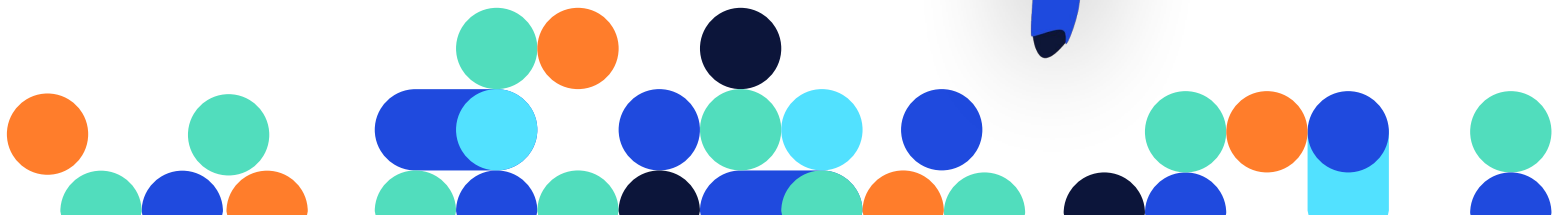
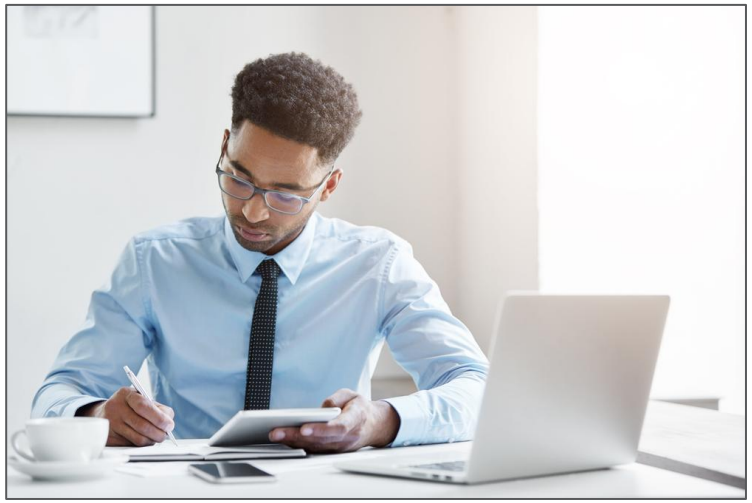
DataStax Developer Day

DATASTAX

DSE Analytics



What is your data telling you?



Top Uploaders

```
from cassandra.cluster import Cluster

cluster = Cluster(['172.31.28.68'])
session = cluster.connect("killrvideo")

d = session.execute("select * from user_videos")

counts = {}

for row in d:
  if(row.userid not in counts):
    counts[row.userid] = 1
  else:
    counts[row.userid] += 1

largestCount = 0
largestKey = ""
for key in counts:
  if(counts[key] > largestCount):
    largestCount = counts[key]
    largestKey = key

print("Largest count: " + str(largestCount))
print("Largest user id: " + str(largestKey))
user = session.execute("select firstname, lastname from users where userid = " + str(largestKey))[0]
print("Largest user name: " + user.firstname + " " + user.lastname)
```


WHAT
DA?

Can’t immediately tell what it’s doing
Pulls data from the cluster to a client machine
Execution on one core
Pain to write, pain to read

Think of the children!

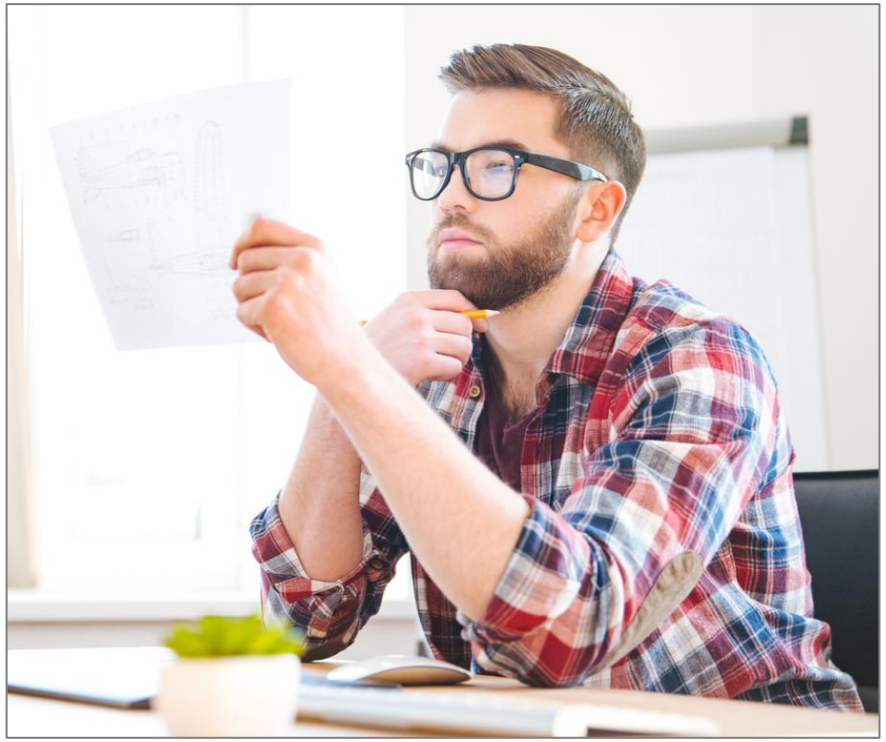
SQL...yes... S.Q.L.

```
SELECT firstname, lastname, count(*)
AS uploadCount
FROM users NATURAL JOIN user_videos
GROUP BY userid, firstname, lastname
ORDER BY uploadCount DESC
LIMIT 10;
```

OLTP vs. OLAP

I want it, and I want it now!

Cassandra



Hmmmmmm...

DSE Analytics

Transactional vs. Analytical

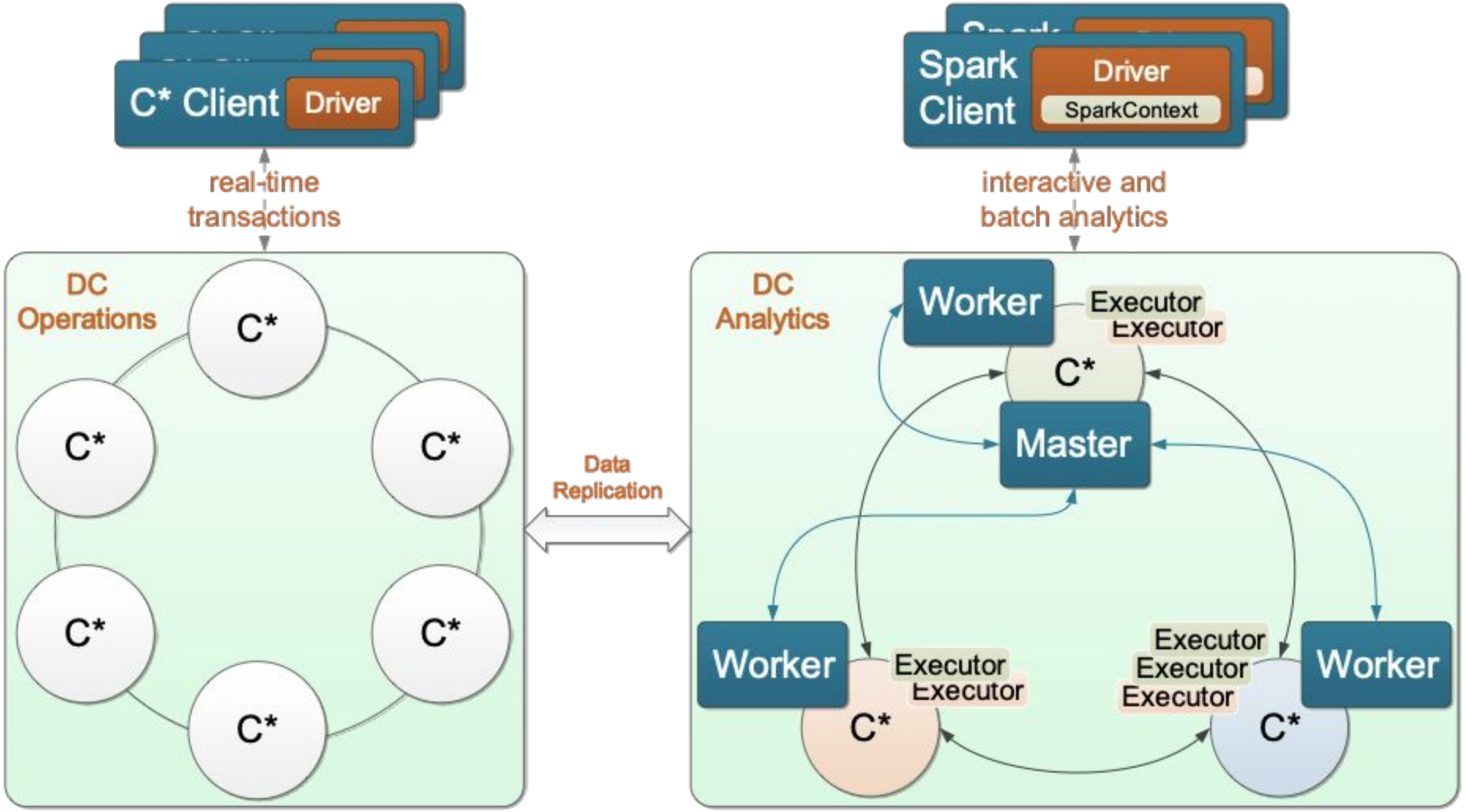
- Transactional
  - Fast
  - Short transactions
  - Many of them
  - INSERT, UPDATE, DELETE
  - Real time

