Redundancy

Scope

- **Fault Tolerance**, Resilience and Redundancy provide high-availability of JobScheduler for a number of outage scenarios:
  - High Availability requires the system including JobScheduler, database, storage etc. to be available, not just one component.
  - High Availability is oriented towards specific outage scenarios, not towards any possible failure.
- **Master / Agent Resilience** includes a number of measures for operational robustness:
  - **Master / Agent Reconciliation** allows continued execution of tasks in case of recoverable **Network Connection Loss**.
  - **Master Service Recovery** includes supported measures after a **Master Service Failure**.
  - **Database Service Recovery** includes the capability to recover in case of **Database Connection Loss**.
- **Master / Agent Redundancy** includes a number of architecture decisions:
  - **Master Clusters** provide redundancy of Master instances in a network.
  - **Agent Clusters** can be used to compensate the outage of a server that runs an Agent.
- **Recovery Strategies** provide an overview of means how to restore the scheduling service

Master Cluster

Feature

- JobScheduler Master supports **Cluster Operation** with redundancy of the involved cluster members.
  - see **Passive Cluster**
  - see **Active Cluster**
  - see **Master / Agent Cluster**
- Clustering is frequently used for high-availability and in some cases for improved performance.

Agent Cluster
Feature

- The JobScheduler allows multiple Agents to be specified for a single Process Class.
  - The JobScheduler Master contacts Agents
    - by fixed priority scheduling:
      - JobScheduler Master will select the first available Agent for execution of jobs.
      - If an Agent is not available then the next available Agent is selected from the Agent Cluster.
    - JS-1554 - Getting issue details...
  - by round-robin scheduling:
    - JobScheduler Master switches the Agent for each job execution.
    - If an Agent is not available then the next available Agent is selected from the Agent Cluster.
    - JS-1188 - Getting issue details...
  - If the designated Agent is not available then the next Agent listed in the process class configuration will be contacted. This procedure will be repeated until an Agent is found that can execute the job.
  - Use cases for this scenario include
    - all Agents running on different server nodes: the switch to the next available Agent implements a fail-over to the next server node.
    - a number of Agents running on the same server node: the switch to the next available Agent implements redundancy of Agents within a single server node.

Feature Availability

- Scenario supported with release 1.10

Delimitation

- This feature is not intended for load sharing as the JobScheduler will always use the first available Agent.
- This feature is not intended for scalability as it does not allow the execution of jobs in parallel on a number of Agents (clustering).

References

Change Management References
<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>T</th>
<th>Created</th>
<th>Updated</th>
<th>Due</th>
<th>Assignee</th>
<th>Reporter</th>
<th>P</th>
<th>Status</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS-1554</td>
<td>JobScheduler Master supports Fixed Priority Scheduling and Round-Robin Scheduling for Tasks with Agents</td>
<td>📅</td>
<td>Dec 09, 2015</td>
<td>Feb 23, 2016</td>
<td></td>
<td>Joacim Zschimmer</td>
<td>Andreas Püschel</td>
<td></td>
<td>🔙 RELEASED</td>
<td>Fixed</td>
</tr>
<tr>
<td>JS-1188</td>
<td>A process_class can reference multiple JobScheduler instances</td>
<td>📅</td>
<td>Aug 15, 2014</td>
<td>Aug 05, 2016</td>
<td></td>
<td>Oliver Haufe</td>
<td>Stefan Schädlich</td>
<td></td>
<td>🔙 RELEASED</td>
<td>Fixed</td>
</tr>
</tbody>
</table>

2 issues

**See also**

- How to use Agents with Fixed Priority Scheduling and Round-Robin Scheduling