GMS Watchdog Capability: Enhancing GMS Failure Detection One Pager

1. Introduction

1.1. Project/Component Working Name:

GMS Watchdog capability

1.2. Name(s) and e-mail address of Document Author(s)/Supplier:

Joseph Fialli: joe.fialli@sun.com

1.3. Date of This Document:

18 Feb 2009

2. Project Summary

2.1. Project Description:

In this document, we introduce a new GMS functionality that enables an external entity or an in-process GMS client to report a member failure to GMS. This functionality addresses an incompatibility between GMS and other external processes(NA, potentially SAF/AMF) that monitor and restart a GMS member faster than GMS heartbeat-based failure detection is capable of detecting failure. Currently, GMS misses reporting failure of a member when a Glassfish NodeAgent detects and restarts a server instance in a shorter period of time than GMS failure detection can verify that the server instance has failed.

The Glassfish NodeAgent would need to become a GMS WatchDog member to notify GMS cluster that a server instance has failed. GMS subsystem, specifically the GMS Master node of the cluster, would then notify all members of the cluster that the instance has failed. The GMS WatchDog functionality enables external entities using more efficient failure detection to enhance overall GMS failure detection time and accuracy.

Faster and accurate failure reporting such as Node Agent's process monitoring based approach helps GMS clients such as SGCS 1.5's ConvergedLoadBalancer to failover faster and thus reject lower number of incoming calls/requests than at present. Additionally, this will benefit the Replication module by providing a more predictable state transition machinery than at present helping it become more efficient in managing replication partner connections.

2.2. Risks and Assumptions

- Assume that SGCS/SAF integration[1] will allow Glassfish NodeAgent to run in a GMS watchdog mode. GMS watchdog only requires detection of failed server instance by the NA, it does not require that the NA restart the server instance. Of course, for this proposal to work, NA must detect failure before SAF does. Since SAF is using heartbeat failure detection, it is probably a safe assumption that it will not detect and restart the server instance before NA is able to report that the server instance has failed to GMS subsystem. (See Section 3.1 for

Default SGCS Group Management Service failure detection times and observed NA failure detection times.)

- Node Agent's memory footprint would increase marginally. A prototype must be built to get a good estimate of how much the Node Agent footprint is currently and to what extent it would increase with GMS integration. We will consider ways to turn off any unnecessary Shoal feature that is irrelevant to Node Agent such as initializing and participating in the shared Distributed Cache.
- Node Agent will need to initialize a GroupManagementService instance (and resulting threads) for each cluster to which instances it is managing belong. This will have a marginal impact on footprint as well. This should be measured as part of the prototype exercise.

3. Problem Summary

3.1. Problem Area:

The Glassfish NodeAgent daemon is able to detect that a server instance, that is also a GMS cluster member, has failed quicker than GMS. The NodeAgent restarts the failed instance sooner than GMS heartbeat based failure detection is able to detect the failure. This only became a more pronounced problem recently due to some combination of faster machine speeds and Glassfish optimizations that enable a SGCS/Glassfish app server to restart quicker than the default SGCS/Glassfish Group Management Service defaults for detecting failure.

Current SGCS/Glassfish Group Management Service defaults are:

```
Heartbeat frequency:   2000 ms (2 seconds)
Max Missed Heartbeats: 3
Validate Failure:      1500 ms. (1.5 seconds)
```

GMS default configuration settings from SGCS domain.xml:

```
<group-management-service fd-protocol-max-tries="3"
                          fd-protocol-timeout-in-millis="2000"
                          vs-protocol-timeout-in-millis="1500">
```

Given the above default configurations, it takes GMS minimally 6 seconds to suspect a server instance to have failed and a further 1.5 – 3 seconds to verify that a server instance has failed. Once the server instance has restarted, the failure verification step will fail and GMS has missed its chance to send out a FAILURE notification. GMS does detect that a server instance has been restarted before it detected failure with following ShoalLogger events that are sent to SGCS server log of DAS (or if DAS is down, which ever member of the GMS cluster is the master node.)

```
 [#|...|WARNING|sun-glassfish-comms-server1.5|ShoalLogger|...;
Instance n2c1m4 was restarted at  4:13:19 PM PST on Feb 4, 2009.|#]

[#|...|WARNING|sun-glassfish-comms-server1.5|ShoalLogger|...;
```

Note that there was no Failure notification sent out for this instance that was previously started at 4:11:31 PM PST on Feb 4, 2009|#]

However, it is too late to report a FAILURE_NOTIFICATION when GMS detects that the instance has already been restarted.