- Analytical
  - Slow
  - Aggregations
  - Few of them
  - INSERT, UPDATE, DELETE
  - Seconds, minutes, hours

DataStax Enterprise Analytics

- Apache Spark™
- Integrated
- Start your cluster with `dse -k`
- Or set it in the configuration file

DSE Analytics - Recommended Deployment



Spark-sql – getting started

- Create a terminal window in CHE
- Start the shell

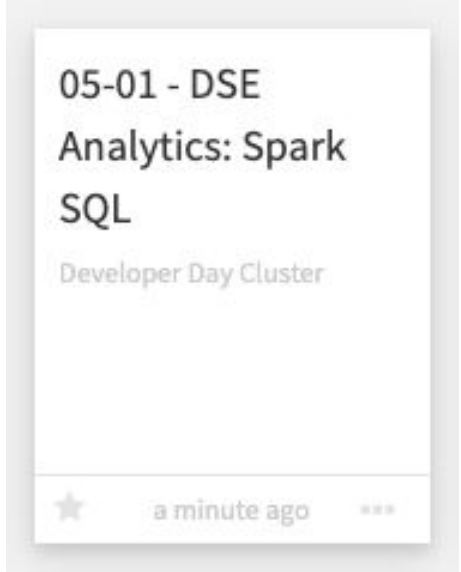
```
ubuntu@dse-node:~$ dse spark-sql
The log file is at
/home/ubuntu/.spark-sql-shell.log
spark-sql>
```

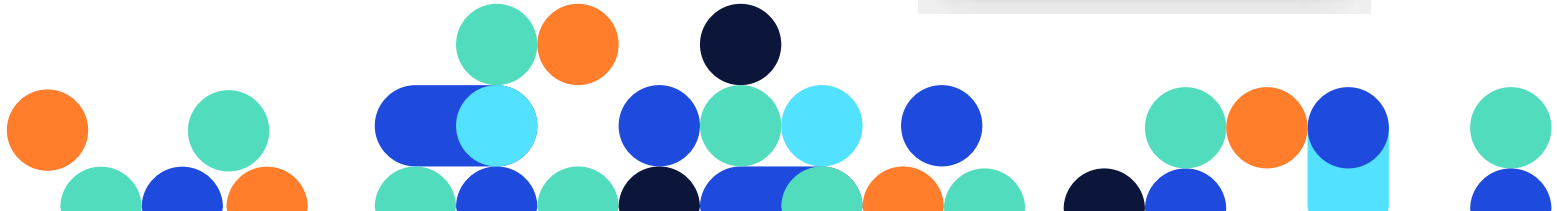
- SELECT some data from a KillrVideo table

Time for an exercise!

DATASTAX

“Spark SQL” Notebook

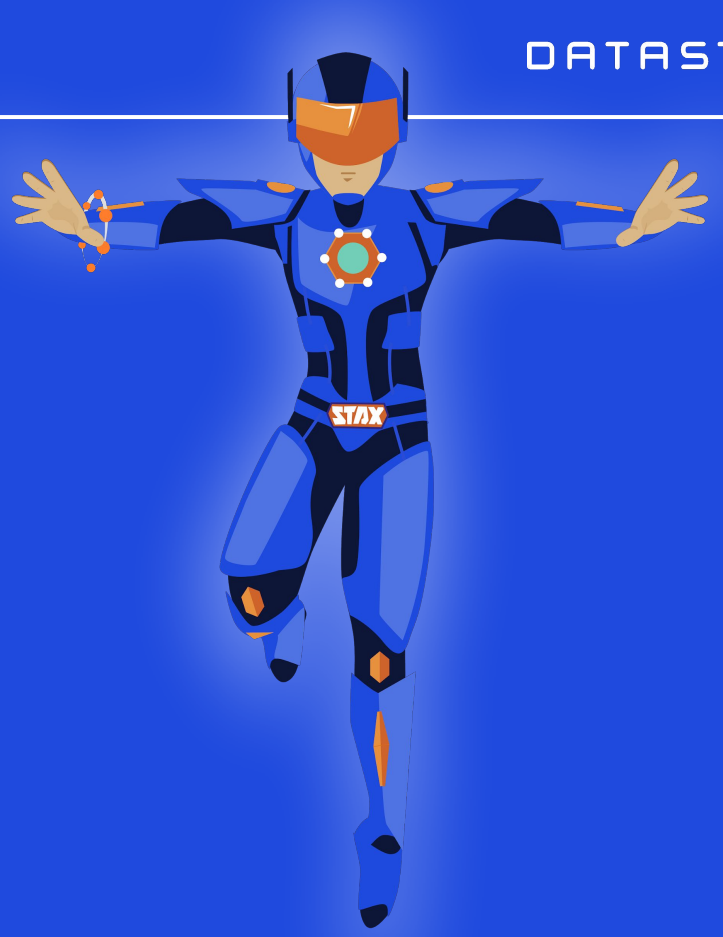




DSE Analytics

DATASTAX

Apache Spark™ REPL



Read-Eval-Print-Loop

REPL

- Read-Evaluate-Print Loop
- Terminal for Apache Spark™ commands
- Variations
  - Spark SQL
  - Scala
  - Python

dse spark

```
ubuntu@developer-day-node1:~$ dse spark
The log file is at /home/ubuntu/.spark-shell.log
Creating a new Spark Session
Spark context Web UI available at http://13.57.198.178:4041
Spark Context available as 'sc' (master = dse://?, app id = app-20180410160751-0001).
Spark Session available as 'spark'.
Spark SqlContext (Deprecated use Spark Session instead) available as 'sqlContext'
Welcome to

      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0.11
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_161)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Scala

- Built on top of Java
- More terse
- Functional


```
scala> println("Hello World")
Hello World
```

SparkSession

- REPL automatically sets up a variable named `spark`
  - Instance of a `SparkSession`
- `SparkSession` is your entry point for all things Spark

Spark SQL

- Run SQL commands via `spark.sql()`
- SQL statement doesn't need semicolon because no longer in Spark SQL shell
- Returns a `DataFrame`

```
spark.sql("""SELECT firstname,
lastname, count(rating) as theCount,
  AVG(rating) as theAvg
  FROM killrvideo.users NATURAL
  JOIN killrvideo.video_ratings_by_user
  GROUP BY userid, firstname, lastname
  HAVING count(rating) > 7
  AND AVG(rating) < 2.5""")
```

Multi-Line Entry

- Sometimes the Scala REPL struggles interpreting multi-line statements
- Type `:paste` then hit enter
  - Puts the REPL into paste mode
- Paste your multi-line expression, then type CTRL-D to get out of paste mode
- ***Be sure your cursor is on a new line before you type CTRL-D***

vars and vals

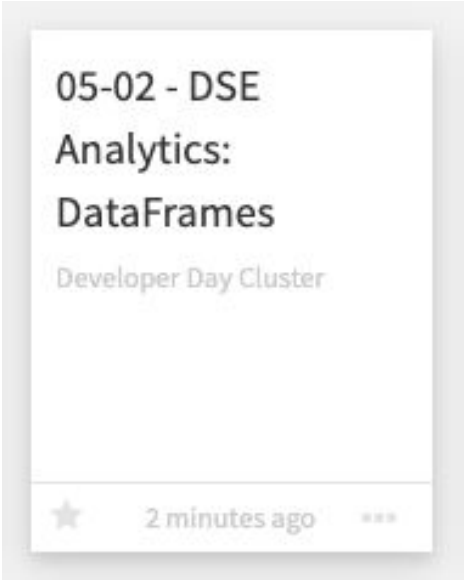
- vars are variable (value can change)
- vals are constant (immutable)

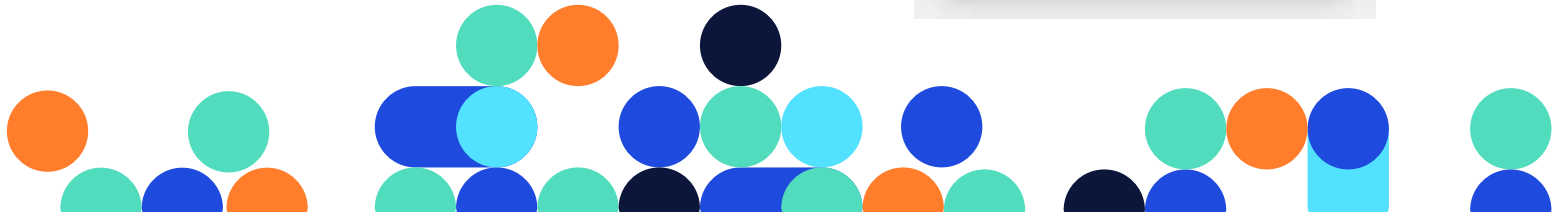
```
scala> val cowName = "Betsy"
cowName: String = Betsy

scala> cowName = "Bessie"
<console>:12: error: reassignment to
val
       cowName = "Bessie"
                 ^

