HARDWARE REFERENCE ARCHITECTURE: DIRECT ATTACHED STORAGE

ERIN MILLER • MANAGER, PERFORMANCE ENGINEERING • JUNE 2016

Contents

OVERVIEW

MarkLogic®, the leading Enterprise NoSQL database, is designed to scale to large amounts of content. This guide is meant to provide details about our recommended hardware for Direct Attached Storage (DAS).

We'll cover commodity hardware recommendations, recommended hardware and RAID configurations for commodity hardware, and basic forest layout recommendations for DAS.

This guide applies to MarkLogic 7 and higher.

MARKLOGIC SERVER ARCHITECTURE AT A GLANCE

MarkLogic Server has many different moving parts, all of which require varying levels of system resources. For the purposes of this whitepaper, we wanted to present an architectural diagram that provides some information about the various components, as well as how these components use system resources.

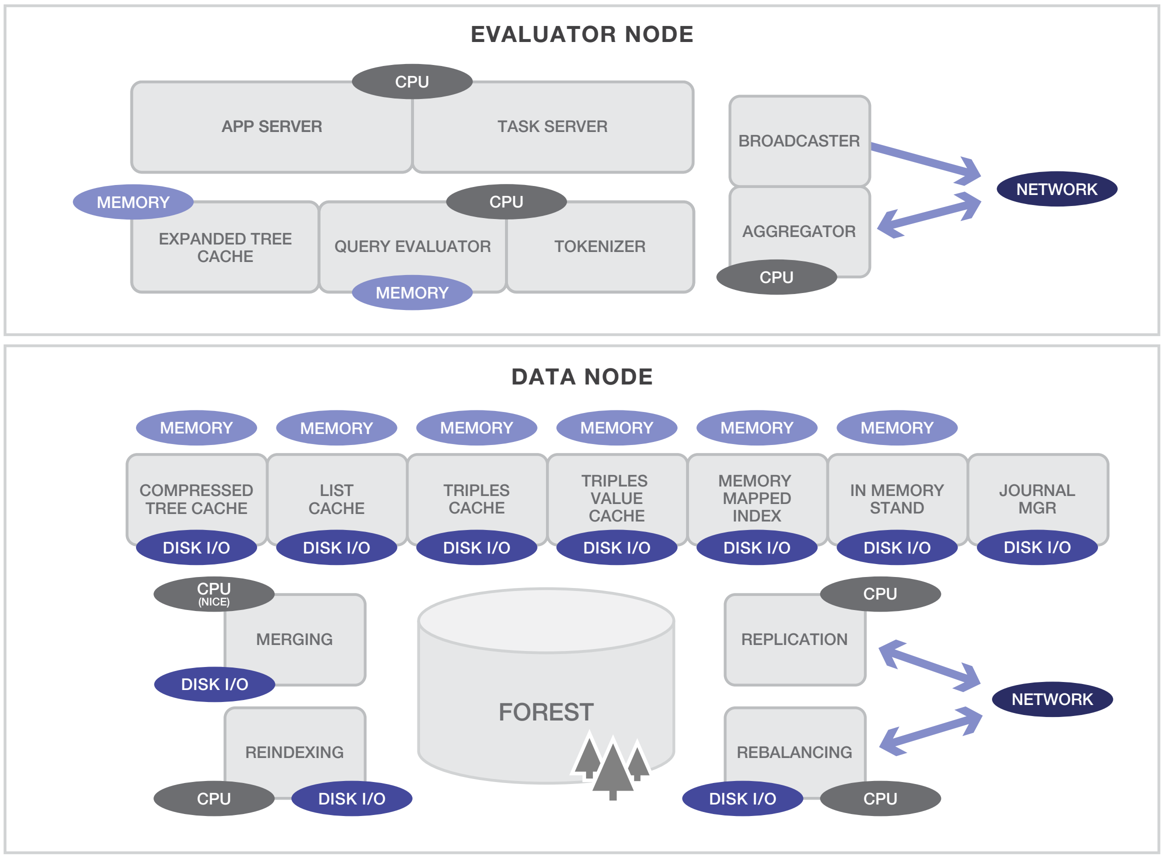

Figure 1: E- and D-nodes

TERMINOLOGY AND GENERAL CONCEPTS

A database consists of one or more *forests*. A system can have multiple databases, and generally will, but each query or update request executes against a particular database. A *forest* contains a set of documents; each forest holds a set of documents and all their indexes, mapped to a physical location on disk. Each forest holds zero or more *stands*. A stand holds a subset of the forest data and exists as a physical subdirectory under the forest directory.