Given that a Glassfish NodeAgent is always co-located on the same machine as the server instance that it manages, the NodeAgent can use a more efficient and less reliable technique (based on network reachability) to detect that a server instance has failed. With current Sailfin testing configurations, the Glassfish NA is being observed to detect and restart the server instance in less than 7.5 seconds, but not quicker than 6 seconds. GMS sends out the failure suspected notification after max missed heartbeats. Given that GMS relies on heartbeats and not necessarily being co-located with other members of the GMS cluster, its technique for detecting failure must balance waiting a certain amount of time before declaring that an instance has failed; otherwise, GMS runs the risk of falsely declaring a server instance as failed due to network congestion/slow server instance response.

This proposal enables GMS to leverage already existing efficient server instance failure detection of the Node Agent by bringing the Node Agent into GMS member state tracking and reporting.

3.2. Justification:

Functional:

GMS will no longer miss sending out FAILURE notifications for failed members that restart in a shorter period than GMS heartbeat failure detection is capable of detecting.

Given that NodeAgent detects FAILURE much quicker than GMS does, this optimization will result in GMS clients receiving FAILURE notification far quicker than the minimum of 7.5 seconds that it takes for GMS heartbeat based notification to detect failure.

The enhanced FAILURE notification times could benefit Sailfin CLB, DCU and SSR, all subsystems that handled GMS notifications.
For instance, at current call rates, we estimate that the CLB will be able to save a minimum of 1200 calls (with replication enabled) and a maximum of 3600 calls (without replication enabled) due to the faster failure reporting capability.

4. Technical Description:

4.1.

1) Shoal Modifications

Provide a WatchDog capability (Member Type definition and API extension) to allow for notification of failed members.

2) Glassfish NodeAgent Modifications

- Become a GMS WatchDog – requires a minimal lines of code to integrate and an api call to report failure.

Report server instance failure via GMS WatchDog capability. The server instance name should be sufficient for GMS.

4.2. Details:

Strawman:
- Add WATCHDOG enumerator to GMS GroupManagementService.MemberType.

- For each cluster defined in domain.xml, NA joins the cluster as a watchdog during its startup or during its lifetime, when it determines that it is managing one or more clustered instance(s).

```
import com.sun.enterprise.ee.cms.core.GMSFactory;

GroupManagementService gmsModule =
GMSFactory.startGMSModule("GlassfishNodeAgentName",
               clusterName, MemberType.WATCHDOG, props);
```

Place each gmsModule in a hashmap mapping clustername to gmsModule.

- When NA detects that a server instance has failed, determine the server instance's cluster that it belongs to and lookup gmsModule reference for the cluster. Call the following new method on this reference to signal that the server instance as failed.

```
gmsModule.broadcastFailure(String serverInstanceName);
```

(Quick inspection of Server info in nodeagent did not look like it was tracking which cluster an instance belonged to. This could easily be taken care of using ConfigAPI.)

- The GMS call above will broadcast that serverInstanceName has been observed to fail by the WATCHDOG. The GMS MasterNode will discontinue outstanding efforts to validate that this server instance has failed and will send the FAILURE notification to all instances in the cluster. If there is no MASTER node for cluster at the time this method is called, all instances will record the WATCHDOG report (in HealthMonitor cache) and when a MasterNode is elected subsequently, it will see the outstanding WATCHDOG event and send failure event view change to all instances in the cluster and clear the outstanding WATCHDOG event.

4.3. Bug/RFE Number(s):

None that I am aware of.

SSR, CLB and DCU are aware that FAILURE notifications may be missed when Glassfish NodeAgent is running. SSR relies on GMS Join notification to trigger reshape. I believe that the result of the missed/non-timely FAILURE notification for CLB and DCU is SIP clients receiving a 503 error. As soon as CLB and DCU are aware of failure, they transfer traffic to other instances in the cluster that are up. Quicker notification will result in less 503 errors.

4.4. In Scope:

4.5. Out of Scope:
-

4.6. Interfaces:

4.6.1. Exported Interfaces

- GMS Interface to export to service instance monitors (specifically NA)

TBD

4.6.2. Imported interfaces

-

4.6.3. Other interfaces (Optional)

-

4.7. Doc Impact:

4.8. Admin/Config Impact:

4.9. HA Impact:

Only beneficial. It would receive FAILURE notifications that GMS was missing when NodeAgent restarted failed instance in shorter period of time than GMS could detect failure in.

4.10. I18N/L10N Impact:

NONE

4.11. Packaging & Delivery:

-

4.12. Security Impact:

NONE

4.13. Compatibility Impact

NONE

4.14. Dependencies:

-External processes that monitor services:
Glassfish NA
MMAS SAF/AMF

5. Reference Documents:

[1] SGCS/SAF Improved Integration One Pager, 19 Jan 2009,olivier.corbun@ericsson.com

6. Schedule:

6.1. Projected Availability: