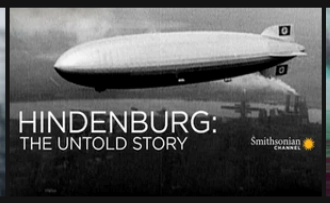
Productionizing Spark on Yarn for ETL

Ashwin Shankar
Nezih Yigitbasi

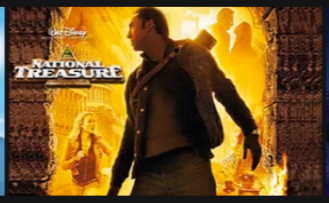
NETFLIX

NETFLIX Browse Kids DVD Search eva

Suspenseful TV Shows

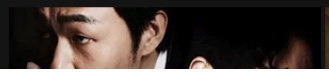
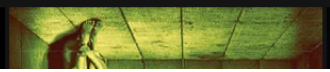


Children & Family Movies

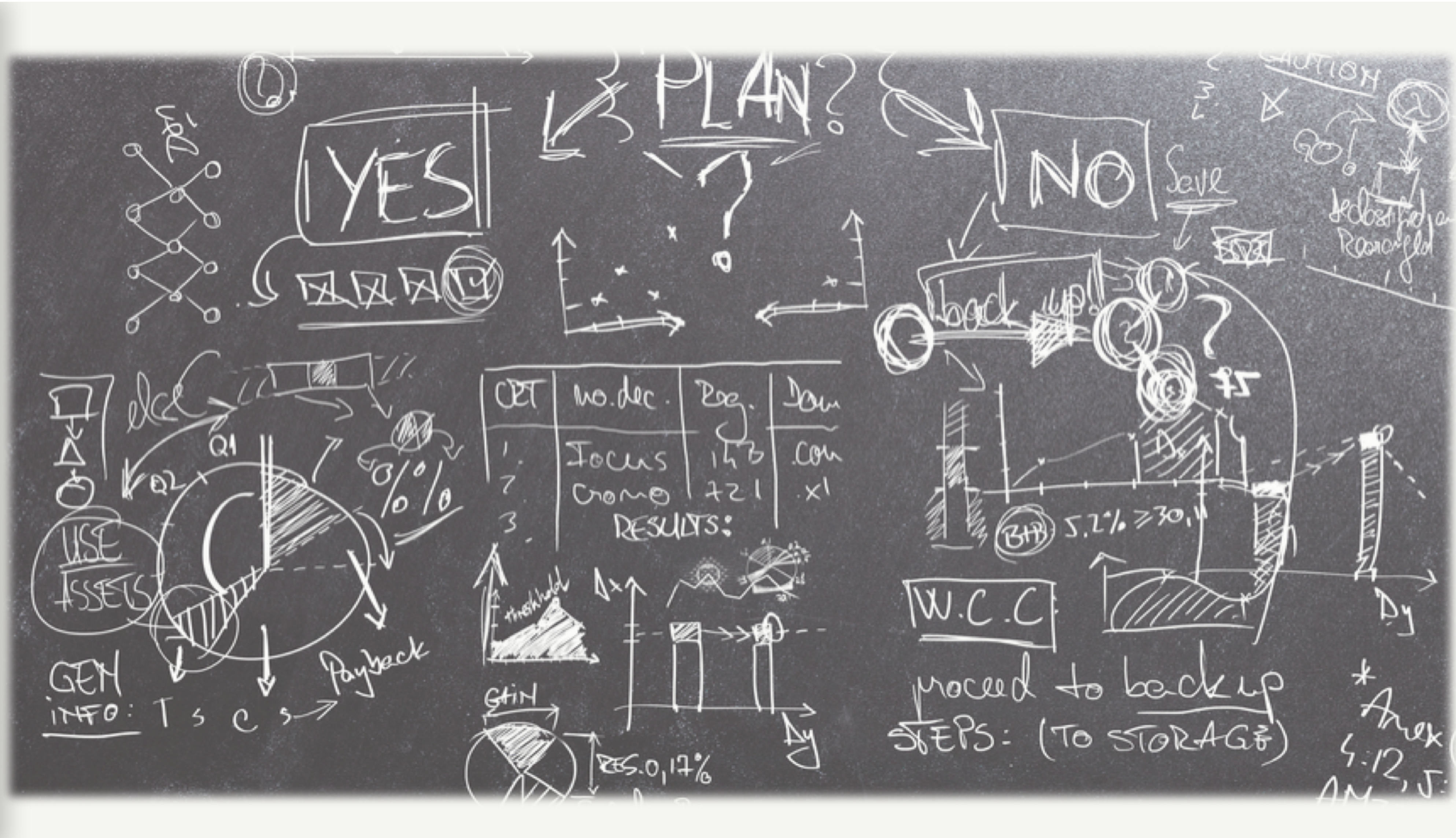


Social & Cultural Documentaries

Foreign Movies




PLAN?
YES
NO
back up!
RESULTS:
W.C.C
Payback
USE ASSETS
GEN INFO:
STEPS: (TO STORAGE)
Focus
threshold

