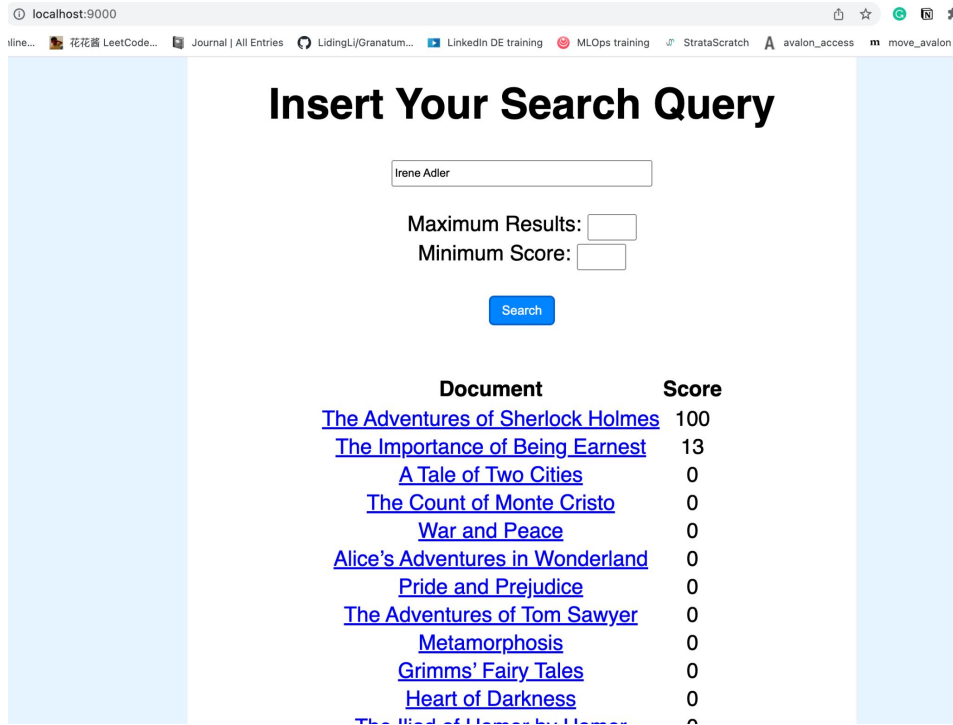


# System Design of Distributed Search System in Java

*(project and notes from Udemy course)*

copyright ©HarrisonLL

# 1. Demo



localhost:9000

Journal | All Entries

## Insert Your Search Query

Irene Adler

Maximum Results:

Minimum Score:

Search

Document	Score
<a href="#">The Adventures of Sherlock Holmes</a>	100
<a href="#">The Importance of Being Earnest</a>	13
<a href="#">A Tale of Two Cities</a>	0
<a href="#">The Count of Monte Cristo</a>	0
<a href="#">War and Peace</a>	0
<a href="#">Alice's Adventures in Wonderland</a>	0
<a href="#">Pride and Prejudice</a>	0
<a href="#">The Adventures of Tom Sawyer</a>	0
<a href="#">Metamorphosis</a>	0
<a href="#">Grimms' Fairy Tales</a>	0
<a href="#">Heart of Darkness</a>	0
<a href="#">The Wind of Heaven by Heaven</a>	0

Frontend server (port 9000)

```
(base) harrisonli@Harrissons-MacBook-Pro-2 DistributedSearch % java -jar target/distributed_search-1.0-SNAPSHOT-jar-with-dependencies.jar 8080
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successfully connected to Zookeeper
znode name /election/c_0000000017
I am the leader
The cluster addresses are: []
Registered to service registry
The cluster addresses are: [http://localhost:8081/task]
The cluster addresses are: [http://localhost:8081/task, http://localhost:8082/task]
The cluster addresses are: [http://localhost:8081/task, http://localhost:8082/task, http://localhost:8083/task]
Received search query: sherlock
Received 3/3 results
Calculating score for all the documents
Received search query: alice
Received 3/3 results
Calculating score for all the documents
Received search query: Irene Adler
Received 3/3 results
Calculating score for all the documents
```

```
(base) harrisonli@Harrissons-MacBook-Pro-2 DistributedSearch % pwd
/Users/harrisonli/IdeaProjects/DistributedSearch
(base) harrisonli@Harrissons-MacBook-Pro-2 DistributedSearch % java -jar target/distributed_search-1.0-SNAPSHOT-jar-with-dependencies.jar 8081
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successfully connected to Zookeeper
znode name /election/c_0000000018
I am not the leader
Registered to service registry
Watching znode c_0000000017
```

```
Received 7 documents to process
Received 7 documents to process
Received 7 documents to process
```

```
...rch-1.0-SNAPSHOT-jar-with-dependencies.jar 8082 ...cts/distributed-systems-leader-election --zah ...
(base) harrisonli@Harrissons-MacBook-Pro-2 bin % cd /Users/harrisonli/IdeaProjects/DistributedSearch
(base) harrisonli@Harrissons-MacBook-Pro-2 DistributedSearch % java -jar target/distributed_search-1.0-SNAPSHOT-jar-with-dependencies.jar 8082
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successfully connected to Zookeeper
znode name /election/c_0000000019
I am not the leader
Registered to service registry
Watching znode c_0000000018

Received 7 documents to process
Received 7 documents to process
Received 7 documents to process
```

```
(base) harrisonli@Harrissons-MacBook-Pro-2 distributed-systems-leader-election % cd /Users/harrisonli/IdeaProjects/DistributedSearch
(base) harrisonli@Harrissons-MacBook-Pro-2 DistributedSearch % java -jar target/distributed_search-1.0-SNAPSHOT-jar-with-dependencies.jar 8083
log4j:WARN No appenders could be found for logger (org.apache.zookeeper.ZooKeeper).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Successfully connected to Zookeeper
znode name /election/c_0000000020
I am not the leader
Registered to service registry
Watching znode c_0000000019

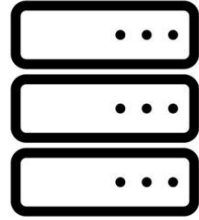
Received 6 documents to process
Received 6 documents to process
Received 6 documents to process
```

