Medicaid Management Information System Replacement (MMISR) Project

C-1 Infrastructure Recommendations and Configuration Modifications

HSD Deliverable Owner: James Lilly
System Integrator (SI) Deliverable Owner: Dawn Gelle
Configuration Number v0.1
April 21, 2020

TurningPoint
Turning Point Global Solutions
Software Services Company Inc.
## Contents

1 **Introduction** .................................................................................................................................................. 3  
1.1 Overview.................................................................................................................................................. 3  
1.2 Purpose.................................................................................................................................................. 3  

2 **Hardware Recommendations** .......................................................................................................................... 4  
2.1 Introduction............................................................................................................................................. 5  
2.2 MarkLogic Hosts ........................................................................................................................................ 6  
2.2.1 CPU .................................................................................................................................................. 6  
2.2.2 Storage ............................................................................................................................................ 6  
2.2.3 Memory ........................................................................................................................................... 7  
2.2.4 Operating System ............................................................................................................................. 8  
2.3 MarkLogic Cluster Configuration ................................................................................................................ 9  
2.3.1 Monitor ........................................................................................................................................... 9  
2.3.2 Server Software ............................................................................................................................... 10  
2.3.3 Forests .......................................................................................................................................... 10  
2.3.4 Threads ......................................................................................................................................... 11  
2.3.5 Group Level Cache Settings .......................................................................................................... 12  
2.3.6 Task Server .................................................................................................................................... 13  
2.3.7 Data and Evaluator Nodes ............................................................................................................. 13  
2.3.8 High-Availability and Redundancy ................................................................................................. 13  
2.4 Network Infrastructure .............................................................................................................................. 14  

3 **Risks** ............................................................................................................................................................. 14  

4 **Requirements Traceability** .......................................................................................................................... 15  

5 **Appendices** .................................................................................................................................................. 15  
5.1 Appendix A: Record of Changes .............................................................................................................. 15  
5.2 Appendix B: List of Acronyms .................................................................................................................. 15  
5.3 Appendix C: Glossary .............................................................................................................................. 15
List of Tables

Table 1: Record of Changes ......................................................................................................................... 15

Table 2: List of Acronyms ............................................................................................................................ 15
1 Introduction

This is the S-MARKLOGIC MMSR Subcontract deliverable C-1 – Detailed Report Document Listing Recommendations for Improvements (related to infrastructure) and configuration modifications to expedite data processing and materialization of data for the MMISR System Integration (SI) project.

1.1 Overview

The New Mexico (NM) Human Services Department (HSD) has adopted the Health and Human Services (HHS) 2020 vision, a transformational, enterprise approach to the health and human services business. HHS 2020 will move service delivery from a program-centric approach to a stakeholder-centric approach. NM HSD will migrate away from program and technology silos into an integrated, flexible framework that supports service delivery and stakeholder interaction across HHS programs and organizations. HHS 2020 is technology-enabled, but includes rethinking organizational design, redesigning and streamlining business processes, and reducing barriers between organizations within the HHS enterprise.

Please see “Section 1: Introduction” in the Project Management Plan for a detailed MMISR Project Overview.

1.2 Purpose

The purpose of this deliverable is to provide recommendations in hardware/software configuration changes to expedite processing in the SMR.
2 Hardware Recommendations