Each MarkLogic Server instance can be configured to be an *Evaluator (E-node),* a *Data Manager (D-node)*, or a shared E/D node. E-nodes have application servers, parse requests, and return responses. E-nodes are also responsible for executing Javascript or XQuery server-side code.

D-nodes hold data along with its associated indexes and are responsible for maintaining transactional integrity during insert, update, and delete operations. This transactional integrity includes forest journaling, forest recovery, backup operations, and on-disk forest management. D-nodes are also responsible for providing forest optimization (merges), index maintenance, and content retrieval.

It's important to realize that MarkLogic can be configured as *either* an E- or D-node or a combined E/D node. This is a configuration option. D-nodes are defined by the presence of an attached forest on the node; E-nodes don't have attached forests, but do have application servers. So the diagram that we looked at originally is pretty much the same for a shared E/D node—it's just that all functions run on a single host. We'll talk more later about the reasons to create an architecture with separated E/D nodes, but for now, just consider that either configuration is fine.

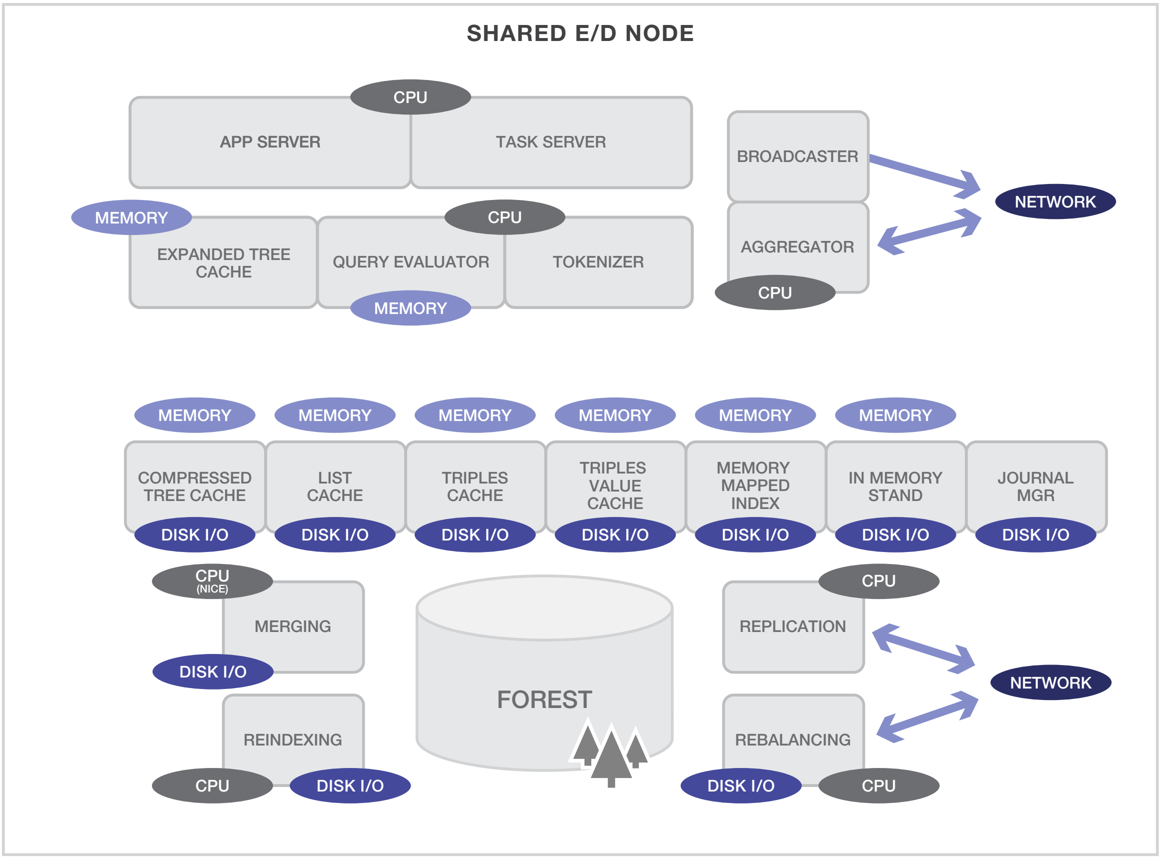

Figure 2: Shared E/D node

LOCAL DISK FAILOVER

The Direct Attached Storage scenario assumes local-disk failover for High Availability support.