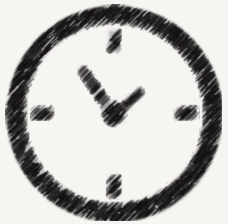
Scale

Netflix Key Business Metrics

81+ million members

Global

1000+ devices supported



125 million hours / day

Netflix Key Platform Metrics

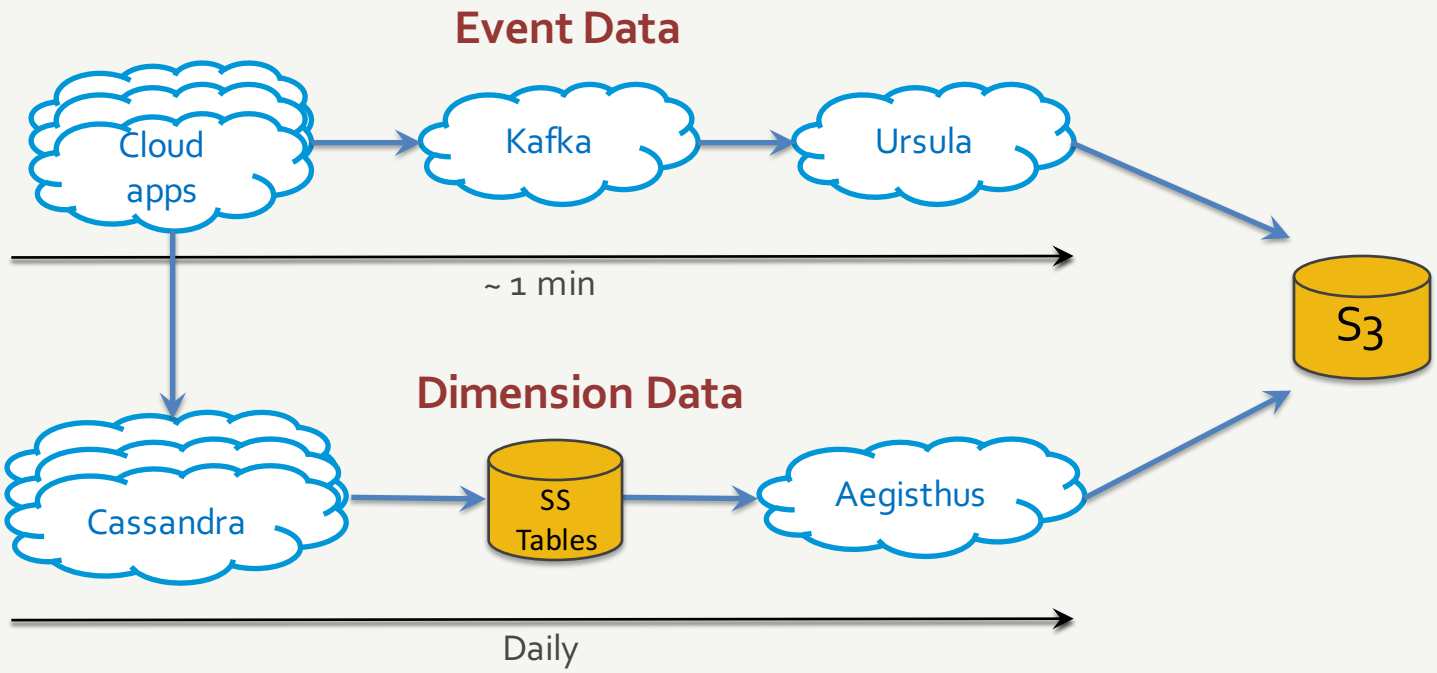
700B Events | 40 PB DW | Read 3PB | Write 300TB

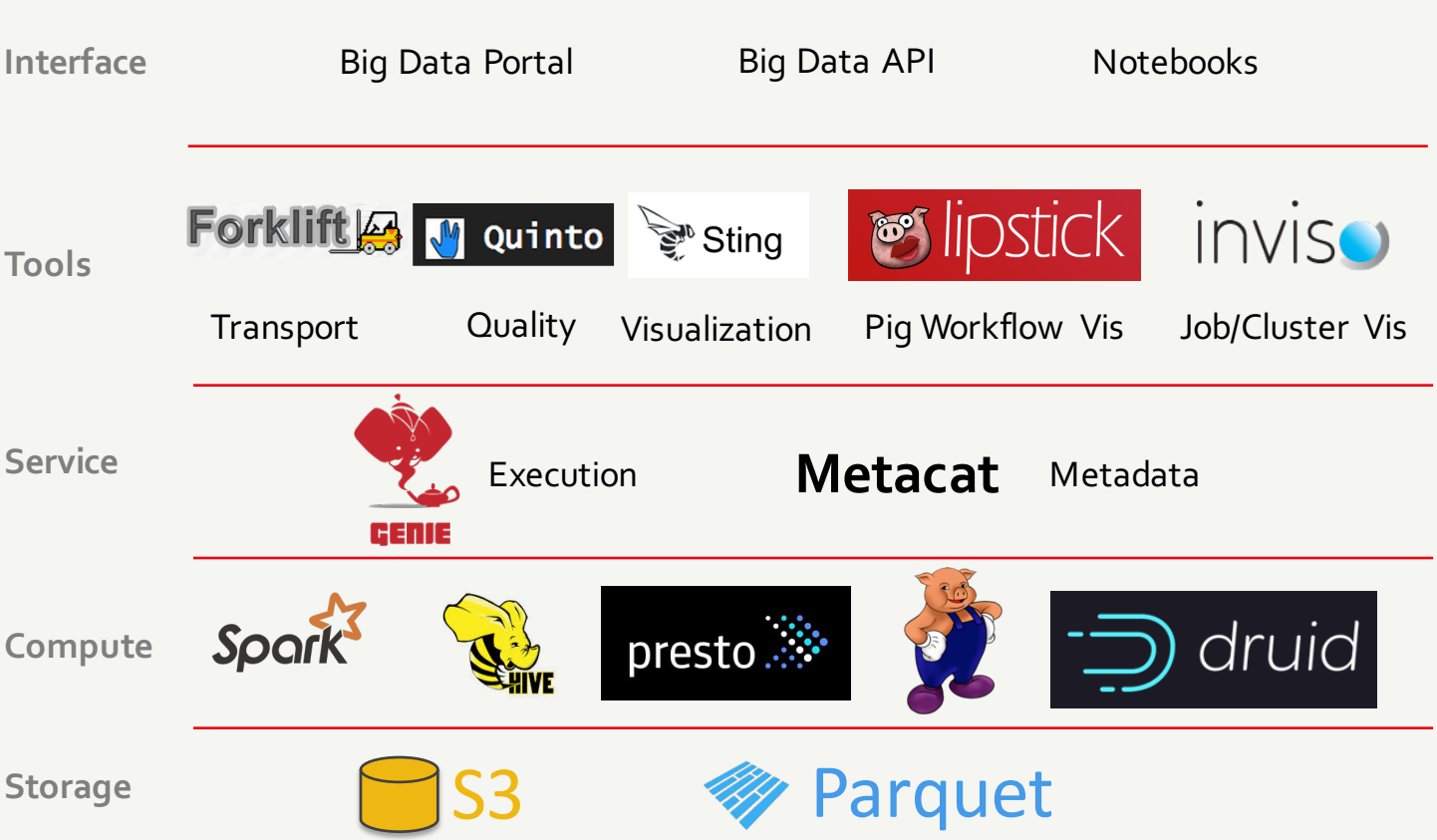
Outline

- Big Data Platform Architecture
- Technical Challenges
- ETL

Big Data Platform Architecture

Data Pipeline

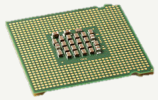



Interface
Big Data Portal
Big Data API
Notebooks
Tools
Forklift
Quinto
Sting
lipstick
invisO
Transport
Quality
Visualization
Pig Workflow Vis
Job/Cluster Vis
Service
GENIE
Execution
Metacat
Metadata
Compute
Spark
HIVE
presto
druid
Storage
S3
Parquet