This section contains several issues requiring action to be addressed ensuring performance and continuity of service. In addition, a number of non-pressing recommendations are made, which should be acted upon as well. The table below provides an overview of key findings. Recommendations are discussed in the next sections.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pressing recommendations for change</td>
<td></td>
</tr>
<tr>
<td>Non-pressing recommendations for change</td>
<td></td>
</tr>
<tr>
<td>No recommendations for change but some best practices are provided</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Status</th>
<th>Recommendation(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Pressing</td>
<td>Provision additional cores to achieve desired performance</td>
</tr>
<tr>
<td>Forests</td>
<td>Pressing</td>
<td>Configure two forests per host to improve ingest performance</td>
</tr>
<tr>
<td>Group Level Cache Settings</td>
<td>Pressing</td>
<td>Reduce Expanded Tree Cache size to free memory for forest stands and app servers</td>
</tr>
<tr>
<td>High-Availability and Redundancy</td>
<td>Pressing</td>
<td>Configure failover for Security and other core DBs on primary node 1 for resiliency</td>
</tr>
<tr>
<td>Storage</td>
<td>Non-pressing</td>
<td>Evenly configure storage for each node</td>
</tr>
<tr>
<td>Memory</td>
<td>Non-pressing</td>
<td>Configure Huge Pages for better memory performance</td>
</tr>
<tr>
<td>Operating System</td>
<td>Non-pressing</td>
<td>Add log scanning and evaluate I/O scheduler change</td>
</tr>
<tr>
<td>Server Software</td>
<td>Non-pressing</td>
<td>Stay current with software releases</td>
</tr>
<tr>
<td>Threads</td>
<td>Non-pressing</td>
<td>Configure threads to support response time and throughput</td>
</tr>
<tr>
<td>Task Server</td>
<td>Non-pressing</td>
<td>Restore default task server queue size</td>
</tr>
<tr>
<td>Data and Evaluator Nodes</td>
<td>Non-pressing</td>
<td>Some changes may be required to tune and optimize your cluster as cluster is scaled</td>
</tr>
<tr>
<td>Monitor</td>
<td>No recommendations</td>
<td>Best practices include monitoring performance meters</td>
</tr>
</tbody>
</table>
2.1 Introduction

These recommendations were created after applying the Utilization Saturation Errors methodology (USE) methodology and are based on examining the current system represented through the UAT environment. A problem statement was used to identify candidate use cases and narrow scope for resource analysis. For each system resource, we selectively examined metrics for utilization, saturation, and errors.

Work began with the problem statement that described the issues to be addressed and the conditions to be improved. The statement was then used to focus the effort and validated the outcome. The efforts focused on developing a workload characterization of ingest and materialization performance.

“MarkLogic personnel will assess the ingestion and materialization performance functions for the MMISR project including the following components: ingestion for each source system and associated data type, materialized data types, code review, analysis and findings.”

We reviewed existing design documents, acquired source code, observed the system under load, and gathered statistics. These efforts were initially conducted onsite, then in developers offices, and ultimately with remote access to source systems.

The workload for ingest and materialization was evaluated to drive a simulation with smaller data sets. We created a model query from a query trace and used the model for analysis and simulation. The characteristics of the load for this query including index utilization, IOPS, and throughput.

As the analysis progressed, we moved from a high-level view of the workload to deeper details with drill-down analysis. Drill-down analysis involved reviewing layers of software and hardware to find the core of the issue using profiling to sample application metrics, and dynamic query tracing to examine the execution of the query and update functions identified in the problem statement.

The identification, monitoring, and analysis included examining:

- MarkLogic support dumps
- Linux system messages
- Health check script output
- Operating system performance monitoring
- MarkLogic meters data
- MarkLogic configuration files
- VMWare configuration RVTools output

The major findings discussed below include:

- Under provisioned CPU cores
- Insufficient number of forests
- Insufficient thread count
- Unusual cache settings
2.2 MarkLogic Hosts

2.2.1 CPU

After optimizing application code, we recommend increasing the number of CPUs available for workload execution. Additional processing throughput can be achieved by increasing the number of CPUs. Adding CPU cores to existing hosts or adding additional E-Nodes with CPUs to the cluster will increase throughput. For the compute load requirements in materialization, the hosts in the UAT environment could benefit from increased computing capacity. This can be achieved through increasing the number of hosts within the environment, increasing the number of CPU cores, or a combination of both techniques. Additionally, this increase could be achieved on a temporary basis during large-scale operations through manual or automatic scaling of E-Node resources. Load testing of ingest and materialization showed these processes are bound by CPU not disk or memory.

CPU at 80-85% utilization tends to be maximum for responsive system. In general, CPU utilization is expected to increase according to workload. Code deployments with additional complexity and data loads with additional index entries are expected to increase CPU utilization. This is a normal and expected growth pattern.

