

APACHE
LUCENE
EUROCON



Solr 4 Highlights

Mark Miller, LucidImagination.com
markrmiller@gmail.com

Presented by

lucid
IMAGINATION



What I Will Cover

- What are some cool features coming in Solr 4?
- Mark Miller - Lucene/Solr committer, PMC member, Apache member, blah blah blah
- DirectSolrSpellChecker
- NRT
- Realtime Get
- SolrCloud - search side
- SolrCloud - indexing side (WIP)

DirectSolrSpellchecker

- New spellcheck implementation that works with the new automaton code.
- A practical benefit of this spellchecker is that it requires no additional data structures (neither in RAM nor on disk) to do its work.
- Also, no build phase - suggestions are always up to date.

First Came...

LUCENE-1606, LUCENE-2089:

Adds AutomatonQuery, a MultiTermQuery that matches terms against a finite-state machine. Implement WildcardQuery and FuzzyQuery with finite-state methods. Adds RegexpQuery.

(Robert Muir, Mike McCandless, Uwe Schindler, Mark Miller)

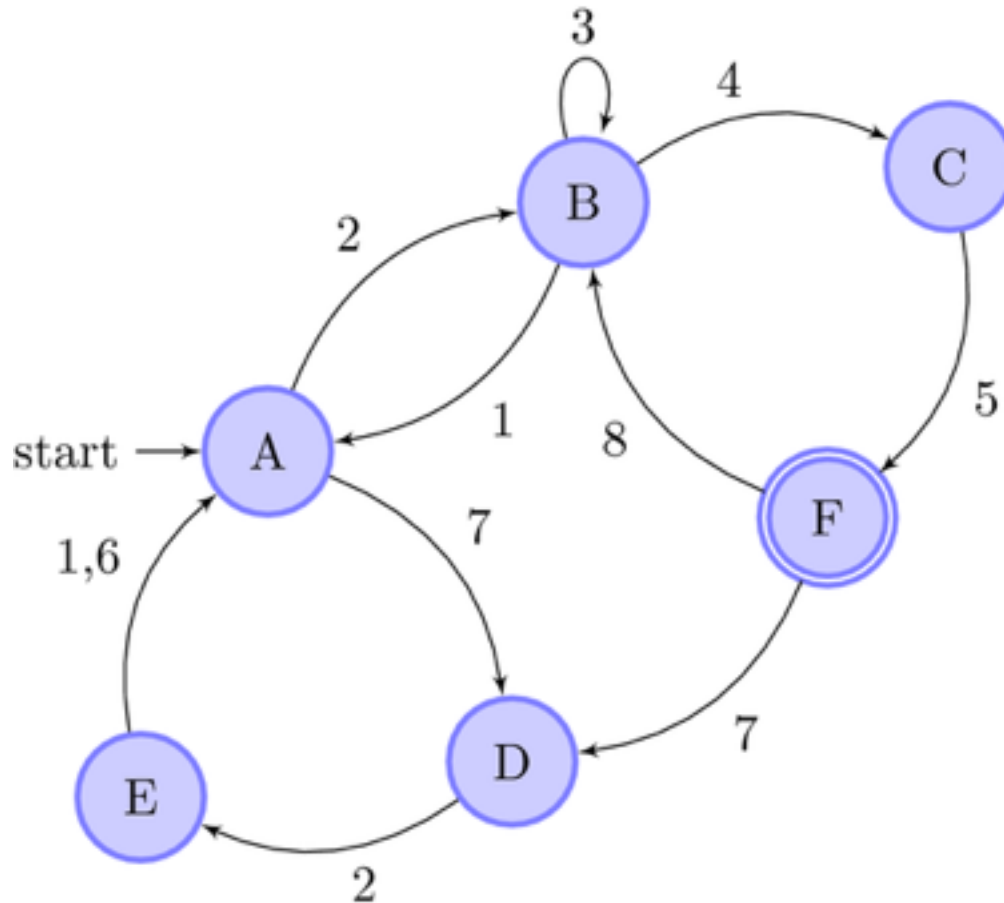
Leading too...

LUCENE-2507, **SOLR-2571**, **SOLR-2576**:

Added DirectSolrSpellChecker, which uses Lucene's DirectSpellChecker to retrieve correction candidates directly from the term dictionary using levenshtein automata.

