Integrating Spark at Petabyte Scale

Daniel C. Weeks

NETFLIX


PLAN?
YES
NO
back up!
RESULTS:
W.C.C
USE ASSETS
Payback
proceed to back up
STEPS: (TO STORAGE)

NETFLIX ignite | Test case XYZ | help

Test case XYZ

(metrics current through 10/31)

contact DL ignite

Allocation Dates: 09/25/2014 10/30/2014 Select Report: Retention & Streaming

ALLOCATION FILTERS

All | United States | Canada | Brazil | Mexico | Latin America - Other | UK | Nordics | Ireland | Benelux | France | DACH | Allocation Type: All

ACTIVITY FILTERS

Activity Window: 126 Days | All Devices

Add filters...

Display Cell	1	2	3	4	5
Cell Name	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
Comparison Cell (Set All)	1	1	1	1	1
# of Allocations	107,377	107,149	107,513	107,590	107,284
Streaming Score	N.A.	N.A.	N.A.	N.A.	N.A.
% Accounts with > 0 Hours	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
% Accounts with >= 1 Hour	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
% Accounts with >= 5 Hours	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
% Accounts with >= 10 Hours	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
% Accounts with >= 20 Hours	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

NETFLIX

USA ISP SPEED INDEX

SELECT COUNTRY

JULY 2015

LIST VIEW GRAPH VIEW

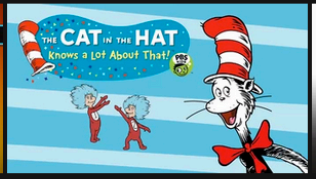
RANK	CHANGE	TYPE	ISP NAME	AVG SPEED (Mbps)
1	–		COX	3.62
2	–		CABLEVISION - OPTIMUM	3.59
3	–		VERIZON - FIOS	3.54
4	+2		CHARTER	3.46
5	-1		COMCAST	3.45
6	+1		BRIGHT HOUSE	3.42
7	-2		SUDDENLINK	3.42
8	+1		TIME WARNER CABLE	3.37
9	-1		MEDIACOM	3.32
10	–		AT&T - U-VERSE	3.20
11	–		AT&T - DSL	2.51

NETFLIX Browse Kids DVD Search eva

Suspenseful TV Shows

Exciting Sci-Fi & Fantasy

Goofy TV Shows



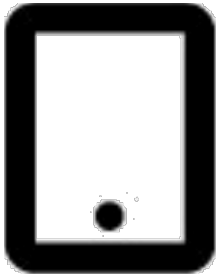
Action & Adventure

Our Biggest Challenge is Scale

Netflix Key Business Metrics

65+ million
members

50 countries



1000+ devices
supported

10 billion
hours / quarter

Global Expansion

200 countries by end of 2016

Big Data Size

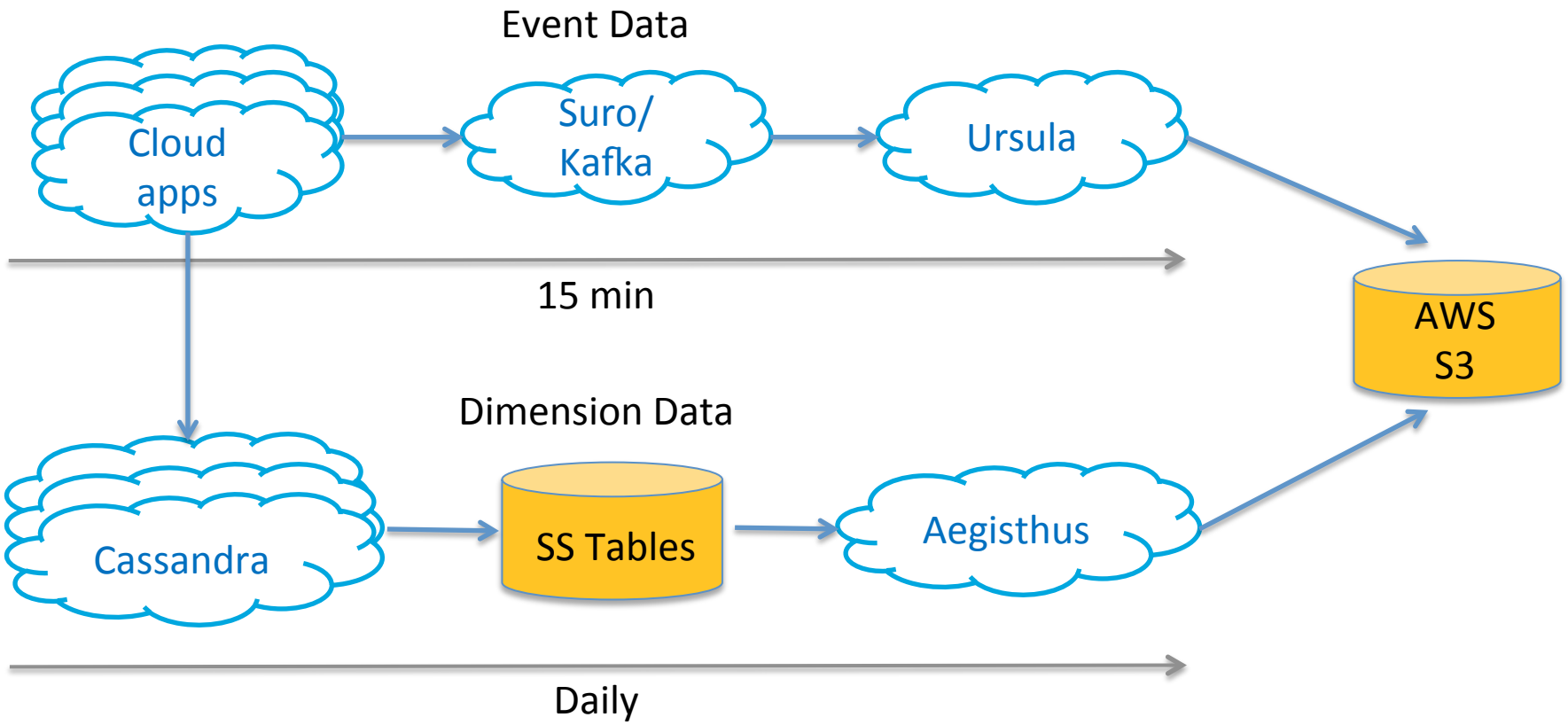
Total ~20 PB DW on S3
Read ~10% DW daily
Write ~10% of read data daily

~ 500 billion events daily

~ 350 active platform users (~100 analysts)

Architecture Overview

Data Pipelines



Storage

Compute

Service

Tools

Parquet FF

(EMR Hadoop clusters)

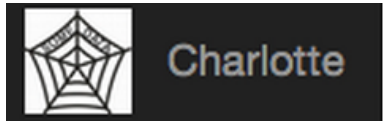
(Federated execution service)

Metacat

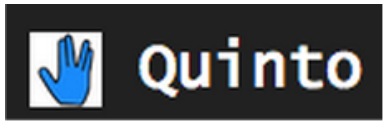
(Federated metadata service)

Data movement

Data visualization



Data lineage



Data quality

Pig workflow visualization

Job/Cluster perf visualization

S3 as our DW Storage

S3 as single source of truth (not HDFS)

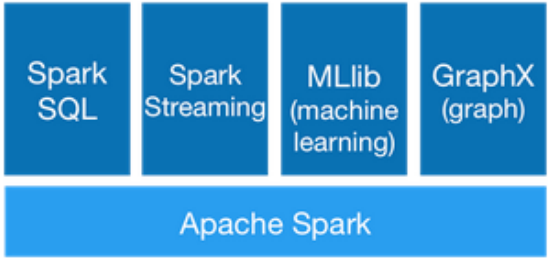
- 11 9's durability and 4 9's availability
- Separate compute and storage

Enables

- Multiple heterogeneous clusters
- Easy upgrade via r/b deployment

Spark

Apache Spark™ is a fast and general engine for large-scale data processing.



Why Spark?

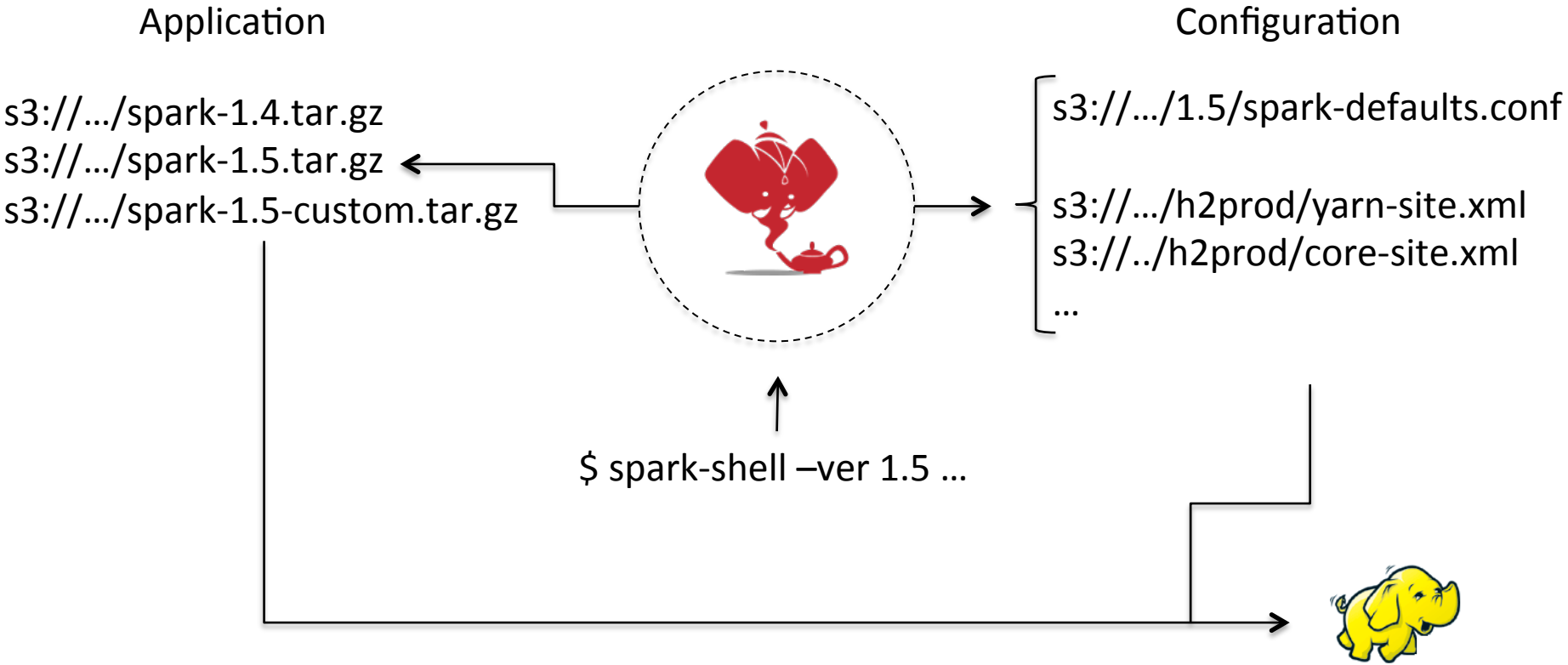
- Batch jobs (Pig, Hive)
  - ETL jobs
  - Reporting and other analysis
- Interactive jobs (Presto)
- Iterative ML jobs (Spark)
- Programmatic Use Cases

Deployments @ Netflix

- Spark on Mesos
  - Self-service AMI
  - Full BDAS (**B**erkeley **D**ata **A**nalytics **S**tack)
  - Online streaming analytics

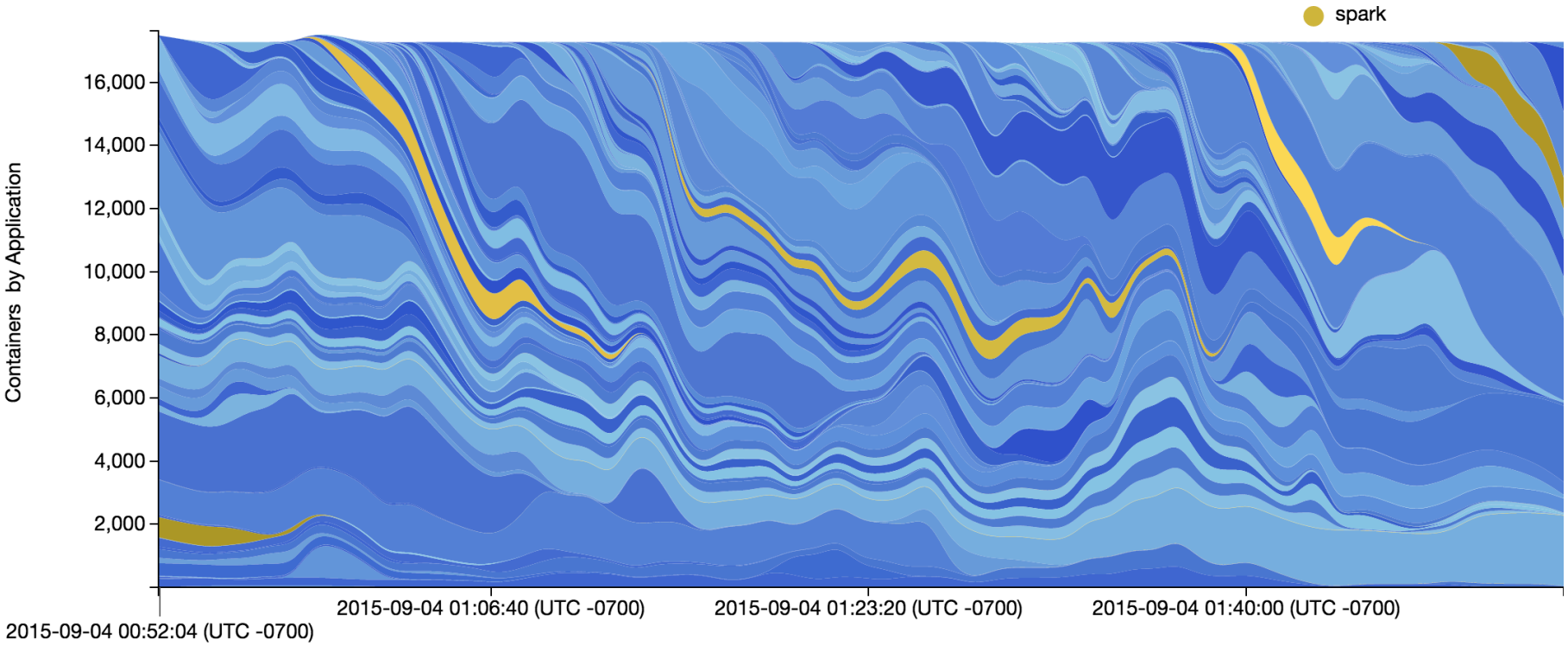
- Spark on YARN
  - Spark as a service
  - YARN application on Hadoop
  - Offline batch analytics

Version Support



Multitenancy

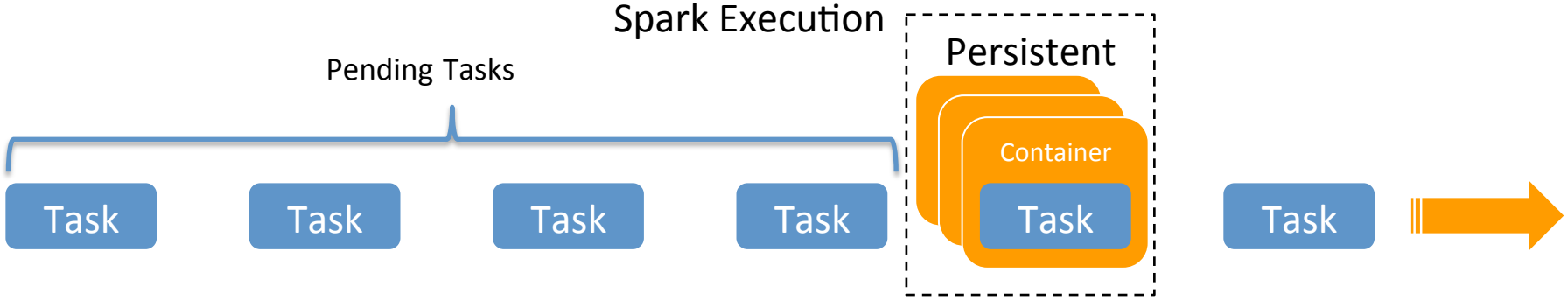
Multitenancy

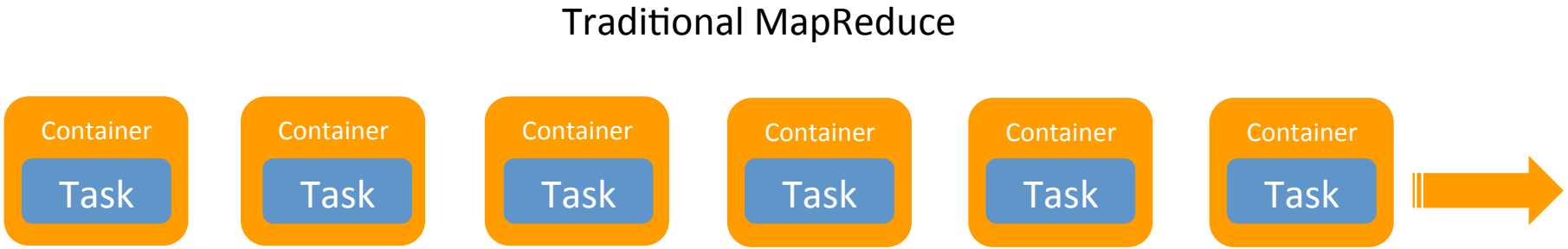


Dynamic Allocation on YARN

- Optimize for Resource Utilization
- Harness Cluster's Scale
- Still Provide Interactive Performance

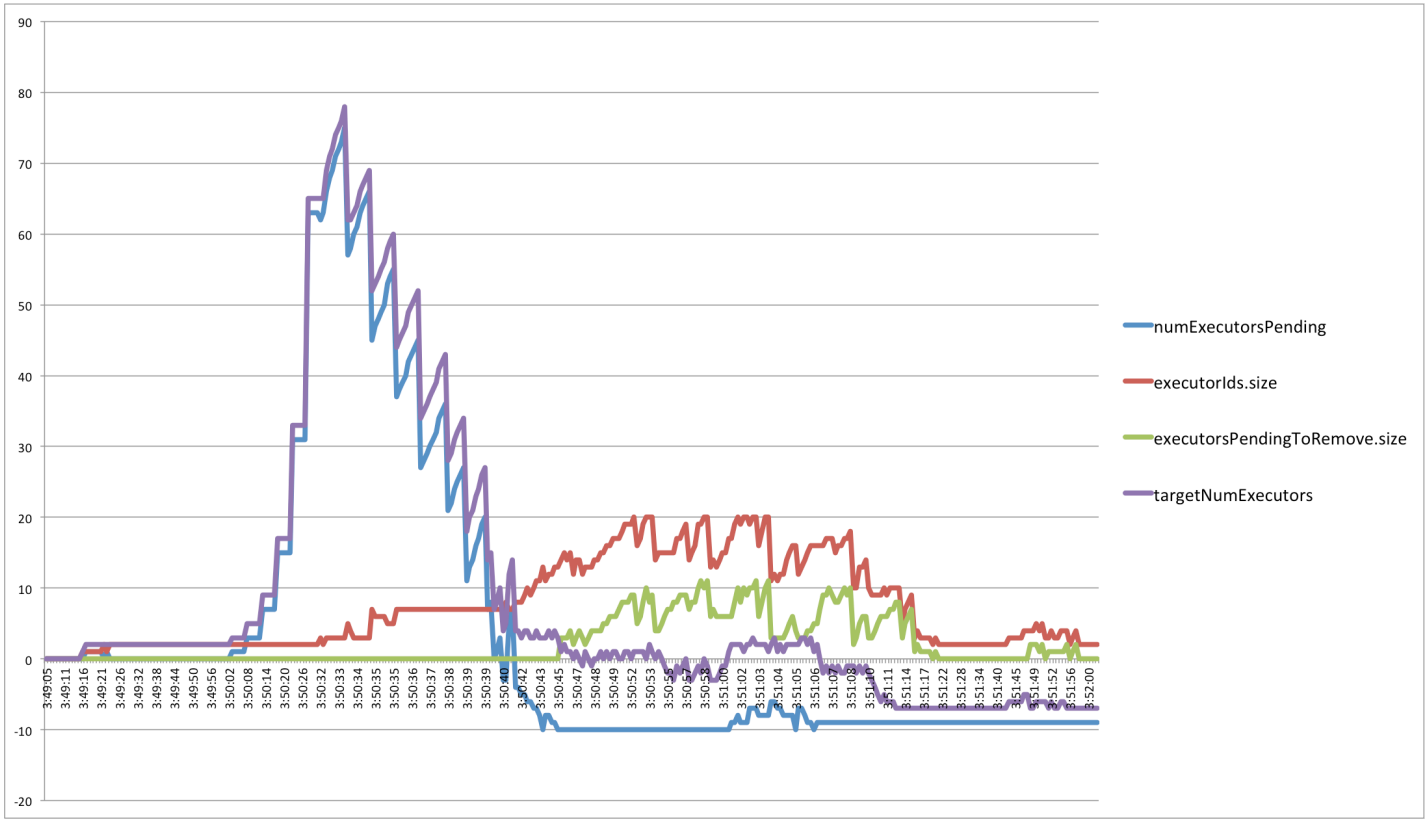
Task Execution Model





Dynamic Allocation

[SPARK-6954]

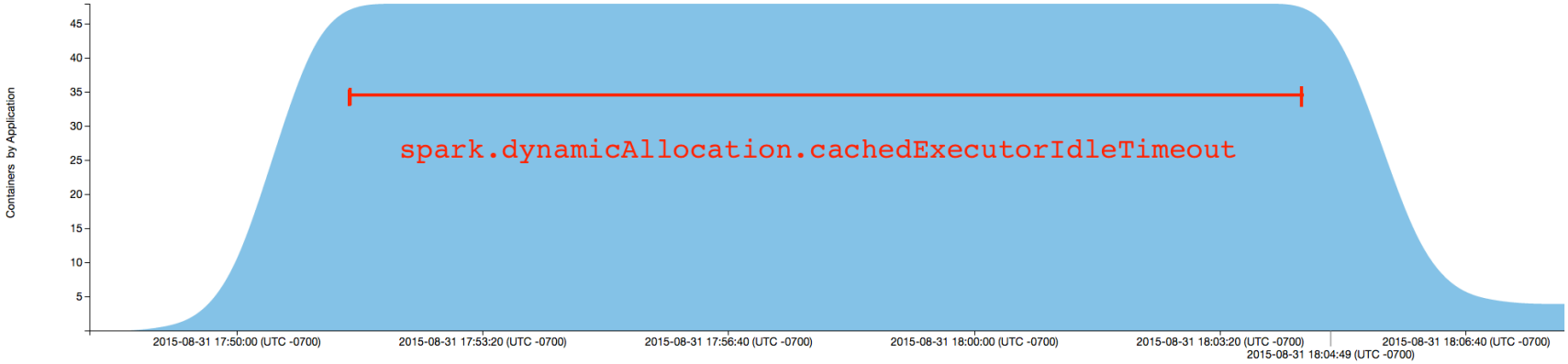


Cached Data

- Spark allows data to be cached
  - Interactive Reuse of Dataset
  - Iterative Usage (ML)

- Dynamic Allocation
  - Removes Executors when no tasks are Pending

Cached Executor Timeout [SPARK-7955]




```
val data = sqlContext
    .table("dse.admin_genie_job_d”)
    .filter($"dateint">=20150601 and $"dateint"<=20150830)
data.persist
data.count
```

Preemption

[SPARK-8167]

Problem

- Spark tasks randomly fail with “executor lost” error

Cause

- YARN preemption is not graceful

Solution

- Preempted tasks shouldn’t be counted as failures but should be retried

Reading / Processing / Writing

Partition Pruning

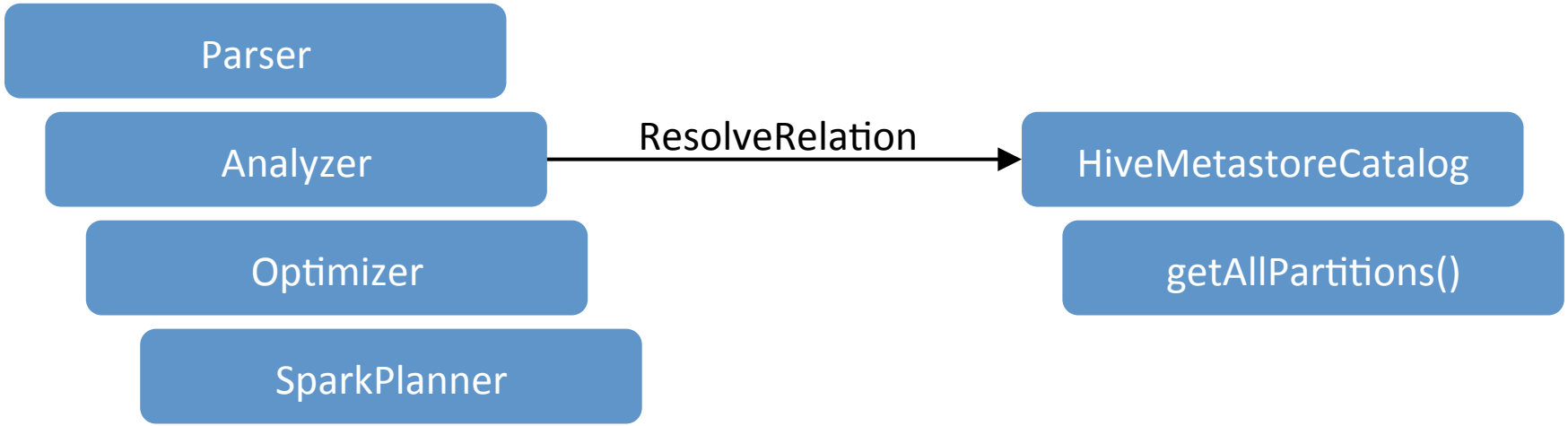
[SPARK-6910]

Problem: Metadata is Big Data

- Tables with millions of partitions
- Partitions with hundreds of files each

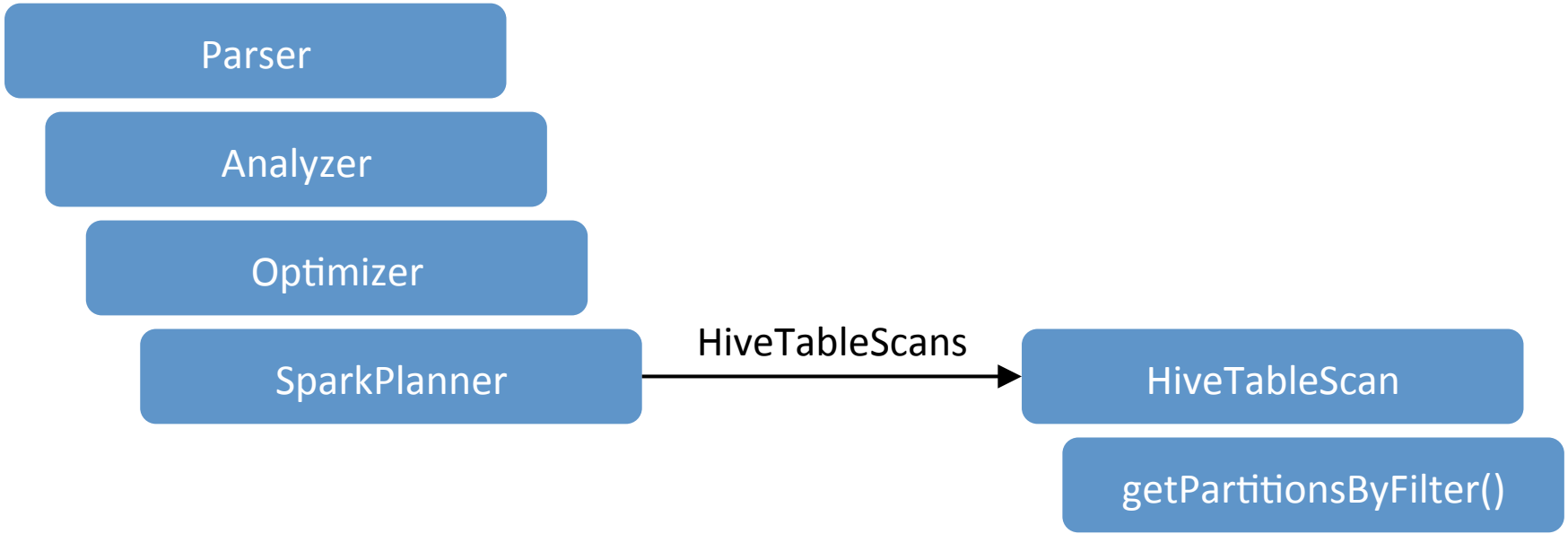
Client processes all partitions locally

Predicate pushdown for metadata



What if your table has 1.6M partitions?