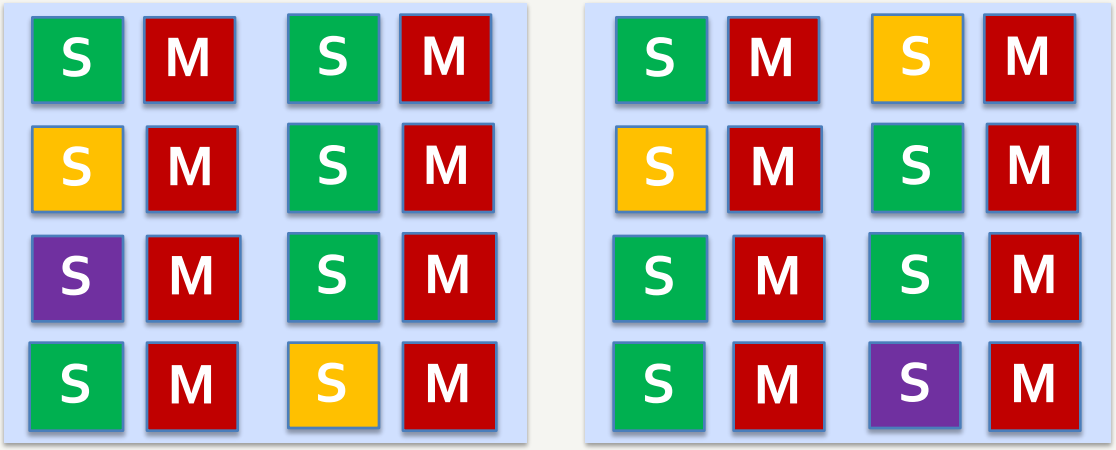
Spark on YARN at Netflix

- 3000 EC2 nodes on two clusters (d2.4xlarge)
- Multiple Spark versions
- Share the same infrastructure with MapReduce jobs

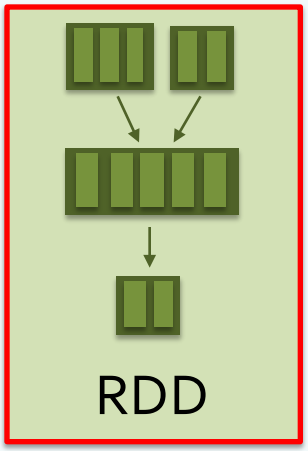
16 vcores



120 GB



Technical Challenges



Node Manager

Spark AM

Resource Manager

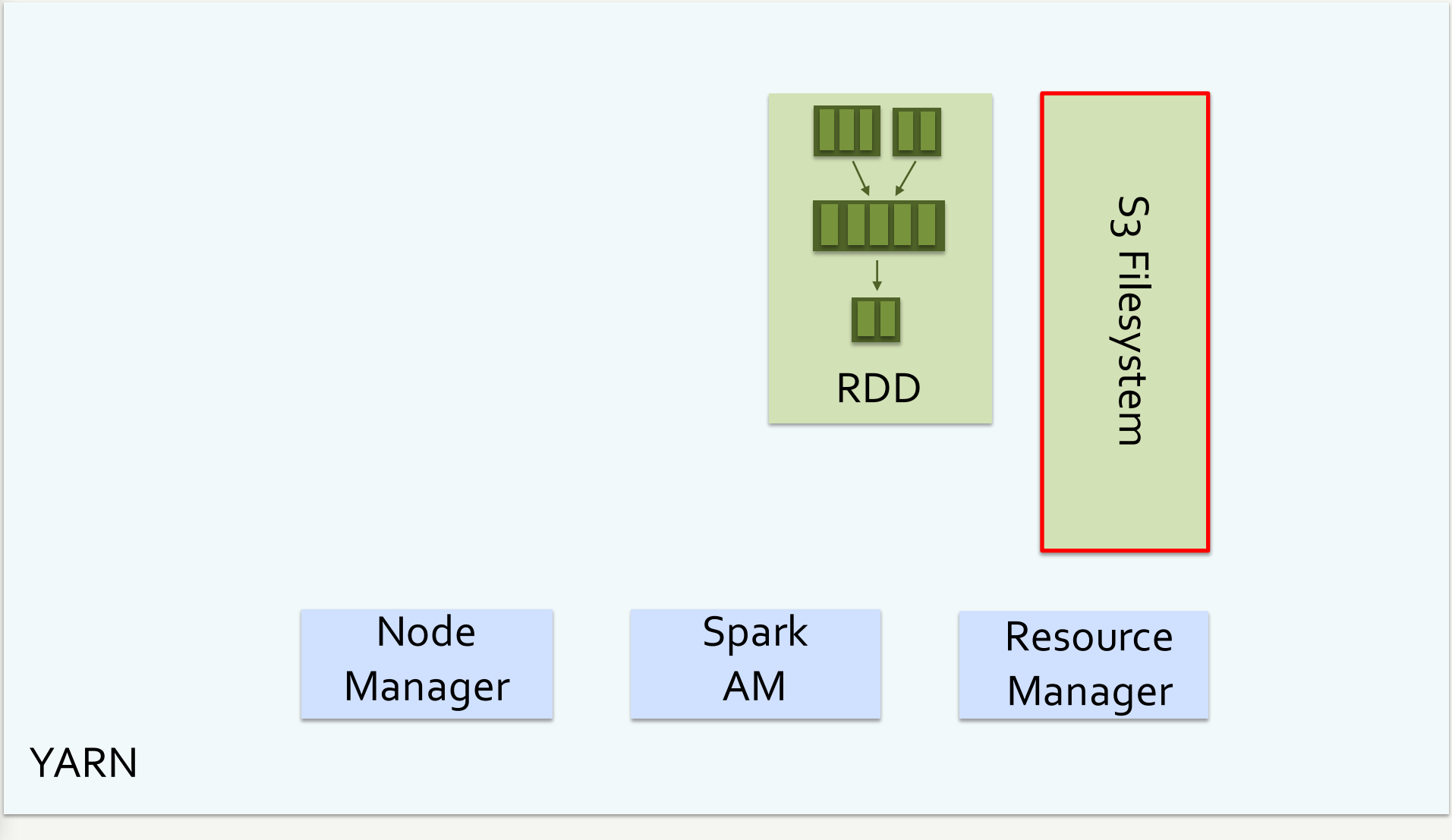
YARN

Custom Coalescer Support [SPARK-14042]

- coalesce() can only "merge" using the given number of partitions
  - how to merge by size?
- CombineFileInputFormat with Hive
- Support custom partition coalescing strategies

UnionRDD Parallel Listing [SPARK-9926]

- Parent RDD partitions are listed sequentially
- Slow for tables with lots of partitions
- Parallelize listing of parent RDD partitions


RDD
S3 Filesystem
Node Manager
Spark AM
Resource Manager
YARN

Optimize S3 Listing Performance [HADOOP-12810]

- Unnecessary getFileStatus() call
- SPARK-9926 and HADOOP-12810 yield faster startup
- ~20x speedup in input split calculation