(James Dyer, rmuir)

Automaton



SolrConfig

```
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">
```

```
<!-- a spellchecker built from a field of the main index -->
```

```
<lst name="spellchecker">
```

```
  <str name="name">default</str>
```

```
  <str name="field">name</str>
```

```
  <str name="classname">solr.DirectSolrSpellChecker</str>
```

Realtime-Get

SOLR-2656:

realtime-get, efficiently retrieves the latest stored fields for specified documents, even if they are not yet searchable (i.e. without reopening a searcher)

(yonik)

Realtime Get

```
<updateLog class="solr.FSUpdateLog">  
  <str name="dir">${solr.data.dir:}</str>  
</updateLog>
```

<!-- realtime get handler, guaranteed to return the latest stored fields of any document, without the need to commit or open a new searcher. The current implementation relies on the updateLog feature being enabled.

```
-->  
<requestHandler name="/get" class="solr.RealTimeGetHandler">  
  <lst name="defaults">  
    <str name="omitHeader">true</str>  
  </lst>  
</requestHandler>
```

Realtime Get

```
curl 'http://localhost:8983/solr/update/json?commitWithin=10000000'  
-H 'Content-type:application/json' -d '  
[  
  {  
    "id" : "mydoc",  
    "title" : "realtime-get test!"  
  }  
]'
```

<http://localhost:8983/solr/get?ids=mydoc,mydoc>

<http://localhost:8983/solr/get?id=mydoc&id=mydoc>

```
{"doc":{"id":"mydoc","title":["realtime-get test!"]}}
```

NRT

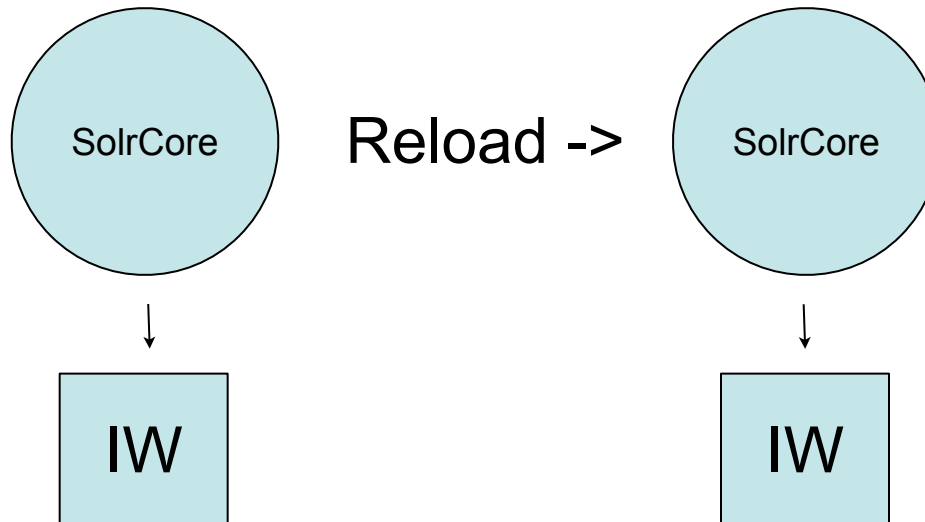
UpdateHandler Re-Architecture

- Fix commit waits for background merges
- Fix reloaded core does not share IndexWriter
- Remove unnecessary locking
- Add NRT
- SoftCommit
- SoftAutoCommit

