SPARK SQL 自适应执行引擎

Carson Wang (carson.wang@intel.com)

Agenda

- Challenges in Spark SQL* High Performance
- Adaptive Execution Architecture
- Benchmark Result

*Other names and brands may be claimed as the property of others.

Spark SQL* Tuning – Shuffle Partition Number

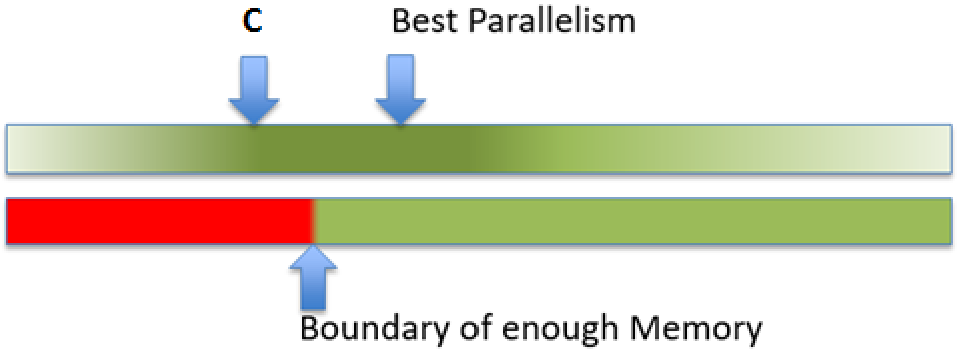
- Partition Num **P** = spark.sql.shuffle.partition (200 by default)
- Cluster Core Num **C** =
  Executor Num * Executor Core Num
- Each Reduce Stage runs the tasks in (P / C) rounds

*Other names and brands may be claimed as the property of others.

Shuffle Partition Problem 1

- Partition Num Too Small：Spill, OOM
- Partition Num Too Large：Scheduling overhead. Too much small output files
- In Practice: Increase partition size starting from C, 2C, … until performance begin to drop

Impractical for each query in production.



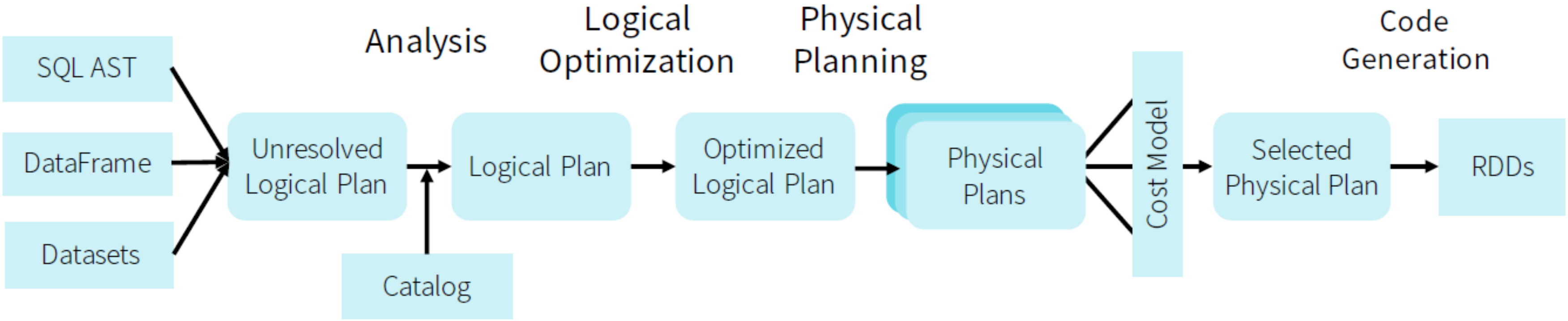
Shuffle Partition Problem 2

- The same Shuffle Partition number doesn’t fit for all Stages
- Shuffle data size usually decreases during the execution of the SQL query

Solution:

Auto Set the Shuffle Partition Number for Each Stage

Spark SQL* Execution Plan



- The execution plan is fixed after planning phase.

*Other names and brands may be claimed as the property of others.

