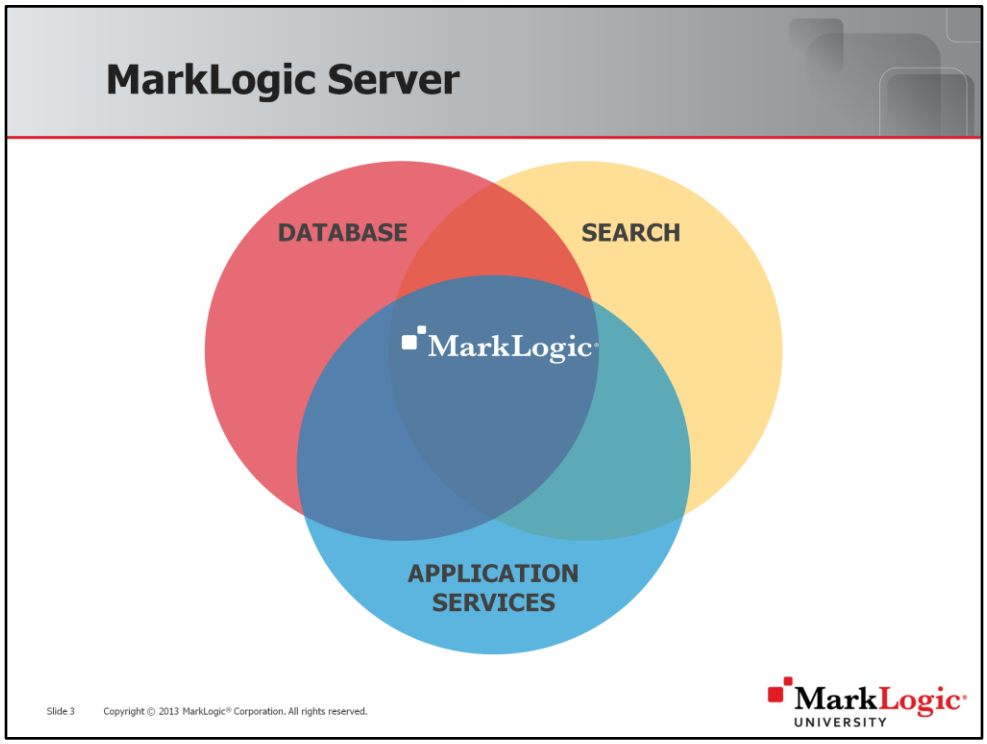
Learning Objectives

- Describe each of the following:
  - Database (Content / Modules / Schema / Triggers / Security)
  - Forest
  - Stand (In Memory / On Disk)
  - Universal Index
  - Application Server (HTTP / XDBC / ODBC / WebDAV)
  - Evaluator Node | Database Node
  - Hosts | Groups
  - List Cache | Compressed Tree Cache | Expanded Tree Cache
- Describe storage options in MarkLogic
- Compare / contrast a single host vs. cluster implementation

MarkLogic UNIVERSITY

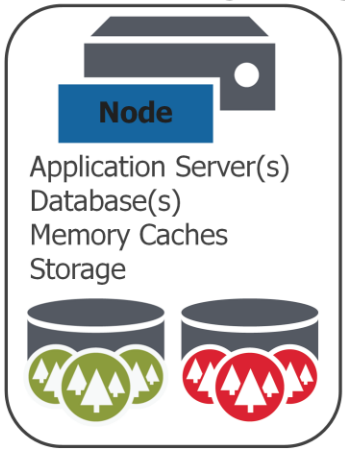


MarkLogic Server is a platform built on three core components:

- Enterprise NoSQL Database
  - MarkLogic is an ACID, transactional database built for dealing with documents such as XML, JSON, Full Text and a host of binary content types. With enterprise features such as failover and database replication, MarkLogic is trusted with organizations mission critical applications and data.
- Search
  - A database designed like a search engine provides customers with the benefit of being identify valuable information across a multitude of content types and structures, without the wasted effort of futile attempts to design the "perfect data model".
- Application Server
  - MarkLogics own application server is part of the package, enabling you to build full functioning MarkLogic applications using a variety of APIs and languages, maximizing developer productivity and reducing project duration.

