DataFrames in Spark

Data Analysts' perspective

Marcin Szymaniuk

About me

- Data Engineer @TantusData
- Have worked for: Spotify, Apple, telcos, startups
- Cluster installations, application architecture and development, training, data team support

marcin@tantusdata.com

marcin.szymaniuk@gmail.com

@mszymani

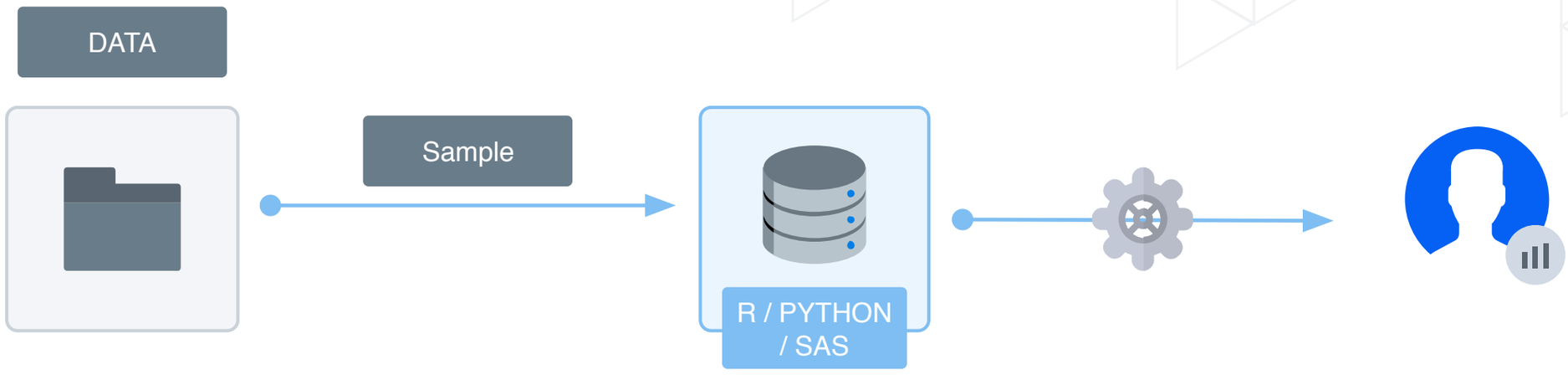
Agenda

- Spark for Data Analysts
- Spark execution model
- Case by case…
- Summary

Data Analysts'/Scientists' perspective

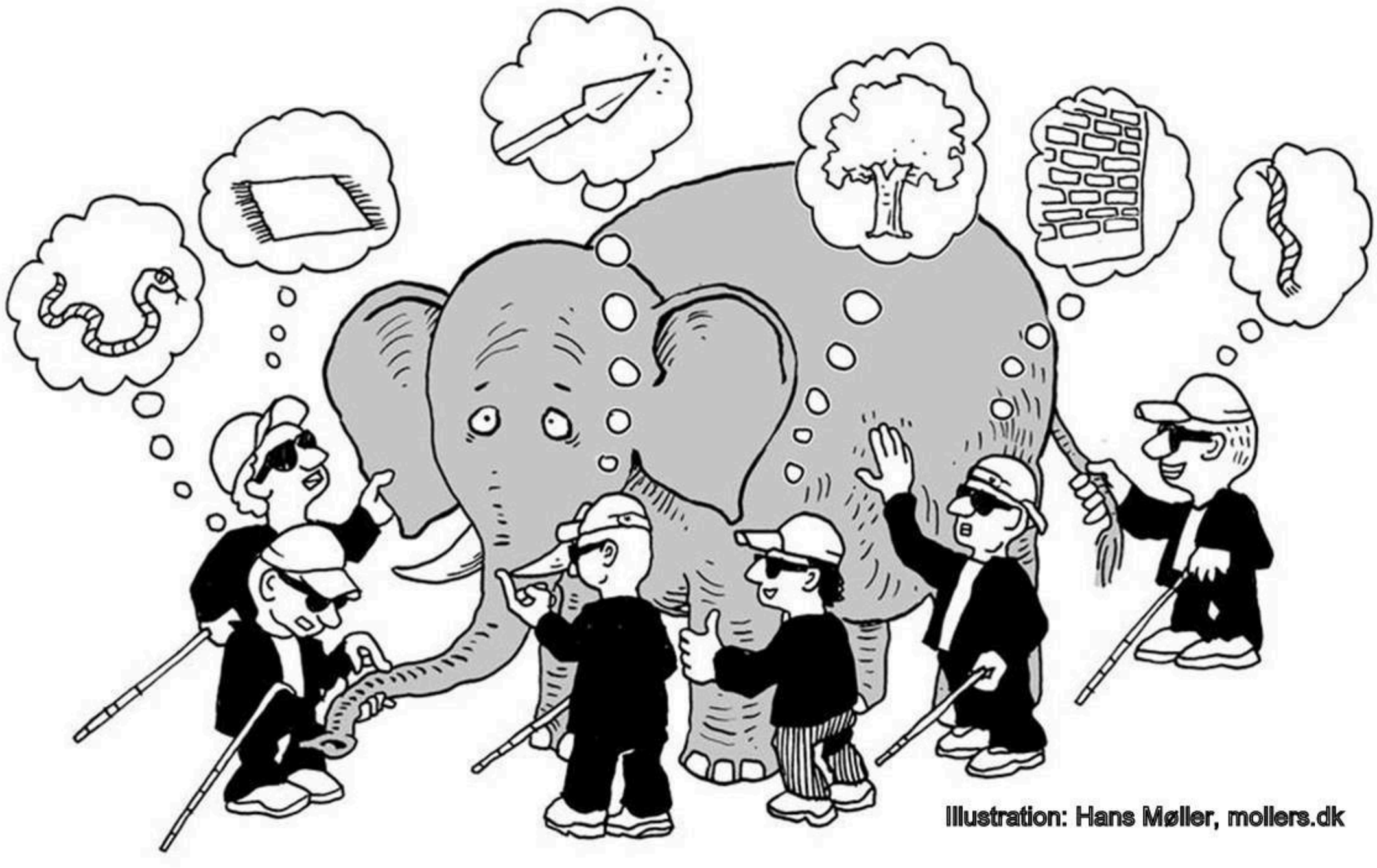
- Explain what happened and why
- Drive decisions to be made
- Build ML models

Bring analysis to data



- Sample only (region, latest month…)
- Coarse aggregate eg. month vs hour (1:720)

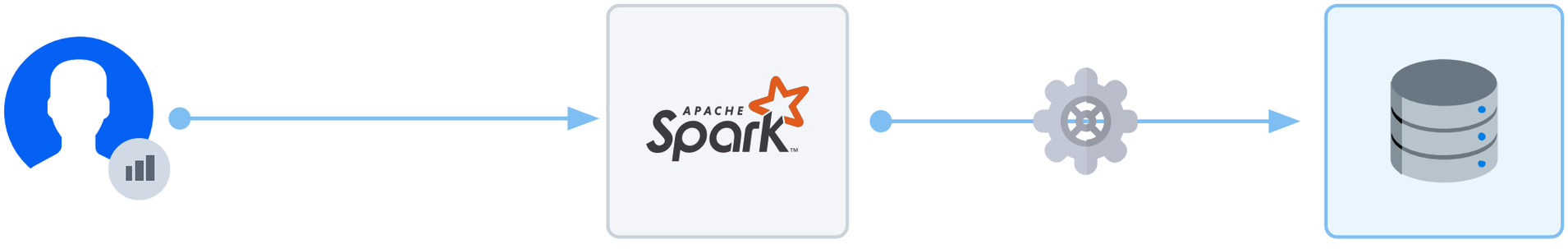
Bring analysis to data



www.tantusdata.com

Photo credit: productcoalition.com

Bring analysis to data



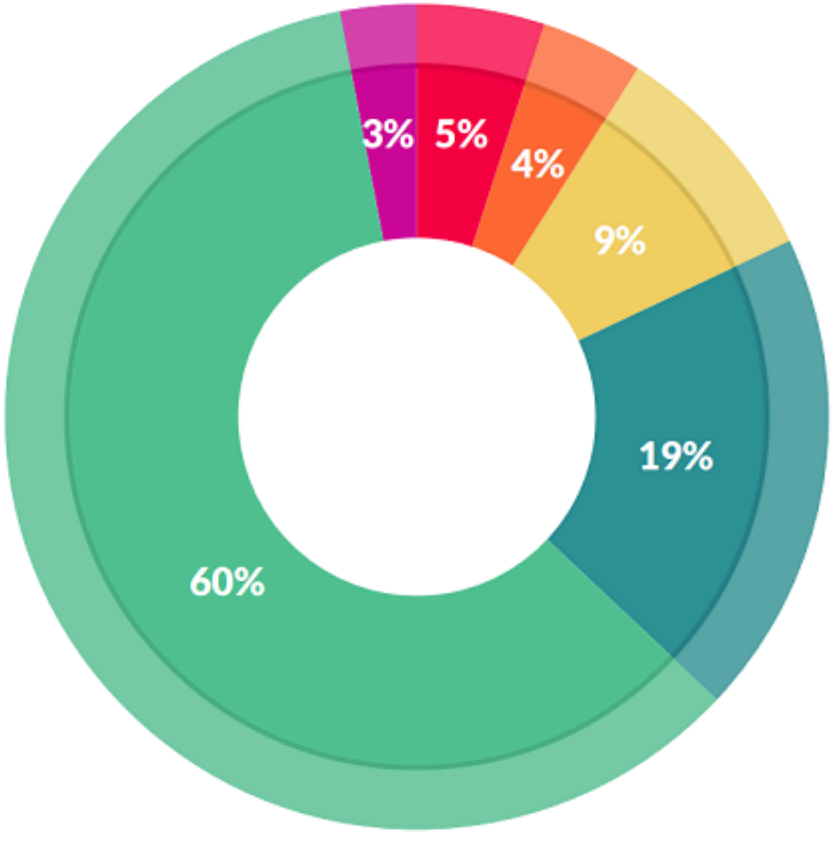
- Analyze all data
- Faster analysis
- No extra data copies (GDPR!)

Data Analysts'/Scientists' perspective

- Ad-hoc queries and reports
- Data cleanup

Data Analysts'/Scientists' perspective

- Ad-hoc queries and reports
- Data cleanup



What data scientists spend the most time doing

- *Building training sets: 3%*
- *Cleaning and organizing data: 60%*
- *Collecting data sets; 19%*
- *Mining data for patterns: 9%*
- *Refining algorithms: 4%*
- *Other: 5%*

www.tantusdata.com

Photo credit: CrowdFlower

Spark API - DataFrame

```
eventsDf
  .groupBy("userId")
  .agg(
    max("value").alias("maxVal"),
    avg("value").alias("avgValue")
  )
  .join(usersDf, usersDf("id") === eventsDf("userId"))
  .select("userId", "maxVal", "avgValue","name")
```

Spark API - DataFrame

```
eventsDf
  .groupBy("userId")
  .agg(
    max("value").alias("maxVal"),
    avg("value").alias("avgValue")
  )
  .join(usersDf, usersDf("id") === eventsDf("userId"))
  .select("userId", "maxVal", "avgValue","name")
```

Data Analysts' perspective

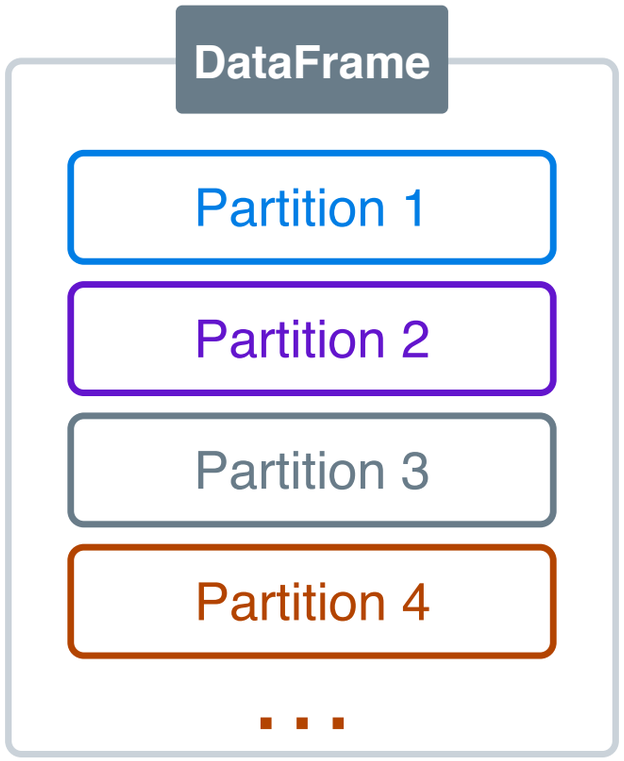
- Simple query fails?
- Simple query never finishes?

Photo credit: thedeliberatemusician.com

Deep dive

Photo credit: amazon.ca

DataFrame



Transformation type

- **Narrow** - single partition at time
- **Wide** - exchange partition’s data

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed",sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col(“str")))
.withColumn(“v1v2Sum”, col("v1") + col(“v2"))
.write
.save(output)
```

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col(“str")))
.withColumn(“v1v2Sum”, col("v1") + col(“v2"))
.write
.save(output)
```

Narrow transformation

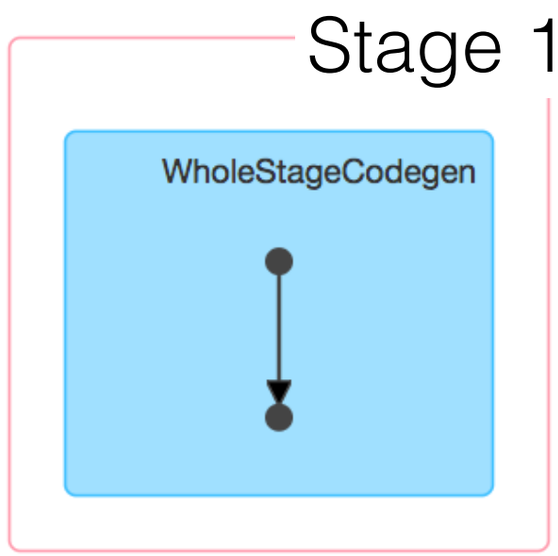
```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col(“str")))
.withColumn(“v1v2Sum”, col("v1") + col(“v2"))
.write
.save(output)
```

Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```

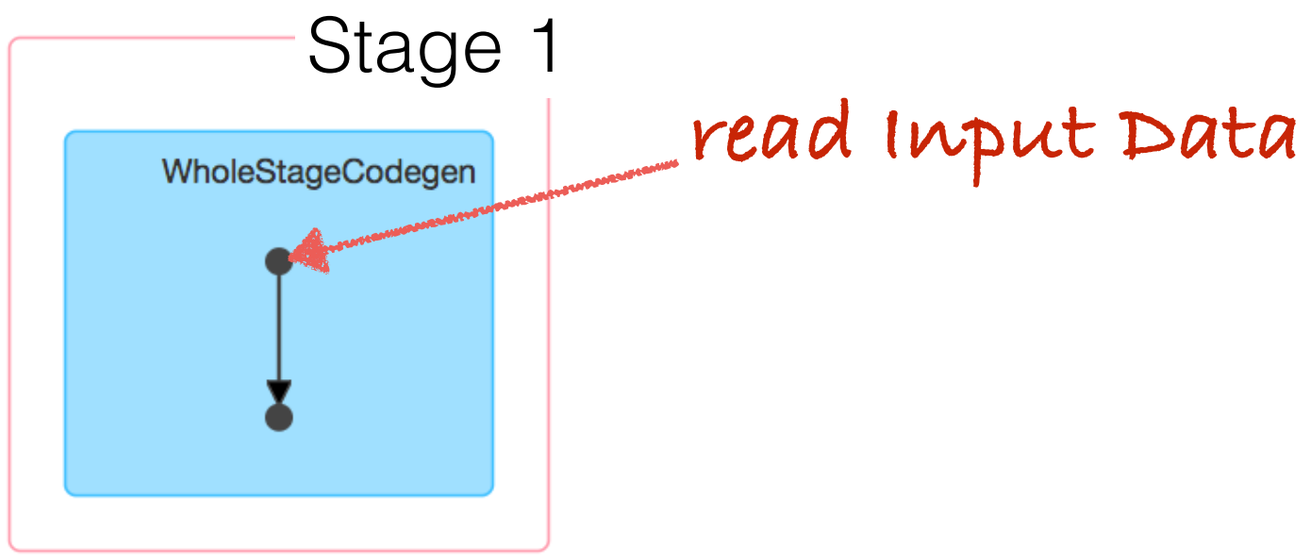
Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```



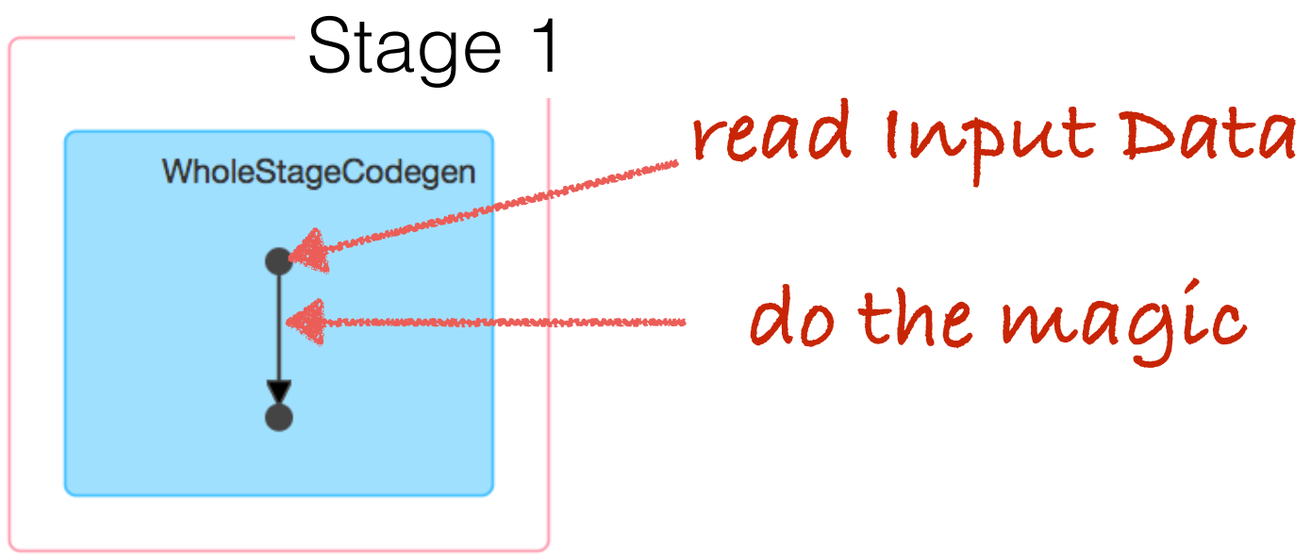
Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```