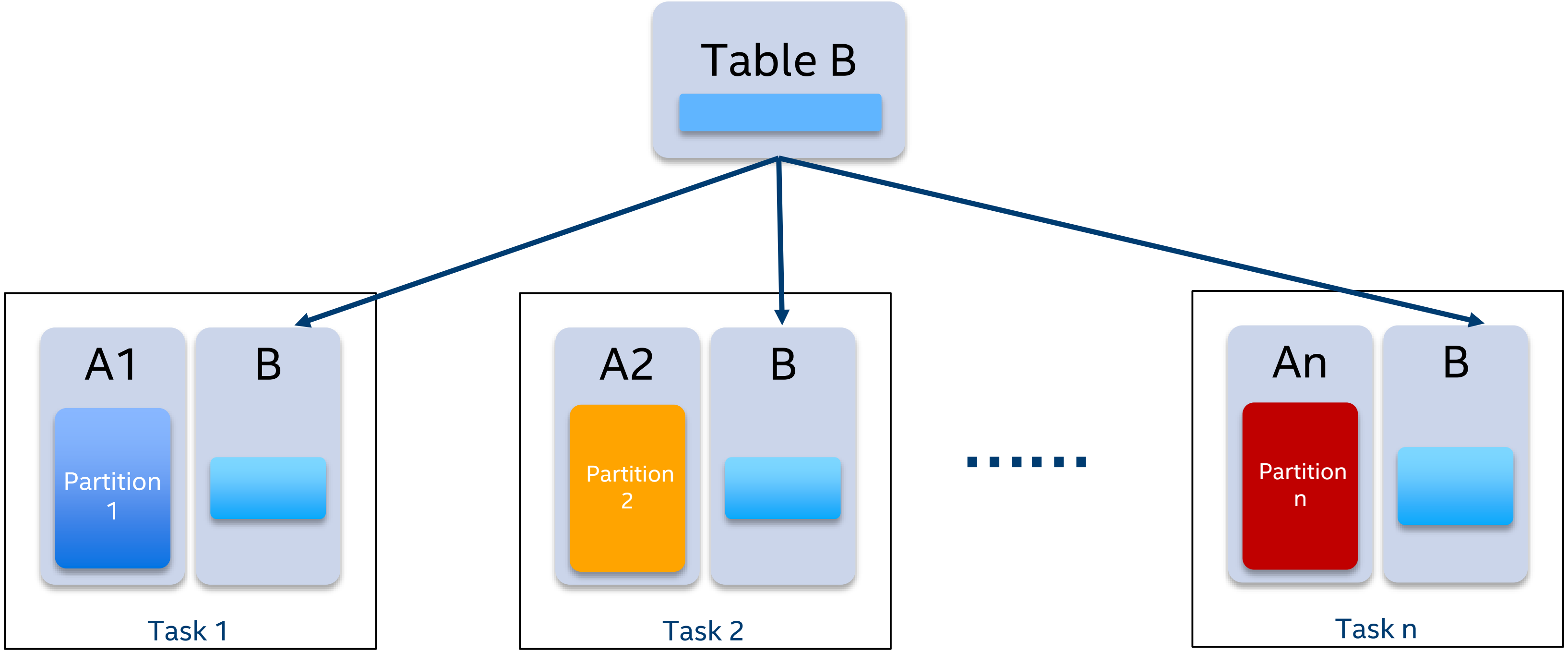
Image from: https://databricks.com/blog/2015/03/24/spark-sql-graduates-from-alpha-in-spark-1-3.html

Spark SQL* Joins

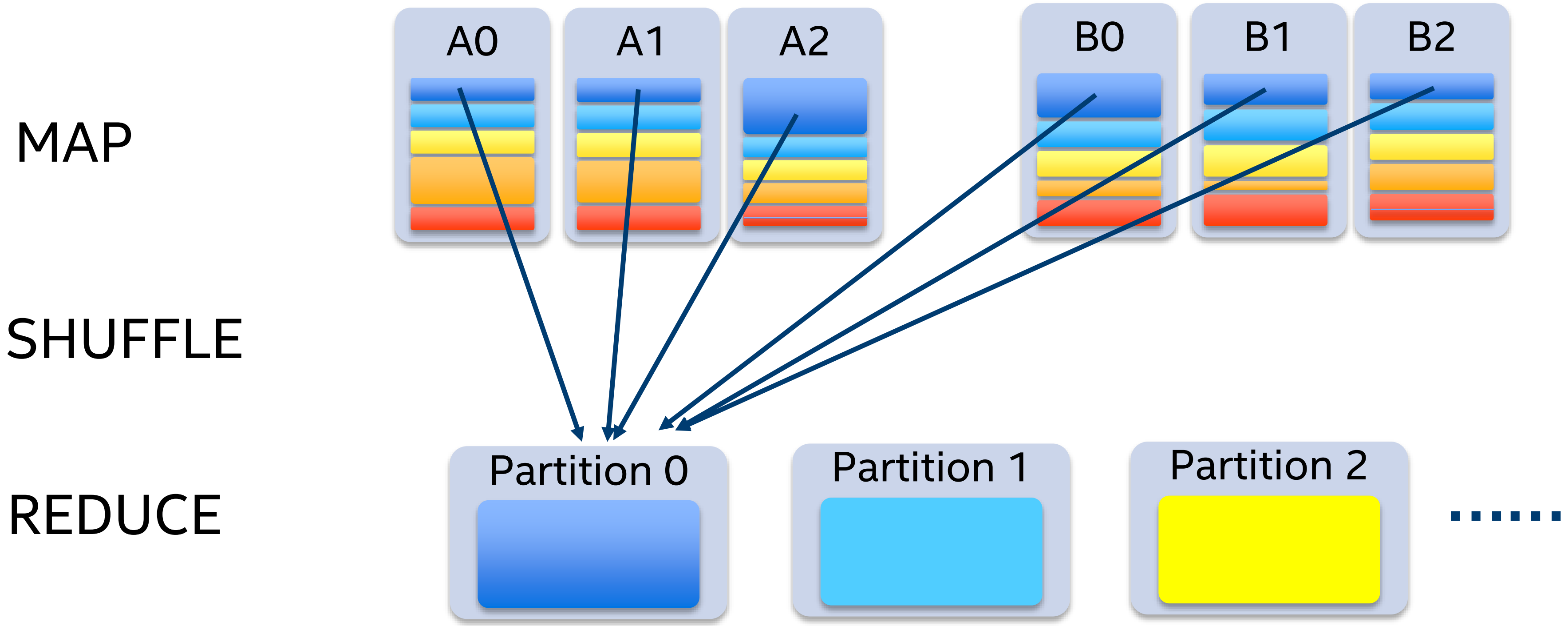
```
SELECT xxx

FROM A

JOIN B

ON A.Key1 = B.Key2
```

*Other names and brands may be claimed as the property of others.