Single Node Architecture

- Example: Our training environment
- Everything configured on a single machine
  - Node = Host = a machine running MarkLogic

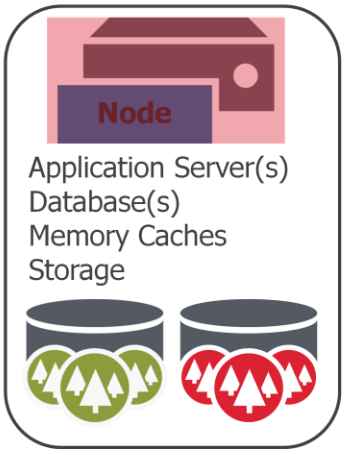


Slide 4

For smaller implementations, or for development work, MarkLogic can be deployed on a single machine. In this scenario, all components of the MarkLogic platform will exist on that single machine, including:

- Logical database configurations
- Application servers, defined on a port, to handle resolution of requests
- In memory caches for performance
- Storage for content, either local disk or shared disk.

Single Node Architecture



- Node / Host / Machine
  - E = Evaluator = App Server Instance
  - D = Data Manager = Database Instance
- MarkLogic runs as a service
- A machine can act as both E/D node
- In a cluster, machines may specialize

Slide 5

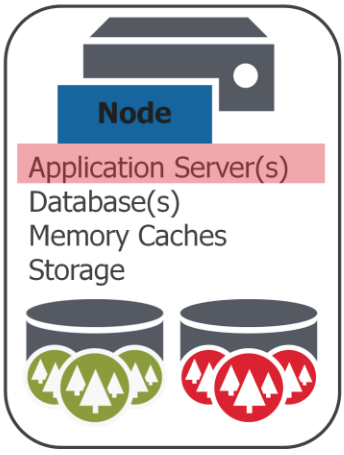
MarkLogic UNIVERSITY

The terms “node” or “host” or “machine” are synonyms. The terms refer to a computer that has MarkLogic installed on the machine. MarkLogic is installed from a single binary installer. When MarkLogic is installed, it runs as a service with a single process (technically there is a 2nd watchdog process, but this has little effect on system resources). Therefore from a software perspective, any machine that has MarkLogic installed on it has the same information installed and is capable of acting as both an E node and a D node.

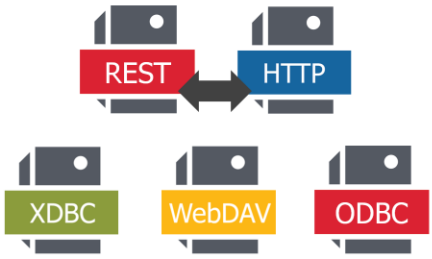
In a clustered implementation (which we will study in detail shortly), we will see that you may choose to have specialization amongst the machines in a cluster. Therefore, it is possible to logically use certain machines as app servers (E Nodes) and certain machines as databases (D Nodes). This specialization enables all of a given machines resources to be devoted to E Node specific or D Node specific tasks.

In a single node deployment, where the node performs both E Node and D Node tasks, the resources on the machine must be shared.

Single Node Architecture



- Application Server
  - Handles requests / responses
  - Defined on a port
  - Evaluates code



Slide 6
MarkLogic UNIVERSITY

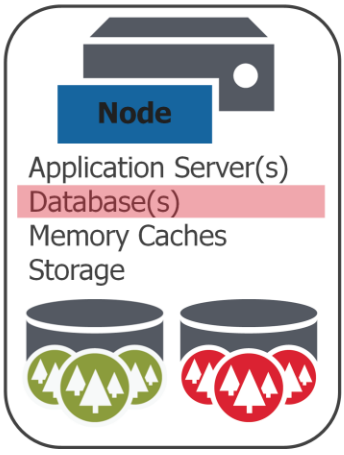
An application server is defined on a port, and when that port is accessed it handles the request, evaluates code, and delivers the response.

There are 4 types that can be configured:

- REST
  - An HTTP application server configured for REST
  - A language agnostic approach enabling you to engage RESTful Web Services through HTTP requests and using the MarkLogic REST API.
  - The foundation for additional client APIs, such as the MarkLogic Java API
- HTTP
  - Develop robust applications using the MarkLogic Application Servers native languages of XQuery, XPath, and XSLT
- XDBC
  - A pipeline to a MarkLogic database (think along the lines of ODBC). When paired with a Java or .Net client library called XCC (XML Content Connector), it enables the developer to use MarkLogic as the underlying database for Java or .Net based applications.
- WebDAV

- A folder / file system-like view of a database enabling drag and drop loading of content.