Trying to do NRT

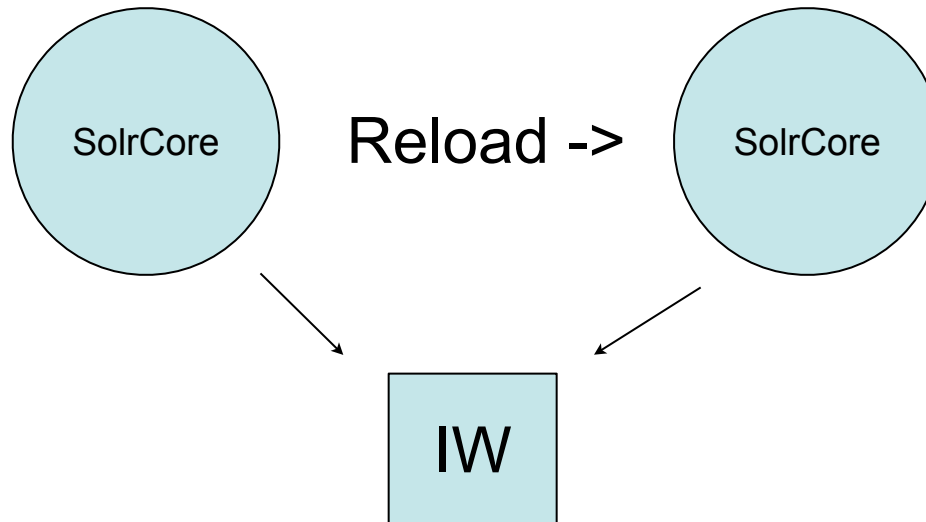
- Commit - 1 second.
- Commit - 0.5 seconds.
- Commit - 1.2 seconds.
- Commit - 1.5 minutes. **BACKGROUND MERGE**
- Commit - 0.9 seconds.

Previously....



Usually worked because IW is lazy instantiated...but overlap is possible. What happens when continuously feeding Solr and you do a core reload?
IW Lock timeout!

Solution



Reloaded SolrCore now shares a State object

NRT AutoCommit

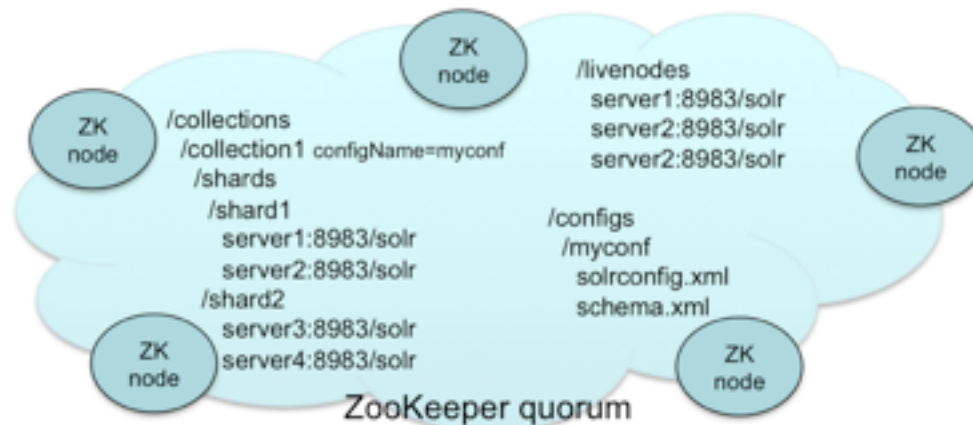
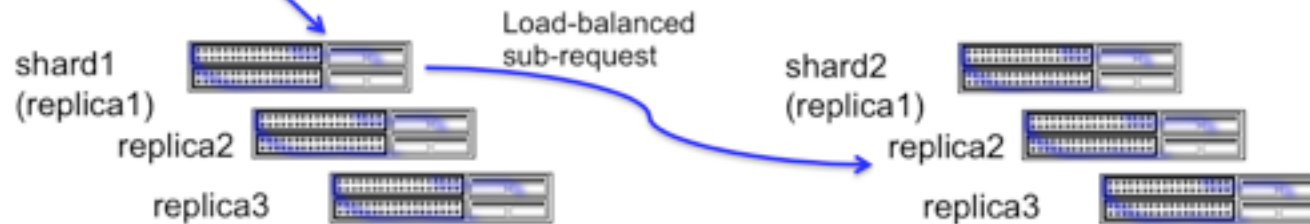
```
<updateHandler class="solr.DirectUpdateHandler2">
```

```
  <autoCommit>  
    <maxTime>60000</maxTime>  
  </autoCommit>
```

```
  <autoSoftCommit>  
    <maxTime>1000</maxTime>  
  </autoSoftCommit>
```

SolrCloud - Search Side

<http://.../solr/collection1?distrib=true>



SolrCloud - Search Side



SOLR-1873:

SolrCloud - added shared/central config and core/shard management via zookeeper, built-in load balancing, and infrastructure for future SolrCloud work.

