Using Hadoop/MapReduce with Solr/Lucene for Large Scale Distributed Search

简报单位：精诚资讯(SYSTEX Corp)云中心/Big Data事业
简报时间：2011.12.2 Hadoop in China 2011
简报人　：陈昭宇(James Chen)

SYSTEX
making it happen

About Myself

陈昭宇 (James Chen)

- 精诚资讯云中心Big Data事业首席顾问
- 十年网络资安产品研发经验,专长Email & Web Security
- 5年Hadoop相关技术实务经验，曾经参与多项海量数据平台的架构设计与开
  - 海量Web日志处理
  - 垃圾邮件采集及分析
  - 电信行业CDR帐务分析
  - 电信加值服务用户行为分析
  - 智能电网数据采集处理与分析
  - 在线服务推荐引擎与电子商务应用
- 长期关注Hadoop ecosystem与Solr/Lucene等开源软件的发展
- Cloudera Certified Developer for Apache Hadoop (CCDH)。

关于精诠Big Data事业

精诠Big Data事业团队以北京为基地，以发展中国的Big Data Killer App为使命。奠基于全球的Hadoop Ecosystem，与所有的Hadoop Distributions、Tools厂商友好，致力于海量数据的End-to-End解决方案，直接面对行业，直接解决特定的商业问题，凸显团队的专业价值。

团队中有亚洲少见的多年Hadoop实战经验成员，在商业Production环境中运行过数百到上万个节点的服务丛集，并有数位拥有Cloudera Certified Developer/Administrator for Apache Hadoop的专业认证。

Agenda

- Why search on big data ?
- Lucene, Solr, ZooKeeper
- Distributed Searching
- Distributed Indexing with Hadoop
- Case Study: Web Log Categorization

Why Search on Big Data ?

- Structured & Unstructured
- No Schema Limitation
- Ad-hoc Analysis
- Keep Everything

A java library for search

- Mature, feature riched and high performance
- indexing, searching, highlighting, spell checking, etc.

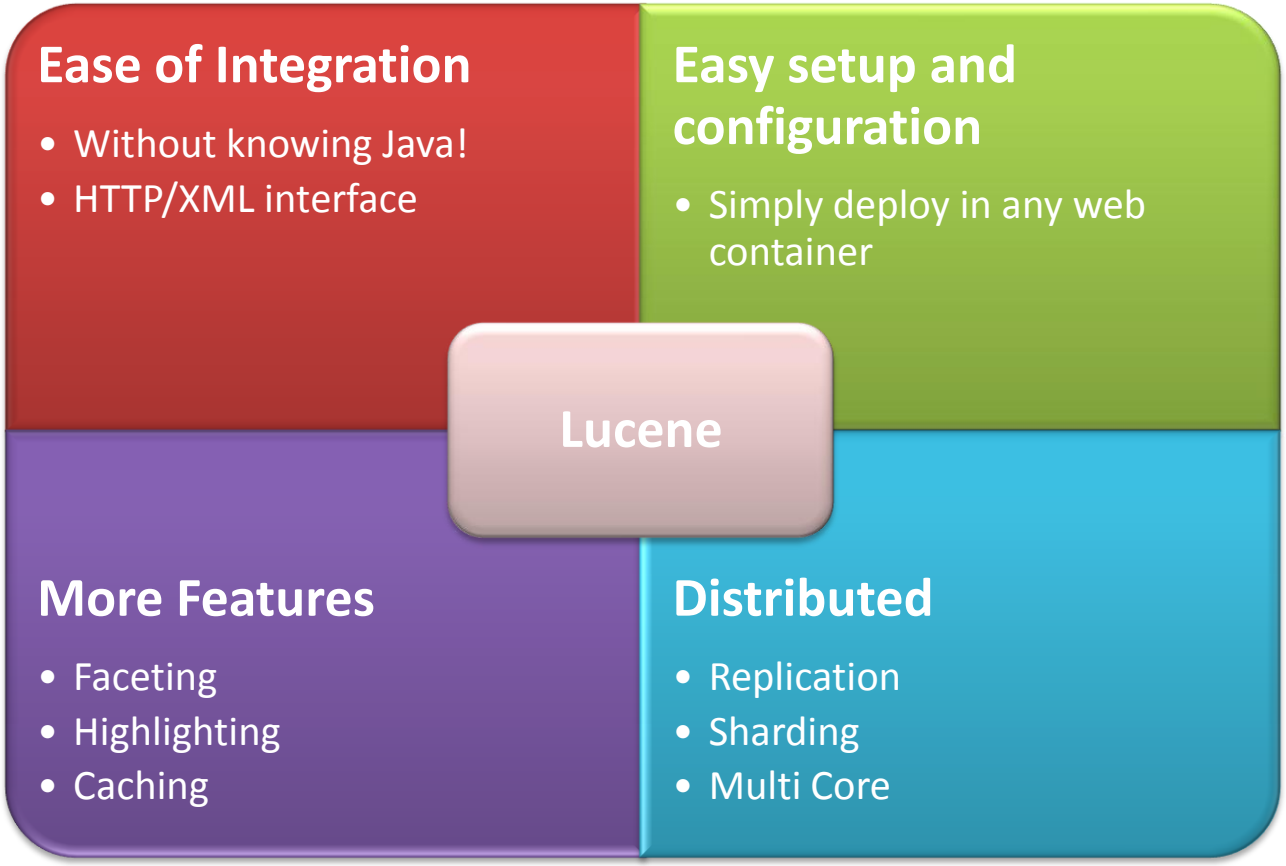
Not a crawler

- Doesn’t know anything about Adobe PDF, MS Word, etc.