Local-disk failover creates one or more replica forests for each failover forest. The replicas contain the exact same data as the primary forest, and are kept up to date transactionally as updates to the forest occur. Each replica forest should be on a different host from the primary forest so that, in the event of the host for the primary forest going down, another host with a copy of the primary forest's data can take over.

Each forest has its own host, and each host has disk space allocated for the forest. The primary forest is the forest that is attached to the database. Any replica forests configured for the primary forest have their own local disk. As updates happen to the database, the primary forest is updated as well as each configured replica forest.

In the event that Host 1 goes down or an I/O error occurs on the F1 forest, the MarkLogic Server cluster will wait until the specified configured timeout and then it will automatically remove the F1 forest from the cluster. Because all of the forests in a database must be available for queries to run, the database is unavailable at this time. At this point, the system fails over the forest to the first available replica forest, and the Replica1 forest is automatically attached to the database. The database once again becomes available. If the failed forest comes back online, it will resume as a replica.

PART I: HARDWARE AND FOREST RECOMMENDATIONS

OPTIMAL NUMBER OF FORESTS

For the reference hardware described in this whitepaper, we'll assume the optimal number of forests to be **six primary and six replica** ("6P/6R") forests. What will change is the size of the forest—so more forests means more hosts, and larger forests means more storage. It's very important to understand that the 6/6 recommendation is based entirely upon the reference hardware we suggest. If the hardware specifications change, the number of forests will also definitely change. We by no means are suggesting that 6/6 is *always* optimal, just that it is definitely optimal on the hardware we describe in section *Definition of Commodity Hardware* below. The 6/6 was based upon performance testing that balances both ingest performance and query performance; 6/6 was found to be optimal given this hardware profile.

DEFINITION OF COMMODITY HARDWARE

What should the hardware look like to support the 6P/6R configuration?

- 2U 255 SF Chassis
- 2 Sockets, 8 Cores per socket, 2.8GHz processor
- 128 – 256 GB RAM
- 22 10K 900 GB Disk Drives
- 2 2GB RAID cards
- 10 Gig E Network

What does this mean in terms of the resources that MarkLogic Server uses?

CONFIGURATION	RESOURCE DELIVERY
2 Sockets, 8 Cores per socket, 2.8GHz processor	32 physical threads, 32 virtual threads (with hyperthreading) @ 2GHz
128 – 256 GB RAM	4 GB/8 GB per thread
22 10K 900 GB Disk Drives* with 2 2GB RAID cards	300GB/Forest + Temp, Binaries, Logs (accounts for both RAID level and I/O throughput requirements), 1 GB/second I/O to disks
10 Gig E Network	1 GB/second I/O to Network

Again, it's important to remember that this hardware definition supports the 6P/6R configuration.

***Because the forest sizes change depending upon use case, the size, speed, and number of disks also depends upon the use case. The disk drives should also be configured into a RAID and the choice of RAID levels will change the number of disks.**

We'll talk about recommended disks and RAID levels in the next section.

USE CASES

Much like any database system, there are many different use cases for MarkLogic Server. This whitepaper will focus on two ends of a spectrum of use cases: high performance, and high capacity. As we'll see, these are not opposites, but rather endpoints on a continuum. The details of where your application falls on this continuum will dictate the best hardware configuration for your environment.

HIGH PERFORMANCE

In this configuration, performance is the driving factor. High performance is subjective, but very much dependent upon customer SLAs. A typical high performance use case might be a search and retrieval application with operational database support for updates and inserts.

High performance environments share some of these characteristics:

- High number of concurrent users and queries
- Sub-second response times
- Multiple facets (range queries)
- Many positions queries (i.e., NEAR queries)

To get high performance from a large-scale system (where documents are measured in the hundreds of millions), you'll need to limit forest sizes both by numbers of documents and by physical size on disk and size in memory. Keeping forests smaller enables sub-second response times for facets, as well as increased parallelism for processing complex queries. Smaller forests mean that you'll also have more forests—think of this as horizontal scaling for forests.

RECOMMENDATIONS FOR HIGH PERFORMANCE	
Forest size	100 GB indexed content maximum
Documents per Forest	8 million docs/forest
Number and Size of Disks	20 2.5" 15K 600 Gb drives
RAID configuration	RAID 10 (striping plus mirror, faster writes)

HIGH CAPACITY

On the other end of the spectrum, capacity is more important than performance. With this configuration, many terabytes of data need to be stored and queried, but users are more tolerant of slower performance. An example might be an archival solution, where a small number of users are running complex, data-rich reports.