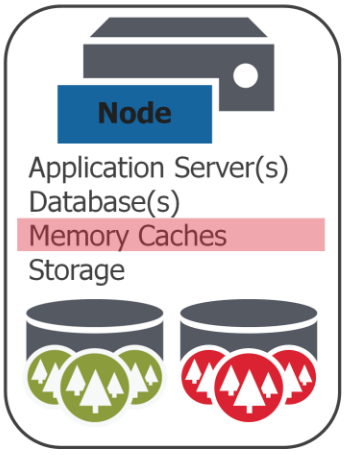
Single Node Architecture



- Database
  - Transaction controller
  - Logical configuration
    - Indexing / Reindexing settings

A database is a logical configuration, where settings such as indexes are defined. A database can be on a single machine, or spread across a cluster of machines in order to achieve massive scale. The data manager (database) manages transactions across all data associated with the database.

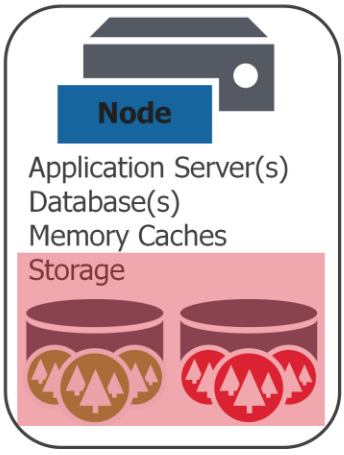
Single Node Architecture



- Configurable memory caches:
  - List Cache
    - Recently accessed indexes
  - Compressed Tree Cache
    - Recently accessed docs, compressed
  - Expanded Tree Cache
    - Recently accessed docs, uncompressed
  - Triple Data and Triple Value Caches
    - Recently accessed triples

MarkLogic utilizes memory caches to maximize performance. As requests are made and queries are executed, the recently accessed documents and indexes are cached, making future queries more performant by reducing disk I/O.

Single Node Architecture



- Forests
  - Physical storage
  - Attached to database (1DB:Many Forests)
  - Stands
    - Memory
    - Disk
    - Documents
    - Indexes
    - Compression
  - Journal

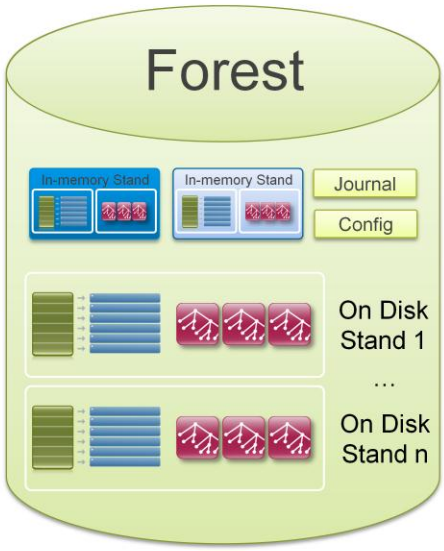
A forest is where the documents and their respective indexes physically reside. Each forest is a partition on disk, so if you are coming from an RDBMS background, think of the forest as you do a partition. On a side note, the reason it is called a forest is because many customers store XML in MarkLogic (as well as JSON, Full Text and Binary content). XML documents have tree structure, and there for many XML documents or "trees" make up a forest.

Forests in turn are made up of stands, and in these stands are where the documents and indexes reside. To maximize ingestion speed, MarkLogic utilizes in memory stands before permanently flushing data to an on disk stand. Transactional integrity is maintained through a journal which is on disk, so even if a memory stand were to be lost, the data is safe.

Each forest is managed by a data manager (D Node), and when queried, all forests are queried in parallel. Therefore placing multiple forests on a single host with multiple cores can help with concurrency.

Inside of the forest stands, documents are compressed.

Forest Details



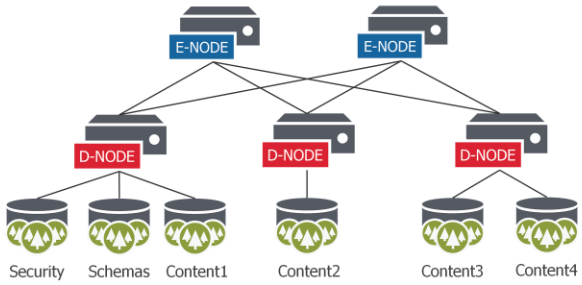
Demo a content load to show stands being created and merged when inserting / updating documents inside a forest.