scala> var dogName = "Sally"
dogName: String = Sally

scala> dogName = "Tex"
dogName: String = Tex
```

Time for an exercise!

DATASTAX

“Data Frames” Notebook





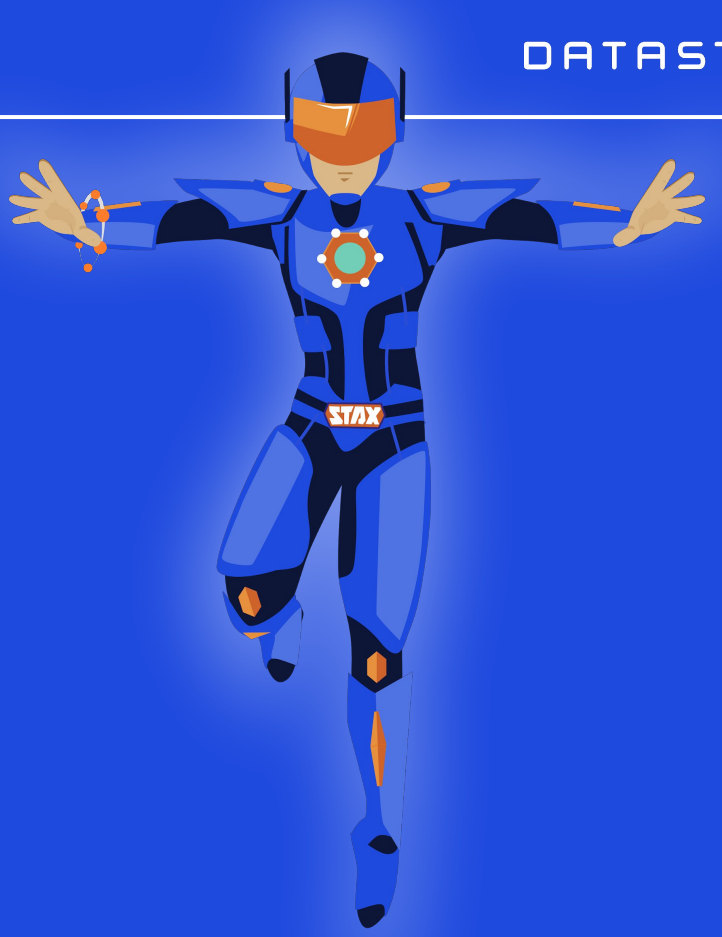
DataFrame Summary

- Schema
- Rows
- Columns
- Queries
- `spark.sql()`

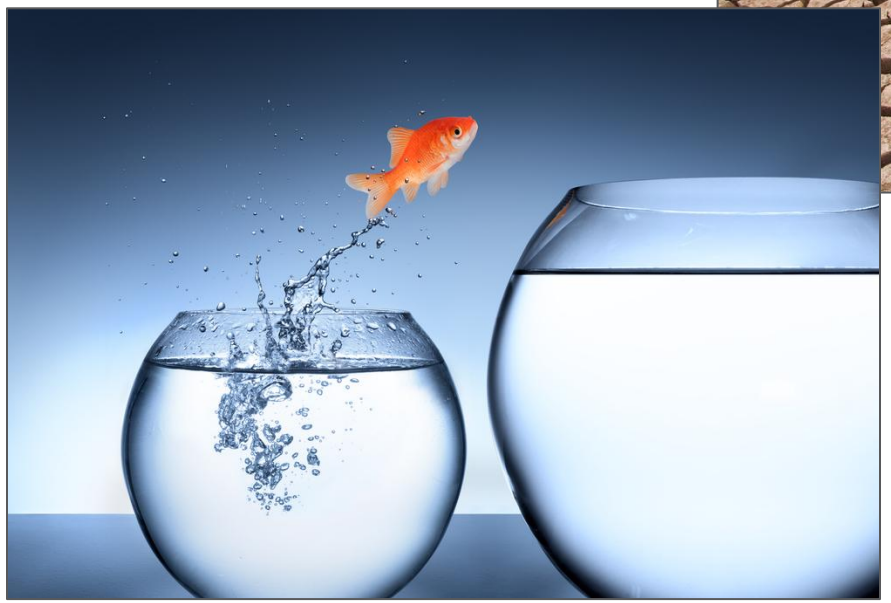
DSE Analytics

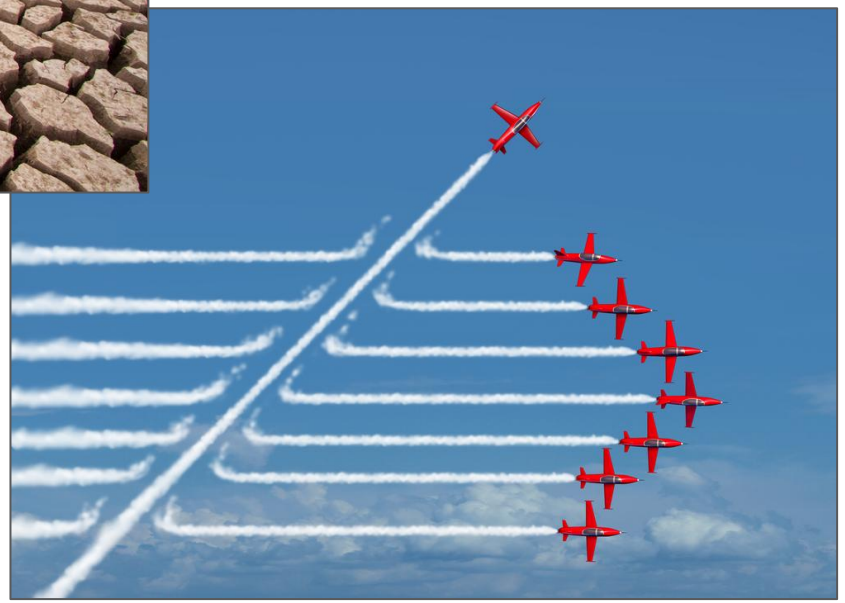
DATASTAX

Extract Transform and Load (ETL)

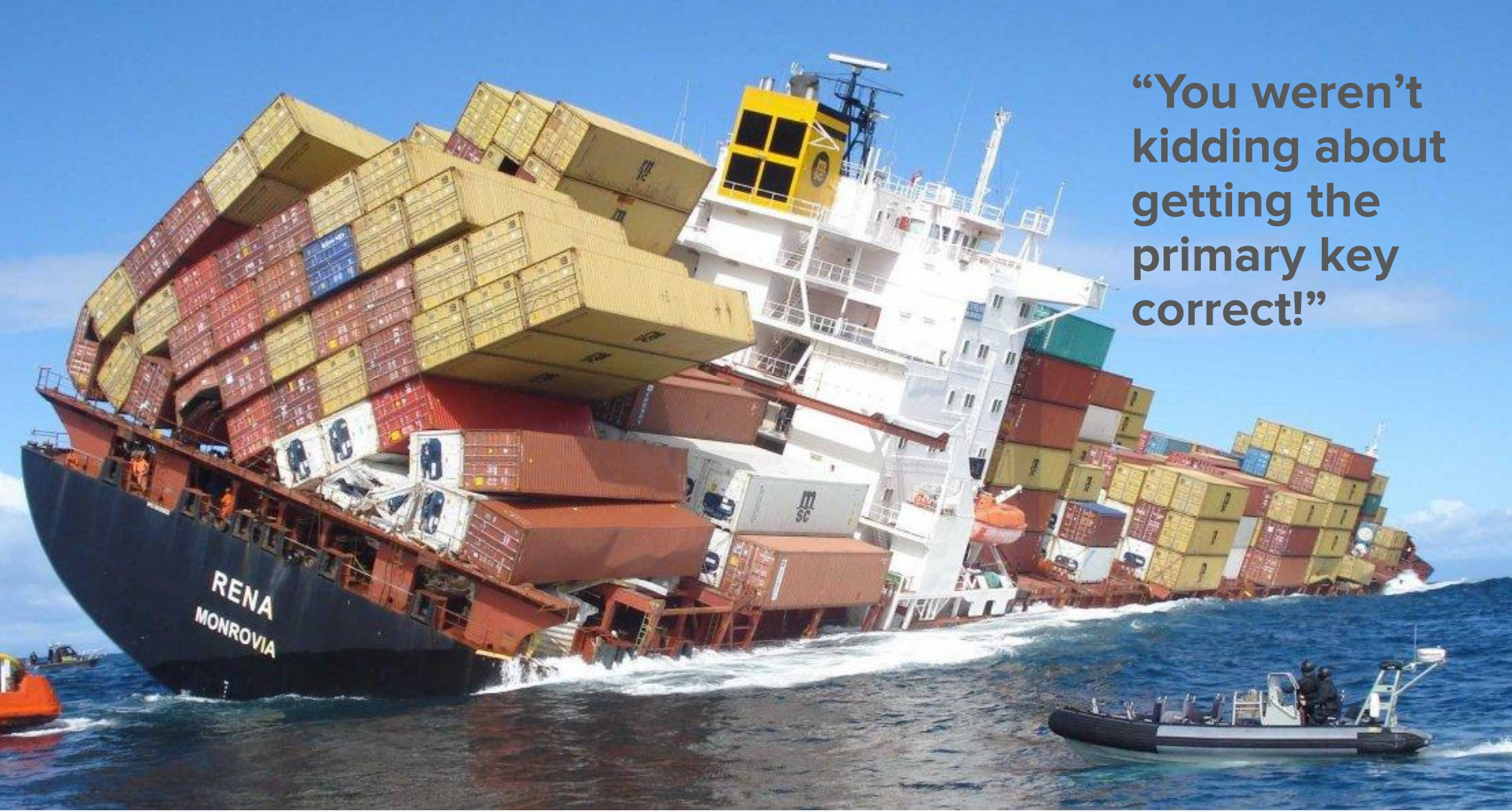


Requirements Change






“Data modeling for Cassandra is so...waterfall.”



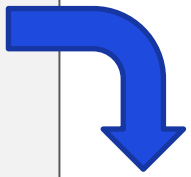
"You weren't kidding about getting the primary key correct!"

Schema Evolution with DSE Analytics

- Pull your data
- Make a new table
- Save
- Party!

THE GOAL

```
CREATE TABLE killrvideo.comments_by_user_video
(
    userid uuid,
    commentid timeuuid,
    comment text,
    videoid uuid,
    PRIMARY KEY (userid, commentid)
)
```



