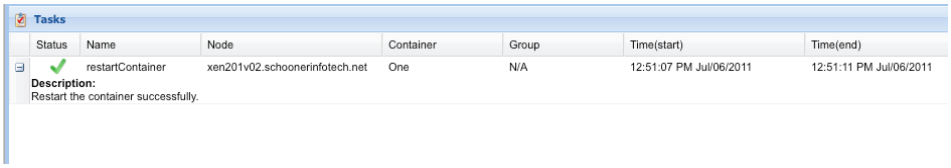


Figure 3-2: Message Panel

Grid

The *Grid Overview* is the first tab displayed after you log into the Administrator

Overview Tab

When you first login, the *Tab Panel* displays all of the Schooner Membrain containers and nodes (servers) configured in the grid. The container list shows various settings and metrics for each:

- Group: the mirror group this container belongs to.
- Host: the server on which this container runs.
- TCP: the TCP port assigned to this container.
- Persistent: persistence flag setting.
- Eviction: eviction flag setting.
- Size: container size in GB.
- State: container state (active, inactive).
- Items: the number of objects in the container.
- Conn: number of connections.
- Get/s: the container get rate.
- Set/s: the container set rate.
- Del/s: the container delete rate.
- Eviction: the container eviction rate.
- Hit: the container hit rate.

The node list shows the following metrics for each node:

- Status: up or down.
- Container: the number of Schooner Membrain containers configured.

- CPU Usage: total CPU utilization.
- RAM Usage: total DRAM utilization.
- Disk Usage: total secondary storage utilization.
- I/O Read: total secondary storage I/O read rate in KB/s.
- I/O Write: total secondary storage I/O write rate in KB/s.
- Network In: total network inbound bandwidth.
- Network Out: total network outbound bandwidth.
- Uptime: system uptime.

Functions

In the *Grid Overview* tab you can perform the following actions:

- Add a node (server).
- Add a group (mirror group).
- Add a Schooner Membrain container.
- Add a stock Memcached instance.

Node

The node tabs display server state information, performance metrics and Schooner Membrain container configuration.

Node Overview

The *Node Overview* tab displays various node functions, performance metrics, network interface status and file system status.

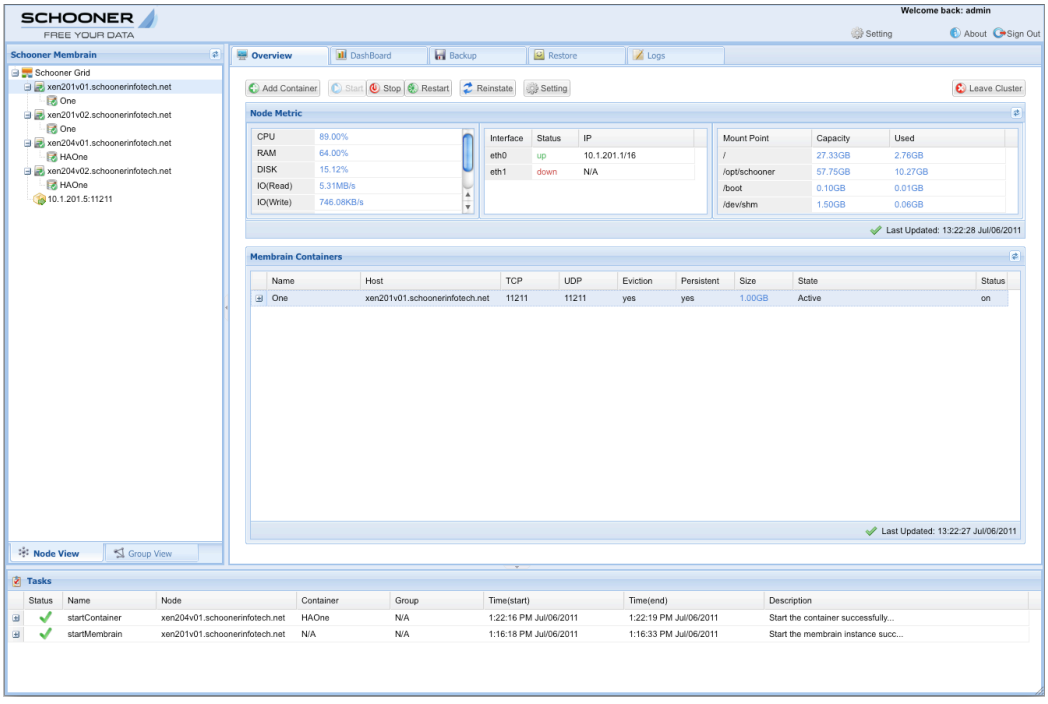


Figure 3-3: Node Overview

The Node Metrics sub-panel displays the following metrics:

- CPU Usage: total CPU utilization.
- RAM Usage: total DRAM utilization.
- Disk Usage: total secondary storage utilization.
- Storage I/O: total I/O read and writes rates in KB/s.
- Network interface:
    - Status: up or down.
    - Interface and gateway IP addresses.
- File systems:
    - Mount point, capacity, storage used.

The Membrain Containers sub-panel lists all Schooner Membrain containers belonging to this node:

- Container and host names.
- TCP/UDP port number.
- Eviction and persistence enabled.
- Size.
- State (active, inactive).
- Status: (on, off).

Each node supports a single Schooner Membrain™ instance. The Node Overview panel supports the following instance functions:

- Add a container to the instance.
- Start instance.
- Stop instance.
- Restart instance.
- Reinstate instance.
    - If both instances of a mirrored group should fail, one of the instances must be designated as the recovery master. Use the reinstate button to select the instance to be used as the recovery master.
- Virtual IP (VIP) and message interface configuration.

Node Dashboard

The Node Dashboard screen displays system performance graphs for server metrics.

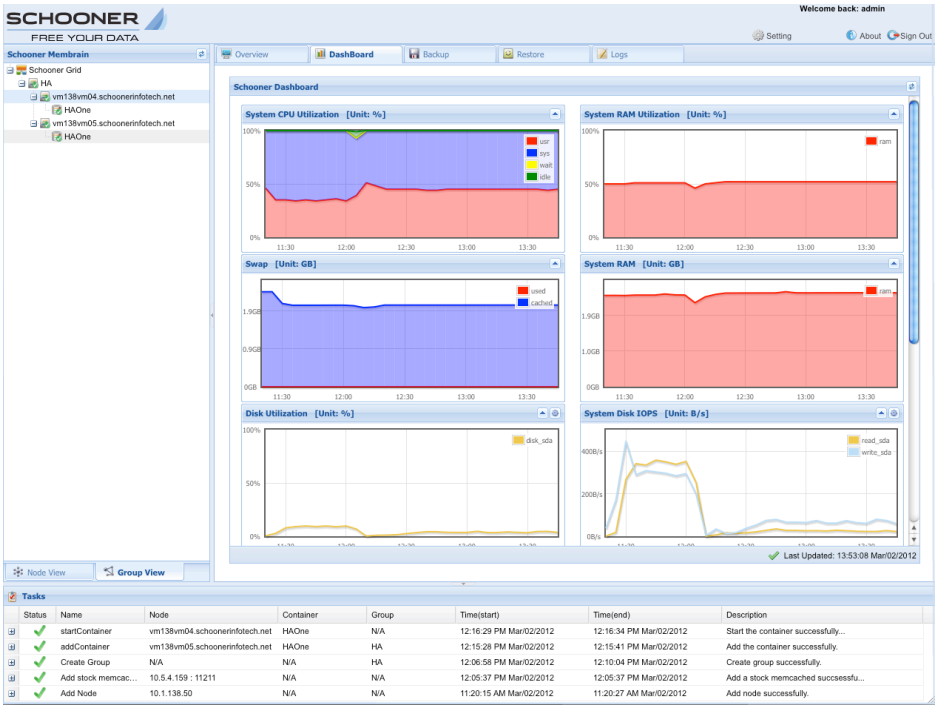


Figure 3-4: Node Dashboard

Container Screens

The container tabs provide interfaces to configure, control, archive and display performance metrics for Schooner Membrain containers.

Container Overview

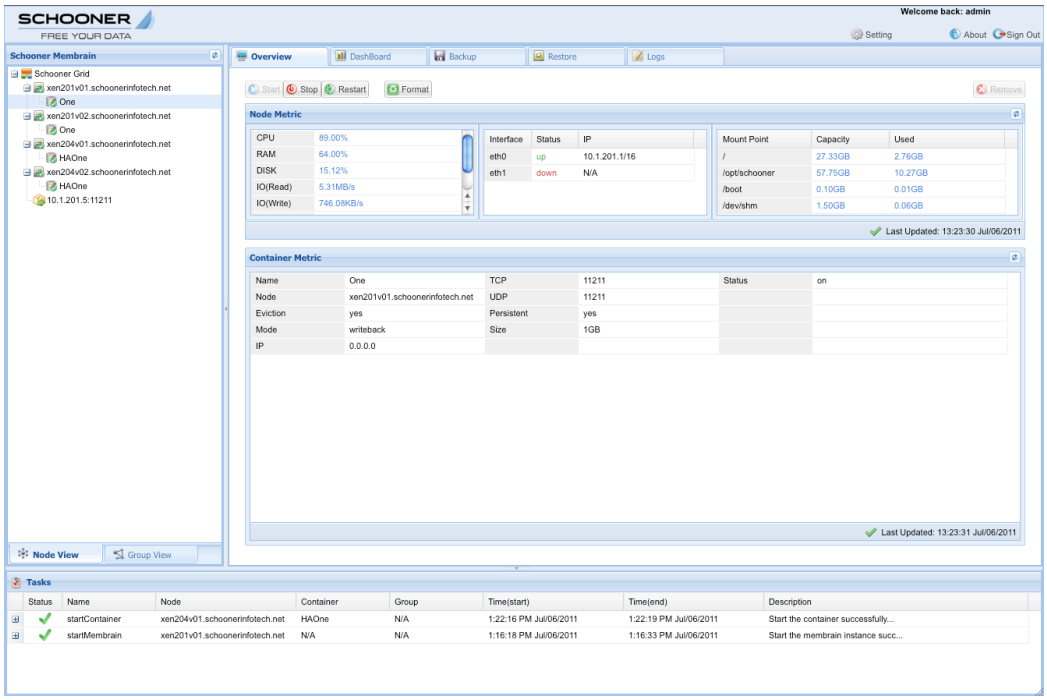


Figure 3-5: Container Overview

The *Container Overview* tab provides access to the following functions:

- Schooner Membrain Control:
  - Start a container.
  - Stop a container.
  - Restart a container.
  - Format a container.
- Schooner Membrain Archive:
  - Backup a container.
  - Restore a container.

The Node Metrics sub-panel is the same as displayed in the Node Overview tab.

The Container Metric sub-panel shows the following metrics for the container:

- Name: container name.
- Node: the node hosting this container.
- Eviction: eviction enabled.
- Mode: cache mode (write-back, write-through).
- IP: IP addresses assigned to this container.
- TCP: TCP port assigned to this container.
- UDP: UDP port assigned to this container.
- Persistent: persistence enabled.
- Size: container size.
- Status: on or off.

Dashboard

The *Dashboard* tab displays Schooner Membrain container performance charts.

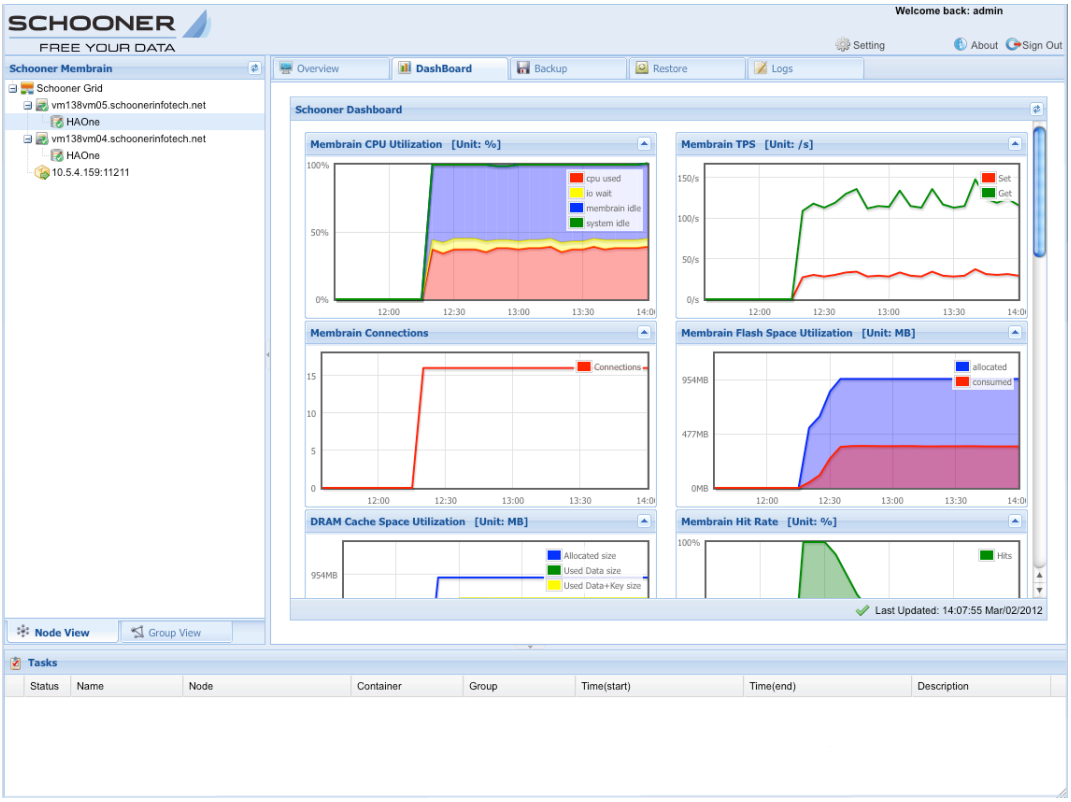


Figure 3-6: Container Dashboard

Backup

The *Backup* tab displays all of the container backup tasks and their status.

