APACHE SLING & FRIENDS TECH MEETUP

BERLIN, 26-28 SEPTEMBER 2012

Oak / Solr integration

Tommaso Teofili

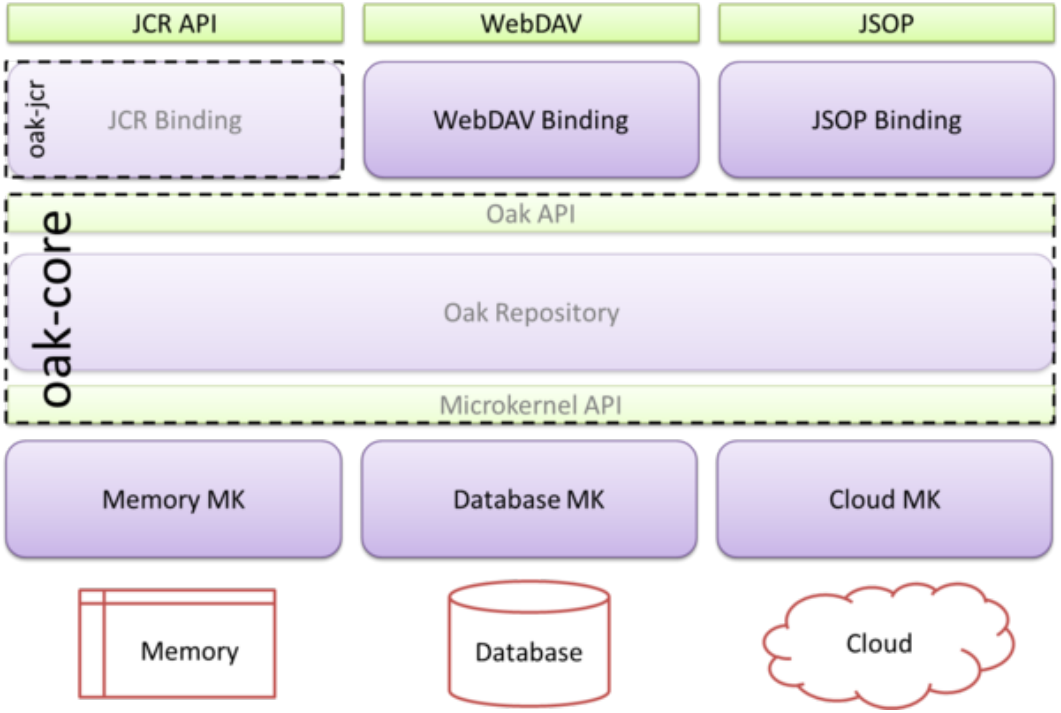
Agenda

- Why
- Search on Oak with Solr
- Solr based QueryIndex
- Solr based MK
- Benchmarks

Why

- Common need:
  - Once you have content
  - You (usually also) want search
  - Both need to be:
    - Fast
    - Scalable
    - Faul tolerant
- Less common need:
  - Substitute / decorate internal default query engine for enhanced performance / expressivity

Why


JCR API
WebDAV
JSOP
oak-jcr
JCR Binding
WebDAV Binding
JSOP Binding
oak-core
Oak API
Oak Repository
Microkernel API
Memory MK
Database MK
Cloud MK
Memory
Database
Cloud