Inside of the forest are stands, and stands contain documents in a compressed form and indexes. By default, all words and document structure get indexed. This Universal Index enables fast search regardless of document structure, and is a big part of what enables a schema agnostic approach to data modeling in MarkLogic.

Forests also need to be sized appropriately. Reindexing and merge operations occur in the background (enabling access to data while these operations occur) but also impact forest sizing guidelines.

Clustered Architecture

- MarkLogic is a shared-nothing distributed database, allowing linear scale out and high availability
  - 3 node minimum for High Availability

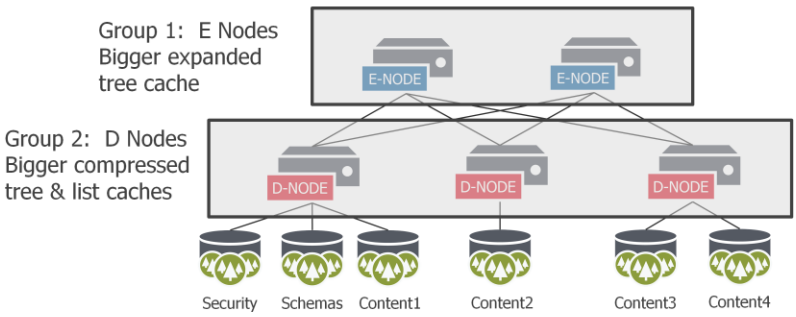


A MarkLogic cluster allows for scalability using a shared nothing architecture and deployed on commodity hardware. Machines in a cluster may specialize as an E Node or a D Node, dedicating more of their resources to the tasks and components associated with App Servers or Databases. This is not required – a cluster can also be a group of machines acting as both E/D Nodes.

As this design shows, each machine in the cluster can talk with the others. Queries are sent from the E Nodes to the D Nodes concurrently, and the results are aggregated and displayed back on the E Node. This map – reduce approach provides performance and scalability over large amounts of data.

Clustered Architecture

- Groups
  - Groups of host machines within a cluster
  - Enables more specific configuration

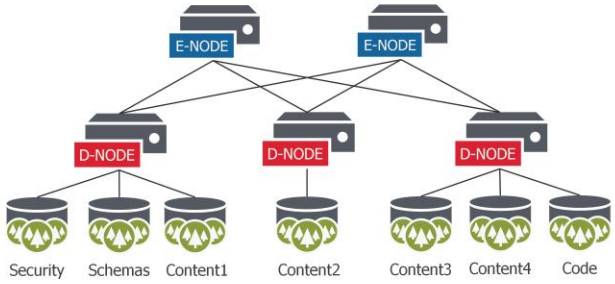


MarkLogic UNIVERSITY

Groups allow for specialization of machine configuration within a cluster.

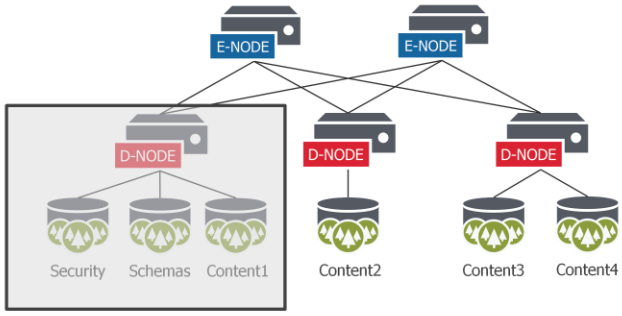
Clustered Architecture

- Common Databases
  - Can be shared across multiple projects
  - Security | Schemas | Triggers | Modules
  - Impacts to HA / DR design



Security 101

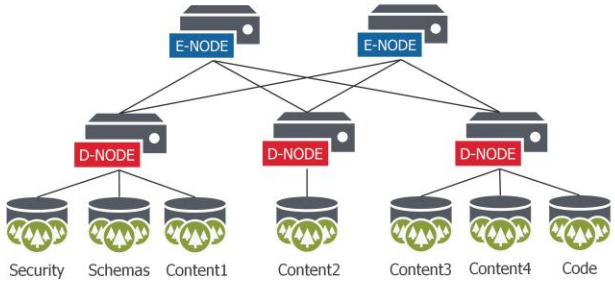
- Role based model
- Authentication
  - Performed on the application server
  - Utilize LDAP / Kerberos external authentication protocol
- Database Level
  - Who can read / write / update documents within a database?
- Code Level
  - Who can execute application code?