(yonik, Mark Miller)

ZooKeeper Layout

```
/solr
```

```
/solr/configs
```

```
/solr/configs/conf1
```

```
/solr/configs/conf1/mapping-ISOLatin1Accent.txt
```

```
DATA: ...supressed...
```

```
/solr/configs/conf1/protwords.txt
```

```
DATA: ...supressed...
```

```
/solr/configs/conf1/old_synonyms.txt
```

```
DATA: ...supressed...
```

```
/solr/configs/conf1/solrconfig.xml
```

```
DATA: ...supressed...
```

```
/solr/configs/conf1/stopwords.txt
```

```
DATA: ...supressed...
```

```
/solr/configs/conf1/schema.xml
```

```
DATA: ...supressed...
```

← Centralized Configuration

Cluster Composition



```
/solr/collections/collection1/shards
```

```
/solr/collections/collection1/shards/shard4/localhost:50270_solr_
```

```
DATA:
```

```
node_name=localhost:50270_solr
```

```
url=http://localhost:50270/solr/
```

Improvements Coming...

- Modeling the cluster as ZooKeeper nodes is cool and convenient. But reading the cluster is costly - many requests.
- This can be mitigated with an incremental update mode.
- But that makes it difficult to monitor for on the fly property changes...too many ZooKeeper Watches needed.
- Solution - store the cluster state as the data for a single node.
- Downsides: 1MB max data size on zk node. Possibly less convenient.

Ephemeral Nodes

```
/solr/live_nodes
```

```
/solr/live_nodes/localhost:50272_solr
```

```
/solr/live_nodes/localhost:50268_solr
```

```
/solr/live_nodes/localhost:50274_solr
```

```
/solr/live_nodes/localhost:50270_solr
```

```
/solr/live_nodes/localhost:50266_solr
```

Nodes know which other nodes are down. There is some lag - nodes also manage their own list of good nodes to handle the lag time.

Getting Started

<http://wiki.apache.org/solr/SolrCloud>

```
java -Dbootstrap_confdir=./solr/conf  
-Dcollection.configName=myconf  
-DzkRun  
-jar start.jar
```

Run an internal
ZK server

Upload /solr/conf
to ZK and call it
"myconf"

<http://localhost:8983/solr/collection1/admin/zookeeper.jsp>

SolrCloud - Index Side

Goals

- Distributed Indexing
- Fault Tolerant
- Elastic
- Durable

Current Vision

- You figure out you need 20 servers at least to serve your index.
- You specify 20 shards to Solr and launch 60 servers with minimum configuration needed.
- This gives you 20 shards, each with 3 replicas.
- You begin indexing to any of the 60 servers and documents are pushed around to where they belong and are visible to search within seconds.
- If servers die, you don't lose data (unless a full shard goes down and the data on each server is unrecoverable).

SolrCloud - Index Side

What's Being worked on this very moment?

- Realtime Get
- Transaction Log
- Versions
- Leader per Shard
- Auto Assign Nodes to Shards
- Distributed IDF

Versions

- If two updates to a document are issues around the same time, how do we know which document should be the latest? More important, how can we be sure the same document ‘wins’ on each replica.
- Lamport Clocks, Vector Clocks, a Leader per shard?