Not fully distributed

- Does not support distributed index

Apache Solr™


Ease of Integration
• Without knowing Java!
• HTTP/XML interface
Easy setup and configuration
• Simply deploy in any web container
Lucene
More Features
• Faceting
• Highlighting
• Caching
Distributed
• Replication
• Sharding
• Multi Core

Lucene Default Query Syntax

Lucene Query Syntax [; sort specification]

1. mission impossible; releaseDate desc
2. +mission +impossible –actor:cruise
3. “mission impossible” –actor:cruise
4. title:spiderman^10 description:spiderman
5. description: “spiderman movie” ~10
6. +HDTV +weight:[0 TO 100]
7. Wildcard queries: te?t, te*t, test*

for more information:
http://lucene.apache.org/java/3_4_0/queryparsersyntax.html

Solr Default Parameters

Query Arguments for HTTP GET/POST to /select

param	default	description
q		The query
start	0	Offset into the list of matches
rows	10	Number of documents to return
fl	*	Stored fields to return
qt	standard	Query type; maps to query handler
df	(schema)	Default field to search

Solr Search Results Example

http://localhost:8983/solr/select?q=video&start=0&rows=2&fl=name,price

```
<response><responseHeader><status>0</status>
  <QTime>1</QTime></responseHeader>
  <result numFound="16173" start="0">
    <doc>
      <str name="name">Apple 60 GB iPod with Video</str>
      <float name="price">399.0</float>
    </doc>
    <doc>
      <str name="name">ASUS Extreme
    N7800GTX/2DHTV</str>
      <float name="price">479.95</float>
    </doc>
  </result>
</response>
```

Zookeeper

Configuration

- Sharing configuration across nodes
- Maintaining status about shards

Group Service

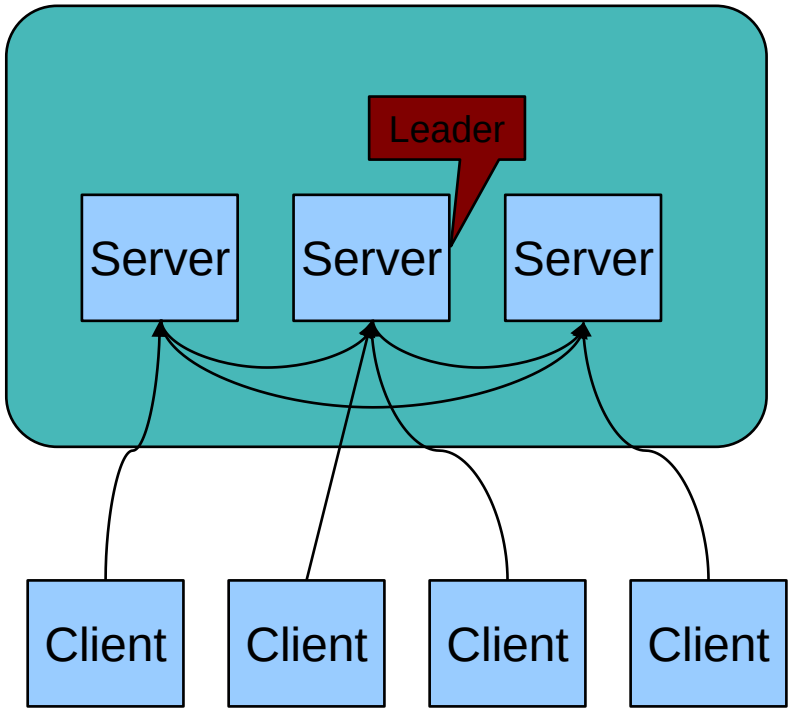
- Master/Leader Election
- Central Management
- Replication
- Name Space

Synchronization

- Coordinating searches across shards/load balancing

Coordination

Zookeeper Architecture



- All servers store a copy of the data (in memory)
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