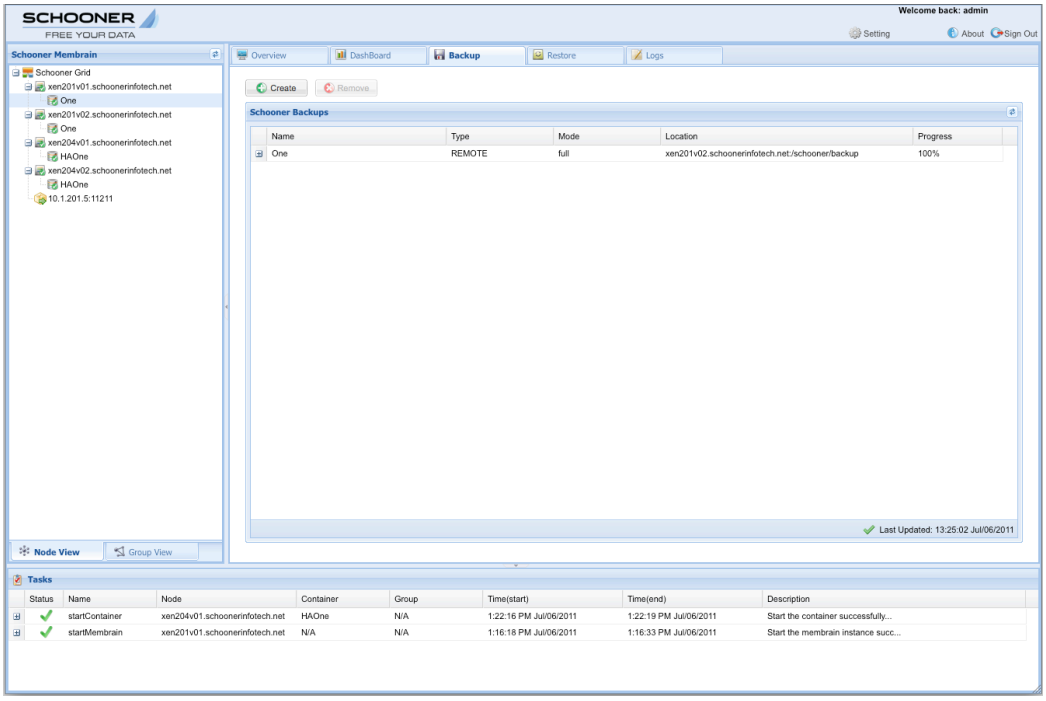


Figure 3-7: Backup Status

Restore

The *Restore* tab displays all of the container restore tasks and their status.

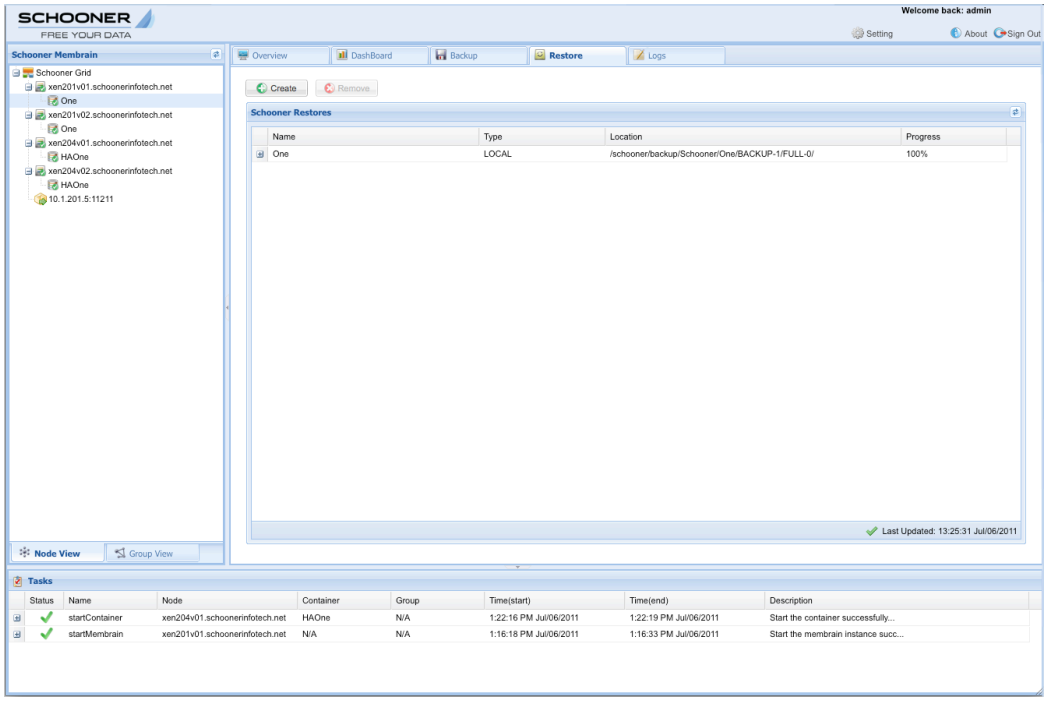


Figure 3-8: Node Restore

Logs

The *Logs* tab displays system and Membrain logs.

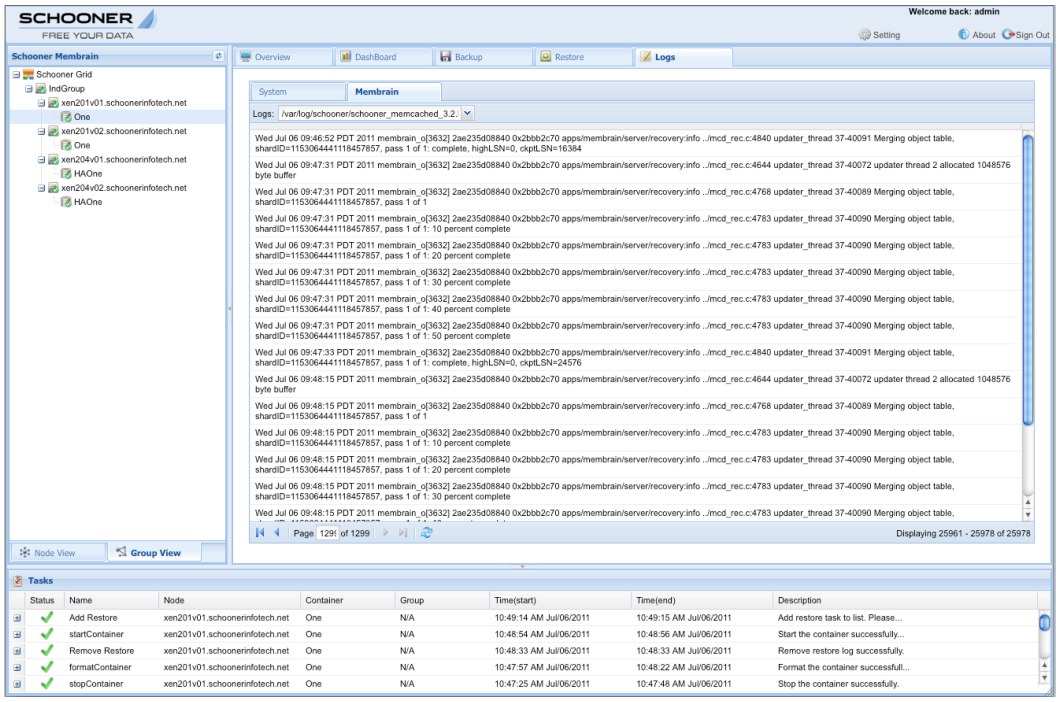


Figure 3-9: Logs

Chapter 4. Management Tasks

After initial setup with the First Time Wizard (FTW), an operational Membrain instance will be created. This chapter shows you how to manage Schooner Membrain using the GUI and CLI.

Types of Schooner Membrain Containers

Schooner Membrain containers are mainly defined by their eviction and persistence attributes. Usage of containers is dependent on these attributes.

Table 4-1 briefly describes the main attributes of each type of Schooner Membrain containers in terms of eviction and persistence.

Table 4-1: Schooner Membrain Containers

Container Type	Eviction	Persistence	Usage
Standard Cache	Yes	No	Same as standard Memcached instance
Standard Non-Evicting Cache	No	No	Same as standard Memcached with –M option
Schooner Persistent Cache	Yes	Yes	A persistent cache container ensures cached data is available on restart. Avoids performance-degrading multi-hour to multi-day cache warm-up periods
Schooner Store Mode	No	Yes	With Schooner Store Mode a container can be used as the permanent store of data. It can be used in place of a database to maintain the permanent copy of the data.

Standard Cache Mode

A standard cache mode container has the following attributes:

- Eviction is enabled
  - This means that when the container is filled, room for new objects will be created by the automatic removal of current objects by the Schooner Membrain server.
  - Flushing of objects to flash storage is controlled by the writeback setting.
  - Writeback: objects will be flushed only on explicit request or when the system decides to flush. This mode yields the best performance at

the expense of potential inconsistencies between cached and stored values.
    - Writethru: objects are written to flash when they are created or updated. This mode yields the best consistency between cached and stored values at the expense of some performance.
- Persistence is disabled
    - Container objects will be removed if Schooner Membrain is restarted.
    - This means that the cache will be "cold" on server restart, i.e., it must be re-filled by the application.
- Replication
    - Cache containers may be replicated.
- Backup
    - Cache containers may not be backed up.

Standard Non-Evicting Cache Mode

A standard non-evicting cache mode container has the following attributes:

- Eviction is disabled
    - This means that when the container is filled, objects must be explicitly removed by the application to make room for new objects.
    - Non-evicting caches are write-thru, i.e., all cache object creates and updates are immediately written to flash.
- Persistence is disabled
    - Container objects will be removed if Schooner Membrain is restarted.
    - This means that the cache must be re-filled on restart.
- Replication
    - Non-evicting cache containers may be replicated.
- Backup
    - Non-evicting cache containers may not be backed up.

Schooner Persistent Cache Mode

A Schooner Persistent Cache mode container has the following attributes:

- Eviction is enabled
    - This means that when the container is filled, room for new objects will be created by the automatic removal of current objects.
    - Flushing of objects to flash storage is controlled by the writeback setting.
    - Writeback: objects will be flushed only on explicit request or when the system decides to flush. This mode yields the best performance at the expense of potential inconsistencies between cached and stored values.
    - Writethru: objects are written to flash when they are created or updated. This mode yields the best consistency between cached and stored values at the expense of some performance.
- Persistence is enabled
    - Container objects will remain in flash if Schooner Membrain is restarted.
    - This means that the cache will be "warm" on restart.
    - If replication is enabled, objects in the restarted cache will be up-to-date. The replication recovery process ensures that the latest updates are propagated to the restarted container.

        - If replication is disabled, the application might be required to synchronize the cache with the permanent copy of the data. This step is application dependent and may or may not be required.
    - Replication
        - Warm-start cache containers may be replicated.
    - Backup
        - Warm-start cache containers may be backed up.

Schooner Store Mode

A Schooner Store mode container has the following attributes:

- Eviction is disabled
    - This means that when the container is filled, objects must be explicitly removed by the application to make room for new objects.
    - Non-evicting caches are write-thru, i.e., all cache object creates and updates are immediately written to flash
- Persistence is enabled
    - Container objects will remain in flash if Schooner Membrain is restarted.
    - This means that the cache will be "warm" on restart.
    - If replication is enabled, objects in the restarted cache will be up-to-date. The replication recovery process ensures that the latest updates are propagated to the restarted container.
    - If replication is disabled, the application might be required to synchronize the cache with the permanent copy of the data. This step is application dependent and may or may not be required.
- Replication
    - Store containers may be replicated.
- Backup
    - Store containers may be backed up.

About Data Persistence

Setting Persistence to **YES** will cause data stored in flash memory to be persistent across a reboot, power loss, or node failure. However, because your Schooner Membrain uses DRAM memory as a cache front-end, some "persistent" data in flash memory may actually be "stale." The Schooner-specific `sync/sync_all` commands allow applications to periodically force transient object data to flash (see page 4).

Write-Back Caching vs. Write-Through Caching

Write-back caching mode is a feature that allows writes to be performed within the Membrain in-memory cache without performing a flash write. Data modified in this way is not written to flash until one of the following occurs:

- The object is LRU replaced by the in-memory cache.
- The object is explicitly flushed by the application using `"sync"` or `"sync_all"`.
- The background flush process flushes the object automatically.

The write-back policy can improve performance by reducing the number of writes to flash. Workloads that benefit from write-back mode have these three characteristics:

- A high write rate.
- A high overwrite rate.

- High locality of overwrites in the in-memory cache.

Only eviction-mode containers can be configured with write-back mode.

Containers not configured for write-back mode use a write-through in-memory caching policy. Write-through mode performs a flash write for every Membrain write operation. This ensures that flash storage is always up-to-date. For workloads with a high overwrite rate, however, it generates more traffic to flash than write-back mode, which can reduce performance.

With write-back mode there may be objects for which the up-to-date value is in the in-memory cache but not yet written to flash.

The cache has an automatic background flush process that ensures that all cache data is periodically written to flash. There will always be some lag, however, between a Membrain write operation and the time at which the data is pushed to flash.

With a write-back container, an application can always force cached values to flash using the Schooner-specific "sync" or "sync_all" commands (see Ensuring Current, Persistent Data in Chapter 2).

Note that it may take hours for a sync_all operation to complete if a large portion of the in-memory cache (approximately 30GB of data) must be written out to flash.

Using Membrain Containers

A standard cache container can be used in the same way as a standard configuration Memcached instance is used.

A standard non-evicting cache container is used in the same manner except that, on Schooner Membrain restart, the cache will contain warm data. If required, the application should first synchronize the cache with the permanent store by sending updates to the cache that have occurred since the cache went down.