High capacity environments share some of these characteristics:

- Lower number of concurrent users and queries
- Batch report processing that may run offline, or over longer periods of time
- Tolerance for longer running reports
- Data volume is the driving factor

In this scenario, the forests will be much larger than in the high performance scenario. Tiered storage may also be a good choice here.

MarkLogic Server allows you to manage your data at different *tiers* of storage and computation environments, with the top-most tier providing the fastest access to your most critical data and the lowest tier providing the slowest access to your least critical data. Infrastructures such as Hadoop and public clouds make it economically feasible to scale storage to accommodate massive amounts of data in the lower tiers. Segregating data among different storage tiers allows you to optimize trade-offs among cost, performance, availability, and flexibility.

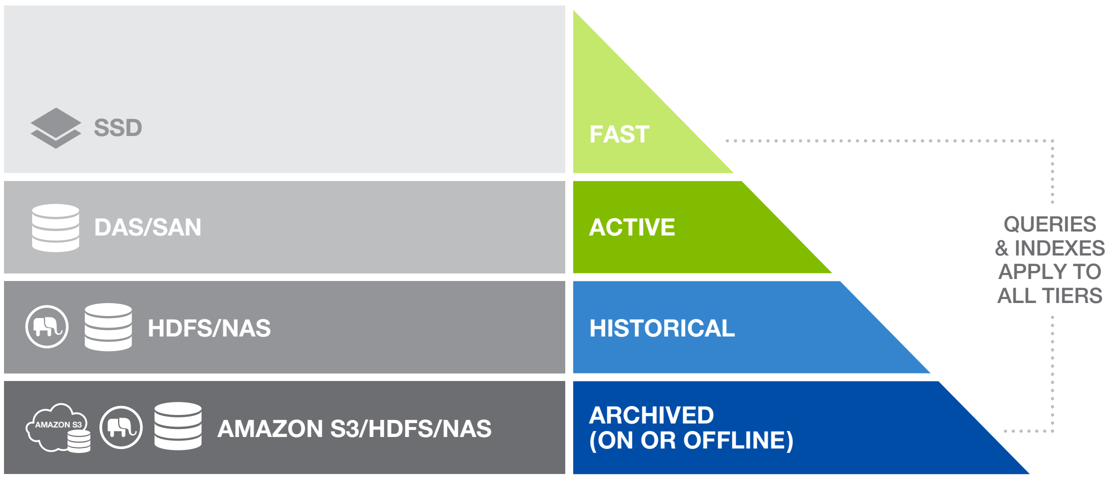

Figure 3: Tiered Storage

In the example above, the data has been organized into four categories: fast, active, historical, and archived. Depending upon the importance of the data (as defined by application requirements), the data is partitioned into one of these categories. Each category has its own hardware, corresponding to how quickly queries need to return. "Importance" in this context depends upon application requirements; for example, an application that mines social media will consider content freshness to be the most important, whereas a different application might consider most frequently accessed content to be the most important.

RECOMMENDATIONS FOR HIGH CAPACITY	
Forest size	500 GB indexed content maximum
Documents per Forest	90 million docs/forest
Number and Size of Disks	20 2.5" 10K 1200 Gb drives
RAID configuration	RAID 50 (striping plus parity, fast reads, slower writes)

IMPORTANT LIMIT FOR HIGH CAPACITY

More than 96 million fragments in a forest is considered to be an error condition.
At 256 million fragments, your forests may become corrupt due to integer overflow.

The server writes messages about these conditions to the ErrorLog:

FRAGMENT COUNT	LOG LEVEL
96 million or more	Error
120 million or more	Critical
144 million or more	Alert
160 million or more	Emergency

SPECTRUM OF REQUIREMENTS AND FOREST SIZES

Sometimes applications clearly fall on one end of this spectrum or the other. But sometimes, requirements and SLAs suggest an application that is somewhere in the middle.