There are two steps to efficiently scale:

1. Code optimization:
   https://docs.marklogic.com/guide/performance/request_monitoring
2. Scaling and expansion—add RAM, hosts, scale out and/or up:
   https://docs.marklogic.com/guide/cluster/scalability

A useful process to follow for analyzing performance is the Utilization Saturation Errors methodology (USE) methodology: http://www.brendangregg.com/usemethod.html. For monitoring internal MarkLogic application metrics, use Performance Meters to isolate bottlenecks in disk space/storage, IOPS, CPU, or RAM.

- Monitoring History Documentation:
  https://help.marklogic.com/Knowledgebase/Article/View/284

2.2.2 Storage

We recommend being consistent with configuration of storage size on nodes within a cluster. Allocate the same amount of disk to each node.

With the D-Node bucket assignment policy, data will be distributed across all nodes equally. If a node runs out of space before other nodes and high-availability is not configured, the database will become unavailable. The most likely outcome from running out of space is that merges will begin to fail, resulting in an XDMP-MERGESPACE error in the error log. It is also possible that forests will go offline. If a forest goes offline, the database will also be offline, halting all access to the database. When this happens, an administrator will need to take manual corrective action to either free up some disk space or add more disk space.

All data loaded uses system resources. Memory and storage requirements increase as data increases. Data size impacts search speed, term list sizes, and other caches. Data which is not needed can be archived to free up resources.
The following MarkLogic articles provide useful guidance regarding disk space utilization:

- Understanding MarkLogic Minimum Disk Space Requirements: https://help.marklogic.com/Knowledgebase/Article/View/284
- Recovering from Low Disk Space: https://help.marklogic.com/Knowledgebase/Article/View/49

### 2.2.2.1 Document Size

We recommend changing the SMR to contain fewer, larger documents.

MarkLogic’s document size recommendation is for 100KB sized documents +/- 2 orders of magnitude

MarkLogic’s internal algorithms are optimized for documents around 100 KB (note: in MarkLogic, each document should be one unit of query and is typically equivalent to a row in a relational database, not a table). While documents do not have to be within this size range, there can be a trade-off. For example, documents smaller than 1KB have higher memory, storage and lock overhead for the amount of data within the document. Additionally, documents can be larger than 10MB, however, at these sizes, the time to read it a document off of disk starts to be noticeable.

### 2.2.3 Memory

MarkLogic recommends that most customers set the Expanded Tree Cache size to 1/8th (12.5%) of the physical memory on the server. The current Expanded Tree Cache is allocated 79GB out of 180GB (43%). Recommendations can be found below in the section Group Level Cache Settings.

By default, MarkLogic Server will allocate roughly one third of physical memory to the aforementioned caches, but the server will try to utilize as much memory as possible. The "Rule of Thirds" provides a conceptual explanation of how MarkLogic uses memory on a server:

- One third of physical memory for MarkLogic group-level caches (including the Expanded Tree Cache)
- One third of physical memory for in-memory content (range indexes and in-memory forest stands)
- One third of physical memory for workspace, app server overhead, and Linux filesystem buffer

Memory use, broken out by major areas, is periodically recorded to the ErrorLog. These diagnostic messages can be useful for quickly identifying memory resource consumption at a glance and aid in determining where to investigate memory-related issues. These logs allow for easy periodic monitoring of memory consumption over time, and records it in a summary fashion in the same place as other data pertaining to the operation of a running node in a cluster. Since all these figures have at their source raw Meters data, more in-depth investigation should start with the Meters history.

Refer to Memory Consumption Logging and Status: https://help.marklogic.com/Knowledgebase/Article/View/527

### 2.2.3.1 Huge Pages

Huge Pages are not configured and should be used. MarkLogic utilizes caches to increase retrieval performance of frequently-accessed objects. In particular, MarkLogic’s caches.
memory use falls into two major categories: large block and small block. Caches and in-memory stands look for large blocks of contiguous memory space, while range indexes, workspace memory, and the Linux filesystem buffer utilize smaller blocks of memory. In order to efficiently allocate the large blocks of memory for the group-level caches and in-memory stands, MarkLogic recommends the usage of Linux Huge Pages. Instead of the kernel allocating 4k pages of memory, huge pages are 2048k in size and can be quickly allocated for larger blocks of memory. At a minimum, MarkLogic recommends allocating enough huge pages to cover the group-level caches (roughly one third of physical memory). The upper end of recommended huge pages includes both the caches and in-memory stands.

