Dr. Elephant: Achieving Quicker, Easier, and Cost-Effective Big Data Analytics

Akshay Rai, Linkedin

#EUdev9

Overview

- Why Dr. Elephant?
- How does Dr. Elephant work?
- Spark Integration Challenges
- Spark Rules/Heuristics
- Roadmap

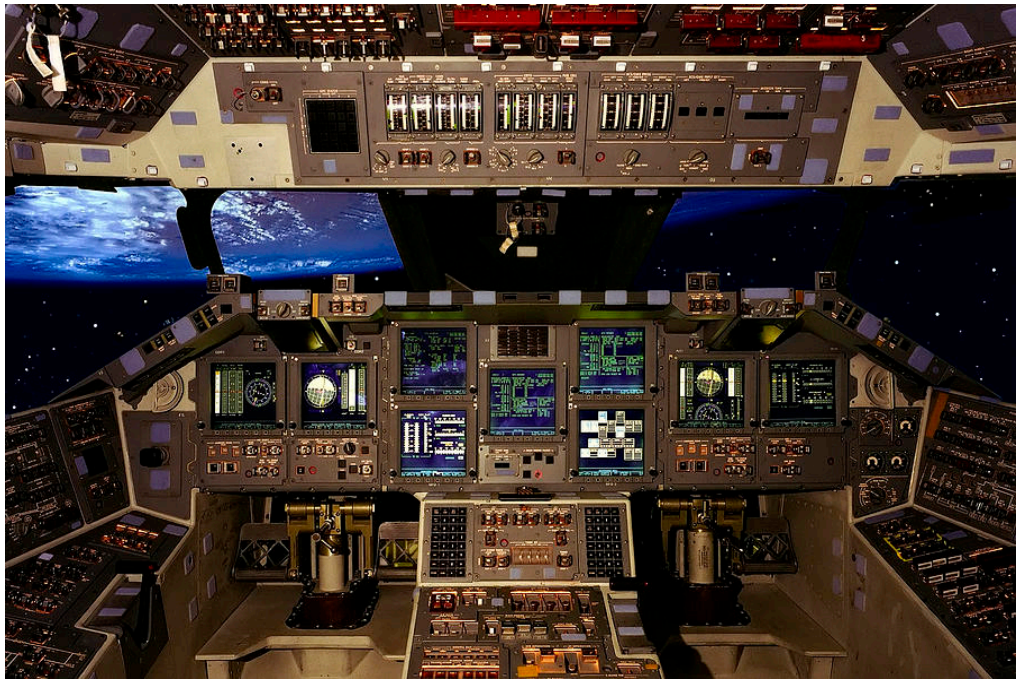
Grid Scale at Linkedin

2008	2018
1 cluster	10+ clusters
20 nodes	1000s of nodes
5 users	1000s of active users
MapReduce	Pig, Hive, Spark, etc.
Few workflows	hundred-thousand workflows

How much can we scale?

Cluster and Job Tuning

Challenges in Tuning



- Too many frameworks
- Too Many Different Knobs
- Overwhelming Data
- Scattered Data

Developer Productivity

Vs

Cluster Efficiency

Training Users to Tune

- Train and educate

Issues:

- Newer Frameworks
- More & more People
- Varied Experience

Expert Review

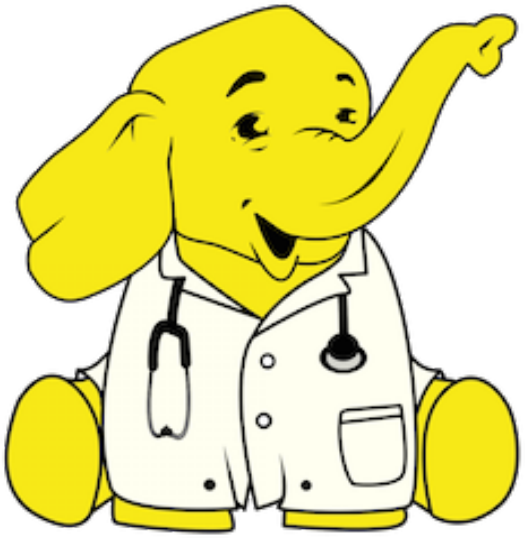
- Experts review & approve jobs
- Every job is reviewed

