Spark User Concurrency and Context/RDD Sharing at Production Scale

Jorge Alarcon

VP Engineering, Zoomdata

https://dev.zoomdata.com/zoomdata/admin.html#sources

ZOOMDATA

Add a New Data Source

CSV, JSON, TSV, XML

S3


HDFS


Upload API


Twitter

Impala

Search

Redshift

Kinesis

Solr


Elastic Search


MongoDB

Spark

Oracle

MySQL


SQL Server


PostgreSQL

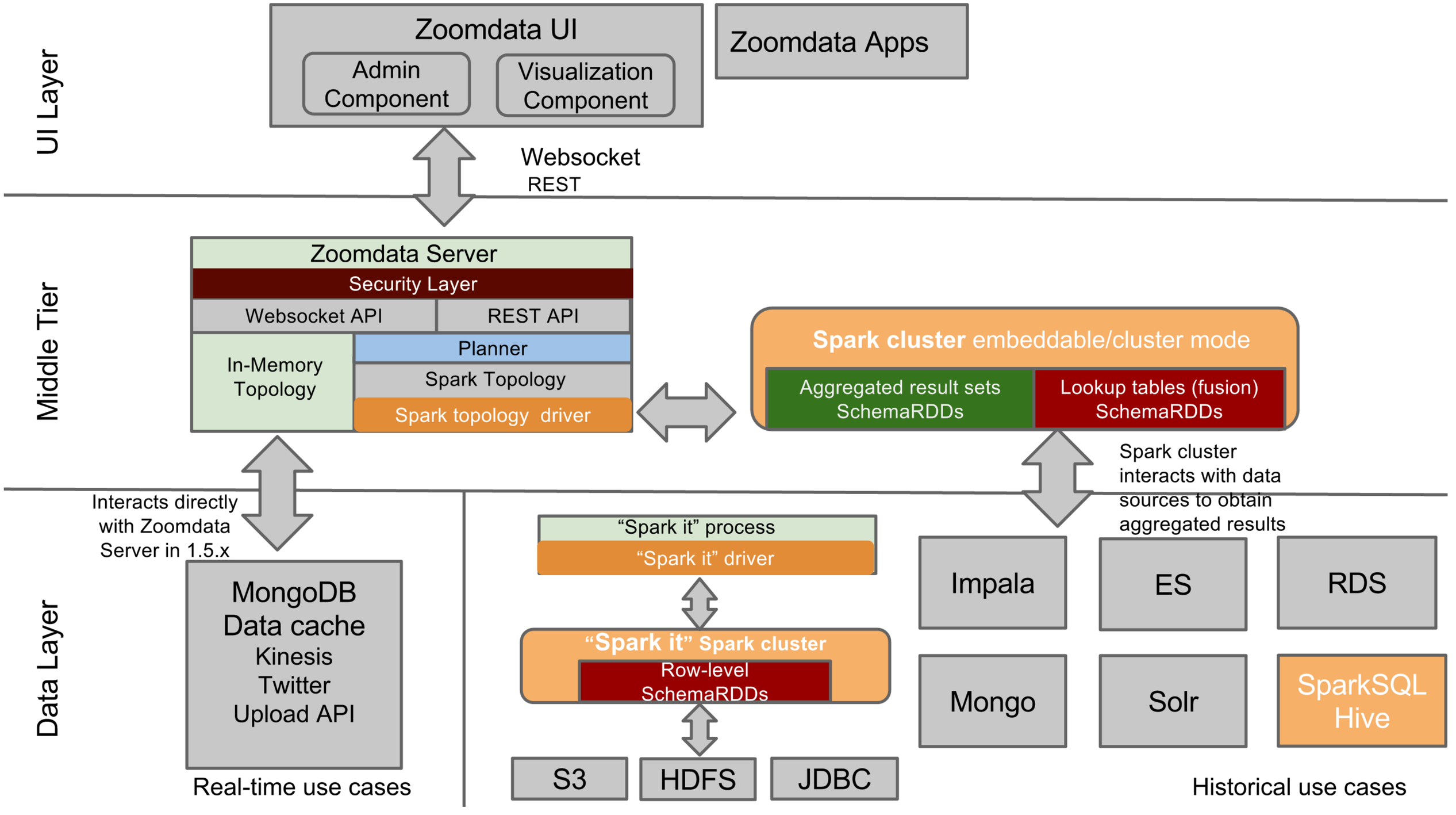
My Data Sources

Manage Connections

Type	Data Source Name	Connection	Cache	Enable	Delete	Download

Import Source

Imported Source Name | Browse source file | Clear | Import


UI Layer
Zoomdata UI
Admin Component
Visualization Component
Zoomdata Apps
Websocket
REST
Middle Tier
Zoomdata Server
Security Layer
Websocket API
REST API
In-Memory Topology
Planner
Spark Topology
Spark topology driver
Spark cluster embeddable/cluster mode
Aggregated result sets SchemaRDDs
Lookup tables (fusion) SchemaRDDs
Spark cluster interacts with data sources to obtain aggregated results
Interacts directly with Zoomdata Server in 1.5.x
Data Layer
MongoDB
Data cache
Kinesis
Twitter
Upload API
Real-time use cases
"Spark it" process
"Spark it" driver
"Spark it" Spark cluster
Row-level SchemaRDDs
S3
HDFS
JDBC
Impala
ES
RDS
Mongo
Solr
SparkSQL Hive
Historical use cases

binary compatibility challenge

- Reading from HDFS
- Spark driver's hadoop libraries need to match those on the spark cluster
- We made these libraries configurable by:
  - Decoupling the spark driver into a separate process
  - Allowing administrators to configure what libraries through a UI
  - Kicking off a spark driver process with the configured hadoop libraries

rdd sharing

- Achieved by remoting the Spark driver process so that multiple Zoomdata servers can point to it
- Zoomdata tests if the process is alive
- If not alive, it spawns the process
- Running this process transparent to users

Spark Driver Remoting

	Zoomdata Spark driver process	Hive Thrift Server	Spark Job Server
Fine grained configuration of Spark Context	Yes	No	Some
Progress Reporting	Yes	No	Evaluating
Custom Parsers of Raw Values JdbcRdd.convertValue	Yes	No	Evaluating
TSV support	Yes	No	Evaluating
XML support	Yes	No	Evaluating
YARN support	Yes	Yes	Recently added
Spark programming model	Yes	No	impacted
load and read requests in separate scheduler's pools	Yes	N/A	Evaluating
Cancel queries	Yes	Yes	Yes