```
CREATE TABLE killrvideo.comments_by_user_video
(
    userid uuid,
    commentid timeuuid,
    comment text,
    videoid uuid,
    PRIMARY KEY ((userid, videoid), commentid)
)
```

Pull Your Data

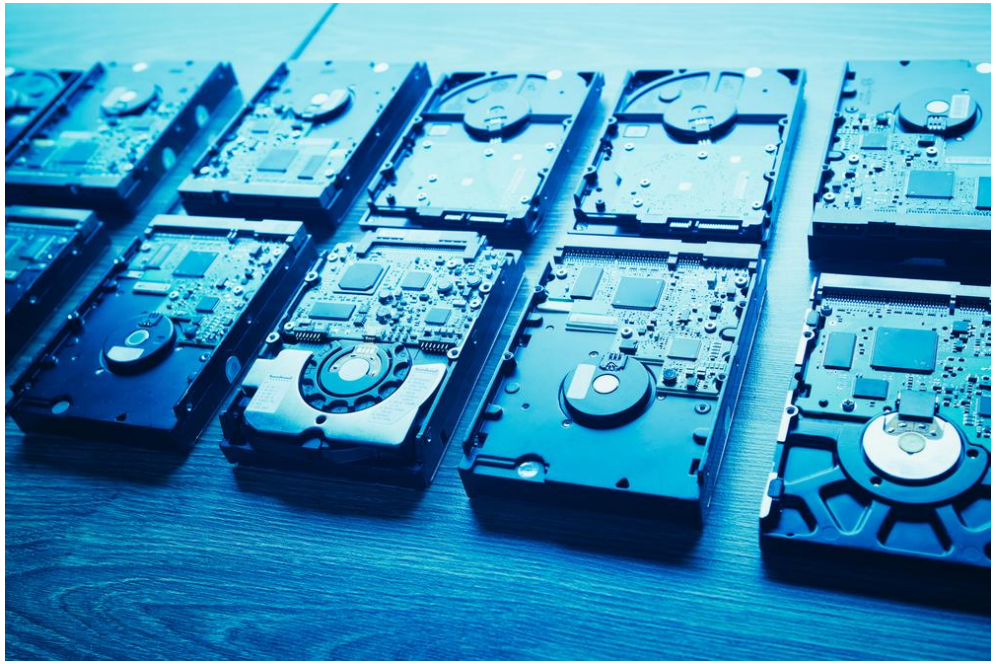
```
val pulled = spark.sql("SELECT * FROM killrvideo.comments_by_user")
```

Make a New Table

```
pulled.createCassandraTable(
  "killrvideo",
  "comments_by_user_video",
  partitionKeyColumns = Some(Seq("userid", "videoid")),
  clusteringKeyColumns = Some(Seq("commentid"))
)
```

Save

```
pulled
  .write
  .cassandraFormat("comments_by_user_video", "killrvideo")
  .save()
```




PARTY!

Pulling Your Data

- Transform your query however you like in your SQL statement
- You don't actually pull until you `save()`
- And the *cluster* then does all the work!

- yeah, that's cool

Likely Story

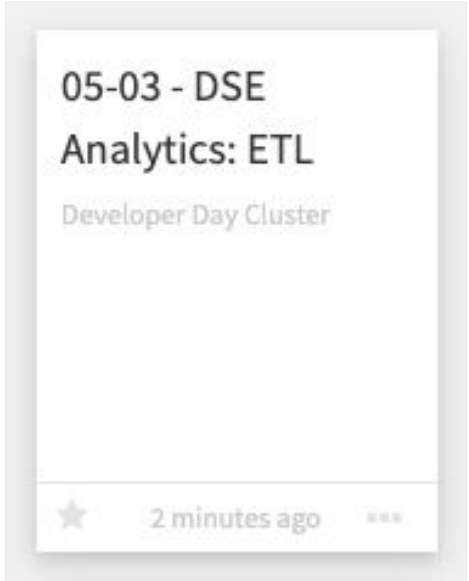
You boss comes to you and asks, “Hey Jimmy, I need you to add a report where I can view how many times a person has rated our videos and what their average rating is. I need to view this data on a daily basis, that is, what are their numbers for any given day.”

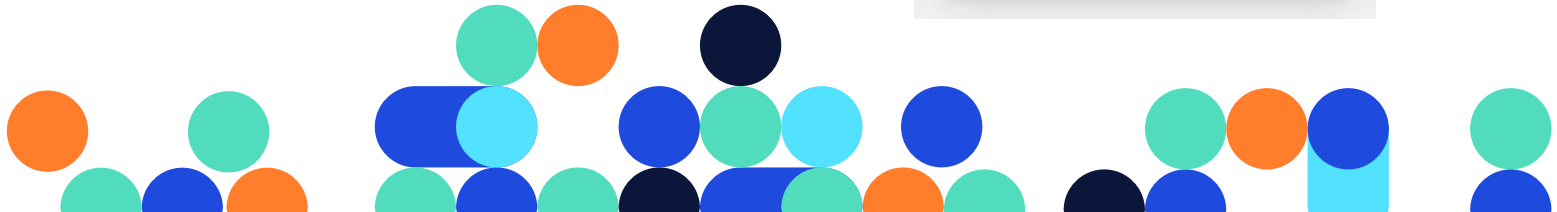
Whatcha gonna do?

Time for an exercise!

DATASTAX

“DSE Analytics ETL” Notebook





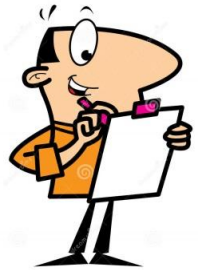
DSE Analytics

DATASTAX

Spark Architecture

Characters

- Spark Master
  - Book-keeper



- Spark Worker
  - Middle management

- Spark Executor
  - Actually does stuff

- Spark Driver
  - Really in charge

Who's important?

- Spark Driver
  - Your program
  - Runs on your machine
    - Or you can ask the Master to have a cluster node run it for you (cluster mode)
  - Decides which Executors do which work
  - Makes all the decisions
    - Well… not all, but…

Who's Also Important?

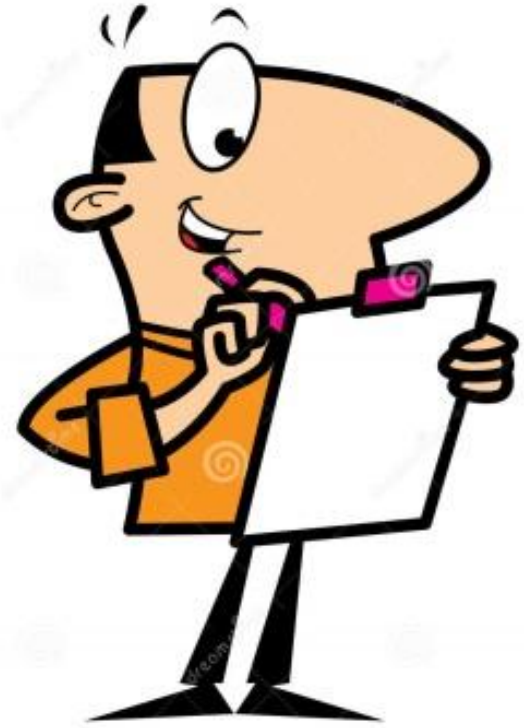
- Spark Executor
  - Does **all** the work
    - Well… not *all*, but…
  - Does whatever work the Driver says
    - Does *not* take direction from the Master or the Worker
    - But is launched by the Worker
  - Gets jars/code/work items/etc from the Driver directly
  - Reads / Writes from sources (e.g., Cassandra)
  - Plays nicely with its peers

Who's not that Important?

- Spark Worker
  - Responsible for launching Executors
  - Responsible for telling the Master what's running
  - And… um…

Who's Also not that Important?

- Spark Master
  - Driver submits jobs to the Master
  - So, you **do** need to know how to talk to him
  - Assigns Executors to a job
    - Tells the Workers to launch Executors
  - Lots of bookkeeping

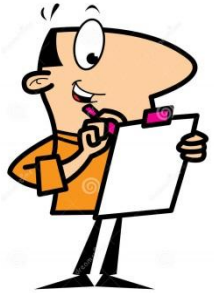


But they all have roles to play...

"I have work to do"

Sends request for Executors

But they all have roles to play…



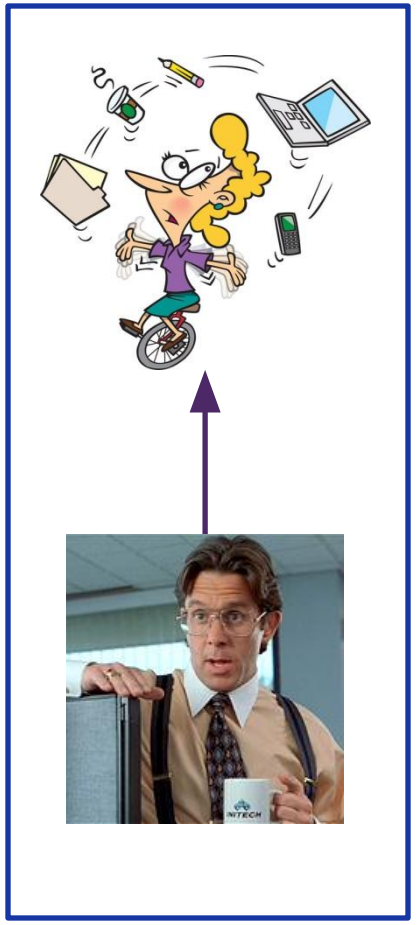
"Okay, we have a work order"

Tells Workers to create Executors for this task

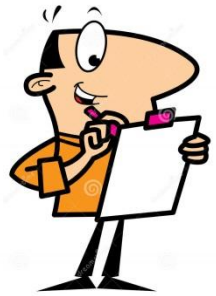
But they all have roles to play…

"Okay guys, I'm assigning you to this work order. Go talk to the foreman for the actual tasks"

Spins up Executors for the job

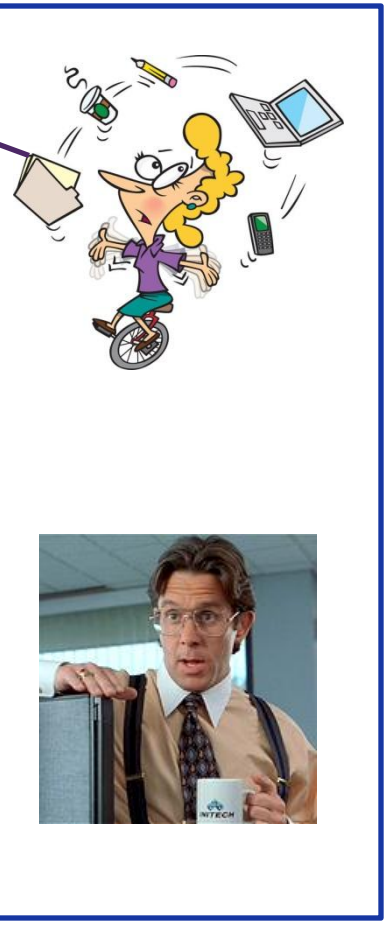


But they all have roles to play…

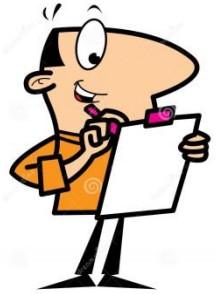


"Okay, let's get this stuff done. Here's what you need. Give me what I need. Etc."

Driver and Executors do the work together

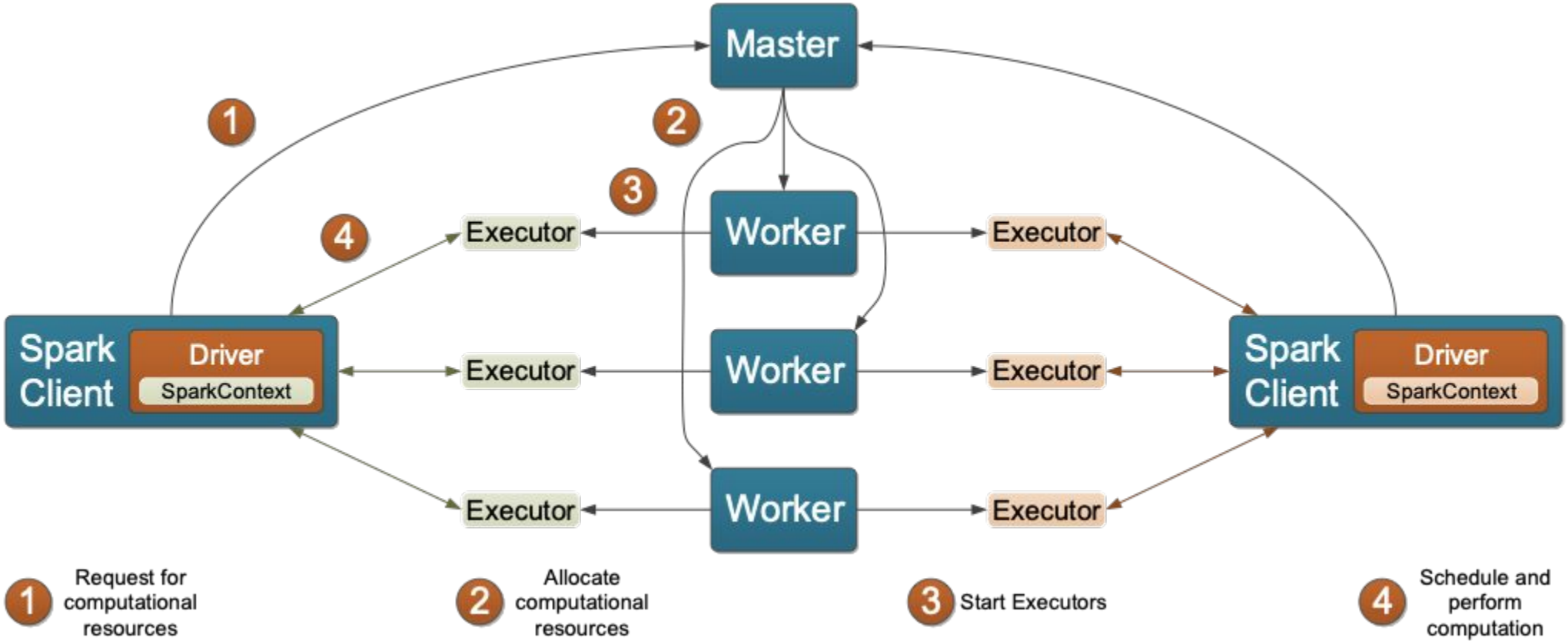


But they all have roles to play…



While the Workers and Master sip coffee, and wait for more requests.

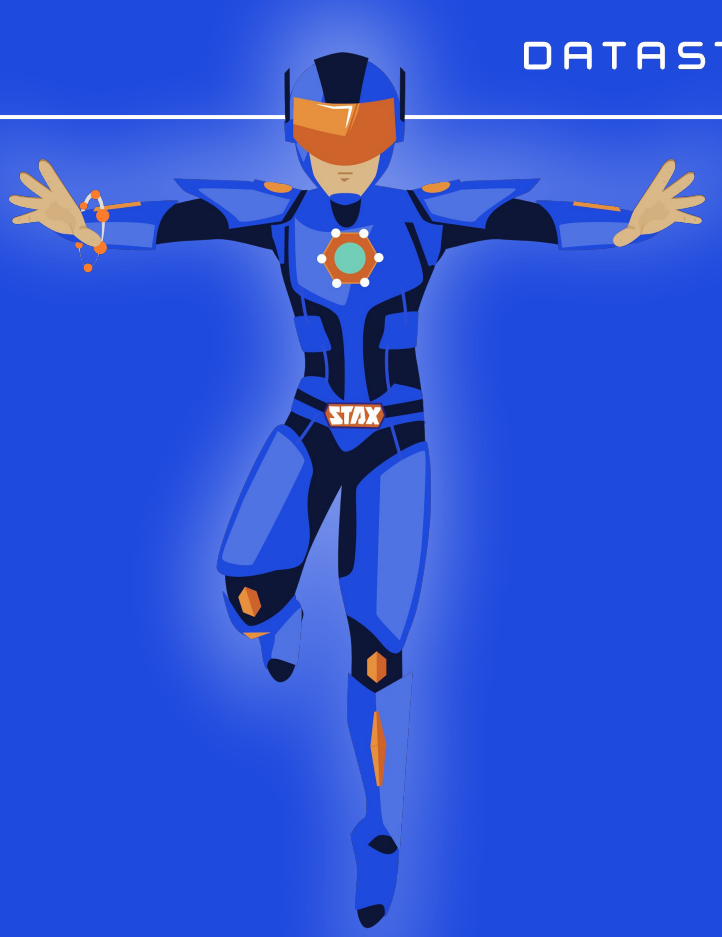
Spark Architecture – Client, Master, Executor, Worker



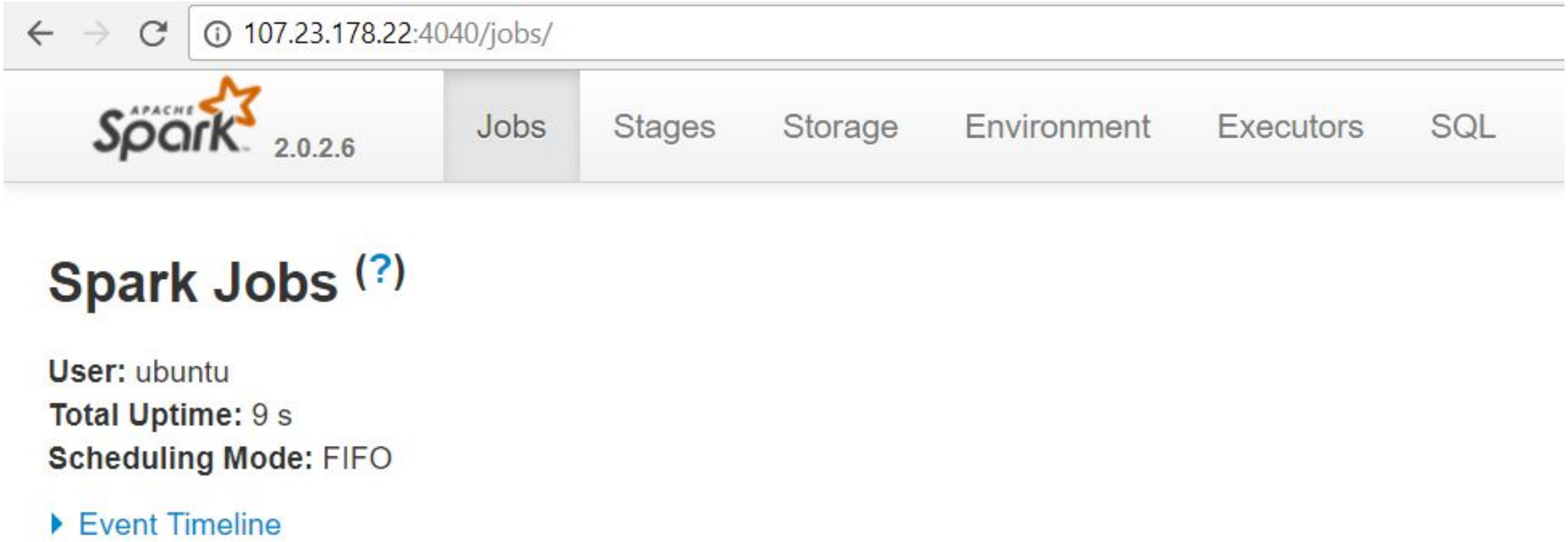
DSE Analytics

DATASTAX

Jobs, Stages, and Tasks



Fresh Application UI http://<your-ip>:4040



Demo

- `dse spark`
- Master UI
- Application UI
- RDD magic and shuffling

- Let's roll

Which node is the master?

- You don’t have to care with DSE!
- But!
- If you want to see the master management UI
- http://<master ip address>:7080

Which node is the master?

```
ubuntu@ds320-node1:~$ dse client-tool spark master-address
dse://107.23.178.22:9042?connection.local_dc=Analytics;connection.host=172.31.29.40;

ubuntu@ds320-node1:~$ dsetool ring
Address          DC                  Rack         Workload            Graph  Status
State    Load            Owns                Token
Health [0,1]
172.31.29.40     Analytics           rack1        Analytics(SM)       no     Up
Normal   911.79 MiB      ?                   149367368566488727
0.90
```

Application UI

- http://<your ip address>:4040

Word Count – Only Transformations

```
val meWords = Range(1,1000)
  .map(i => scala.util.Random.nextPrintableChar)

val meRdd = spark.sparkContext
  .parallelize(meWords)
```

Actions Make Things Go!

```
val meWords = Range(1,1000)
  .map(i =>
scala.util.Random.nextPrintableChar)

val meRdd = spark.sparkContext
  .parallelize(meWords)

meRdd.take(2).foreach(println)

// Sample output:
S
Q
```

Spark Jobs (?)

User: ubuntu
Total Uptime: 5.8 min
Scheduling Mode: FIFO
Completed Jobs: 1

▸ Event Timeline

Completed Jobs (1)

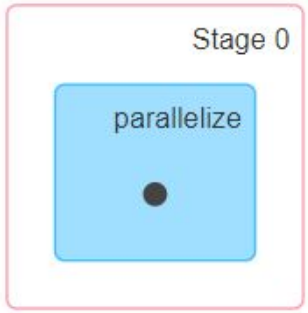
Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
0	take at <console>:42	2017/07/21 17:43:34	0.8 s	1/1	1/1

Job Details

Details for Job 0

Status: SUCCEEDED
Completed Stages: 1

▸ Event Timeline
▾ DAG Visualization



Completed Stages (1)

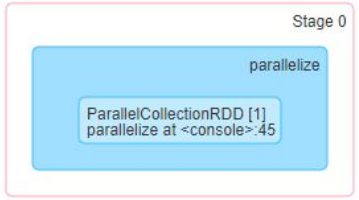
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	take at <console>:42 +details	2017/07/21 17:43:34	0.6 s	1/1				

Stage Details

Details for Stage 0 (Attempt 0)

Total Time Across All Tasks: 19 ms
Locality Level Summary: Process local: 1

▾ DAG Visualization



▸ Show Additional Metrics
▸ Event Timeline

Summary Metrics for 1 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	19 ms	19 ms	19 ms	19 ms	19 ms
Scheduler Delay	0.1 s	0.1 s	0.1 s	0.1 s	0.1 s
Task Deserialization Time	0.6 s	0.6 s	0.6 s	0.6 s	0.6 s
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Result Serialization Time	1 ms	1 ms	1 ms	1 ms	1 ms
Getting Result Time	0 ms	0 ms	0 ms	0 ms	0 ms
Peak Execution Memory	0.0 B	0.0 B	0.0 B	0.0 B	0.0 B

Aggregated Metrics by Executor

Executor ID ▴	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks
0	172.31.29.40:58319	0.8 s	1	0	1

Tasks (1)

Index ▴	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	Scheduler Delay	Task Deserialization Time	GC Time	Result Serialization Time	Getting Result Time	Peak Execution Memory	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	0 / 172.31.29.40	2017/07/21 18:49:42	19 ms	0.1 s	0.6 s		1 ms	0 ms	0.0 B	

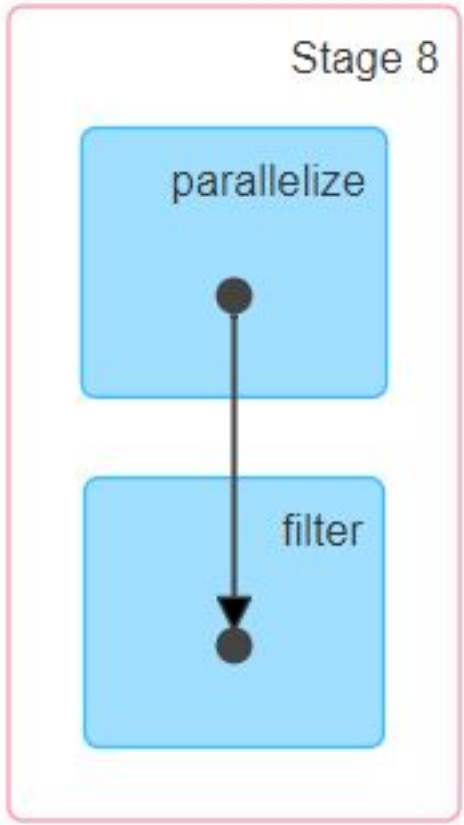
Tasks

Tasks (1)

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	0 / 172.31.24.200 stdout stderr	2018/04/19 22:06:57	41 ms		

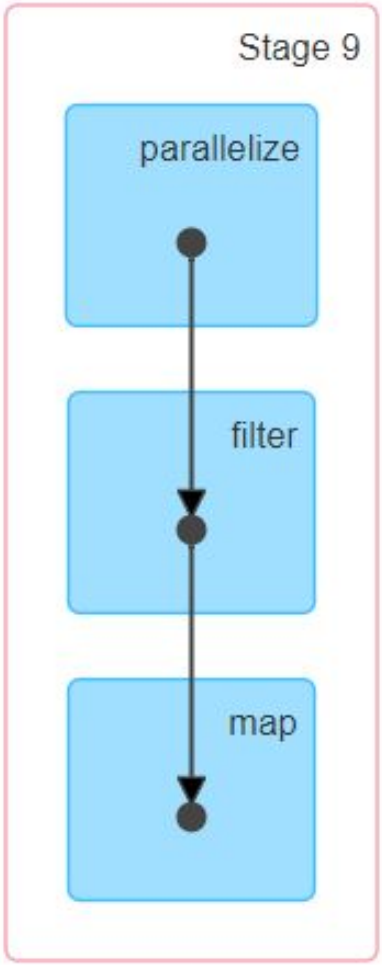
Add a filter() Transformation

```
val meWords = Range(1,1000)
  .map(i =>
scala.util.Random.nextPrintableChar)

spark.sparkContext
  .parallelize(meWords)
  .filter(w => 'A' <= w && w <= 'Z')
  .take(2)
  .foreach(println)
```



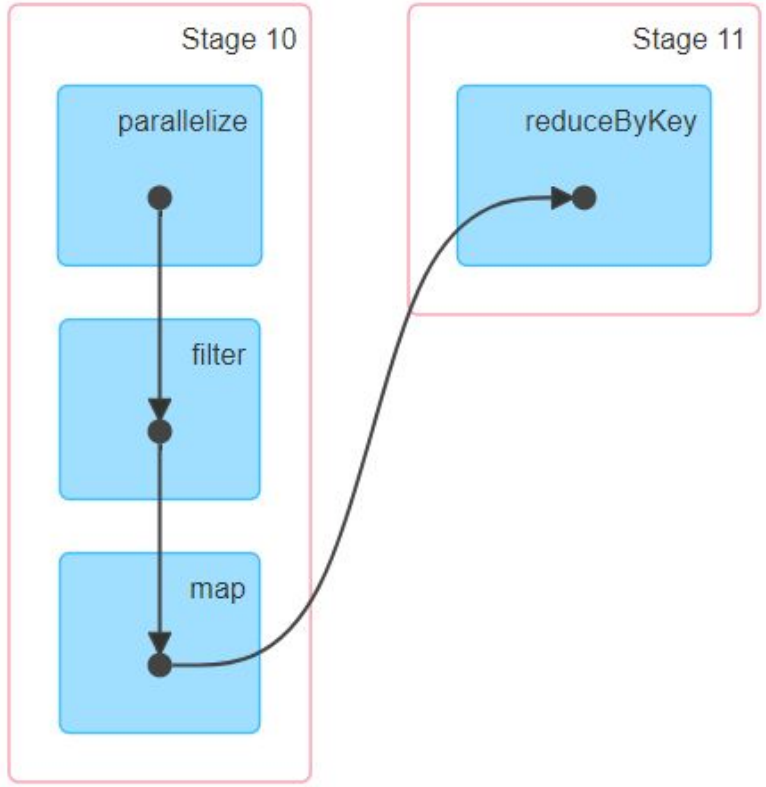
Add a map() Transformation