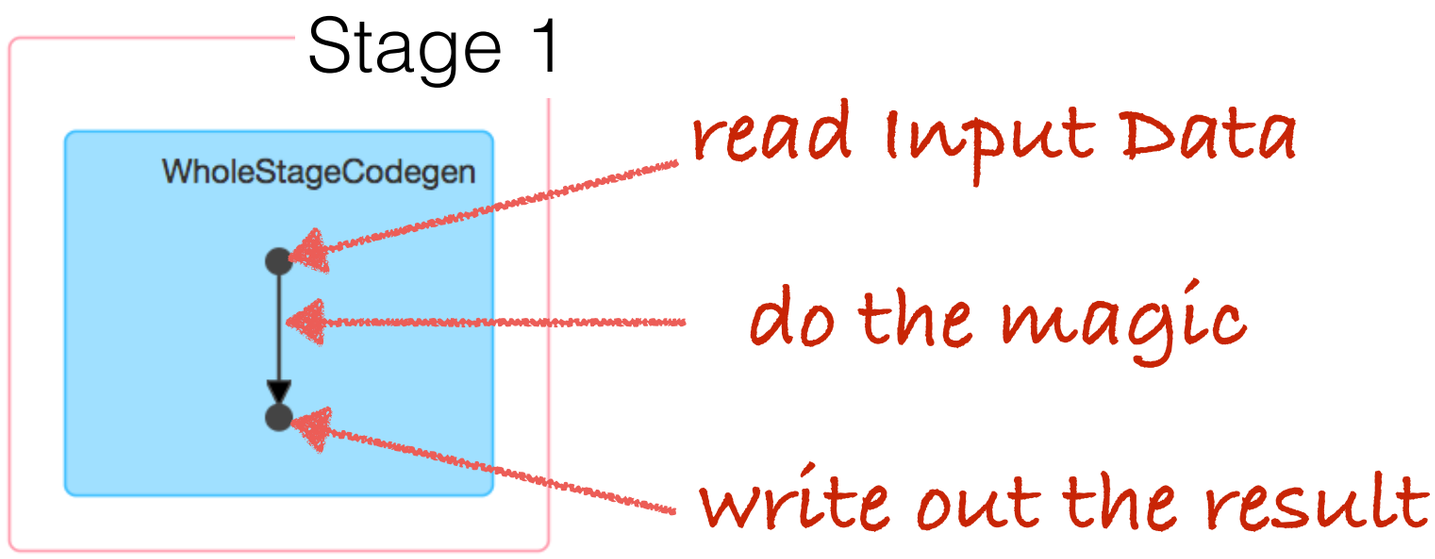
Narrow transformation

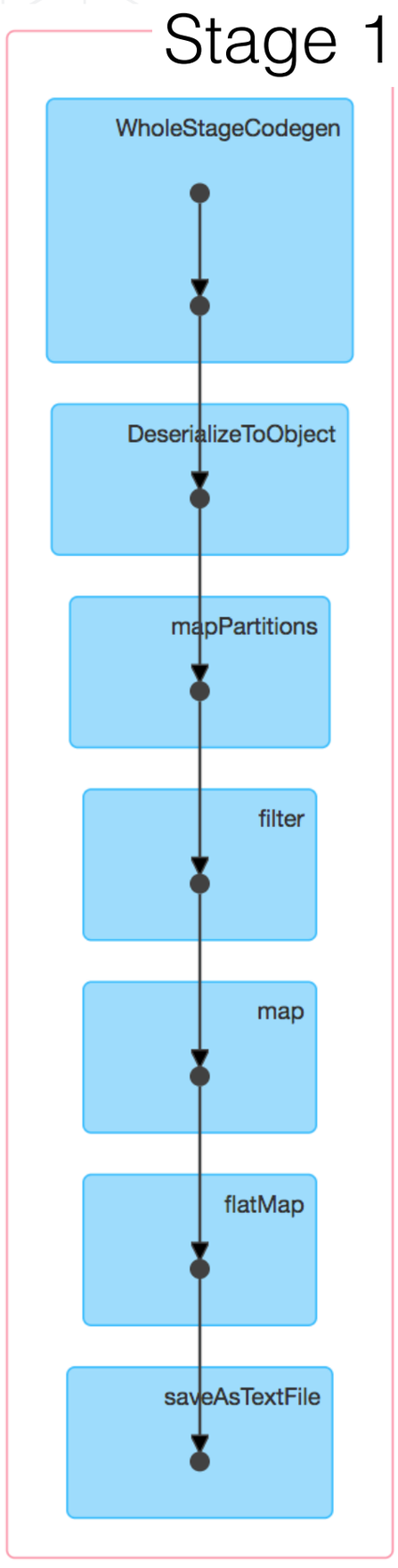
```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```



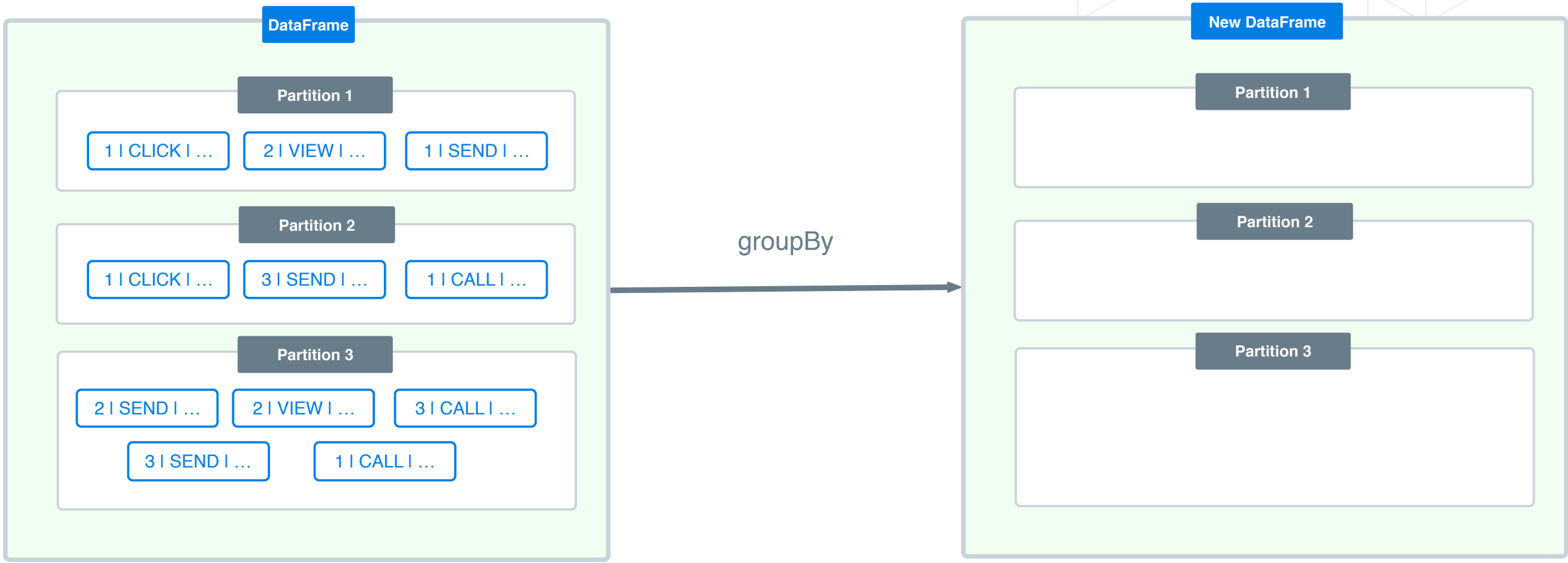
Narrow transformation

```
dataFrame.
.withColumn("uIdHashed", sha2(col("userId"), 256))
.withColumn("strReversed", reverse(col("str")))
.withColumn("v1v2Sum", col("v1") + col("v2"))
.write
.save(output)
```

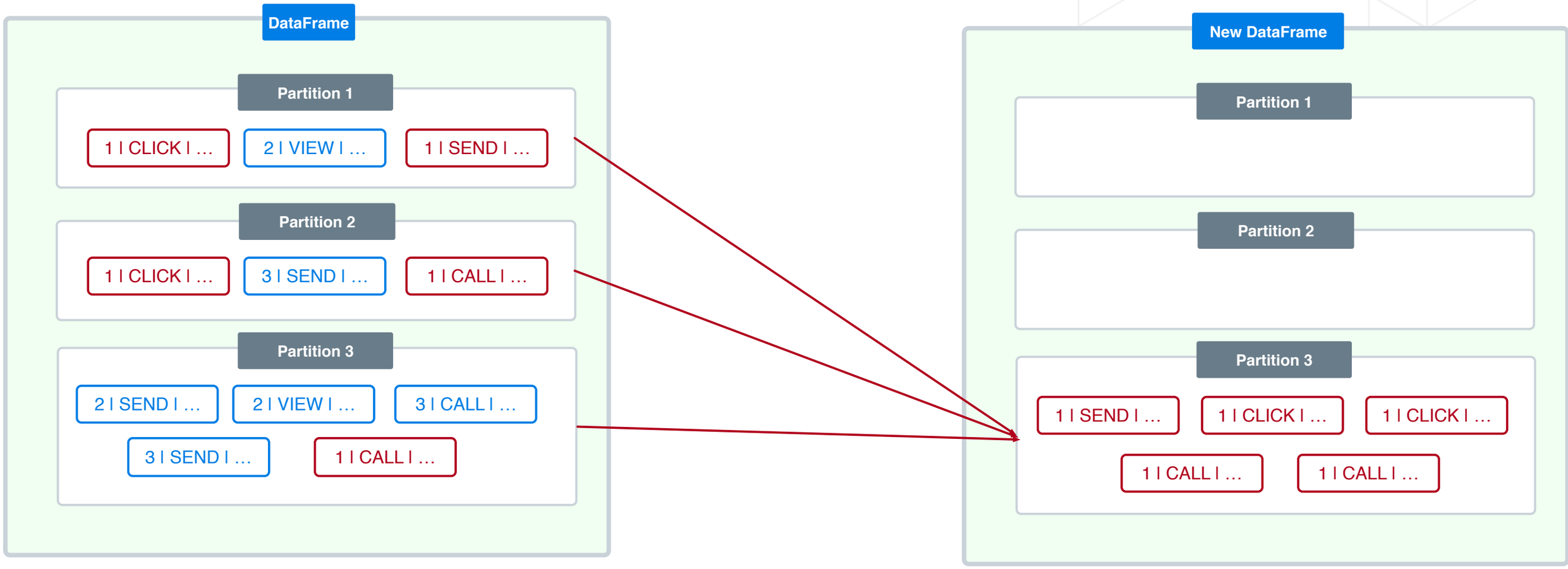



Stage 1
WholeStageCodegen
DeserializeToObject
mapPartitions
filter
map
flatMap
saveAsTextFile

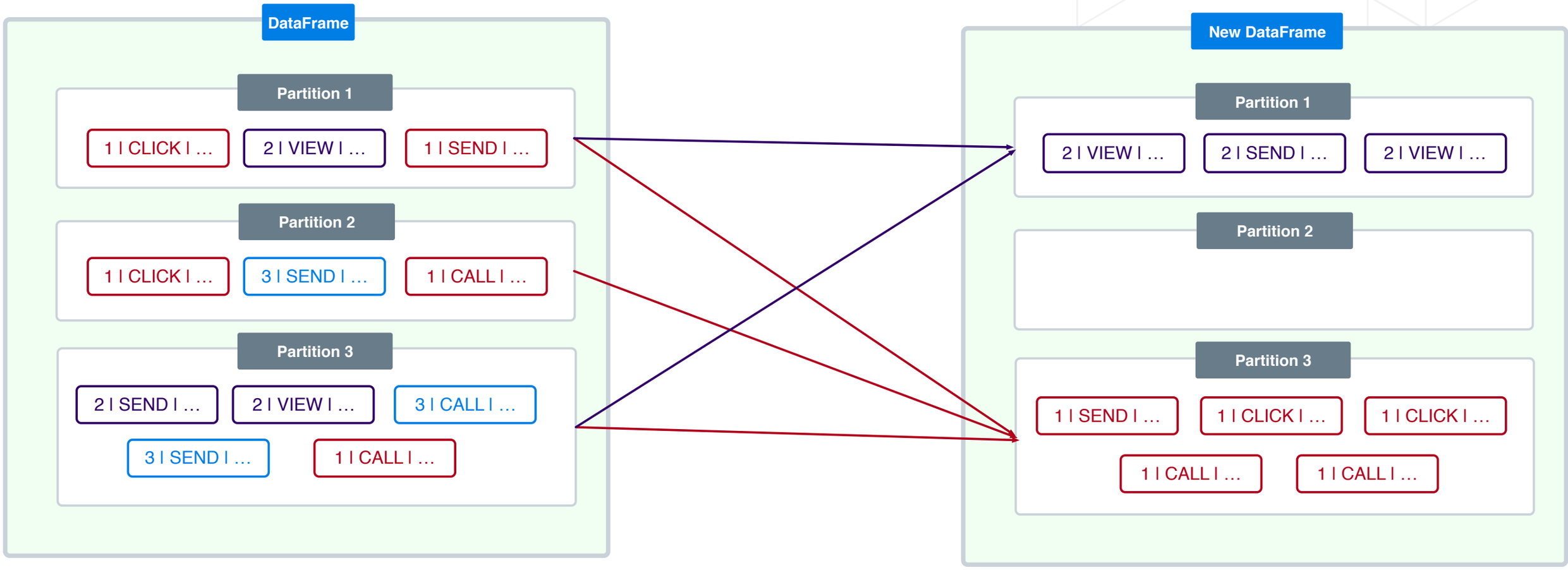
Wide transformations



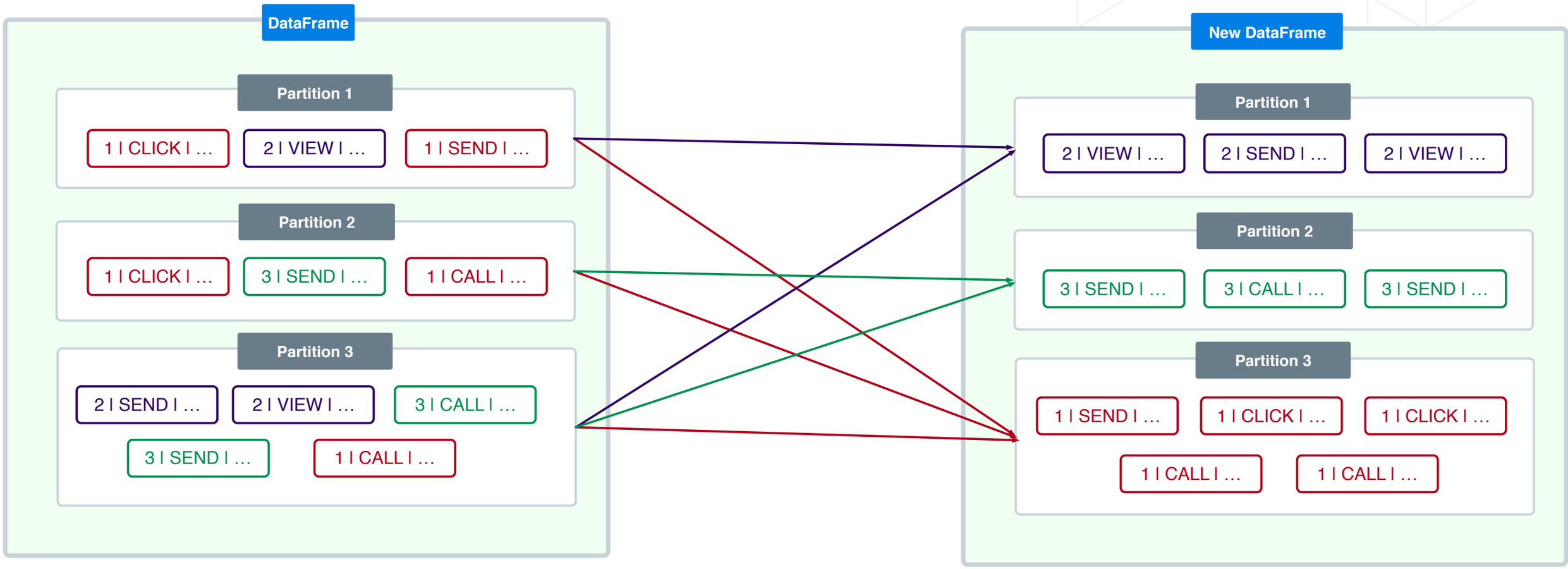
Wide transformations



Wide transformations



Wide transformations

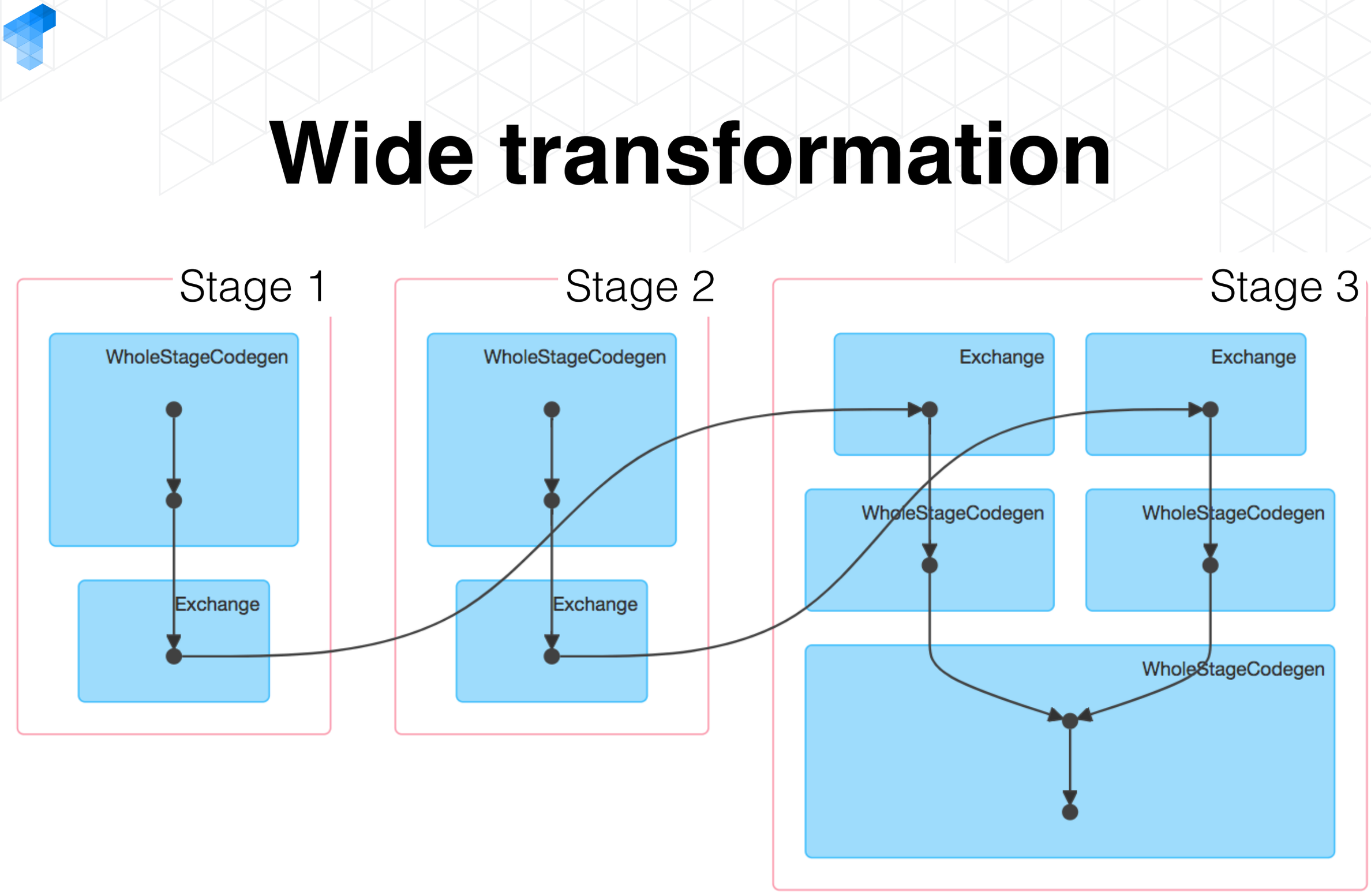


Wide transformations

- join
- groupBy
- repartition
- coalesce
- distinct
- …

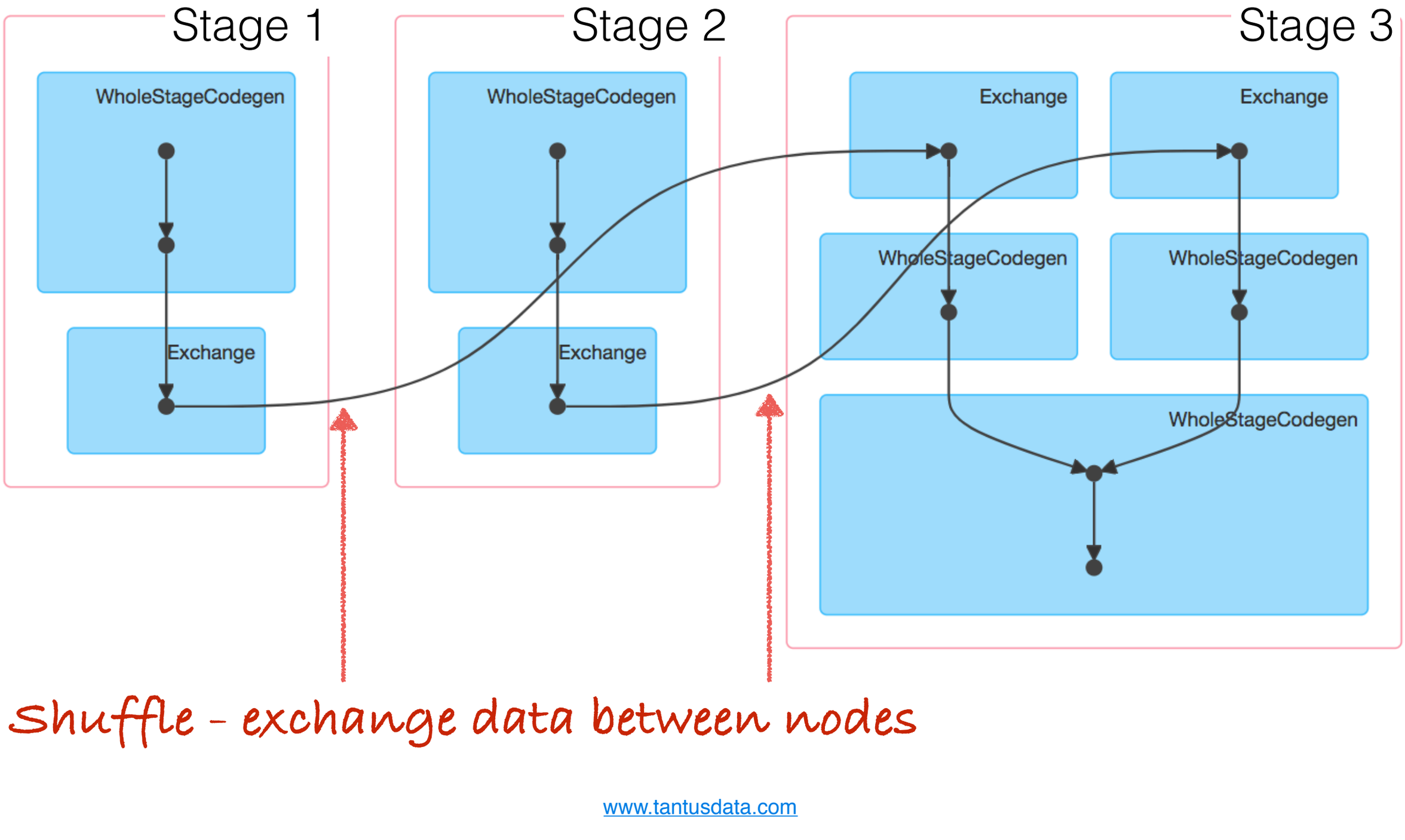
Wide transformations