A non-eviction cache container is used in the same way that a standard Membrain run with the “-M” option is used. This option requires the application to make room for new objects by removing old objects when the cache is filled.

A store container can be used as the permanent store of the data. That is, instead of a database or file system, the store container can be used to maintain the permanent copy of the data.

Container Usage Specifications

Starting/Stopping Containers

Containers may be started and stopped individually without starting or stopping the Schooner Membrain server itself.

When a container is restarted, it may be necessary for applications to synchronize the cached data objects with the versions in permanent storage.

Consistent Hashing

Distribution of data in the Memcached protocol is determined by a function in the Memcached client. Typically, this function uses a consistent hashing function such as the Ketama algorithm.

This algorithm is very efficient in that it takes very few cycles for the Memcached client to determine the location of a data object among the container pool.

Note that if the set of container addresses (VIP:port) that defines a pool is modified in any way (additions, deletions or modification of VIP:port addresses) the distribution of objects across the container pool will potentially change.

For both standard Memcached and Schooner Membrain changes to the client-side pool configuration may require manual redistribution of cache objects.

The reason is that if the newly updated pool specifies that object "A" which previously lived on server "A" should now be moved to server "B", any attempts to access object "A" on server "B" will result in a cache miss until the object is inserted into the cache on server "B".

Also, the old copy of object "A" residing on server "A" will take up space until it is explicitly deleted by the application.

Administration Using the Schooner Administrator GUI

Managing Containers and Instances

Each Schooner Membrain™ node supports a single instance of Schooner Membrain™ and each instance may support from 1 to 128 containers.

A Schooner Membrain™ instance is created when the first container is created (see below).

Start an Instance

To start a Schooner Membrain™ instance:

1. Select the instance you wish to start in the navigation panel.
2. Click on the "Start" button:

Figure 4-1: Start Instance Button

3. The Schooner Membrain™ instance executes.

Stop an Instance

To stop a Schooner Membrain™ instance:

1. Select the instance you wish to stop in the navigation panel.
2. Click on the "Stop" button:

Figure 4-2: Stop Instance Button

3. The Schooner Membrain™ instance is stopped.

Restart an Instance

To restart a Schooner Membrain™ instance:

1. Select the instance you wish to restart in the navigation panel.
2. Click on the "Restart" button:

Figure 4-3: Start Instance Button

3. The Schooner Membrain™ instance is restarted.

Reinstate an Instance

To reinstate a Schooner Membrain™ instance (select it as the recovery master for a failed mirrored group):

1. Select the instance you wish to restart in the navigation panel.
2. Click on the "Reinstate" button:

Figure 4-4: Reinstate Instance Button

Add a Container

To add a container:

1. Select the node to host the container.
2. Click "Add Container":

Figure 4-5: Add Container Button

3. The "Add Container" dialog displays:

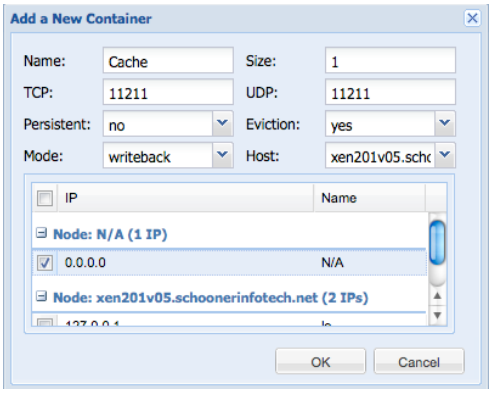


Figure 4-6: Add Container Dialog

- Enter the container name.
- Enter the TCP/UDP port numbers.
- Set persistence and eviction.
- Select cache writeback or writethru mode.
- Select the IP addresses for the container.
- Click "Add".

4. The message panel will display the status of the container add operation.
5. After the container has been created, you must click the "Start" button in order to activate the container.

Start a Container

To start a container:

1. Select the container.
2. Click the “Start” button:

Figure 4-7: Container Start Button

3. The message panel will display the status of the container start.

Stop a Container

To stop a container:

1. Select the container.
2. Click the “Stop” button:

Figure 4-8: Container Stop Button

3. The message panel will display the status of the container stop.

Restart a Container

To restart a container:

1. Select the container.
2. Click the “Restart” button:

Figure 4-9: Restart Container Button

3. The message panel will display the status of the container restart.

Format a Container

To format a container and delete all of its data:

1. Select the container.
2. Click the “Format” button:

Figure 4-10: Format Container Button

3. The message panel will display the status of the container format.

Remove a Container

You can remove any container that is no longer needed. Keep in mind that you must stop a container before you attempt to delete it.

To delete a container:

1. Select the container.
2. Click the “Stop” button:

Figure 4-11: Stop Container Button

3. Click the “Remove” button:

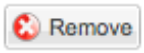


Figure 4-12: Remove Container Button

4. The message panel will display the status of the container removal.

Manage Nodes

Add Nodes to Schooner Cluster

Before you can add a node to a cluster, you must complete the initial setup of the node using the Schooner First Time Wizard. After that, you can join the node with another node to form a cluster or simply add it to an existing cluster.

Keep in mind that you cannot add the same node to more than one cluster. So if you want to add a node that is already in a cluster to another cluster, you must remove it from the existing cluster before you can add it to another cluster or form a new cluster with another node.

To add a node to the cluster:

1. Click on “Schooner Grid” in the navigation panel.
2. Click on the “Add Node” button:

Figure 4-13: Add Node Button

3. The “Add Node” dialog displays:

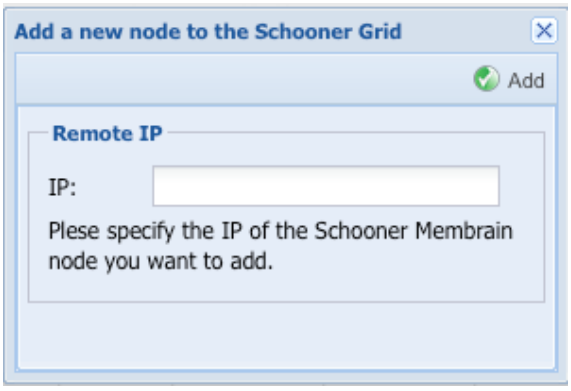


Figure 4-14: Add Node Dialog

   - Enter the IP address of the node to be added.
   - Click “Add”.
4. The message panel will display the status of the node add.
5. Once added, the node will be displayed in the “Node View” navigation panel.

Remove Nodes from Schooner Cluster

Before a node can be removed, all mirror groups residing on that node must be first removed.

To remove a node from the cluster:

1. Click on the node in the “Node View” navigation panel.
2. Click on the “Leave Cluster” button:

Figure 4-15: Leave Cluster Button

3. The message panel will display the status of the leave cluster operation.

Manage Replication

When creating a mirrored group, make sure to keep the following in mind:

- The two nodes must be on a common network subnet or there must be a routable path between them.
- Since replication is synchronous, the network latency between mirrored nodes will dictate the performance of replication.
- Existing containers will be destroyed in a newly formed mirror group.
- You must stop the Membrain instances involved before creating or deleting a mirrored group.

Create Mirrored Groups

Mirrored groups support the Schooner Membrain™ replication feature.

To create a mirrored group:

1. Click on the “Schooner Grid” icon.
2. Click on the “Add Group” button:

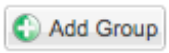


Figure 4-16: Add Group Button

3. The “Add Group” dialog displays:

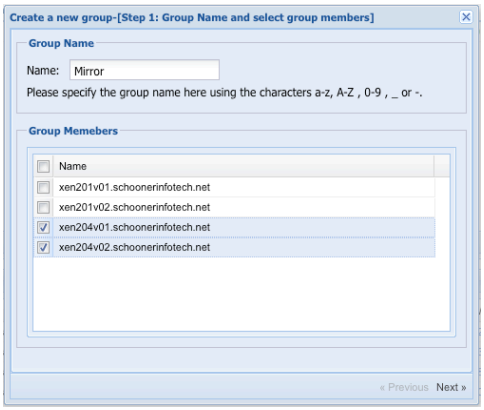


Figure 4-17: Add Group Dialog

- Enter the name of the mirror group.

- Select two nodes to form the mirror group.
- Click “Next”.

4. The “VIP” dialog displays:

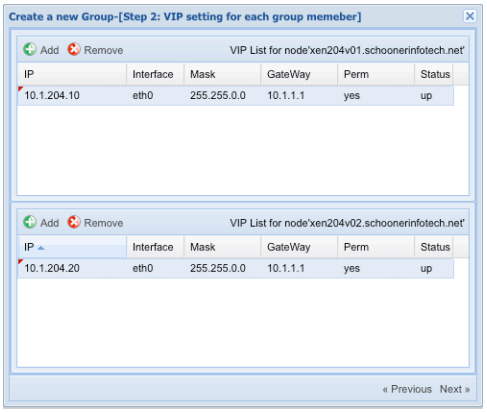


Figure 4-18: VIP Dialog

- Click “Add” to enter VIP configuration for each node.
- Click “Next”.

5. The “Admin Interface” dialog displays:

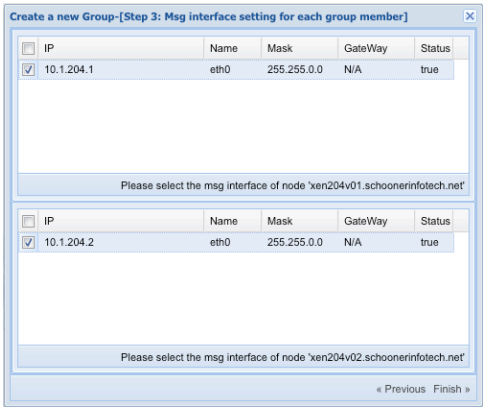


Figure 4-19: Admin Interface Dialog

- Select the admin interface for each node.
- Click “Finish”.

6. The message panel will display the status of the mirror group create.

Delete Mirrored Groups

To delete a mirrored group:

1. Click on the icon of the mirrored group.
2. Click on the “Remove Group” button:

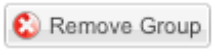


Figure 4-20: Remove Group Button

3. The message panel will display the status of the mirrored group delete.

4. On successful completion, the members of the mirrored group will be assigned to the independent pool. Their containers will be left intact but no longer in replication mode.

Add Stock Memcached Nodes to Schooner Cluster

You can monitor a stock memcached node with the *Schooner Administrator GUI*. To add a stock memcached node:

1. Click the "Add Stock Node" button:

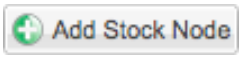


Figure 4-21: Add Stock Node Button

2. Enter the IP address of the stock node and the memcached port:

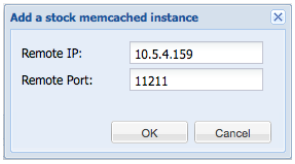


Figure 4-22: Add Stock Node Dialog

3. Click "OK".
4. The stock memcached node will appear in the navigation panel.
5. To monitor the stock memcached node, click on its icon.
6. The stock memcached "stats" panel will appear:

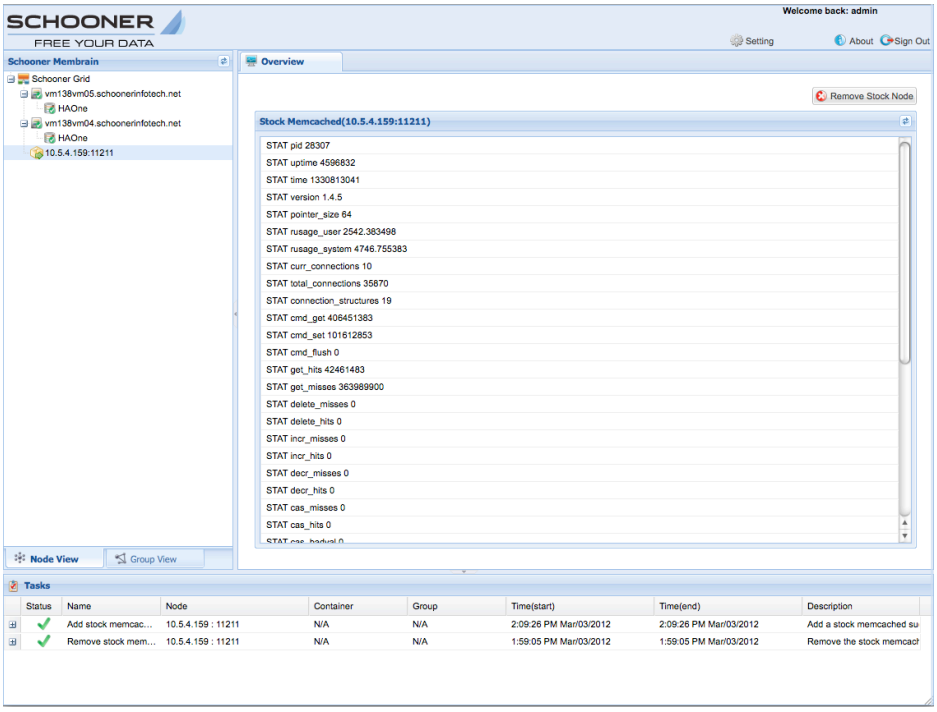


Figure 4-23: Stock Node Monitor

To remove a stock memcached node:

1. Click the “Remove Stock Node” button:

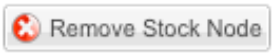


Figure 4-24: Remove Stock Node Dialog

2. The stock memcached node will be removed from the *Schooner Administrator*.

Container Data Backup/Restore

Backup

Schooner Membrain™ supports two types of backup operations: full backup and incremental backup. A full backup is a logical copy of all objects in a container. An incremental backup is a logical copy of objects in a container that are new or have changed since the previous backup, as well as a logical representation of deleted objects. A full backup is taken to start a new backup "series", which contains the full backup, plus zero or more incremental backups. There is no limit on the number of incremental backups that may be taken in a backup series (though there are practical considerations).

Restore

Restoring a backup is the process of replaying backup streams to the server. A backup can be restored to any container. The target container must already exist at the time of the backup because the restore process does not create a container. The user must also make sure that there is enough space in the container to hold all the restored objects.

Users can restore to any arbitrary backup within a series, but all backups within the selected series, from the full backup up to, and including, the desired backup, are restored as part of a single restore operation.

Backup and Restore Procedures

Backup operations can only be performed on persistent containers. Depending on your need, you can perform either a full backup or an incremental backup. If you plan to do a combination of full and incremental backups, you must start with a full backup first, followed by one or more incremental backups, depending on your situation. You can start a backup operation even while the clients are accessing the container.

Backup Container

To back up a container:

1. Click on the container in the “Navigation Panel”.
2. Click on the “Backup” tab.
3. Click the “Create” button”:

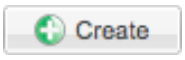


Figure 4-25: Create Button

4. The “Backup Job” dialog displays:

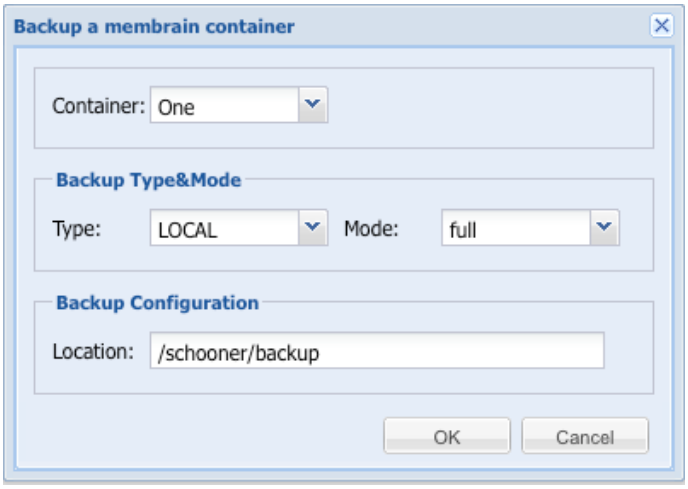


Figure 4-26: Backup Job Dialog

- Select the container to backup.
- Select the type of backup, local or remote.
  - If remote, select the node.
- Select the mode, full or incremental.
  - If incremental, select the full backup file associated with this backup.
- Enter the directory path for the backup.
- Click "OK".

5. The message panel displays that status of the backup job.

Restore Container

To restore a container from backup:

1. Click on the container in the "Navigation Panel".
2. Click on the "Restore" tab.
3. Click the "Create" button":

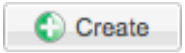


Figure 4-27: Create Button

4. The "Restore Job" dialog displays:

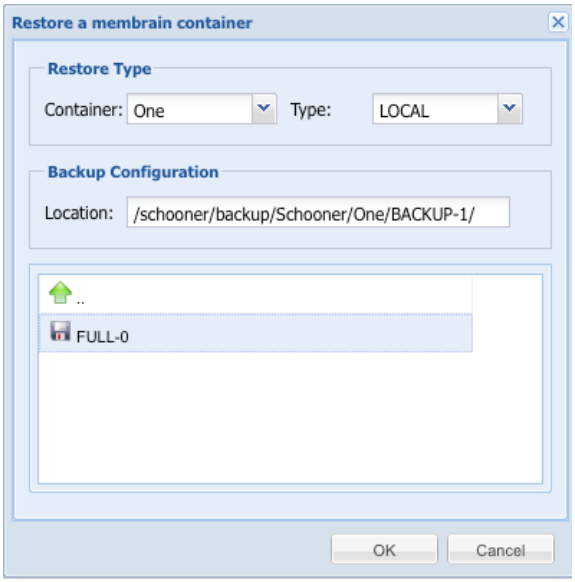


Figure 4-28: Restore Job Dialog

- Select the container to restore.
- Select the type of backup, local or remote.
  - If remote, select the node.
- Select the backup file.
- Click "OK".

5. The message panel displays that status of the restore job.

Administration Using the CLI

Manage a Membrain Server

You can perform basic management functions on application instances using the CLI:

1. At the > prompt, execute `'enable'`.
2. At the # prompt, execute `'configure terminal'`.
3. At the (config) # prompt, execute `'membrain'`.
4. Use the start or stop menu to start or stop the Membrain server.

Select the Correct Version of Membrain

The Schooner Membrain™ version 4.0 release supports the open-source Memcached software versions 1.2.6 and 1.4. Some features in the open-source Memcached are version-specific. In order to take advantage of features that are available only in one version of the open-source Memcached, you must make the Schooner Membrain™ server compatible to that version of the open-source Memcached software.

To switch the Schooner Membrain™ server to a desired version of the open-source Memcached:

1. Navigate to the CLI `'membrain'` menu.

2. Using the `'switch'` command set the Memcached version selecting `1.2` or `1.4`.

Managing Containers

Add a Container

To add a container:

1. Navigate to the CLI `'membrain'` menu.
2. Using the `'container'` command add a new container.
3. Use the command options as described in Table 4-2 below.

Table 4-2: Configuring Container Attributes

Parameters	Description
Name	The name of the container, which can consist of any number of alphanumerical characters, plus hyphen (-) and underscore (_). No space or special character is allowed.
Capacity	The amount of the flash memory (in GB) allocated to the container.
Eviction	Select either of the following: ▪ **YES** – If selected, the server will start discarding old data, when the container is full, to make room for new data. You will then need to select between write-back caching and write-through caching. ▪ **NO** – If selected, the container will be in non-evicting or "store" mode. The server will not discard any data. Instead, it will stop accepting new data once the container is full. If eviction is disabled, then the container can only use write-through caching. **Note:** See below.
Mode	Choose between write-back and write-through caching. This option is available only in eviction mode. ▪ **WriteBack** – allows writes to be performed within the Membrain in-memory cache without performing a cache write. ▪ **WriteThru** – performs a flash write for every Membrain write operation. Note: See the section "Write-back Caching vs. Write-through Caching" following this section.
TCP Port	Assign a TCP port number to the container. **Note:** Do not use any port number from 51350 through 51600 because all ports within that range are reserved for Schooner enterprise services.
UDP Port	Assign a UDP port number to the container. **Note:** Do not use any port number from 51350 through 51600 because all ports within that range are reserved for

	Schooner enterprise services.
Persistence	Select either of the following: ▪ **YES** – If selected, the server will retain the data in the flash when the system is shut down so that it will be available upon system reboot. ▪ **NO** – If selected, the server will not retain any data when it shuts down. Data will be lost upon system reboot.
IP	Use `'anyip'` to listen on all IPs or a comma-separated list of specific IP addresses. **Note:** If you select `'anyips'`, then clients can access the container from all the IP addresses available on the node. If you want to make the node accessible only through certain IPs, then you must specify them from the list of IP addresses under the node. It is important to know that the term "IP" covers all types of IP addresses, including all interface IPs as well as the VIPs and VLAN IPs that you have created,

Delete a Container

You can delete any container that is no longer needed. Keep in mind that you must stop a container before you attempt to delete it.

To delete a container:

1. Navigate to the CLI `'membrain'` menu.
2. Stop the container using the `'container stop'` command.
3. Delete the container using the `'container delete'` menu.

Manage Nodes

Add Nodes to Schooner Cluster

Before you can add a node to a cluster, you must complete the initial setup of the node using the Schooner First Time Wizard. After that, you can join the node with another node to form a cluster or simply add it to an existing cluster.

Keep in mind that you cannot add the same node to more than one cluster. So if you want to add a node that is already in a cluster to another cluster, you must remove it from the existing cluster before you can add it to another cluster or form a new cluster with another node.

To add a node to the cluster:

1. Navigate to the CLI `'cluster'` menu.
2. Using the `'join'` command, specify the IP address of a node in an existing cluster.

Remove Nodes from Schooner Cluster

To remove a node from the cluster:

1. Navigate to the CLI `'cluster'` menu.

2. Use the 'leave' command to remove the node from its cluster.

Manage Replication

Create Mirrored Groups

To create a mirrored group:

1. Navigate to the CLI 'membrain' menu.
2. Use the 'mirrorgroup' command to create a mirrored group.
3. Use the command options shown in Table 4-3.

Table 4-3: Creating a Mirror Group

Parameter	Description
Add/Delete	Add or delete a mirror group.
Group Name	The name of the mirror group, which can only consist of alphanumeric characters, plus hyphen (-) and underscore (_). No special characters or spaces are allowed.
Node 1	Specify the hostname of the first node to add to the mirror group.
Node 2	Specify the hostname of the second node to add to the mirror group.

Back Container

To back up a container:

1. Log into the CLI.
2. Execute the following commands:

```
1) > enable
2) # config terminal
3) (config)# Membrain
4) (membrain)# container backup full
5) (Membrain)# container backup incr
```

Note: 'backup full' means a full backup, whereas 'backup incr' means an incremental backup.

3. Make sure to remember the backup directory.

To view container backup files:

1. Log into CLI.
2. Execute the following command:

```
(membrain)# show backups
```

To restore container backup:

1. Log into CLI.

2. Execute the following commands:

   ```
   (membrain)# container restore CNAME local FILE,
   ```

   where CNAME is the name of the container to be restored and FILE is the name of the backup file.

Chapter 5. System and Data Recovery

In the real world, incidents such as bad or corrupted configuration files and/or persistent data errors in Membrain may render your Schooner Membrain™ server non-operational. So you need to perform recovery operations to bring the system back to service. System recovery involves the restore of system configuration and/or container data to a previous healthy state using backup files.

This chapter discusses the various scenarios where system recovery operations are required and the recovery procedures.

Recover Independent Nodes

This section discusses the procedures for the recovery of an independent node.

Recover an Independent Node

This is a situation in which a single, independent node (as opposed to a node in a mirrored pair) goes down and is not automatically recoverable. This can be caused by a bad or corrupted configuration file or data errors in the containers. The recovery operation in this case requires the use of configuration and container backup files that can restore the system to a previous, healthy operating state. So it is important that you back up your system and container data at a regular basis and keep the backup files handy just in case.

To recover an independent node:

1. Recover the container configuration using the steps outlined in Chapter 3.
2. Recover the data for each persistent container using the steps outlined in Chapter 3.
3. Non-persistent containers are not recoverable.

Automatic Recovery of a Malfunctioned Mirrored Node

This is a situation in which one of the nodes in a mirrored group temporarily goes down and then comes back to service. Since, in this case, the system configuration on the node is intact and the node is fully functional, no action is needed to recover the configuration on the node. All the user needs to do is to wait for the node to automatically recover Membrain data from its peer node.

Recovery of a Dead Node in a Mirrored Pair

This is a situation where one of the nodes in a mirrored group is completely dead. To bring it back to service, you must first manually restore system configuration on that node.

Here are the steps for recovering or replacing a dead node:

1. We have a mirrored pair, nodes A and B.
2. Assume node B is to be restored or recovered.

3. Backup container configuration from node A (node B has the same container configuration) using the CLI “container_conf” command.
4. Save these node B files:
   `/opt/schooner/membrain/config/memcached.properites.cluster`
   `/etc/sysconfig/network-scripts/admin_vips.conf`
5. Uninstall Membrain from node B (or replace the server).
6. On node B create the directory:
   `/opt/schooner/membrain/config`
7. Restore the files from step 3 into the directory created in 6.
8. Install and initialize Membrain.
9. Configure the admin interface if you are not using the default, eth0.
10. Use “show groups” to check the mirror status.
11. On node B, use CLI container_conf to restore the container configuration file from node A (backed up in step 2).
12. Node B will recover using data from node A.
13. When recovery is complete node B will be active.

Data Recovery upon “Double Failure”

Double failure refers to a situation in which both nodes in a mirrored group go down. By design, the nodes themselves are unable to recover persistent containers even though they are able to automatically recover non-persistent containers after they restart. User intervention is required to recover persistent containers on the nodes. This is because the system cannot determine which node has the most current copy of a persistent container.

To recover persistent containers after a double failure:

1. Both the nodes are up and running as a mirrored pair after they come up but persistent containers are turned off automatically.
2. Select one of the nodes as the recovery master say, node A. This should be the node with the most up-to-date container data. Lets call the peer node B.
3. Stop Membrain on node B.
4. Navigate to the CLI ‘membrain’ menu on node A and execute the command ‘reinstate’ to signify that this node A will be the recovery master.
5. Start Membrain on node B.
6. The Membrain server on node B will recover from node A and failover VIPs after the recovery is complete.

Data Recovery Phases

During data recovery, the recovering node copies data from the surviving node. The length of time it takes to complete the data recovery depends on a number of variables, including the amount of data to be recovered.

During data recovery, you cannot add, delete, start, or stop a container. Nor can you carry out such operations when either node is down.

Chapter 6. The Schooner CLI

The Schooner CLI (Command Line Interface)

The Schooner CLI provides all the commands that the user needs to perform all configuration and administration tasks. However, it important to note that the CLI can be used only after the node has been successfully set up using the Schooner FTW.

This chapter provides detailed descriptions of the CLI commands as well as reference information about how to use the commands to configure and manage Schooner Membrain™.

The chapter covers the following topics:

1. Start the CLI
2. CLI Operating Modes
3. Get Help
4. Put the CLI to Work
5. Configure System
6. Configure Membrain
7. View System Configuration and Status Information

Start the CLI

The CLI can be launched using a terminal program like PuTTY or Secure Shell (SSH).

To start the CLI:

1. Start PuTTY, SSH, or the system console.
2. Connect to Schooner Membrain™ using its IP address.
3. Log into Schooner Membrain™ using the administrator account name and password (both are "`mbradmin`" by default).
4. Wait for the following message to appear:

```
Welcome to Schooner Appliance for Membrain/NoSQL Command Line Interface (CLI)!
...
License Status :   (You have 94 hours to get a production license)
>
```

1. At the > prompt, execute 'enable'.
2. At the # prompt, execute 'configure terminal'
3. At the (config)# prompt, execute 'system', 'network', or 'membrain'.