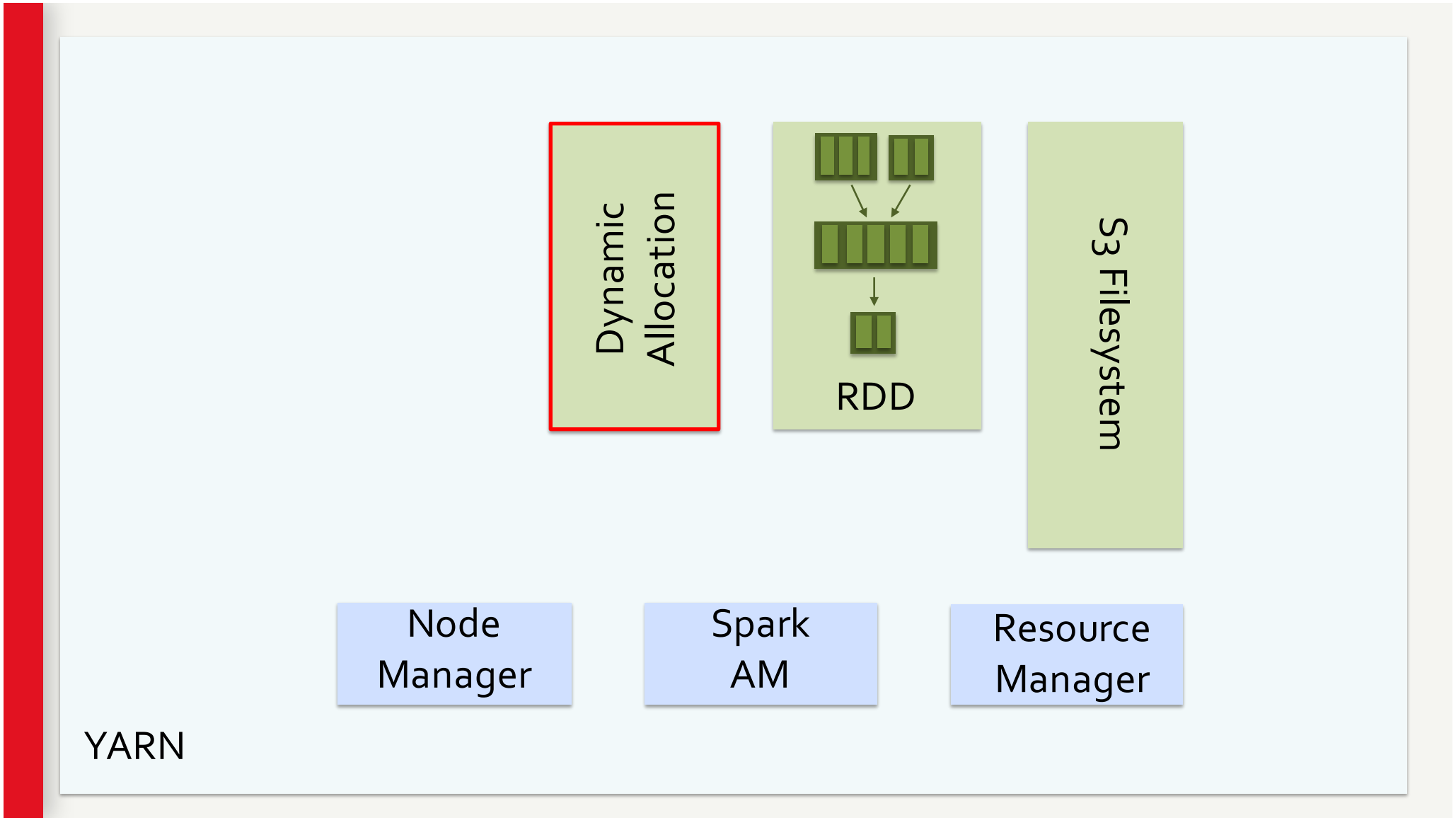
Output Committers

Hadoop Output Committer

- Write to a temp directory and rename to destination on success
  - S3 rename => copy + delete
  - S3 is eventually consistent

S3 Output Committer

- Write to local disk and upload to S3 on success
  - avoid redundant S3 copy
  - avoid eventual consistency


Dynamic Allocation
RDD
S3 Filesystem
Node Manager
Spark AM
Resource Manager
YARN

Poor Broadcast Read Performance [SPARK-13328]

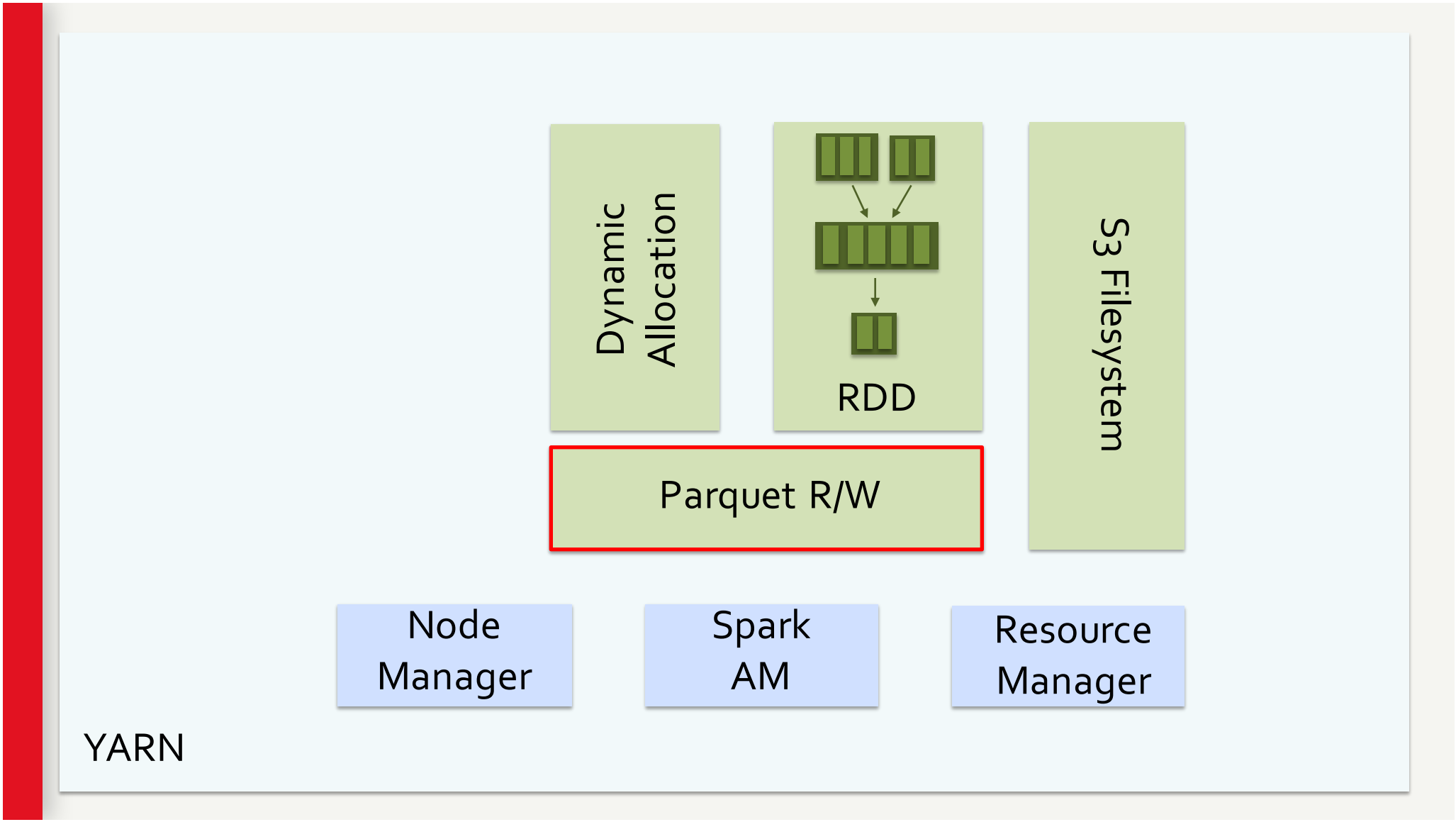
- Broadcast joins/variables
- Replicas can be removed with dynamic allocation

```
…
16/02/13 01:02:27 WARN BlockManager:
Failed to fetch remote block broadcast_18_pieceo (failed attempt 70)
…
16/02/13 01:02:27 INFO TorrentBroadcast:
Reading broadcast variable 18 took 1051049 ms
```