```
events
  .join(users, users("id") === events("userId"))
//Extra logic
  .write
  .parquet(output)
```



Wide transformation

Wide transformation



Shuffle - exchange data between nodes

Simplest scenario ever

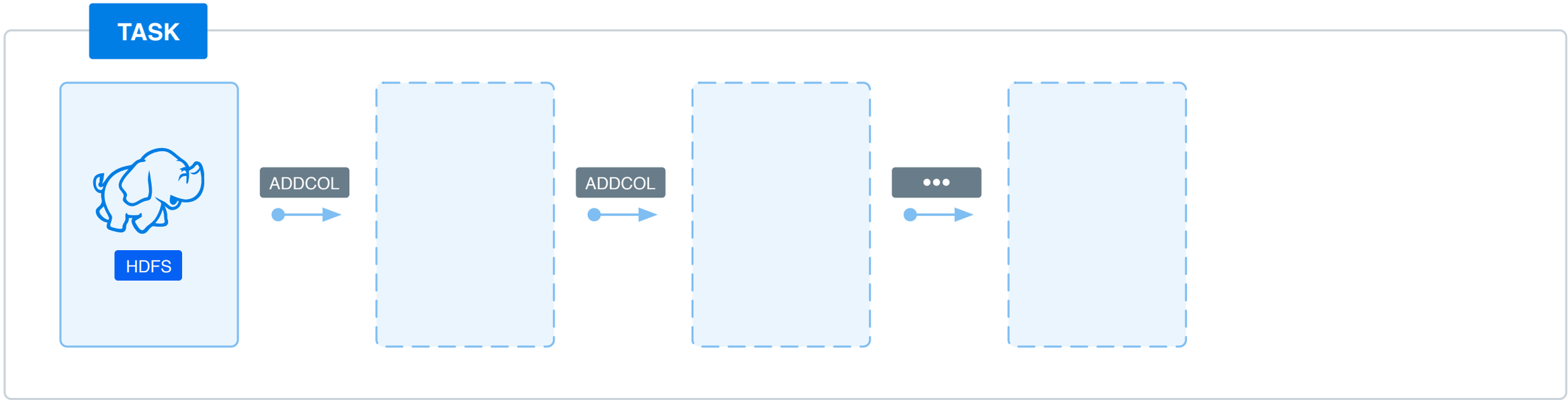
```
val df = spark.read.parquet(“…”)
```

TASK

HDFS

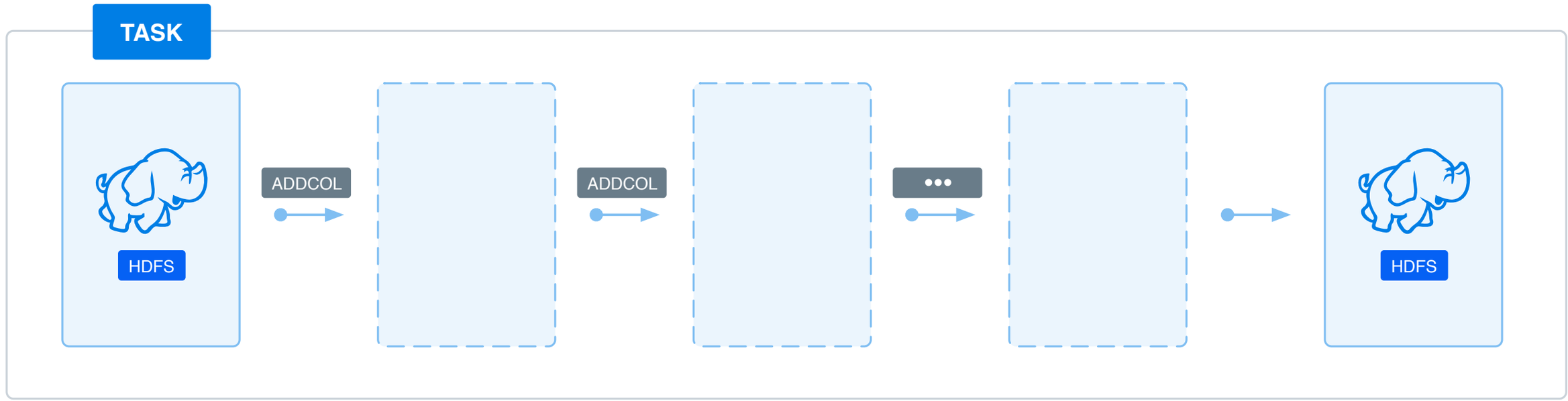
Simplest scenario ever

```
val df = spark.read.parquet(“…”)
df
   .withColumn("year", year(col(“timestamp")))
   .withColumn("month", month(col(“timestamp”)))
   .withColumn("day", dayofmonth(col(“timestamp”)))
```



Simplest scenario ever

```
val df = spark.read.parquet(“…”)
df
  .withColumn("year", year(col(“timestamp”)))
  .withColumn("month", month(col("timestamp")))
  .withColumn("day", dayofmonth(col("timestamp")))
  .write.save(output)
```



Simplest scenario ever

1TB of events

8000 blocks

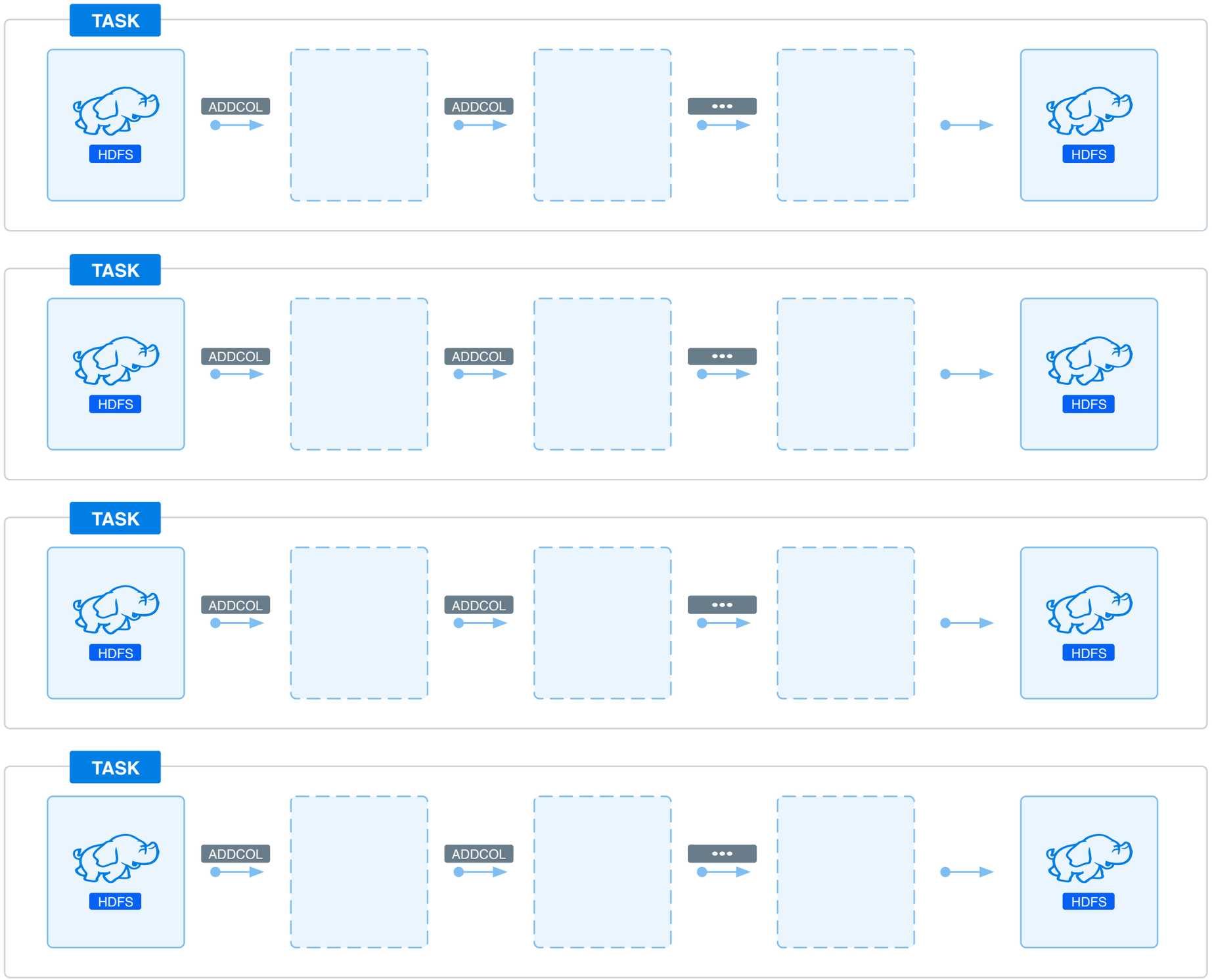
Simplest scenario ever

1TB of events

8000 blocks

8000 tasks

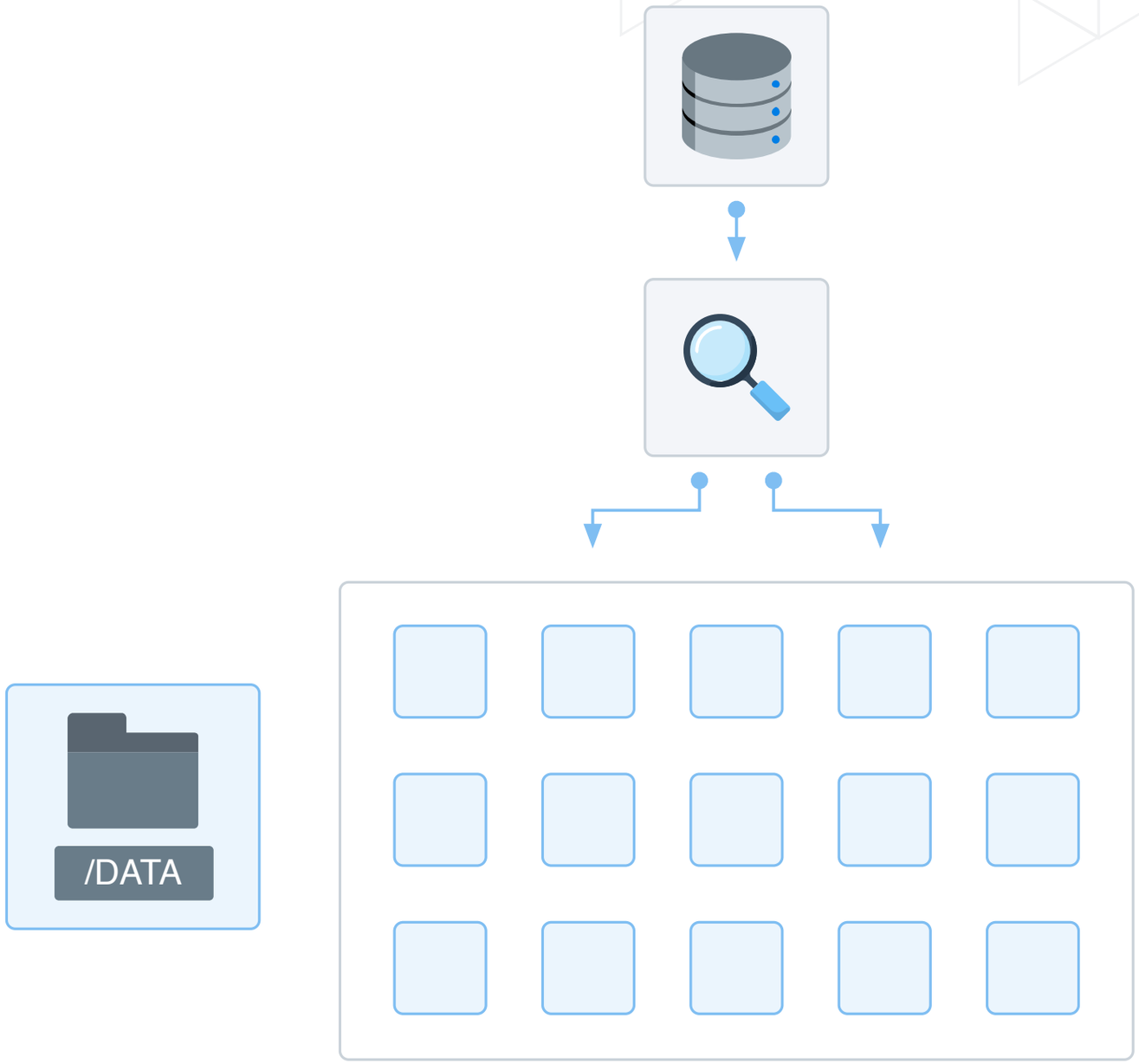
8000


TASK
HDFS
ADDCOL
ADDCOL
HDFS
TASK
HDFS
ADDCOL
ADDCOL
HDFS
TASK
HDFS
ADDCOL
ADDCOL
HDFS
TASK
HDFS
ADDCOL
ADDCOL
HDFS

Case #1

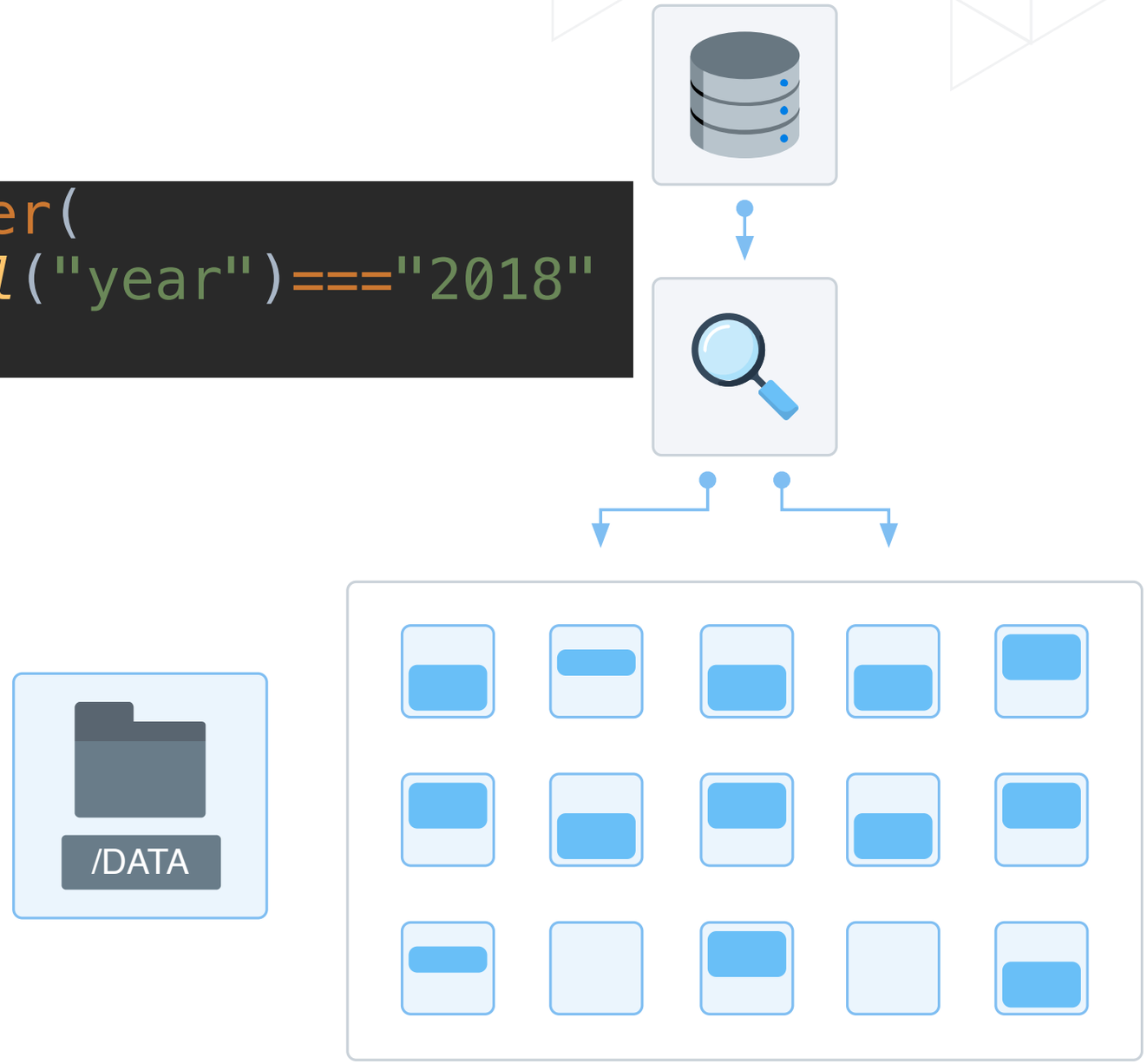
- Faster queries with date condition?
- Organize your data by date!

Case #1

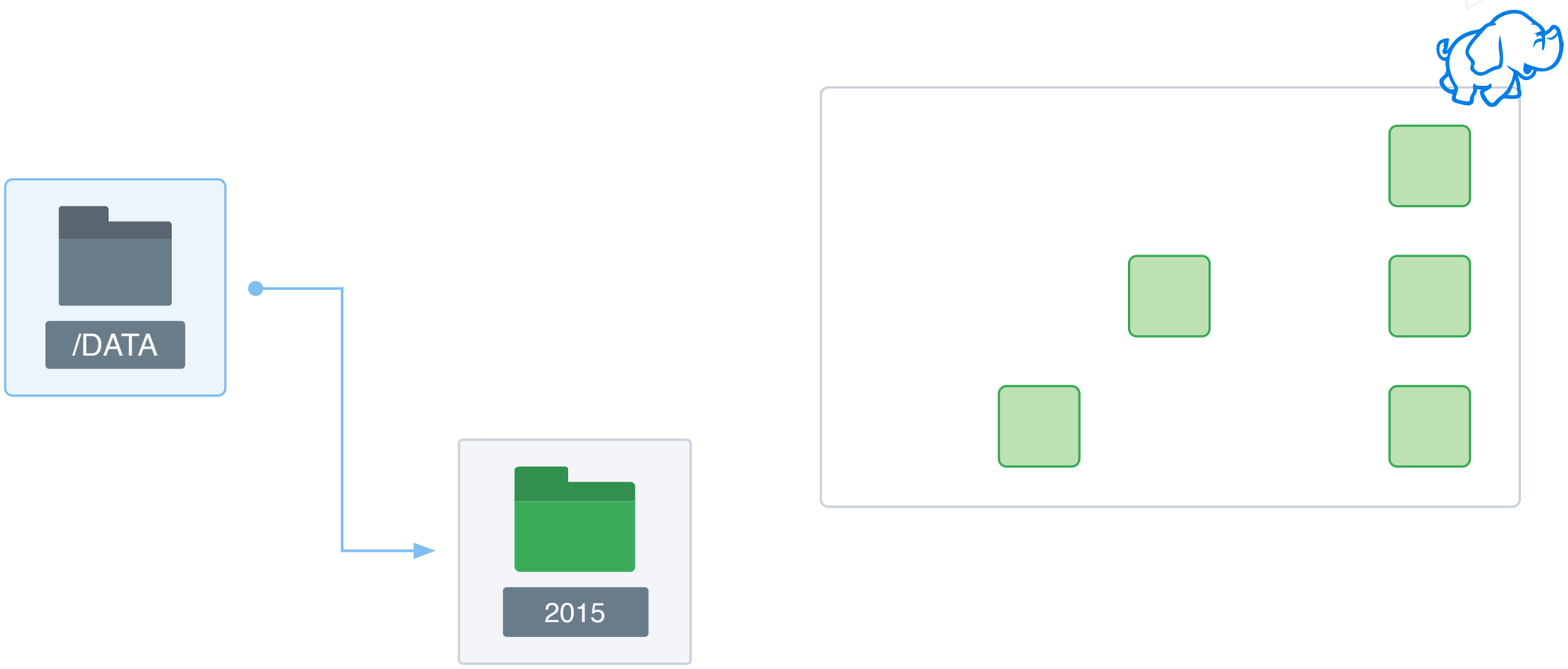


Case #1

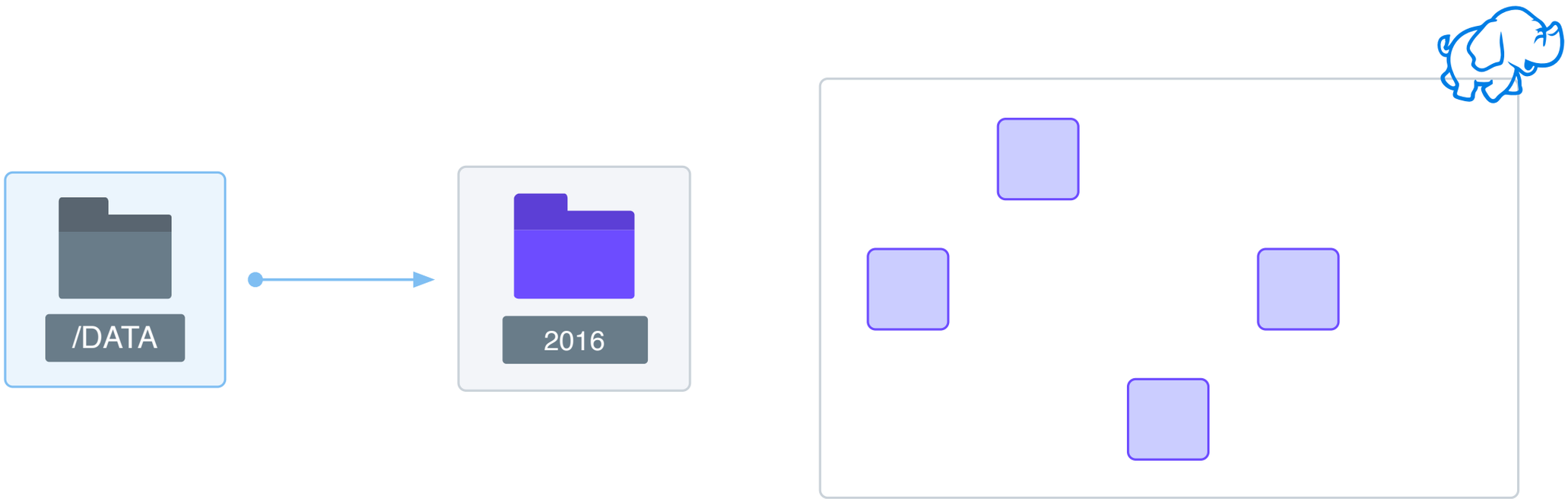
```
.filter(
    col("year")==="2018"
)
```



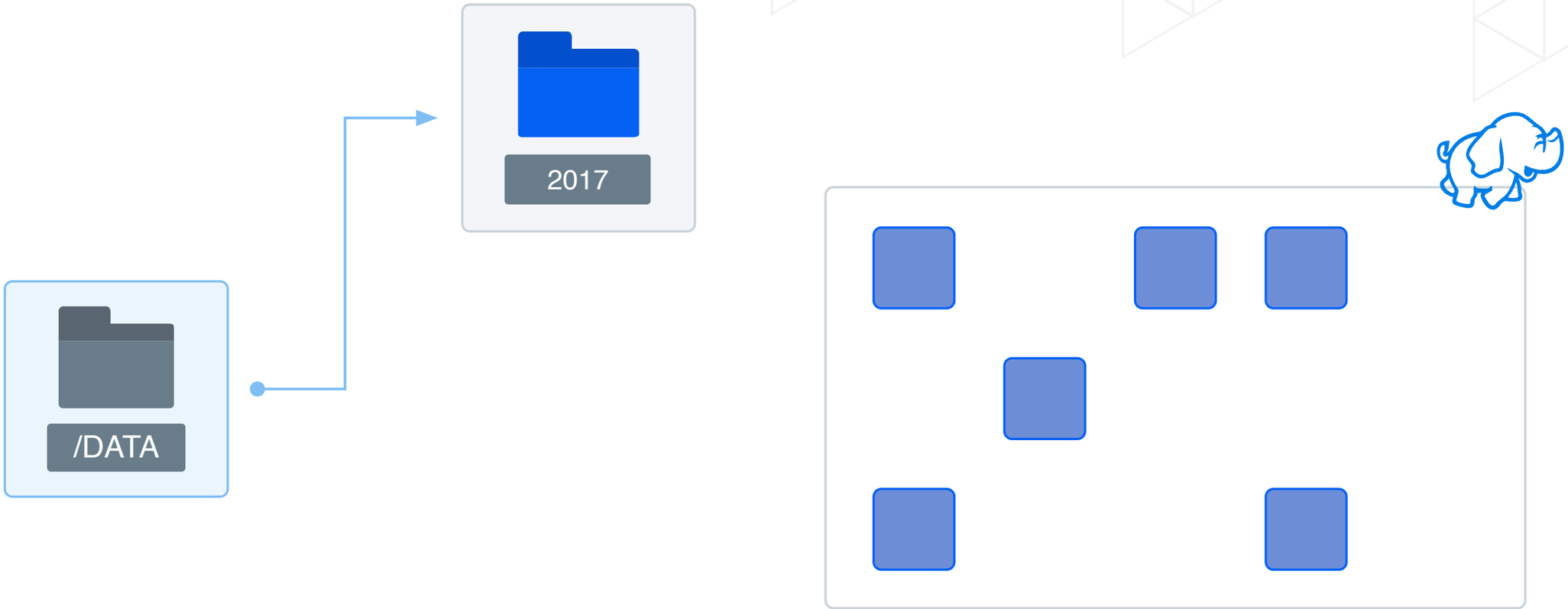
Case #1



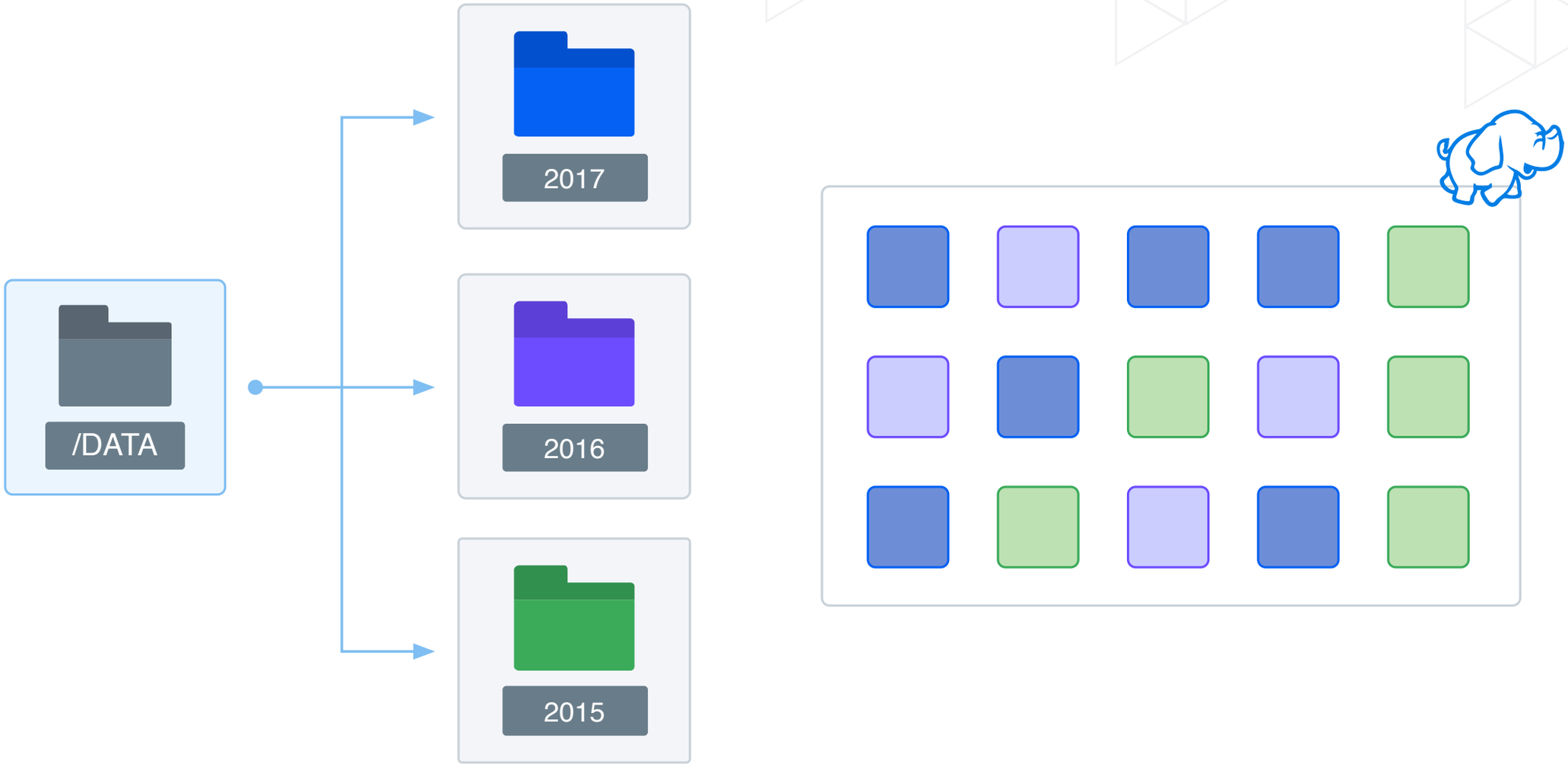
Case #1



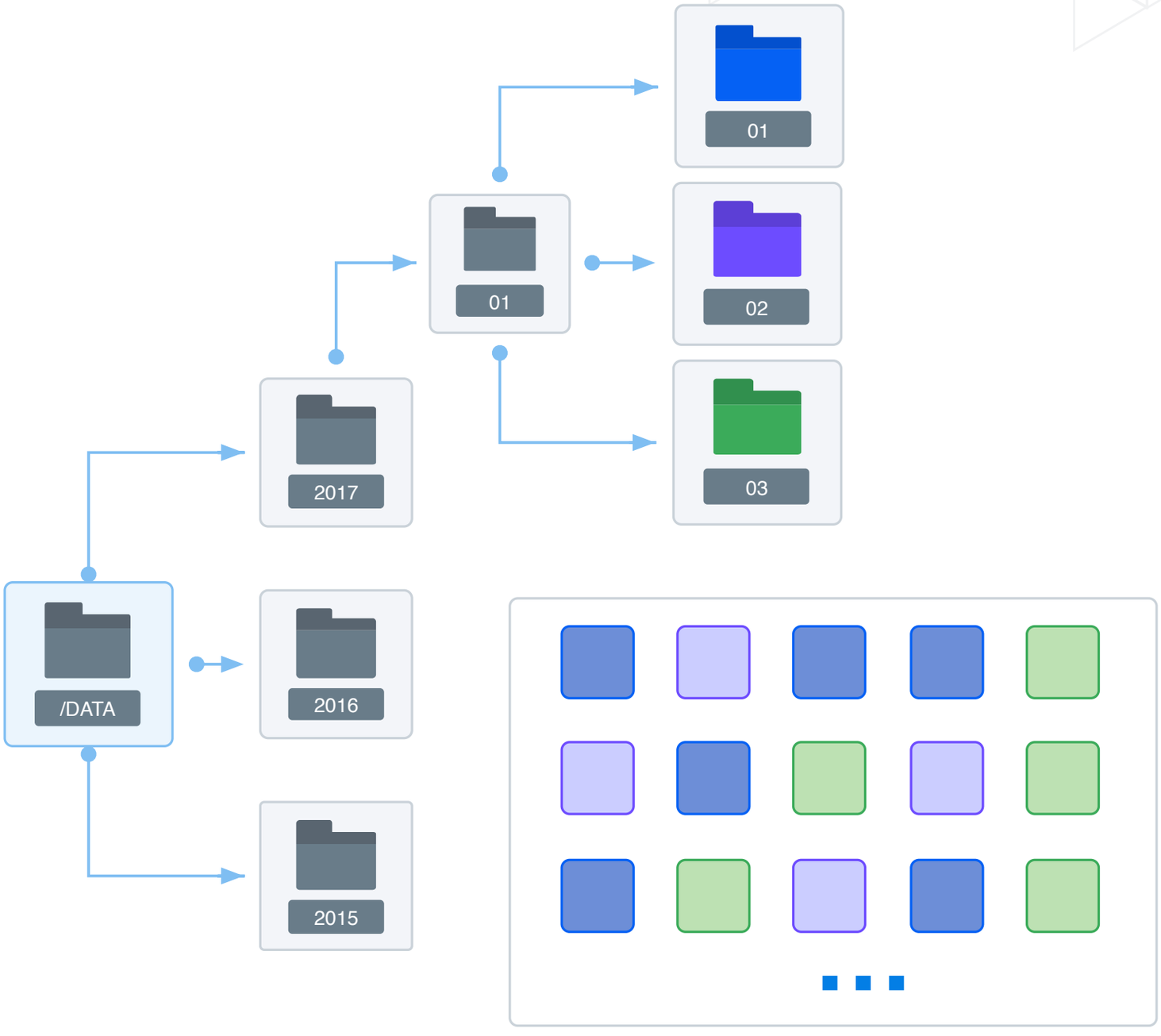
Case #1



Case #1



Case #1



Case #1

