Katta & Hadoop

Katta - Distributed Lucene Index in Production

Stefan Groschupf
Scale Unlimited, 101tec.

sg{at}101tec.com
Intro

- Business intelligence reports from event stream
- Existing event stream processing platform V1
  - Build on top of oracle
  - Scale problems
    - Expensive
    - Slow
    - High star schema
  - New report expensive to develop
  - Expensive to keep old data
Goals

- Build next generation platform for event stream processing
  - Faster report development - plugins
  - Reduce total coast of ownership
    - No license fees, open source based
    - Commodity hardware
    - Lower maintenance coasts
  - Better scalable
  - Better performance
  - Cheap storage
Challenge

- Integrate system into big picture
  - Log data via JMS
  - Report WebApp uses jdbc
  - Report developers do not know Map Reduce but SQL, XPath etc.
- Which format store data in?
- Which format process records in?
- Where store processing results in?
Challenges II

- *Teleskop to Microscope* - zoom to log record level
- One report - many mr jobs
- Job Scheduling
- Enterprise 24/7 monitoring - SNMP
- Work with open source releases cycles
Our Solution I

- **Monitor and manage everything**
  - Web - Console

- **Files by organized by day**
  - DFS
  - Convert logs to measures

- **Aggregate data and generate report data**
  - Pig

- **Distributed index for log message retrieval**
  - Katta

- **Store results of pig queries**
  - Database

- **Customer Userinterface**
  - Web Page

- **Convert logs to measures**
  - JMS MSG

- **Files organized by day**
  - JMS MSG
Our Solution II

Binary tree format

JMS

DFS

xml > tuples

MR

text tuples

PIG

SQL Schema

DB
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
Contreras

- No realtime updates like Solr, Couch DB or Cassandra yet*
  * though on roadmap
- Index serving tool, not indexer
What is a Katta index?

- Folder with Lucene indexes
- Shard Indexes can be zipped
Overview

- Hadoop cluster or single server
- Create index and copy to shared filesystem
- Master Node
- Node
- Secondary Master Node
- Zookeeper
- Assign shards
- Download shards
- Fail over
- SHARDS, NAS or shared local filesystem
- Command line management
- Java API
- Server nodes in the grid
- Shard replication (plug-able policy)
- Multicast query
- Distributed ranking plug-able selection policy (custom load balancing)
- Java client API
- REST API
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()`
Lucene Queries

- title: "The Right Way" AND text: go
- test or test* or test
- mod_date: [20020101 TO 20030101]
- state: CA AND age: [1 TO 15] AND product: ipod
- state: CA AND age: [16 TO 21] AND product: ipod
Teleskop to Microscope

- Create Index from XML in MR stage
- Deploy indexes in katta
- Merge indexes frequently together
- Find documents by key
- Find documents by query
XML to Lucene Document

<event id="aKey" type="sell">
  <product id="ipod"/>
  <user id="stefan" state="CA" age="31"/>
</event>

/event/@id:aKey
/event/@type:sell
/event/product/@id:ipod
/event/user/@id:stefan
/event/user/@state:CA
/event/user/@ age:31
Range Queries

/event/product/@id:ipod AND /event/user/@state:CA AND /event/user/@ age:[001 TO 010]
/event/product/@id:ipod AND /event/user/@state:CA AND /event/user/@ age:[011 TO 020]
/event/product/@id:ipod AND /event/user/@state:CA AND /event/user/@ age:[021 TO 030]
/event/product/@id:ipod AND /event/user/@state:CA AND /event/user/@ age:[031 TO 040]

Counting results -> one network round trip
Range Queries Result Graph
Pros

• Easy reports can be generated from katta index
• Complex reports generated with many pig statements (>30 job)
• Zoom in data from complex reports

System scales
• Scaling is cheap
• We keep more data
• Report developing is easy
Problems

- There was no cascading, hive or jaql, pig was very young
- Develop against changing open source project (hadoop, pig)
- Pig is/was slow (always text) and (was) buggy
- Katta indexes need to merged frequently
- Monitoring and management
Roadmap

- 0.1 released
- 0.2 Hadoop 0.17
- 0.3 Hadoop 0.18
- Performance improvements
- EC2 support
- Add realtime update support
  - Not yet clear how exactly
  - Might be similar to Dynamo
Thanks

katta.sourceforge.net

sg{at}101tec.com