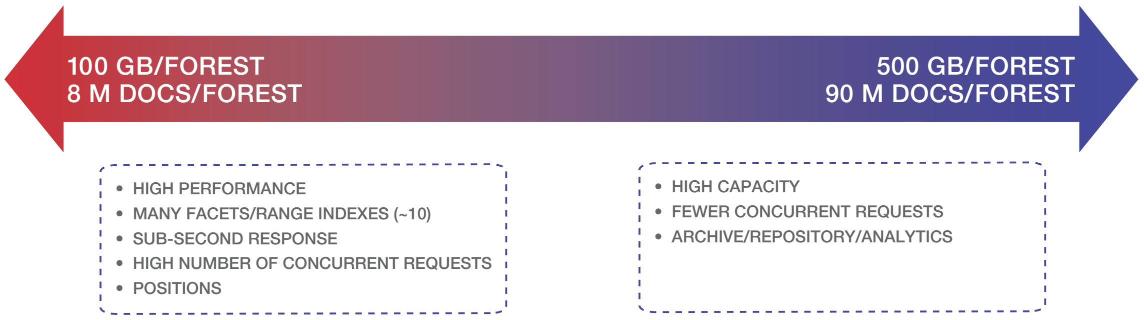
100 GB/FOREST
8 M DOCS/FOREST
500 GB/FOREST
90 M DOCS/FOREST
• HIGH PERFORMANCE
• MANY FACETS/RANGE INDEXES (~10)
• SUB-SECOND RESPONSE
• HIGH NUMBER OF CONCURRENT REQUESTS
• POSITIONS
• HIGH CAPACITY
• FEWER CONCURRENT REQUESTS
• ARCHIVE/REPOSITORY/ANALYTICS

Figure 4: Forest sizes for high performance and high capacity applications

The best way to determine optimal forest size is by careful test cycles and observation. Once you understand the optimal forest size for your application, it is easy to plan capacity and number of hosts, given a reference hardware.

CONTENT VERSUS INDEXED CONTENT

When we talk about "content size" in MarkLogic Server, we are usually referring to the content size *after* ingest, rather than the size of the raw data on disk. When content is ingested, MarkLogic adds information about the documents to its indexes which it uses to quickly resolve queries. We consider this indexed data to be the size of the indexed content. There are cases where the indexed content may be smaller than the raw content on disk, because the data is compressed into a binary format on disk. However, to support functionality like NEAR queries or wildcards, there will generally be some index expansion.

That expansion may vary widely. For example, a Digital Asset Management System may require indexes only for metadata—the binary digital data is stored in MarkLogic as large or external files which are not indexed. Therefore, the indexes may be 1% of the raw data on disk.

On the other hand, you may have many small documents that are 100% indexed, and you may want to do a fair bit of index enrichment, adding support for near queries and wildcards. This will result in a much larger storage and forest footprint.

EXAMPLE APPLICATIONS AND ASSOCIATED CONFIGURATIONS

Here's an example of four different types of applications with their indexed content and RAID levels. The application on the far left is high performance; on the far right is high capacity. There are also two mid-point applications with different configurations. This diagram assumes a three-node cluster.

Understanding this continuum will help you understand the general scale that you'll need to support for your application.

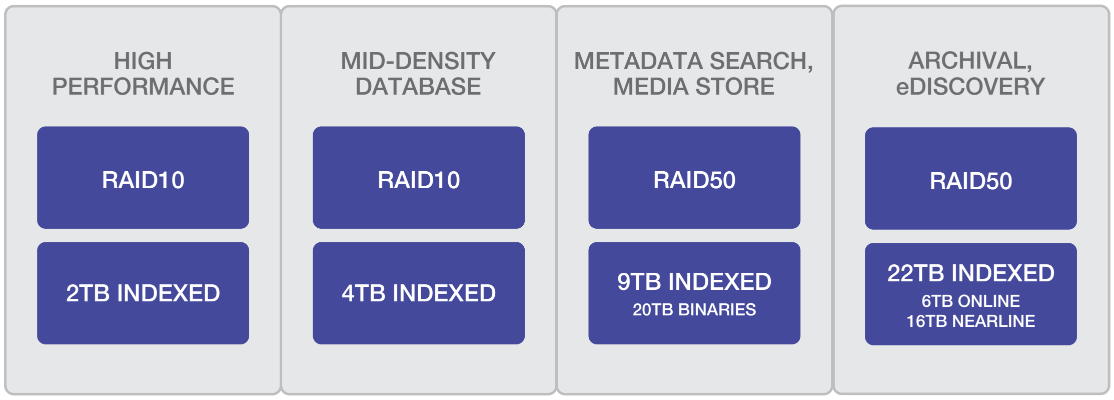

Figure 5: Storage configurations by use case

PART II: CONFIGURATION AND CLUSTER BEST PRACTICES

FOREST LAYOUT: ANCILLARY DATABASES

MarkLogic ships with a number of ancillary databases that are important to properly configure and replicate for high availability. The databases include:

Critical:

- **Security:** If the Security database is unavailable, other MarkLogic databases cannot start. This represents a production outage.
- **Modules:** If application code is stored in the Modules database and the Modules database is unavailable, the application will be unavailable resulting in a production outage.
- **Meters:** Although the Meters database being unavailable doesn't represent an outage, it is often the best way to understand production performance problems.