Issues:

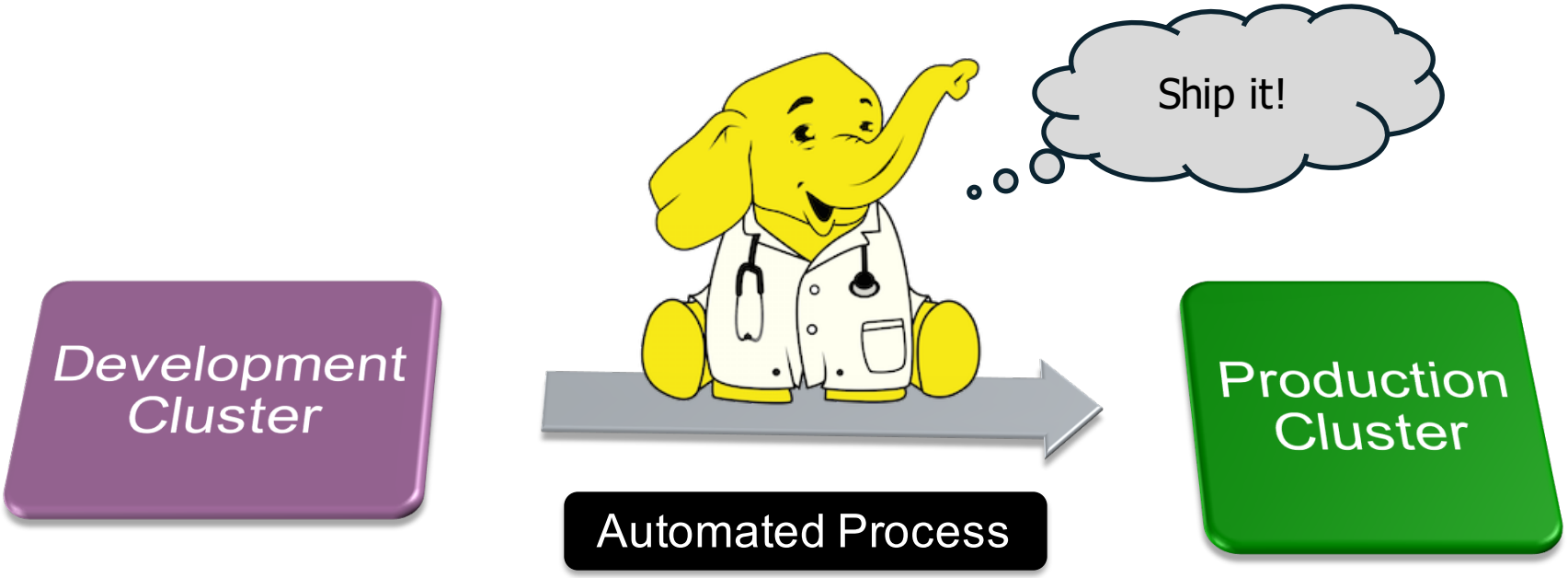
- Not Scalable
- Delayed Reviews
- Manual Review Process