Current setting → HugePages_Total: 0
Recommendation → Change Huge Pages: from: "0" to "20480"


<table>
<thead>
<tr>
<th>Parameter</th>
<th>MarkLogic Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total RAM MB (GB)</td>
<td>131,072 (128GB)</td>
</tr>
<tr>
<td>Default Huge Page Ranges MB</td>
<td>20,480 to 21,020</td>
</tr>
</tbody>
</table>

Linux Huge Pages should be set to 3/8 the size of the physical memory. Swap space should be set to the size of your physical memory minus the size of your Huge Pages (because Linux Huge Pages are not swapped), or 32GB, whichever is lower. Huge Pages allow areas of memory to be reserved for resources which are likely to be accessed frequently, such as group level caches. Configuring Huge Pages can increase performance because caches should always be resident in memory.

### 2.2.4 Operating System

Some practices and configuration changes for the operating system are recommended:

- Log scanning
- I/O scheduler

#### 2.2.4.1 Log scanning

The logs for MarkLogic Server are an important source of information regarding server activity, particularly information about error conditions. The logs are all stored in the Logs directory under the MarkLogic Server data directory (/var/opt/MarkLogic/Logs/ErrorLog.txt). A script or program can send alerts when a condition has been detected. Some customers use log monitoring services like splunk to scan logs and report on critical issues.

Two sample scripts for processing ErrorLogs are provided below:

- Removes common messages from a MarkLogic ErrorLog.txt file including stand update information (Save/Merge/Delete), replication information, and forest mounting / unmounting https://gist.github.com/ableasdale/5960481
• Bash Script for extracting counts of message types and information from ErrorLogs
  https://gist.github.com/ableasdale/5960481

Recent releases of MarkLogic provide much more about infrastructure performance issues:
  “slow” messages for write, read, fsync, and background I/O
  “Memory” messages and warnings
  “Hung” messages indicates the underlying VM is starved of CPU cycles

2.2.4.2 Operating System changes

2.2.4.2.1 I/O Scheduler
On VMWare hosted servers, the “noop” scheduler is recommended. In any situation where a hardware controller can manage scheduling or where disk seek times are not important (such as on SSDs), any extra work performed by the scheduler at Linux kernel level is wasted.

Recommendation → Change the I/O Scheduler: from "deadline" to "noop"

Per https://help.marklogic.com/Knowledgebase/Article/View/8/18/notes-on-io-schedulers
Device Scheduler: /sys/block/sdb
  cat /sys/block/sdb/queue/scheduler
  noop [deadline] cfq

2.3 MarkLogic Cluster Configuration

2.3.1 Monitor
MarkLogic Monitor History provides insight into application performance. Monitor performance as jobs are run to ensure the cluster is loaded as expected. Use MarkLogic Monitor History to get clear visual representations of key data and metrics representing cluster health and performance. Monitor disk space available to ensure MarkLogic never runs out of disk. Monitor CPU %idle to ensure jobs are fully using available resources. Monitor memory swap out rate and ensure the server never swaps to disk.

https://docs.marklogic.com/guide/monitoring/history

Basic MarkLogic Server Monitoring Guidelines:
https://help.marklogic.com/Knowledgebase/Article/View/31

Various Linux command can be used to provide insight into overall server performance:
  dmesg | tail (system log messages)
2.3.2 Server Software

Stay current with MarkLogic software minor releases and major release upgrades. MarkLogic Server typically gets faster and more stable with each release. Upgrade to the latest maintenance release of your current version before upgrading to the next feature version (for example: when upgrading from v9.0-8 to v10.0-1, first upgrade to the latest 9.0-x release). This may be required for older maintenance releases.

