CMUG – March 2018

Inventory Re-design and Tuner Stability – 21st March 2018

Marimba Inventory Redesign
(using Hazelcast as the medium)

Inventory Redesign – Quick Recap.

- Replaces checksums based workflow.
- The new model comes with a much more easy way of tracking the continuity of scan reports in the whole reporting process (by using a SEQ_ID that is now generated and included as part of each scan report).
- Performance bottlenecks caused by checksum mismatch at the plugins (resulting in full report insertion) is now prevented by an elegant, de-centralized tracking engine.
- The Plugin is accompanied by a tracking engine that ensures the reports sequence is maintained. Upon encountering a mismatch, it is auto-fixed online with the endpoint in the same conversation and the correct scan report is sent up in the processing chain.
- Assurance of differential reports is shot up to a higher extent with this State Database enabled Tracking Engine residing in each plugin (repeaters and mirrors both applicable)

- **Advantages**
  - Enable it easily - It takes just one change in configuration to get the Tracking engine in action.
  - Bulk Checksum sync in repeaters and mirrors is out of scene now. This reduces load on the network on the repeaters and saves database roundtrips in the mirrors. Especially for geographically distributed repeater / mirror farms this will be a nice to have feature.
  - Checksums are not sent along with the scan report (in the request headers) from the endpoints either. This reduces data being sent over the network.
  - Error detection and correction both taken care of in the initial stage of the reporting chain.
  - Tracking is also done at the endpoint for easy resolution of conflicts in continuity of reports sent to the plugin.

Inventory Redesign – Existing Solution.

- Existing solution maintains the scan status of end-points in database (referred as State Database)
- Requires additional database to be configured, which can be
  - Cassandra
  - SQL Server
  - Oracle

Inventory Redesign – Using Hazelcast.

- Hazelcast is a clustering and highly scalable data distribution platform for Java.
- JVMs that are running Hazelcast will dynamically cluster so it allows us to easily share and partition our application data across our cluster.
- Hazelcast is a peer-to-peer solution (there is no master node, every node is a peer) so there is no single point of failure. Default configuration comes with 1 back up so if one node fails, no data will be lost
- Although by default Hazelcast will use multicast for discovery, it can also be configured to only use TCP/IP for environments where multicast is not available or preferred.
- Instead of approaching the database in all the time, with Hazelcast it allows us to store and share information between different nodes in a cluster, at a much faster rate.
- The cluster consists of multiple of nodes. Node is referenced here as a piece of code that can able to run the Hazelcast instance to make cluster and changes in it

- Since the operation on scan reports happen at the plugin, the nodes in the Hazelcast cluster need to be the JVMs that host the plugin. In other words, all transmitters that host inventory plugin will act as the nodes in the Hazelcast cluster

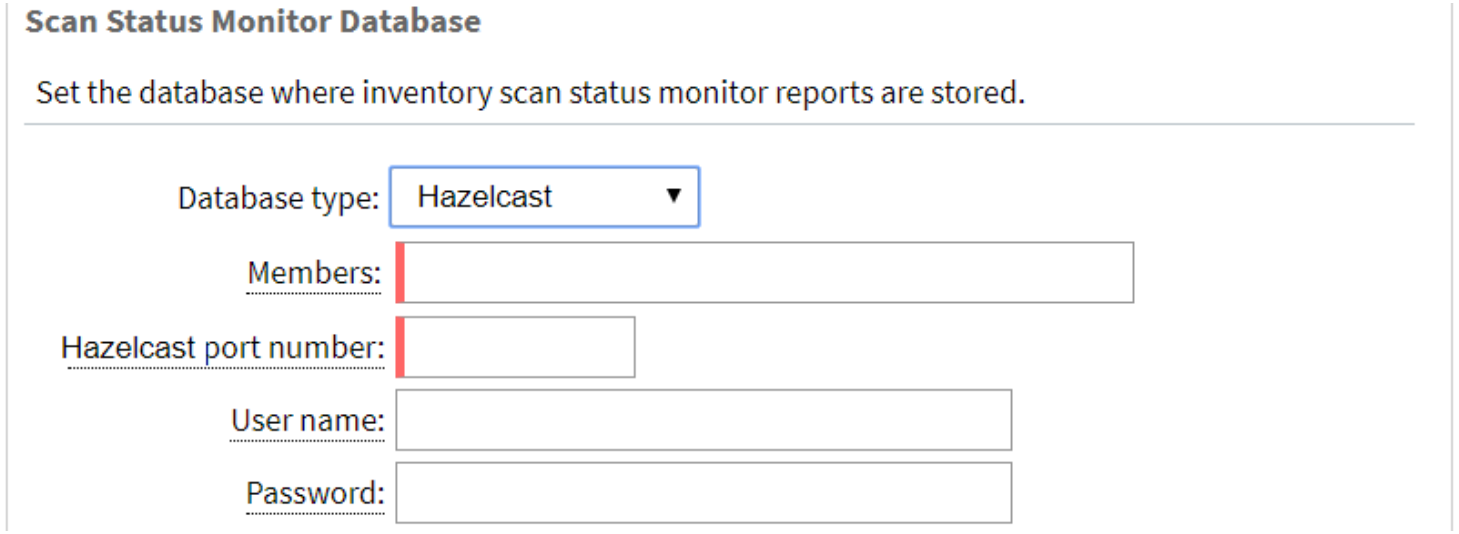
Inventory Redesign – Using Hazelcast : Pre-Requisites