## Backend Server:

1 master node on port 8080, 3 worker nodes on port 8081, 8082, 8083

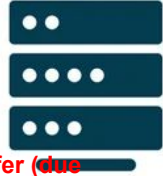
# 2. System Design

Frontend Server:  
Web based UI in JS & html  
User types search input

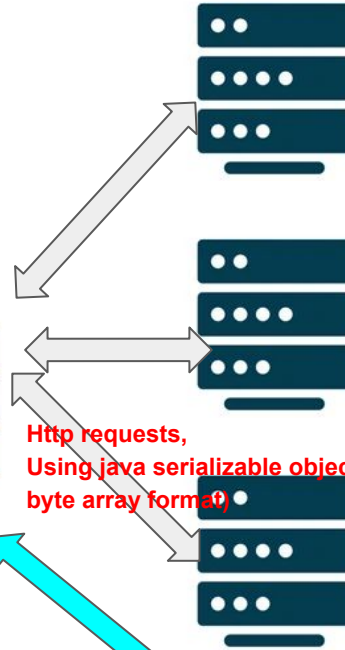


Http requests,  
Using Protocol Buffer (due  
to language compatibility)

Master node:  
Split computing task by  
documents.  
  
Aggregate to compute  
inverse document  
frequency and sort the  
score upon receiving  
results from workers



Http requests,  
Using java serializable object (in  
byte array format)



Worker nodes:  
Compute term frequency of passed  
documents for all search terms

Read from document repository  
(File System)



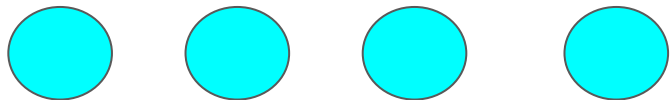
## ZooKeeper Level

- Store all registered node address
- Reelect Leader once a master node failed

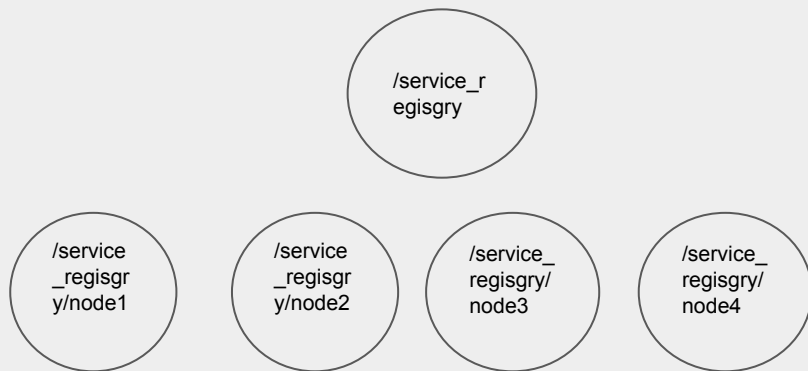
Coordinators(Ma  
ster) Service  
Registry

Workers  
Service  
Registry

# ZooKeeper Service Registry



*Actual Node (communicating thru `getChildren`, `getData`  
`NodeChildrenChanged` to `Znode`)*



*Permanent Znode (upon reconnection, node keeps all data )*

*Ephemeral Znode (upon reconnection, node is deleted)*

*Node stores address info*

### 3. Side Notes

## Master-Worker architecture: Leader Reelection Algorithm

Once a master node fails, the service registry must reelect a new leader.

If all nodes register a watcher process to listen to all the other nodes, the cost will be huge.

Instead, we have a cost efficient way by registering a watcher for each node to only listen to its predecessor. This way we only need to maintain a sorted list, once new node joins or the failed node rejoins the system, increase its index.

The algorithm goes:

...

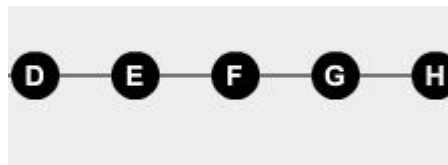
sort zookeeper node by index  
if the node being watched failed is leader:

    Its watcher becomes the new leader

else:

    Find the node's predecessor and register a new watcher

...



### 3. Side Notes

#### Some thoughts on the project

- In real case, the master node and worker nodes are usually machines, rather than a process. This system design simulates the situation where servers communicates through http.
- It is possible that the file system locates at another physical server or any cloud server. The communication would not be simple as retrieving files in local computer. Http request should be implemented.
- In terms of scaling of service, it is possible that numbers of users using the service at the same time, then the design needs load balancer between 1) user and frontend service, and 2) frontend service and backend service.

## 3. Side Notes

# Network Communication Choices

- Json is most common
  - Human readable but no strict schema
  - Msg is in plain text, so its network overhead is larger than binary array
- Protocol Buffer is another (Google)
  - The benefit is speed and easy communication between different programming languages
- Serializing Object
  - Java Serializable object: Serializable interface
  - Python: pickle package
  - ...
  - The benefit is having a smaller overhead and clear schema