Important or Critical, depending upon application:

- **Schemas:** May be critical if schema validation is required in the application. Schema validation will fail if the Schemas database is unavailable.
- **Triggers:** May be critical if flexible replication or Content Processing Framework (CPF) is in or CPF is in use.

Because of how important these ancillary databases are, it makes good sense to replicate them in a production environment and have multiple copies for critical databases. *Security* and *Modules* (if application code is stored in modules rather than file system), in particular, should be replicated because those databases being unavailable would represent a production outage.

This may also be true for *Schemas*, *Triggers*, and *App-Services*, depending upon use cases. Schemas is required for any application-level schema validation. *Triggers* is used to support pre- and post-commit custom triggers as well as Flexible Replication and the Content Processing Framework (CPF). *App-Services* is needed to use the REST API.

With the exception of *Meters*, these databases are often very small and shouldn't be considered part of the 6P/6R equation—they should simply reside on one of the servers and critical ancillary databases should be replicated.

Meters is a slightly different use case and may require multiple forests at scale. *Meters* collects usage and performance data about MarkLogic including all databases and all hosts in the cluster. The amount of time that you retain data is configurable, as is the collection interval, but in a very large cluster, this may require multiple forests so it does not become a bottleneck.

An important note: if forests attached to ancillary databases fail-over, it's important to fail back **before an upgrade** so that all ancillary databases are attached to one host at upgrade-time. When upgrading, you don't want to have your security database on a separate host in the cluster when you upgrade your first host—so make sure they are all together before you upgrade.

Here's an example of what a cluster layout might look like for ancillary databases:

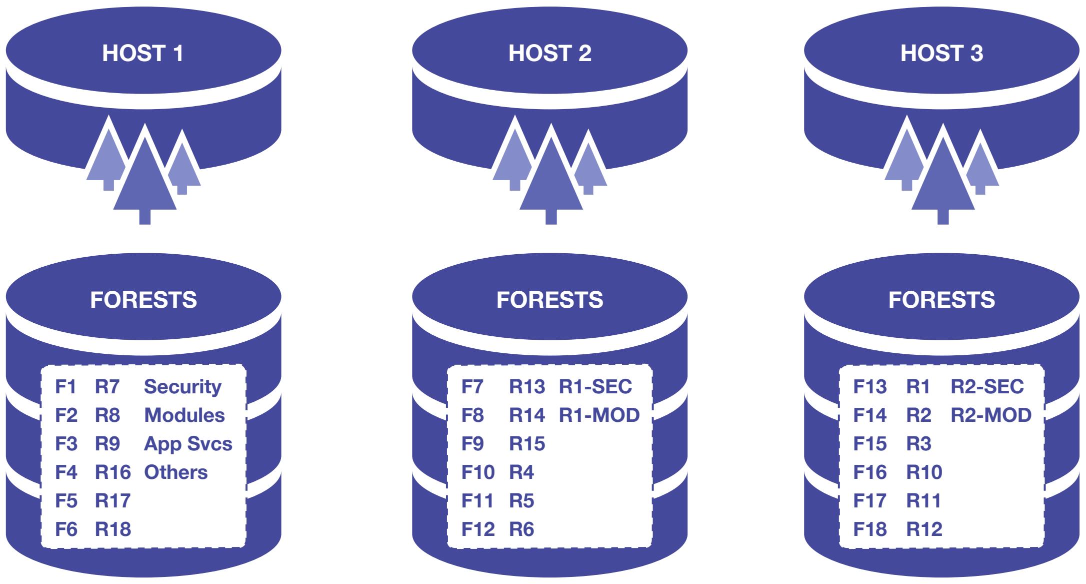

Figure 6: Forest layout

FOREST LAYOUT: DESIGN PATTERNS FOR HIGH AVAILABILITY

Assuming 6 primary and 6 replica forests per host, it's important to distribute forests equally across hosts. Specifically, you *don't* want to replicate all forests from host 1 to host 2. If host 1 then goes down, host 2 will be supporting 12 primary forests, since the six replicas will have changed roles to primary.