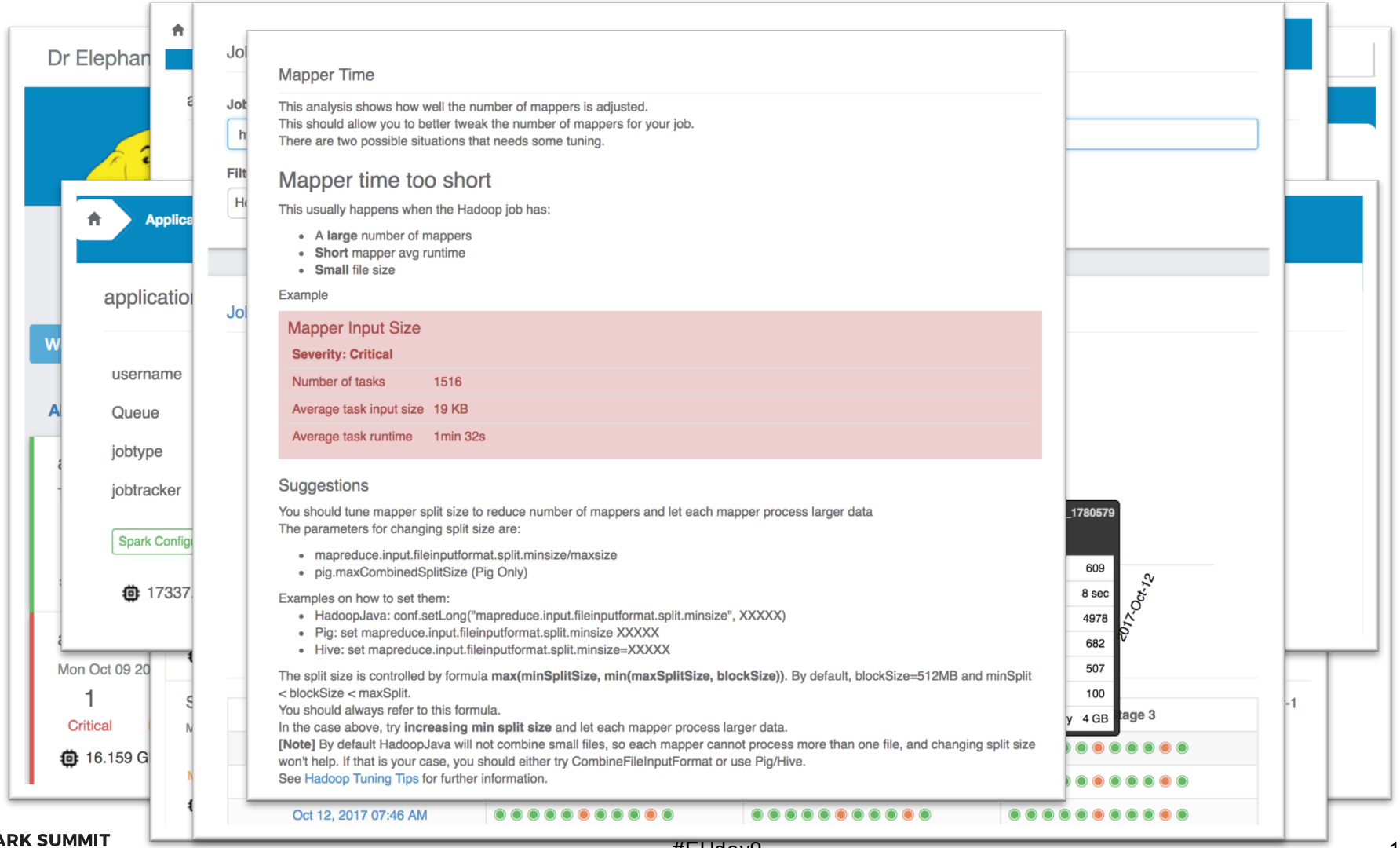
Birth of Dr. Elephant

- Identifies Common Issues
- Gives Actionable Advice
- Recommends Best practices
- Tracks Flow and Job Health

