Log Query Interface

CS 211 Course Project




Sandeep Singh Sandha **Yue Xin** **Zhehan Li** **Xin Xu**

Mentors: Yuanjie Li & Prof. Songwu Lu

Problem Statement

“Web framework to conveniently and efficiently query very large datasets of MobileInsight.”

Challenges

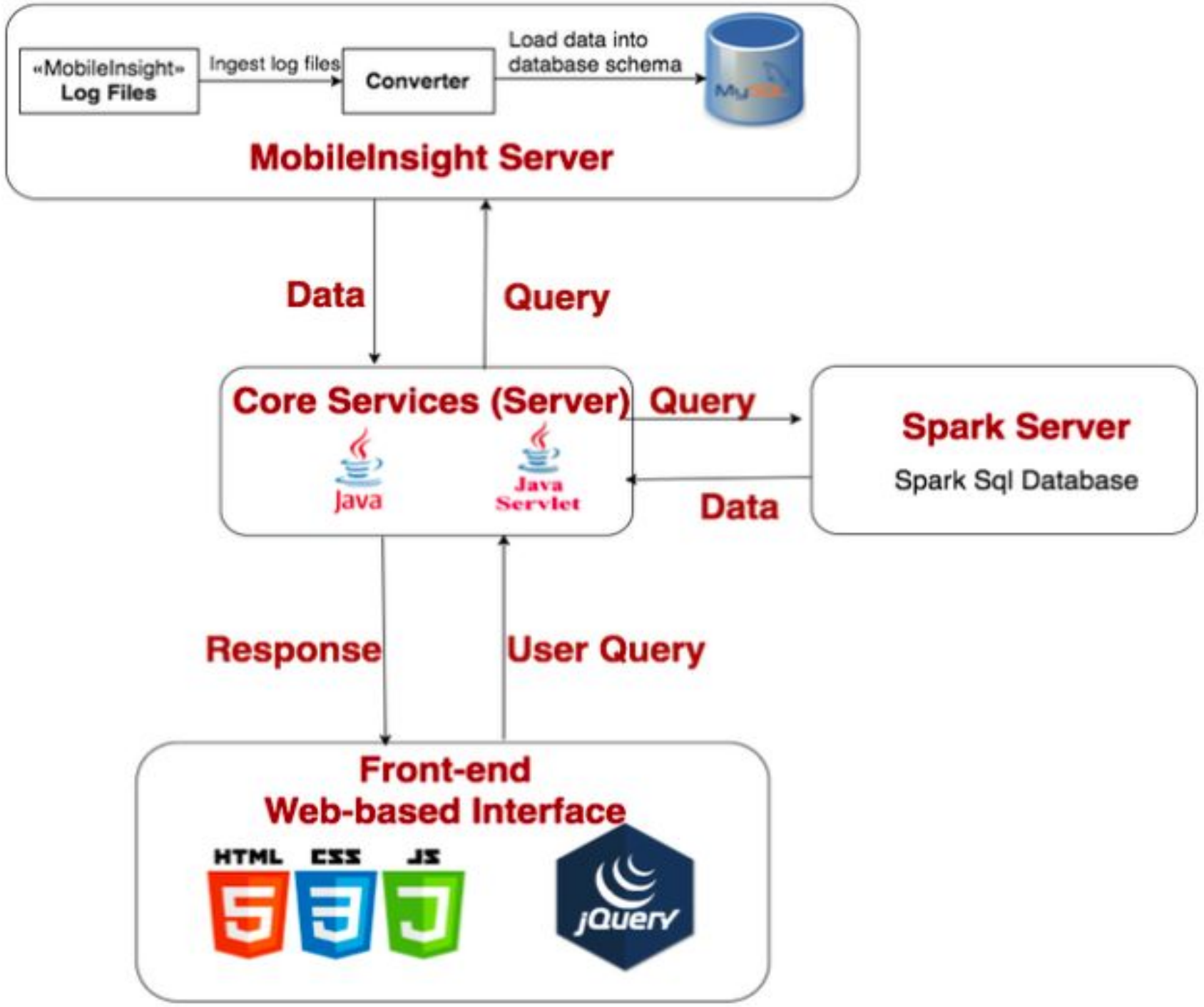
- ★ Big Data Storage
- ★ Scalability
- ★ Query Efficiency
- ★ Availability

References and Tools used

- Data Source: Mobile Insight Logs.
  - Li, Yuanjie, et al. "Mobileinsight: Extracting and analyzing cellular network information on smartphones." ACM Mobicom. 2016.
- Database: MySQL, Spark SQL, HDFS.
  - http://spark.apache.org/sql/
  - https://wiki.apache.org/hadoop/HDFS
- Front end: HTML, CSS.
- Backend: Java Server, Spark Server.