[Note] Upgrading across 2 feature releases in a single step may not be supported - please refer to the Installation and Upgrade Guide for supported upgrade paths.

When planning an upgrade across feature releases of MarkLogic Server, plan for a full test cycle of your application. In addition to testing your application on the new version, test the upgrade process. Always read the release notes before performing an upgrade. Pay particular attention to the "Known Incompatibilities" section.

2.3.3 Forests

We recommend changing from 1 to 2 forests per node. An additional forest per node will support concurrent ingest updates for better ingest performance.

There is not a hard limit on forest maximum data size. More forests support higher write concurrency. Maximum forest size is driven by the update concurrency requirements. Forests have only one journal and only one in-memory stand. More forests means more concurrent journal writes and more fragments inserted concurrently into in-memory stands.

It is the number of fragments in a stand that is important, not the number of fragments in a forest. At 256 million fragments in a stand, data may be at risk of becoming corrupted due to integer overflow. If a stand grows past 109 million fragments the database will start to log warnings to ErrorLog.txt. There is a hard limit of 64 stands per forest. Generally, MarkLogic shouldn’t ever hit this 64 stand limit unless there is an I/O limitation or merge blackout. With a merge max size of 32GB it is conceivable for a forest to reach more than one TB but update performance could also degrade significantly.
The rule-of-thumb maximum size for a forest is 512GB. Each forest should ideally have two vCPUs of processing power available on its host, with 8GB memory per vCPU. The rule-of-thumb is useful for general read/write workloads and serve as a guideline.

https://help.marklogic.com/knowledgebase/article/View/291/0/emergency-forest-has-n-fragments-message

Increasing forests per host will increase memory usage. The current forest stand “in-memory size” on a QAT node with 13 stands for RDL and 175 million documents is 33GB. The MarkLogic in-memory caches (list-size, tree-size, range-index-size, and triple-index-size) consume system memory to increase performance. The current in-memory cache sizes are:

<table>
<thead>
<tr>
<th>In memory list size</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>In memory tree size</td>
<td>64</td>
</tr>
<tr>
<td>In memory range index size</td>
<td>64</td>
</tr>
<tr>
<td>In memory reverse index size</td>
<td>8</td>
</tr>
<tr>
<td>In memory triple index size</td>
<td>21</td>
</tr>
<tr>
<td>In memory geospatial region index size</td>
<td>32</td>
</tr>
</tbody>
</table>

For more information on adding forests to a cluster refer to:

https://docs.marklogic.com/guide/admin/database-rebalancing#id_23094

### 2.3.4 Threads

Given the current workload of ingest and harmonization, without concurrent ad hoc user queries, we recommend using client threads to fully load server threads and CPU utilization. Currently 256 server threads are configured. Configure the client processes to load CPU, memory, and disk to fully utilize cluster resources. Ideally, a thread count should be configured which will keep most of the job threads busy and keep MarkLogic busy without overwhelming the cluster. MarkLogic Content Pump (mlcp) manages a workload by calculating the number of threads assigned to a batch of records called a split. The maximum number of threads and size of the splits can be configured by the user. Then the source data is divided into splits and processed by groups of threads. The number of threads in a split group is determined by the formula: thread_count divided by (<number of source records> divided max_split_size) equals thread_count_per_split. If MarkLogic Server is not I/O bound, raising the thread count, and possibly threads per split, can improve throughput when the number of splits is small but each split is very large. To tune these settings, first run a performance test with the defaults and measure the response time and throughput. Then, increase the number of threads. When the number of threads is low, the server should be able to provide the fastest response time. When the number of threads is increased, the response time for individual requests may increase, but the throughput should increase as well.
Using the Monitoring History, observe the available system resources (CPU, memory, disk bandwidth). As the number of threads is increased, usage of system resources should increase; the goal is to find the ideal number of threads before system resources are exhausted. Refer to the MarkLogic Performance - Understanding System Resources Guide for resource consumption specifics:


We observed mlcp ingesting a CSV file with size 677,421,003 and 1,127,157 rows. With a client thread count of 40 the elapsed time was 203 seconds. We then modified the thread count and increased it to 300 and observed a faster result with an elapsed time of 102 seconds.