Broadcast Hash Join



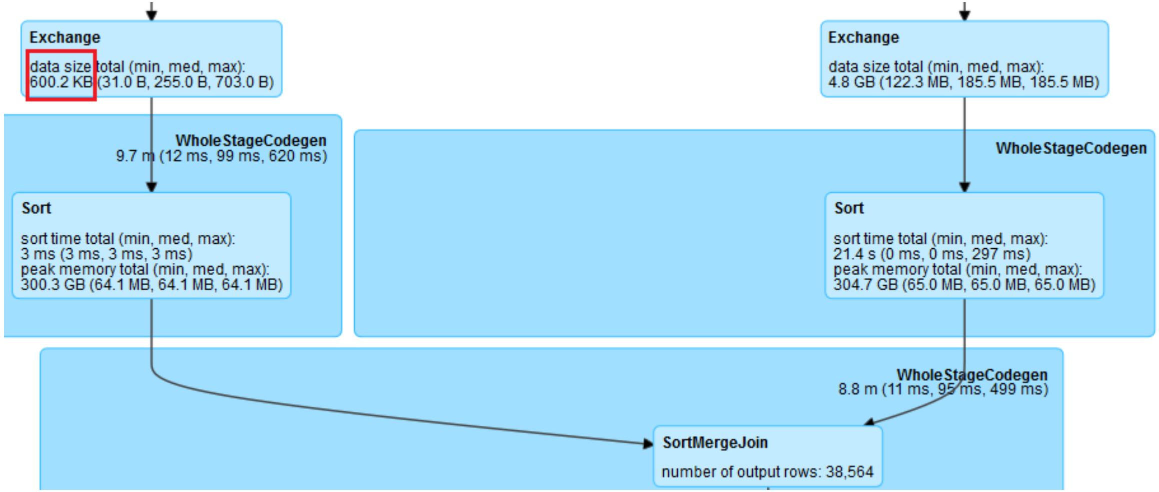
Shuffle Hash Join / Sort Merge Join



Spark SQL* Join Selection

- spark.sql.autoBroadcastJoinThreshold is 10 MB by default

- For complex queries, a Join may takes intermediate results as inputs. At planning phase, Spark SQL* doesn't know the exact size and plans it to SortMergeJoin.



Exchange
data size total (min, med, max):
600.2 KB (31.0 B, 255.0 B, 703.0 B)

WholeStageCodegen
9.7 m (12 ms, 99 ms, 620 ms)

Sort
sort time total (min, med, max):
3 ms (3 ms, 3 ms, 3 ms)
peak memory total (min, med, max):
300.3 GB (64.1 MB, 64.1 MB, 64.1 MB)

Exchange
data size total (min, med, max):
4.8 GB (122.3 MB, 185.5 MB, 185.5 MB)

WholeStageCodegen

Sort
sort time total (min, med, max):
21.4 s (0 ms, 0 ms, 297 ms)
peak memory total (min, med, max):
304.7 GB (65.0 MB, 65.0 MB, 65.0 MB)

WholeStageCodegen
8.8 m (11 ms, 95 ms, 499 ms)

SortMergeJoin
number of output rows: 38,564

*Other names and brands may be claimed as the property of others.

Solution:

Need a Way to Optimize The Execution Plan at Runtime

Data Skew in Join

- Data in some partitions are extremely larger than other partitions.
- Data skew is a common source of slowness for Shuffle Joins.