```
val meWords = Range(1,1000)
  .map(i =>
scala.util.Random.nextPrintableChar)

spark.sparkContext
  .parallelize(meWords)
  .filter(w => 'A' <= w && w <= 'Z')
  .map(w => (w, 1))
  .take(2)
  .foreach(println)
```



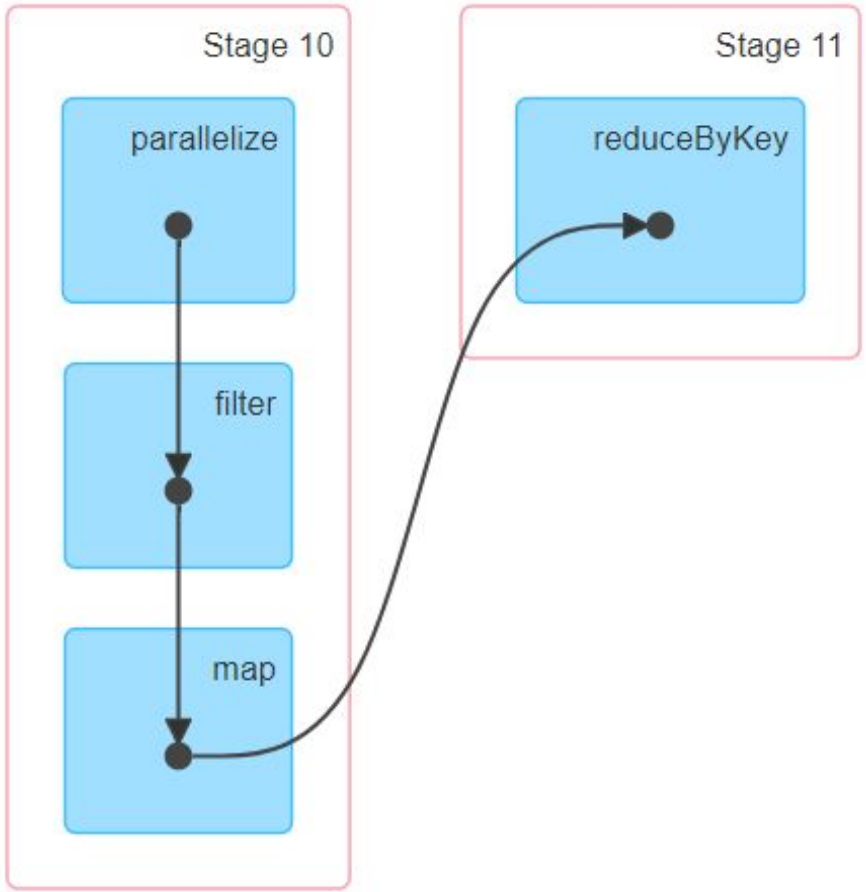
Add a map() Transformation

```
val meWords = Range(1,1000)
  .map(i =>
scala.util.Random.nextPrintableChar)

spark.sparkContext
  .parallelize(meWords)
  .filter(w => 'A' <= w && w <= 'Z')
  .map(w => (w, 1))
  .reduceByKey((x,y) => x + y)
  .take(2)
  .foreach(println)
```



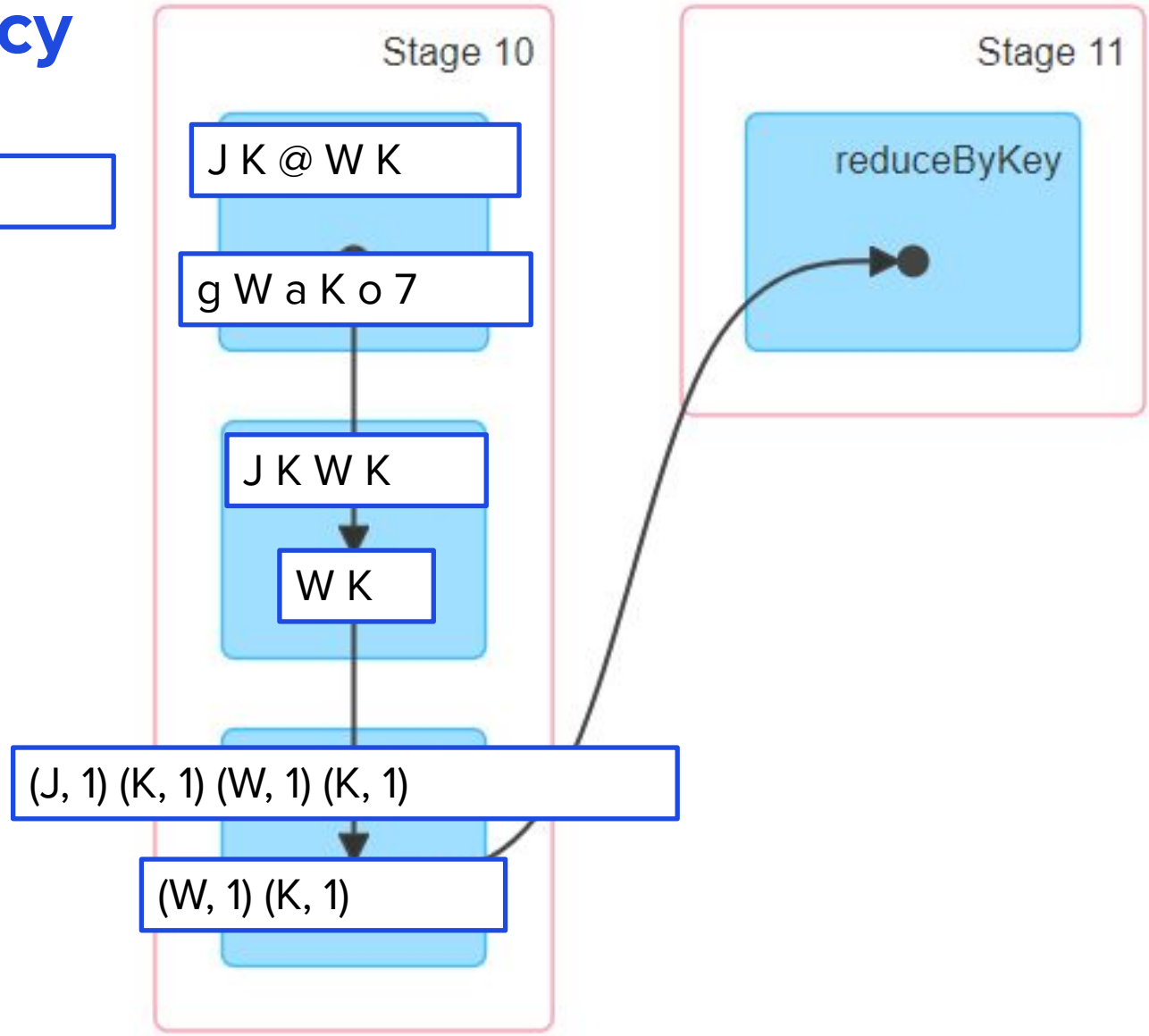
Two Types of Dependencies

- Narrow
- Wide

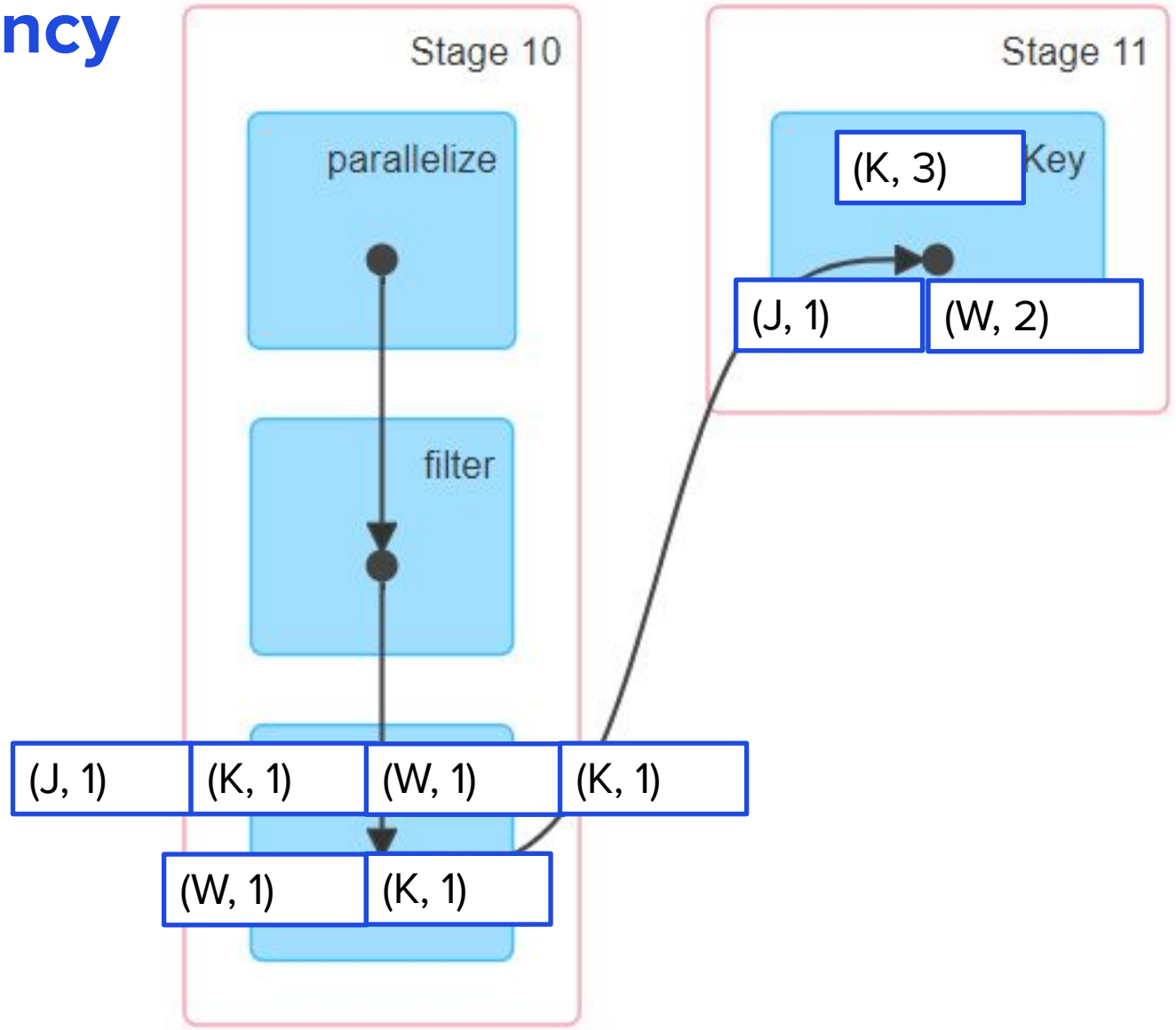


Narrow Dependency

J K g W @ a K o W K 7

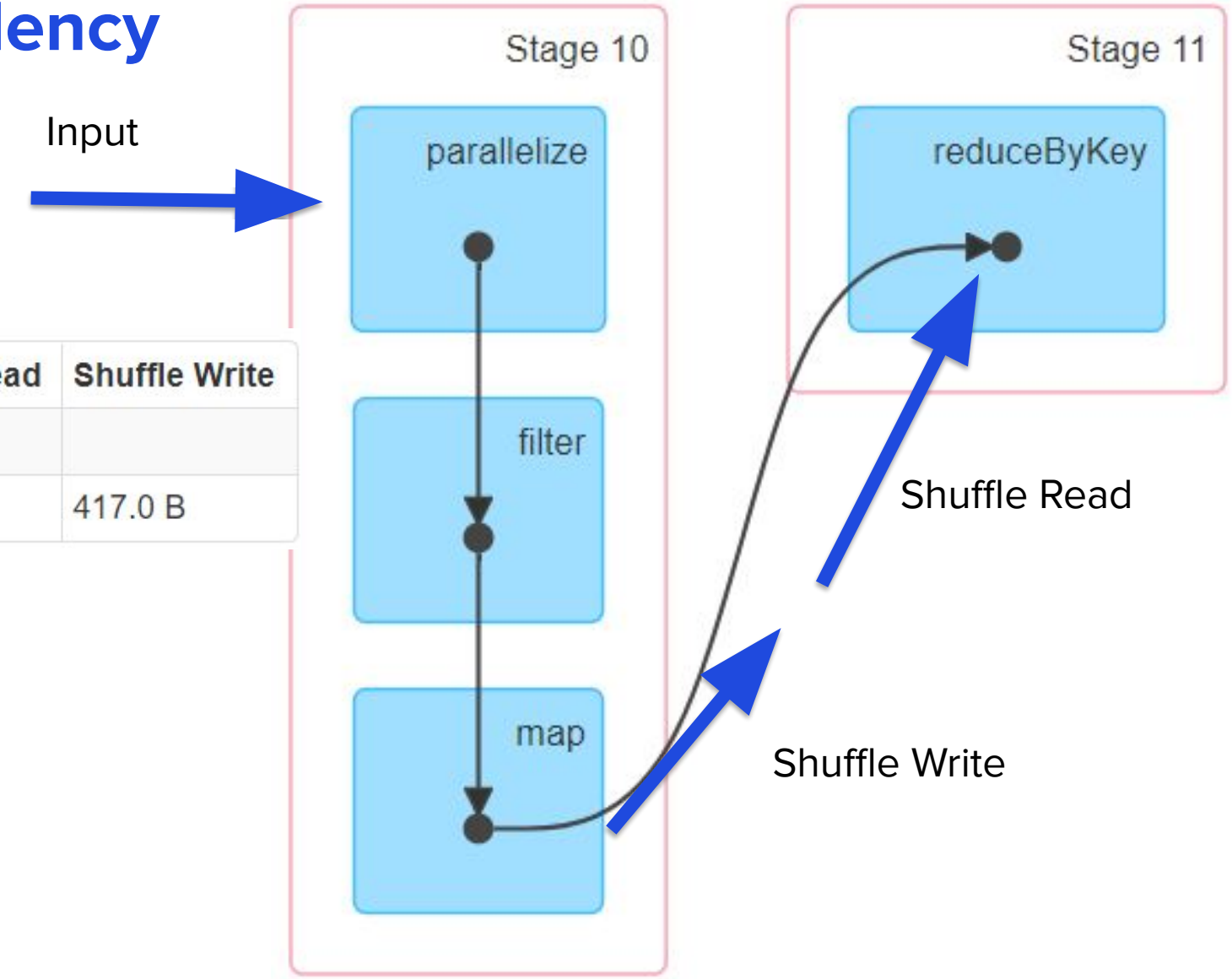


Narrow Dependency



Narrow Dependency

Input	Output	Shuffle Read	Shuffle Write
		211.0 B	
			417.0 B



Stages and Transformations

In summary:

- Transformations are the individual steps
- Narrow transformations can be optimized together into a single stage
- Wide transformations require shuffling and create a new stage
- One task per partition (unit of parallelism).

Shuffle Less

- Filter before shuffling
- Shuffle once; shuffle early
- Use built-in aggregation functions instead of group-by then aggregate

Maintain a Reasonable Degree of Parallelism

- One task per partition
- A few hundred tasks is not a big deal
- Several thousand tasks on small partitions is wasted overhead starting/stopping the task
- Check number of partitions with `spark.partitions.length`
- Use `coalesce()` to reduce the number of partitions
- Be sure to have at least as many partitions as you have cores
  - Otherwise idle cores

repartition() vs. coalesce()

- `repartition()` always causes a shuffle; here's the code:

```
def repartition(numPartitions: Int)
 (implicit ord: Ordering[T] = null): RDD[T] = withScope {
   coalesce(numPartitions, shuffle = true)
 }
```

- If you have several cores and just a few large partitions, repartitioning to the number of cores you have will keep all cores busy
- Tradeoff moving data around the cluster though

Serializer

- By default, Apache Spark™ uses Java's built in serialization
  - Flexible, but slower
- You may want to try using `org.apache.spark.serializer.KryoSerializer`

- Serialization affects shuffling and caching

Caching

- By default, caching to memory stores objects as they are
  - Fast access
  - Increased heap pressure
- `MEMORY_ONLY_SER` serializes all objects into a single buffer
  - “One” object as far as the garbage collector is concerned
  - Time penalty to read/write because data is serialized

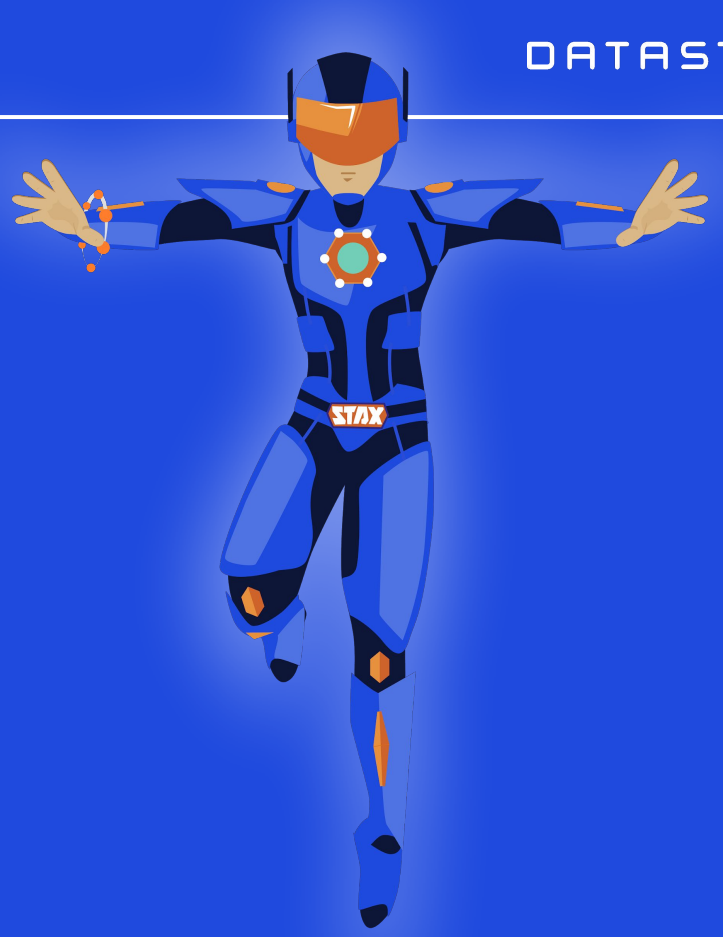
Optimizing Joins

- Joins cause shuffling
  - Records with matching keys shuffled to same node
- Consider your data set, ideally:
  - Your group condition has enough cardinality to spread evenly through the cluster
  - Your group condition has a higher number of values than the cores in your cluster (parallelism; utilize all cores)

DSE Analytics

DATASTAX

Data Validation with Apache Spark



Denormalization

- Why does it hurt?
- Why do we do it?

Needles in a Haystack

- Use DataStax Enterprise Analytics to solve the problem
- Are there any orphaned records in these tables????
  - videos
  - user_videos
  - videos_by_tag

BOOM!

```
select * from user_videos where videoid not in (select videoid from videos);

select * from user_videos where userid not in (select userid from users);

select * from videos where videoid not in (select videoid from user_videos);

select * from videos_by_tag where videoid not in (select videoid from user_videos);
```

Next Steps

- Continue learning on DataStax Academy:
  - DS201: DataStax Enterprise Foundations of Apache Cassandra™
  - DS210: DataStax Enterprise Operations with Apache Cassandra™
  - DS220: Data Modeling with Apache Cassandra™
  - DS320: DataStax Enterprise Analytics with Apache Spark™

https://academy.datastax.com

DATASTAX®

Thank You