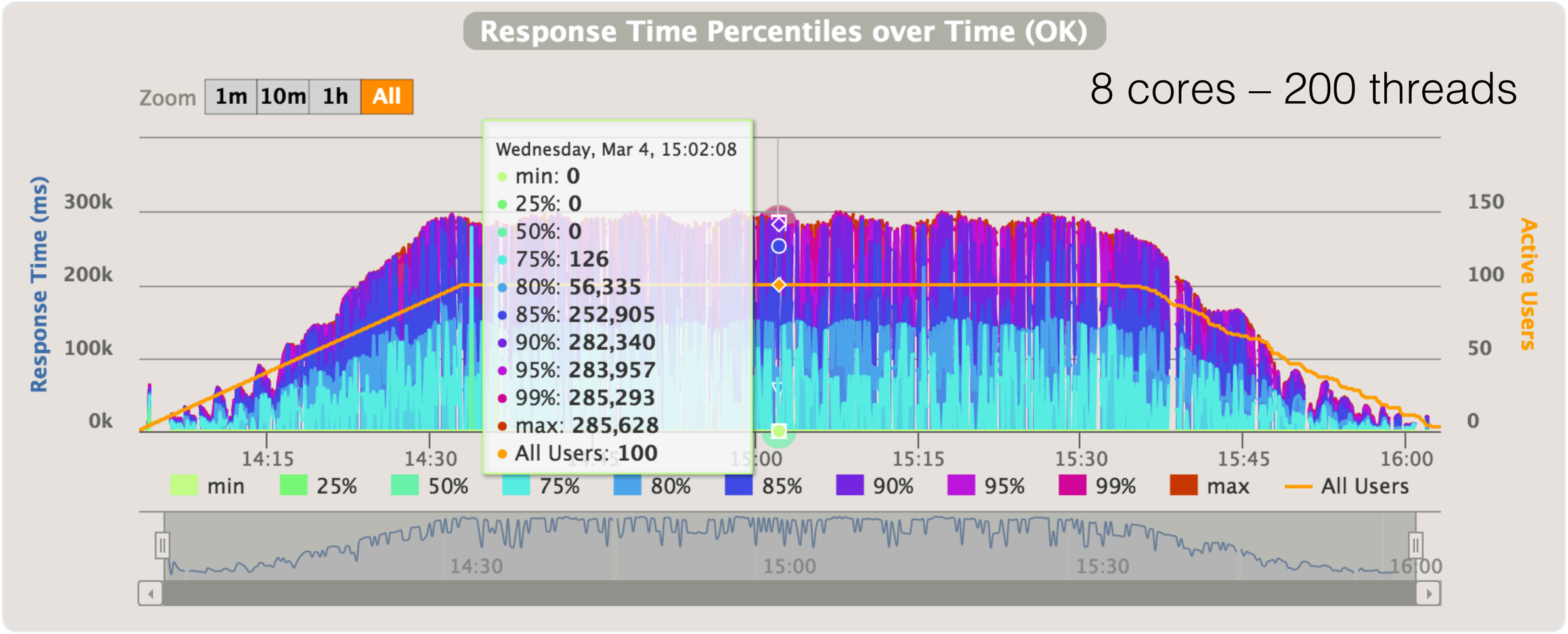
performance challenge

- Initial implementation used columnar format borrowed from Shark
  - 3 times less memory and effective GC
- Currently moving to SchemaRDDs and SparkSQL
- Different scheduler pools for load and read requests

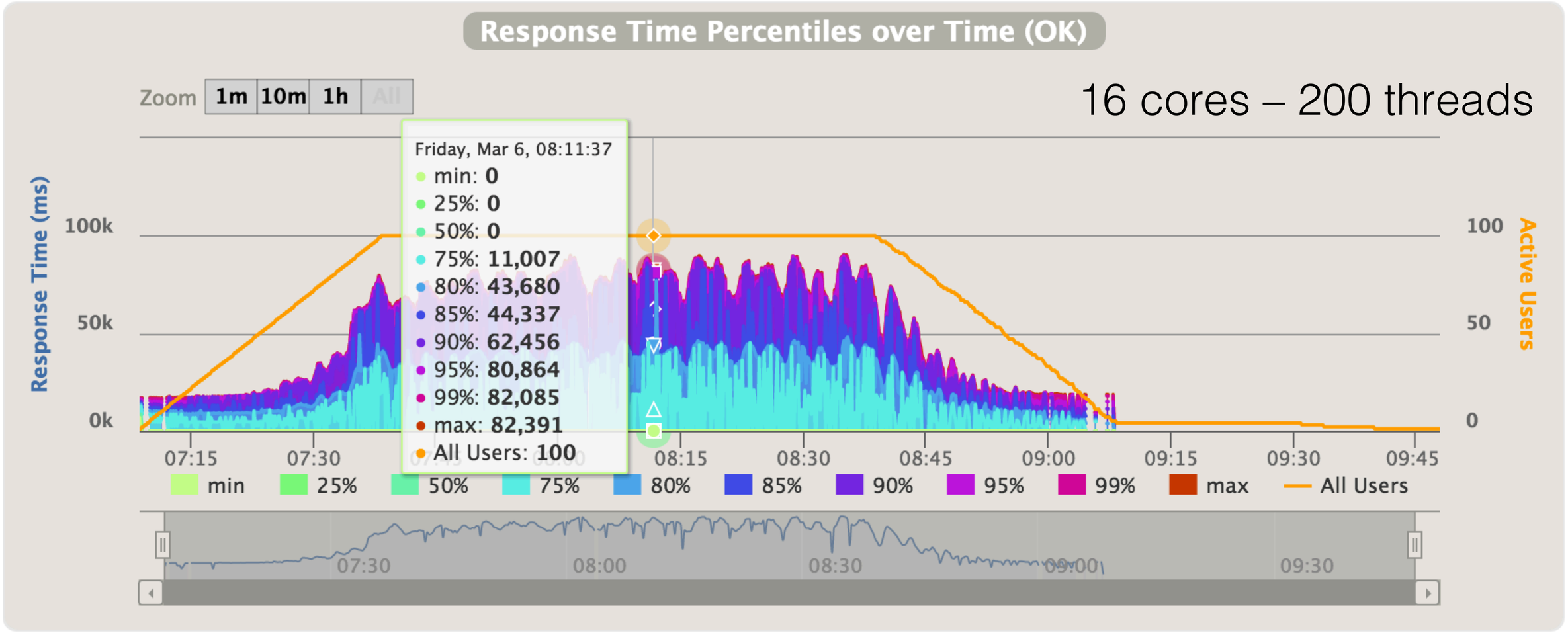
```
public static void assignToLoadersPool(JavaSparkContext sparkContext) {
    sparkContext.setLocalProperty("spark.scheduler.pool", "loaders");
}

public static void assignToReadersPool(JavaSparkContext sparkContext) {
    sparkContext.setLocalProperty("spark.scheduler.pool", "readers");
}
```

