APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 23-25 SEPTEMBER 2013

Oak for Users

Michael Marth | Adobe Engineering

What is Oak?

What is Oak?

Oak Project

- Project “Oak”
  - aka Jackrabbit 3
  - all open source
  - Apache Software Foundation
    http://jackrabbit.apache.org/oak/
- Implementation of the JCR spec
  - Most used parts of JSR-283
  - **Not** another reference implementation
  - Complete rewrite from scratch

Oak Project Goals

- Scalability
  - Big repositories
  - Distributed, many cluster nodes
- Write throughput
  - Parallel writes
  - Write performance
- Many child nodes
- Many ACLs

Architecture concepts

Hamburger architecture

Oak API

MK API

Oak JCR

JCR specifics (e.g.versioning, nodetypes, namespaces)

1 per language

Oak core

search, locking, access control

1 in Java

Micro kernel

tree persistence

Many

The Burger's Bottom Bun: Micro Kernel

- Implements simply a tree model
  - Nodes and properties
- Exposes Micro kernel API
  - Tree + binaries
  - HTTP friendly, independent of technology
    - String /* jsonTree */ getNodes(String path, String revisionId, …);
    - String /* blobId */ write(InputStream in)
  - "Stringly typed", format is JSOP
    - lightweight HTTP protocol for manipulating JSON-based object models

The Burger's Patty: Oak Core

- Where most of the heavy lifting takes place
- Adds to MK's tree model:
  - ACLs
  - Search and Indexing
- Observation
- Exposes essentially a decorated tree model

adaptTo()

The Burger's Top Bun: Oak JCR, etc

- Implements the JCR API
- Mostly transforms JCR semantics into into tree operations
- Also contains "Commit hooks" that implement JCR constraints, e.g. node types
- Now implemented for JCR
  - Non-Java implementations possible and part of concept

Micro Kernel Implementations

Existing MK Implementations

- Segment MK
  - Tar storage “Tar MK”
  - Can also use other storage formats
- Mongo MK
  - Uses MongoDB for persistence
- HBase
  - Experimental, OAK-784
- H2
  - Based on H2 relational DB

adaptTo()

TarMK: how it works

- Segments
  - Immutable segments
  - Each segment identified by a UUID
  - Typically contains continuous subset of content tree
  - Locality of reference
  - Content in segment references other content
  - Segment keeps list of UUIDs of referenced segments

adaptTo()

TarMK: how it works

- Journals
  - Record the state of the hierarchy of journals for distributed writes
  - One journal for small systems
- Records
  - Content inside of segments
  - types are: blocks, lists, maps, values, templates and nodes

TarMK: how it works

- Segment MK is independent of physical storage format
- For TarMK: storage as .tar files
- For 64 bit Java systems: memory mapped files

adaptTo()

TarMK: when to use

- Standalone repositories (not clustered)
  - SegmentMK is designed as a distributed MK, but TarMK not (yet) taking advantage
- When performance matters
  - Memory mapped files
  - Designed to minimize network or disc access

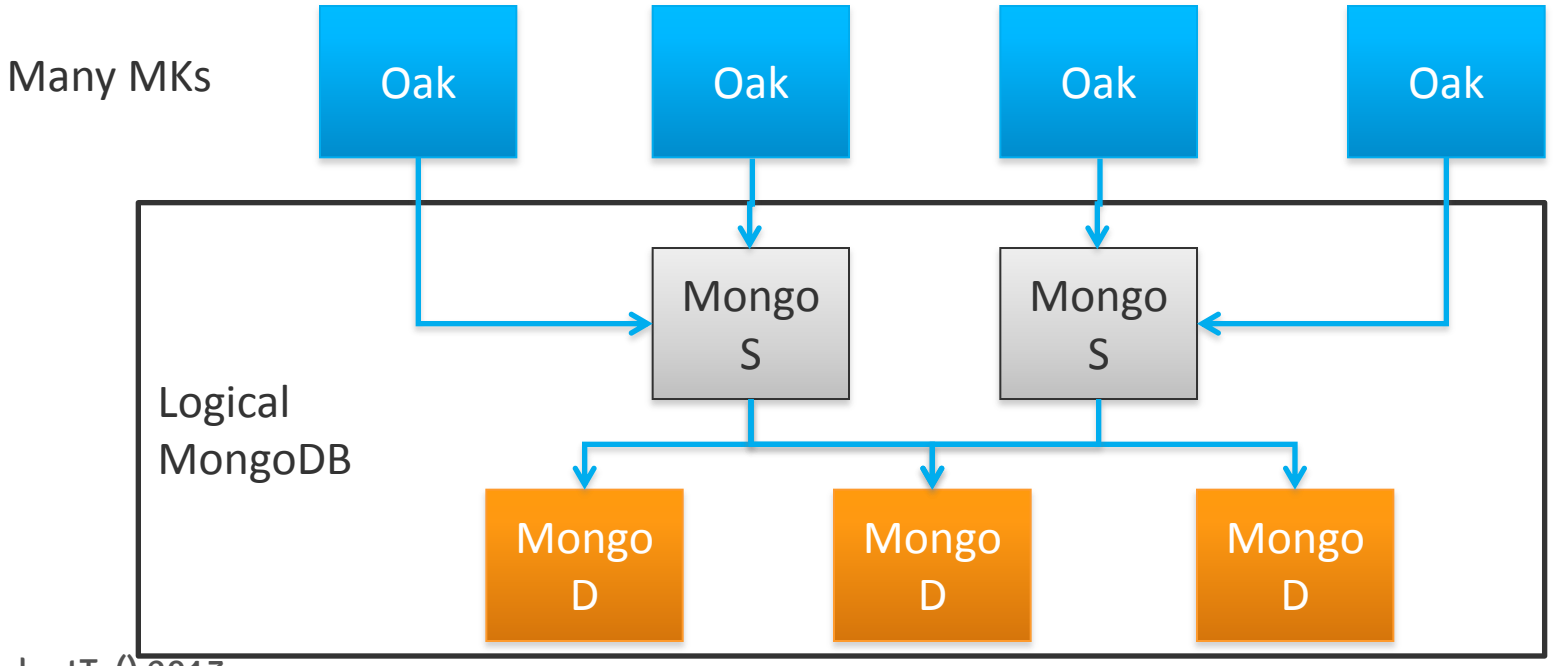
adaptTo()