```
val df = spark.read.parquet(“…”)
df
  .withColumn("year", year(col(“timestamp”)))
  .withColumn("month", month(col("timestamp")))
  .withColumn("day", dayofmonth(col("timestamp")))
   .partitionBy("year", "month", "day")
  .write.save(output)
```

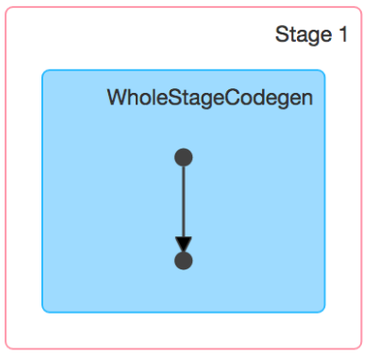
Case #1

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | **MarcinTest** application UI

Details for Job 1

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
1	save at Demo.scala:105 +details	2018/05/30 19:22:28	8.2 min	123/123	11.6 GB			

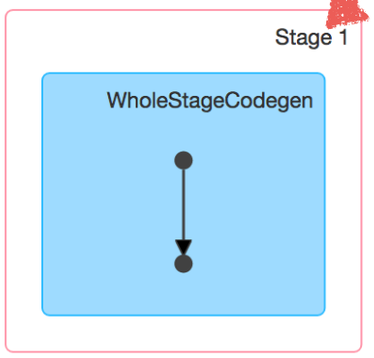
Case #1

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | **MarcinTest** application UI

Details for Job 1

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
1	save at Demo.scala:105 +details	2018/05/30 19:22:28	8.2 min	123/123	11.6 GB			

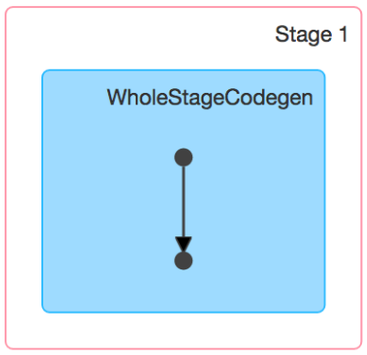
Case #1

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | **MarcinTest** application UI

Details for Job 1

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
1	save at Demo.scala:105 +details	2018/05/30 19:22:28	8.2 min	123/123	11.6 GB			

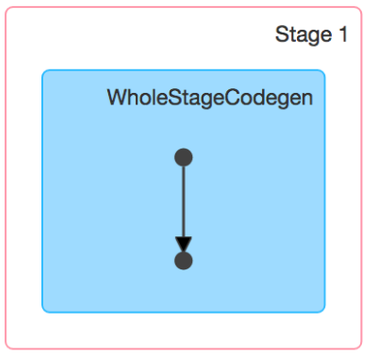
Case #1

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | **MarcinTest** application UI

Details for Job 1

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
1	save at Demo.scala:105 +details	2018/05/30 19:22:28	8.2 min	123/123	11.6 GB			

Case #1

Tasks (123)

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a page. Go

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Input Size / Records	Errors
0	18	0	SUCCESS	NODE_LOCAL	14 / [redacted] stdout stderr	2018/05/30 19:23:49	1.4 min	1 s	0.0 B / 1236428	
1	7	0	SUCCESS	NODE_LOCAL	7 / [redacted] stdout stderr	2018/05/30 19:22:58	1.6 min	1 s	0.0 B / 1236085	
2	1	0	SUCCESS	NODE_LOCAL	1 / [redacted] stdout stderr	2018/05/30 19:22:28	1.8 min	0.7 s	128.0 MB / 1235920	
3	26	0	SUCCESS	NODE_LOCAL	27 / [redacted] stdout stderr	2018/05/30 19:24:02	1.3 min	0.9 s	0.0 B / 1235831	
4	14	0	SUCCESS	NODE_LOCAL	18 / [redacted] stdout stderr	2018/05/30 19:23:39	1.3 min	0.8 s	0.0 B / 1236428	
5	47	0	SUCCESS	RACK_LOCAL	20 / [redacted] stdout stderr	2018/05/30 19:25:12	1.3 min	0.6 s	128.0 MB / 1236098	
6	16	0	SUCCESS	NODE_LOCAL	17 / [redacted] stdout stderr	2018/05/30 19:23:47	1.3 min	0.7 s	0.0 B / 1235936	
7	44	0	SUCCESS	NODE_LOCAL	18 / [redacted] stdout stderr	2018/05/30 19:25:01	1.2 min	0.4 s	128.0 MB / 1235858	
8	6	0	SUCCESS	NODE_LOCAL	6 / [redacted] stdout stderr	2018/05/30 19:22:53	1.7 min	2 s	0.0 B / 1236428	
9	8	0	SUCCESS	NODE_LOCAL	8 / [redacted] stdout stderr	2018/05/30 19:22:58	1.7 min	1 s	0.0 B / 1236096	
10	27	0	SUCCESS	NODE_LOCAL	26 / [redacted] stdout stderr	2018/05/30 19:24:03	1.3 min	0.9 s	0.0 B / 1235921	
11	5	0	SUCCESS	NODE_LOCAL	3 / [redacted] stdout stderr	2018/05/30 19:22:48	1.8 min	0.9 s	0.0 B / 1235838	

Case #1

Tasks (123)

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a page. Go

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Input Size / Records	Errors
0	18	0	SUCCESS	NODE_LOCAL	14 / [redacted] stdout stderr	2018/05/30 19:23:49	1.4 min	1 s	0.0 B / 1236428	
1	7	0	SUCCESS	NODE_LOCAL	7 / [redacted] stdout stderr	2018/05/30 19:22:58	1.6 min	1 s	0.0 B / 1236085	
2	1	0	SUCCESS	NODE_LOCAL	1 / [redacted] stdout stderr	2018/05/30 19:22:28	1.8 min	0.7 s	128.0 MB / 1235920	
3	26	0	SUCCESS	NODE_LOCAL	27 / [redacted] stdout stderr	2018/05/30 19:24:02	1.3 min	0.9 s	0.0 B / 1235831	
4	14	0	SUCCESS	NODE_LOCAL	18 / [redacted] stdout stderr	2018/05/30 19:23:39	1.3 min	0.8 s	0.0 B / 1236428	
5	47	0	SUCCESS	RACK_LOCAL	20 / [redacted] stdout stderr	2018/05/30 19:25:12	1.3 min	0.6 s	128.0 MB / 1236098	
6	16	0	SUCCESS	NODE_LOCAL	17 / [redacted] stdout stderr	2018/05/30 19:23:47	1.3 min	0.7 s	0.0 B / 1235936	
7	44	0	SUCCESS	NODE_LOCAL	18 / [redacted] stdout stderr	2018/05/30 19:25:01	1.2 min	0.4 s	128.0 MB / 1235858	
8	6	0	SUCCESS	NODE_LOCAL	6 / [redacted] stdout stderr	2018/05/30 19:22:53	1.7 min	2 s	0.0 B / 1236428	
9	8	0	SUCCESS	NODE_LOCAL	8 / [redacted] stdout stderr	2018/05/30 19:22:58	1.7 min	1 s	0.0 B / 1236096	
10	27	0	SUCCESS	NODE_LOCAL	26 / [redacted] stdout stderr	2018/05/30 19:24:03	1.3 min	0.9 s	0.0 B / 1235921	
11	5	0	SUCCESS	NODE_LOCAL	3 / [redacted] stdout stderr	2018/05/30 19:22:48	1.8 min	0.9 s	0.0 B / 1235838	

Case #1

```
$ hdfs dfs -ls
```

Case #1

```
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1
```

Case #1

```
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
```

Case #1

```
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
22
```

Case #1

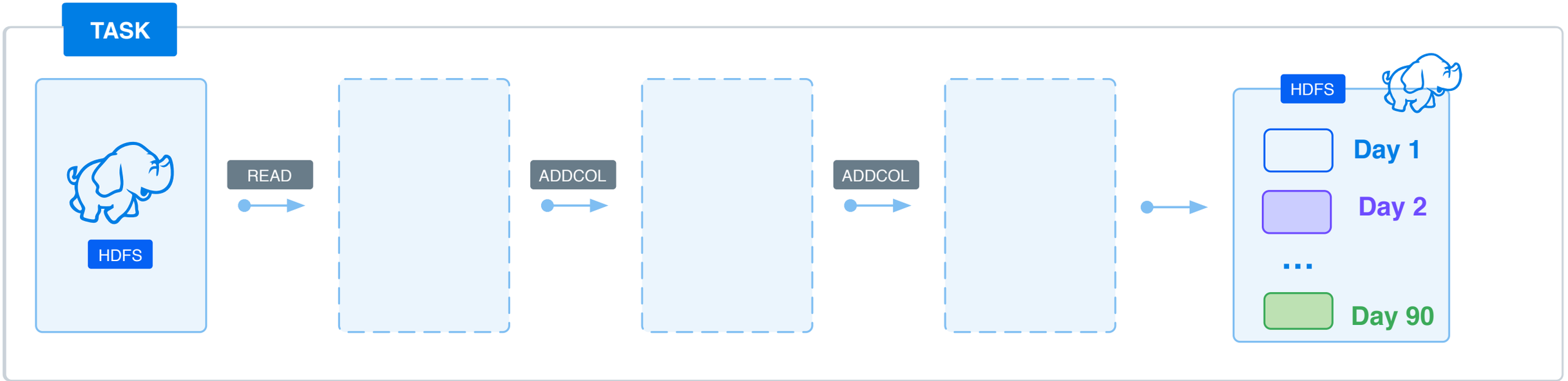
```
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
22
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
25
```

Case #1