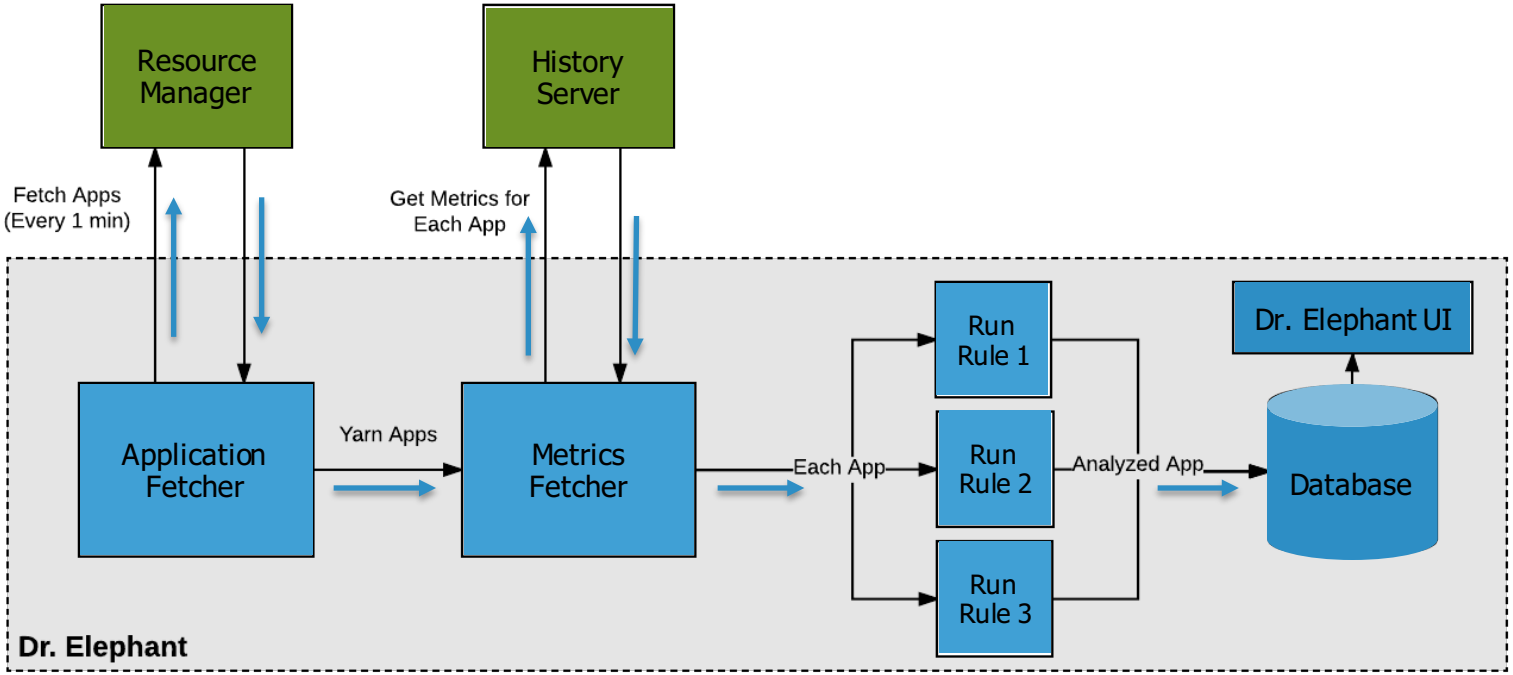


@Dr. Elephant, please review



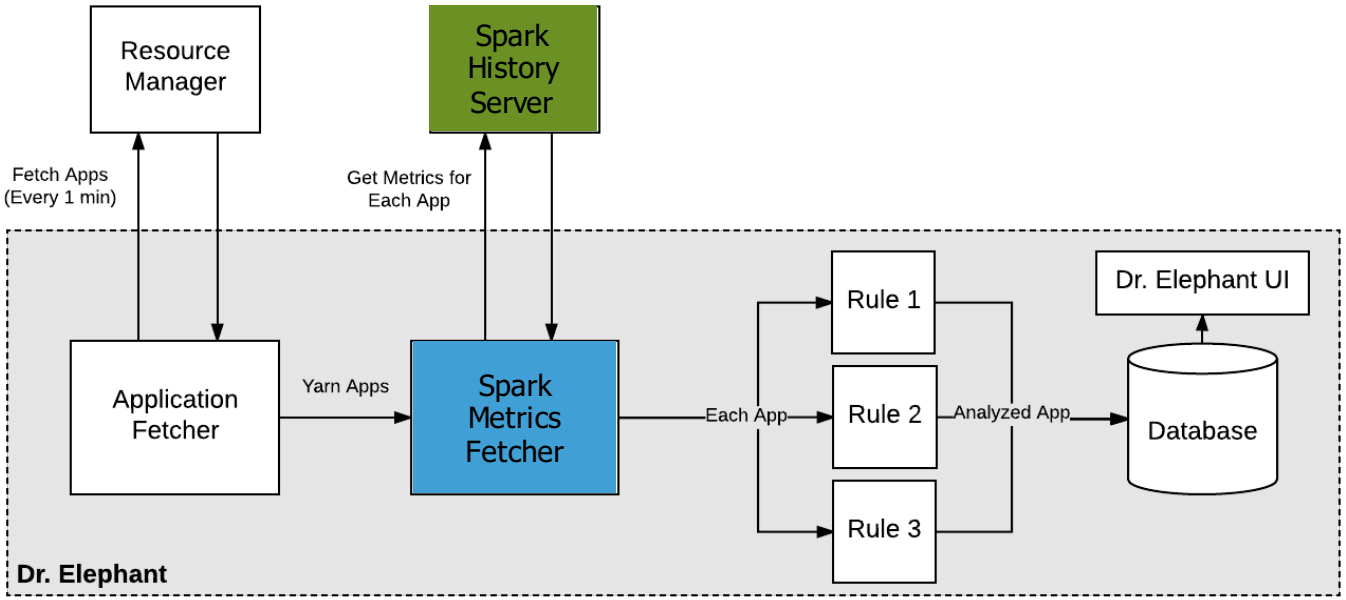

Mapper Time
This analysis shows how well the number of mappers is adjusted.
This should allow you to better tweak the number of mappers for your job.
There are two possible situations that needs some tuning.
Mapper time too short
This usually happens when the Hadoop job has:
A large number of mappers
Short mapper avg runtime
Small file size
Example
Mapper Input Size
Severity: Critical
Number of tasks 1516
Average task input size 19 KB
Average task runtime 1min 32s
Suggestions
You should tune mapper split size to reduce number of mappers and let each mapper process larger data
The parameters for changing split size are:
mapreduce.input.fileinputformat.split.minsize/maxsize
pig.maxCombinedSplitSize (Pig Only)
Examples on how to set them:
HadoopJava: conf.setLong("mapreduce.input.fileinputformat.split.minsize", XXXXX)
Pig: set mapreduce.input.fileinputformat.split.minsize XXXXX
Hive: set mapreduce.input.fileinputformat.split.minsize=XXXXX
The split size is controlled by formula max(minSplitSize, min(maxSplitSize, blockSize)). By default, blockSize=512MB and minSplit < blockSize < maxSplit.
You should always refer to this formula.
In the case above, try increasing min split size and let each mapper process larger data.
[Note] By default HadoopJava will not combine small files, so each mapper cannot process more than one file, and changing split size won't help. If that is your case, you should either try CombineFileInputFormat or use Pig/Hive.
See Hadoop Tuning Tips for further information.
Dr Elephan
applicatio
username
Queue
jobtype
jobtracker
Spark Config
Mon Oct 09 20
1
Critical
16.159 G
Oct 12, 2017 07:46 AM

How does it work?