Solr Search Engine

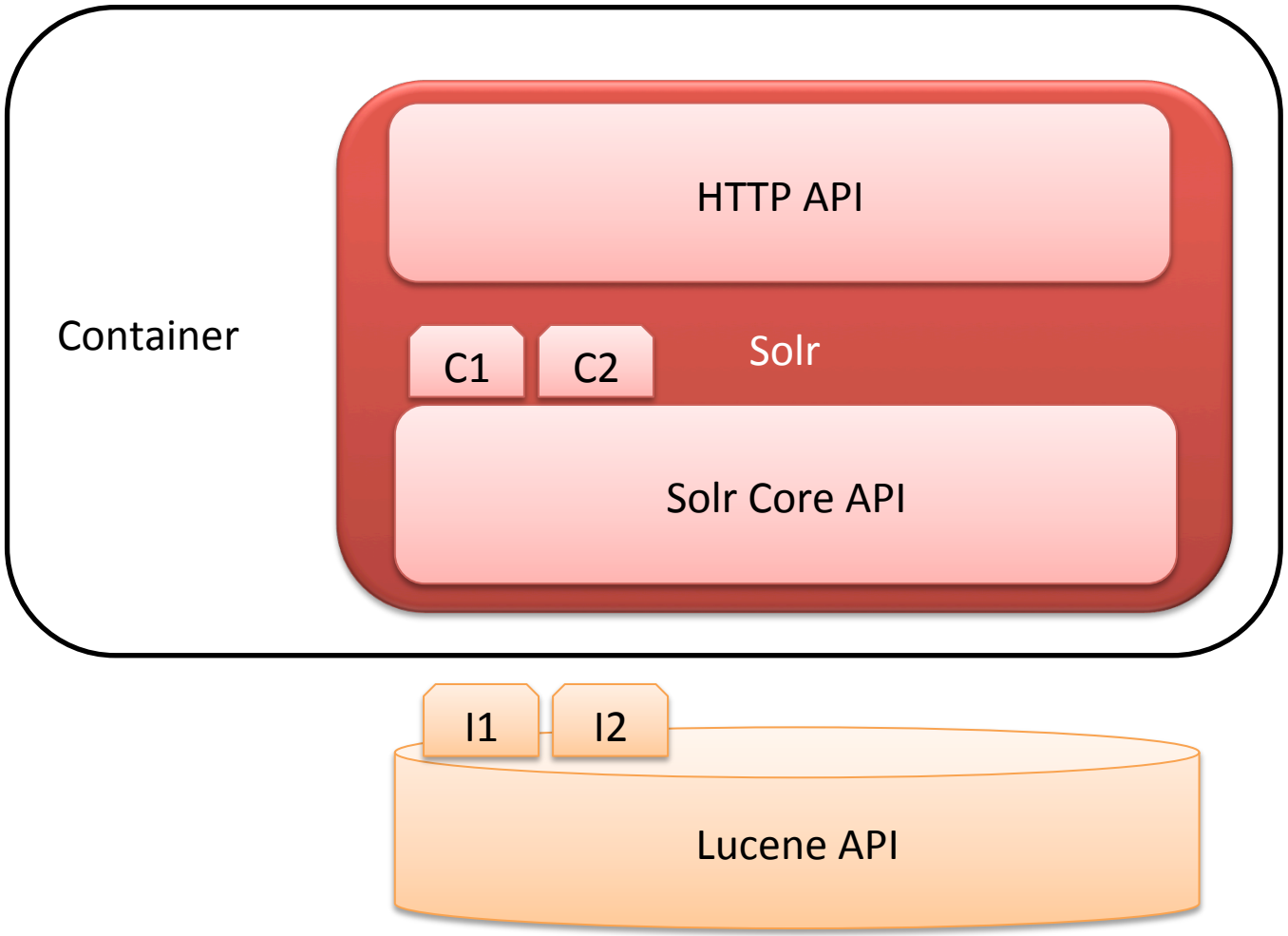

Solr MK

adaptTo()

Apache Solr 101

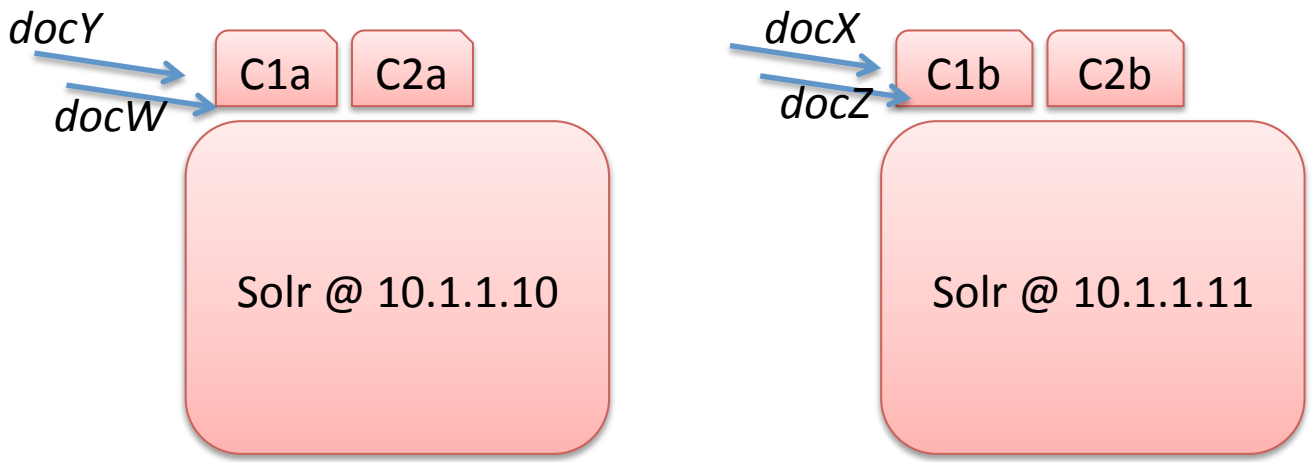
- Apache project
- Enterprise search server
- Based on Apache Lucene
- HTTP API
- Easy and quick setup
- Scaling architectures