```
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
22
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
25
$ hdfs dfs -ls /user/marcin/out/year=2019/month=3/day=1 | wc -l
40
```

Case #1

```
val df = spark.read.parquet(“…”)
df
  .withColumn("year", year(col(“timestamp”)))
  .withColumn("month", month(col("timestamp")))
  .withColumn("day", dayofmonth(col("timestamp")))
   .partitionBy("year", "month", "day")
  .write.save(output)
```



TASK
HDFS
READ
ADDCOL
ADDCOL
HDFS
Day 1
Day 2
...
Day 90
TASK
HDFS
READ
ADDCOL
ADDCOL
HDFS
Day 1
Day 2
...
Day 90
TASK
HDFS
READ
ADDCOL
ADDCOL
HDFS
Day 1
Day 2
...
Day 90

Case #1

90 days of data

8000 blocks

Case #1

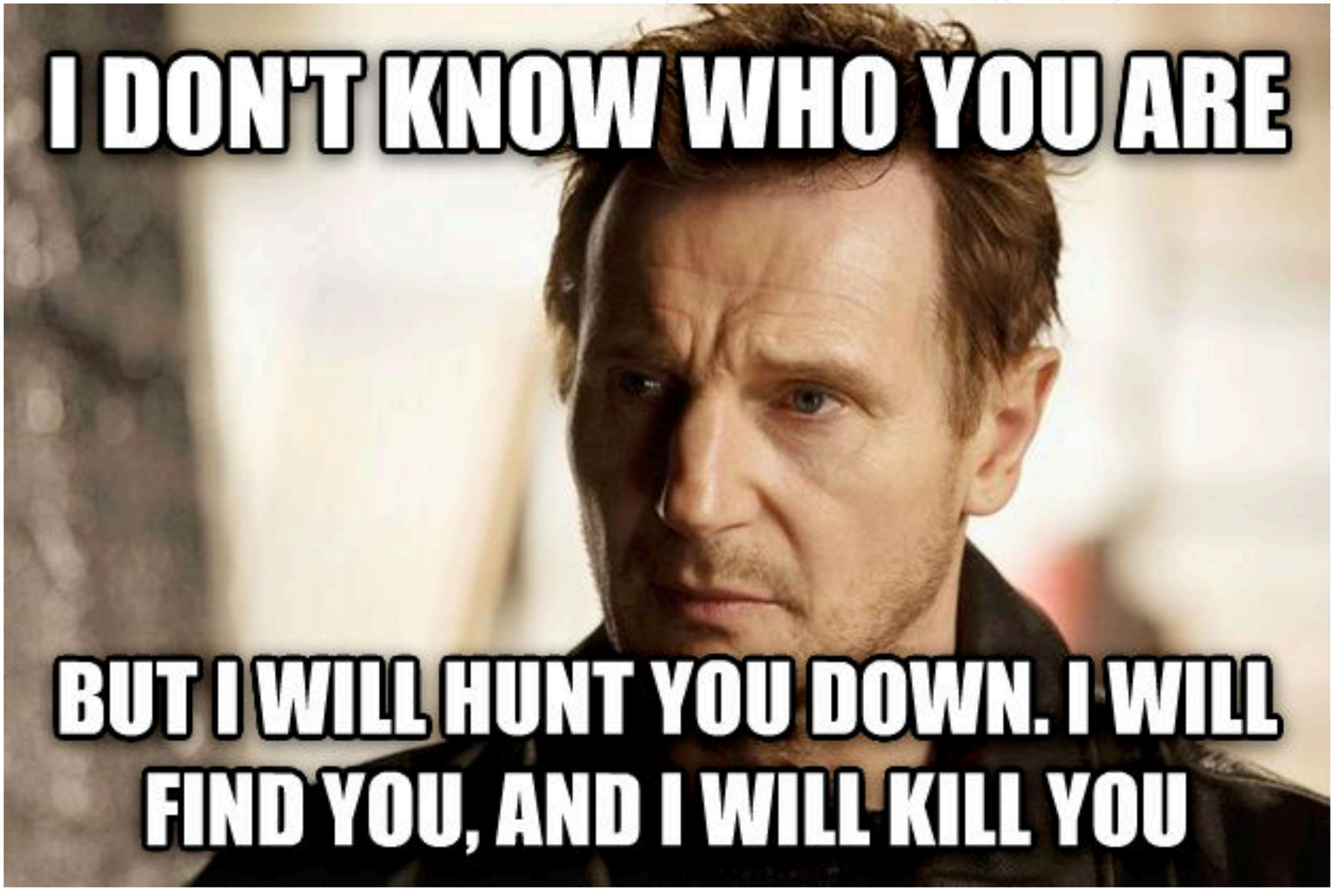
90 days of data

8000 blocks

90*8000 files produced

Consequences

- Slow!
- Downstream jobs will struggle
- Memory pressure on NameNode
- Might kill the cluster!



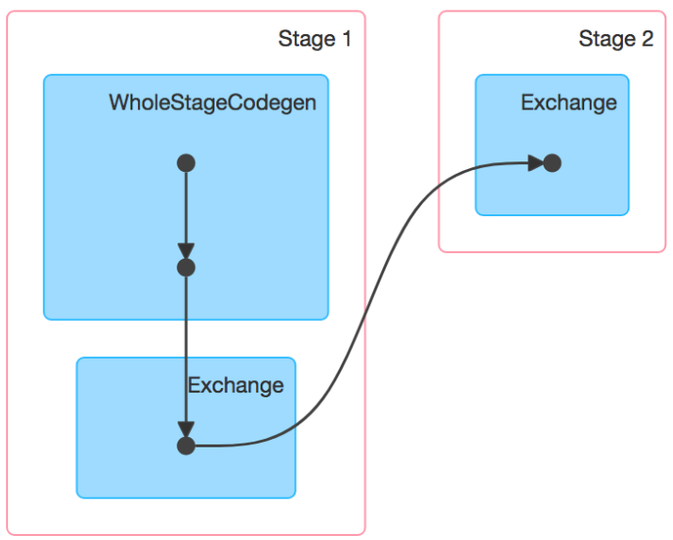
Consequences

Case #1

- repartition before partitionBy in order to limit #files

Repartition

```
df
  .withColumn("year", year(col("eventTimestamp")))
  .withColumn("month", month(col("eventTimestamp")))
  .withColumn("day", dayofmonth(col("eventTimestamp")))
.repartition(col("year"), col("month"), col("day"))
  .write
  .partitionBy("year", "month", "day")
  .save(output)
```

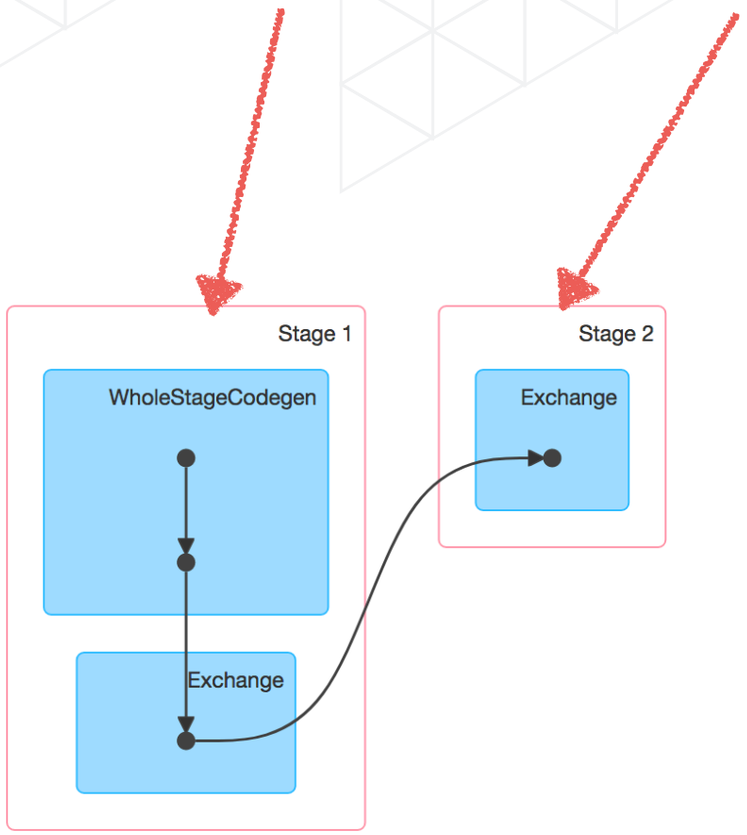


Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	save at Demo.scala:147 +details (kill)	2018/06/04 21:28:25	23 s	197/200			7.2 GB	

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	save at Demo.scala:147 +details	2018/06/04 21:27:51	34 s	240/240	26.3 GB			32.1 GB



Active Stages (1)

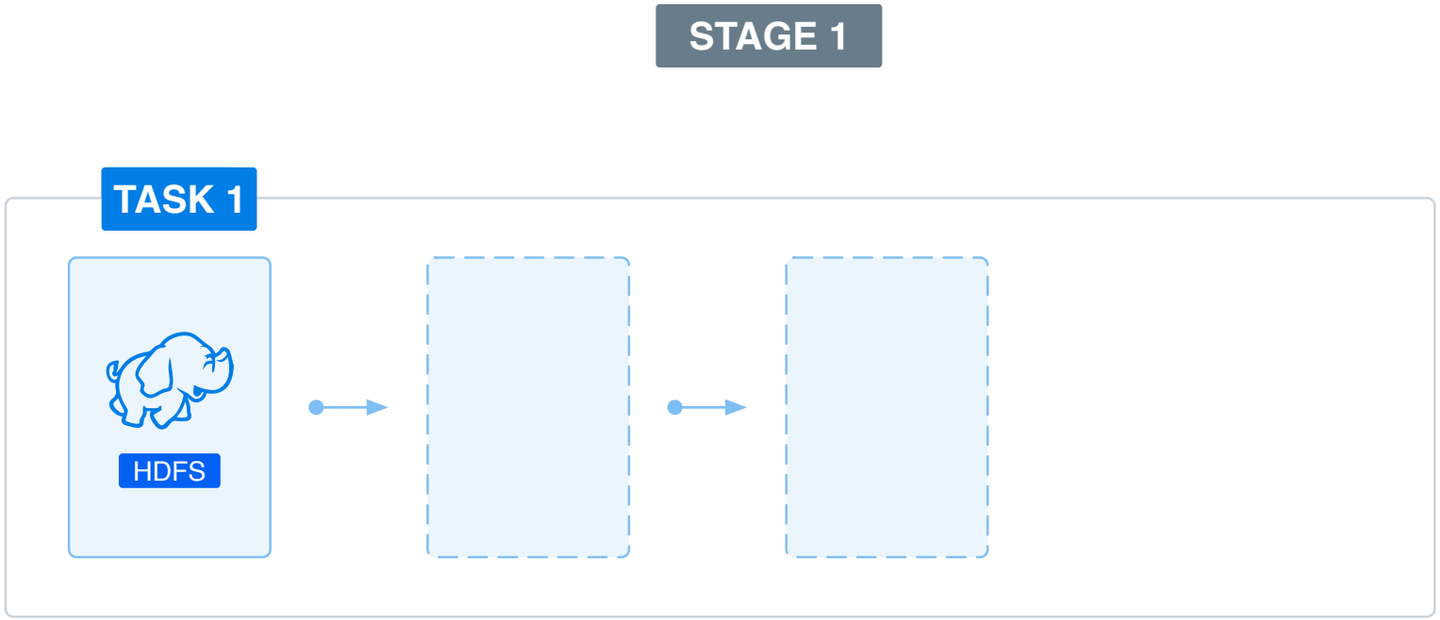
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	save at Demo.scala:147 +details (kill)	2018/06/04 21:28:25	23 s	197/200			7.2 GB	

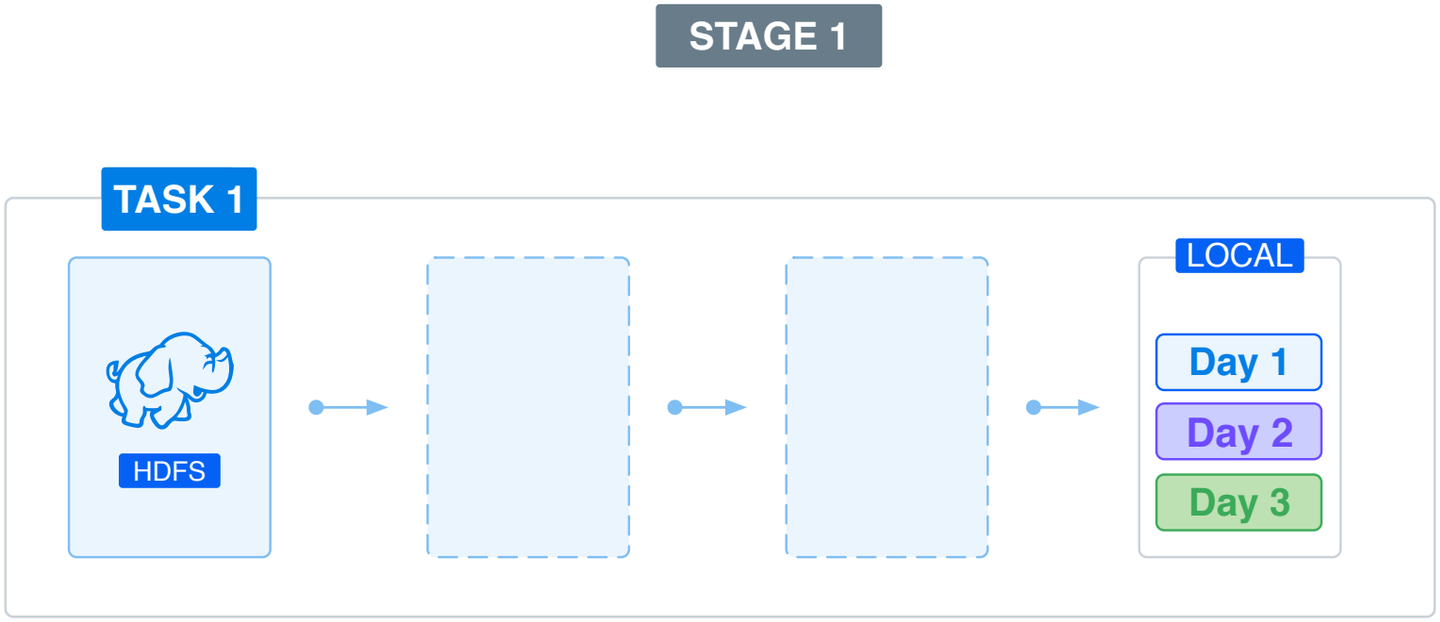
Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	save at Demo.scala:147 +details	2018/06/04 21:27:51	34 s	240/240	26.3 GB			32.1 GB

STAGE 1

TASK 1


STAGE 1
TASK 1
HDFS


STAGE 1
TASK 1
HDFS
LOCAL
Day 1
Day 2
Day 3

STAGE 1

TASK 1

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 2

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK X

HDFS

LOCAL

Day 1

Day 2

Day 3

STAGE 1

STAGE 2

TASK 1

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 2

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK X

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 1

STAGE 1

STAGE 2

TASK 1

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 1

TASK 2

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 2

TASK X

HDFS

LOCAL

Day 1

Day 2

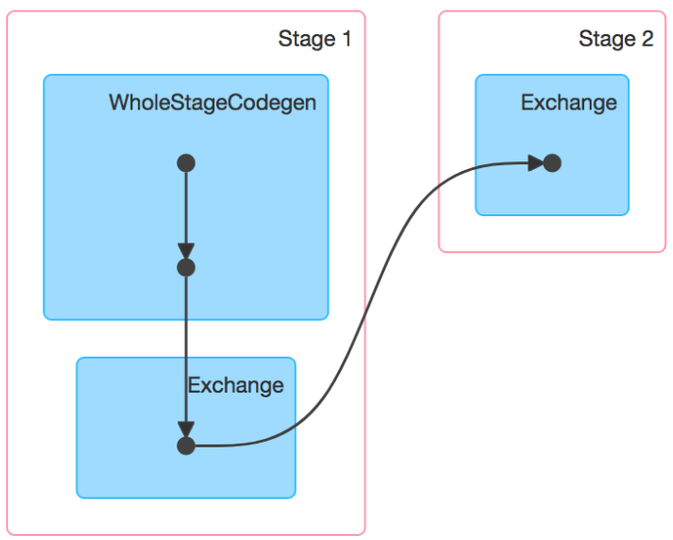
Day 3

Case #1 - summary

- each *partitionBy* task may create many files in HDFS
- *repartition* to the rescue

Case #2

- It fails after repartitioning…

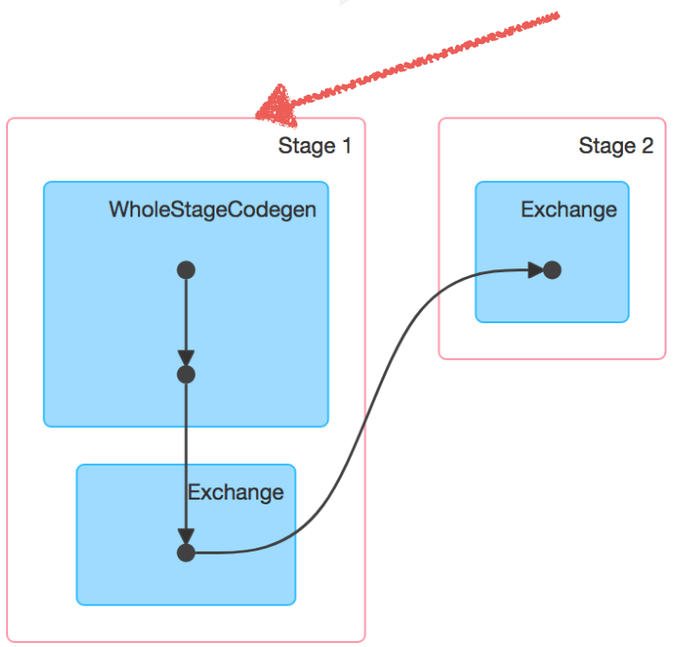


Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	save at Demo.scala:147 +details (kill)	2018/06/04 21:28:25	23 s	197/200			7.2 GB	

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	save at Demo.scala:147 +details	2018/06/04 21:27:51	34 s	240/240	26.3 GB			32.1 GB

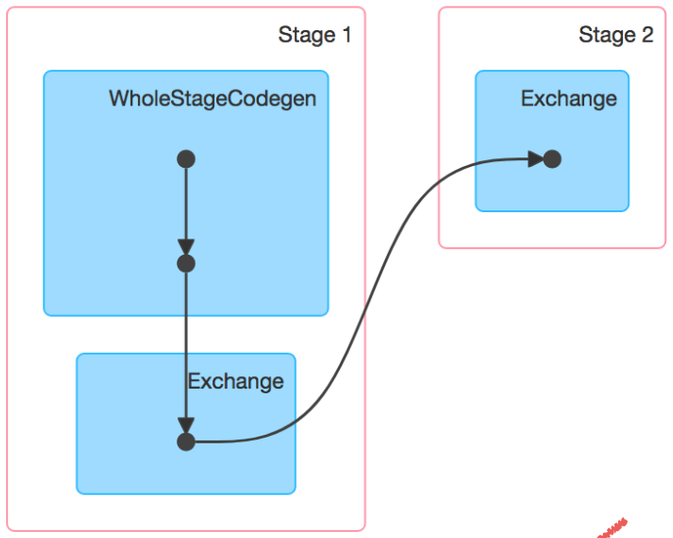


Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	save at Demo.scala:147 +details (kill)	2018/06/04 21:28:25	23 s	197/200			7.2 GB	

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	save at Demo.scala:147 +details	2018/06/04 21:27:51	34 s	240/240	26.3 GB			32.1 GB



Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
2	save at Demo.scala:147 +details (kill)	2018/06/04 21:28:25	23 s	197/200			7.2 GB	

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	save at Demo.scala:147 +details	2018/06/04 21:27:51	34 s	240/240	26.3 GB			32.1 GB

Case #2

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a page

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
89	330	0	RUNNING	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	32 s	1 s	3.6 GB / 33554431
176	417	0	RUNNING	PROCESS_LOCAL	17 / [redacted] stdout stderr	2018/06/04 21:28:26	32 s	0.4 s	3.3 GB / 30452163
5	246	0	RUNNING	PROCESS_LOCAL	25 / [redacted] stdout stderr	2018/06/04 21:28:25	32 s	0.8 s	3.0 GB / 27962026
0	241	0	SUCCESS	PROCESS_LOCAL	16 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
1	242	0	SUCCESS	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
2	243	0	SUCCESS	PROCESS_LOCAL	20 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0

Case #2

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a pag

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
89	330	0	RUNNING	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	42 s	1 s	4.8 GB / 44739241
176	417	0	RUNNING	PROCESS_LOCAL	17 [redacted] stdout stderr	2018/06/04 21:28:26	41 s	0.4 s	4.2 GB / 39146836
5	246	0	RUNNING	PROCESS_LOCAL	25 [redacted] stdout stderr	2018/06/04 21:28:25	42 s	0.8 s	4.2 GB / 39146836
0	241	0	SUCCESS	PROCESS_LOCAL	16 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
1	242	0	SUCCESS	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
2	243	0	SUCCESS	PROCESS_LOCAL	20 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0

Case #2

Tasks (200)

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items in a pag

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
176	417	0	RUNNING	PROCESS_LOCAL	17 / [redacted] stdout stderr	2018/06/04 21:28:26	2.4 min	0.9 s	10.7 GB / 99988910
5	246	0	RUNNING	PROCESS_LOCAL	25 / [redacted] stdout stderr	2018/06/04 21:28:25	2.4 min	1 s	10.7 GB / 100000929
89	330	0	RUNNING	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	2.4 min	2 s	10.7 GB / 100010161
0	241	0	SUCCESS	PROCESS_LOCAL	16 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
1	242	0	SUCCESS	PROCESS_LOCAL	2 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0
2	243	0	SUCCESS	PROCESS_LOCAL	20 / [redacted] stdout stderr	2018/06/04 21:28:25	0.2 s		0.0 B / 0

Case #2

One day of events is 100G

STAGE 1

STAGE 2

TASK 1

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 1

TASK 2

HDFS

LOCAL

Day 1

Day 2

Day 3

TASK 2

TASK X

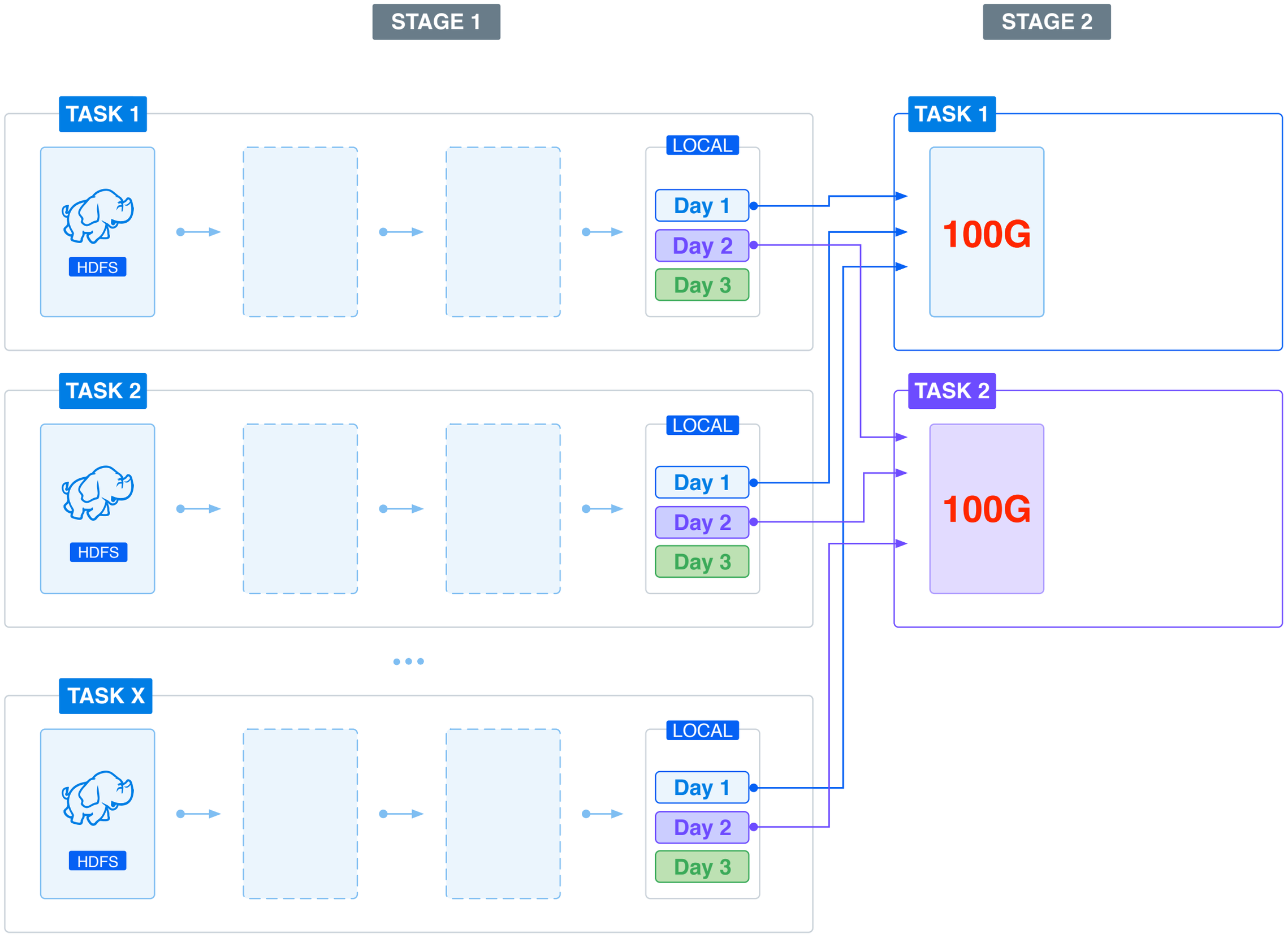
HDFS

LOCAL

Day 1

Day 2

Day 3


STAGE 1
STAGE 2
TASK 1
HDFS
LOCAL
Day 1
Day 2
Day 3
TASK 1
100G
TASK 2
HDFS
LOCAL
Day 1
Day 2
Day 3
TASK 2
100G
TASK X
HDFS
LOCAL
Day 1
Day 2
Day 3

Problems with shuffle

- Spill to disk
- Timeouts
- GC overhead limit exceeded
- OOM
- ExecutorLostFailure

Case #2

