1. Introduction

1.1. Project/Component Working Name:

Self managemenet Rules

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

Sankara Rao Bhogi sankara@dev.java.net

1.3. Date of This Document:

08/09/06

Updated on 10/11/06

2. Project Summary

2.1. Project Description:

To provide out of the box Self Management features, set of builtin management rules will be provided to alert critical conditions and in some cases to take actions when those events happen.

2.2. Risks and Assumptions:

Self Configuration/Self Healing rules requires some experimentation, thus they might go through some iterations before they stabilize.

3. Problem Summary

3.1. Problem Area:

// What problem or need does this project solve?

Application servers expose wealth of monitoring information. As of now, this information has to be processed by users or other external tools to make some meaningful representation of this collected data. Based on the data, user needs to take an appropriate action. Unfortunately in varying load conditions, by the time data is processed and an action is taken manually by the user, that action may not be ideal at that point of time. Indeed, if we could embed the intelligence to process the monitoring information with in the server and take an appropriate action based on that data with in the appserver will be an ideal thing to do and makes appserver management simpler.

3.2. Justification:

// Why is it important to do this project?

Simplified/auotomated management/administration.

4. Technical Description:

4.1. Details:

Introduction to self management framework in glassfish can be found at:

- GlassFish Project : Self Management home
  https://glassfish.dev.java.net/javaee5/selfmanagement/selfmanagementhome.html
- Blog: "Self Management Framework in GlassFish"
  http://weblogs.java.net/blog/sankara/archive/2006/02/self_management.html
- Blog: "GlassFish : Self Management Rules"
  http://blogs.sun.com/roller/page/technical?entry=self_management_rules

Proposed Management Rules:

- Self Configuring/Self tuning of JDBC Connection pools
- Self Healing/Diagnosing
  - Out of memory detection, cleanup and alert notification
  - Hang detection and recovery: Instance and thread hangs
  - Dead queue Message Alert
  - Disk full detection and server logs cleanup

Dead Queue Message Alert:

Dead Message may be defined as a message that is removed from the system for a reason other than normal processing or explicit administrator action. A message might be considered dead because it has expired, because it has been removed from a destination due to memory limit overruns, or because of failed delivery attempts.
When a message is dead, typically it would be kept in a specialized destination called "Dead Queue Message" and this would be an event of interest to the administrator.
This management rule, identifies such an event and sends a mail alert to the configured reciepient.

Connection Pool Management:

Primary objective of a Connction Pool is to reduce the high cost of acquiring new connections and at the same time, not to hold resources unnecessarily. While manual staic configuration, satisfies the need of reducing the number of new connections acquired, there may not be a signle suitable configuration which caters to dynamic load.
This management rule, attempts to dynamically tune the connection pool.
At present maximum connection pool is specified at the connection pool

defintion and would apply to each instance in the cluster.
As instances added/deleted to the cluster, this value may probably
needs to be changed, so that the resource usage is optimum. This
mangement rule decides the max pool size for a given maximum number
of connections possible to a given data source. Typically steady
pool
size is used so that for an average load, new connections don't
have to
be made. This rule, tunes stead pool size according to the load.

Out of Memory Management:
Low Memory is an important system event and administrator would like
to
know, when such an event happens. Further, if internal pools and
caches
can be pruned and cleaned it might help to an extent. Some times,
it might occur because of a continuous high load, in such a case,
quiescing the instance for a short time might help.

Thread Hang Detection and Instance hang Detection:
For various reasons (bug in logic, resource contention, dead locks
etc),
a request thread might not finish its execution in a reasonable time
and
if most of threads goes into such state, server may not be in a
usable
state. Such a state might be of interest and some times restarting
the
instance may be an approriate action.

Log Cleanup Rule:

As old logs get accumulated, available disk space to write new logs
might be running out. This is a mechanical task and can be taken
care
by a management Rule.

4.2. Bug/RFE Number(s):

4.3. In Scope:

4.4. Out of Scope:

4.5. Interfaces:
4.5.1 Exported Interfaces
Interface: Management Rules and their parameters will be

pre populated into domain.xml in the cluster and enterprise profiles.

1) Management Rule: DeadQueueMessageAlert

Threshold: 1
Interval: 180 secs
OffSet: 0
MailRecepients : No Default
MailResource : No Default

2) Management Rule: ConnectionPoolManagement

DefaultMaxConnections : 30
PoolNames : No Default . Possible values * and comma
separated name value pairs of PoolName and Max Conns
Sample Size : 5
Sample Interval : 30 secs

3) Management Rule: OutOfMemoryManagement

MemoryPoolnames : "Perm Gen, Tenured Gen"
Threshold : 80 (percent)
OffSet : 5 (Percent)
MailRecipients: No Default
MailResource: No Default
ListenerClasses: No Default, Comma separated fully qualified class names.

4) Management Rule: ThreadHangDetection
ThresholdWait: 80 secs
StopThread : false (true | false)
MailAlertAppRef : MailAlert

5) Management Rule: InstanceHang

RequestTimeoutInSeconds : 10
Restart : true

6) Management Rule: LogCleanupRule

PartitionSpace : ${com.sun.aas.instanceRoot}/logs
serverLogFilter : true
AsadminAccessLogFilter : true
ServerAccessLogFilter: true

Stability: Evolving

Comments: All the in built management rules are visible

Interface : com.sun.appserv.management.event.TraceEventHelper class
Stability: Evolving
Comments: Helper to class to get the trace event details. Would be
part of appserv-ext.jar.

Interface : com.sun.appserv.management.event.StatisticMonitorNotification class
Stability: Unstable
Comments: Notification type for complex attribute monitoring. Would
be part of appserv-ext.jar.

4.5.2 Imported interfaces

Interface: JMX 1.2
Stability: Standards
Exporting Project: JSR 3

Interface: AMX API
Stability: Evolving
Exporting Project: Application Server

4.5.3 Other interfaces (Optional)

4.6. Doc Impact:
Management rules along with the configurable parameters have to be
documented.

4.7. Admin/Config Impact:

4.8. HA Impact:

4.9. I18N/L10N Impact:

4.10. Packaging & Delivery:

4.11. Security Impact:

4.12. Compatibility Impact

4.13. Dependencies:

5. Reference Documents:

6. Schedule:

6.1. Projected Availability:
Aligns with glassfish V2 schedule.