CLI Operating Modes

The CLI has three different modes, which govern the way the CLI works as well as the commands that are available.

View-Only Mode

By default, the CLI starts in View-Only mode. In this mode, you cannot do anything other than viewing the existing configuration of the node. You cannot set or change the configuration of Schooner Membrain™ in this mode. You can tell that the CLI is in View-Only mode by the right arrow "`>`" at the end of the prompt string. Below is the CLI's main menu for View-Only mode.

```
> ?
enable      Enter to privileged mode to configure and manage the system
exit        Exit current mode and go back to previous mode
help        Get help information about how to use the CLI
history     Manage CLI command history
list        Print command list
ping        send echo messages
show        Show current system information
telnet      Open a telnet connection
traceroute  Trace route to destination >
```

Of all the command options in View-Only, the following two are worth mentioning:

- `enable` – turns the CLI to Enable mode, which is the transition stage to get into the Configure mode.
- `help` – gets help on how to use the CLI.
- `exit` – closes the CLI.

Enable Mode

You can turn the CLI from View-Only mode to Enable mode using the "`enable`" command (see above). The Enable mode is a required transitional step between View-Only mode and Configure mode. You cannot go directly from View-Only mode to Configure mode, without Enable mode in between. You can tell that the CLI is in Enable mode by the "`#`" prompt at the end of the prompt string. Below is the main menu of Enable mode.

```
# ?
configure   Configure the system
disable     Enter to non privileged mode to just view the system status
exit        Exit current mode and go back to previous mode
help        Get help information about how to use the CLI
history     Manage CLI command history
list        Print command list
ping        Send echo messages
show        Show current system information
telnet      Open a telnet connection
terminal    Set terminal line parameters
traceroute  Trace route to destination #
```

The following highlight some of the most important commands of Enable mode:

- `Configure (terminal)` - turns the CLI from Enable mode to Configure mode.
- `disable` – turns the CLI from Enable mode back to View-Only mode.

- `exit` – closes the CLI. (If you want to go back to View-Only mode from Enable mode, use the `'disable'` command.)

Configure Mode

While in Privileged mode, you must use the "`configure terminal`" command to turn the CLI to Configure mode, which offers all commands for viewing, configuring, and modifying the system settings on Schooner Membrain™. You can tell that the CLI is in Configure mode by the "`(config) #`" at the end of the prompt string. Below is the main menu of Configure mode.

```
(config)# ?
cluster    Configure cluster(credential/join/leave)
exit       Exit current mode and go back to previous mode
help       Get help information about how to use the CLI
history    Manage CLI command history
list       Print command list
membrain   Configure Membrain
memcached  Configure Memcached
show       Show current system information
system     System configuration(ntp,name,snmp,callhome ...)
(config)#
```

As shown in the above menu, once you are in Configure mode, you can use the following commands to configure and manage the Schooner Membrain™ system:

- `(config)# cluster` – configure the Schooner cluster `(cluster)#`
- `(config)# Membrain` – configure Schooner Membrain™ (`membrain)#`
- `(config)# Memcached` – configure Memcached (`memcached)#`
- `(config)# system` – configures the Schooner system `(system)#`

You can turn the CLI from Configure mode to Enable mode by a single execution of the "`exit`" command. Using the "`exit`" command twice lets you log out of the CLI. To return to View-only mode from Configure mode, first execute the "`exit`" command to get out of Configure mode and into Enable mode, and then execute the "`disable`" command to move into View-only mode.

Note: Currently, all user accounts have full access to both View-only mode and Enable/Configure mode. Password will be required to access Enable/Configure mode in the future.

Get Help

This section contains information on how to call for help on the CLI. It is suggested that you read this section before starting to use the CLI.

- You can type a question mark `'?'` at any time to call for help or information on how to proceed with the CLI.
- At any level of the CLI, you can type `'?'` at the prompt to display the main menu for that level, `'show ?'` to display all commands, or `'list'` to display all the commands plus their applicable arguments, if any.
- To configure and manage your Schooner , you must turn the CLI to Configure mode

  using the following series of commands:

  1) At the > prompt, execute `'enable'`.

2) At the # prompt, execute `'configure terminal'`.

3) At the (config) # prompt, execute `'system', 'membrain' or 'memcached'`.

4) Start managing the Schooner system using a command of your choice. For example,

   at the `(system) #` prompt, you can execute `'adminface eth0'` to set 'eth0" as the 's administration interface.

- You can find out the arguments needed to complete a command by typing a `'?'` in place of the arguments, one at a time. You can then add the arguments to the command as you go until it is completed.
- You can find out all the possible arguments of the 'show' command without description by pressing the <tab> key twice.
- You can let the CLI automatically complete a command by typing in the first few letters of a command and then pressing the <tab> key to complete each entry. For example, at the # prompt, you can execute `'c<onfigure> t<erminal>'`.
- If you see the error message 'Command Incomplete' after executing a command, it means that some of the required arguments for the command have not been entered. You can use `'?'` after the current argument to learn about the following argument.
- You can use the UP and the DOWN arrow keys to browse for previously entered commands.
- You can return to the help page at any time by executing the 'help' command.

The CLI Operation Workflow

After logging into the master node in your Schooner network system from the CLI, you must execute the following commands to turn the CLI from View-Only mode to Configure mode before you can start configuring and managing Schooner Membrain™.

1. Turn the CLI to Enable mode using the following command:
   `> **enable**`
2. Turn the CLI to Configure mode using the following command:
   `# **configure terminal**`
3. Start configuring Schooner system components using any of the following commands:

   1) `(config) # system`

   2) `(config) # membrain`

   3) `(config) # memcached`

   4) `(config) # cluster`

For initial Schooner system configuration, it is recommended that the user start with system configuration, followed by Membrain, and cluster configuration. This is because the configuration of some parameters in the Schooner system depends on the configuration of some other parameters. In other words, the configuration of certain parameters depends on the configuration of some other parameters. The Schooner CLI is very intelligent in this respect in that it requires the user to follow a certain workflow while configuring the system. If you fail to follow the sequence of configuration, you will get an error message, which will eventually lead you to the correct path. This is also true when you modify your existing configuration.

Configure the System

This section describes the CLI commands for configuring the system part of your Schooner system, which is Schooner Membrain™, also called server or node in this document. You can start

configuring the system part of the Schooner system by using the following commands on the CLI:

1. At the ">" prompt, execute the command "enable".
2. At the "#" prompt, execute the command "configure terminal".
3. At the (config) # prompt, execute the command "system".
4. At the (system) # prompt, use the "?", "show ?", and/or "list" command to learn about the commands used for system configuration.

The following paragraphs provide detailed descriptions about all CLI's system configuration commands.

Note: During system configuration, the CLI will return the (system) # prompt if a command is executed successfully or an error message if otherwise. So if you see the (system) # prompt after executing a command, it implies that the command has succeeded. You can then move on to configure other parameters.

Configure administration interface

```
(system)# adminface IFNAME  Interface name

IFNAME  Interface name
```

Sets the Membrain administration interface the specified network port.

Configure emt

```
(system)# emt (enable|disable)

(enable|disable): enable or disable emt data collection
```

Enables or disables the emt performance data collection feature on the Schooner Membrain™ server.

Send system report

```
(system)# send_incident
```

This command is related to the Call Home feature on Schooner Membrain™. To use the command, make sure that call home is enabled.

Configure Membrain

This section describes the CLI commands used for configuring the Schooner Membrain™ in the Schooner system. You can start configuring Schooner Membrain™ by using the following commands on the CLI:

1. At the ">" prompt, execute the command "enable".
2. At the "#" prompt, execute the command "configure terminal".
3. At the (config)# prompt, execute the command "membrain".

The following paragraphs provide detailed descriptions about all commands for configuring Schooner Membrain™.

Note: During Schooner Membrain™ configuration, the CLI will return the (membrain) # prompt if a command is executed successfully or an error message if otherwise. If you encounter an error message, you can use "?", "show ?", and/or "list" to learn about the commands and arguments for Schooner Membrain™ configuration.

Auto restart Membrain

```
(membrain)# autorestart (enable|disable)

Enable:     Enable auto restart
Disable:    Disable auto restart
```

Enables or disables the automatic restart feature on Schooner Membrain™. If enabled, Schooner Membrain™ should be able to restart automatically after it goes down. Otherwise, the user has to restart Schooner Membrain™ manually if it goes down unexpectedly.

Configure Membrain cache size

```
(membrain)# cache SIZE

SIZE: Size in GB of the DRAM cache
```

Configures the size of the Membrain DRAM cache in GB.

Create a container in eviction mode

```
(membrain)# container add TCPPORT UDPPORT CAPACITY yes (writeback|writethru)
(yes|no) CNAME (anyip|IPLIST)

TCPPORT:    Container TCP port number e.g., 11211
UDPPORT:    Container UDP port number e.g., 11211
CAPACITY:   Container capacity in gigabytes
yes:        Enable eviction
writeback:  Enable eviction with write-thru mode
writethru:  Enable eviction with write-back mode
(yes|no):   Enable/Disable persistence
CNAME:      Container name
anyip:      Any IP address
IPLIST:     A list of IP addresses separated by comma
```

Creates a container in eviction mode.

When setting TCP/UDP port numbers, do not use any port number from 51350 through 51600, because ports within that range are reserved the Schooner enterprise services.

When creating a container in eviction mode, the user can choose between write-back caching and write-through caching, which are two cache management schemes that dictate the manner in which data is written to disk. In write-back caching, modifications to data are held in cache and are not written to disk until is absolutely necessary. In write-through caching, all write operations are performed simultaneously both to cache and disk. To some degree, write-back caching provides better performance than write-through caching in that it reduces the number of write operations to disk, with a slight risk of data loss in case the system crashes.

Containers are identified by name, so make sure that you assign a unique name to each container you create. Since each Schooner Membrain™ server can support up to 128 containers, you can add as many as 128 containers if needed. If you enter the 'anyip' argument, the container can be accessed by clients via any system IP address; if you use the 'IPLIST' argument, the container can only be accessed through the system IP address(es) you have specified in this command. You can add up to 8 container IPs (excluding the local IP) per node for an independent node or 16 if the node is one of the two nodes of a mirror group. When adding a container to a mirror node, you should specify the VIPs of the container on both of the nodes in order for the container to have the failover capability.

Create a container in store mode

```
(membrain)# container add TCPPORT UDPPORT CAPACITY no (yes|no) CNAME
(anyip|IPLIST)

TCPPORT:    Container TCP port number e.g., 11211
UDPPORT:    Container UDP port number e.g., 11211
CAPACITY:   Container capacity in gigabytes
no:         Disable eviction
(yes|no):   Enable/Disable persistence
CNAME:      Container name
anyip:      Any IP address
IPLIST:     A list of IP addresses separated by comma
```

Creates a container with eviction disabled.

When setting TCP/UDP port numbers, do NOT use any port number from 51350 through 51600 because ports within that range are reserved for Schooner enterprise services.

Containers created using this command only support write-through cache. Containers are identified by name, so make sure that you assign a unique name to each container you create. Since each Schooner Membrain™ server can support up to 128 containers, you can add as many as 128 containers if needed. If you enter the 'anyip' argument, the container can be accessed by clients via any system IP address; if you use the 'IPLIST' argument, you must specify the IP address(es) in this command and the container can only be accessed through the system IP address(es) you have specified. You can add up to 8 container IPs (excluding the local IP) per node for an independent node or 16 if the node is one of the two nodes of a mirror group. When adding a container to a mirror node, you should specify the VIPs of the container on both of the nodes in order for the container to have the failover capability.

Add IPs to a container

```
(membrain)# container addip CNAME (anyips|IPLIST)

CNAME:      Container name
anyips:     Any IP address
IPLIST:     A list of IP addresses separated by comma
```

Adds IP addresses to a given container. If you enter the 'anyips' argument, the container can be accessed by clients via any system IP address; if you use the 'IPLIST' argument, the container can only be accessed through the system IP address(es) you have specified. Since containers are identified by name, make sure that you specify the container of interest using its unique name.

Back up a container

```
(membrain)# container backup CNAME (incr|full) (local)

CNAME:      Container Name
incr:       Backup mode type incremental
full:       Backup mode type full
local:      Back up to the local machine
```

Backs up a given container. Because containers are identified by name, you must specify the container by its unique name.

Delete a container

```
(membrain)# container delete CNAME

CNAME:      Container Name
```

Deletes a given container. Because containers are identified by name, you must specify the container by its unique name.

Remove IP Addresses from a container

```
(membrain)# container deleteip CNAME IPLIST

CNAME:      Container name
IPLIST:     A list of IP addresses separated by comma
```

Removes IP address(es) from a container. Because clients access a container through system IP addresses added to the container, removing an IP address from a container means blocking clients' access to the container via that IP address. If all system IP addresses are removed from a container, the container will become inaccessible to clients until IP addresses are added to it using the 'container addip' command. For this reason, you must exercise caution when removing IP addresses from a container.

Format a container

```
(membrain)# container format CNAME

CNAME:      Container Name
```

Formats a given container. Since containers are identified by name, make sure that you specify the container by its unique name.

Restore a container

```
(membrain)# container restore CNAME (local|nfs) BACKUPFILE

CNAME:      Container Name
local:      Back up to the local machine
nfs:        Back up to an NFS machine
BACKUPFILE: Name of the backup file
```

Restores a given container. Because containers are identified by name, make sure that you specify the container by its unique name. You also need to know the name of the backup file you want to use. You can find out all the available backup files using the "show backups" command.

Start a container

```
(membrain)# container start CNAME

CNAME:      Container Name
```

Starts a given container. Because containers are identified by name, make sure that you specify the name of the container you want to start.

Stop a container

```
(membrain)# container stop CNAME

CNAME:      Container Name
```

Stops a given container. Because containers are identified by name, make sure that you specify the name of the container you want to stop.