System Architecture

- MobileInsight
- MySql
- HDFS & Spark Sql
- Java Server
- Spark Server
- Front-end Interface



Current Progress

- MobileInsight 
- MySql ✓
- HDFS & Spark Sql ✓
- Java Server ✓
- Spark Server ✓
- Front-end Interface ✓

Log Query | Homepage | Search | Submit

Log Query

Select template query

Customize query message

```
SELECT * FROM tMsg LIMIT 10
```

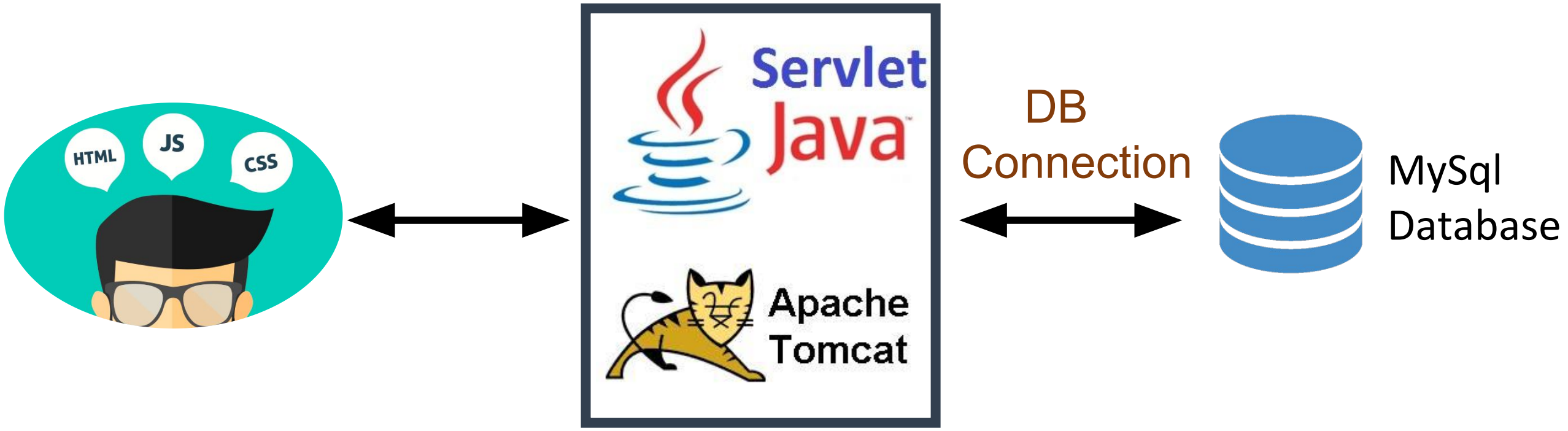
Submit Query

Query Result

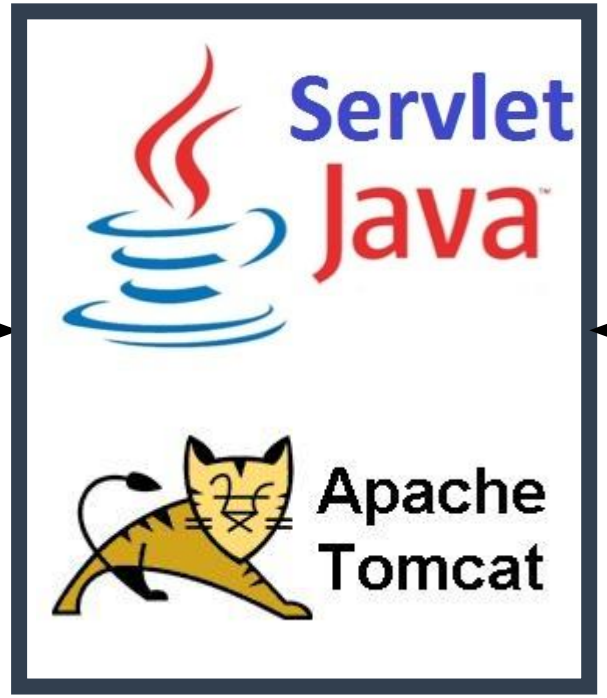
Result for query message "SELECT...FROM...WHERE..."

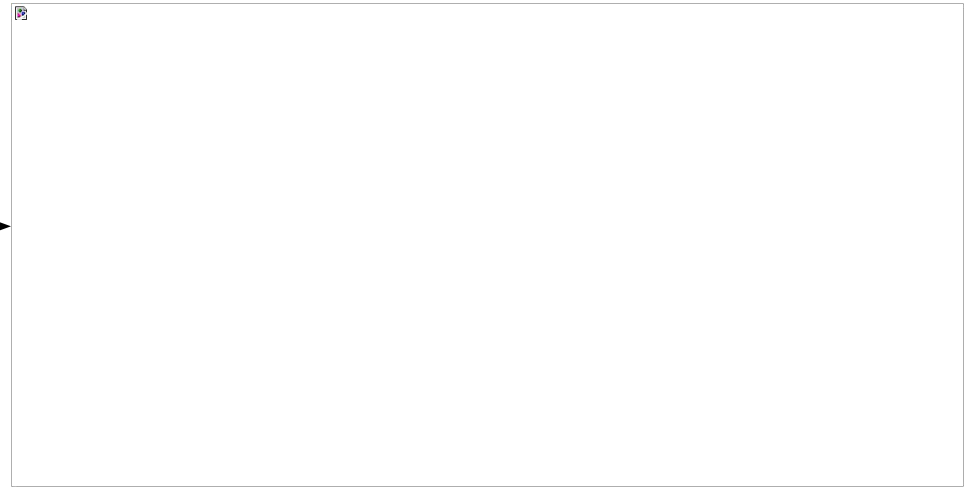
File Path	Phone	Carrier	Timestamp
/mnt/milog/by_operator_model/ATT-MicroCell_Samsung-SM-G900T/diag_log_20151110_161002_351881062060429_samsung-SM-G900T_ATT-MicroCell.mi2log	samsung-SM-G900T	ATT-MicroCell	2015-11-10 16:10:02
/mnt/milog/by_operator_model/ATT-MicroCell_Samsung-SM-G900T/diag_log_20151110_161002_351881062060429_samsung-SM-G900T_ATT-MicroCell.mi2log	samsung-SM-G900T	ATT-MicroCell	2015-11-10 16:10:02

System with MySql

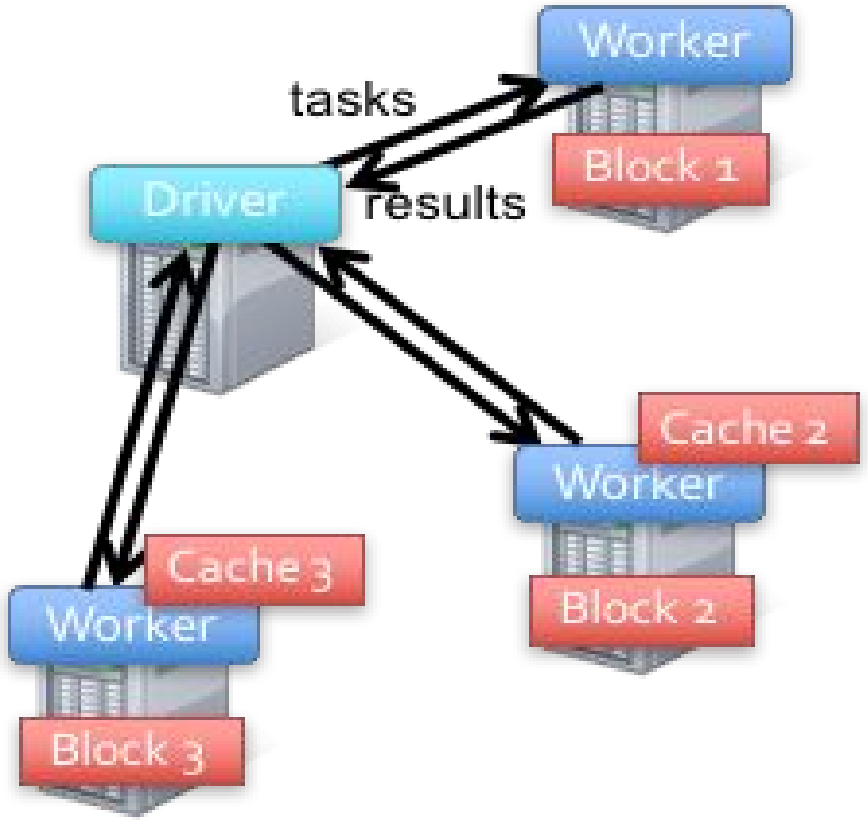


System with Spark Cluster





Spark Cluster Deployment



Spark Master at spark://s-164-67-66-46.resnet.ucla.edu:7077

URL: spark://s-164-67-66-46.resnet.ucla.edu:7077
REST URL: spark://s-164-67-66-46.resnet.ucla.edu:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 4 Total, 4 Used
Memory in use: 3.0 GB Total, 1024.0 MB Used
Applications: 1 Running, 2 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20161205183635-164.67.66.46-49804	164.67.66.46:49804	ALIVE	4 (4 Used)	3.0 GB (1024.0 MB Used)

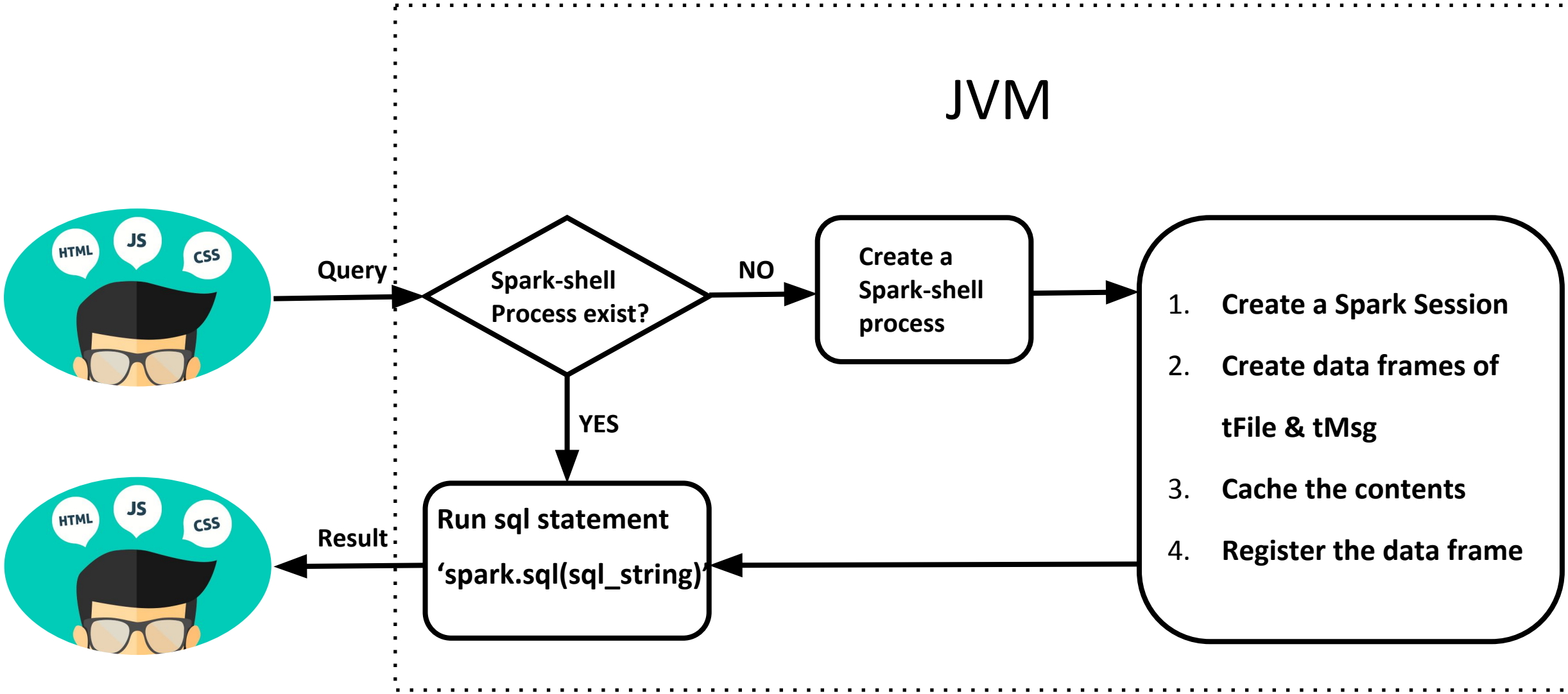
Running Applications

Application ID		Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20161205192646-0002	(kill)	Spark shell	4	1024.0 MB	2016/12/05 19:26:46	Connie.x	RUNNING	3.7 h

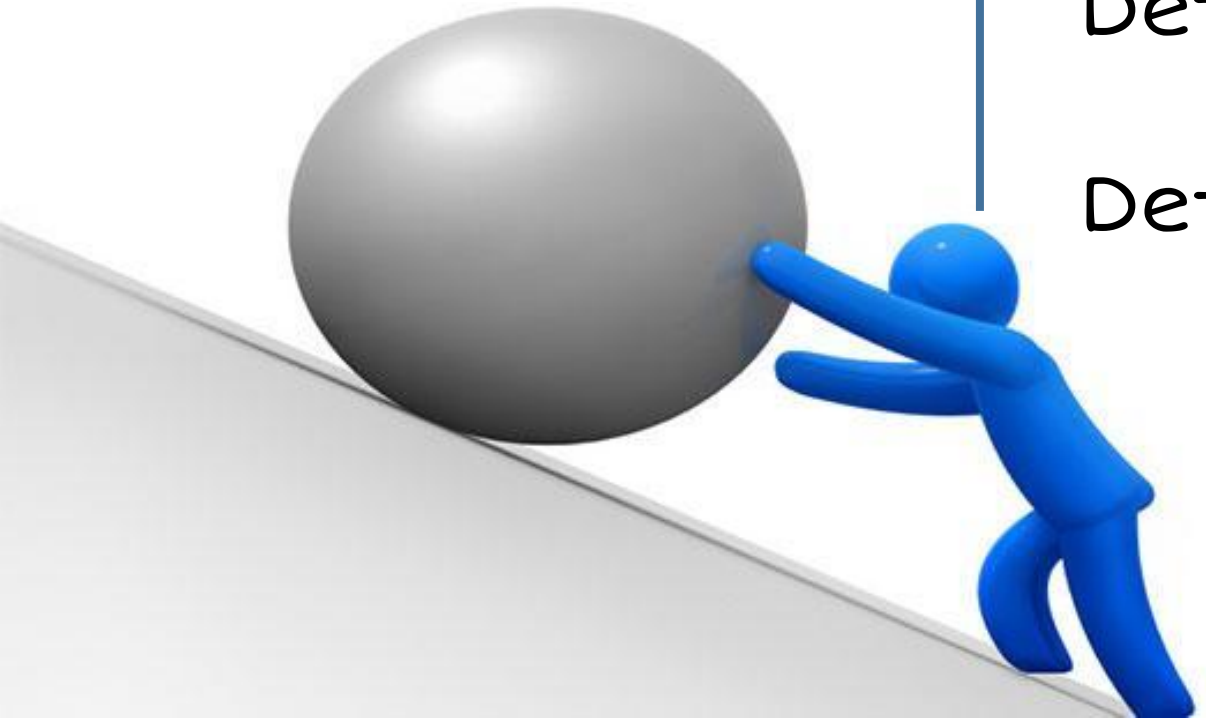
Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
app-20161205185415-0001	Spark shell	4	1024.0 MB	2016/12/05 18:54:15	Connie.x	FINISHED	17 min
app-20161205183820-0000	Spark shell	4	1024.0 MB	2016/12/05 18:38:20	Connie.x	FINISHED	15 min

Spark Server Logic



Testbed



Scenarios considered:

1) MySQL on one machine
2) Spark on one machine (without caching)
3) Spark on cluster (without caching)
4) Spark on cluster (with caching)

Data Size

6 GB of Logs (tFile and tMsg tables)

Details of one machine:

4 cores (1.6 GHz) , 4 GB RAM

Details of cluster:

3 machines: 2 cores (3.40GHz), 1GB RAM
13 machines: 2 cores (2.40GHz), 1 GB RAM

Total: 32 Cores and 16GB RAM

Demo

Results Analysis

-- MySQL queries on one machine

SELECT COUNT(*) FROM tMsg

Time = 60 Sec

Query Result

Running Time: 60119

Result for query message"SELECT COUNT(*) FROM tMsg;"

COUNT(*)
17628085

SELECT * FROM tFile LIMIT 10

Time = 0.26 Sec

Query Result

Running Time: 259

Result for query message"SELECT * FROM tFile LIMIT 10;"

Filepath	Phone
/home/sparker/zhouchang/ucla_milog/milogpart/ATT-MicroCell_Samsung-SM-G900T/diag_log_20151110_161002_351881062060429_samsung-SM-G900T_ATT-MicroCell.mi2log	samsung-SM-G900T

SELECT COUNT(*) FROM tMsg join tFile on tMsg.Filepath = tFile.Filepath

Time = 65 Sec

Query Result

Running Time: 65265

Result for query message"select count(*) from tMsg join tFile on tMsg.Filepath =

count(*)
17628085

Results Analysis Cont.

