Katta, an overview...

Stefan Groschupf
Scale Unlimited, 101tec.

sg{at}101tec.com

Katta, lucene in a grid

Agenda

- Problem
- Goals
- Solution
- Cons.
- History
- Overview
- What is a Katta Index?
- Communication, Master, Node and Client
- Distributed TF-IDF
- Different Network Topologies
- “Realtime” update?
- Hadoop integration
- CLI
- API
- Status
- Roadmap

Problem

- Lucene is great, but:
  - Slow if index is very big
  - Index bigger than on HDD
  - No load balance when there is high traffic
  - No failover

Goals

- Reliable index serving - by failover (master and nodes)
- Scalable for traffic and index size - by adding nodes
- Distributed TF-IDF
- Additions and deletes visible immediately*

Our Solution: Katta

- Serving indexes the hadoop distributed file system way
- Index as index shards on many servers
- Replicate shards on different servers for performance and fault-tolerance
- Lightweight
- Master fail over
- Fast*
- Easy to integrate
- Plays well with hadoop clusters
- Apache Version 2 License

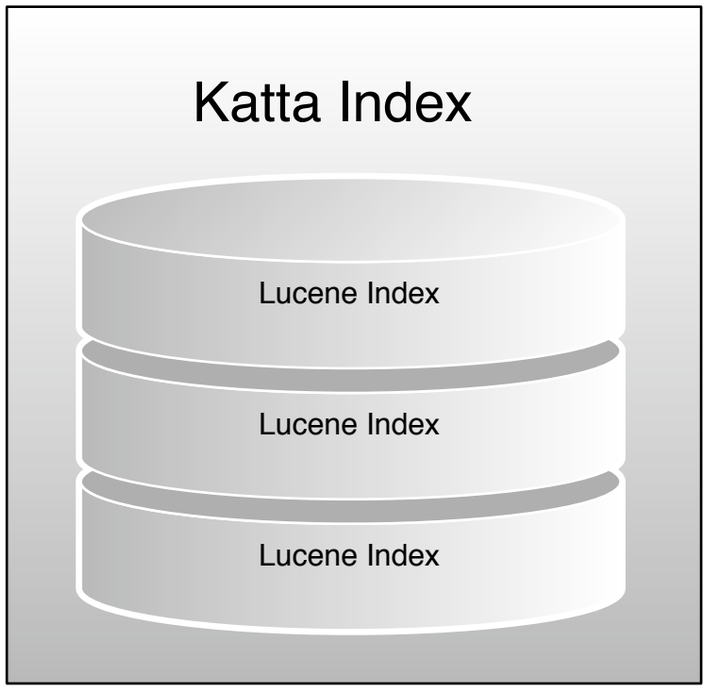
Cons.

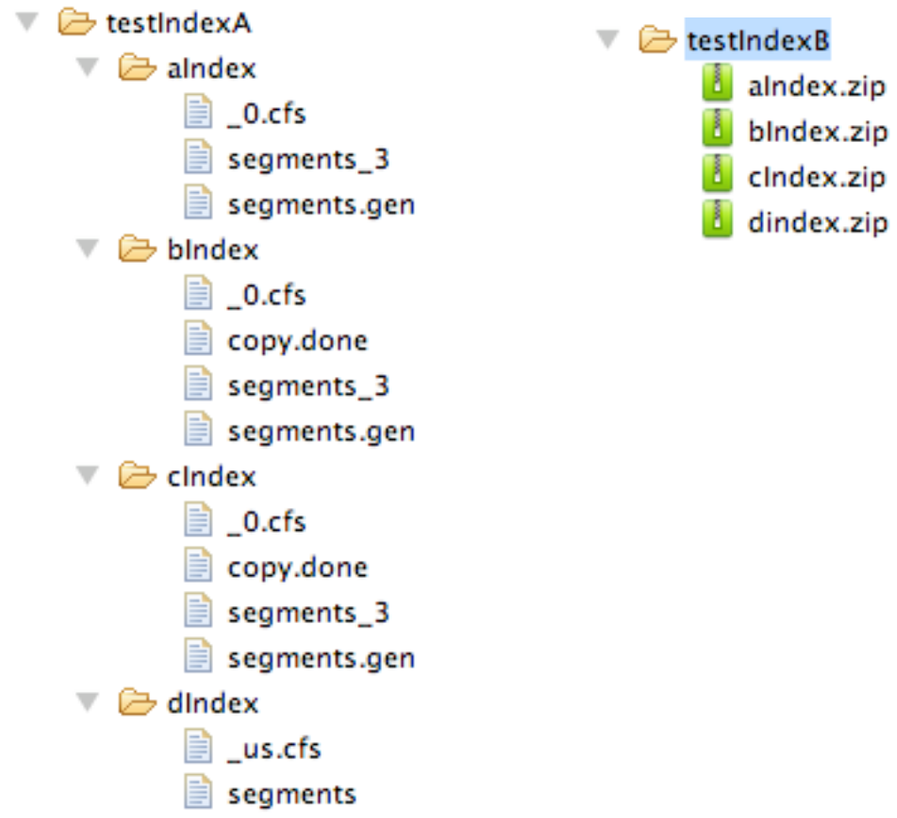
- No realtime updates like Solr, Couch DB or Cassandra yet*
  * though on roadmap
- Index serving tool, not indexer
- Early Stage* (Although it is currently in production)