Spark Support in Dr. Elephant

Spark Metrics Fetcher



Unstable Spark History Server

- Report delayed at Peak Load
- Too many Dr. Elephant REST calls - SHS Crashes

Solution

- Event Log Parser
- Fix Spark History Server

Event Log Parser

- Replay HDFS Logs with Listeners

Issues:

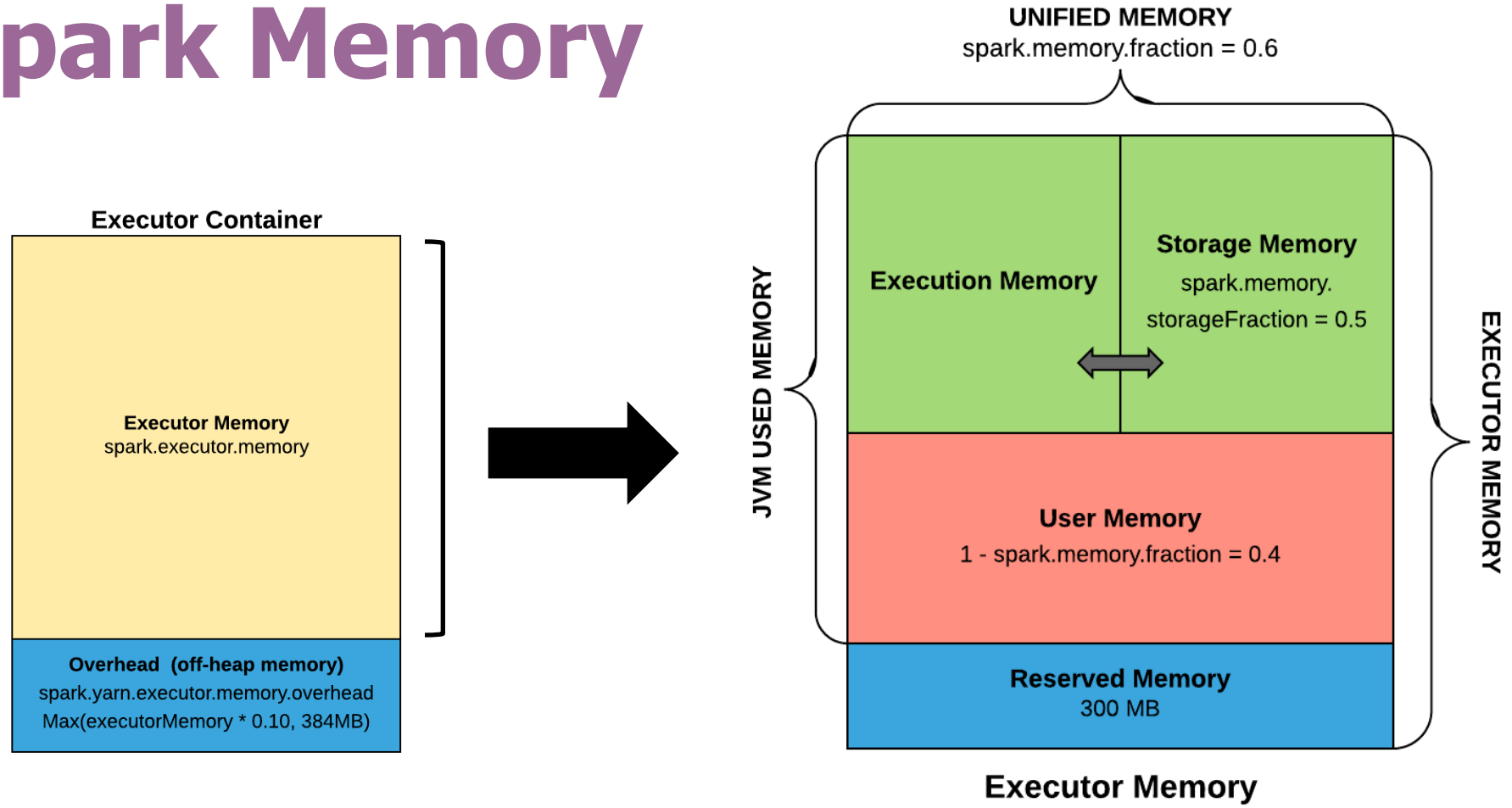
- Large Log Files.
  - More Resources
  - Backlog of jobs due to blocked threads
- Parsing logic is not public