MarkLogic UNIVERSITY

Clustered Architecture

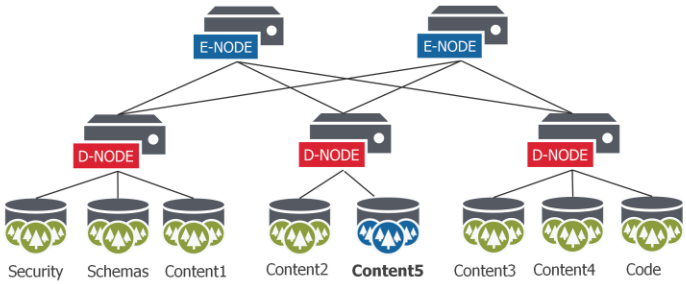
- Scalability
  - Scaling a cluster to support more data and/or users
    - Add more forests
    - Add more E nodes and / or D nodes



MarkLogic UNIVERSITY

Clustered Architecture

- Scalability & Rebalancing
  - Assume your data needs have increased
  - One of your D Nodes still has capacity
  - You add another forest – what happens when you load more data?
    - Pre-MarkLogic 7 (no automatic rebalancing)
    - MarkLogic 7 (rebalancing on by default)

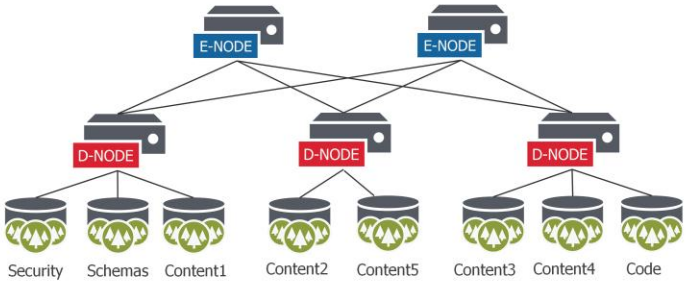


When you load data into a MarkLogic cluster, MarkLogic will load that data to a specific host with an instance of that database and put it into a specific forest. Based on this round robin approach, over time data tends to balance out across all hosts and forests in the cluster.

However, when you increase capacity, you need to determine if and how to automatically rebalance that data across the updated environment.

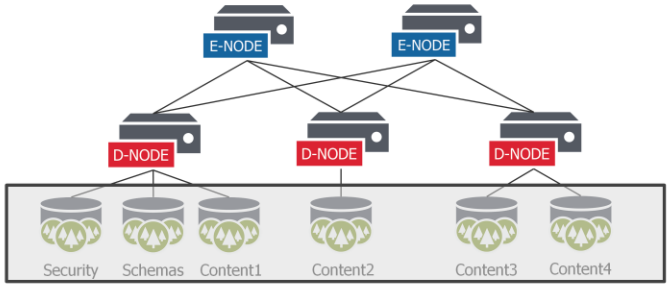
Clustered Architecture

- Rebalancing
  - Administrator can choose appropriate **assignment policy** and the **rebalancer** will automatically handle the data movement



Clustered Architecture

- Storage
  - Local, Shared, SSD, HDFS, Amazon S3
  - Tiered Storage
  - Optimized storage of large binary data



Slide 18

MarkLogic UNIVERSITY

For full details on supported storage types, please consult the MarkLogic documentation.

http://docs.marklogic.com

More on storage...

- MarkLogic enables you to choose a portfolio of storage options
- Design for your business, budget and performance goals

Rotating Disks	SSD	Amazon S3*	HDFS
Low Cost	Greater Cost	Low Cost	Low Cost
Performance impacted by many variables (controllers, latency, disk quality)	Fastest	Cloud based, globally distributed, access via HTTP, tight EC2 integration	Distributed, sequential I/O, configurable, replication
		Good for backups	

