Schooner Membrain 3.1.1

Release Notes

May 17, 2011

Table of Contents

Introduction

This *Release Notes* highlights the main features as well as the known issues in this Schooner Membrain version 3.1.1. It also provides the basic instructions and special notes regarding the installation and operation of the product.

Important Notes

1. This Schooner Membrain release is compatible with Memcached version 1.2.6 and 1.4.

About Schooner Membrain

Schooner Membrain is a fully integrated technology solution that provides order-of-magnitude improvements in performance, scalability, service availability, and the potential for significant reduction in total cost of ownership for Web and cloud computing data centers. Schooner Membrain delivers industry-leading performance and caching capacity. It provides comprehensive data and service availability through a rich set of high-availability (HA) and disaster-recovery (DR) features, including Persistent Caching, Active/Active Replication, and Backup & Restore. Schooner Membrain is fully compatible with existing Memcached applications and tools, making it easy to integrate into existing Memcached environments.

Main Features

This section highlights the main product features of Schooner Membrain. Descriptions of the features follow after the list.

- Optimized for High-Performance Hardware Platforms.
- Comprehensive Data and Service Availability.
- Easy to Deploy, Manage, and Scale.
- World-Class Technical and Product Support.

Optimized for High-Performance Hardware Platforms

Schooner Membrain seamlessly integrates hardware and software into a powerful, yet easy-to-use technology solution that is optimized for dynamic, large-scale Web environments, enabling access to terabytes of high-performance, persistent cache capacity.

- **Schooner Membrain**
  - Unlocks the full potential of the hardware by optimizing utilization of all the available physical resources across a broad spectrum of workloads.
  - Enables full utilization and linear scalability of multi-core processors, intelligently manages caching from DRAM to flash, and delivers highly parallel read- and write-optimized flash memory access.
- **Optimized for High-Performance Hardware**
  - Multi-core X86 64-bit CPUs.
  - Solid-state secondary storage.
  - 512 GB of high performance caching capacity in a single server.

Comprehensive Data and Service Availability

With Web sites increasingly dependent on Memcached to deliver application performance, failures in the Memcached tier can lead to large, unexpected loads on the database, resulting in significantly degraded end-user experience. Schooner Membrain provides a comprehensive set of HA/DR features to ensure a consistent, satisfying user experience through both planned and unplanned downtime.

- **Schooner Replication & Automatic Failover**
  - Active/active replication fully utilizes the nodes in a mirror group.
  - No data is lost during outages, thanks to synchronous replication.
  - Automated failover delivers continuous service availability and eliminates the need for manual, error-prone user intervention.
  - Failover is transparent to client-side applications and requires no additional overhead.
- **Schooner Persistent Caching**
  - Cached data is immediately available, even after a power outage.
  - Delivers the persistence of flash memory with the performance of DRAM.
  - Eliminates performance-degrading, multi-hour or multi-day cache warm-up periods.
- **Schooner Back-up & Restore**
  - Full and incremental back-ups can be done to onboard HDDs or network attached storage.
  - Enables the cache data to be restored to a previous state in the event of application-driven data corruption, user error, or other data loss.

Easy to Deploy, Manage, and Scale

Schooner Membrain is compatible with existing tools and applications, with easy-to-deploy, simple, yet powerful centralized management and reporting:

- **Schooner Dynamic Containers**
  - Allows multiple "virtual" Memcached domains with different classes of service to be configured on a single appliance.
  - Enables higher consolidation, efficient resource utilization, and optimized handling of spikes.
- **Hot Key/Client Management**
  - Provides visibility into the most frequently used keys and the most active clients.
  - Both current and historical information is available to optimize performance.
- **Schooner CLI**
  - Easy-to-use CLI makes it easy to run scripts and manage multiple appliances.
  - Media Key for optional remote presence, predictive failure analysis, and automatic restart.
- **Integration with 3rd-party Tools**
  - Integration with 3rd-party tools, such as Nagios, Ganglia, and Cacti.
- **Compatibility**
  - Fully compatible with virtually all existing Memcached applications and tools – no client-side changes is required.

World-Class Technical and Product Support

Schooner offers customer support services to ensure reliable operations and timely response to service interruptions. Detailed information is available at http://www.schoonerinfotech.com/support.

Support for Binary Protocol

With support for Binary Protocol, the Schooner Appliance 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. Applications do not need to perform complex operations to split large objects into 1-MB-sized chunks to store them on the appliance.

Support for 128 Schooner Dynamic Containers

This feature enables 128 containers to operate simultaneously on a single Schooner Membrain server. Containers may be configured as small as 256MB. The enhanced user interface (GUI and CLI) makes it easy to manage a large number of containers.

The feature allows for simplified consolidation of a large number of Memcached 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.

Flexible Container Addressing Schemes

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

This feature works as follows:

- Container addressing using 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 unique names on the Schooner Centralized Administrator or CLI.