Fix Spark History Server

- Use SPARK-18085 fork
- Uses Level DB with improved caching
- Some patches to add More metrics
  - JVM Used Memory
  - Executor disk space used

Spark Rules / Heuristics

Spark Memory

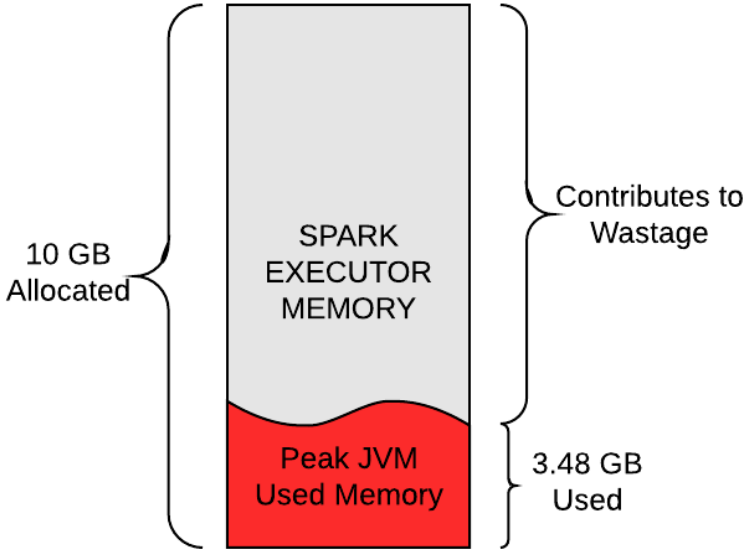


JVM Used Memory Heuristic

Spark JVM Used Memory

Severity: Critical

Max driver peak JVM used memory	1 GB
Max executor peak JVM used memory	3.48 GB
spark.driver.memory	2 GB
spark.executor.memory	10 GB



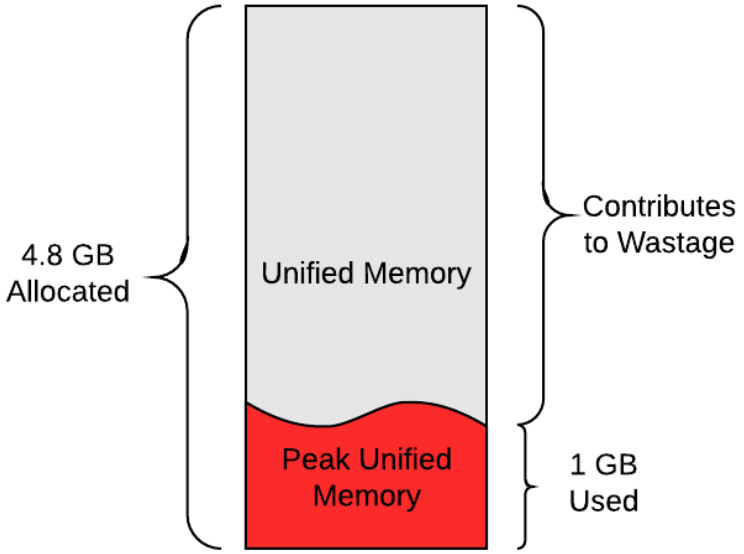
Reduce *spark.executor.memory*!