Solr simplest architecture

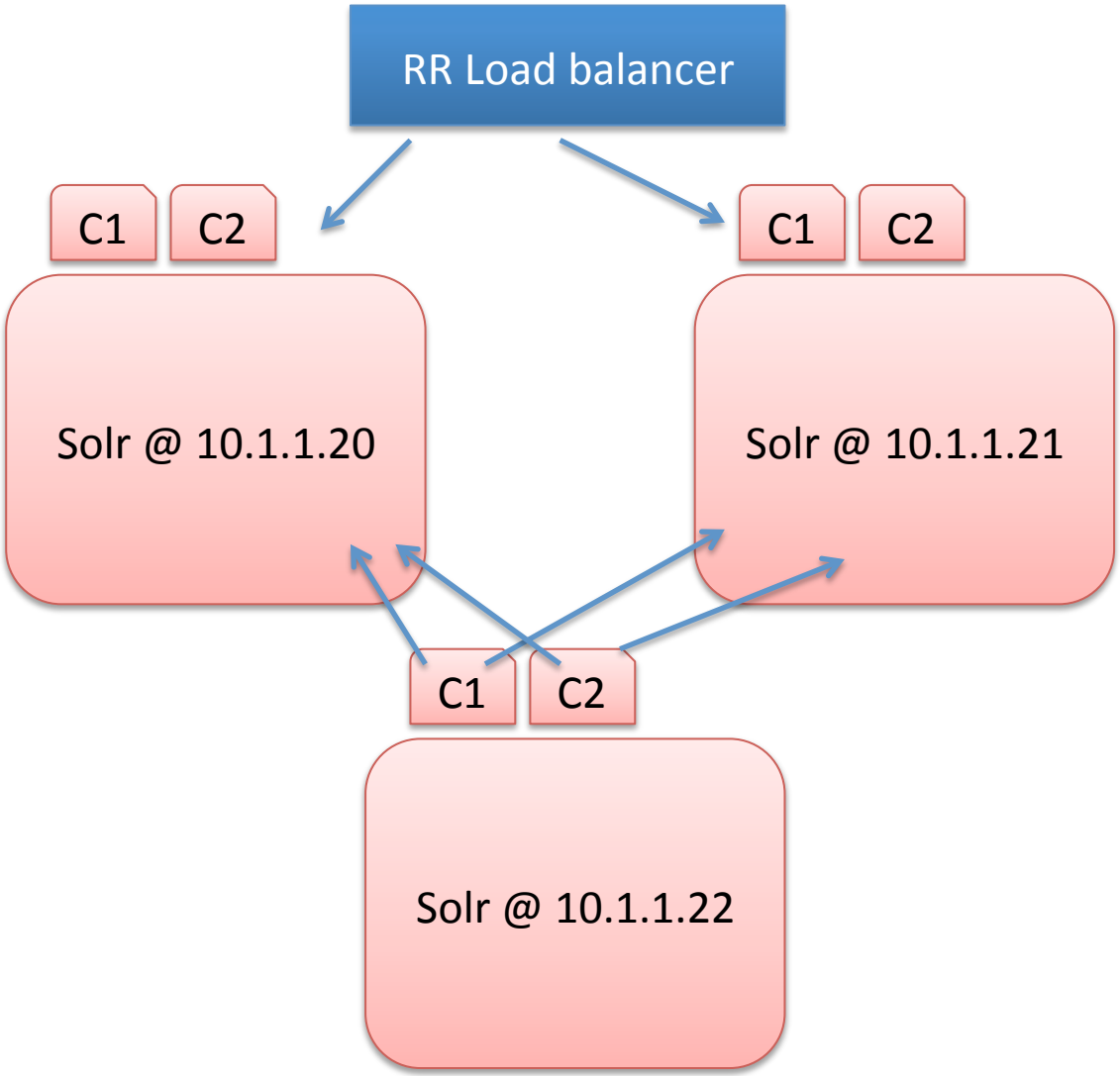


Solr sharded architecture

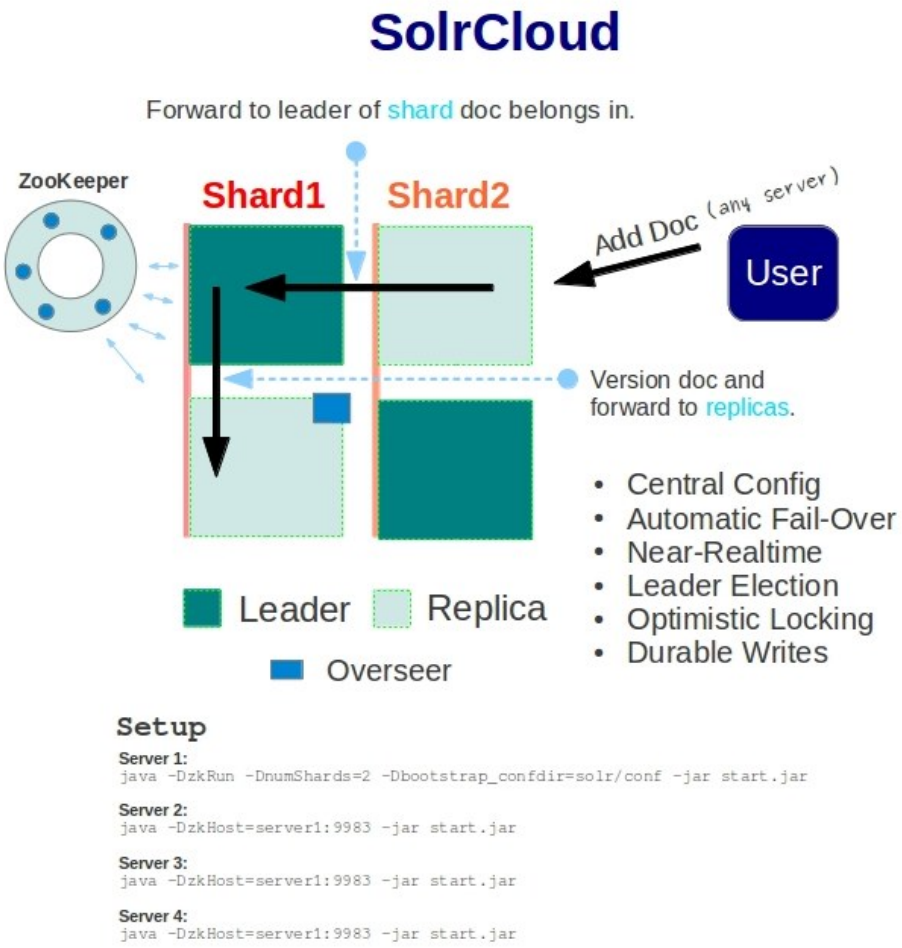
- Split a collection in N shards when indexing
- Search on both with distributed search:
  - http://.../select?q=...&shards=10.1.1.10/solr/c1a,10.1.1.11/solr/c1b



Solr replicated architecture



SolrCloud*



*: *Sorry I was too lazy to re picture this from scratch*

Search on Oak with Solr

- Content is created inside Oak
- Oak repository gets synchronized with Solr
- Solr handles search requests
  - Separation of concerns
  - Scaling independently:
    - Oak can scale for CRUD operations on repository
    - Solr can scale for indexing / search

Synchronizing Oak and Solr

- Known options
  - Push from Oak
  - Pull with Solr (use Oak HTTP API with Solr DIH)
  - Apache ManifoldCF ?
  - ...

adaptTo()

Push Oak commits to Solr

- CommitHook API
  - evaluate repository status before and after a specific commit
  - create a “diff”
  - do something with the diff
    - -> send adds / deletes to Solr
- The CommitHook is executed right before content is actually persisted
  - See https://github.com/apache/jackrabbit-oak/blob/trunk/doc/nodestate.md

Push Oak commits to Solr

```
// add the SolrCommitHook to the node store
store.setHook(new SolrCommitHook());
// obtain the node store root for a specific workspace
Root root = new RootImpl(store, “solr-test”, ...);
// add a node with path ‘doc1’ and a text property
root.getTree("/").addChild("doc1").setProperty("text",
valueFactory.createValue("hit that hot hat tattoo"));
// commit the change to the node store
r.commit(DefaultConflictHandler.OURS);
```

```
INFO: [collection1] webapp=null path=/update params={} {add=[doc1 (1413611393772421120)]} 0 117
Sep 20, 2012 9:07:59 AM org.apache.solr.update.DirectUpdateHandler2 commit
INFO: start commit{flags=0,_version_=0,optimize=false,openSearcher=true,waitSearcher=false,expungeDeletes=false,softCommit=t
Sep 20, 2012 9:07:59 AM org.apache.solr.search.SolrIndexSearcher <init>
INFO: Opening Searcher@64428527 main
Sep 20, 2012 9:07:59 AM org.apache.solr.update.DirectUpdateHandler2 commit
INFO: end_commit_flush
```