History

- Founded April 2008
- Open source spinoff project from a telecom service company
- 4 developers
  - 2 out of 4 do paid development

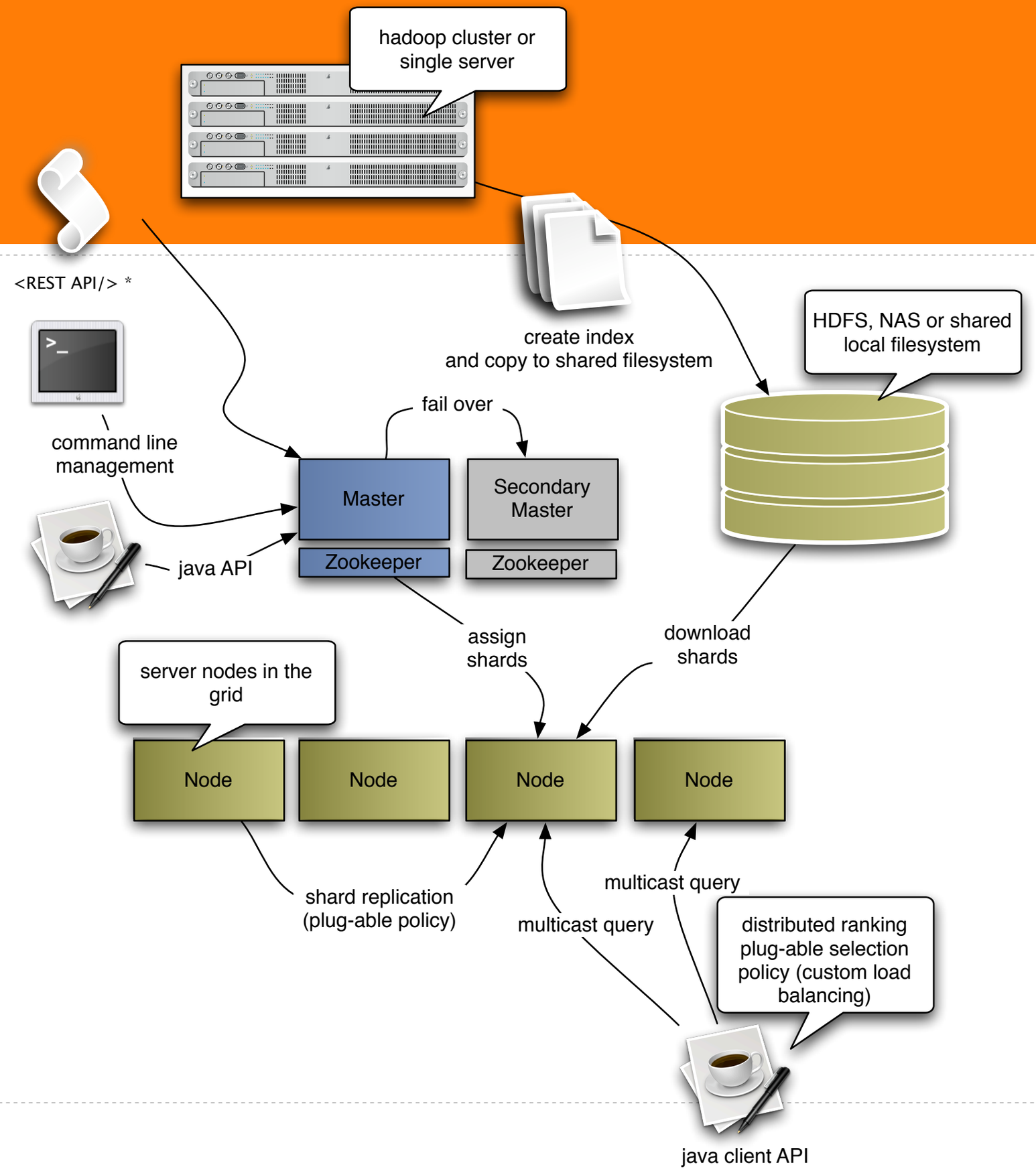
What is a Katta index?