- Refresh replica locations from the driver on multiple failures

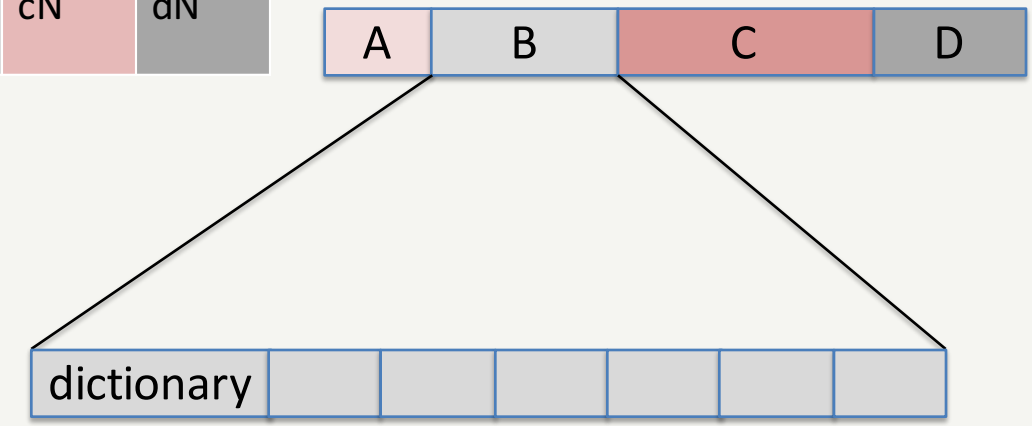
Incorrect Locality Optimization [SPARK-13779]

- Cancel & resend pending container requests
  - if the locality preference is no longer needed
  - if no locality preference is set
- No locality information with S3
- Do not cancel requests without locality preference


Dynamic Allocation
RDD
S3 Filesystem
Parquet R/W
Node Manager
Spark AM
Resource Manager
YARN

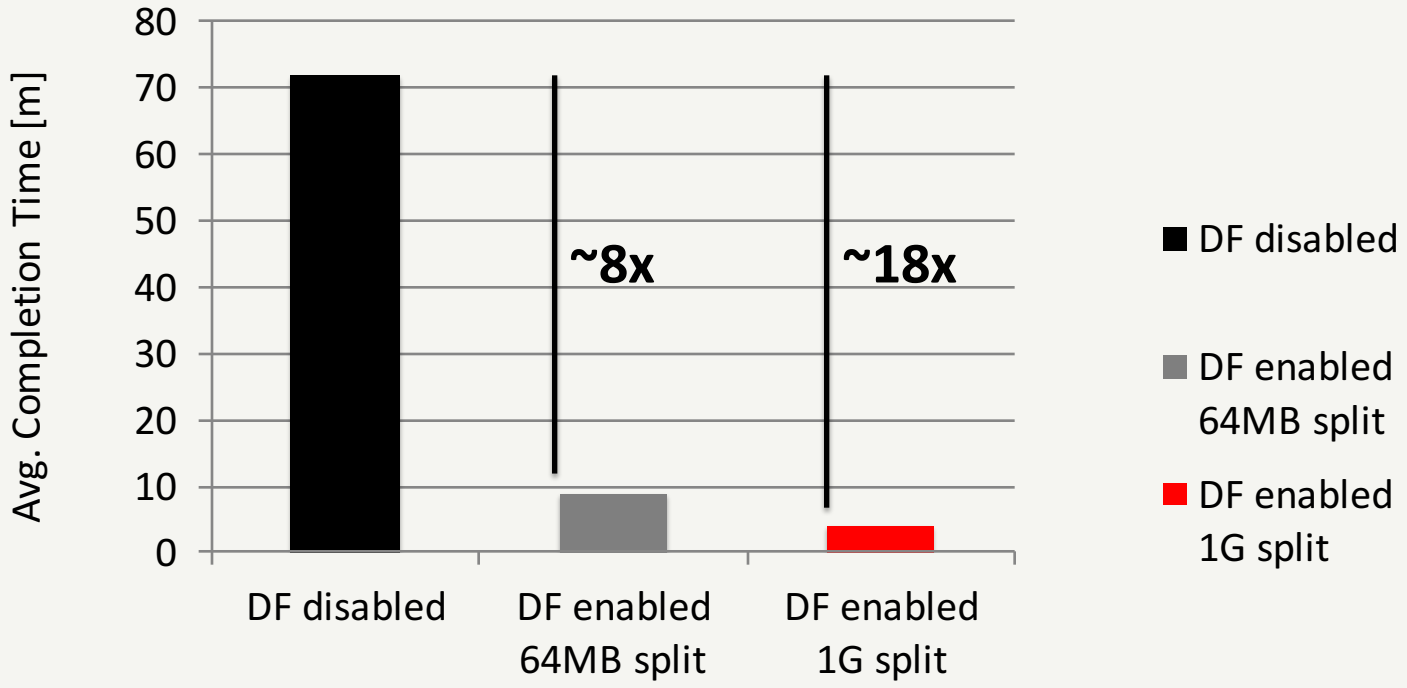
Parquet Dictionary Filtering [PARQUET-384*]

A	B	C	D
a1	b1	c1	d1
…	…	…	…
aN	bN	cN	dN



from “Analytic Data Storage in Hadoop”, Ryan Blue

Parquet Dictionary Filtering [PARQUET-384*]