Handle Data Skew in Join Manually

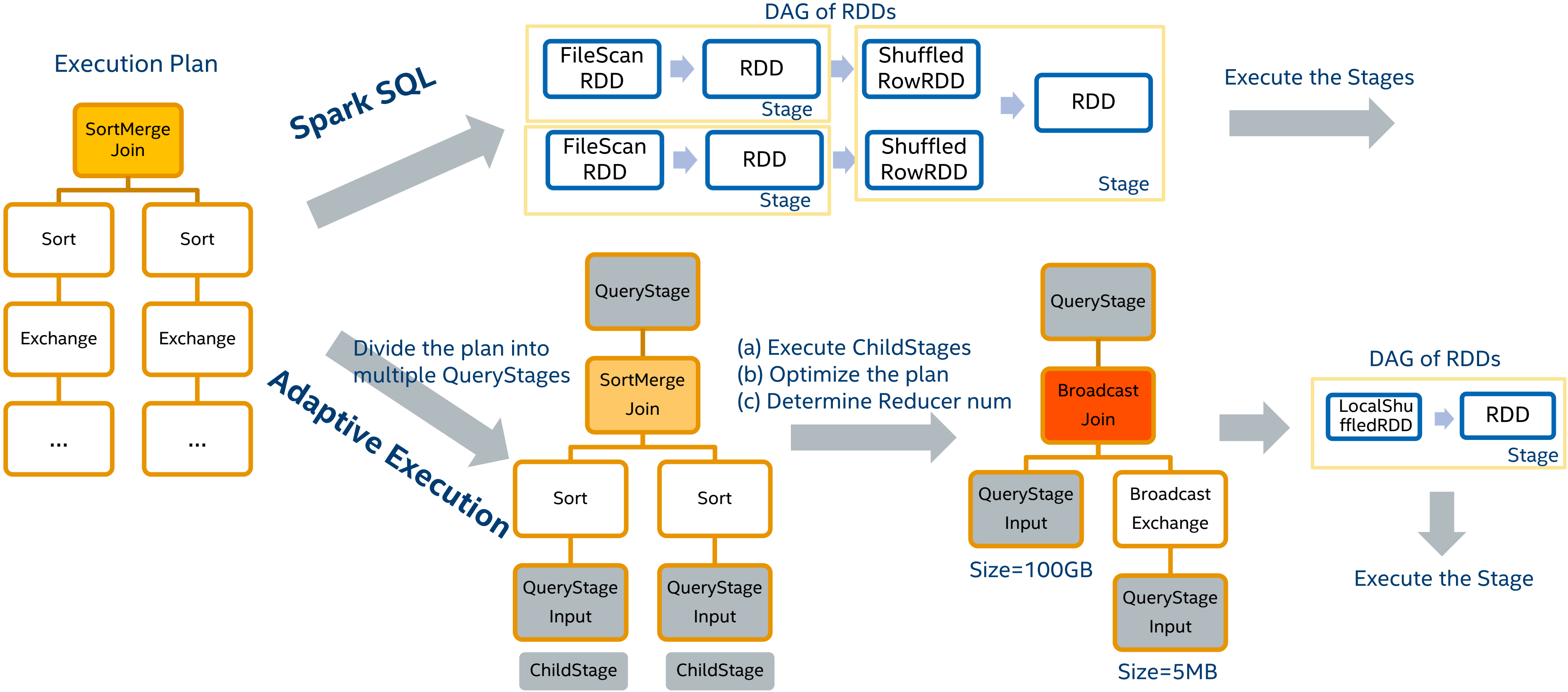
- Increase shuffle partition size
- Increase BroadcastJoin threashold to change Shuffle Join to Broadcast Join
- Add prefix to skewed keys
- ……

These involve many manual efforts. We need a way to handle data skew in join at runtime automatically!

A New Adaptive Execution Engine in Spark SQL*

*Other names and brands may be claimed as the property of others.

Adaptive Execution Architecture



Shuffle Join => Broadcast Join

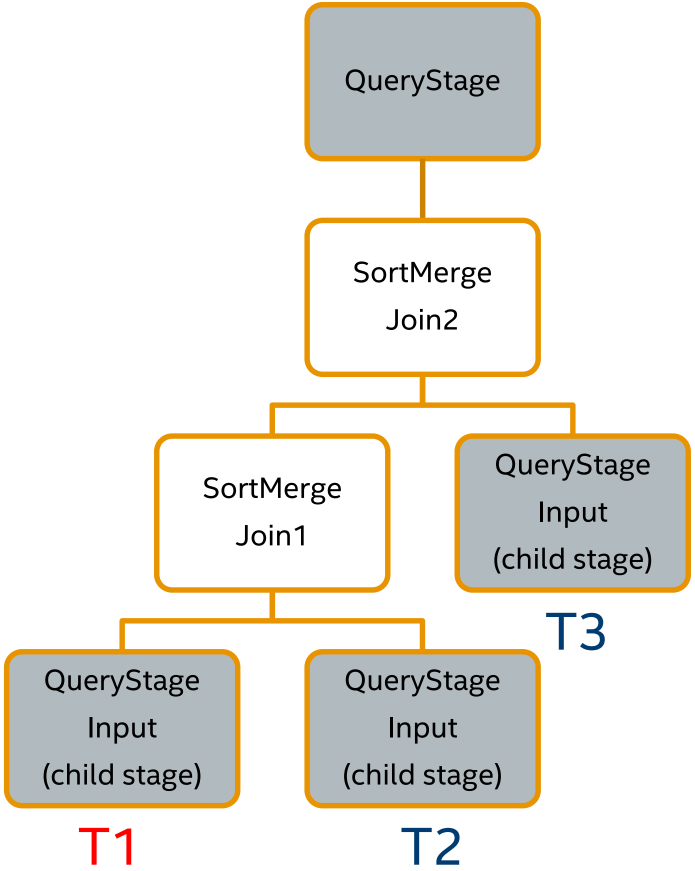
- The Challenge:
  - Change Shuffle Join to Broadcast Join may add additional Shuffles

- We only change the Join if below requirements are met:
  - One input table size is less than the broadcast threshold.
  - The change doesn’t introduce additional Shuffles