Unified Memory Heuristic

Spark Peak Unified (Execution + Storage) Memory

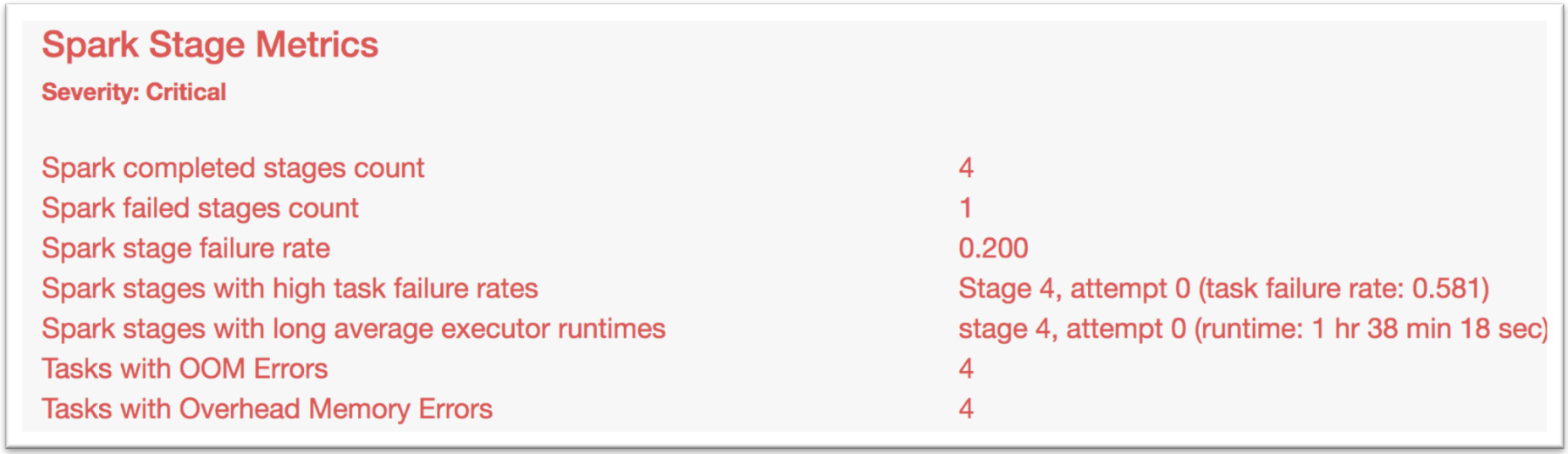
Severity: Critical

Max Peak Unified Memory among all executors	1 GB
spark.memory.fraction	0.6
Allocated Unified Memory Region	4.8 GB
Executor Memory Allocated	8 GB




```
If JVM used memory is low then reduce spark.executor.memory.
If JVM used memory is high then reduce spark.memory.fraction.
```

Spark Stage Heuristic



Spark Stage Metrics

Severity: Critical

Spark completed stages count	4
Spark failed stages count	1
Spark stage failure rate	0.200
Spark stages with high task failure rates	Stage 4, attempt 0 (task failure rate: 0.581)
Spark stages with long average executor runtimes	stage 4, attempt 0 (runtime: 1 hr 38 min 18 sec)
Tasks with OOM Errors	4
Tasks with Overhead Memory Errors	4

Spark Configuration Heuristic

Spark Configuration

Critical

spark.application.duration	552 Seconds
spark.driver.memory	2 GB
spark.dynamicAllocation.enabled	false
spark.executor.cores	1
spark.executor.instances	200
spark.executor.memory	10 GB
spark.shuffle.service.enabled	false
Spark Jars Parameter	./lib/*

More Heuristics

- Skewness in the Memory Regions
- Execution or Storage Spill
- GC Time heuristic

Customization

- Write and Configure New Rules
- Easily integrate with new schedulers
- More Fetchers
- More Job types

Open Source

- Active and Vibrant Community
- Hangout Meetings
- Meet-ups
- Spark Discussions

github.com/linkedin/dr-elephant

Roadmap

- Specific Suggestions
- Spark Debugging
- More Spark Heuristics
- Apache Incubation

References

- Engineering Blogs:
  - Open Source Blog: Click here
  - A year later after open source: Click here
- Open Source Github Link: github.com/linkedin/dr-elephant
- Mailing List: dr-elephant-users
- Hangout Meetings:
  - Agenda: Click here
  - Minutes: Click here
- Hadoop Summit 2015 (Before Open Source): Click here
- Fifth Elephant Conference 2016 (After Open Source): Click here

Thank You