Sync a container

```
(membrain)# container sync CNAME
```

```
CNAME:      Container Name
```

Synchronizes a container's cache with secondary storage.

Back up a system configuration

```
(membrain)# container_conf backup DIRECTORY

DIRECTORY:  Backup directory path
```

Backs up the system's current configuration.

Restore a system configuration

```
(membrain)# container_conf restore PATH

PATH:       Name of the backup file to be deleted
```

Restores the specified system configuration backup file.

Allocate CPU cores

```
(membrain)# cores NUMBER

NUMBER:     Number of CPU cores to allocate to the Schooner Membrain
```

Schooner Membrain™ requires exclusive allocation of CPU cores. This command sets the number of CPU cores to allocate to Schooner Membrain™.

Initialize hotkey stats

```
(membrain)# hotkey init CNAME (disable|enable) (enable|disable)

CNAME:      Container Name
(disable|enable): Disable/enable ip_tracking on a container
(enable|disable): Disable/enable cmd_types on a container
```

Initializes hotkey statistics on a given container. Because containers are identified by name, make sure that you specify the name of the container of interest.

Turn off hotkey stats

```
(membrain)# hotkey off CNAME

CNAME:      Container Name
```

Disables hotkey statistics on a container. Because containers are identified by name, make sure that you specify the name of the container of interest.

Turn on hotkey stats

```
(membrain)# hotkey on CNAME

CNAME:      Container Name
```

Enables hotkey statistics on a given container. Because containers are identified by name, make sure that you specify the name of the container of interest.

Reset hotkey stats

```
(membrain)# hotkey reset CNAME

CNAME:      Container Name
```

Resets hotkey statistics on a given container. Because containers are identified by name, make sure that you specify the name of the container of interest.

Add a VIP

```
(network)# interface addvip (perm|temp) IFNAME IP/Prefix/GW

perm:       Permanent VIPs (persistent on reboots)
temp:       Temporary VIPs (non persistent on reboots)
IFNAME:     Interface name
IP/Prefix/GW: IP address, subnet mask, or gateway of the interface
```

Adds a virtual IP address (VIP).

Delete all VIPs

```
(network)# interface deleteallvip IFNAME

IFNAME      Interface name
```

Removes all VIPs on a given node.

Delete a VIP

```
(network)# interface deletevip IFNAME IP

IFNAME:     Interface name
IP:         Virtual IP address
```

Removes a given VIP.

Add a mirror group

```
(membrain)# mirrorgroup add GRPNAME NODE1 NODE2

GRPNAME:    Group Name
NODE1:      Name of the first node
NODE2:      Name of the second node
```

Creates a mirror group using two selected nodes.

Delete a mirror group

```
(membrain)# mirrorgroup delete GRPNAME

GRPNAME:    Group Name
```

Deletes a given mirror group.

Configure the messaging interface

```
(membrain)# msgiface IFNAME

IFNAME:     Interface name
```

Sets up the messaging interface for the given node. For initial configuration, this must be “eth0’ on Schooner Membrain™. If the messaging interface is invalid (i.e., if no such an interface exists, or the interface has not been configured yet, or the interface is disabled), the system will return an error message.

Messaging interfaces handle communication between nodes. It must be configured with a valid interface name after the administration interface is configured.

If a bonded interface is to be configured as a messaging interface, the corresponding bond interface name (e.g., Bond0) must be configured as the messaging interface. If a VLAN-enabled interface (e.g., either the ethX or BondX interface) is to be configured as the messaging interface, the corresponding VLAN interface name must be specified as the messaging interface.

Reinstate a node

```
(membrain)# reinstate
```

Reinstates a node as the persistent container recovery master node.

Remove a backup or restore task

```
(membrain)# remove (backup|restore) CNAME

(backup|restore): Task type
CNAME:            Name of the container
```

Restores the specified system configuration backup file.

Start Membrain

```
(membrain)# start
```

Starts a Schooner Membrain™ instance.

Enable/Disable automatic start of Membrain on system boot

```
(membrain)# start_onboot (enable|disable)

(enable|disable): Enable or disable auto start of Membrain when the system
boots
```

Starts a Membrain instance upon system boot.

Stop Membrain

```
(membrain)# stop
```

Stops a Membrain instance.

Path to secondary storage

```
(membrain)# storage (device|filesystem) LOCATION [SIZE]
(device|filesystem): Specifies storage type
LOCATION:            Path to device or file system
SIZE:                Size of file system (does not apply to device)
```

Sets the file path to secondary storage. You may specify a storage device or a file system directory. If a file system is selected, you must also specify the size of storage file to create.

Control secondary storage sync command

```
(membrain)# storage_sync (enable|disable)

(enable|disable): Enable or disable secondary storage sync
```

Certain secondary storage devices require an explicit sync command to be sent in order to ensure data durability. This command is sent whenever a “container sync” command is issued.

Change Memcached Version

```
(membrain)# switch (1.2|1.4)
```

```
1.2          Memcached version 1.2
1.4          Memcached version 1.4
```

Makes the Schooner Membrain™ server compatible with the selected version of open-source Memcached. Some Memcached features are version-specific. This command enables users to switch the Schooner Membrain™ server to match the selected version of open-source Memcached so that they can use the features that are available only in that version. Please note that this command requires the restart of Schooner Membrain™ in order for it to take effect.

Manage Schooner Cluster

This section describes the CLI commands used for managing the Schooner cluster. You can start managing your Schooner Cluster by using the following commands:

1. At the "`>`" prompt, execute the command "`enable`".
2. At the "`#`" prompt, execute the command "`configure terminal`".
3. At the `(config)#` prompt, execute the command "`cluster`".

The following paragraphs provide detailed descriptions about all CLI's cluster management commands.

Modifying the cluster's password

```
(cluster)# credential

PASSWORD:   The new password for the cluster
```

Changes the Schooner cluster password on the local node. Enter the new cluster password and press Enter. Then enter it for a second time. The command returns the (cluster)# prompt if it is successful or an error message if it fails.

Join the cluster

```
(cluster)# join IP

IP:   IP address of the remote node in the cluster
```

Joins the local node with a remote node in the cluster. The IP address refers to the IP address of the remote node the local node is to be joined.

A cluster is an administration domain. When joining a node to a cluster, keep in mind the following points:

- A node must be initialized using the FTW before it can be joined to a node in a cluster. In other words, a node that has not been initialized can be joined to a node in a cluster or be joined by another node in a cluster.
- A node must pass the version compatibility check with the cluster before it can be added to the cluster. A node that has failed the version compatibility check cannot join that cluster.
- A node that is already in a cluster must leave that cluster in order to join another cluster. .

Leave the cluster

```
(cluster)# leave
```

Removes the local node from the cluster.

When removing a node from a cluster, keep the following in mind:

- A node that is already in a cluster must leave that cluster in order to join another cluster.
- A node that is currently in a mirror group cannot leave a cluster from the CLI. To leave the cluster, the node must be removed from the mirror node first.

View System Status Information

This section provides examples illustrating the results of various `'show'` commands. Most of the show commands are available at all levels of the CLI and can be executed at any time, at any prompt, i.e., `'>', '#', '(config) #', '(system) #', '(network) #', or "(membrain) #'.`

Show available container backup files

```
show backups

backups:    List of available container backups

Sample output:

Backups list:
Cluster      Container              Backup     Type       Date
Schooner     Schooner:11211:1       1.0        FULL       2009-12-25 00:19
Schooner     Schooner:11211:1       2.0        FULL       2009-12-25 00:19
Schooner     Schooner:11211:1       3.0        FULL       2010-01-06 12:39
Schooner     Schooner:11211:1       3.1        INCR       2010-01-06 12:40
Schooner     Schooner:11211:1       3.2        INCR       2010-01-06 12:41
...
```

Shows a list of available Membrain container backup files. The command comes handy when you want to restore your system to a previous configuration. In that case, you can use this command to display all available backup files first and then select the one to be restored.

Show cache

```
show cache

cache: DRAM cache size in GB

Sample output:

64
```

Shows the NTP (Network Time Protocol) severs.

Show Call Home configuration

```
show callhome

Sample output:

disabled
```

Shows whether the call-home feature is enabled or disabled on Schooner Membrain™. System call-home is a feature on Schooner Membrain™ that, if enabled, will automatically send troubleshooting information to Schooner support team.

Show configuration of a specific container

```
show container CNAME

CNAME:      Container name

Sample output:

Container 1
status      : on
name        : ABC
id          : 1
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
persistent  : No
size (MB)   : 1024
IPs         : 0.0.0.0
hotkey      : Disabled
cache_mode  : writethru
```

Shows the status and configuration of a given container. You must specify the name of the container.

Show a container with hot client statistics

```
show container CNAME hotclient NTOP

CNAME:      Container name
Hotclient:  enable hot client output
NTOP:       Number of top hot clients to be displayed. Enter 0 to list all

Example:
(membrain)# show container ABC hotclient 10

Sample output:

Container 1
status      : on
name        : ABC
id          : 9
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
persistent  : No
size (MB)   : 1024
IPs         : 0.0.0.0
hotkey      : Enabled
hot clients:
HOTCLIENT
334304 10.1.20.55
317454 10.1.20.51
cache_mode  : writeback
```

Shows the configuration of the specified container with hot client statistics. Note that the sample output only shows 2 hot clients even though the command specifies 10. This is because 10 is the maximum number of hot clients the command intends to display, but the container has only 2 hot clients.

Show a container with hot key statistics

```
show container CNAME hotkey NTOP

CNAME:      Container name
Hotkey:     enable hot key output
NTOP:       Number of top hot keys to be displayed. Enter 0 to list all

Example:
(membrain)# show container ABC hotkey 10

Sample output:

Container 1
status      : on
name        : ABC
id          : 9
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
persistent  : No
size (MB)   : 1024
IPs         : 0.0.0.0
hotkey      : Enabled
Hot Keys:
HOTKEY_ACCESS
0.0.0.0 181 UDl9UfgVT
0.0.0.0 178 Dl9UfgVT
0.0.0.0 175 ]yRDl9UfgVT
0.0.0.0 174 {tDl9UfgVT
0.0.0.0 172 DZl9UfgVT
0.0.0.0 172 DDl9UfgVT
0.0.0.0 172 FKvOee1c-oLtHggzc.9b
0.0.0.0 171 18yDl9UfgVT
0.0.0.0 170 XTFKvOee1c-oLtHggzc.9b
0.0.0.0 170 xu:Dl9UfgVT
cache_mode  : writeback
```

Shows the configuration of the specified container with hot key statistics. Note that the sample output displays 10 hot keys which is the maximum number of hot keys the command intends to show.

Show a container with hot client and hot key statistics

```
show container CNAME hotclient NTOP hotkey NTOP

CNAME:      Container name
Hotclient:  enable hot client output
NTOP:       Number of top hot clients to be displayed. Enter 0 to list all
Hotkey:     enable hot key output
NTOP:       Number of top hot keys to be displayed. Enter 0 to list all

Example:
(membrain)# show container ABC hotclient 10 hotkey 10

Sample output:

Container 1
status      : on
name        : ABC
```

```
id          : 9
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
persistent  : No
size (MB)   : 1024
IPs         : 0.0.0.0
hotkey      : Enabled
Hot Keys:
HOTKEY_ACCESS
0.0.0.0 226 Dl9UfgVT
0.0.0.0 222 Dl9UfgVT
0.0.0.0 219 U88-FKvOee1c-oLtHggzc.9b
0.0.0.0 218 838-FKvOee1c-oLtHggzc.9b
0.0.0.0 218 UDl9UfgVT
0.0.0.0 217 8-FKvOee1c-oLtHggzc.9b
0.0.0.0 217 xSt-2hoWpsTheEHf5vQ3A3oIKYyw.1HfgmfiRRDEIgvwvBa.4.1E2pWMMSGKUX
0.0.0.0 216 p8-FKvOee1c-oLtHggzc.9b
0.0.0.0 216 18yDl9UfgVT
0.0.0.0 214 ]yRDl9UfgVT
Hot clients:
HOTCLIENT
5753629 10.1.20.55
5500138 10.1.20.51
cache_mode  : writeback
```

Shows the configuration of the specified container with hot key and hot client statistics.

Show a container with hot key and hot client statistics

```
show container CNAME hotkey NTOP hotclient NTOP

Container
status      : on
name        : Uno
id          : 1
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
persistent  : No
size(MB)    : 1024
IPs         : 0.0.0.0
hotkey      : Enabled
cache_mode  : writethru

Hot Keys
No hotkey entries

Hot Clients
No hot client entries
```

Shows the configuration of the specified container with hot key and hot client statistics. The command yields the same output as the command `'show container CNAME hotclient NTOP hotkey NTOP', as` described above. It is included in the CLI just in case the user enters the command in this way.

Show configuration backup files

```
show config_conf PATH
```

```
PATH:  Path to configuration backups

Sample output:

engine1.techpub.net-Feb-10-2010-16:09:19
engine1.techpub.net-Feb-10-2010-16:08:57
engine1.techpub.net-Feb-10-2010-16:04:24
```

Shows a list of available system configuration backup files. Each backup file is identified by the time the backup was made.

Show all containers with detailed information

```
show containers

containers: List of containers

Sample output:

Container 1
status      : on
name        : ABC
id          : 01
tcp_port    : 11211
udp_port    : 11211
eviction    : Yes
size        : 1024(MB)
Container 2
status      : on
name        : DEF
id          : 02
tcp_port    : 11212
udp_port    : 11212
eviction    : yes
size        : 1024(MB)
```

Shows detailed information about all the containers on the node.

Show containers with basic information

```
show containers basic

containers: List of containers
basic:      Basic container information

Sample output:

name  status      persistence       eviction    hotkey      TCP         UDP
C01   started     on                off         on          11211       11211
C02   stopped     off               on          on          11212       11212
C03   started     off               oo          off         11213       11213
C04   stopped     off               on          on          11214       11214
```

Shows all available containers, one container per row, with basic information.