Example 1

- T1 < broadcast threshold
- T2 and T3 > broadcast threshold

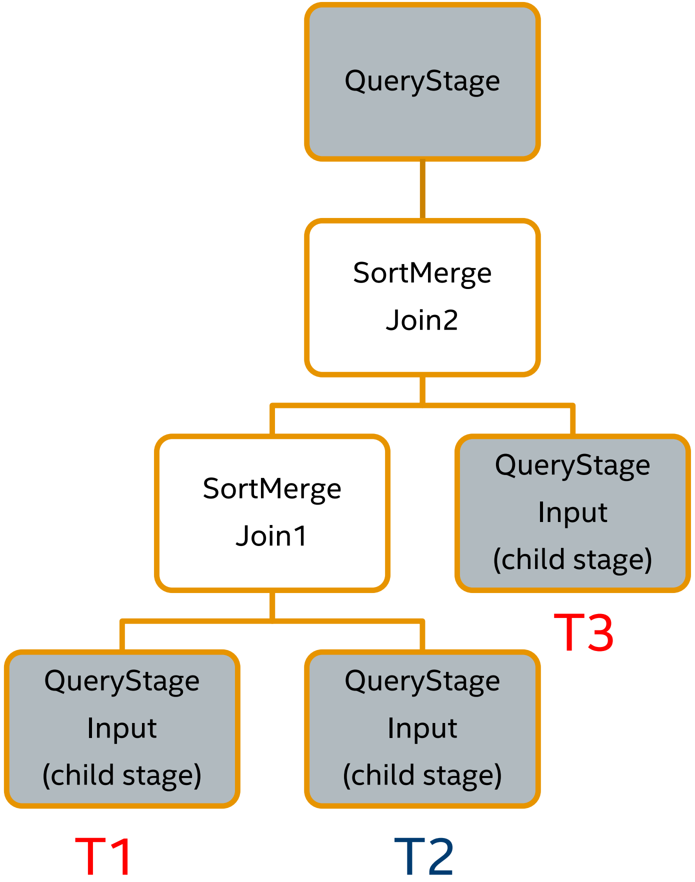
- In this case, both Join1 and Join2 are not changed to broadcast join


QueryStage
SortMerge Join2
SortMerge Join1
QueryStage Input (child stage)
T3
QueryStage Input (child stage)
QueryStage Input (child stage)
T1
T2