- *MarkLogic does not create journals on Amazon S3

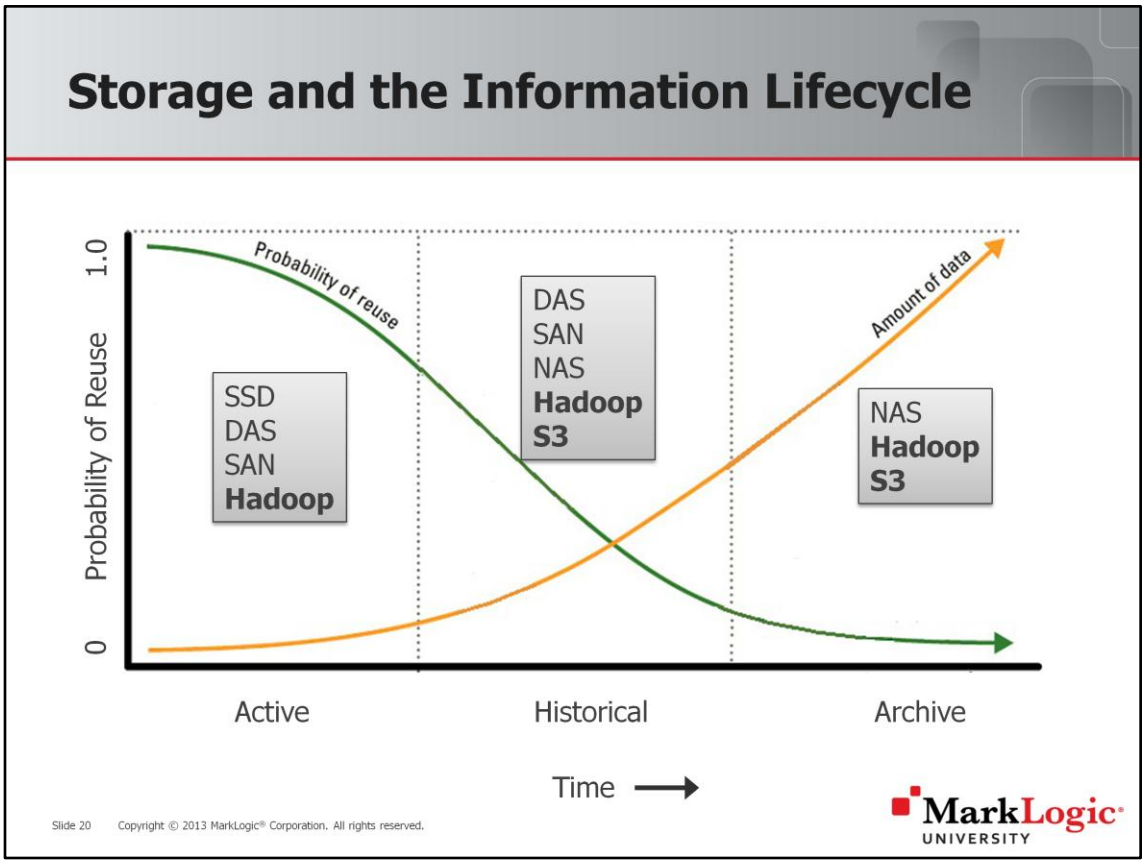
Slide 19

MarkLogic UNIVERSITY

*MarkLogic does not create journals on Amazon S3

If data stored in S3 is read only this is OK.

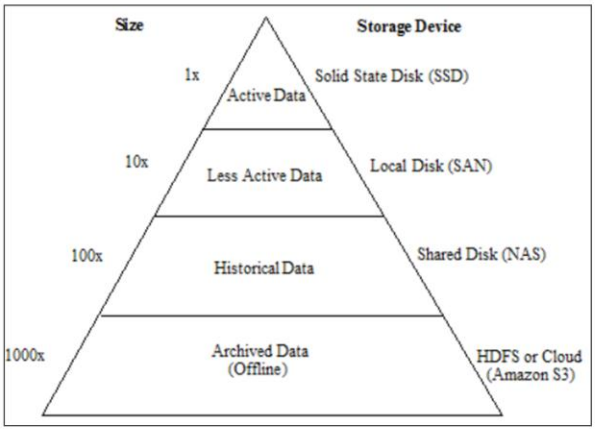
If data stored in S3 is changing, setup another storage type configured as the Fast Data Directory



Picking the right type of storage depends on where your data resides in its overall lifecycle. In MarkLogic, Tiered Storage gives you control and flexibility.

Tiered Storage Overview

- Objective:
  - Manage data using a portfolio of storage options
  - Design for your performance objectives and cost constraints
  - Partitions = groups of forests
    - APIs enabling you to migrate, resize, move offline/online, delete



MarkLogic UNIVERSITY

The objective of Tiered Storage in MarkLogic is to empower you to manage data using a portfolio of storage options to meet your performance objectives and cost constraints.

The MarkLogic tiered storage APIs enable you to actively and easily move your data between different tiers of storage.

As data ages and becomes less updated and queried, it can be migrated to less expensive and more densely-packed storage devices to make room for newer, more frequently accessed and updated data.