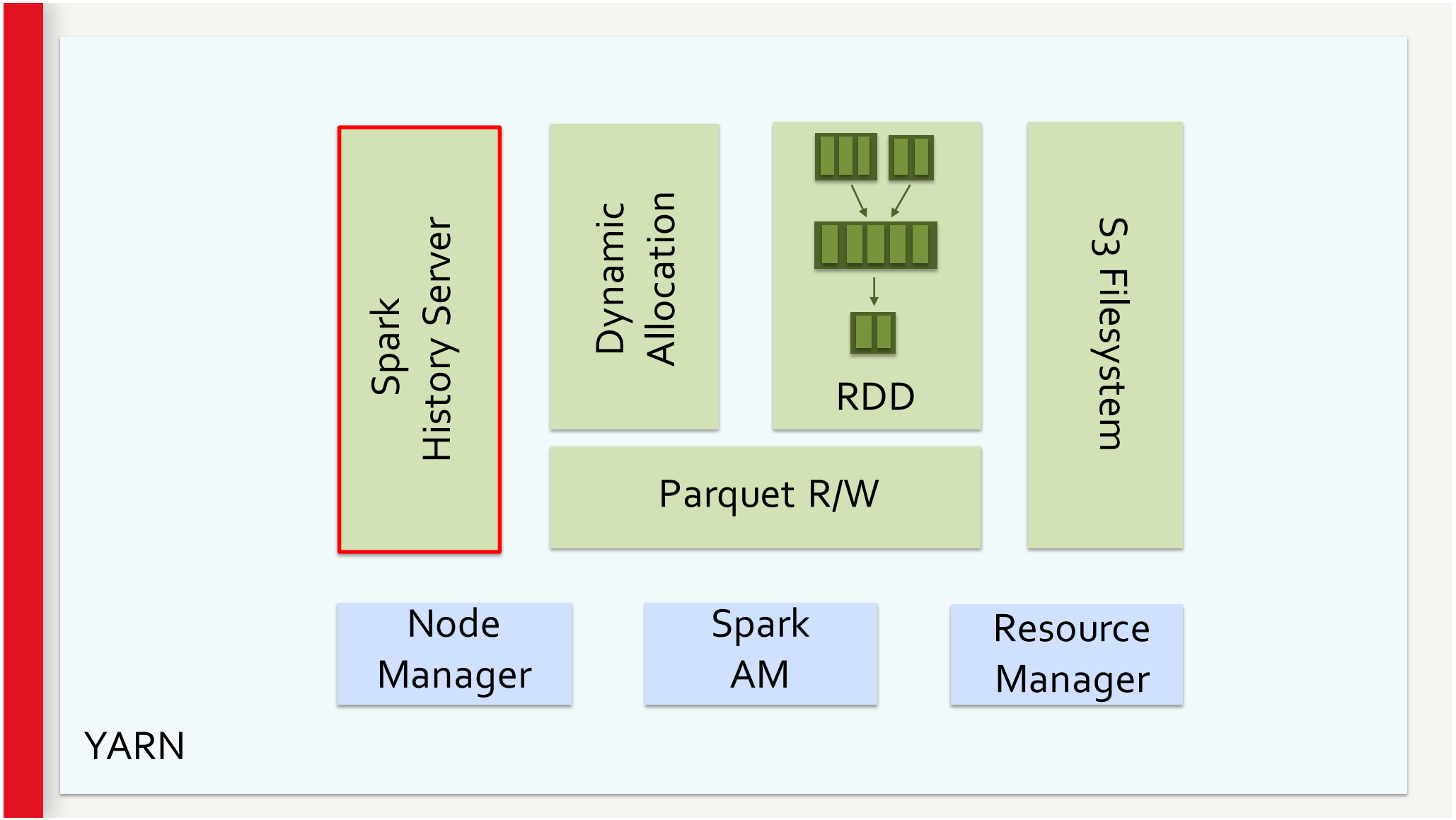


How to Enable Dictionary Filtering?

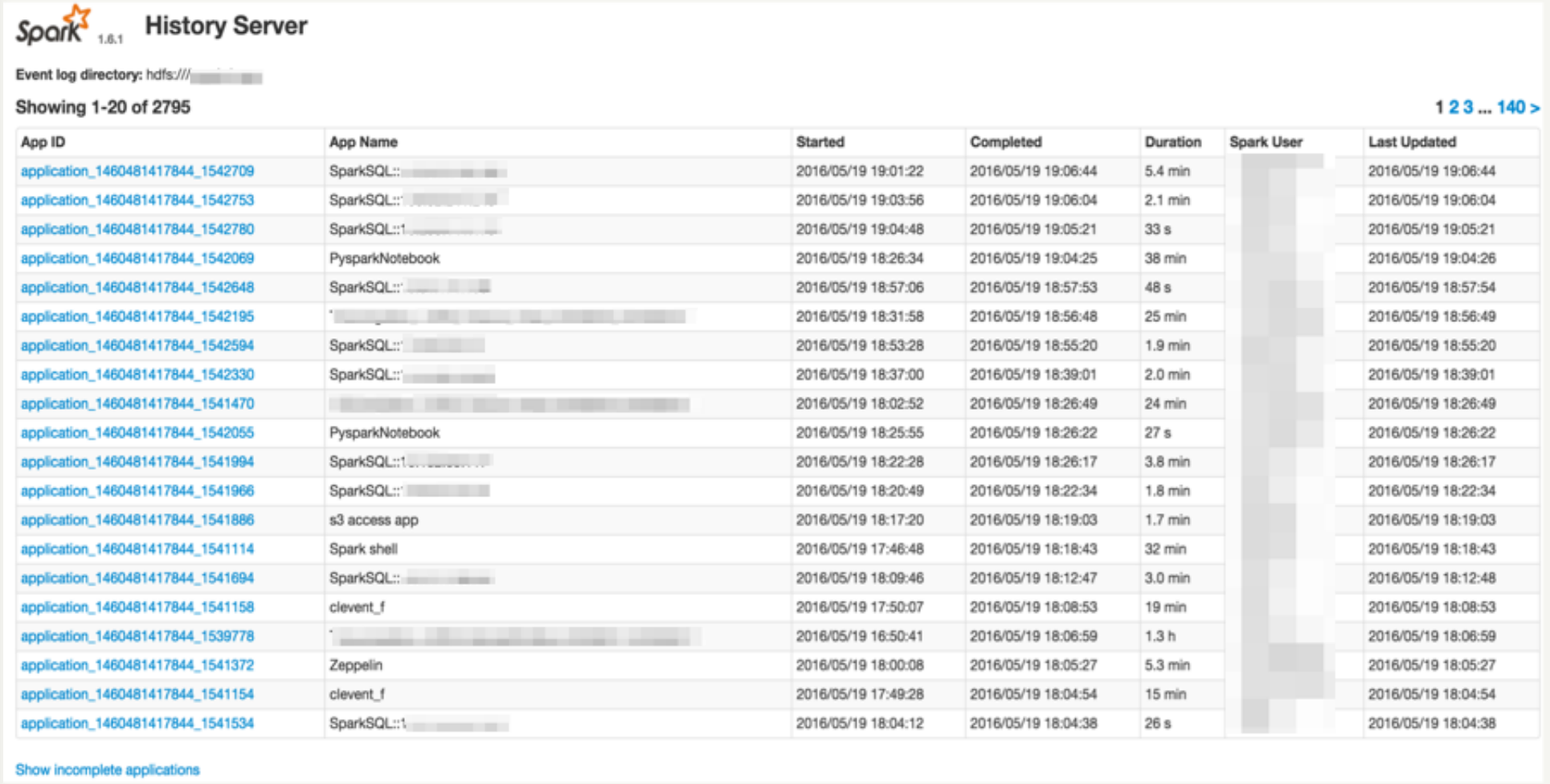
Property	Value	Description
spark.sql.hive.convertMetastoreParquet	true	enable native Parquet read path
parquet.filter.statistics.enabled	true	enable stats filtering
parquet.filter.dictionary.enabled	true	enable dictionary filtering
spark.sql.parquet.filterPushdown	true	enable Parquet filter pushdown optimization
spark.sql.parquet.mergeSchema	false	disable schema merging
spark.sql.hive.convertMetastoreParquet.mergeSchema	false	use Hive SerDe instead of built-in Parquet support