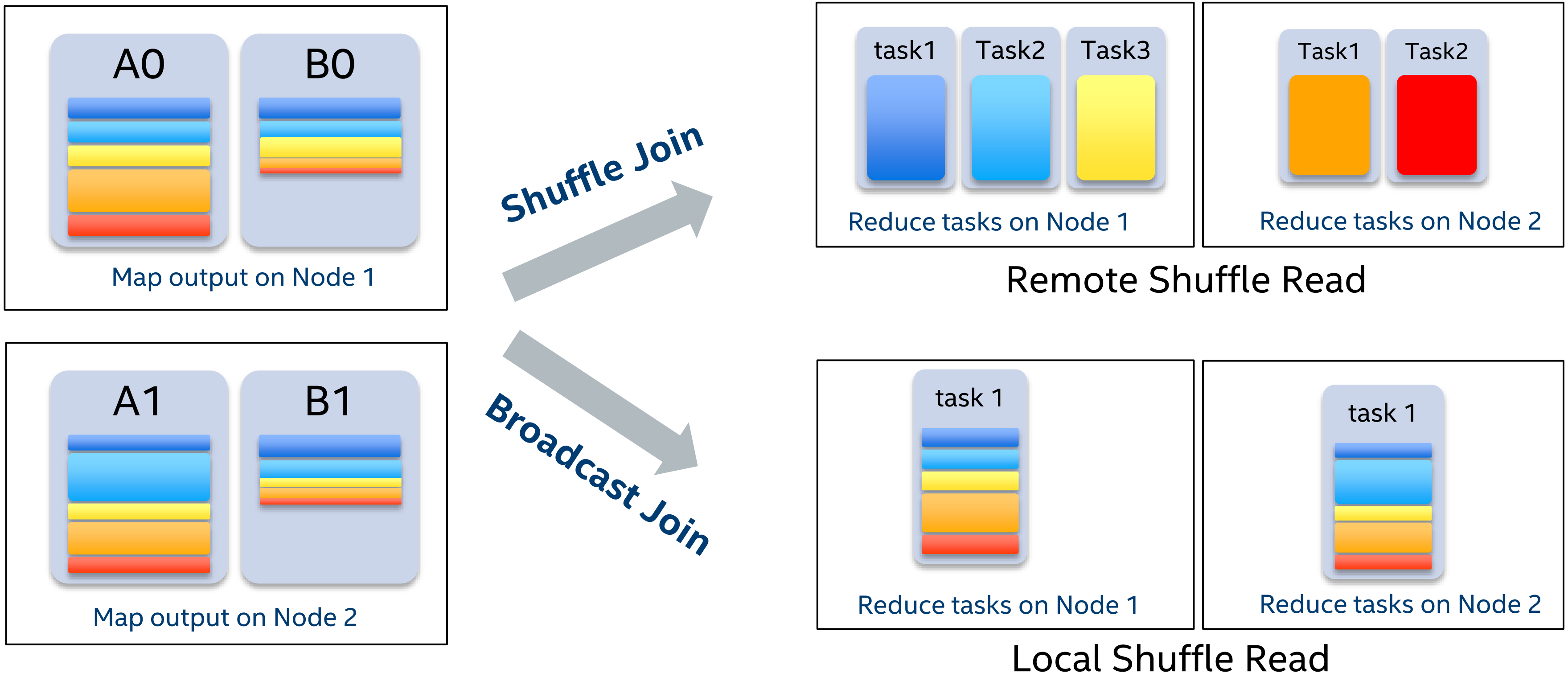
Example 2

- T1 and T3 < broadcast threshold
- T2 > broadcast threshold

- In this case, both Join1 and Join2 are changed to broadcast join


QueryStage
SortMerge Join2
SortMerge Join1
QueryStage Input (child stage)
T3
QueryStage Input (child stage)
QueryStage Input (child stage)
T1
T2

Remote Shuffle Read => Local Shuffle Read

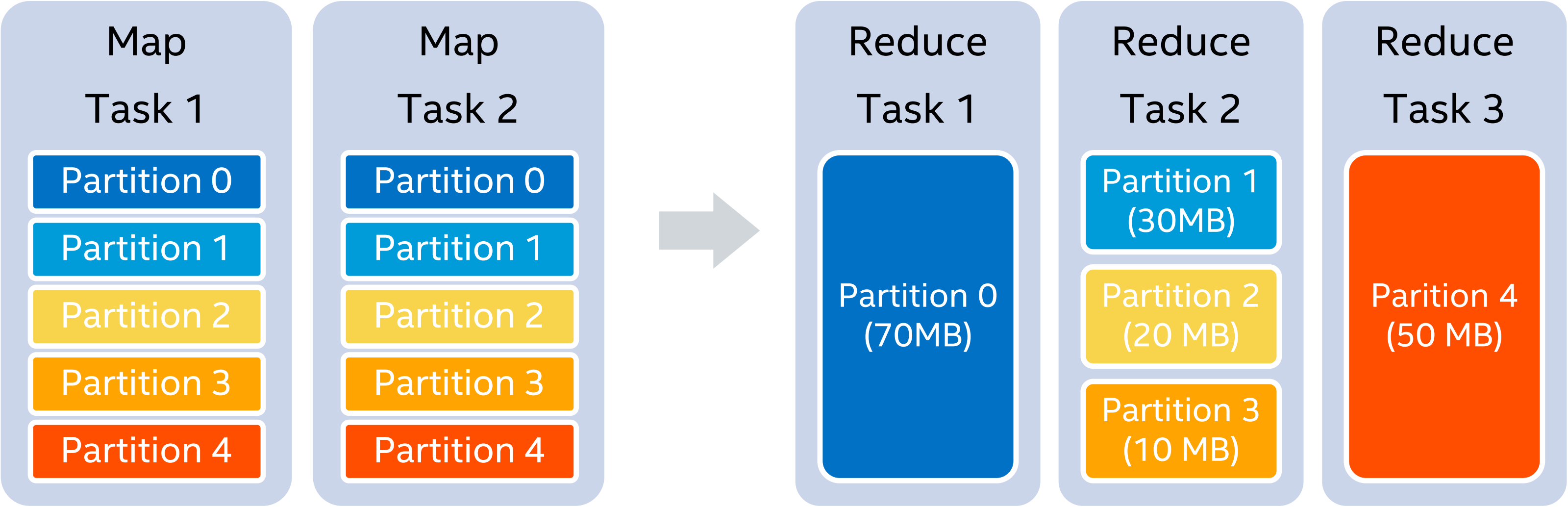


Shuffle Read Interface Change

- We pass a mapId to ShuffleManager’s getReader interface.
- This enables the shuffle reader reading all blocks from a single map output.

Auto Setting the Number of Reducers

- 5 initial reducer partitions with size [70 MB, 30 MB, 20 MB, 10 MB, 50 MB]
- Set target size per reducer = 64 MB. At runtime, we use 3 actual reducers.



Handling Skewed Shuffle Join Input Data

- Use broadcast join to handle skewed partitions and use shuffle join for other.

Example: The size of initial reducer partitions of two input tables of a join operator
Table1: [2000 MB, 50 MB, 60 MB, 70 MB, 100 MB]
Table2: [10 MB, 20 MB, 50 MB, 40 MB,   50 MB].

We can broadcast the first partition of table 2. So, we will not shuffle rows of the first partition of table1 to a single reducer.