Distributed Search

Scalability

- Shard Management
- Sharding Algorithm

Performance

- Query Distribution
- Load Balancing

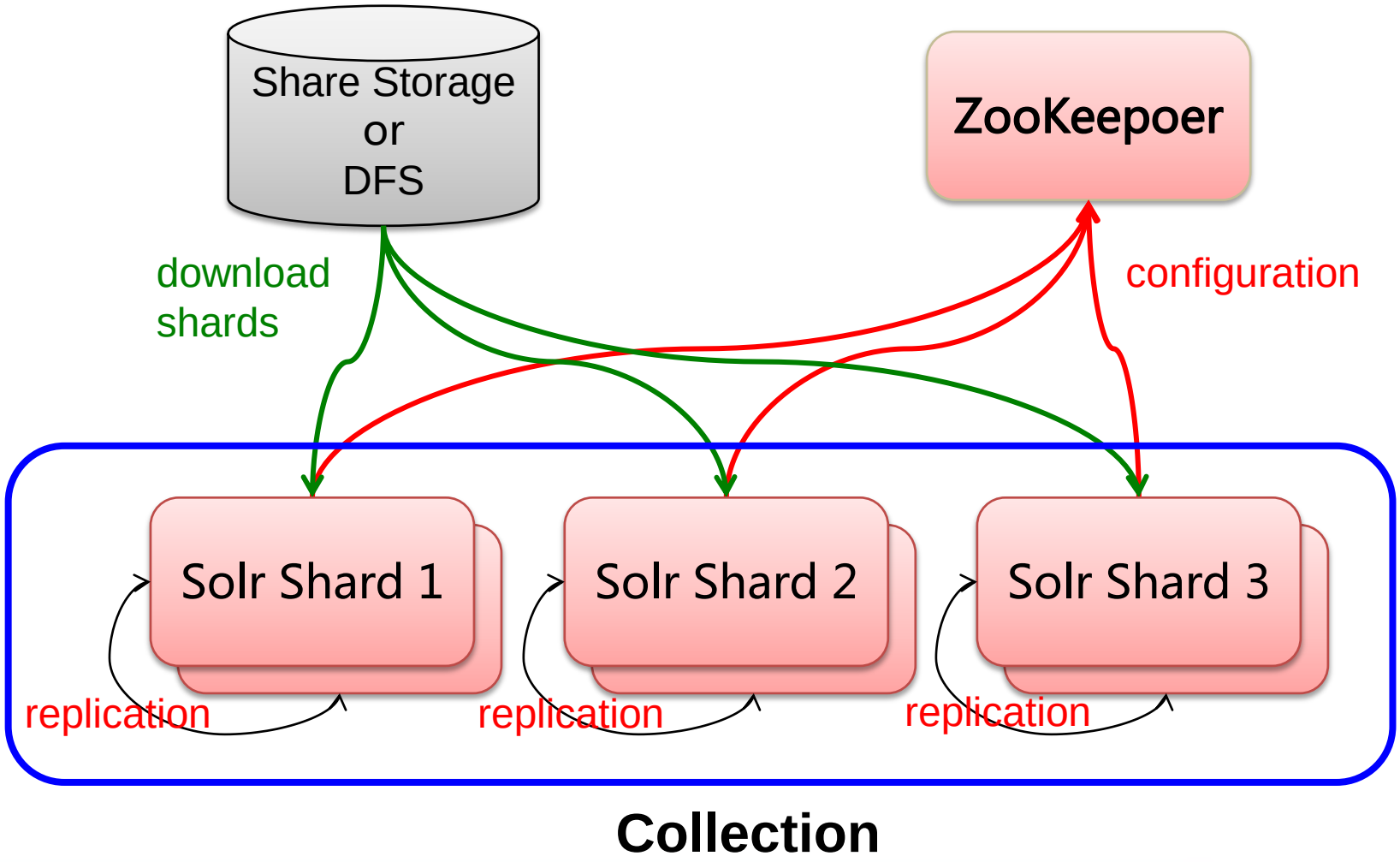
Availability

- Replication
- Server Fail-Over

SolrCloud – Distributed Solr

Cluster Management	• Central configuration for the entire cluster • Cluster state and layout stored in central system
Zookeeper Integration	• Live node tracking • Central Repository for configuration and state
Remove external load balancing	• Automatic fail-over and load balancing
Client don’t need to know shard at all	• Any node can serve any request