- Inventory Service/Plugin Upgrade
- Report Center Upgrade
- "invdb" schema change to add one new table
  - Using Schema Manager (or)
  - Executing a standalone Script

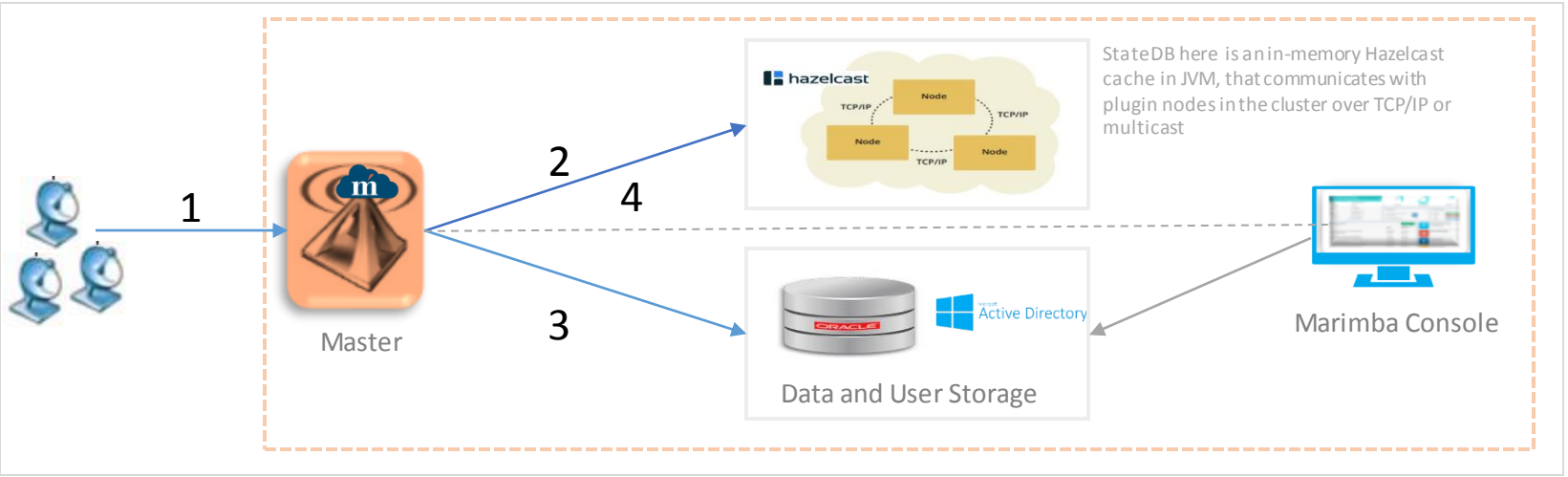
No dependency on external Database or Application