- Backward compatibility: container addressing using a specific TCP port number and <u>any</u> IP address

  This provides the same functionality as in the previous releases. In this scenario, a container can be configured to listen on all IP addresses and one specific TCP port number. Each container uses one 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 all IP addresses; otherwise, the application will not be backwards 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.)

Recommended Hardware Specifications

As an optimized technology solution for the new generation of Web and cloud-computing data centers, Schooner recommends the following hardware configuration:

Feature	Specification
Ports	High-speed Ethernet ports for low-latency interconnect. Either: • Bonded 1-G ports. • 1 x 10-G port.
Memory	64 GB DRAM.
Flash Memory	512 GB SSD.
Backup Storage	600GB on-board or networked storage.
Processor	2 four-core Intel “Nehalem” processors.
Operating System	x86_64/AMD64 Centos 5.4, 5.5, x86_64/AMD64 Redhat 5.4,5.5

Implementation Notes

This section explains how Schooner Membrain implements certain elements of the standard, open-source Memcached protocol.

- **Memcached Protocol 1.2.6 or 1.4**

  Schooner Membrain implementation is based on version 1.2.6 or 1.4 of the Memcached protocol and is compatible with applications based on either of the protocols.

- **Stats Sub-Commands**

  Certain "stats" sub-commands are implementation-dependent. Consequently, Schooner Membrain currently does not support the following stats sub-commands:

  - item statistics
  - item size statistics
  - slab statistics

  Schooner Membrain supports a Schooner-specific “stats all” command with which users can view extensive statistics on:

  - standard Memcached stats
  - Schooner Membrain stats
  - Flash storage stats
  - Schooner DRAM cache stats

- **Non-Eviction Containers**

  When a non-eviction container is nearly full, it is possible that the space freed up by deleting or overwriting existing items might not be immediately available for subsequent creates. Removing more items should solve this problem. In general, it is not recommended to run a non-eviction Memcached container near its full capacity (for example, more than 95% full).

- **Synchronizing Persistent Containers**

  Schooner Membrain introduces the concept of persistent containers and two new commands in the protocol: "sync key" and "sync_all". The former flushes a single object to persistent storage and the

latter flushes all dirty objects in memory. Due to limitations of the flash devices in the current schooner platform, the "sync" commands are relatively expensive. It is recommended that users perform batched commits using sync_all whenever possible instead of persisting objects individually using the "sync key" command.

- **Statistics for Persistent Containers**

  In the current Schooner Membrain implementation, the value of "total_items" in the "stats" command output is set to be equal to "curr_items" after a persistent container is recovered.

- **UDP**

  To provide better UDP performance, Schooner Membrain supports replying to UDP requests through multiple UDP reply sockets on the server. This means that clients might see a response to a UDP request coming from a different UDP port on the server from the port to which the request was sent. In the current implementation, Schooner Membrain allocates the UDP reply sockets from port 60000 and up. Users should not configure containers with UDP ports greater than 60000.

  Users need to be aware that if the client-side software uses UDP with the connect() system call, it will not work with the above scheme since the connect() system call restricts the socket to only receiving packages sent from the specified server port.

- **Inconsistency**

  In a mirrored pair, data is inconsistent when the value stored for a particular key differs between the nodes. Whenever a store is issued to one of the nodes, there is a transient interval when the data for the key differs between the nodes. Once the store has completed successfully, however, the data will be consistent.

  Aside from this basic transient inconsistency that occurs for all store operations, there are other scenarios that can create both transient and non-transient data inconsistency. These scenarios are:

  - **"Simultaneous" Stores to the Same Key on Different Nodes**
    This occurs when clients issue store commands to the same key to both nodes "at the same time". Because of replications, each client store command performs a local and a remote store. When a store on Node 0 occurs at about the same time as a store to Node 1, it is possible that Node 0 will have the data value from the store sent by a client to Node 1, and Node 1 will have the data value from the store sent by a client to Node 0.

    There is currently no automatic mechanism to detect and correct this type of inconsistency when it occurs. Gets to Node 0 will see a different data value than gets to Node 1.

    To avoid this inconsistency scenario, users should issue requests to distinct nodes using distinct keys.

  - **Flush All**
    It is possible for the two nodes in a mirrored group to become inconsistent if a flush_all operation is issued while put/set operations are still in progress. This is because clock skew between the nodes makes it impossible to ensure that both nodes make the same determination whether a put/set operation is before or after the flush point.

    To avoid this inconsistency, all put/set operations should be quiesced (rendered inactive) on both nodes before a flush_all operation is sent to either node.

  - **Skewed Expiry Times**
    Skew between object create times and the time base on each node can result in windows of time in which an object is expired on one node but not yet on the other.

Note: It is strongly recommended that all appliances keep NTP synchronization enabled at all times.

- **Persistent Containers and Double Failures**

  When Membrain is restarted in a recovery (the node crashed and must now be restored from the "surviving" node), it must first determine if a double failure has occurred.

  A double failure occurs when both nodes crash. When this occurs, Schooner Membrain automatically starts up all non-persistent containers and activates VIP groups for both its node and its peer node (because its peer is not active).