SolrCloud Architecture



Start a SolrCloud node

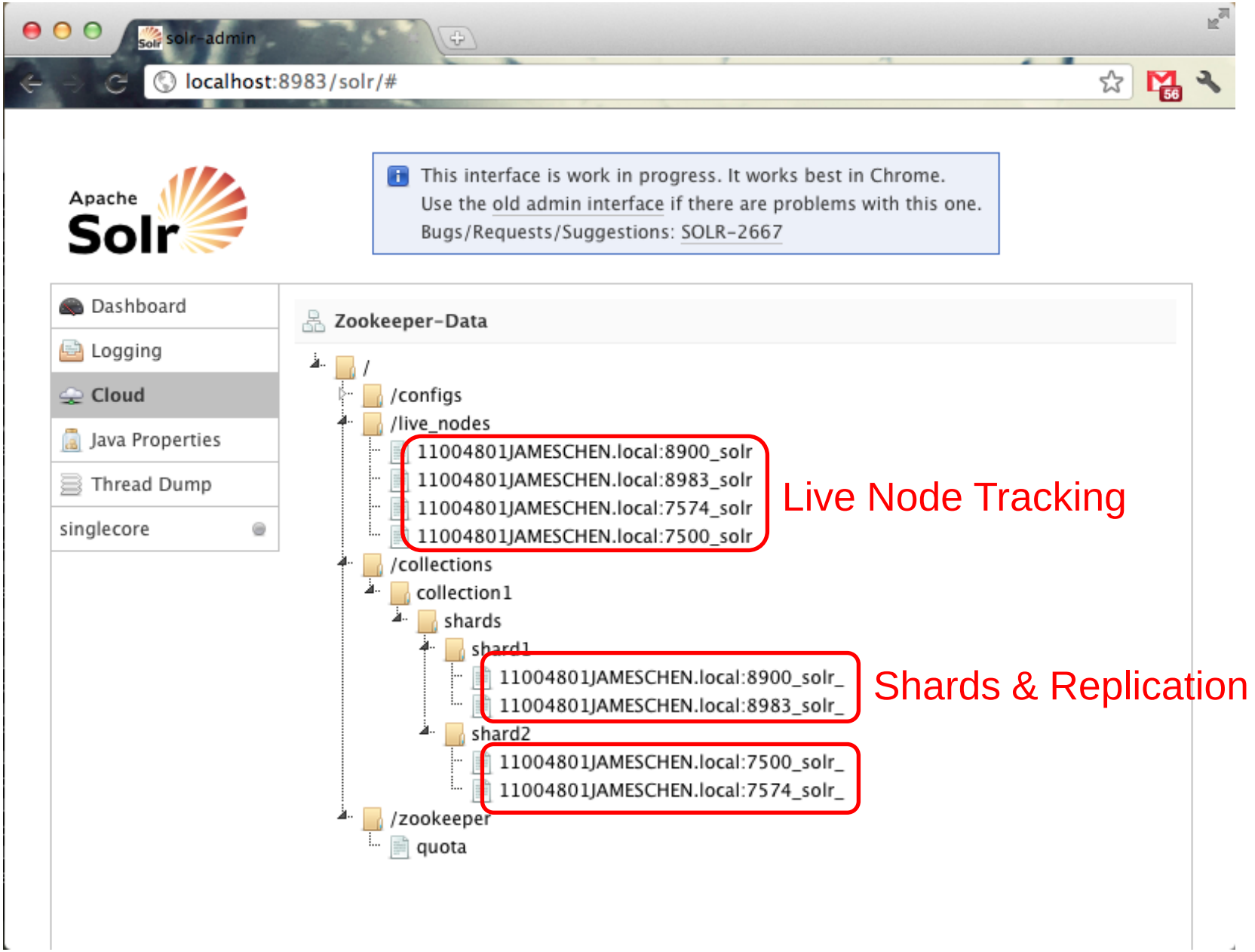
solr.xml

```
<solr persistent="false">

  <!--
  adminPath: RequestHandler path to manage cores.
    If 'null' (or absent), cores will not be manageable via request handler
  -->
  <cores adminPath="/admin/cores" defaultCoreName="collection1">
    <core name="collection1" instanceDir="." shard="shard1"/>
  </cores>
</solr>
```

start solr instance to serve a shard

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



Distributed Requests of SolrCloud

Explicitly specify the address of shards

- shards=localhost:8983/solr,localhost:7574/solr

Load balance and fail over

- shards=localhost:8983/solr | localhost:8900/solr,localhost:7574/solr | localhost:7500/solr

Query all shards of a collection

- http://localhost:8983/solr/collection1/select?distrib=true

Query specific shard ids

- http://localhost:8983/solr/collection1/select?shards=shard_1,shard_2&distrib=true

More about SolrCloud

Still under development. Not in official release

Get SolrCloud:

https://svn.apache.org/repos/asf/lucene/dev/trunk

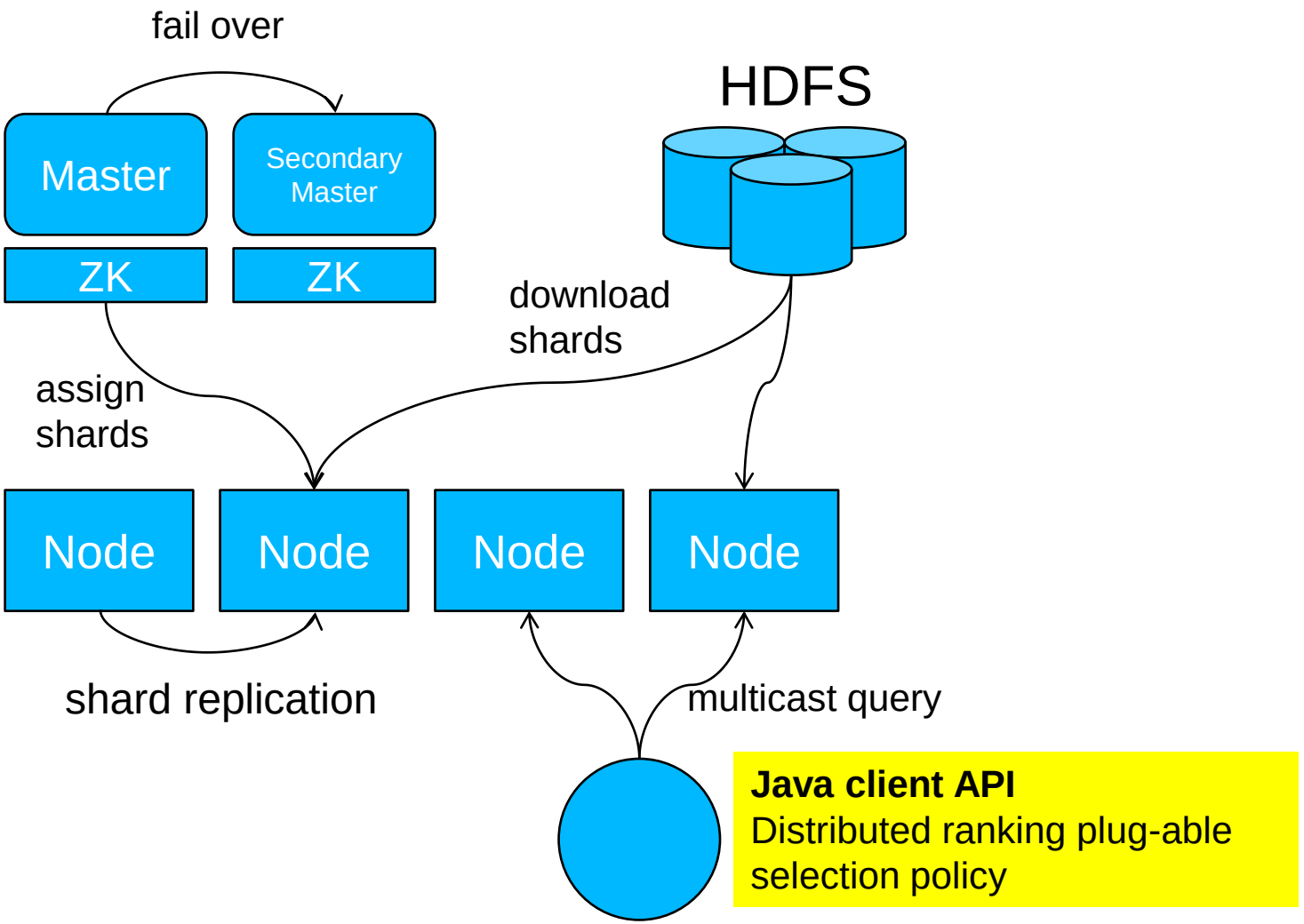
Wiki:

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

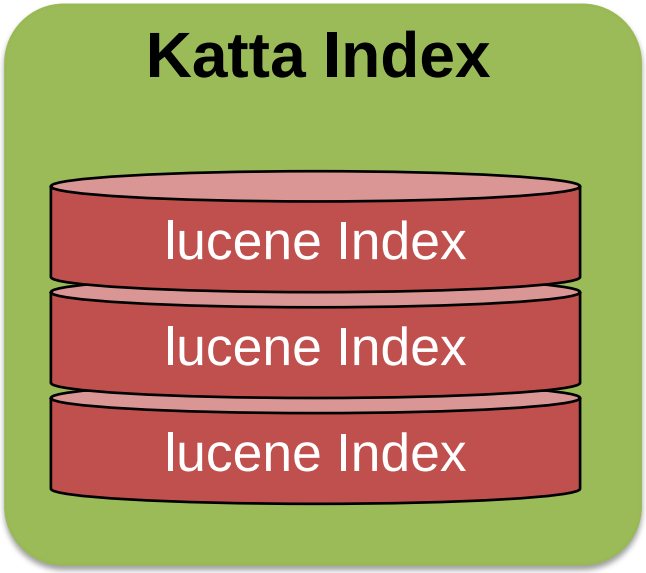
Katta – Distributed Lucene

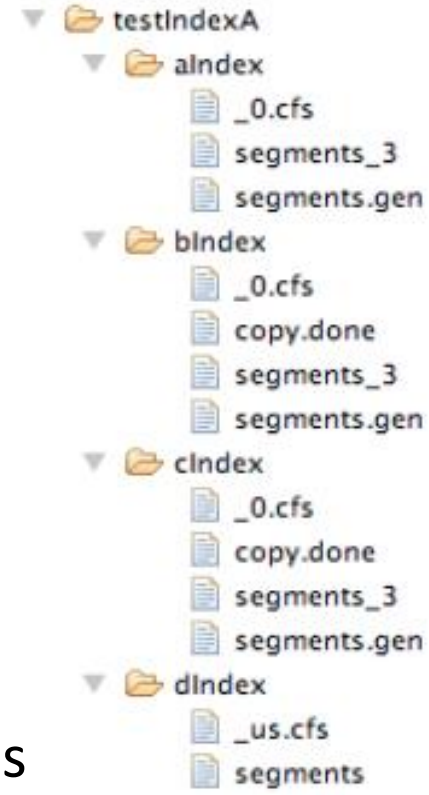
- Master Slave Architecture
- Master Fail-Over
- Index & Shard Management
- Replication
- Fault Tolerance

Katta Architecture



Katta Index





- Folder with Lucene indexes
- Shard Index can be zipped

The different between Katta and SolrCloud

- Katta is based on lucene not solr
- Katta is master-slave architecture. Master provide management command and API
- Katta will automatically assign shards and replication for katta nodes.
- Client should use Katta client API for querying.
- Katta client is Java API, require java programming.
- Katta support read index from HDFS. Plays well with Hadoop