Instead, you want to evenly distribute the forests so that, in case of a host going down, you'd have just half of the replica forests on a host assuming the primary role, as in Figure 6 above.

PART III: WHEN AND HOW TO SCALE OUT

Starting with MarkLogic 7.0-1, it is easy to scale out and add a node or multiple nodes, and remove a node or nodes when you no longer need them.

WHEN TO SCALE OUT

If performance has degraded to the point where SLAs are not being met, it is past time to scale out. It's better to monitor MarkLogic and understand when important thresholds are about to be reached and proactively scale out to keep you in compliance with your SLAs. Monitor CPU, memory, storage space, disk I/O throughput, and network throughput to understand when you are approaching limits.

DISK I/O BOTTLENECK

Here's an example of a disk I/O throughput bottleneck:

```
avg-cpu:  %user   %nice   %system  %iowait  %steal    %idle
          19.47    7.61      2.91    26.05    0.00    43.97

Device:         rrqm/s    wrqm/s      r/s      w/s     rsec/s     wsec/s  avgrq-sz  avgqu-sz   await   svctm   %util
sdb               0.00  29325.00   376.67   713.67   75786.67  235968.00    285.93     49.14   46.68    0.92  100.00
sda               0.00     44.00     0.00     1.67       0.00     365.33    219.20      0.01    4.40    4.40    0.73
sdc               0.00      0.00     0.00     0.00       0.00       0.00      0.00      0.00    0.00    0.00    0.00
sdd               0.00      0.00     0.00     0.00       0.00       0.00      0.00      0.00    0.00    0.00    0.00
```

Figure 7: iostat output

Note the combination of average queue size, await time, service time and utilization. The SSD device is reaching saturation and more disk throughput will likely be needed soon.

You can also use the Monitoring Dashboard to understand disk I/O patterns and behavior. The Monitoring Dashboard can be viewed in a browser at *host:8002/history* for historical data or *host:8002/dashboard* for current data. It is also possible to set alerts if resources are reaching specified thresholds. More information can be found in the Monitoring MarkLogic Server Guide (http://docs.marklogic.com/guide/monitoring).

REMEDIATION FOR DISK I/O BOTTLENECK

There are a few things you can do to alleviate a disk I/O bottleneck:

1. You should first verify that there's no underlying problem with the storage infrastructure. This may require a deep dive into RAID controllers and configuration.

2. **Add RAM.** This may help if you are not utilizing MarkLogic caches well. If your list cache hit/miss ratio is low (below 90%) then adding RAM to your hosts will allow you to increase cache usage and decrease disk burden at query-time. But, if your bottleneck is due to ingest load, this won't help you much.

3. **Balance forests.** You may have a situation where one forest is much larger than the others. This means that any query running against the larger forest will be slower. This may cause performance issues—and you may be bottlenecked on just the device hosting that forest. By rebalancing content across your cluster, you may be able to resolve the bottleneck.

4. **Add hosts.** By adding hosts, you'll add more disks and more disk throughput capacity.

CPU BOTTLENECK

Using the Monitoring History, it is easy to observe a CPU bottleneck. In this example, CPU is maxed out by query load:

Make sure to combine `system%` and `user%` here. Also note that any merges will display as `nice%` and need to be added to the total as well.

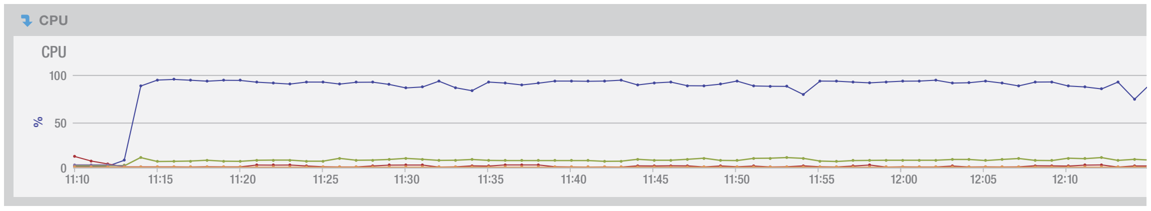

Figure 8: CPU bottleneck

REMEDIATION FOR CPU BOTTLENECK

1. **Code and index config review.** There may be inefficiencies in the code that, if fixed, could alleviate the CPU bottleneck by significantly reducing the workload. For example, rather than finding a document by doing a `cts:uris-match` with wildcards, which can be inefficient for a large result set, try using a range index, or a collection or directory query. There are many such opportunities for optimization; reducing the overall workload will reduce the CPU utilization.