Inventory Redesign – Using Hazelcast: Configuration

- Inventory Plugin Configuration

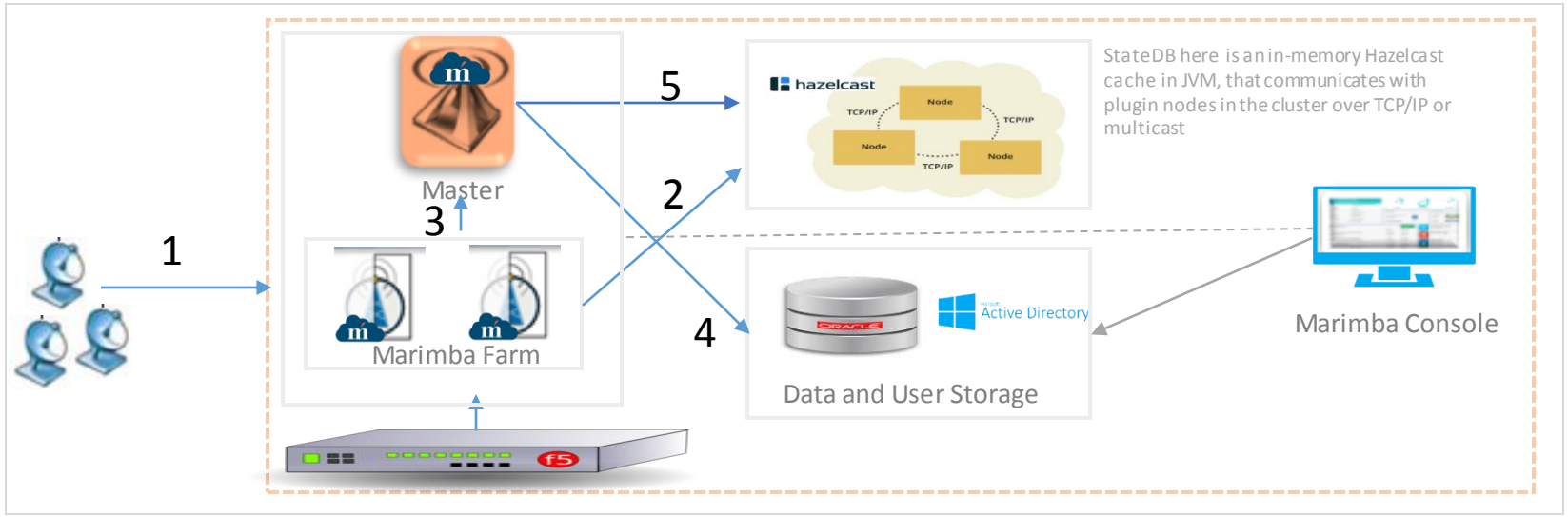


CASE: Plugin is configured to directly insert report into inventory database

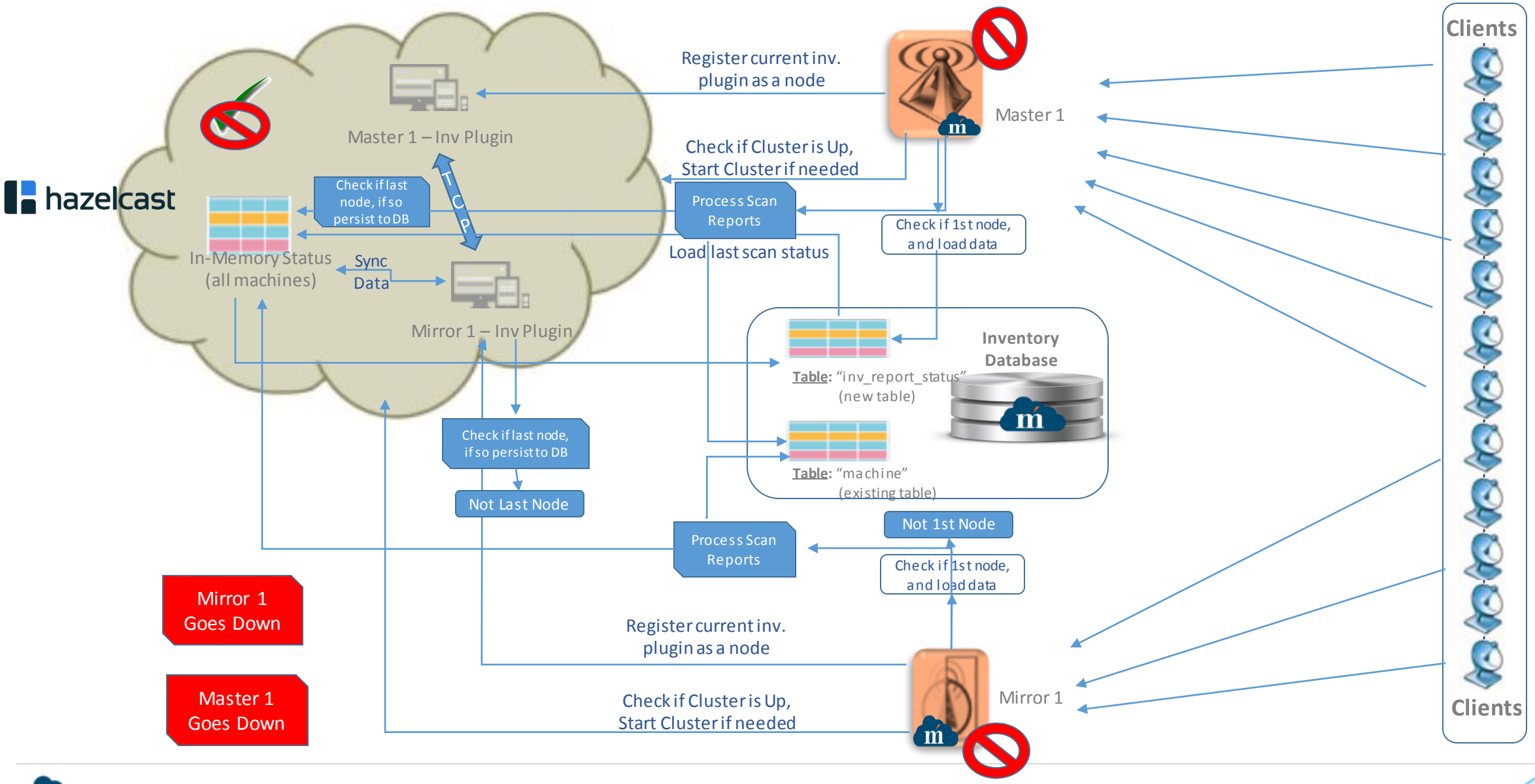


1. Endpoint scans and sends report to plugin based on acknowledgement received from the plugin
2. Plugin accepts/rejects report based on its query status from Hazelcast in-memory cache. It compares last scan status fetched from Hazelcast in-memory cache for an endpoint with the version of report received from the Endpoint.
3. If plugin accepts report it queues and then inserts report into the inventory database
4. Plugin updates Hazelcast in-memory cache with latest status

CASE: Plugin is configured to forward to Inserter and Inserter inserts report into inventory database



1. Endpoint scans and sends report to plugin based on acknowledgement received from the plugin
2. Plugin accepts/rejects report based on its query status from Hazelcast in-memory cache. It compares last scan status fetched from Hazelcast in-memory cache for an endpoint with the version of report received from the Endpoint.
3. If plugin accepts report it queues and then forwards it to the inserter
4. Inserter inserts the report into the inventory database
5. Inserter Plugin updates Hazelcast in-memory cache with latest status