Configure Katta

conf/masters

- Required on **Master**

```
server0
```

conf/nodes

- Required fo **all nodes**

```
server1
server2
server3
```

conf/katta.zk.properties

- zk can be embedded with master or standalone
- Standalone zk is required for master fail-over

```
zookeeper.server=<server0>:2181
Zookeeper.embedded=true
```

conf/katta-env.sh

- JAVA_HOME required
- KATTA_MASTER optional

```
export JAVA_HOME=/usr/lib/j2sdk1.5-sun
Export KATTA_MASTER=server0:/home/$USER/katta-distribution
```

Katta CLI

Search

- **search** <index name>[,<index name>,…] “<query>” [count]

Cluster Management

- **listIndexes**
- **listNodes**
- **startMaster**
- **startNode**
- **showStructure**

Index Management

- **check**
- **addIndex** <index name> <path to index> <lucene analyzer class> [<replication level>]
- **removeIndex** <index name>
- **redeployIndex** <index name>
- **listErrors** <index name>

Deploy Katta Index

Deploying Katta index on local filesystem

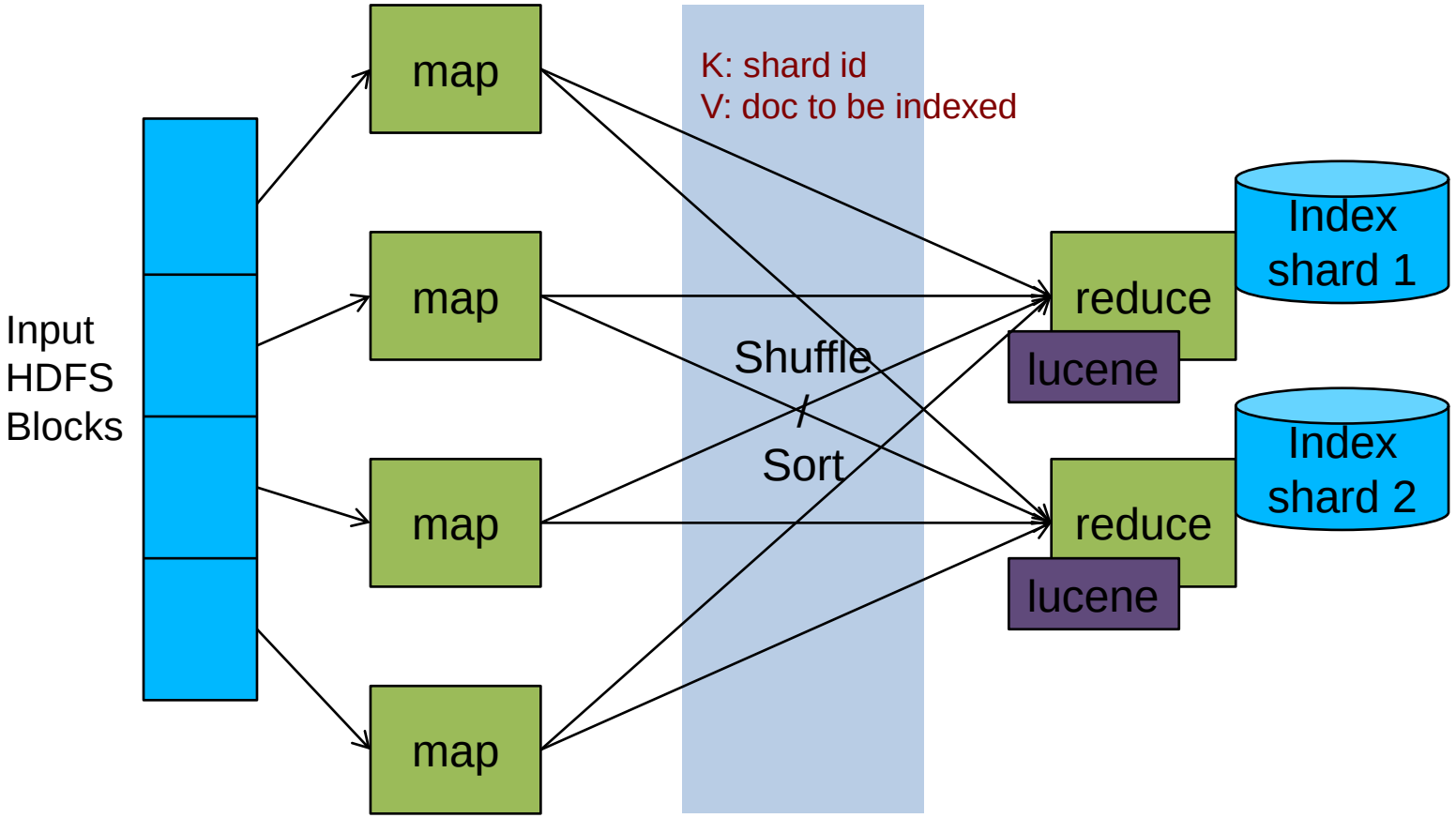
```
bin/katta addIndex <name of index> \
        [file:///<path to index] <rep lv>
```