Efficient Dynamic Partition Inserts [SPARK-15420*]

- Parquet buffers row group data for each file during writes
- Spark already sorts before writes, but has some limitations
- Detect if the data is already sorted
- Expose the ability to repartition data before write


Spark History Server
Dynamic Allocation
RDD
S3 Filesystem
Parquet R/W
Node Manager
Spark AM
Resource Manager
YARN

Spark History Server – Where is My Job?



Spark 1.6.1 History Server

Event log directory: hdfs:///

Showing 1-20 of 2795

1 2 3 ... 140 >

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
application_1460481417844_1542709	SparkSQL::	2016/05/19 19:01:22	2016/05/19 19:06:44	5.4 min		2016/05/19 19:06:44
application_1460481417844_1542753	SparkSQL::	2016/05/19 19:03:56	2016/05/19 19:06:04	2.1 min		2016/05/19 19:06:04
application_1460481417844_1542780	SparkSQL::1	2016/05/19 19:04:48	2016/05/19 19:05:21	33 s		2016/05/19 19:05:21
application_1460481417844_1542069	PysparkNotebook	2016/05/19 18:26:34	2016/05/19 19:04:25	38 min		2016/05/19 19:04:26
application_1460481417844_1542648	SparkSQL::	2016/05/19 18:57:06	2016/05/19 18:57:53	48 s		2016/05/19 18:57:54
application_1460481417844_1542195	`	2016/05/19 18:31:58	2016/05/19 18:56:48	25 min		2016/05/19 18:56:49
application_1460481417844_1542594	SparkSQL::	2016/05/19 18:53:28	2016/05/19 18:55:20	1.9 min		2016/05/19 18:55:20
application_1460481417844_1542330	SparkSQL::	2016/05/19 18:37:00	2016/05/19 18:39:01	2.0 min		2016/05/19 18:39:01
application_1460481417844_1541470		2016/05/19 18:02:52	2016/05/19 18:26:49	24 min		2016/05/19 18:26:49
application_1460481417844_1542055	PysparkNotebook	2016/05/19 18:25:55	2016/05/19 18:26:22	27 s		2016/05/19 18:26:22
application_1460481417844_1541994	SparkSQL::1	2016/05/19 18:22:28	2016/05/19 18:26:17	3.8 min		2016/05/19 18:26:17
application_1460481417844_1541966	SparkSQL::	2016/05/19 18:20:49	2016/05/19 18:22:34	1.8 min		2016/05/19 18:22:34
application_1460481417844_1541886	s3 access app	2016/05/19 18:17:20	2016/05/19 18:19:03	1.7 min		2016/05/19 18:19:03
application_1460481417844_1541114	Spark shell	2016/05/19 17:46:48	2016/05/19 18:18:43	32 min		2016/05/19 18:18:43
application_1460481417844_1541694	SparkSQL::	2016/05/19 18:09:46	2016/05/19 18:12:47	3.0 min		2016/05/19 18:12:48
application_1460481417844_1541158	clevent_f	2016/05/19 17:50:07	2016/05/19 18:08:53	19 min		2016/05/19 18:08:53
application_1460481417844_1539778	`	2016/05/19 16:50:41	2016/05/19 18:06:59	1.3 h		2016/05/19 18:06:59
application_1460481417844_1541372	Zeppelin	2016/05/19 18:00:08	2016/05/19 18:05:27	5.3 min		2016/05/19 18:05:27
application_1460481417844_1541154	clevent_f	2016/05/19 17:49:28	2016/05/19 18:04:54	15 min		2016/05/19 18:04:54
application_1460481417844_1541534	SparkSQL::1	2016/05/19 18:04:12	2016/05/19 18:04:38	26 s		2016/05/19 18:04:38

Show incomplete applications

Spark History Server – Where is My Job?

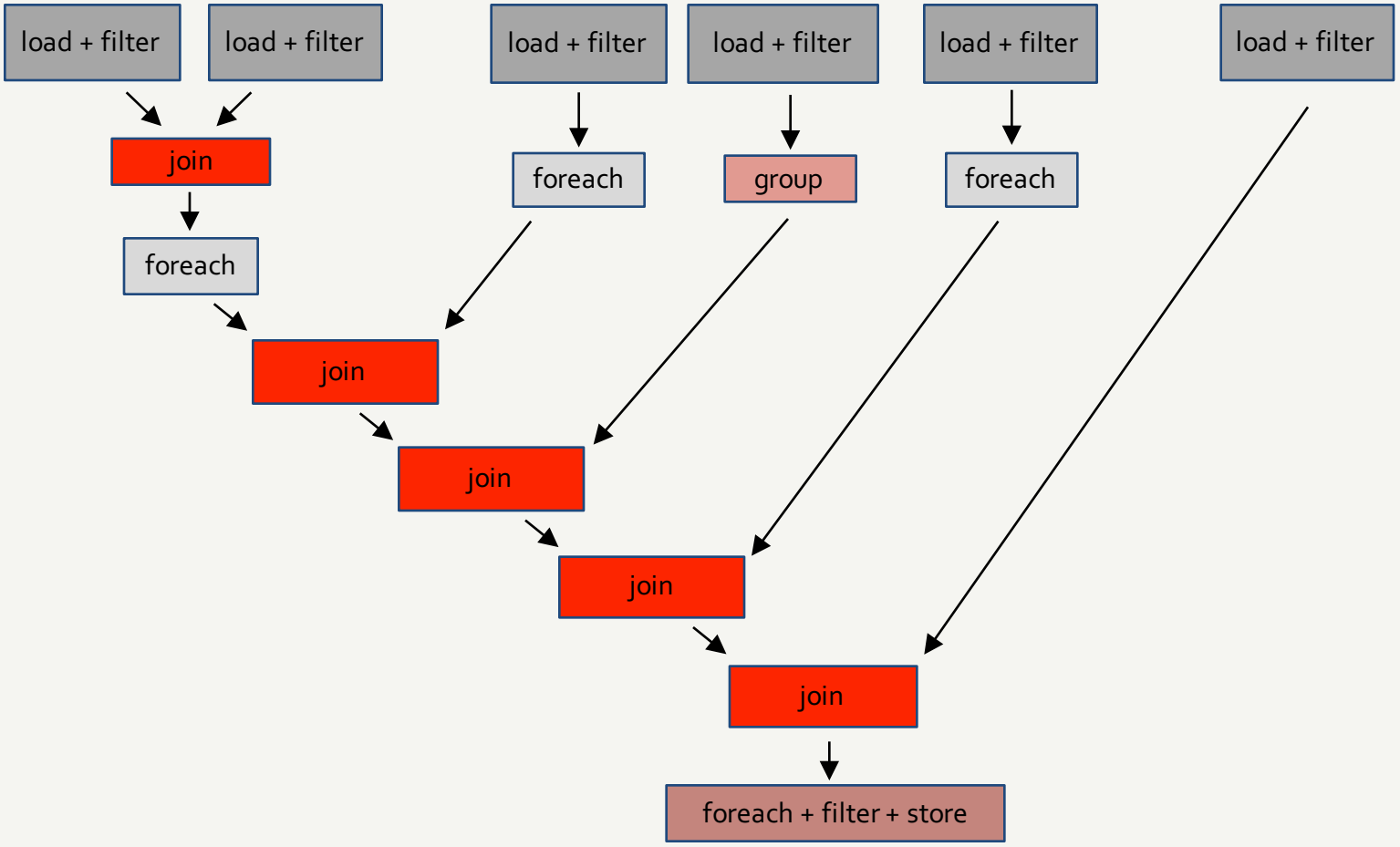
- A large application can prevent new applications from showing up
  - not uncommon to see event logs of GBs
- SPARK-13988 makes the processing multi-threaded
- GC tuning helps further
  - move from CMS to G1 GC
  - allocate more space to young generation