Benchmark Result

Cluster Setup

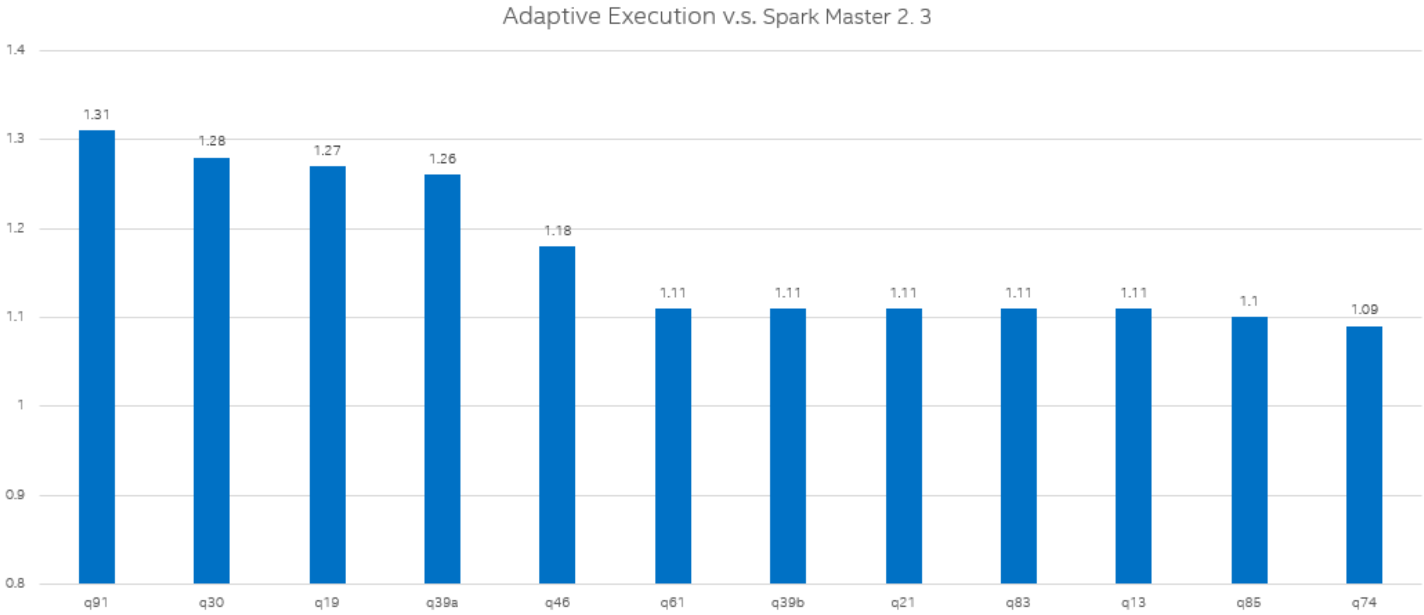
Hardware		BDW
Slave	Node	**ecs.d1.8xlarge x 10**
	CPU	Intel（R） Xeon（R) CPU E5-2682 v4 @ 2.50GHz (32 cores)
	Memory	128 GB
	Disk	1 (40 GB) + 16 × 5.4 TB HDD
	Network	10 Gigabit Ethernet
Master	CPU	Intel（R） Xeon（R) CPU E5-2680 v3 @ 2.50GHz (32 cores)
	Memory	128 GB
	Disk	1 (40 GB) + 1 (80 GB)
	Network	4 Gigabit Ethernet

Software	
OS	CentOS* Linux release 7.2.1511 (Core)
Kernel	3.10.0-514.6.2.el7.x86_64
Spark*	Spark* master (2.3) / Spark* master (2.3) with adaptive execution patch
Hadoop*/HDFS*	hadoop-2.7.2
JDK	1.8.0_121 (Oracle* Corporation)

*Other names and brands may be claimed as the property of others.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

TPC-DS* 10TB on 10 Node Cluster

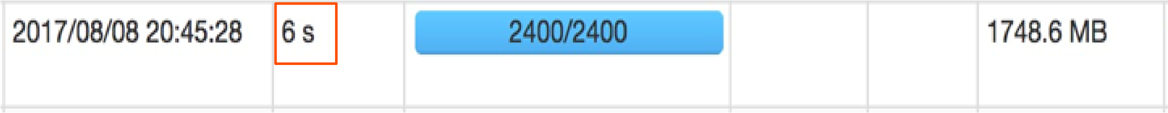


*Other names and brands may be claimed as the property of others.
For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

SortMergeJoin -> BroadcastJoin

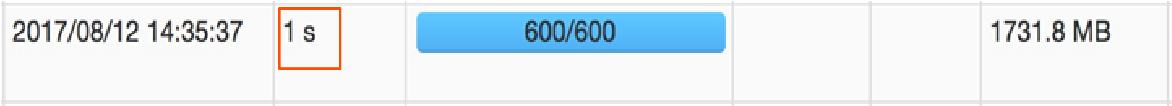
- Eliminate the data skew and straggler in SortMergeJoin
- Remote shuffle read -> local shuffle read.
- Random IO read -> Sequence IO read

SortMergeJoin:



2017/08/08 20:45:28	6 s	2400/2400			1748.6 MB

BroadcastJoin:



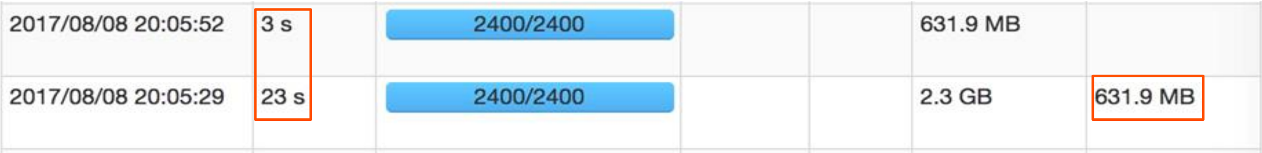
2017/08/12 14:35:37	1 s	600/600			1731.8 MB