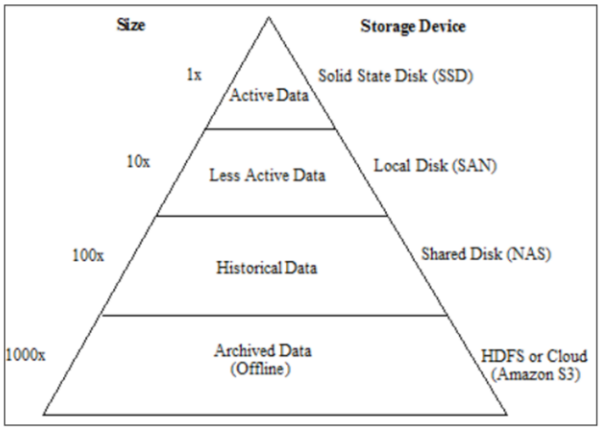
Tiered Storage Overview

- Conceptual implementation – where would these doc's end up?

<doc>
abc 123 xyz
<d>2013-10-22</d>
abc 123 xyz
</doc>

<doc>
abc 123 xyz
<d>1968-05-10</d>
abc 123 xyz
</doc>

<doc>
abc 123 xyz
<d>1985-08-16</d>
abc 123 xyz
</doc>



Partition Name	Forest Name	Range Definition <d>
V1	V1-01, V1-02	> 2013-01-01 < 2013-12-31 *Include LB
V2	V2-01, V2-02, V2-03	> 2000-01-01 < 2012-12-31 *Include LB
V3	V3-01, V3-02, V3-03, V3-04	> 1980-01-01 < 1999-12-31 *Include LB
V4	V4-01, V4-02, V4-03, V4-04, V4-05	> 1900-01-01 < 1979-12-31 *Include LB

Assume a database full of music information, similar to the one we will be building here shortly. Documents in this database have a piece of information representing a date value. We may choose to use this date to drive our storage policies.

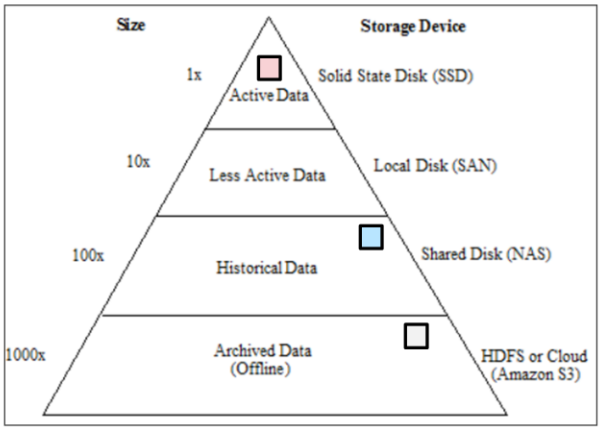
Tiered Storage Overview

- Conceptual implementation – where would these doc's end up?

<doc>
abc 123 xyz
<d>2013-10-22</d>
abc 123 xyz
</doc>

<doc>
abc 123 xyz
<d>05-10-1968</d>
abc 123 xyz
</doc>

<doc>
abc 123 xyz
<d>08-16-1985</d>
abc 123 xyz
</doc>

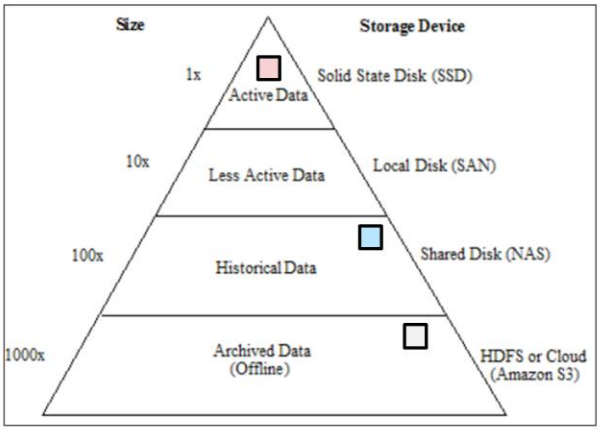


Partition Name	Forest Name	Range Definition <d>
V1	V1-01, V1-02	> 2013-01-01 < 2013-12-31 *Include LB
V2	V2-01, V2-02, V2-03	> 2000-01-01 < 2012-12-31 *Include LB
V3	V3-01, V3-02, V3-03, V3-04	> 1980-01-01 < 1999-12-31 *Include LB
V4	V4-01, V4-02, V4-03, V4-04, V4-05	> 1900-01-01 < 1979-12-31 *Include LB

Assume a database full of music information, similar to the one we will be building here shortly. Documents in this database have a piece of information representing a date value. We may choose to use this date to drive our storage policies.