Inventory Redesign – Using Hazelcast: Advantages

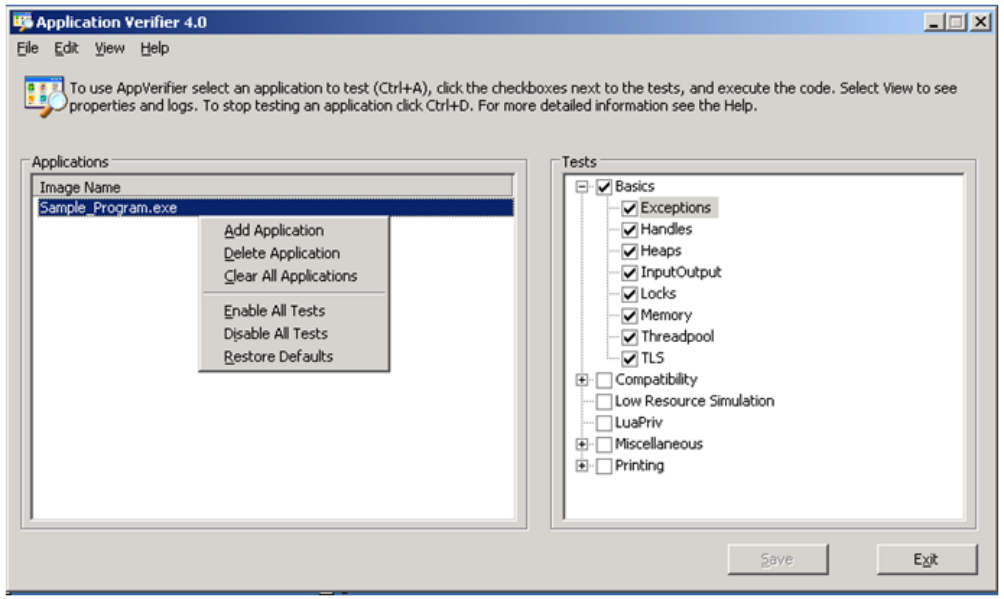
- In a test environment with 1 lakh mock endpoint reports, Hazelcast performance was observed to be much faster. With 1 lakh in-memory objects, the insert/update/select commands on these happens in few milliseconds. No performance impact due to the addition of Hazelcast workflow at Inventory Plugin
- For this testing, we have used machines from Pune lab, Houston machine and also our laptop. With this distributed environment, Hazelcast works just fine. So, there is no limitation with support for globally distributed cluster nodes (it works as long as the nodes have connectivity)
- On the protocol used behind, it is configurable… it can work on multicast as well as TCP.

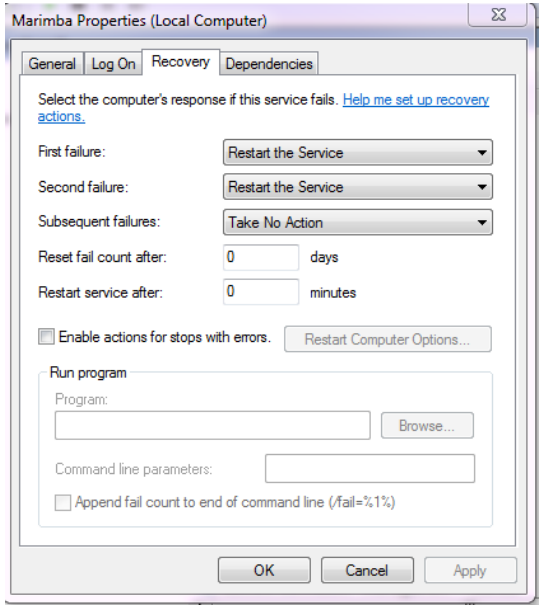
Tuner Stability

Design based on multiple inputs/analyses

Points

- We engaged with Microsoft support/tools to analyze agent issues
  - Windows Error Reporting crash dumps analysis
  - Application Verifier: https://msdn.microsoft.com/en-us/library/windows/desktop/dd371695(v=vs.85).aspx
  - Blogs/service pack bulletins
- **Main takeaway**:
  - Some issues can be resolved by OS updates/application checks
  - Others require runtime automated handling based on the encountered situation
- Drew inspiration from the Error Recovery settings in Windows Services