```
df
  .withColumn("year", year(col("eventTimestamp")))
  .withColumn("month", month(col("eventTimestamp")))
  .withColumn("day", dayofmonth(col("eventTimestamp")))
.repartition(col("year"), col("month"), col(“day”), col(”hour”))
  .write
  .partitionBy("year", "month", "day")
  .save(output)
```

```
.repartition(“year", "month",“day”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", "month",“day”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", "month",“day”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", “month”,“day”,”hour”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", “month”,“day”,”hour”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", “month”,“day”,”hour”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

```
.repartition(“year", “month”,“day”,”hour”)
```

day	month	year	hour	event	…
1	1	2017	11	**CALL**	
1	1	2017	1	**TEXT**	
1	1	2017	4	**CLICK**	
1	1	2017	7	**SEND**	
2	1	2017	1	**CLICK**	
2	1	2017	3	**TEXT**	
2	1	2017	1	**TEXT**	
2	1	2017	1	**CALL**	

Case #2

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records
21	262	0	RUNNING	PROCESS_LOCAL	114 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.5 s	1420.8 MB / 12756818
199	440	0	RUNNING	PROCESS_LOCAL	16 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.7 s	1407.3 MB / 12582910
8	249	0	RUNNING	PROCESS_LOCAL	210 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	1 s	1103.3 MB / 10006559
164	405	0	RUNNING	PROCESS_LOCAL	58 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.9 s	1102.8 MB / 10001580
3	244	0	RUNNING	PROCESS_LOCAL	40 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.9 s	1102.6 MB / 9999737
144	385	0	RUNNING	PROCESS_LOCAL	166 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	1 s	1102.2 MB / 9997449
45	286	0	RUNNING	PROCESS_LOCAL	79 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	1.0 s	1102.2 MB / 9998734
59	300	0	RUNNING	PROCESS_LOCAL	251 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.8 s	1102.2 MB / 9996061
101	342	0	RUNNING	PROCESS_LOCAL	249 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.9 s	1102.2 MB / 9998121
16	257	0	RUNNING	PROCESS_LOCAL	228 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.6 s	1102.1 MB / 9995163
98	339	0	RUNNING	PROCESS_LOCAL	226 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	1 s	1102.0 MB / 9996764
196	437	0	RUNNING	PROCESS_LOCAL	150 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	0.7 s	1097.6 MB / 10004622
109	350	0	RUNNING	PROCESS_LOCAL	89 / [redacted] stdout stderr	2018/06/07 10:19:23	18 s	1 s	1097.6 MB / 10004835

Case #2 - recap

- Know your data distribution when repartitioning
- Keep all your tasks busy - repartition by more columns if necessary

Case #3

```
events
  .join(users, users("id") === events("userId"))
//Extra logic
  .write
  .parquet(output)
```

Case #3

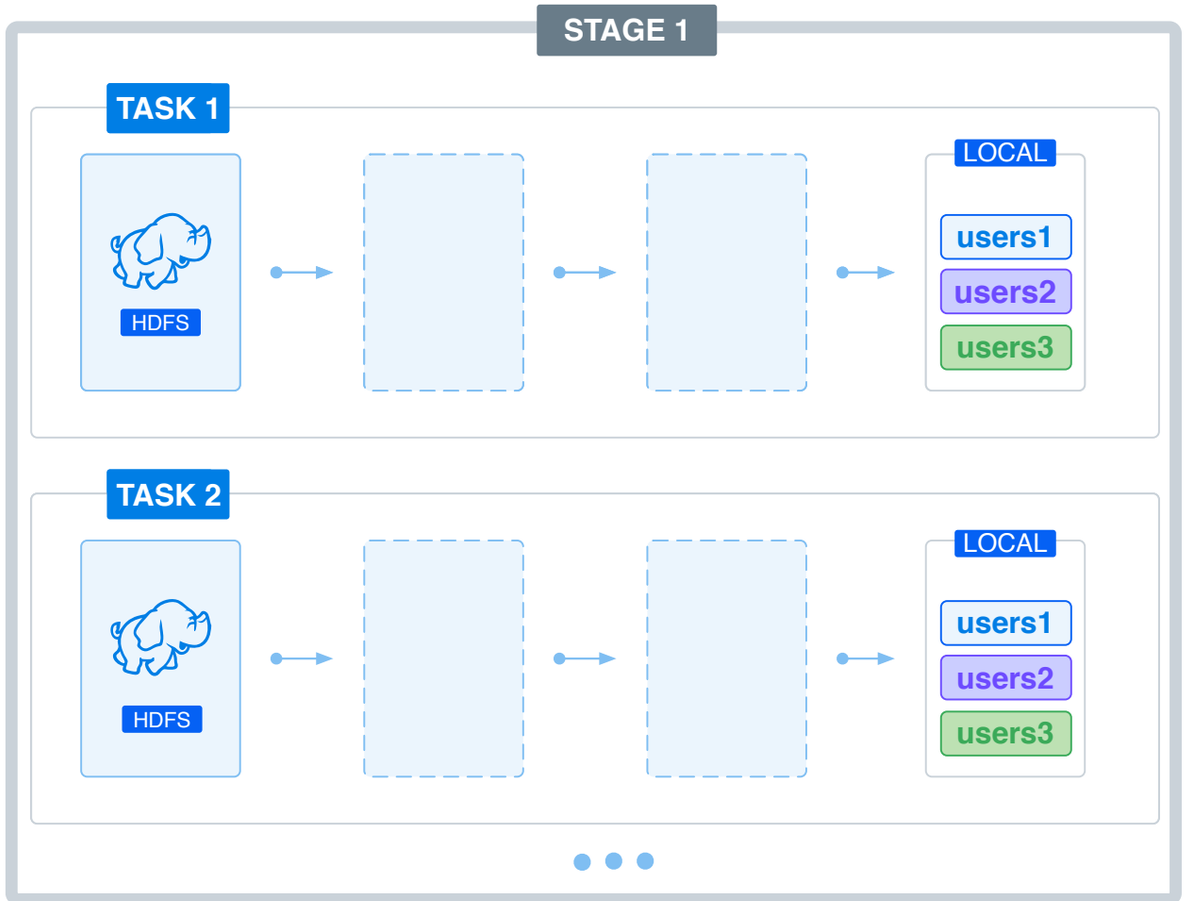
```
events
  .join(users, users("id") === events("userId"))
//Extra logic
  .write
  .parquet(output)
```

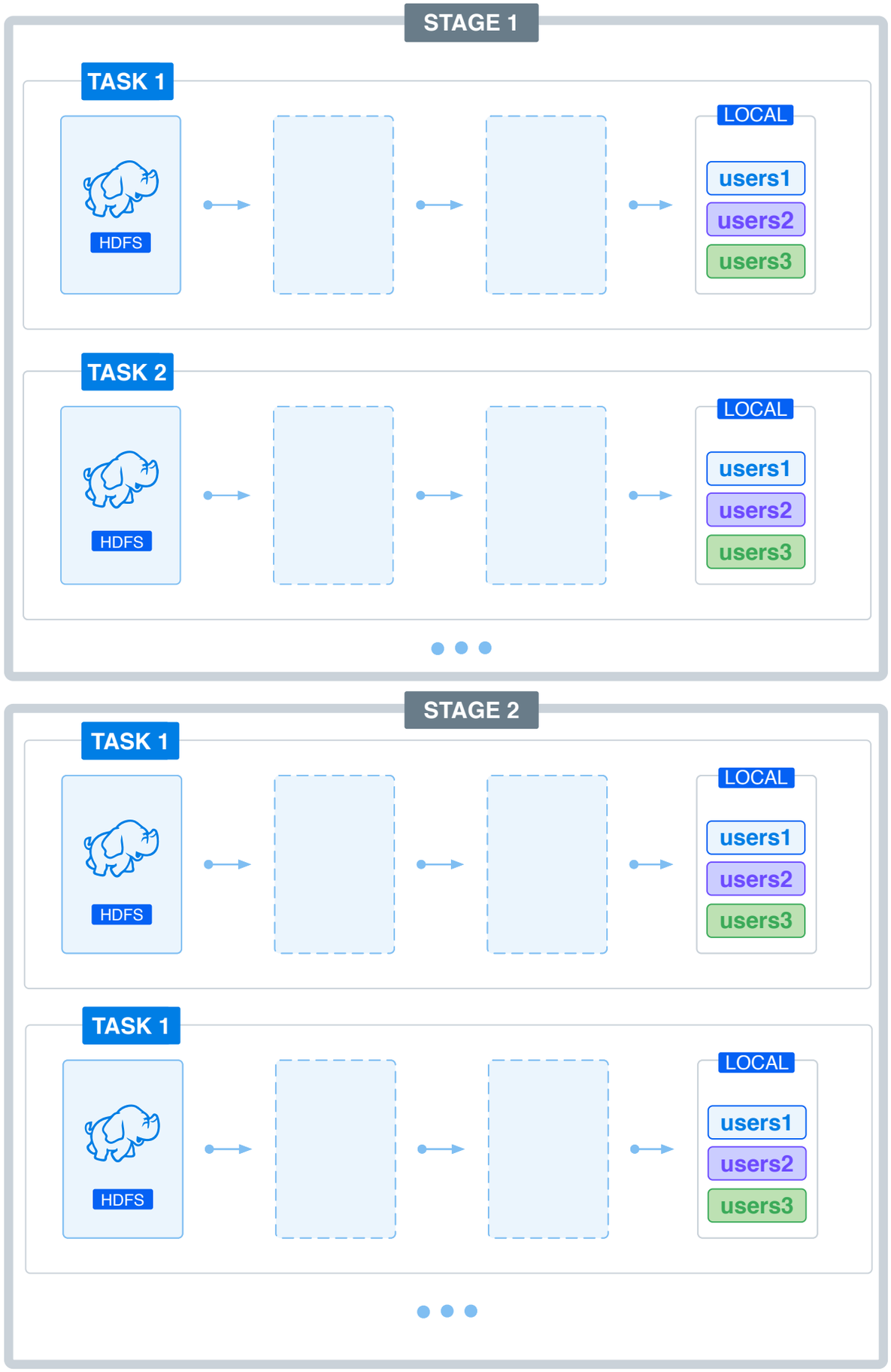
Case #3

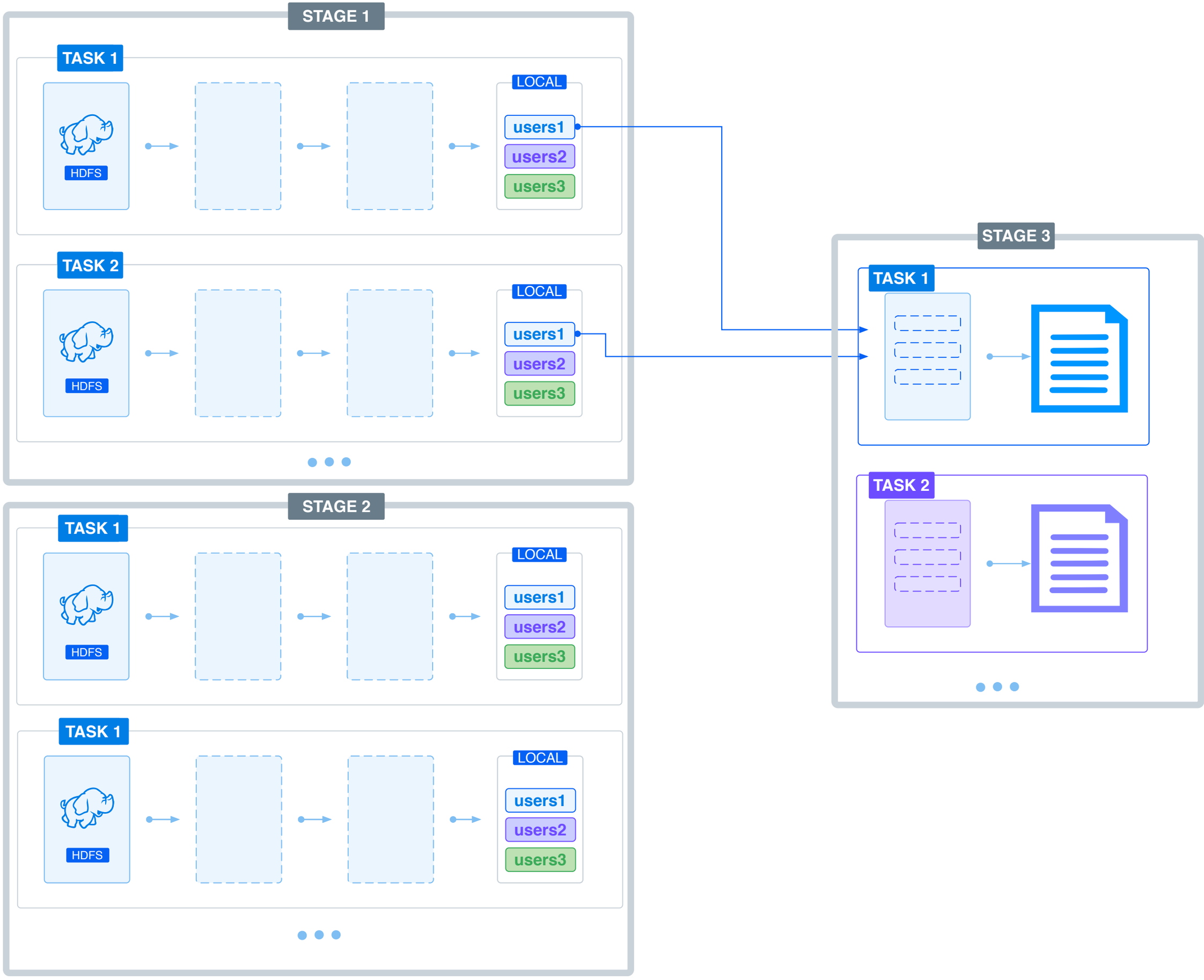
```
events
  .join(users, users("id") === events("userId"))
//Extra logic
  .write
  .parquet(output)
```