*For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Auto Setting the Number of Reducers

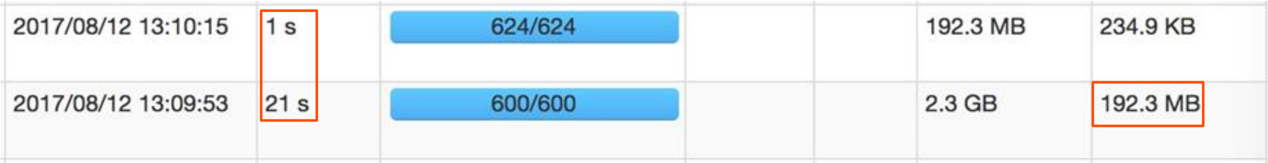
- Less scheduler overhead. Less disk IO requests.
- For aggregation, less data are written to disk because data are aggregatd in less partitions.

Partition Num 2400



2017/08/08 20:05:52	3 s	2400/2400			631.9 MB	
2017/08/08 20:05:29	23 s	2400/2400			2.3 GB	631.9 MB

Partition Num changed to 600 and 624 at runtime.



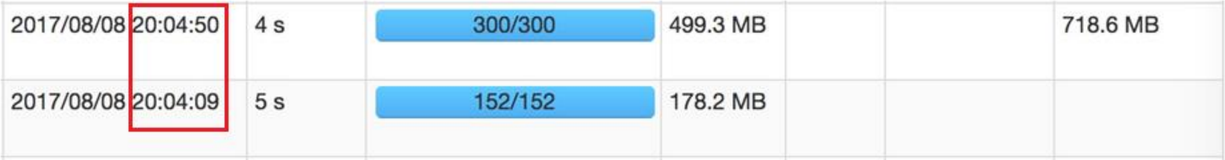
2017/08/12 13:10:15	1 s	624/624			192.3 MB	234.9 KB
2017/08/12 13:09:53	21 s	600/600			2.3 GB	192.3 MB

*For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

Scheduling Difference

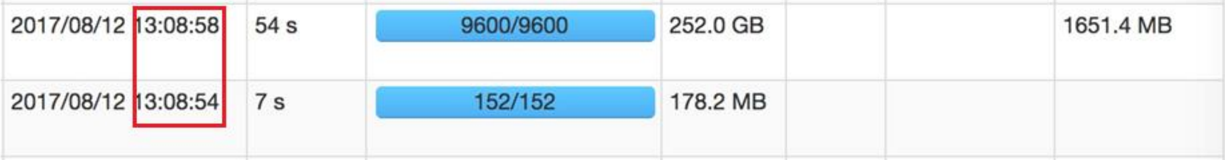
- Spark SQL* has to wait for the completation of all broadcasts before scheduling the execution stages.
- Adaptive Execution can start the stages earlier as long as its dependencies are completed.

Original Spark:



2017/08/08 20:04:50	4 s	300/300	499.3 MB			718.6 MB
2017/08/08 20:04:09	5 s	152/152	178.2 MB			

Adaptive Execution:



2017/08/12 13:08:58	54 s	9600/9600	252.0 GB			1651.4 MB
2017/08/12 13:08:54	7 s	152/152	178.2 MB			

*Other names and brands may be claimed as the property of others.

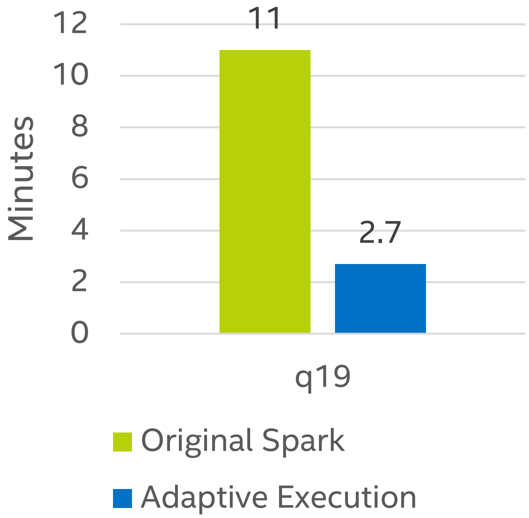
For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

TPC-BB* - 4x Improvement in q19

- TPCx-BB* q19 suffers from data skew issue when shuffle joining the tables. Computing the Shuffled RDD is also time consuming because of the complex UDF.
- It global sorts the data that requires sampling the RDD. This means the RDD is computed at least twice as it is not cached.

By using Adaptive Execution:

- 5 Shuffle Joins are changed to Broadcast Joins at runtime.
- 11 mins -> 2.7 mins (3TB data size, 4 worker nodes)



*Other names and brands may be claimed as the property of others.

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks

THANK YOU

Legal Disclaimer

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others