2.3.5 Group Level Cache Settings

We recommend reducing the size of the Expanded Tree Cache to provide additional memory for query thread execution. The current Expanded Tree Cache is allocated as 79GB out of 180GB (43%). This large individual cache results in 63% overall cache size compared to recommended target 33% cache memory utilization. Reduce the Expanded Tree Cache to within 2x of the defaults. We recommend reworking processes that require more ETC than the recommendation. With the current job profile of ingest and harmonization, most documents will not take advantage of the ETC. Reference tables used frequently will already be available in the ETC and provide some cache hits for better performance. Additionally, explicit code in the SMR codebase clears the Expanded Tree Cache, which is not recommended.

The Expanded Tree Cache stores the document in an expanded tree format. Each time a D-node sends an E-node a fragment, it sends it in the same compressed format in which it was stored. The E-node then expands the fragment into a usable data structure. This cache stores that "expanded tree" structure.

Recommendation: Increase other cache settings to the defaults:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MarkLogic Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total RAM MB (GB)</td>
<td>131,072 (128GB)</td>
</tr>
<tr>
<td>List Cache MB</td>
<td>16,384 (6 partitions)</td>
</tr>
<tr>
<td>Compressed Tree Cache MB</td>
<td>8,192 (11 partitions)</td>
</tr>
<tr>
<td>Expanded Tree Cache MB</td>
<td>16,384 (6 partitions)</td>
</tr>
<tr>
<td>Triple Cache MB</td>
<td>8,192 (11 partitions)</td>
</tr>
<tr>
<td>Triple Value Cache MB</td>
<td>16,384 (22 partitions)</td>
</tr>
</tbody>
</table>

https://help.marklogic.com/Knowledgebase/Article/View/420/0/group-level-cache-settings-based-on-ram
2.3.6 Task Server

We recommend reducing the task server size to the default. Rework processes that require millions of task server queue entries. Reset the task server queue size to 100,000.

The Task Server provides the ability to queue work for asynchronous execution. A program can spawn another program to be started asynchronously from the calling query. A spawned task is placed on the task queue of the MarkLogic task server with the option to modify the priority. The programs may then be executed concurrently if the task server has the available resources.

The SMR environment task server queue size is currently set to be 100x default size, and may cause issues. Together with other memory-intensive settings such as Expanded Tree Cache size, thread count, and substantial range index use, the large task server queue will use memory. Monitoring of the task server during ingest and materialization did not indicate significant numbers of tasks being queued.

Extensive use of the Task Server in a mature production system is not generally a best practice. Queued task are not easily managed during execution. If the SMR is restarted while tasks are queued, they are lost upon restart. Note that external tools such as corb, mlcp, and code using the MarkLogic Java APIs have more precise orchestration control over threads and loads.

2.3.7 Data and Evaluator Nodes

Some workloads may benefit from spinning up additional E-Nodes for CPU bound processes. In particular, consider temporary E-Nodes for the materialization jobs. To add an E-Node bring another MarkLogic Server machine online and joining that machine to the cluster. With VMWare, adding E-Nodes is accomplished by adding hosts to a workload domain cluster. Adding E-Nodes could help with CPU and thread intensive tasks. Scaling E-Nodes horizontally provide a proportional increase in CPU and thread capacity.

- Scalability Reference Documentation: https://docs.marklogic.com/guide/cluster/scalability
- Scalability Reference Documentation: https://docs.marklogic.com/guide/cluster/scalability

As mentioned above, available disk storage on D-Nodes should be continuously monitored and evenly distributed across nodes in the cluster. D-Nodes are used to scale data resources and service data queries. Queries can only run as fast as the slowest D-Node. Rates can be found from examining D-Node round-trip-rates.