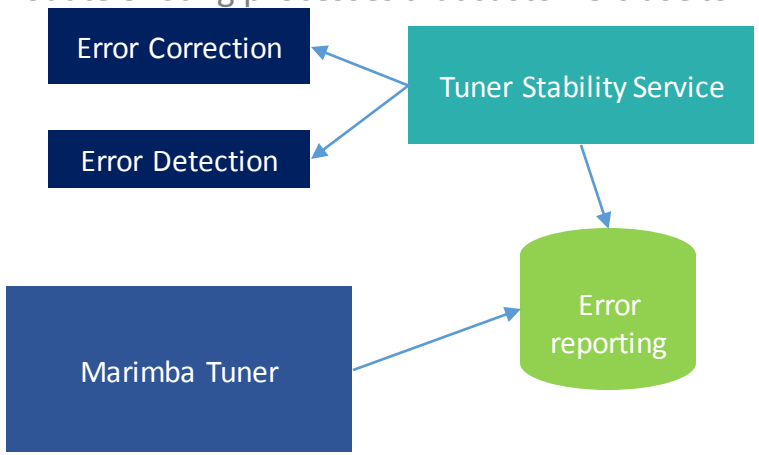




Overall Design: Introducing a configurable error detection/correction service

Points

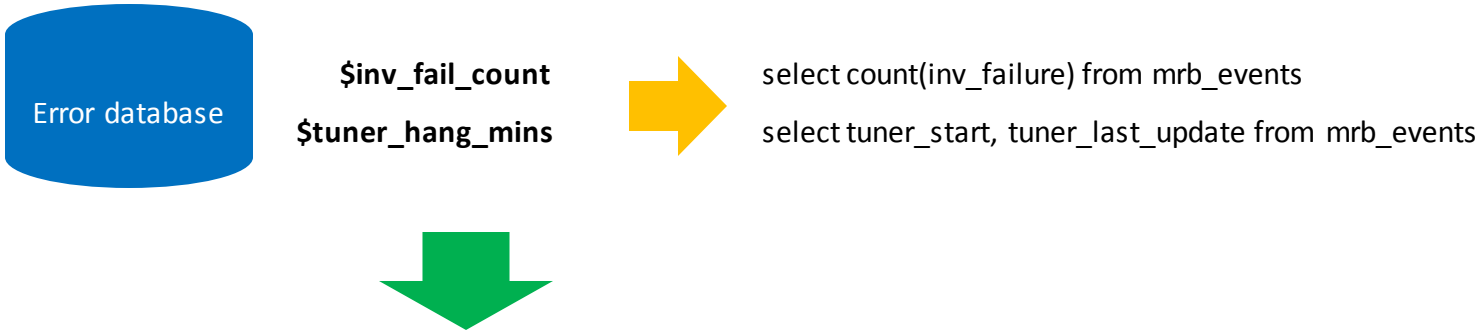
- Design: a separate service that co-exists with the Marimba agent
- Works off principle of "error detection" and "error correction"
- Hosts a minimal low-memory, low-footprint database *inside each agent* that stores key events in Marimba operations in an audit trail
  - Tables that contain records pertaining to events related to Marimba operations
  - Becomes the basis for quickly identifying events related to "health" of an agent
  - Is subsequently used in the "detection" phase
- Service periodically queries the database to look for issues
  - Queries are "constructed" at runtime off of error detection scripts
  - The query results determine what actions must be taken by the error correction scripts
- Both error detection and error correction modules are configurable
  - Work off of macros (substituted at runtime)
  - Follow a conditional execution syntax (to be published by Marimba)
  - Syntax and sample scripts will be made available for customers
- Advantage: Easily accommodate existing processes that customers use to manually remediate agents



Error Detection: Scripting language for introducing error “detection” scripts

Points

- You can configure "error detection" scripts:
  - A combination of macros/logic



- This is delivered as an "error detection" script
- can incorporate multiple conditions:
- ($scan_fail >= 2 | $tuner_hang_mins >= 120 | $tuner_crash_count >= 2) ? <problem> : <no-problem>

Error Correction: Triggering/history of remediation "logic"/scripts

Points

- configuring "error correction" scripts:
- macros for handling the "actions" to perform
- Roll those actions into the execution scripts for Tuner Stability

sc.exe start marimba	Msiexec -u <x> Rm –rf * Msiexec –I <x>	Tuner_ns <invsvc> -fullScan
$tuner_restart	$wipe_and_install	$force_inv_scan

($scan_fail >= 2 | $tuner_hang_mins >= 120 | $tuner_crash_count >= 2) ?
$wipe_and_install : $force_inv_scan

Main Features and Availability : Easy to configure, extend and install

Points

- Key take-aways
  - Use our scripts/macros, or add your own scripts/macros
  - Scripts delivered via typical content deployment procedures (i.e. policy updates)
    - Easy to understand scripting language
    - Can make very extensive conditional instructions using our formula analyzer
    - Full control over execution
  - Support for Windows, Linux and Mac platforms
  - Available at April end in 9.0.05.001 patch release
    - Will be backported to all active releases on which customers are running

Thank You

Send queries to Nitish.Shrivastava@harman.com