- Folder with Lucene indexes
- Shard Indexes can be zipped

Overview


hadoop cluster or single server
<REST API/> *
create index and copy to shared filesystem
HDFS, NAS or shared local filesystem
command line management
fail over
Master
Secondary Master
Zookeeper
Zookeeper
java API
assign shards
download shards
server nodes in the grid
Node
Node
Node
Node
shard replication (plug-able policy)
multicast query
multicast query
distributed ranking plug-able selection policy (custom load balancing)
java client API

Communication

- Zookeeper for node communication
  - No heart beats
  - Failure detection
- Hadoop IPC for search
  - Light weight
  - Fast

Master

- Master
  - Failover to secondary (unlimited secondaries)
  - Assign shards to Nodes
  - Replicate - in case of Node failure
  - Plug-able - shard distribution policy
  - No State – All States are in Zookeeper

Node

- Node
  - Gets todos from zookeeper
  - Serves many index shards from different indexes (DataNode style)
  - Status updates written into zookeeper

Client

- Client
  - Get Shards and Node map from Zookeeper
  - Multicast Query to Nodes (one thread per node)
  - Plug-able node selection schemas (load balancing, low latency node selection, different racks, data center, etc.)
  - Sort results based on distributed score

Distributed TF-IDF

- Two network roundtrips necessary
  - Get document frequency for each node query
  - Query with document frequency
- Count method with one network roundtrip
- No distributed TF-IDF method planned (useful for pure filtering)

Different Netwok Topologies

- IDistributionPolicy
  - Custom implementation – Which node gets shard assigned?
  - Default implementation - even distribution
  - Should be maximal distance (Datacenter, Rack, Node)
- INodeSelectionPolicy
  - Which node is queried by the client?
  - Default implementation – random selection, round robin
  - Should be: closest nodes load balanced

Realtime update work around

- Index timestamp with each document
- Deploy small indexes that give updates frequently
- Dedup based on time stamp in the client
- Merge big and small indexes together
  - (we use a daily map reduce job)

Hadoop integration

- Create index with hadoop (Ning Li distributed indexing Contribution)
- Merge index with hadoop
- Store Index in HDFS or other hadoop file systems

CLI

```
grid@master:~/katta$ bin/katta
Usage:
        search <index name>[,<index name>,...] "<query>" [count]        Search in supplied indexes. The query should be in ". If you supply
 a result count hit details will be printed. To search in all indices write "*"
        listIndexes     Lists all indexes.
        listNodes       Lists all nodes.
        startMaster     Starts a local master.
        startNode       Starts a local node.
        showStructure   Shows the structure of a Katta installation.
        check   Analyze index/shard/node status.
        addIndex <index name> <path to index> <lucene analyzer class> [<replication level>]     Add a index to a Katta installation.
        removeIndex <index name>        Remove a index from a Katta installation.
        redeployIndex <index name>      Tries to deploy an index.
        listErrors <index name> Lists all deploy errors for a specified index.
grid@master:~/m2m/katta$
```

API

- Client
  - Client(INodeSelectionPolicy)
  - Client()
  - Client(INodeSelectionPolicy, ZkConfiguration)
  - search(IQuery, String[])
  - search(IQuery, String[], int)
  - getDetails(Hit)
  - getDetails(Hit, String[])
  - getQueryPerMinute()
  - count(IQuery, String[])
  - close()

Status

- First release candidate 09/17/08
- In Q&A of a 101tec customer
- **Contributors and sponsors needed :-)**

Roadmap

- Stability
- Release
- Performance
- Release
- Add realtime update support
  - Not yet clear *how* exactly
  - Might be similar to Dynamo

Thanks

katta.sourceforge.net

sg{at}101tec.com