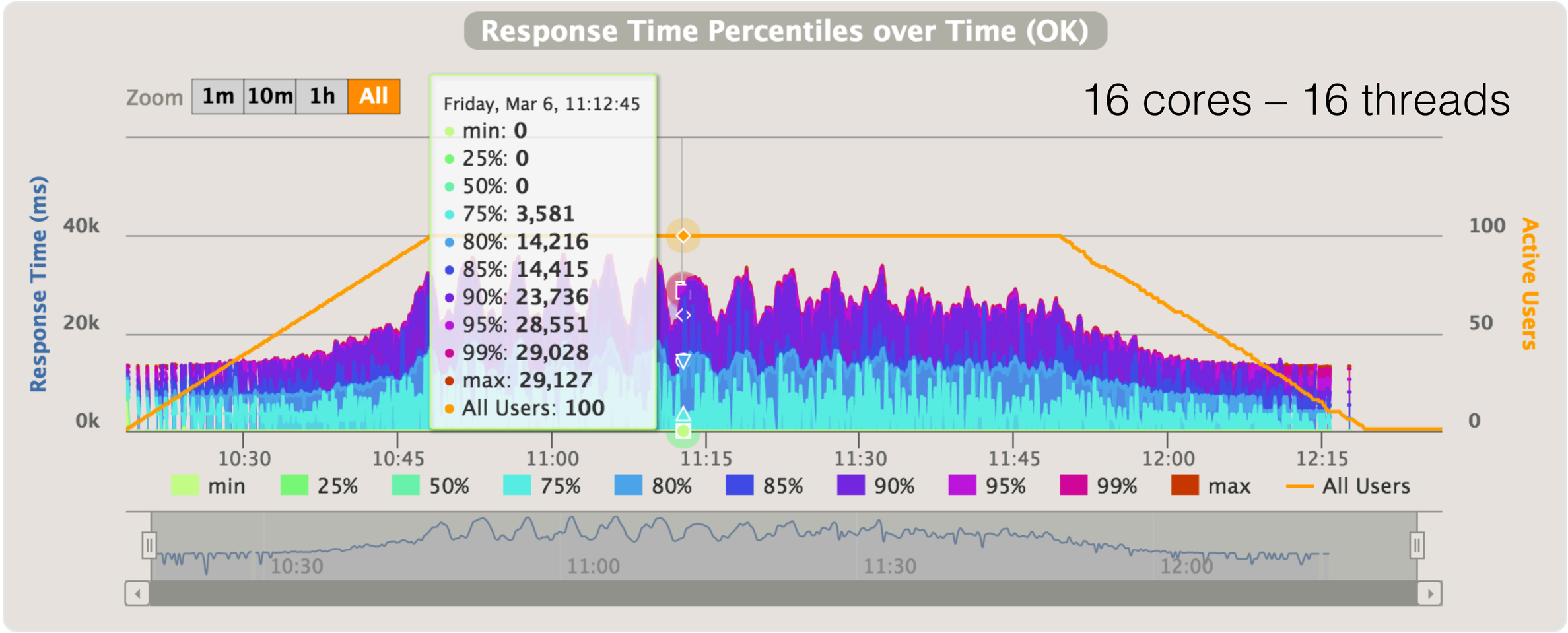
worker threads and context switching



worker threads and context switching



worker threads and context switching



what is next for us

- Consolidate Spark drivers
- Improve RDD re-use: not only supersets but also sub-set RDDs
- Keep watching HiveThriftServer and the Spark Job Server as a replacement to our Spark driver process