2. **Add hosts.** Adding more cores will spread the load more widely across the cluster.

MEMORY BOTTLENECK

One of the most obvious ways to observe an approaching memory bottleneck is to use the Monitoring Dashboard to observe memory utilization.

If you're approaching the available memory for your host or cluster, you're in trouble. When memory is exhausted, MarkLogic will begin to swap and at that point, you may well need to restart the server in order to recover. On Linux, it's also likely that the Out Of Memory Killer will restart the server for you by killing the process.

REMEDIATION FOR MEMORY BOTTLENECK

1. **Code and index/config review.** In particular, if using SPARQL and Semantics features, make sure that you've written the most efficient query possible. Large joins require significant amounts of memory. Use the Monitoring Dashboard to observe memory utilization during load tests.

2. **Add hosts.** The memory bottleneck may be occurring because the forests are too big, or because there is too much data per host. Adding additional hosts will reduce the memory required by each host. This is especially true if you have many range indexes, which are memory mapped.

NETWORK BOTTLENECK

In addition to using tools like SAR to observe Network capacity and performance, a simple way to determine if the network is saturated is to notice a ceiling on data ingest or queries. If you're reaching saturation but memory, CPU, and disk I/O are all still available, you likely have a network bottleneck.

REMEDIATION FOR NETWORK BOTTLENECK

1. 10 Gig Ethernet generally provides enough I/O throughput (1 GB/second) to support MarkLogic. If you're running on a 1 Gig Ethernet, consider upgrading.

2. If you're already on a 10 Gig Ethernet, make sure that you're actually configured properly. In particular, if you observe that network utilization is peaking at about 1.2 Gb, then you're likely using a 1 Gig Ethernet.

STORAGE SIZE

It's important to monitor for available storage and to make sure that you're not approaching a disk space limit. If there's not enough disk space available, MarkLogic won't be able to merge and eventually this can cause an outage, so it is critically important to monitor available disk space.

MarkLogic requires two times the merge-max-size per forest for automatic merges. Assuming the merge-max-size is the default, 32GB, that means that we need 64GB per forest. At any one time, there can be any number of merges occurring, but the total size of the merges per forest is never more than two times merge-max-size.

Therefore, per forest, assuming the default merge-max-size, you'd need 1x the total indexed content plus 64 GB per forest for storage. The total storage can be calculated as:

Average indexed doc size x total number of documents + 64 GB x total number of forests + 2 GB for journals x total number of forests.

HOW TO SCALE OUT

MarkLogic 7.0-1 and above provides enterprise features which make scale-out easier. Here are the basic steps to follow:

1. **Add hosts to the cluster**, in groups of three if possible. If you add hosts in groups of three, you can easily follow the distribution pattern for high availability.

2. **Redistribute data to new hosts**. Use the rebalancer to automatically redistribute data across the cluster.

3. **Optionally, migrate forests if needed**. If you're not adding in groups of three, you can migrate forests to local disk on new hosts. Make sure that you follow the distribution pattern—don't just add two new hosts and then put all the masters on host 1 and all the replicas on host 2. If host 1 goes down, host 2 will be supporting 12 primary forests.

CONCLUSION

This paper provided a reference hardware architecture for Direct Attached Storage and MarkLogic Server. The commodity hardware description we propose is a good fit for MarkLogic Server for DAS. In addition, we provided guidance around MarkLogic forest distribution and configuration for high availability. Finally, we provided guidelines for how to monitor and alleviate system bottlenecks.

ADDITIONAL RESOURCES

- MarkLogic Administrator's Guide
- Monitoring MarkLogic Guide
- Scalability, Availability and Fallover Guide

All representations of system configuration, storage capacity, query latency or other performance data in the sizing communications are estimates and averages based on certain assumptions and conditions. No representation is made that these throughputs, capacities, response times or other performance data will be accurate or achieved in any given deployment of MarkLogic® software: Customer results will vary depending a variety of factors. Any configuration recommended by the sizing information communication should be tested and verified by Customer subsequent to tuning the final production code and queries.

MARKLOGIC CORPORATION
999 Skyway Road, Suite 200 San Carlos, CA 94070
+1 650 655 2300 | +1 877 992 8885 | www.marklogic.com | sales@marklogic.com