Extract

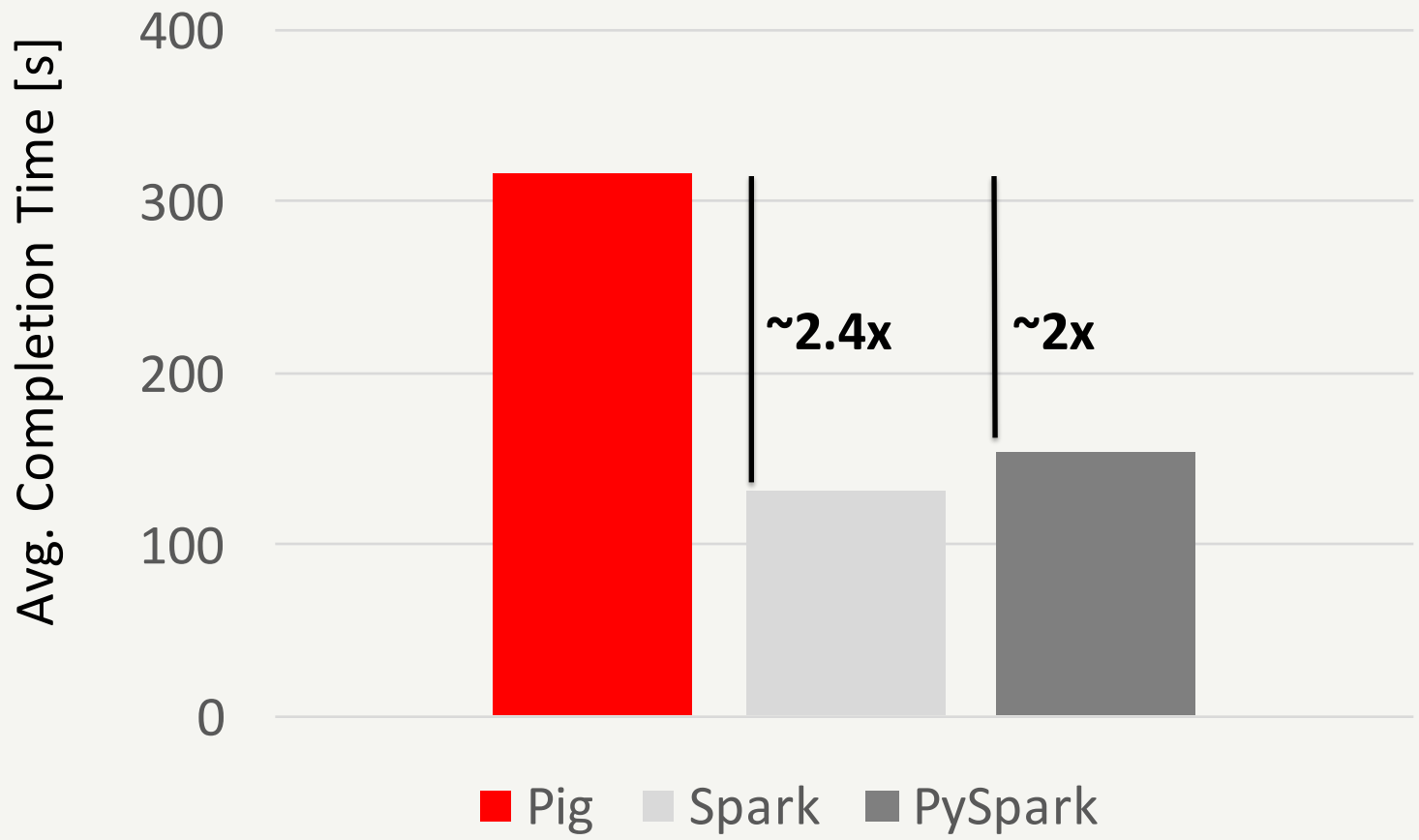
Transform

Load

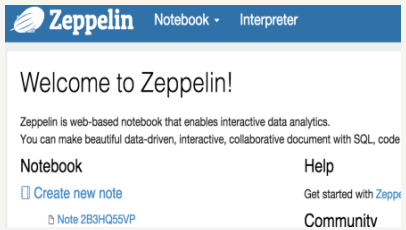
Pig vs. Spark

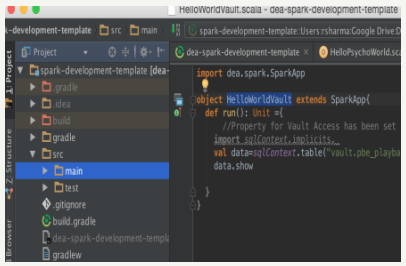

load + filter
load + filter
load + filter
load + filter
load + filter
load + filter
join
foreach
group
foreach
foreach
join
join
join
join
foreach + filter + store

Pig vs. Spark (Scala) vs. PySpark



Production Workflow





Prototype

Build

Deploy

GENIE

Run

Production Spark Application #1: Yogen

- A rapid innovation platform for targeting algorithms
- **5 hours** (vs. 10s of hours) to compute similarity for all Netflix profiles for 30-day window of new arrival titles
- **10 minutes** to score 4M profiles for 14-day window of new arrival titles

Production Spark Application #2: ARO

- Personalized ordering of rows of titles
- Enrich page/row/title features with play history
- 14 stages, ~10Ks of tasks, several TBs

What's Next?

- Improved Parquet support
- Better visibility
- Explore new use cases


Questions?