MongoMK: how it works

- stores tree in one Mongo data base
- each node is a separate document
- updates are stored by adding new revision/value pairs
  - i.e. each document contains the current and past revisions of the node
- leverages Mongo sharding for clustering

MongoMK: when to use

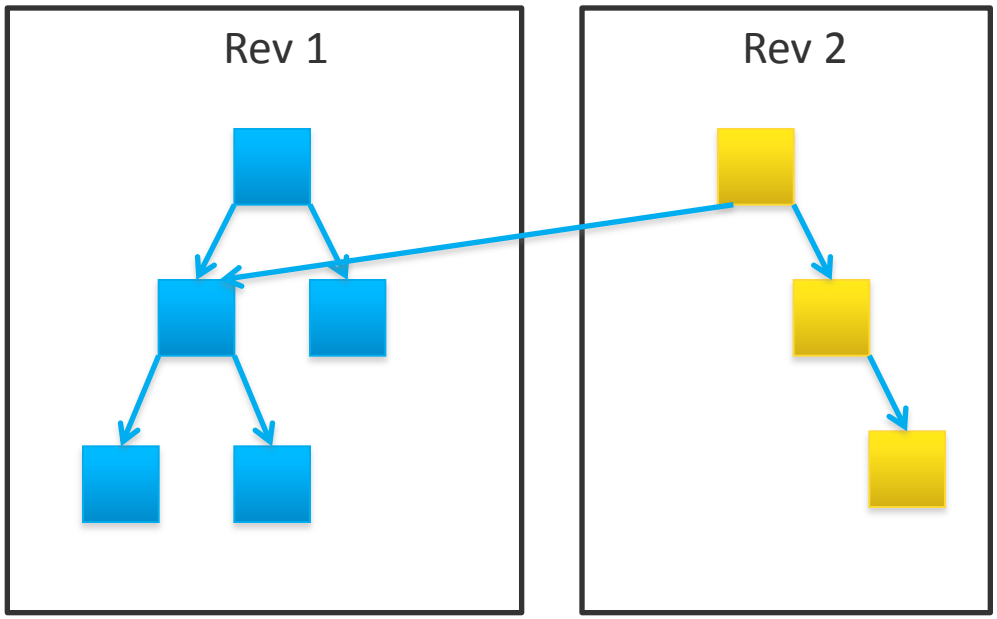
- use for clustered setups
- now: sharding, future: reading from replicas for increased scalability



Implementation Concepts

Multiversion concurrency control (MVCC)

- Each session sees a snapshot of the repo
- Snapshot isolation (repeatable reads)
- Improved concurrency



adaptTo()

Search and Indexing

- All indexes are user created, not automatically created
- Query Engine evaluates indexes and picks index with lowest cost to answer query
- Index providers are pluggable, e.g.:
  - Full text (Lucene)
  - Property index (in repo)
  - Solr (for distributed index)

Commit Hooks

- Oak Core is pluggable
- One important type of plugin: “Commit Hooks”
  - Validate and modify commits
  - Examples:
    - indexing
    - some security checks
    - node type consistency
    - conflicting commits

Branches

- MK-level concept: persist large commits into private “branch” of MK
- Merge into head when full commit is persisted
- Useful for very large commits that cannot be held in memory in transient space

Changes to Jackrabbit2 on user level

Almost Nothing (aka the Power of JCR)

Sessions

- Sessions will not automatically refresh (MVCC)
  - Manually call session.refresh()
  - Changes from sessions lower in the call hierarchy are visible to sessions higher up (the “service that uses admin session”-case)

Indexing

- Need to create custom indexes for queries you intend to perform
  - how to: add index definition node to repo
- Otherwise “TraversingIndex” will be used

Observation

- Observation Events generated from periodically pulled repository diffs
  - getUserId() and getUserData() only available for locally generated events (i.e. on the same cluster node)
  - Adding and deleting the same node will not trigger any event if it happens inside one pull interval

Misc

- Same Name Siblings
- Security (Authentication, Access Control Management, Permissions, etc)

  jackrabbit.apache.org/oak/docs/0.9-SNAPSHOT/differences.html

State of the project

State

- Still lots to do (and *opportunity* to get *involved*), but
  - AEM (very large JCR application) already running on Oak
  - Adobe R&D engaging with AEM customers for alpha testing soon

Thank you! Any questions?