Release Notes

Schooner Appliance for Memcached

Version 2.0

SCHOONER
INFORMATION TECHNOLOGY

Technical Support

Go to the IBM support Web site at http://www.ibm.com/systems/support/to check for technical information, hints, and tips.

Documentation ID: I-M-v2.0-10120_619-RN-002

Schooner Appliance for Memcached™ - Release Notes

Issued November 2009.

Schooner Information Technology
501 Macara Ave., Suite 101
Sunnyvale, CA 94085, USA
Tel: (408) 773-7500 (Main)
(877) 888-5064 (Sales and Support)
Fax: (408) 736-4212

Contents

Chapter 1. Introduction

What's New in This Release

This is the second Beta release of the Schooner Appliance for Memcached. Changes since the first Beta include:

- Container replication
- Container backup and restore
- SSD RAID
- Rolling upgrade
- Dynamic container management
- Hotkey management

System Requirements

Network Requirements

High-speed networking. This release assumes that you have:

- a 10Gb Ethernet network to handle application traffic
- a 1Gb Ethernet network to handle administration traffic

Be mindful when connecting the Schooner appliance to your network, since both connectors may look the same. See the *Installation & Setup Guide* for detailed connection instructions.

- **Note:** If you don't have a 10Gb application subnet, you may use a 1Gb subnet on **eth0** through **eth7**. It is recommended that **eth0** be used only for administration and monitoring. Contact IBM Support for configuration instructions.

Administration System Requirements

Windows or 32-bit Linux. Graphical displays in the Schooner Administrator™ control console use Adobe Flash. In this release, the displays work well with Flash plug-ins in Windows and 32-bit Linux, but not in 64-bit Linux.

As a workaround, you may manually install the 32-bit Flash plug-in on a 64-bit Linux system, but this is not recommended. If possible, monitor and administer your Schooner system using a Windows or 32-bit Linux system.

- **Note**: This restriction does not affect Memcached client applications, only systems that run Schooner Administrator to manage the Schooner system.

Your browser must have cookies enabled in order to run the Schooner Administrator.

Chapter 2. Known Bugs

This chapter documents known bugs and problems you may encounter, along with workarounds if available. The number in parentheses, like (#798), refers to Schooner's internal tracking number, if any.

Please note that bug fixes are not guaranteed in any specific release.

Defunct Container

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

Workaround: A restart of Memcached should clear the error and allow container operations. If this does not solve the problem, then please contact Schooner support.

Delete Time Parameter

For 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.

Workaround: None

System Unresponsive During Core Dump

If the Schooner Memcached application crashes, the system may become unresponsive for possibly several minutes while a core file is being generated. However, Schooner Memcached will restart after the core file has been written completely.

Hotkey Utility

Currently, the Hotkey utility may report incorrect data after running for an extended period of time.

Workaround: None

Degraded Performance When Restarting Memcached with No Container

When restarting Memcached with no containers defined, a minimum-sized buffer is allocated for a background I/O-intensive thread. If a large, persistent container is subsequently added, performance may be significantly degraded after some time due to the small buffer.

Workaround: Restart Memcached after adding all persistent containers.

Backup Utility Issue

If you attempt to do a backup on a stopped container, the backup utility may become frozen.

Workaround: Make sure that the container has been started (i.e., status is “on”) before you perform a backup operation.

Default IPs Causing Split-Brain Handler to Malfunction

Each NIC on the Schooner Memcached appliance comes with a default IP address. The issue arises when the user runs the application in mirrored mode and brings down one of the Memcached instances. In that situation, the split-brain handler will mistakenly treat the surviving node as the dead node (because its default IP is exactly the same as the one on the dead node) and take it down as well. As a result, both nodes of the mirrored pair are brought down.

The best solution is not to provide default IPs for all the NICs except for eth0. The user can also work around this issue either by editing each `ifcfg-ethX` file and setting `ONBOOT=no` (recommended) or by assigning a unique IP to each of the NICs to make sure that they do not collide with other addresses.

Chapter 3. Memcached Implementation Notes

This chapter explains how Schooner Memcached implements certain elements of the standard, open-source Memcached protocol. The number in parentheses, like (#269), refers to Schooner's internal tracking number, if any.

Memcached Protocol 1.2.6

The Schooner Memcached implementation is based on version 1.2.6 of the Memcached protocol, and is intended to be plug-compatible with applications based on that protocol.

Stats Sub-Commands

Certain "stats" sub-commands are specific to the actual implementation of memcached, as a result, Schooner Memcached currently does not support the following stats sub-commands:

- item statistics
- item size statistics
- slab statistics

Schooner Memcached supports a "stats all" command with which users can view statistics on:

- standard memcached stats
- Schooner Memcached stats
- Flash storage stats
- Schooner 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 Memcached introduces the concept of persistent memcached containers and correspondingly, 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 Memcached 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 Memcached supports replying to UDP requests through multiple UDP reply sockets on the server. This means that clients might see a response for a UDP request coming from a different UDP port on the server from the one that the request was sent to. In the current implementation, Schooner memcached allocates the UDP reply sockets from port 63000 and up. Therefore, users are encouraged to avoid configuring containers with UDP ports greater than 63000.

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 restricts the socket to only receive 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

Clients issue store commands to the same key on both nodes "at the same time". Each 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 are advised to issue requests to distinct nodes using distinct keys.

Flush All

It is possible for the two nodes in a mirrored cluster 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 and 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 memcached 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, memcached 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 Memcached to determine which node holds the authoritative copy of the containers.

The user must manually select the ‘recovery master node” from the Schooner Administrator GUI to start the recovery of persistent containers. Typically the node that crashed 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 therefore should be designated as the “recovery master node”.

The "container 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. Upon coming back to life, the non-recovery master node will start to 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 Memcached appliances can result in peculiar behavior. To prevent this, users should only change the time-of-day on a node when Memcached is not running. In other words, memcached should be shut down before changing the time.

Chapter 4. Planned Enhancements

The following features are not implemented in the current release, but are planned for the future. The number in parentheses, like (#550), refers to Schooner's internal tracking number, if any.

Some of these features are described in the product documentation. Please note that this documentation is preliminary, and implementation is not guaranteed when currently planned.

Online Help

A future release will implement an online, context-sensitive help system, accessed from the right-hand panel of Schooner Administrator.

Workaround: Use the included documentation instead.

Additional Clusters

A future release will implement support for more than one Schooner Appliance cluster in the Schooner Administrator.

Workaround: None.