2.3.8 High-Availability and Redundancy

The Forests for the Security, Modules, Schemas and Triggers databases are on a single node within each environment. MarkLogic best practice for all multi-node environments is to share the modules, security, schemas and triggers databases across more than one node for redundancy. Local-disk failover should be utilized to mitigate host availability should there be a problem with the primary node that serves these forests. Access to the Security databases is required to administer the cluster. Administering the cluster is needed to diagnose the situation and restore functionality. Consider local-disk failover for these databases.
MarkLogic Server provides high availability for content hosted on D-Nodes with failover for forests. There are two types of failover: local-disk failover and shared-disk failover. Local-disk failover allows an administrator to specify replica forests on one or more hosts for failover. Shared-disk failover allows an administrator to specify failover hosts. The failover host takes over the role of a failed host if it becomes unavailable. Both types of failover detect when a host is down and perform the needed changes to have another host take over. For more details about failover, see High Availability of Data Nodes With Failover https://docs.marklogic.com/guide/cluster/failover#.

Using VMWare High Availability (HA) is not a configuration MarkLogic specifically tests, but works when configured correctly, in combination with MarkLogic's own failover functionality. One possible way to do this would be to have all nodes in a cluster monitored by the VMware HA software and have local-disk replication set up in such a way that the replicas are on other nodes. When a node fails, the HA software should then recreate a node and have it rejoin the cluster. The cluster will continue to work and serve requests throughout the failure, as long as there is a minimum quorum of nodes working.

Using VMWare High Availability (HA) is not a configuration MarkLogic specifically tests, but works when configured correctly, in combination with MarkLogic's own failover functionality. One possible way to do this would be to have all nodes in a cluster monitored by the VMware HA software and have local-disk replication set up in such a way that the replicas are on other nodes. When a node fails, the HA software should then recreate a node and have it rejoin the cluster. The cluster will continue to work and serve requests throughout the failure, as long as there is a minimum quorum of nodes working.

https://www.vmware.com/products/vsphere/high-availability.html

2.4 Network Infrastructure

While we did not observe network issues, there are MarkLogic best practice recommendations for network infrastructure. Use a dedicated physical NIC for MarkLogic cluster communications and a separate NIC for application communications. If multiple NICs are unavailable, use separate VLANs for cluster and application communication. Separating communications ensures optimal bandwidth is available for cluster communications while spreading the networking workload across multiple CPUs.

Use dedicated physical NICs for vMotion traffic on ESXi hosts running MarkLogic. If additional physical NICs are unavailable, move vMotion traffic to a separate VLAN. Separating vMotion traffic onto separate physical NICs, or at the very least a VLAN, reduces overall network congestion while providing optimal bandwidth for cluster communications.

https://help.marklogic.com/Knowledgebase/List/Index/18/performance-tuning (Network-General section)

3 Risks

The content within this document is based upon information available at the time requested and that has been provided to MarkLogic staff. If this information changes or information has not been provided to MarkLogic staff, the content of the deliverable may compromised.
4 Requirements Traceability

This deliverable meets the following requirements:

- Subcontractor SOW: Page 1, 3.iii

5 Appendices

5.1 Appendix A: Record of Changes

The following table records the changes for this document:

<table>
<thead>
<tr>
<th>Version Number</th>
<th>Date</th>
<th>Author/Owner</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>04/14/2020</td>
<td>Bob Starbird</td>
<td>Initial draft</td>
</tr>
<tr>
<td>V1.0</td>
<td>04/21/2020</td>
<td>Bob Starbird</td>
<td>Final</td>
</tr>
</tbody>
</table>

5.2 Appendix B: List of Acronyms

For a comprehensive, project-wide list of acronyms, consult the Master Acronyms list on the SI Contractor team SharePoint site at Shared Resources on SharePoint.

The following is a list of all acronyms used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-Node</td>
<td>Data Node</td>
</tr>
<tr>
<td>E-Node</td>
<td>Evaluator Node</td>
</tr>
<tr>
<td>IOPS</td>
<td>Input/Output Operations Per Second</td>
</tr>
<tr>
<td>SMR</td>
<td>Systems Migration Repository</td>
</tr>
</tbody>
</table>

5.3 Appendix C: Glossary

A glossary of project-specific terminology is maintained on the SI Contractor team SharePoint site at Shared Resources on SharePoint.