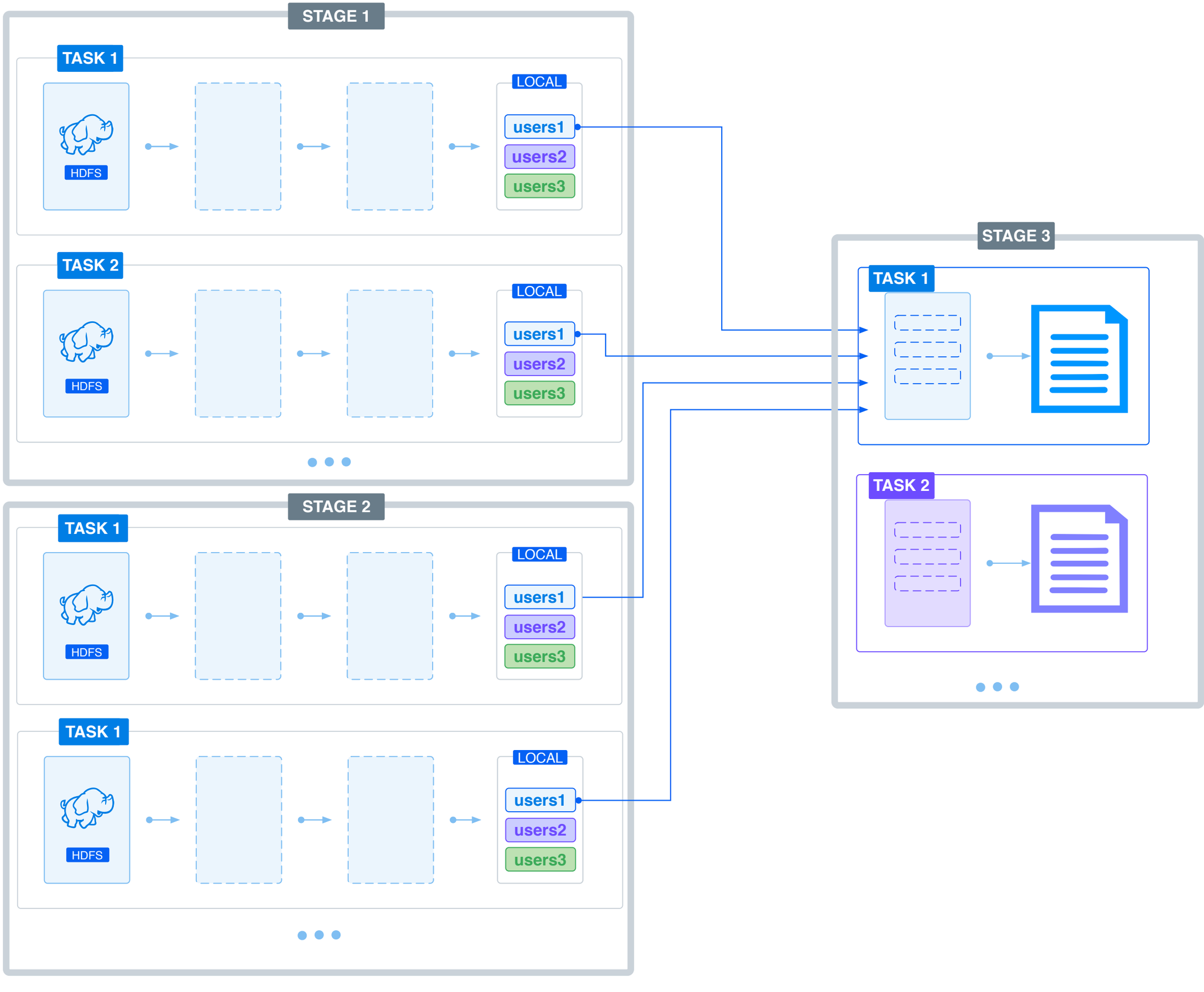
Case #3

```
events
  .join(users,users(“id")===events("userId"))
//Extra logic
  .write
  .parquet(output)
```


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3
STAGE 3
TASK 1
TASK 2


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3
STAGE 3
TASK 1
TASK 2

STAGE 1

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 2

HDFS

LOCAL

users1

users2

users3

STAGE 2

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 1

HDFS

LOCAL

users1

users2

users3

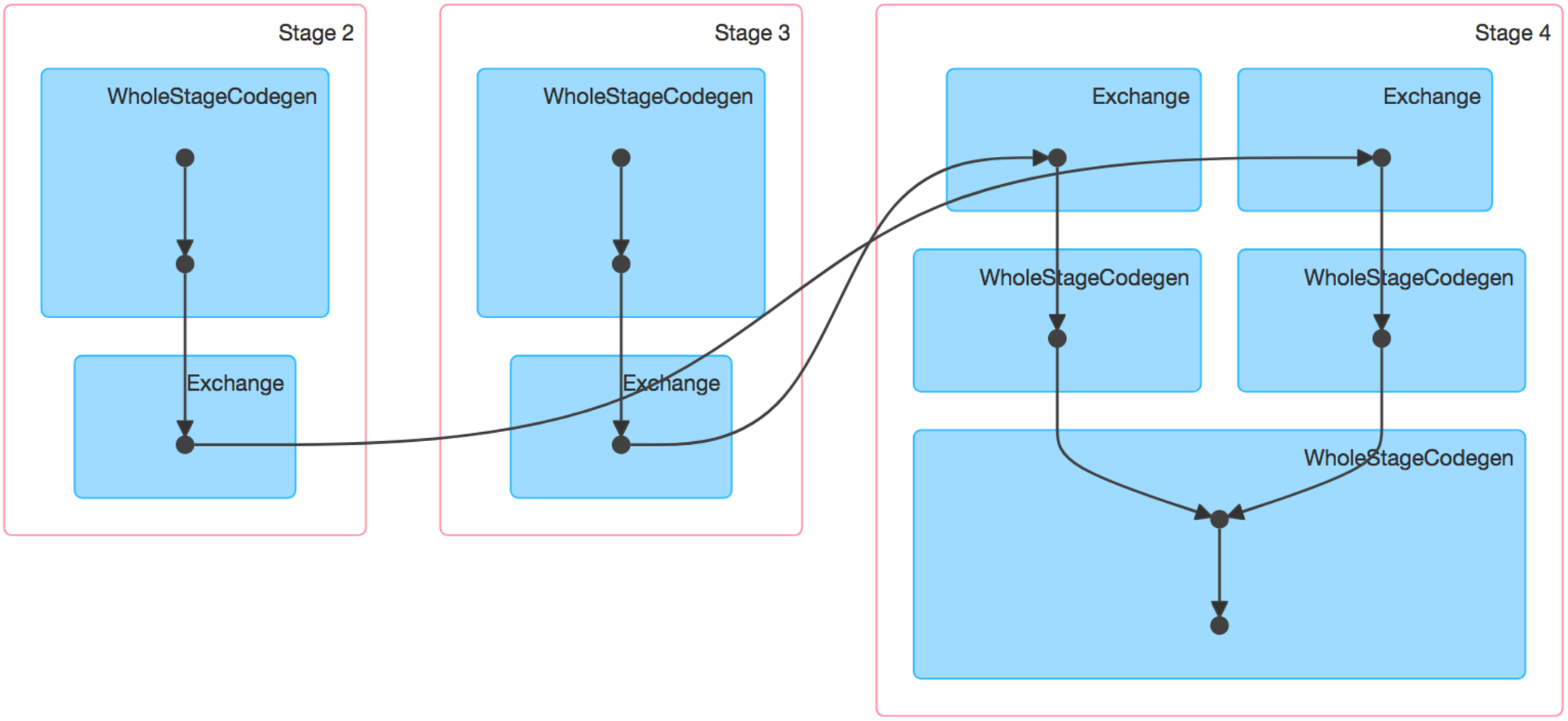
STAGE 3

TASK 1

TASK 2

▸ Event Timeline

▾ DAG Visualization



Active Stages (1)

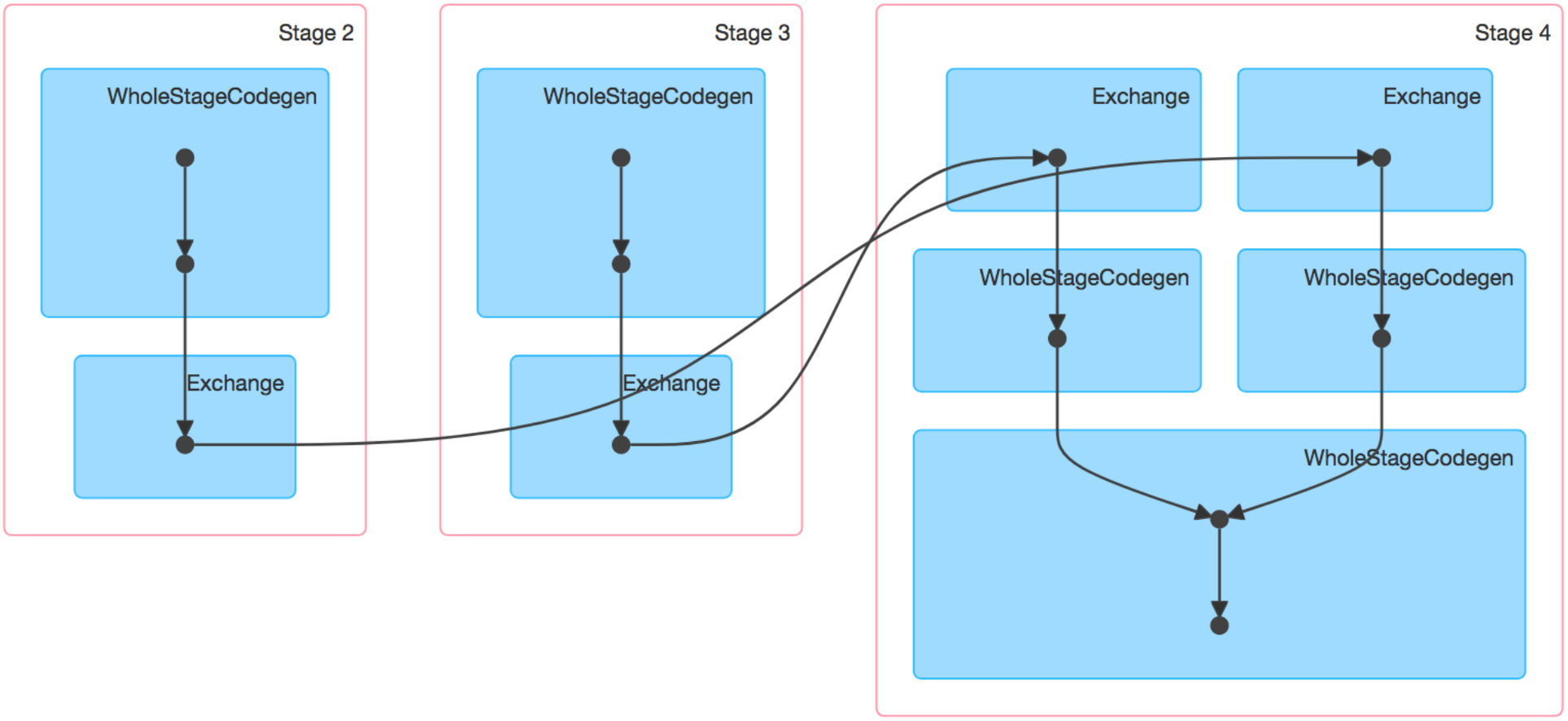
Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total
4	parquet at Demo.scala:218 +details (kill)	2018/06/06 18:23:31	32 s	0/200

Completed Stages (2)

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total
3	parquet at Demo.scala:218 +details	2018/06/06 18:18:52	4.6 min	12021/12021
2	parquet at Demo.scala:218 +details	2018/06/06 18:18:47	11 s	167/167

▸ Event Timeline

▾ DAG Visualization



Active Stages (1)

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total
4	parquet at Demo.scala:218 +details (kill)	2018/06/06 18:23:31	32 s	0/200

Completed Stages (2)

Stage Id ▾	Description	Submitted	Duration	Tasks: Succeeded/Total
3	parquet at Demo.scala:218 +details	2018/06/06 18:18:52	4.6 min	12021/12021
2	parquet at Demo.scala:218 +details	2018/06/06 18:18:47	11 s	167/167

Page: 1 2 >

2 Pages. Jump to 1 . Show 100 items

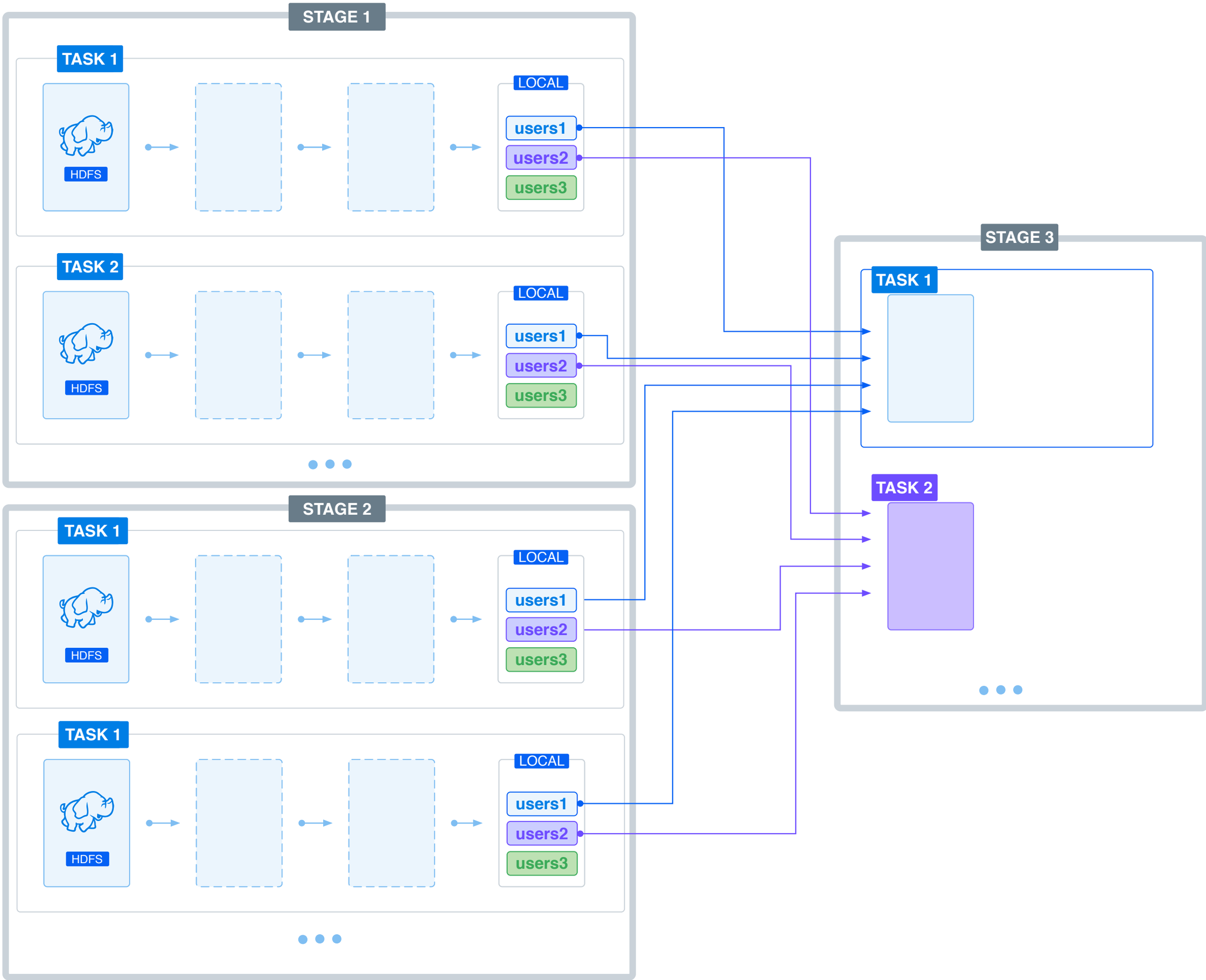
Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host		Launch Time	Duration	GC Time	Shuffle Read Size / Records
0	12190	0	RUNNING	PROCESS_LOCAL	196 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.4 GB / 112794344
1	12191	0	RUNNING	PROCESS_LOCAL	12 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	3 s	9.4 GB / 112729327
2	12192	0	RUNNING	PROCESS_LOCAL	29 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	5 s	9.4 GB / 112972119
3	12193	0	RUNNING	PROCESS_LOCAL	142 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.4 GB / 112641898
4	12194	0	RUNNING	PROCESS_LOCAL	87 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.4 GB / 112893337
5	12195	0	RUNNING	PROCESS_LOCAL	109 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	6.6 GB / 79318780
6	12196	0	RUNNING	PROCESS_LOCAL	110 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.4 GB / 112749028
7	12197	0	RUNNING	PROCESS_LOCAL	23 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.1 GB / 109051879
8	12198	0	RUNNING	PROCESS_LOCAL	150 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	9.4 GB / 112863394
9	12199	0	RUNNING	PROCESS_LOCAL	224 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	6.7 GB / 80596525
10	12200	0	RUNNING	PROCESS_LOCAL	222 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	6 s	8.6 GB / 103459475
11	12201	0	RUNNING	PROCESS_LOCAL	9 / [redacted]	stdout stderr	2018/06/06 18:23:31	2.2 min	4 s	8.5 GB / 102482162

Case #3

10TB of events

uniform distribution

1GB of users


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3
STAGE 3
TASK 1
TASK 2

```
.config(
 “spark.sql.shuffle.partitions",
 “200”
)
```

STAGE 1

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 2

HDFS

LOCAL

users1

users2

users3

STAGE 2

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 1

HDFS

LOCAL

users1

users2

users3

STAGE 3

TASK 1

TASK 2

```
.config(
 “spark.sql.shuffle.partitions",
 “200”
)
```

STAGE 1

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 2

HDFS

LOCAL

users1

users2

users3

STAGE 2

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 1

HDFS

LOCAL

users1

users2

users3

STAGE 3

TASK 1

TASK 2

10TB/200 = 50GB/TASK

```
.config(
 “spark.sql.shuffle.partitions",
 “200”
)
```

STAGE 1

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 2

HDFS

LOCAL

users1

users2

users3

STAGE 2

TASK 1

HDFS

LOCAL

users1

users2

users3

TASK 1

HDFS

LOCAL

users1

users2

users3

STAGE 3

TASK 1

50G

TASK 2

50G

10TB/200 = 50GB/TASK

Problems with shuffle

- Spill to disk
- Timeouts
- GC overhead limit exceeded
- OOM
- ExecutorLostFailure

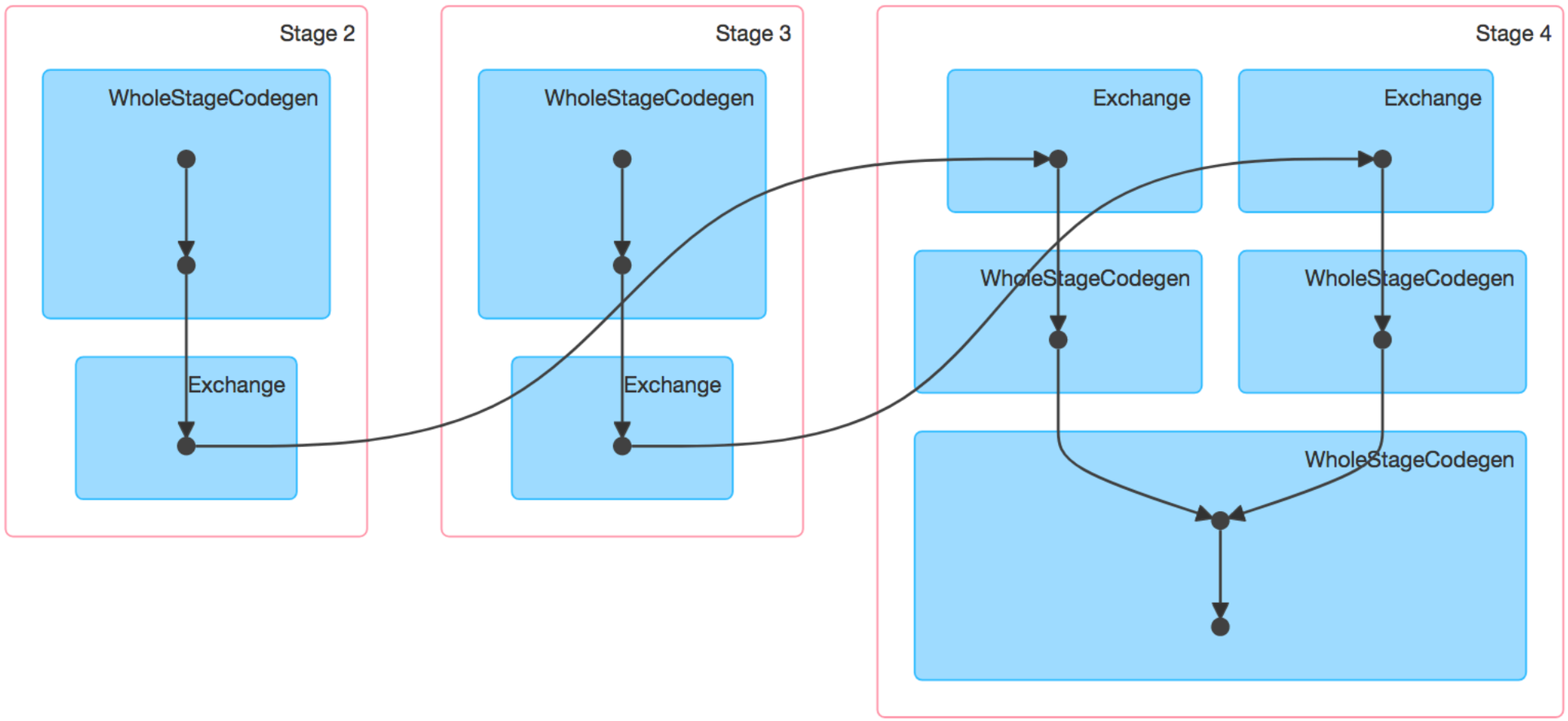
What to do?