Predicate pushdown for metadata



AWS S3 Listing Optimization

Problem: Metadata is Big Data

- Tables with millions of partitions
- Partitions with hundreds of files each

Clients Take a Long Time to Launch Jobs

Input split computation

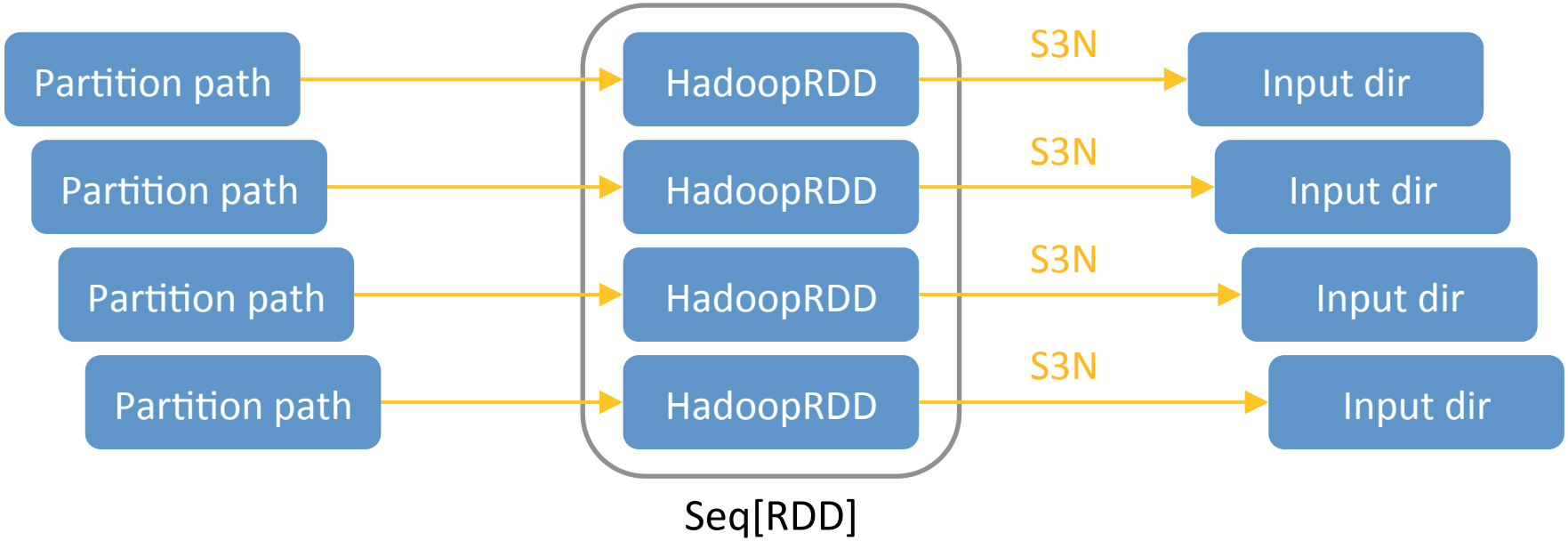
Parallelize:

`mapreduce.input.fileinputformat.list-status.num-threads`

- The number of threads to use list and fetch block locations for the specified input paths.

Setting this property in Spark jobs doesn't help

File listing for partitioned table



Sequentially listing input dirs via S3N file system.