Tiered Storage Overview

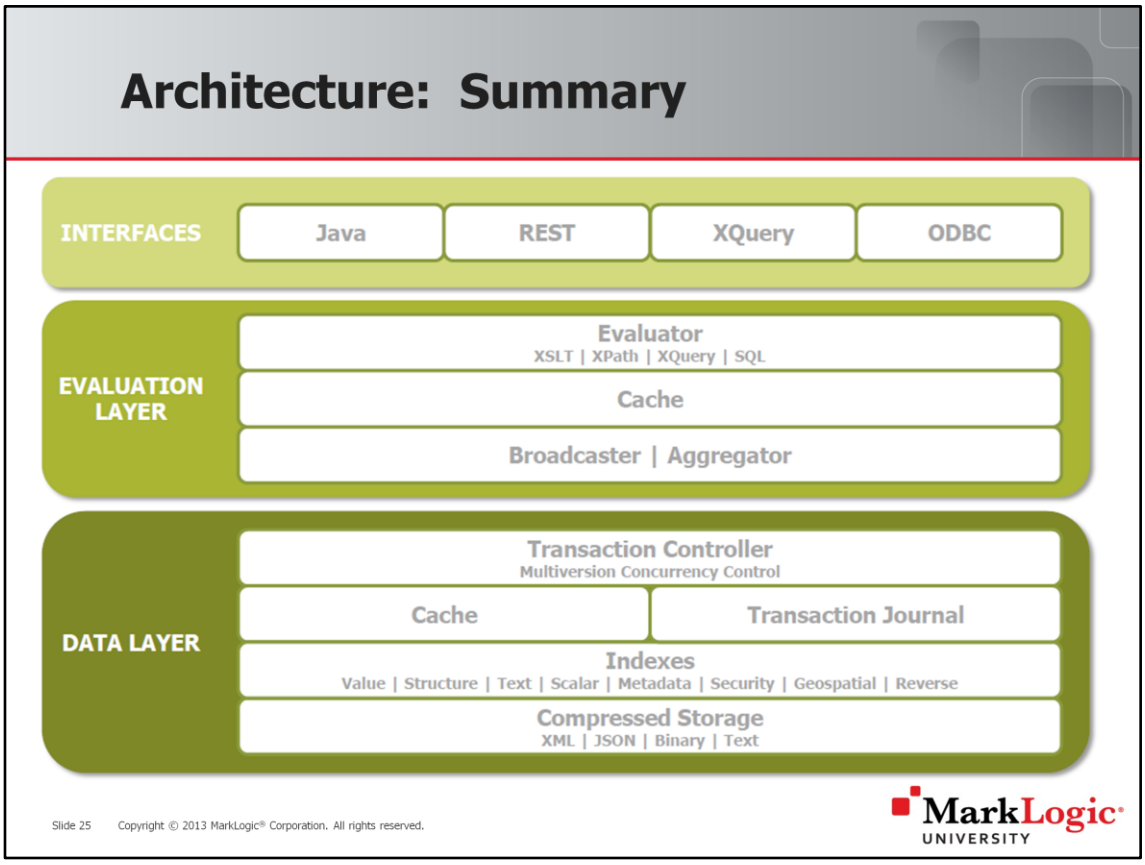
- What about access?
  - Each tier can be accessed individually
  - Or combined into a single unified system



Slide 24

It is possible to access each tier in the storage solution individually, or search across all tiers providing relevance ranked results across your entire unified system of data.

Search across the whole is accomplished through the use of a super-database, which is made up of sub-databases for each tier.



Here is a useful reference guide to visualize all the components of a MarkLogic architecture on a single page.

Unit 3: Applying the Learning Objectives

- Describe each of the following:
  - Database (Content / Modules / Schema / Triggers / Security)
  - Forest
  - Stand (In Memory / On Disk)
  - Universal Index
  - Application Server (HTTP / XDBC / ODBC / WebDAV)
  - Evaluator Node | Database Node
  - Hosts | Groups
  - List Cache | Compressed Tree Cache | Expanded Tree Cache
- Describe storage options in MarkLogic
- Compare / contrast a single host vs. cluster implementation
  - Exercise 1

MarkLogic UNIVERSITY