Show selected containers with basic information

```
show containers basic (started|stopped|persistent|non-persistent|
eviction|non-eviction|hotkey-on|hotkey-off)

containers: List of containers
```

```
basic:      Basic container information
started:    Containers that are started
stopped:    Containers that are stopped
persistent: Persistent containers
non-persistent:   Non-persistent containers
eviction:   Container in eviction mode
non-eviction:     Containers in non-eviction mode
hotkey-on:  Containers with hotkey stats enabled
hotkey-off: Containers with hotkey stats disabled

Sample output:

name  status      persistence       eviction    hotkey      TCP         UDP
C01   started     on                off         on          11211       11211
C03   started     off               on          off         11213       11213
```

Shows basic information about the containers specified by the state-filter. The state-filter is an operating mode or status used to categorize the containers to be displayed. You can only use one filter per command.

Show all containers with detailed information

```
show containers detail

containers: List of containers
detail:     Detailed container information
```

Shows detailed information about all available containers. The screen output format of this command is the same as that of the “show container CNAME” command. This difference is that this command displays detailed information of all containers, one after another.

Show selected containers with detailed information

```
show containers detail (started|stopped|persistent|non-persistent|
eviction|non-eviction|hotkey-on|hotkey-off)

containers:       List of containers
detail:           Basic container information
started:    Containers that are started
stopped:    Containers that are stopped
persistent: Persistent containers
non-persistent:   Non-persistent containers
eviction:   Container in eviction mode
non-eviction:     Containers in non-eviction mode
hotkey-on:  Containers with hotkey stats enabled
hotkey-off: Containers with hotkey stats disabled
```

Shows detailed information about the containers specified by the state-filter. The state-filter is an operating mode or status used to categorize the containers to be displayed. You can only use one filter per command.

Show CPU cores allocation

```
show cores

cores:  CPU cores

Sample output:
```

```
8
```

Shows the number of CPU cores allocated to Schooner Membrain™.

Show grid configuration

```
show grid

grid:  List of Schooner nodes in the Schooner cluster
Sample output:

Host:lab24.schoonerinfotech.net
      IP    :10.1.20.24
      Status:DOWN
Host:ga0.schoonerinfotech.net
      IP    :10.1.20.100
      Status:UP
```

Shows the IP addresses and status information of the nodes in the grid.

Show groups

```
show groups

groups:     List of groups (independent vs. mirrored) configured on the node

Sample output:

Group: IndGroup
    Type:INDEPENDENT
    Node: ga0.schoonerinfotech.net
        Status: Up
        Msg Ifs: eth0
        Admin If: eth0
        Application: Membrain
        App Status: Active
    Node: lab24.schoonerinfotech.net
        Status: Down
        Msg Ifs: N/A
        Admin If: N/A
        Application: N/A
        App Status: Down
```

Shows the existing groups in the grid. There are two types of groups: independent and mirrored.

Show command history

```
show history

history:    List of commands executed since the start of the session

Sample output:

enable
configure terminal
membrain
show history
```

Shows all the commands that have been executed since the start of the current session.

Clear command history

```
history clear

history clear:    clears the command history
```

Clears all commands listed in the command history.

Show Membrain logs

```
show logs membrain NLINES

logs Membrain: membrain logs
NLINES:     Number of lines to show or 0 for complete log


Example:
(membrain)# show logs membrain 2

Sample output:

Mon Dec 28 11:37:37 PST 2009 membrain-fthread-sdf_o[15688] 435fb940
0x2aaaac1a82c0 apps/membrain/server:info ../mcd_ipf.c:1610 ipf_notify 26-
20259 Virtual IP group 1 removed from node 0
```

Shows Membrain logs. You must specify the number of log entries to be displayed.

Show system logs

```
show logs system NLINES

logs system:  system logs
NLINES:     Number of lines to show or 0 for complete log

Example:
(system)# show logs system 5

Sample output:

Dec 22 08:49:56 localhost ntpd[2399]: synchronized to 38.117.195.101, stratum
2
Dec 22 10:19:35 localhost ntpd[2399]: synchronized to 38.229.71.1, stratum 2
Dec 22 15:02:20 localhost ntpd[2399]: synchronized to 38.117.195.101, stratum
2
Dec 22 16:00:55 localhost ntpd[2399]: synchronized to 64.6.144.6, stratum 2
Dec 22 17:56:22 localhost ntpd[2399]: synchronized to 38.117.195.101, stratum
2
```

Shows system logs. You must specify the number of log entries to be displayed. In the example above, the command specifies that 5 log entries should be displayed.

Show messaging interface

```
show msgiface

msgiface:   Messaging interface

Sample output:

eth0
```

Shows the Schooner Membrain™ messaging interface, which is ‘eth0’ in this case.

Show node configuration

```
show node

node:       Node status

Sample output:

Node: ga0.schoonerinfotech.net
    Status      : Up
    Msg Ifs     : eth0
    Admin If    : eth0
    Application : membrain
    App Status  : Active
```

Shows the information about the specified node; otherwise, it shows the information about the master node.

Show storage

```
show storage

storage:  Path to secondary storage device or file system

Sample output:

Storage file path : /opt/schooner/data
Storage file size : 10GB
```

Shows the SNMP (Simple Network Management Protocol) communities.

Show backup task progress

```
show task backup

Sample output:

CNAME:      Container-1
TYPE:       LOCAL
LOCATION:   /schooner/backup
PROGRESS:   100
ERROR:
MODE:       full
```

Shows the progress of a container backup operation.

Show restore task progress

```
show task restore

CNAME:      Foo
TYPE:       LOCAL
LOCATION:   /schooner/backup/Schooner/Foo/BACKUP-1/FULL-0
PROGRESS:   100
ERROR:
```

Shows the progress of a container restore operation.

Show Schooner software version

```
show version

version:    Schooner Membrain  version

Sample output:

Schooner Membrain/NoSQL 3.1.0-15061-1306
========================================
schooner-ldap-config-5-1
schooner_helm-3.1.0-15037
schooner_helm-3.1.0-15048
schooner_helm-3.1.0-15054
schooner_helm-3.1.0-15061
schooner_membrain-3.1.0-15037
schooner_membrain-3.1.0-15048
schooner_membrain-3.1.0-15054
schooner_membrain-3.1.0-15061
schooner_scm-3.1.0-15037
schooner_scm-3.1.0-15048
schooner_scm-3.1.0-15054
schooner_scm-3.1.0-15061
schooner_sendincident-3.1.0-15037
schooner_sendincident-3.1.0-15048
schooner_sendincident-3.1.0-15054
schooner_sendincident-3.1.0-15061
schooner_utils-3.1.0-15037
schooner_utils-3.1.0-15048
schooner_utils-3.1.0-15054
schooner_utils-3.1.0-15061
schoonerinfotech-yum-config-5.5-1
```

Shows the version of the Schooner Membrain™ software.

Troubleshooting and Diagnostics

This section discusses the following commands used for troubleshooting and diagnosing the Schooner system.

```
ping WORD         send echo messages
telnet WORD PORT  Open a telnet connection
traceroute WORD   Trace route to destination
```

Ping a destination IP address or hostname

```
ping IPADDR

IPADDR:    Destination IP address or hostname

Example:
# PING 10.1.20.100

Sample output:

PING 10.1.20.100 (10.1.20.100) 56(84) bytes of data.
64 bytes from 10.1.20.100: icmp_seq=1 ttl=64 time=0.029 ms
...
--- 10.1.20.100 ping statistics ---
```

```
16 packets transmitted, 16 received, 0% packet loss, time 14998ms
rtt min/avg/max/mdev = 0.017/0.021/0.029/0.005 ms
#
```

Pings a destination IP address or hostname to determine if there is any network connection issue.

Connect to a telnet server via the default TCP port

```
telnet IPADDR

IPADDR:    IP address or hostname of a remote system
PORT:      TCP Port number

Examples:
# telnet 192.168.1.200

Sample output:

Trying 192.168.1.200...
telnet: connect to address 192.168.1.200: No route to host
```

Opens a telnet connection to a telnet server using the default TCP port number (.i.e., 23).

Connect to a telnet server via a specific TCP port

```
telnet IPADDR PORT

IPADDR:    IP address or hostname of a remote system
PORT:      TCP Port number

Examples:
# telnet 192.168.1.200 11211

Sample output:

Trying 192.168.1.200...
telnet: connect to address 192.168.1.200: Connection refused
```

Opens a telnet connection to a remote telnet server using a specific TCP port number other than the default one.

Test network latency along

```
traceroute IPADDR

IPADDR:    IP address or hostname of a remote system

Example:
# traceroute 10.1.20.100

Sample output:

traceroute to 10.1.20.100 (10.1.20.100), 30 hops max, 40 byte packets
1 ga0 (10.1.20.100) 0.044 ms 0.024 ms 0.020 ms
```

Pings all hosts between the given node and the specified destination, showing time for each hop and the latency caused by each host. The command is used for troubleshooting network connections.

Set the terminal screen parameters

```
terminal length

terminal:  Set terminal line parameters
length:    Set the number of lines on a screen

Example:
# terminal length 10
```

Limits the display of command output on the screen to the number of lines of text specified in this command. For example, if you set the number of lines per screen to 10, then the CLI will only display up to 10 lines of text per screen for a command that generates more than 10 lines of screen text output. You need to press the 'Enter' key on your keyboard to display more lines of text.

Chapter 7. Mirrored Group Network Setup Requirements

Mirrored Group Failover

When a node fails, the Virtual IP of the failed node would automatically be configured on the surviving node's corresponding interfaces. eth0 VIP of the failed node would be configured on eth0 of the surviving node, eth1 VIP of the failed node would be configured on eth1 of the surviving node, bond0 VIP of the failed node would be configured on bond0 of the surviving node and so on

VIP Failover

Each VIP is associated with a network interface. When a node fails, the VIPs of all the containers in the failed node will automatically be reconfigured at the survivor node on the corresponding interface. For example VIPs of interface ethN of the failed node would be configured to ethN of the survivor node. So similar interfaces of the mirrored nodes must be in the same physical/logical network so that applications can reach the VIPs under normal and failed conditions.

Messaging Interface

The messaging interface must be configured with a valid interface name after interface configuration is completed.

If a bonded interface is to be configured as the messaging interface, the corresponding bond interface name (e.g., Bond0) must be configured as the messaging interface.

If a VLAN-enabled interface (either ethx interface or bondx interface) is to be configured as the messaging interface, the corresponding VLAN interface name must be specified as the messaging interface on the CLI.

The VLAN interface name is derived as vlan<tag>.

- Example 1:
  - If eth8 is configured with VLAN `100` and eth8 needs to be the messaging interface, then `vlan100` must be specified as the messing interface, not eth8.
- Example 2:
  - If bond0 is configured with the VLAN `10` and the bonded interface needs to be the messaging interface, then interface `vlan10` must be specified as the messing interface, not bond0.

Split Brain Handler

Split-brain situation is a condition where the replication link between the two nodes in a mirrored group is down. When this occurs, both nodes will assume that the peer node is dead and will start hosting the peer node's VIP addresses, resulting in the same VIP addresses being hosted by two nodes. This situation will cause unpredictable client connection failures.

With the split-brain handler, Schooner Membrain™ can actively monitor for split-brain conditions on the network and automatically stop Membrain on one node the moment a split-brain condition is detected. Once stopped, the node will remain inactive until the system administrator manually restarts it from the CLI.

Chapter 8. Hot Keys and Hot Clients

The Membrain concept

Membrain is built upon the simple concept that frequently accessed data is stored in a database as well as Membrain. In the event a client queries certain data, the cache is checked first; if the data can't be found in the cache, the system will then tap into the database to fetch it and updates the cache at the same time.

Data stored in Membrain is in fact stored in memory and can be fetched using a network protocol that is both simple and lightweight. Since the data is accessed using a single hash key, the lookup time is very consistent.

Client-server communication

Membrain clients communicate with a Membrain server via TCP connections. Typically, the server listens on a TCP port to which the clients connect, send commands to and get responses from the server, and eventually close the connection.

Key/value store

Membrain is a simple key/value store in that data stored in Membrain is identified with keys. Each key is a unique text string that identifies the data that are being stored or retrieved by clients.

Membrain hot keys and hot clients

Schooner Membrain™ provides detailed statistical information about the hot keys and hot clients on a given Membrain server. Hot keys are the object keys that are most frequently used on a given Membrain server; hot clients are the Membrain clients (identified by IP address and port number) that access a given Membrain server most frequently. Such information offers website developers and operators a clear picture of what is being stored on a given Membrain server and helps them understand and troubleshoot the placement of object keys and how they are being used.

Schooner Membrain™ Hot key and Hot Client Commands

This section discusses the CLI commands used for managing hot key and hot client data in Schooner Membrain™. Note that because hot keys and hot clients are closely related, the following CLI commands cover both hot client and hot key even though they only contain the term hot key in them.

- hotkey init CNAME
- hotkey off CNAME
- hotkey on CNAME

- hotkey reset CNAME

For more information on hot key and hot client management, refer to Chapter 8 of this manual.

Initialize hot keys and hot clients

hotkey init CNAME

Initializes hot keys and hot clients in a given container. Either TCP port number or name can specify the container.

Enable hot keys and hot clients

hotkey on CNAME

Enables hot keys and hot clients in a given container. By default, the hot key and hot client data on a container is disabled. You must enable it if you want to get hot key ad hot client stats on a container.

Turn off hot keys and hot clients

hotkey off CNAME

Disables hot key and hot client data in a given container.

Reset hot keys and hot clients

hotkey reset CNAME

Clears existing hot key and hot client statistics in a container.

Show hot key data output

show container CNAME hotkey NTOP

Shows the configuration of a given container with hot key statistics. The 'hotkey' argument enables hot key output and the 'NTOP' argument determines the maximum number of hot keys to be displayed.

show hot client data output

show container CNAME hotclient NTOP

Shows the configuration of a given container with hot client statistics. The 'hotclient' argument enables hot client output and the 'NTOP' argument determines the maximum number of hot clients to be displayed.