  Persistent containers, however, are stopped because there is no reliable way for Schooner Membrain to determine which node holds the authoritative copy of the containers.

  The user must manually issue the “reinstate” command from the CLI to start the recovery of persistent containers. Typically the node that exited most recently (per log records) will have the most current data. The user should use system mechanisms such as Syslog to determine which node has the most recent data and should be designated as the “recovery master node”.

  The "reinstate" command does the following: Specifies and starts the “recovery master node”. At the same time, causes the other node to go down and then come up again. Once restarted, the other node will recover both persistent and non-persistent container data from the “recovery master node”.

- **Changing Time-of-Day**

  Indiscriminate changes to the time-of-day on replicated Schooner Membrain servers can result in peculiar behavior (because of flush_all and expiry operations). To prevent this, users should only change the time-of-day on a node when Schooner Membrain is not running. In other words, Schooner Membrain should be shut down before changing the time.

- **Defunct Container**

  A container with status "defunct" means that the container has encountered an error and no further operations can be performed on it.

  Workaround: Restarting Schooner Membrain will clear the error and allow container operations.

Persistent Storage Considerations

Schooner Membrain extends the standard memcached client command set to control the flushing of persistent objects to secondary storage: "`sync`" and "`sync_all`". These commands ensure that buffered writes are written to storage devices. This is sufficient to ensure that the objects are persisted if the storage devices and storage device controllers guarantee that successful write operations are persistent. Not all storage devices or controllers make this guarantee.

Many SSDs, those built using Sandforce components for example, guarantee that any buffered data is written to flash even if there is a power interruption. These SSDs must be used with controllers that do not cache writes (or cache writes using a non-volatile RAM).

Intel SSDs do not guarantee persistence because write data is staged in a volatile DRAM cache within the SSD. They do provide a mechanism to flush this cache to flash, however. Schooner Membrain has the option to issue this Intel-specific flush operation after all "`sync`" and "`sync_all`" calls. The CLI “`storage_sync`” command provides the opportunity to enable this feature. This assumes that the SSD controller has write caching disabled or has a non-volatile cache.

Similar precautions must be observed when using hard disks for secondary storage.

Changes in this Release

This release includes fixes for the following issues:

1. Membrain can now uses bond0 as an admin interface.
2. Problems with Linux NUMA configuration have been fixed.

Known Issues

The following are the major known issues found in this Schooner Membrain release:

3. This version of Schooner Membrain supports configuration via CLI only. A GUI will be supported in a subsequent release.
4. SASL is not supported in this release. SASL will be supported in a subsequent release.
5. The maximum number of connections is limited to 50,000.
6. For non-persistent eviction mode containers, there is a small chance for an object deleted with an expiry time in the future to be evicted from the server before the expiry time is reached.
7. By default, the open-source Memcached auto-negotiates client connections to find out whether they are using the binary or ASCII protocol. However, the user can explicitly specify the protocol clients must speak using the "-B" command line option. Currently, the Schooner Membrain only supports auto-negotiation.
8. For persistent containers, the Schooner "sync/sync_all" commands are currently only supported in the ASCII format. Therefore, if the user is using the binary protocol, at least one separate TCP connection may be needed for issuing the sync commands.
9. Schooner Membrain allows the user to dynamically enable or disable SASL support for a container (though the container must be stopped to make the change). However, the change only affects new connections and has no effect on existing connections. Therefore, a client might be able to access the container without authentication if it keeps an open connection and reuses it after SASL is enabled.
10. The server might run into out-of-memory issues under the combination of a large number of concurrent connections and large-sized objects.
11. Before creating or deleting mirror groups, ensure that membrain is shutdown on the 2 nodes that are affected.
12. The message interface setting and Virtual IP addresses may be reset when a server joins a Schooner cluster. You will have to set them back to your desired values after joining the cluster.
13. This version of Membrain supports a maximum of 32 CPU cores and 512GB of flash storage.
14. If the Schooner Cluster Manager (SCM) fails to start, it is likely due to a java version mismatch. Please do the following:
    - edit /etc/init.d/scm
    - set the variable: javaExe=/usr/java/jre1.6.0_18/bin/java

Release History

- 05/17/2011: Schooner Membrain v3.1.1 release
- 02/07/2011: Schooner Membrain v3.1 GA release
- 11/21/2010: Schooner Membrain v3.1 Beta 1 release

- 06/11/2010: Schooner Appliance for Membrain v2.5.0 release.
- 02/26/2010: Schooner Appliance for Membrain v2.1 Service Pack 3 release.
- 02/03/2010: Schooner Appliance for Membrain v2.1 Service Pack 2 release.

Technical Support

Technical support for Schooner products is available from the following sources:

- **Phone:** (408) 888-1619; (877) 888-5064 (toll free)
- **Fax:** (408) 736-4212
- **Email:** support@schoonerinfotech.com
- **Web:** http://www.schoonerinfotech.com/support