Push Oak commits to Solr

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">802</int>
    <lst name="params">
      <str name="wt">xml</str>
      <str name="q">*:*</str>
    </lst>
  </lst>
  <result name="response" numFound="1" start="0" maxScore="1.0">
    <doc>
      <str name="id">doc1</str>
      <arr name="text">
        <str>hit that hot hat tattoo</str>
      </arr>
      <long name="_version_">1413554115032645632</long>
    </doc>
  </result>
</response>
```

Search on Oak with Solr - Scaling

- *SoftCommitHook* uses SolrJ's *SolrServer*
  - *SolrServer solrServer = …*
    - *new HttpSolrServer("http://..:8983/solr") ; // standalone*
    - *new LBHttpSolrServer("http://..:8983/solr", "http://..:7574/solr"); // replication – sw load balancing*
    - *new CloudSolrServer("100.10.13.14:9983"); // SolrCloud*

adaptTo()

What if commit fails?

- Could lead to not consistent status
  - Oak commits not going through
  - Related data indexed on Solr
- Use ***Observer***
  - *contentChanged(NodeStore store, NodeState before, NodeState after);*
    - Node state is just read
    - Only triggered on successfully persisted commits

Solr based QueryIndex

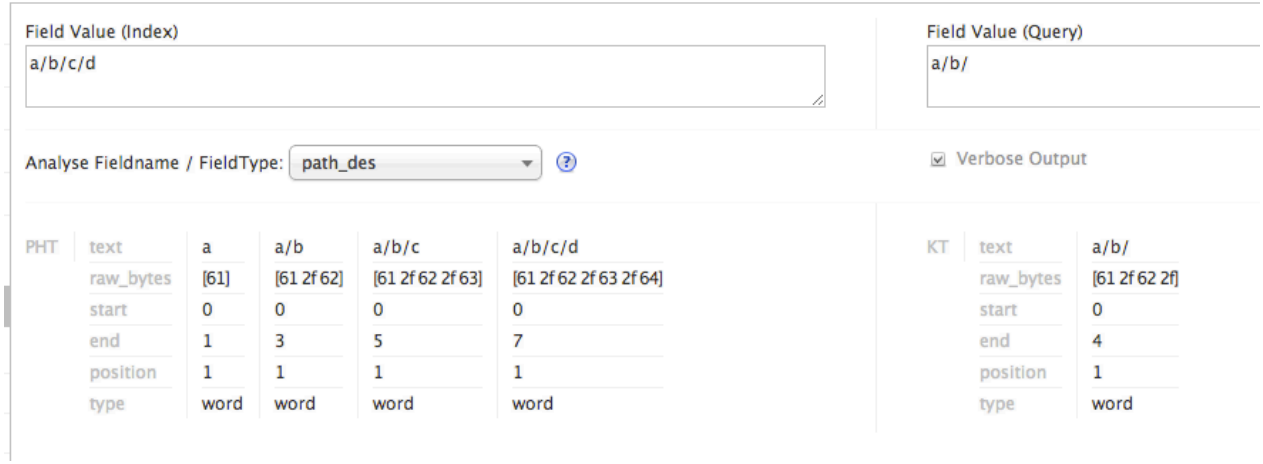
- QueryIndex API
  - Evaluate the query (Filter) cost
  - Eventually view the query “plan”
  - Execute the query against a specific revision and root node state

Solr based QueryIndex

- getCost(Filter);
  - Property restrictions:
    - Each property is mapped as a field
    - Can use term queries for simple value matching
    - Can use range queries (with trie fields) for “first to last”

Solr based QueryIndex

- getCost(Filter);
  - Path restrictions
    - Indexed as strings
    - And as paths with PathHierarchyTokenizerFactory
    - Ancestor / descendant is as fast as exact match
    - Direct children needs special handling



Solr based QueryIndex

- ancestors

http://localhost:8983/solr/collection1/select?q=path_anc%3Aa%5C%2Fb&wt=xml

Click entity for XPath. Double-click to collapse/expand. Enter XPath or XML string then click XPath/Render for results or to XML Tree-ify, respectively.

Source Options

XPath/Render X