- Understand your data!
- Control the level of parallelism
- Choose number of partitions according to your data size

```
.config("spark.sql.shuffle.partitions", “2000")
```

```
.repartition(2000)
```

Case #3 - result



Completed Stages (3)

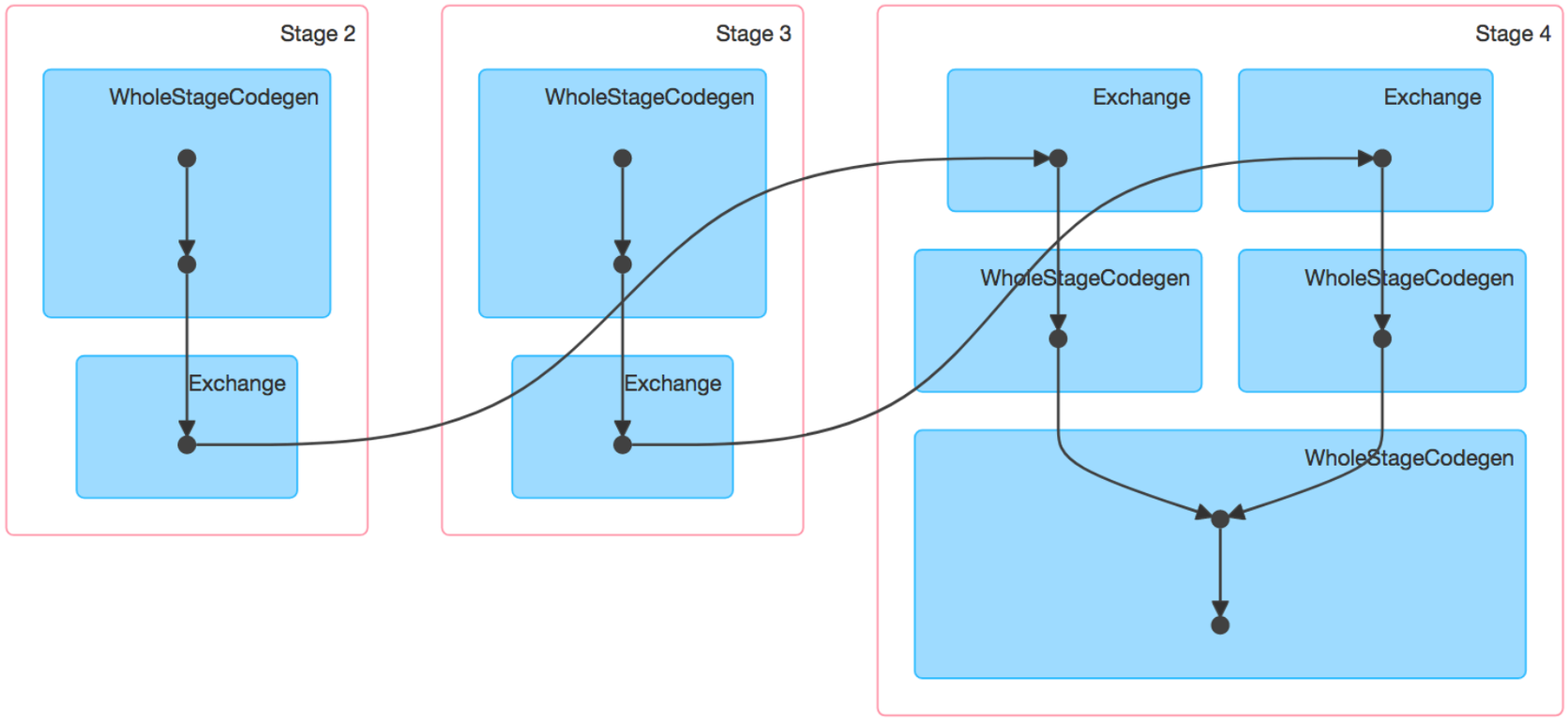
Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total
4	parquet at Demo.scala:218	+details	2018/06/09 17:23:00	8.9 min	2000/2000
3	parquet at Demo.scala:218	+details	2018/06/09 17:18:43	16 s	250/250
2	parquet at Demo.scala:218	+details	2018/06/09 17:18:42	4.2 min	12021/12021

Case #3 - result

Duration	GC Time	Shuffle Read Size / Records
1.4 min	4 s	962.8 MB / 11239611
1.4 min	3 s	963.9 MB / 11251397
1.5 min	3 s	973.0 MB / 11358655
1.4 min	2 s	968.1 MB / 11300823
1.4 min	3 s	970.0 MB / 11323585
1.4 min	3 s	976.9 MB / 11403719
1.2 min	2 s	963.2 MB / 11243664
1.5 min	3 s	972.6 MB / 11353934
1.4 min	3 s	966.5 MB / 11282067
1.4 min	3 s	961.6 MB / 11225279

Case #4 - Skewed join

Case #4



Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input
4	parquet at Demo.scala:246 +details (kill)	2018/06/09 14:43:18	26 s	670/2000	

Completed Stages (2)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input
3	parquet at Demo.scala:246 +details	2018/06/09 14:42:41	12 s	250/250	15.2 GB
2	parquet at Demo.scala:246 +details	2018/06/09 14:42:41	36 s	875/875	77.5 GB

Photo credit: hiveminer.com

Case #4

Duration	GC Time	Shuffle Read Size / Records ▾
19 s	41 ms	1268.4 MB / 8325934
6 s	0.3 s	124.6 MB / 835648
6 s	0.2 s	124.6 MB / 835362
9 s	0.3 s	124.6 MB / 835208
5 s	0.3 s	124.6 MB / 835142
7 s	0.2 s	124.5 MB / 834944

Photo credit: hiveminer.com

Skewed join

Duration	GC Time	Shuffle Read Size / Records ▾
19 s	41 ms	1268.4 MB / 8325934
6 s	0.3 s	124.6 MB / 835648
6 s	0.2 s	124.6 MB / 835362
9 s	0.3 s	124.6 MB / 835208
5 s	0.3 s	124.6 MB / 835142
7 s	0.2 s	124.5 MB / 834944

Photo credit: hiveminer.com

Skewed join

Duration	GC Time	Shuffle Read Size / Records ▾
32 s	0.3 s	2.2 GB / 14821755
6 s	0.3 s	124.6 MB / 835648
6 s	0.2 s	124.6 MB / 835362
9 s	0.3 s	124.6 MB / 835208
5 s	0.3 s	124.6 MB / 835142
7 s	0.2 s	124.5 MB / 834944

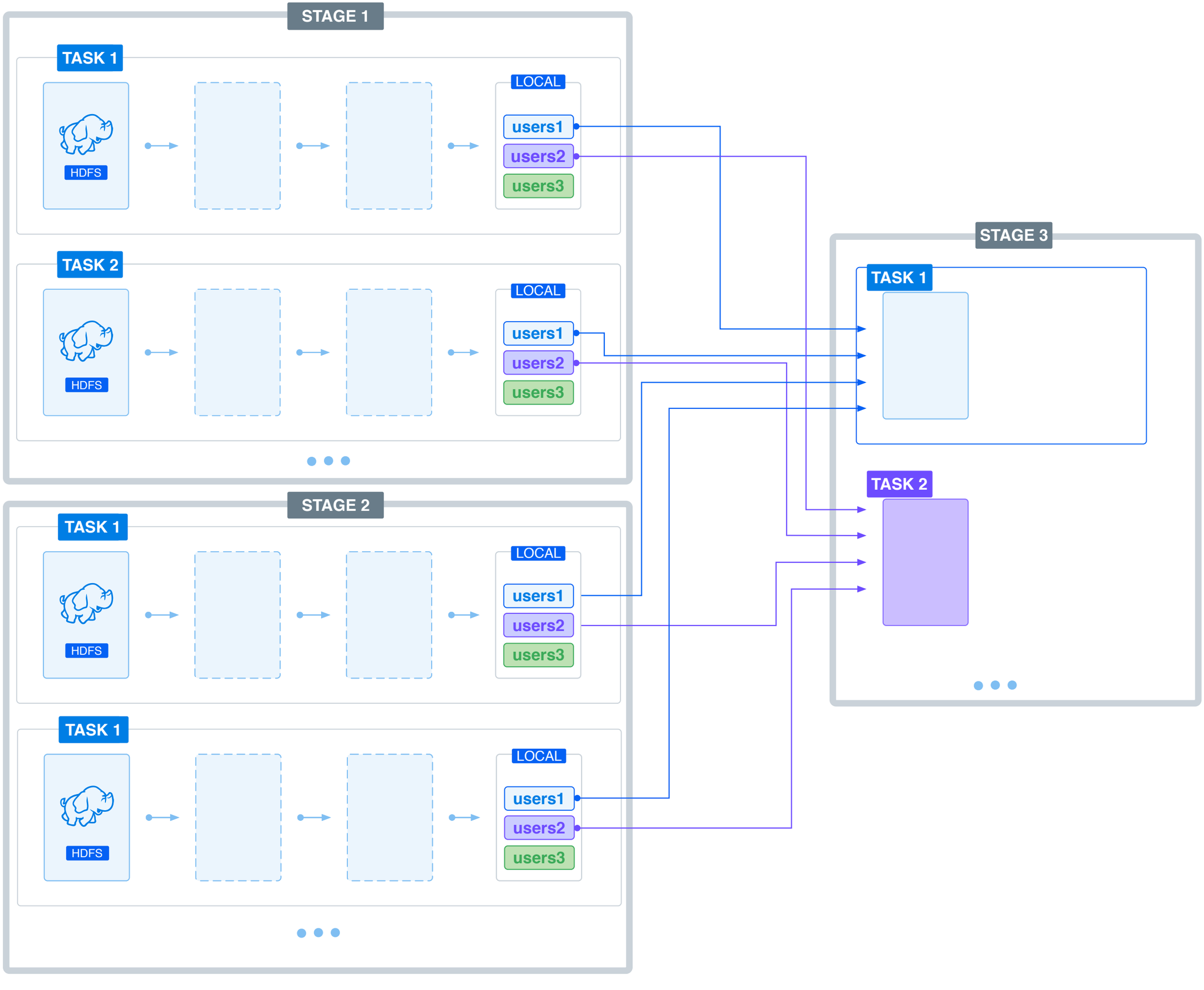
Photo credit: hiveminer.com

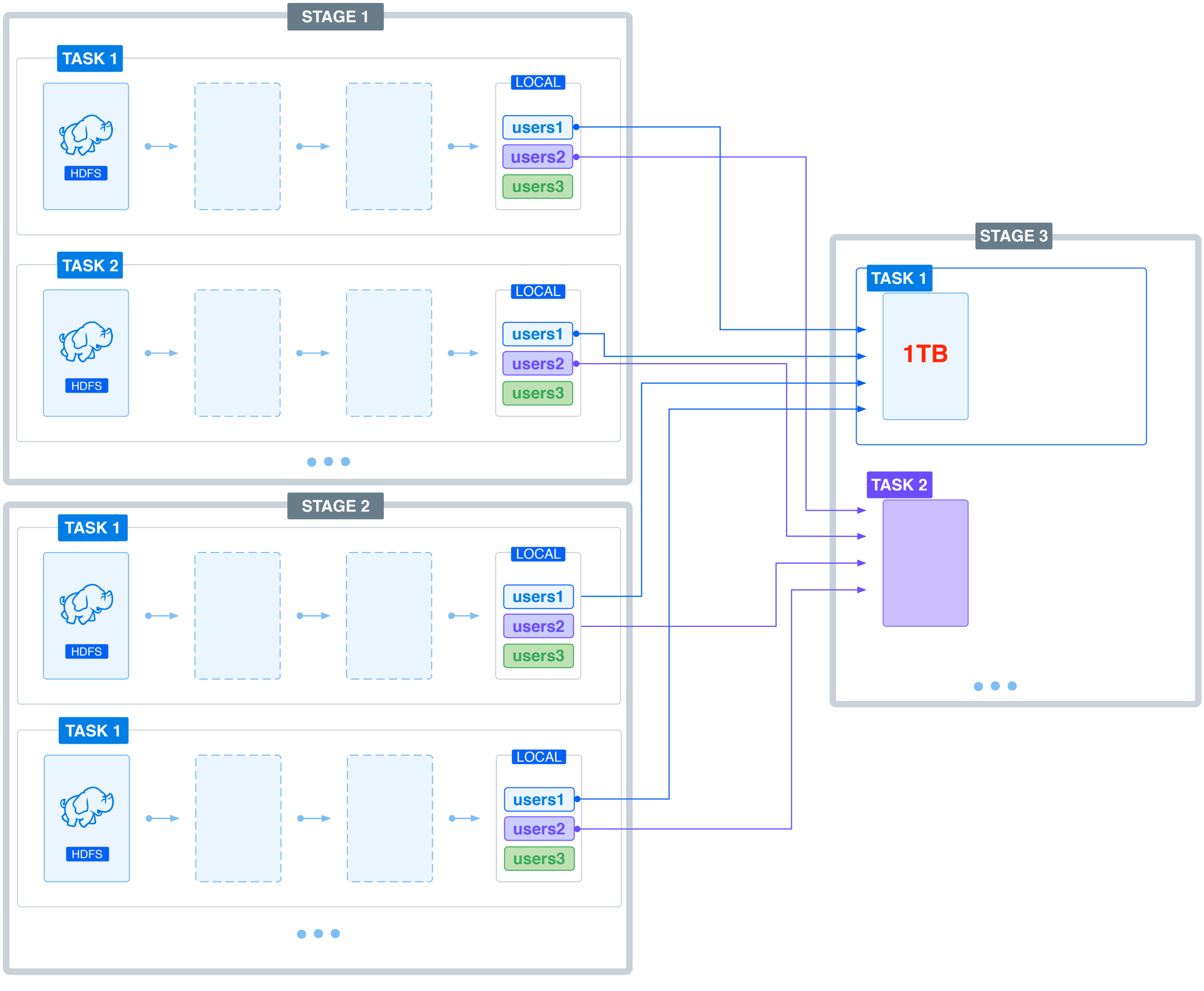
Case #4

10TB of events

One user with 1 TB of events

others are uniformly distributed


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3
STAGE 3
TASK 1
TASK 2


STAGE 1
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 2
HDFS
LOCAL
users1
users2
users3
STAGE 2
TASK 1
HDFS
LOCAL
users1
users2
users3
TASK 1
HDFS
LOCAL
users1
users2
users3
STAGE 3
TASK 1
1TB
TASK 2

Skewed join

- Bad data?
- Wrong logic?
- Just ok?

www.tantusdata.com

Photo credit: hiveminer.com

Skewed Join

eventId	userId	…
af8	1	
bf9	1	
ff1	1	
881	1	
91f	2	
cc6	1	
b22	1	
ee4	1	

userId	…
1	
2	
3	
…	…

Skewed Join

eventId	userId	…
af8	1	
bf9	1	
ff1	1	
881	1	
91f	2	
cc6	1	
b22	1	
ee4	1	

userId	…
1	
2	
3	
…	…

Skewed Join

eventId	userId	…	salt
af8	1		1
bf9	1		2
ff1	1		1
881	1		3
91f	2		2
cc6	1		3
b22	1		3
ee4	1		1

userId	…
1	
2	
3	
…	…

Skewed Join

eventId	userId	…	salt
af8	1		1
bf9	1		2
ff1	1		1
881	1		3
91f	2		2
cc6	1		3
b22	1		3
ee4	1		1

userId	…	salt
1		**1**
1		**2**
1		**3**
2		**1**
2		**2**
2		**3**
3		**1**
3		**2**
3		**3**
…	…	

Skewed Join

eventId	userId	…	salt
af8	1		1
bf9	1		2
ff1	1		1
881	1		3
91f	2		2
cc6	1		3
b22	1		3
ee4	1		1

userId	…	salt
1		1
1		2
1		3
2		1
2		2
2		3
3		1
3		2
3		3
…	…	

Skewed Join

eventId	userId	…	salt
af8	1		1
bf9	1		2
ff1	1		1
881	1		3
91f	2		2
cc6	1		3
b22	1		3
ee4	1		1

userId	…	salt
1		1
1		2
1		3
2		1
2		2
2		3
3		1
3		2
3		3
…	…	

Skewed Join

eventId	userId	…	salt
af8	1		1
bf9	1		2
ff1	1		1
881	1		3
91f	2		2
cc6	1		3
b22	1		3
ee4	1		1

userId	…	salt
1		1
1		2
1		3
2		1
2		2
2		3
3		1
3		2
3		3
...	...	

Skewed Join

```
val eventsSalted = events
  .withColumn("salt", toInt(rand() * 100))
```

Skewed Join

```scala
val eventsSalted = events
  .withColumn("salt", toInt(rand() * 100))
```

Skewed Join

```
val eventsSalted = events
  .withColumn("salt", toInt(rand() * 100))
```

Skewed Join

```
val allSalts = spark.sparkContext
  .parallelize(1 to 100).toList)
  .toDF("salt")

val usersSalted = users.join(allSalts)
```

Skewed Join

```scala
val allSalts = spark.sparkContext
  .parallelize((1 to 100).toList)
  .toDF("salt")

val usersSalted = users.join(allSalts)
```

Skewed Join

```scala
val allSalts = spark.sparkContext
  .parallelize((1 to 100).toList)
  .toDF("salt")

val usersSalted = users.join(allSalts)
```

Skewed Join

```
eventsSalted
.join(
  usersSalted,
    usersSalted("id") === eventsSalted("userId")
    && usersSalted("salt") === eventsSalted("salt")
)
```

Skewed Join

```
eventsSalted
.join(
  usersSalted,
    usersSalted("id") === eventsSalted("userId")
    && usersSalted(“salt") === eventsSalted("salt")
)
```

Skewed Join

```
eventsSalted
.join(
  usersSalted,
    usersSalted("id") === eventsSalted("userId")
    && usersSalted(“salt") === eventsSalted("salt")
)
```

Skewed Join

```
eventsSalted
.join(
  usersSalted,
    usersSalted("id") === eventsSalted("userId")
 &&
usersSalted("salt")===eventsSalted("salt")
)
```

Skewed Join

www.tantusdata.com

Use cases - recap

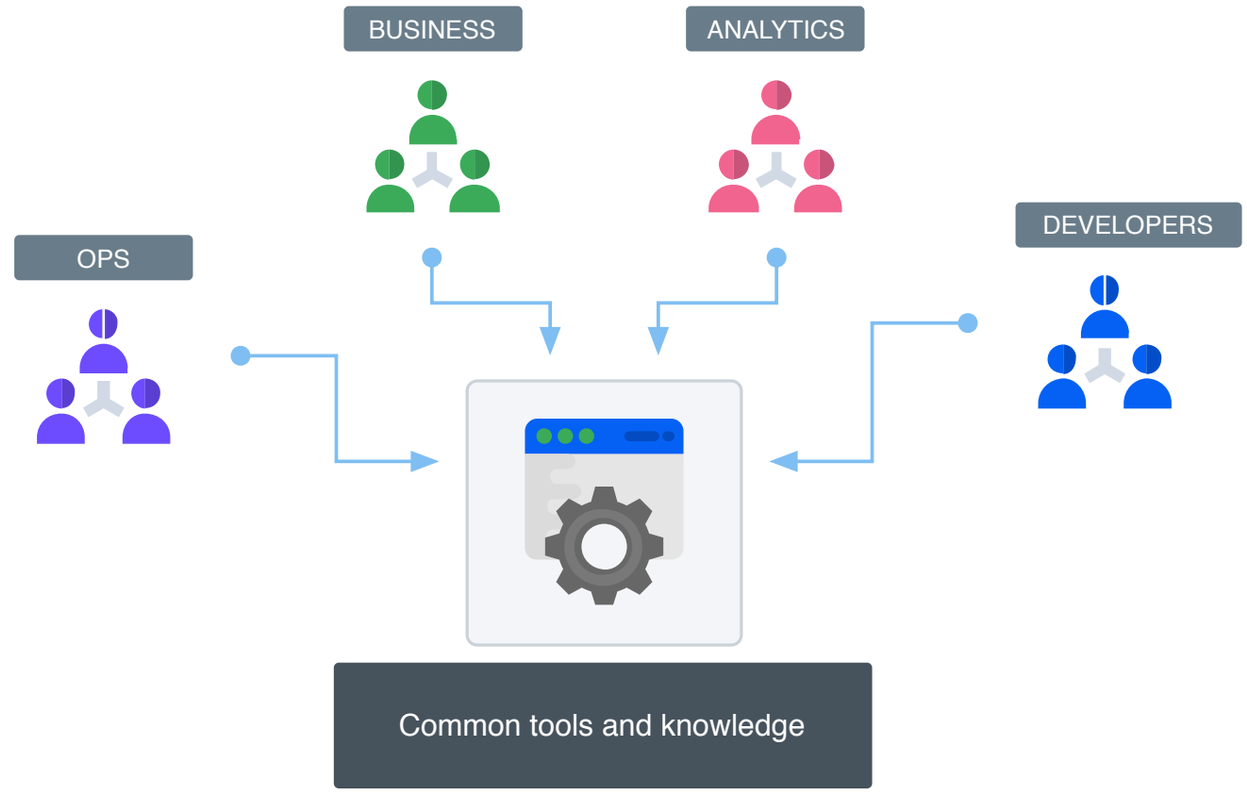
- Know your data!
- Understand the transformations you are doing
- Keep your tasks busy

Photo credit: shamakern.com

Challenges ahead

- Broadcast
- Cache
- Locality
- File formats
- Sizing executors
- …

Challenges ahead



Q&A

- marcin@tantusdata.com
- marcin.szymaniuk@gmail.com
- @mszymani

Thank you!

www.tantusdata.com