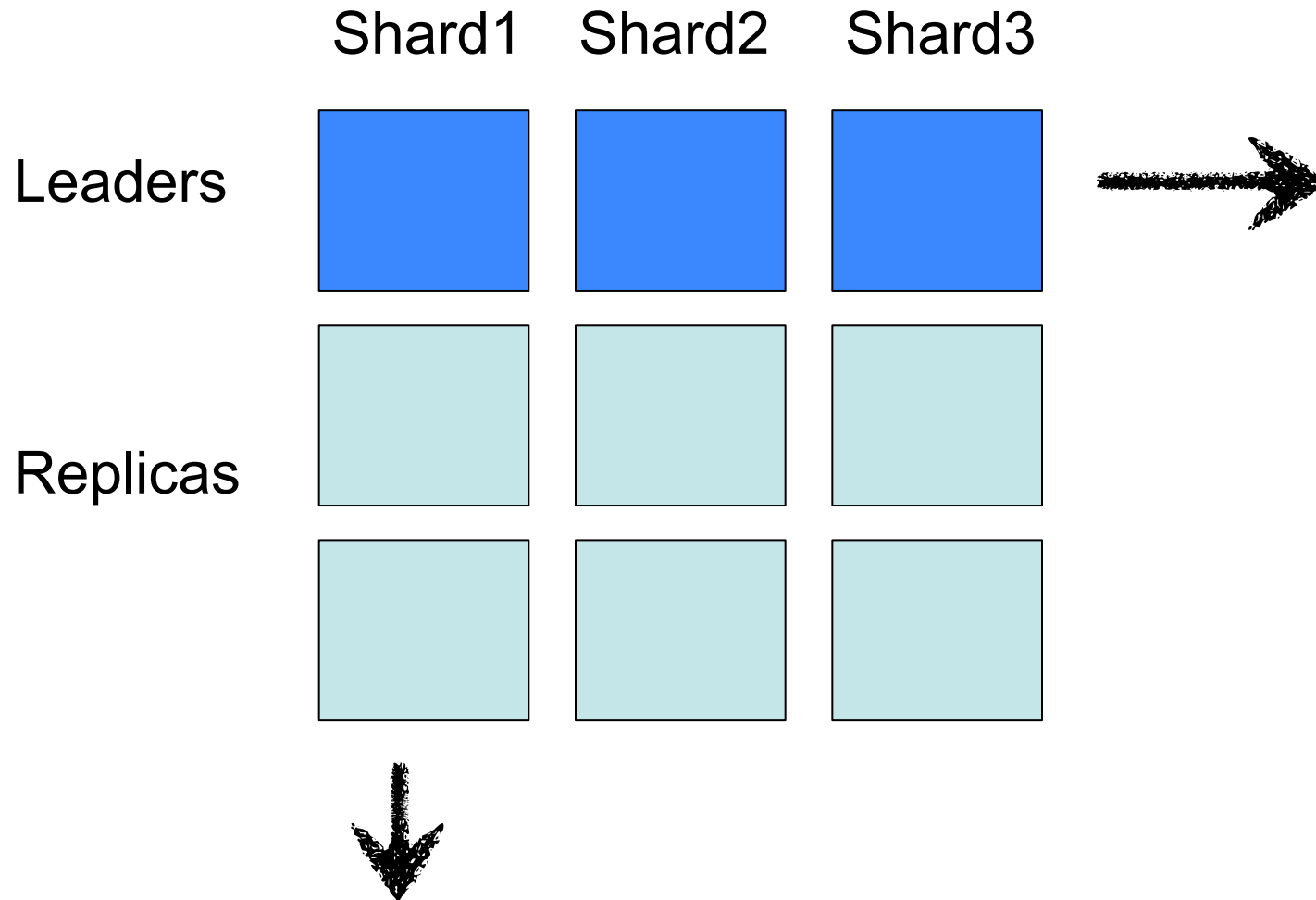
Leader Per Shard

- Use ZooKeeper for Leader election.
- Leader can easily version documents.
- Leader is the true record - recovery.
- If the Leader goes down, a replica takes it's place.
- Updates briefly block or are buffered with transaction log.

Auto Assign Shards

- Collection lock?
- Optimistic lock?
- Overseer?

Cluster



Wrap Up

- These are only a few features - there are many more already, and more coming.
- Pseudo Join, Pivot Faceting, Pseudo Fields, Per Segment Faceting, Configure Codec Per Field, Use Alternate Scoring Algorithms...
- When? Your guess is as good as mine ;) I hope sometime early next year.

Sources

- Links

- <http://wiki.apache.org/solr/SpellCheckComponent>
- <http://wiki.apache.org/solr/NearRealtimeSearch>
- <http://www.lucidimagination.com/blog/2011/09/07/realtime-get/>

Contact

- Mark Miller
 - markrmiller@gmail.com / mark.miller@lucidimagination.com
 - <http://www.lucidimagination.com/blog/author/markmiller/>
 - <http://twitter.com/heismark> 