-- Spark queries on one machine (without caching)

SELECT COUNT(*) FROM tMsg

Time = 100 Sec

```
Query Result
Running Time: 100035
Result for query message"SELECT Count(*) FROM tMsg"
scala> spark.sql("SELECT Count(*) FROM tMsg").show(10000,false)
+--------+
|count(1)|
+--------+
|17628086|
```

SELECT * FROM tFile LIMIT 10

Time = 0.35 Sec

```
Query Result
Running Time: 349
Result for query message"SELECT * FROM tFile LIMIT 10"
scala> spark.sql("SELECT * FROM tFile LIMIT 10").show(10000,false)
+----------------------------------------------------------------------------------------------------
|Filepath |Phone |Carrier|Timestamp |
+----------------------------------------------------------------------------------------------------
|Filepath |Phone |Carrier|Timestamp |
|/home/sparker/zhouchang/ucla_milog/milogpart/Verizon_LGE-VS985/diag_lo
VS985|Verizon|2015/12/23 11:34|
```

SELECT COUNT(*) FROM tMsg join tFile on tMsg.Filepath = tFile.Filepath

Time = 102 Sec

```
Query Result
Running Time: 102733
Result for query message"select count(*) from tMsg join tFile on tMsg.Filepath = tFile.Filepath"
scala> spark.sql("select count(*) from tMsg join tFile on tMsg.Filepath = tFile.
Filepath").show(10000,false)
+--------+
|count(1)|
+--------+
|17628086|
+--------+
```

Results Analysis Cont.

--Spark queries on cluster (without Caching)

SELECT COUNT(*) FROM tMsg

Time = 5.53 Sec

Query Result

Running Time: "5530"

Result for query message"SELECT Count(*) FROM tMsg"

```
scala> spark.sql("SELECT Count(*) FROM tMsg").show(10000,false)
+--------+
|count(1)|
+--------+
|17628086|
```

SELECT * FROM tFile LIMIT 10

Time = 0.35 Sec

Query Result

Running Time: "353"

Result for query message"SELECT * FROM tFile Limit 10"

```
scala> spark.sql("SELECT * FROM tFile Limit 10").show(10000,false)
+------------------------------------------------------------------------
|Filepath |Phone |Carrier|Timestamp |
+------------------------------------------------------------------------
|Filepath |Phone |Carrier|Timestamp |
|/home/sparker/zhouchang/ucla_milog/milogpart/Verizon_LGE-VS985/diag_
VS985_Verizon.mi2log|LGE-VS985|Verizon|2015/12/23 11:34|
```

SELECT COUNT(*) FROM tMsg join tFile on tMsg.Filepath = tFile.Filepath

Time = 5.68 Sec

Query Result

Running Time: "5681"

Result for query message"SELECT Count(*) FROM tMsg join tFile on tMsg.Filepath=tFile.Filepath"

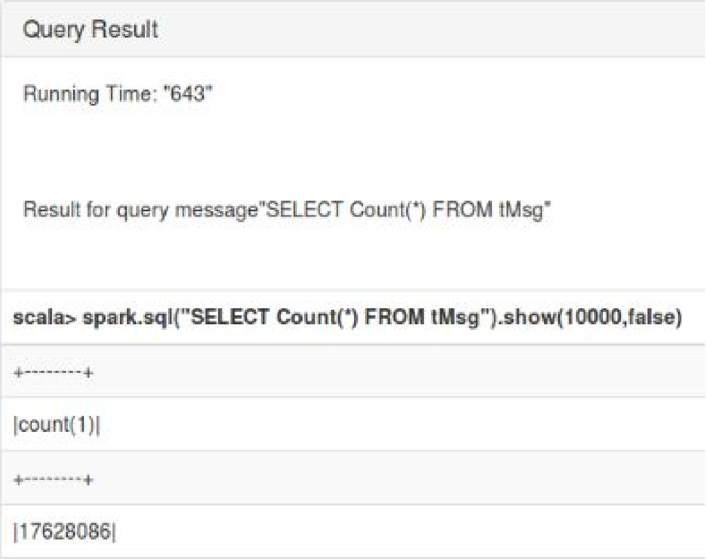
```
scala> spark.sql("SELECT Count(*) FROM tMsg join tFile on tMsg.Filepath=tFile.Fi
lepath").show(10000,false)
+--------+
|count(1)|
+--------+
|17628086|
```

Results Analysis Cont.