Show hot client and hot key data output

show container CNAME hotclient NTOP hotkey NTOP

Shows the configuration of a given container with hot client ad hot key data.

Show a container with hot key and hot client data output

show container CNAME hotkey NTOP hotclient NTOP

Shows the configuration of a given container with hot key and hot client data. The command yields the same output as the command `'show container CNAME hotclient NTOP hotkey NTOP', as` described above. It is included in the CLI to provide the user some flexibility when viewing hot key and hot client data.

Chapter 9. Performance Monitoring

Schooner Membrain™ provides a set of performance charts monitoring metrics for cpu, memory, IO and Schooner Membrain™ at 5 minute intervals. Data is displayed for the past 2 hours.

CPU Utilization

Schooner Membrain™ uses a polling mechanism to minimize response latency and maximize transaction throughput. This means that the CPU cores assigned to it will be 100% busy from the view of a system monitor.

The container dashboard shows the idle time spent within Schooner Membrain™. This is the available CPU headroom.

System Dashboard

The System Dashboard displays a summary of server metrics:

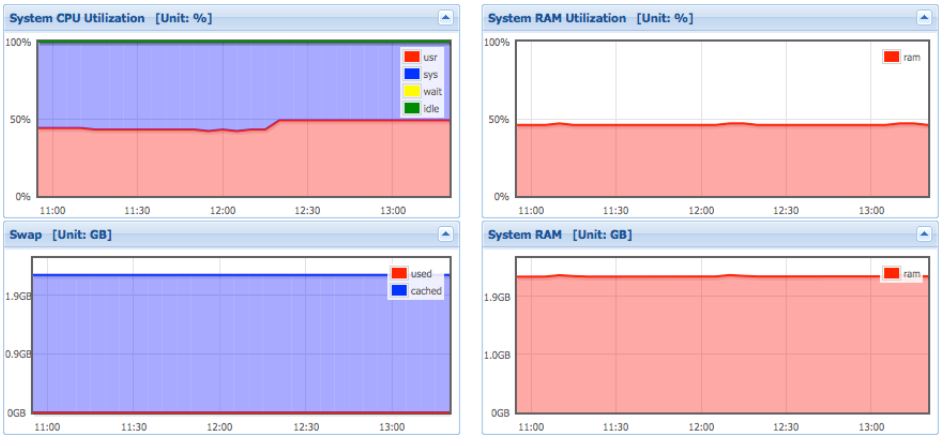


Figure 9-1: System Dashboard Charts

- Average CPU Utilization
  - Average CPU utilization over all cores for the following states: system, user, and wait.
- % Memory Utilization
  - Percentage of memory used.
- Swap
  - Used in GB.
  - Cached in GB.
- Memory
  - Total memory in GB.

- Disk Utilization
  - Disk usage %.
- Disk IOPS.
  - Disk read operations/second.
  - Disk write operations/second.
- Disk Bandwidth
  - Disk bandwidth in MB/s.
- Disk IO Queue length.
  - Per device queue size.
- Network Bandwidth
  - Network bandwidth in MB/s.

Container Dashboard

The performance charts display important system and Membrain metrics that can be used to quickly determine bottlenecks or analyze the effect of database or workload changes on performance.

The Container Dashboard displays a summary of per container Membrain metrics:

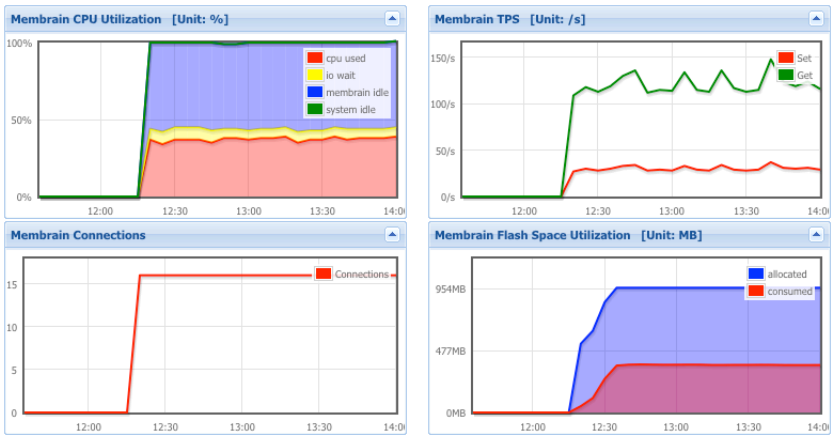


Figure 9-2: System Dashboard Charts

- Membrain CPU Utilization
  - Per container % CPU usage.
  - Per container %CPU IO wait.
  - Per container %CPU Membrain idle.
  - Per container %CPU system idle.
- Membrain Total TPS
  - Per container get transaction rate.
  - Per container set transaction rate.
- Membrain Connections
  - Per container connection count.

- Membrain Disk Utilization
  - Per container disk allocated and consumed space.
- Membrain Cache Space Utilization
  - Per container allocated size.
  - Per container used size.
  - Per container data + key size.
- Membrain Hit
  - Per container hit rate.
- Membrain Cache Miss
  - Per container cache miss rate.
- Membrain Overwrites Rate
  - Per container storage overwrites.
  - Per container cache overwrites.
- Membrain Evictions
  - Per container evictions.
- Membrain Average Key Length
  - Per container average get key length.
  - Per container average set key length.
- Membrain Average Data Length
  - Per container average get data length.
  - Per container average set data length.
- Membrain TPS(2)
  - Per container add transaction rate.
  - Per container delete transaction rate.
  - Per container replace transaction rate.
- Membrain TPS(3)
  - Per container append transaction rate.
  - Per container prepend transaction rate.
- Membrain TPS(4)
  - Per container decrement transaction rate.
  - Per container increment transaction rate.
  - Per container cas transaction rate.
- Membrain TPS (5)
  - Per container transaction sync transaction rate.
  - Per container sync all transaction rate.
  - Per container flush all transaction rate.

Chart Configuration

Each chart in a dashboard may be hidden or displayed by using the arrow buttons:

Figure 9-3: Hide/Display Chart Buttons

Certain charts, such as disk and network interfaces, may be configured to display a specific set of devices. For example, you may configure the System Disk Bandwidth chart to show two devices:

1. Click on the “properties” button for the chart:

Figure 9-4: System Dashboard Charts

2. The chart properties dialog will appear:

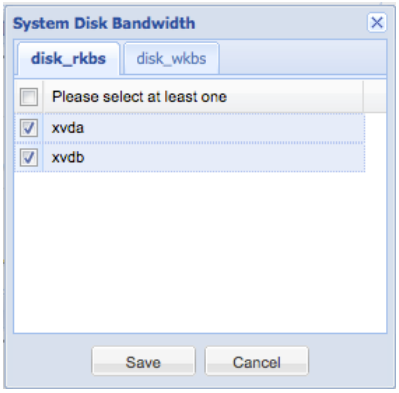


Figure 9-5: System Disk Bandwidth Chart Properties

3. Select the disks you wish to display.
4. The chart will be updated:

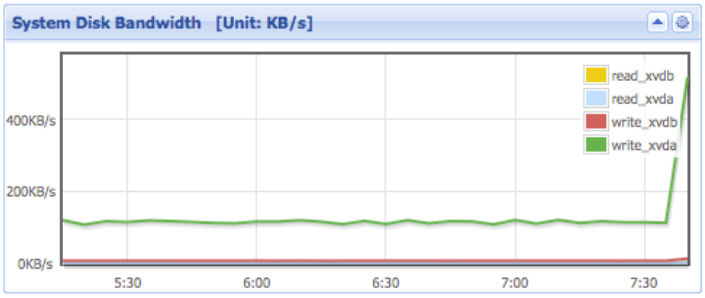


Figure 9-6: System Disk Bandwidth Chart

Appendix: Schooner Membrain™ Statistics

Schooner Membrain™ supports the standard Memcached statistics as well as its own set of statistics.

Standard Memcached Statistics

The following are the standard Memcached statistics:

```
stats
STAT pid 30621
STAT uptime 17345
STAT time 1296879516.80556
STAT version 1.4.4-schooner 16105 membrain 3.1.0
STAT pointer_size 64
STAT rusage_user 152136.043805
STAT rusage_system 56005.107929
STAT curr_items 24010321
STAT total_items 159778658
STAT bytes 68554034688
STAT curr_connections 208
STAT total_connections 4021
STAT connection_structures 88
STAT cmd_get 316670533
STAT cmd_set 159778627
STAT get_hits 316362614
STAT get_misses 307919
STAT evictions 132660018
STAT bytes_read 585039132263
STAT bytes_written 1256905395107
STAT limit_maxbytes 68719476736
STAT threads 1
END
```

Schooner Membrain™ Statistics

The following are the Schooner Membrain™ statistics:

```
stats all
STAT pid 30621
STAT uptime 17466
STAT time 1296879637.580330
STAT version 1.4.4-schooner 16105 membrain 3.1.0
STAT pointer_size 64
STAT rusage_user 152836.930254
STAT rusage_system 56512.392810
STAT vol_ctx_sw 319544048, invol_ctx_sw 943219029
STAT page_reclaims 14894424, page faults 1
STAT signals_recv 0, swaps 0
STAT curr_items 24015705
STAT total_items 161351408
STAT bytes 68550572032
STAT curr_connections 208
STAT total_connections 4047
STAT connection_structures 88
STAT cmd_get 318070195
STAT cmd_set 161351371
STAT get_hits 317753976
STAT get_misses 316219
STAT evictions 134224998
STAT bytes_read 589147395006
STAT bytes_written 1260845338749
STAT limit_maxbytes 68719476736
STAT threads 1
STAT last_reset_time 1296862171.0
STAT Mcd_cmds (GT)318070195 (ST)161351371 (AD)0 (RP)0 (AP)0 (PP)0 (CS)0 (IC)0 (DC)0 (DL)0
(SN)0 (SA)0 (FA)0
STAT Mcd_get_key: evs=691334227 min=16 max=250 avg=98.757274 geo=64.943544 std=83.525954
STAT Mcd_get_key: (4)120826376 (5)145798441 (6)0 (7)292545633 (8)132163777
```

```
STAT Mcd_get_data: evs=691334227 min=2 max=131033 avg=4499.248537 geo=1348.033010
std=7441.630371
STAT Mcd_get_data: (0)0 (1)5944249 (2)11772920 (3)29725748 (4)35681783 (5)4064491
(6)7602208 (7)21073171 (8)26610657 (9)23536415 (10)11596632 (11)4157047 (12)408616314
(13)29385794 (14)39991560 (15)13240310 (16)17523313 (17)811615
STAT Mcd_set_key: evs=332895319 min=16 max=250 avg=91.178272 geo=60.177729 std=81.236290
STAT Mcd_set_key: (4)45066895 (5)104261112 (6)0 (7)126920250 (8)56647062
STAT Mcd_set_data: evs=332895319 min=2 max=131033 avg=5236.494234 geo=317.225716
std=10726.449866
STAT Mcd_set_data: (0)0 (1)5952836 (2)11782916 (3)29761710 (4)35715014 (5)21117052
(6)24672320 (7)24961283 (8)26687640 (9)6683313 (10)11637003 (11)4171510 (12)28790694
(13)29381560 (14)39998657 (15)13241708 (16)17527451 (17)812652
STAT Mcd_get_time: evs=0
STAT Mcd_set_time: evs=0
STAT flash_class_map 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STAT flash_slab_map 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
STAT flash_space_allocated 68719476736
STAT flash_space_consumed 68550595584
STAT flash_num_objects 24015725
STAT flash_num_created_objects 158259939
STAT flash_num_evictions 134224998
STAT flash_num_overwrites 19216
STAT flash_num_ops 112118063
STAT flash_num_read_ops 112006791
STAT flash_num_get_ops 35596839
STAT flash_num_put_ops 158259939
STAT flash_num_del_ops 0
STAT sdf_cache_hits 282473466
STAT sdf_cache_misses 35596854
STAT sdf_flash_hits 35280617
STAT sdf_flash_misses 316219
STAT sdf_cache_evictions 0
STAT sdf_n_overwrites 1261
STAT sdf_n_in_place_overwrites 20541
STAT sdf_n_new_entries 196610257
STAT sdf_cache_only_items 0
END
```

The interesting additions provided by Schooner Membrain™ “stats all” are:

- Mcd_cmds - counts for the following commands:
    - GT - get
    - ST - set
    - AD - add
    - RP - replace
    - AP - append
    - PP - prepend
    - CS - cas
    - IC - increment
    - DC - decrement
    - DL - delete
    - SN - sync
    - SA - sync_all
    - FA - flush_all
- Mcd_get_key, Mcd_get_data, Mcd_set_key, Mcd_set_data
    - evs - events
    - min - minimum size
    - max - maximum size
    - avg - average size
    - geo - geometric mean
    - std - standard deviation

    - data buckets in power of 2 size
- flash_space_allocated
    - secondary storage allocated
- flash_space_consumed
    - secondary storage consumed
- flash_num_objects
    - secondary storage objects
- flash_num_created_objects
    - secondary storage object creations
- flash_num_evictions
    - secondary storage evictions
- flash_num_overwrites
    - secondary storage object overwrites
- flash_num_ops
    - secondary storage operations
- flash_num_read_ops
    - secondary storage reads
- flash_num_put_ops
    - secondary storage writes
- flash_num_del_ops
    - secondary storage deletes
- sdf_cache_hits
    - DRAM cache hits
- sdf_cache_misses
    - DRAM cache misses
- sdf_cache_evictions
    - DRAM cache evictions
- sdf_flash_hits
    - Secondary storage get hits
- sdf_flash_misses
    - Secondary storage object misses
- sdf_n_overwrites
    - Cached object overwrites