Deploying Katta index on HDFS

```
bin/katta addIndex <name of index> \
        [hdfs://namenode:9000/path] <rep lv>
```

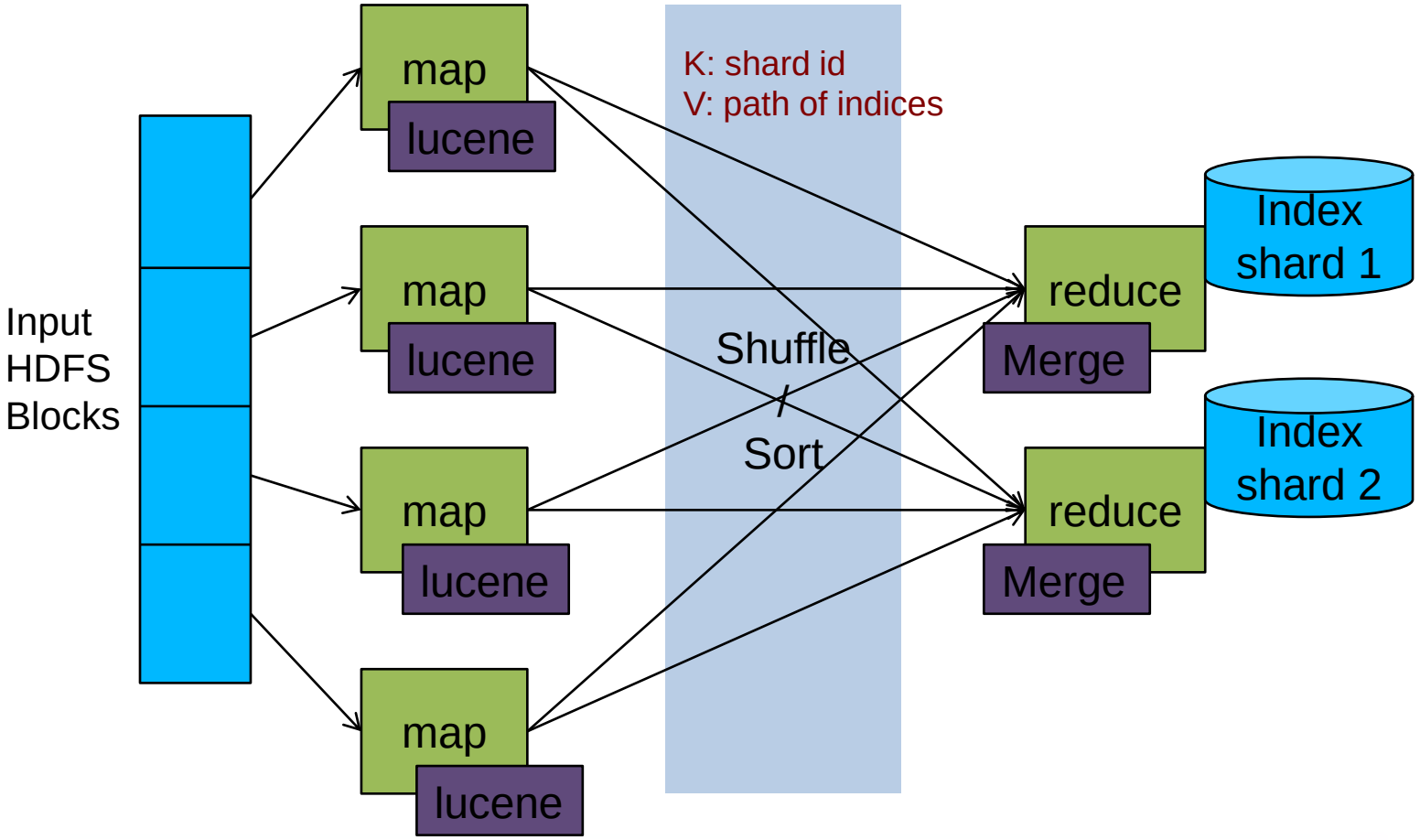
Building Lucene Index by MapReduce

Reduce Side Indexing



Building Lucene Index by MapReduce

Map Side Indexing



Mapper

```
public class IndexMapper extends Mapper<Object, Text, Text, Text> {
    private String [] shardList;
    int k=0;

    protected void setup(Context context) {
            shardList=context.getConfiguration().getStrings("shardList");
    }

    public void map(Object key, Text value, Context context)
            throws IOException, InterruptedException
      {
       context.write(new Text(shardList[k++]), value);
        k %= shardList.length;
    }
```

Reducer

```
public class IndexReducer extends Reducer<Text, Text, Text, IntWritable> {
    Configuration conf;
    public void reduce(Text key, Iterable<Text> values, Context context)
            throws Exception
    {
            conf = context.getConfiguration();
            String indexName = conf.get("indexName");
            String shard = key.toString();
            writeIndex(indexName, shard, values);
    }
    private void writeIndex(String indexName, String shard, Iterable<Text> values)
    {
            Iterator<Text> it=values.iterator();
            while(it.hasNext()){
                //...index using lucene index writer...
            }
    }
}
```

use lucene here !