-- Spark queries on cluster (with Caching)

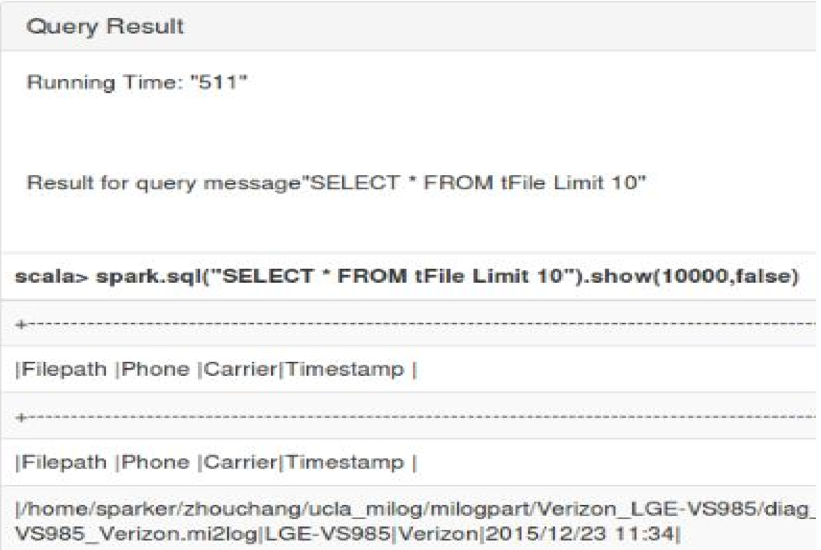
SELECT COUNT(*) FROM tMsg

Time = 0.64 Sec



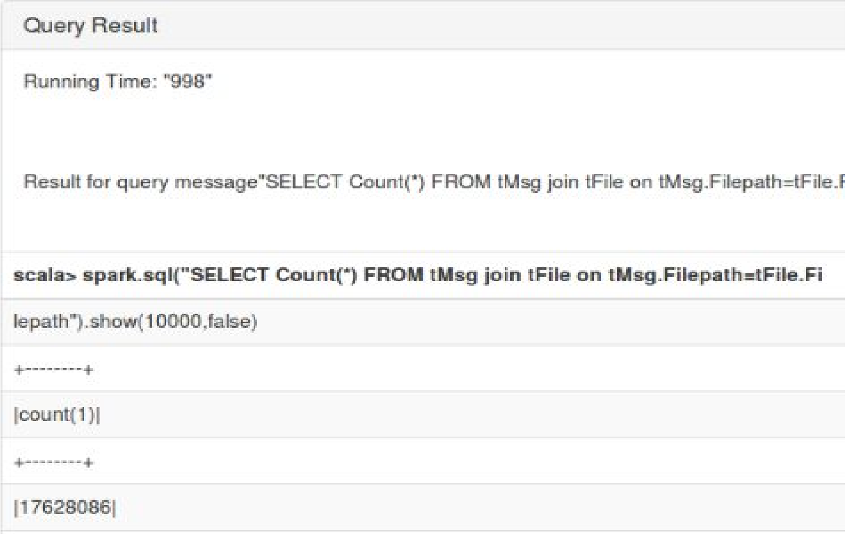
SELECT * FROM tFile LIMIT 10

Time = 0.35 Sec



SELECT COUNT(*) FROM tMsg join tFile on tMsg.Filepath = tFile.Filepath

Time = 1 Sec



Results Analysis Cont.