SPARK-9926, SPARK-10340

Problem

- Input split computation for partitioned Hive table on S3 is slow

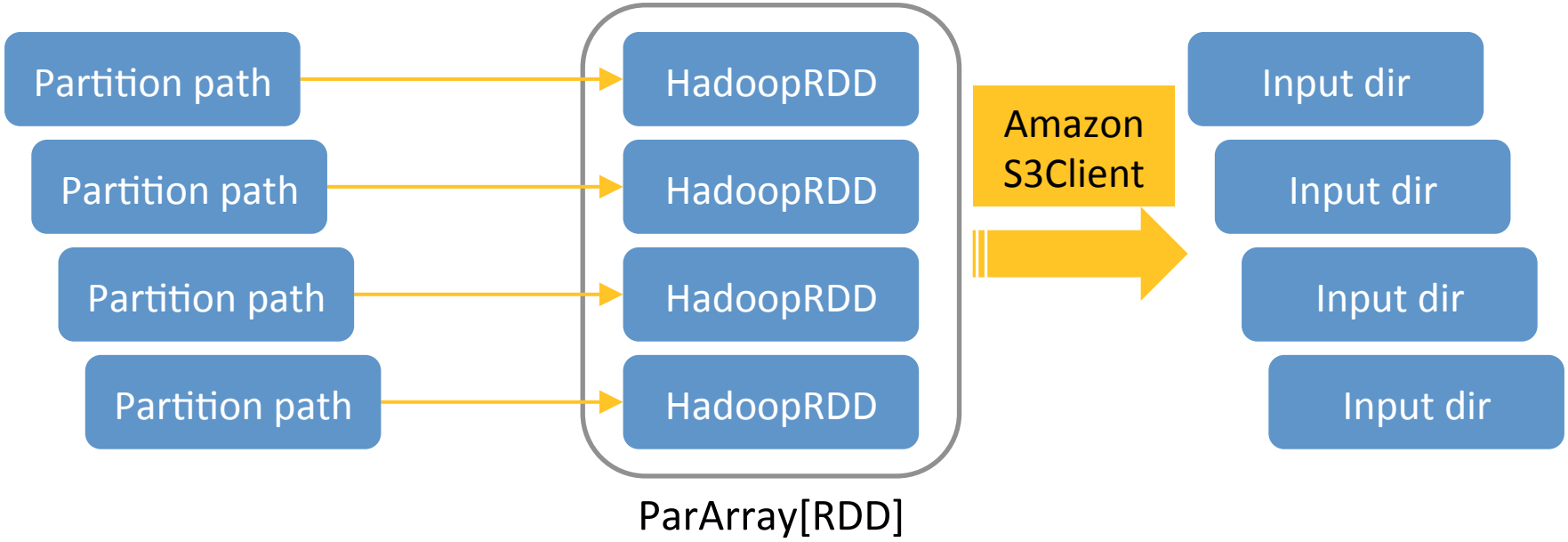
Cause

- Listing files on a per partition basis is slow
- S3N file system computes data locality hints

Solution

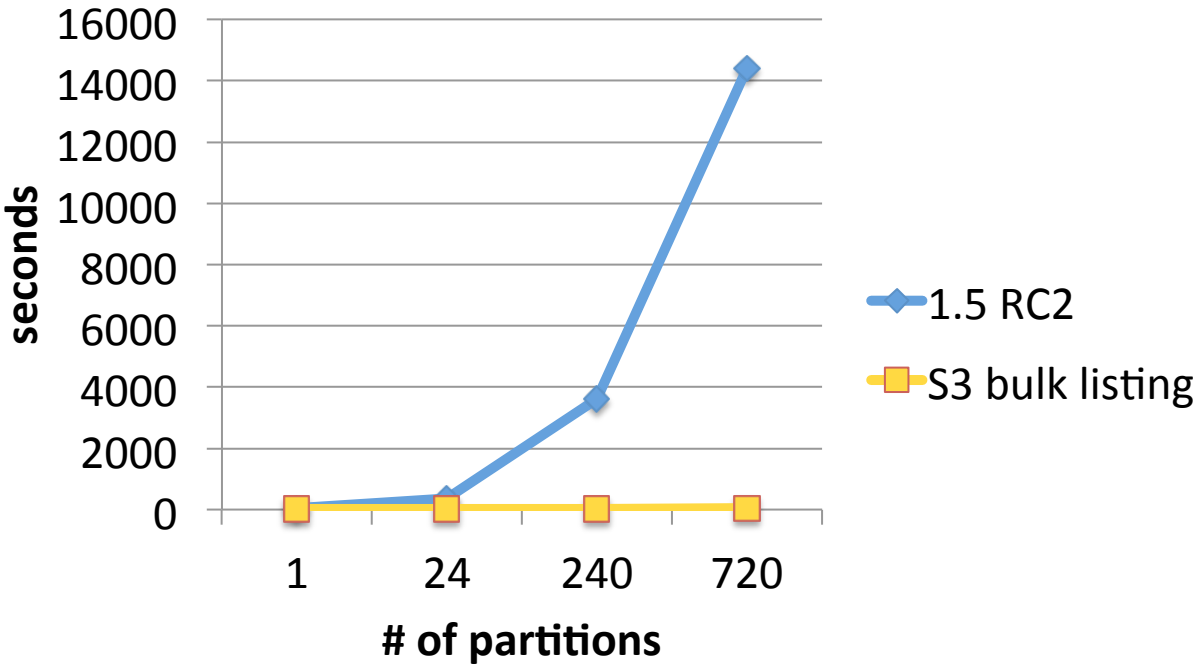
- Bulk list partitions in parallel using AmazonS3Client.
- Bypass data locality computation for S3 objects.

S3 Bulk Listing



S3 listing input dirs in parallel via AmazonS3Client.

Performance Improvement




```
SELECT * FROM nccp_log WHERE dateint=20150801 and hour=0 LIMIT 10;
```

Hadoop Output Committer

How it works

- Each task writes output to a temp dir.
- Output committer renames first successful task's temp dir to final destination

Challenges with S3

- S3 rename is copy and delete (non-atomic)
- S3 is eventual consistent

S3 output committer

How it works

- Each task writes output to local disk
- Output committer copies first successful task's output to S3

Advantages

- Avoid redundant S3 copy
- Avoid eventual consistency
- Always write to new paths

Our contributions

SPARK-6018
SPARK-6662
SPARK-6909
SPARK-6910
SPARK-7037
SPARK-7451
SPARK-7850
SPARK-8355
SPARK-8572
SPARK-8908
SPARK-9270
SPARK-9926
SPARK-10001
SPARK-10340

Next Steps for Netflix Integration

Metrics

Data Lineage

Parquet Integration


THANK YOU
NETFLIX