Sharding Algorithm

- Good document distribution across shards is important
- Simple approach:
  - hash(id) % numShards
  - Fine if number of shards doesn't change or easy to reindex
- Better:
  - Consistent Hashing
    - http://en.wikipedia.org/wiki/Consistent_hashing

Case Study – Web Log Categorization

Customer Background

- Tire-1 telco in APAC
- Daily web log ~ 20TB
- Marketing guys would like to understand more about browsing behavior of their customers

Requirements

- The logs collected from the network should be input through a “big data” solution.
- Provide dashboard of URL category vs. hit count
- Perform analysis based on the required attributes and algorithms

Hardware Spec

- 1 Master Node (Name Node, Job Tracker, Katta Master)
- 64 TB raw storage size (8 Data Node)
- 64 cores (8 Data Node)
- 8 Task Tracker
- 3 katta nodes
- 5 zookeeper nodes

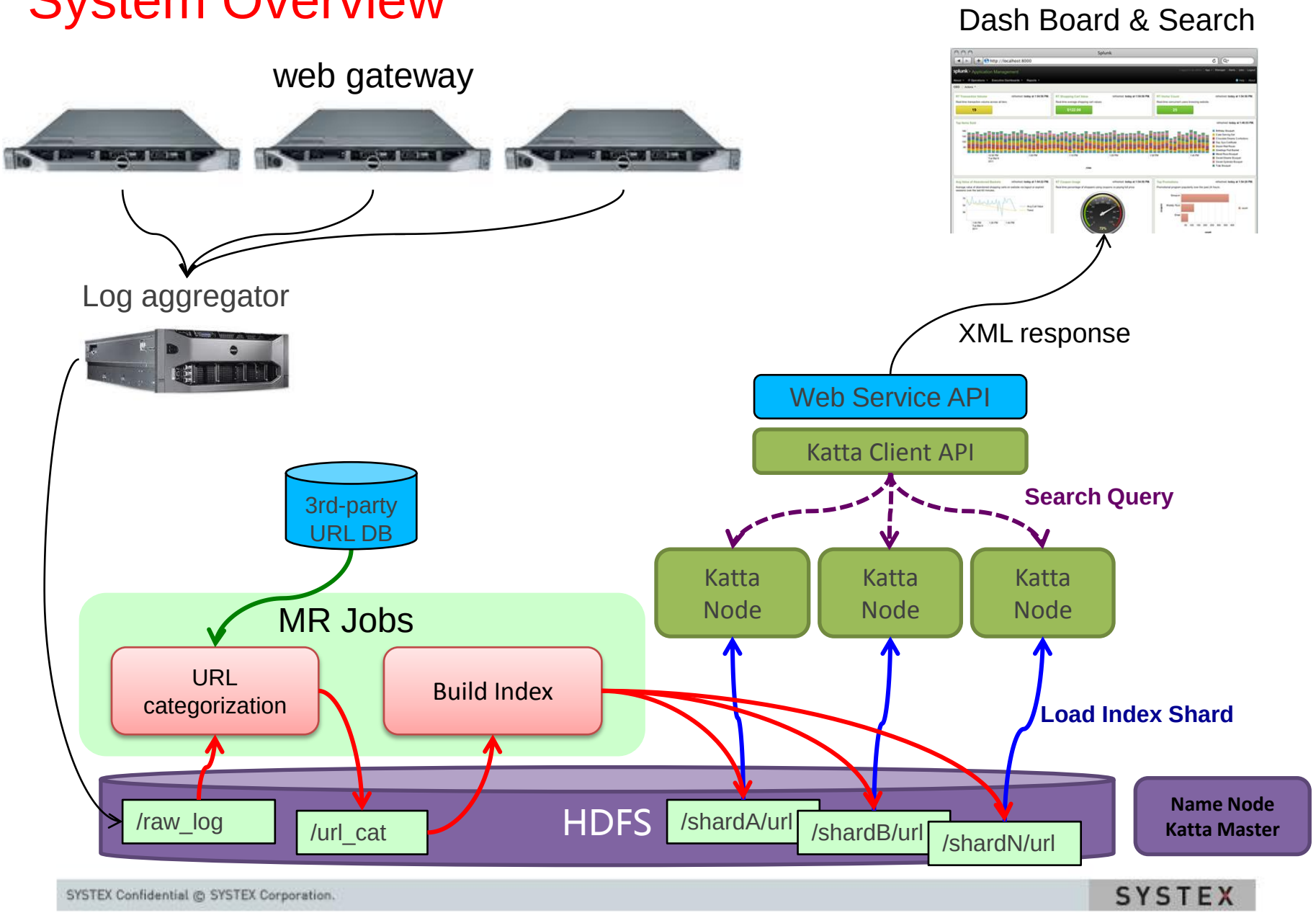
Master Node

- CPU: Intel® Xeon® Processor E5620 2.4G
- RAM: 32G ECC-Registered
- HDD: SAS 300G @ 15000 rpm (RAID-1)

Data Node

- CPU: Intel® Xeon® Processor E5620 2.4G
- RAM: 32G ECC-Registered
- HDD: SATA 2Tx4, Total 8T

System Overview



Thank You
&
We are hiring !!

mailto: jameschen@systex.com.tw