```
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">12</int>
    <lst name="params">
      <str name="wt">xml</str>
      <str name="q">path_anc:a\/b</str>
    </lst>
  </lst>
  <result name="response" numFound="2" start="0" maxScore="1.134265">
    <doc>
      <str name="id">a</str>
      <long name="_version_">1413625697965965312</long>
    </doc>
    <doc>
      <str name="id">a/b</str>
      <long name="_version_">1413625697972256768</long>
    </doc>
  </result>
</response>
```

Solr based QueryIndex

- descendants

```
      <str name="q">path_des:a\/b</str>
    </lst>
  </lst>
  <result name="response" numFound="4" start="0" maxScore="0.625">
    <doc>
      <str name="id">a/b</str>
      <long name="_version_">1413625697972256768</long>
    </doc>
    <doc>
      <str name="id">a/b/doc4</str>
      <arr name="text">
        <str>hats tattoos hit hot</str>
      </arr>
      <long name="_version_">1413625697967013888</long>
    </doc>
    <doc>
      <str name="id">a/b/doc3</str>
      <arr name="text">
        <str>tattoos hate hot hits</str>
```

Solr based QueryIndex

- getCost(Filter);
  - Full text conditions
    - Easiest use case
    - Can use (E)DisMax Solr query parser
      - Q=+term1 +term2 term3
      - And fields are defined by Solr configuration

Solr based QueryIndex

- query(Filter filter, String revisionId, NodeState root);
  - *filter* gets transformed in a SolrQuery
  - *root* and *revisionId* can be used to map Oak revisions to Lucene / Solr commit points
  - the resulting Cursor wraps the Solr query response

Custom indexes configuration

- Discussion is going on at :
  - http://markmail.org/message/bdgi77dd6wy2hkbp
- Something like:

/data [jcr:mixinTypes = oak:indexed]

/oak:indexes

/solr [jcr:primaryType = oak:solrIndex, oak:nodeType = nt:file, url = ...]

adaptTo()

Extending default query syntax

- Combining JCR-SQL2 / XPath queries with full text search on Solr
- i.e.: *select * from parent as p inner join child as c on issamenode(p, c, ['/a/b/c'])* ***and solr(“(+hit +tattoo^2) OR “hot tattoo”~2”)***
  - First part can be handled in Oak as usual
  - Second part can be run on Solr
  - Need to extend the Oak AST

Solr based MK

- MicroKernel API
  - MVCC model
  - Json based data model
  - Possibly use some retention policy
  - Basically noSQL
- Can Solr fit ?

adaptTo()

Solr based MK

- MVCC
  - Solr
    - SVCC
    - per document optimistic lock – free mechanism
    - related issues : SOLR-3178, SOLR-3173
    - Solutions:
      - map Oak revisions to Lucene commit points
      - store revisions inside Solr documents
- Json based data model
  - already available in Solr

Solr based MK

- Retention policy
  - Already available in Solr using Lucene commit points for revisions
  - Leveraging IndexDeletionPolicy API

adaptTo()

Solr based MK

- noSQL
  - Solr
    - not relational, avoid joins
    - no fixed schema (!= schema.xml)
    - scalability
    - See
      - http://searchhub.org/dev/2010/04/30/nosql-lucene-and-solr/
      - http://searchhub.org/dev/2010/04/29/for-the-guardian-solr-is-the-new-database/

Solr based MK

- Still a prototype implementation
- Pros
  - Most features exist out of the box
  - Scaling nicely
- Cons
  - MVCC handling is not straightforward

Benchmarking

- SolrCommitHook with commit():
  - CreateManyChildNodes : some impact (1.4x)
  - DescendantSearch : almost no impact
  - SetProperty : huge impact (3x)
  - SmallFileWrite : some impact (1.3x)
  - UpdateManyChildNodes : almost no impact

adaptTo()

Benchmarking

- SolrCommitHook with autoSoftCommit:
  - CreateManyChildNodes : some impact (1.2x)
  - DescendantSearch : almost no impact
  - SetProperty : **no impact**
  - SmallFileWrite : **almost no impact**
  - UpdateManyChildNodes : almost no impact

Let's improve it

- It's work in progress !
  - Oak/Solr integration Jira:
    - https://issues.apache.org/jira/browse/OAK-307
  - Oak Github fork:
    - https://github.com/tteofili/jackrabbit-oak