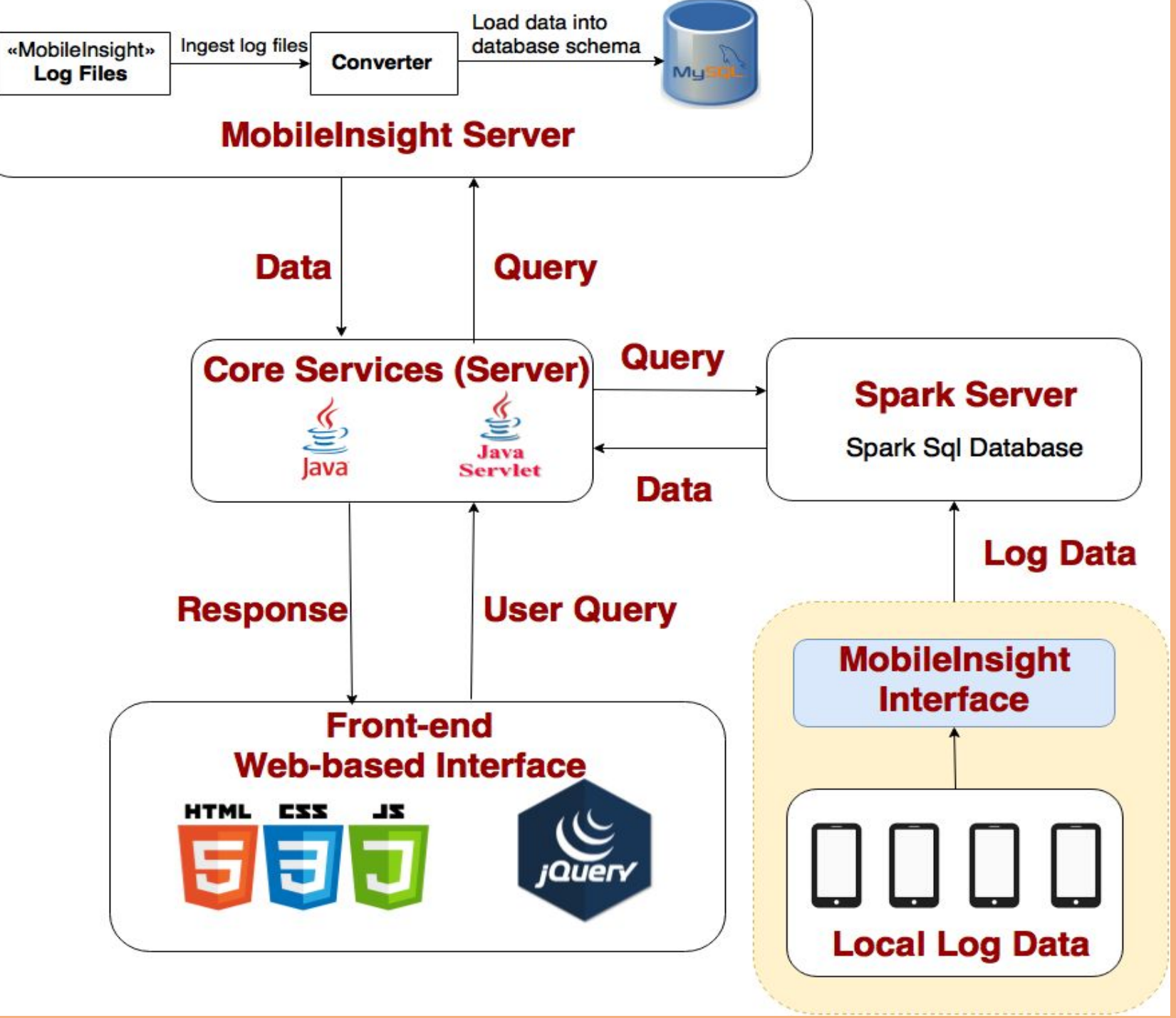
Scenario \ Query	SELECT COUNT(*) FROM tMsg	SELECT * FROM tFile LIMIT 10	select count(*) from tMsg join tFile on tMsg.Filepath = tFile.Filepath
MySql on One Machine	60 sec	0.26 sec	65 sec
Spark on One Machine (without caching)	100 sec	0.35 sec	102.8 sec
Spark on Cluster (without caching)	5.53 sec	0.35 sec	5.68 sec
Spark on Cluster (with caching)	0.64 sec	0.35 sec	0.99 sec

Discussion of results

1. Multiple machines
2. In memory processing

Performance Bottleneck

- Keep data in cache: We kept entire tables in cache.
- Data size is very large: Entire Data cannot be kept in cache.
  - Make most common cases fast.
  - Predict nature of queries.
  - Keep smaller views/tables in memory.



Future Direction

Adding capability to store and process data as it is generated from multiple devices in real-time.

Things to consider:

1) Refreshing cache of spark.
2) Data curation at runtime.
3) Data pre-processing

Summary

- We presented a Webframe to do log query using Spark which added scalability, efficiency and availability.
- We compared its performance with traditional MySQL database.
- We tested its performance on cluster with/without using in-memory processing.

We conclude: Such a system can improve latency by ~100 times, and can be used to query very large datasets.

Thanks!