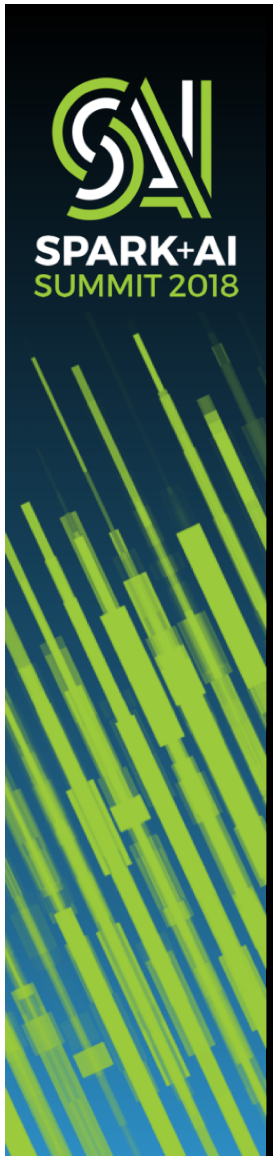


Spark SQL Adaptive Execution Unleashes The Power of Cluster in Large Scale

Carson Wang (Intel), Yuanjian Li (Baidu)

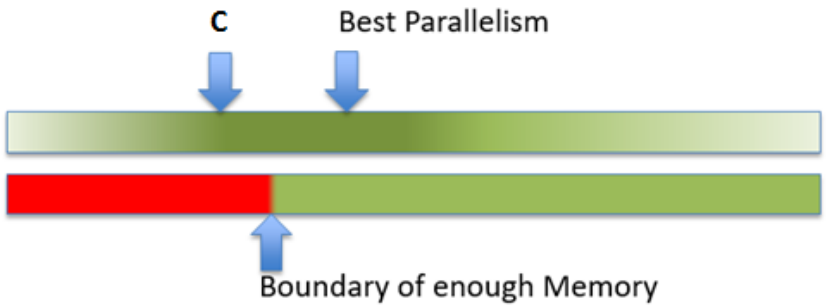
#Exp5SAIS

Agenda

- Challenges in Using Spark SQL
- Adaptive Execution Introduction
- Adaptive Execution in Baidu

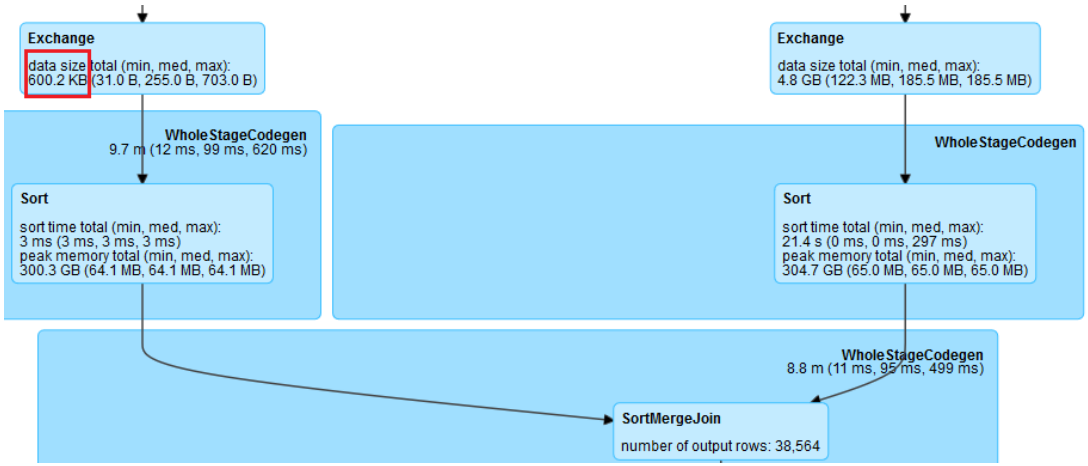
Tuning Shuffle Partition Number

- Too small：Spill, OOM
- Too large：Scheduling overhead. More IO requests. Too many small output files
- The same shuffle partition number doesn’t fit for all stages



Spark SQL Join Selection

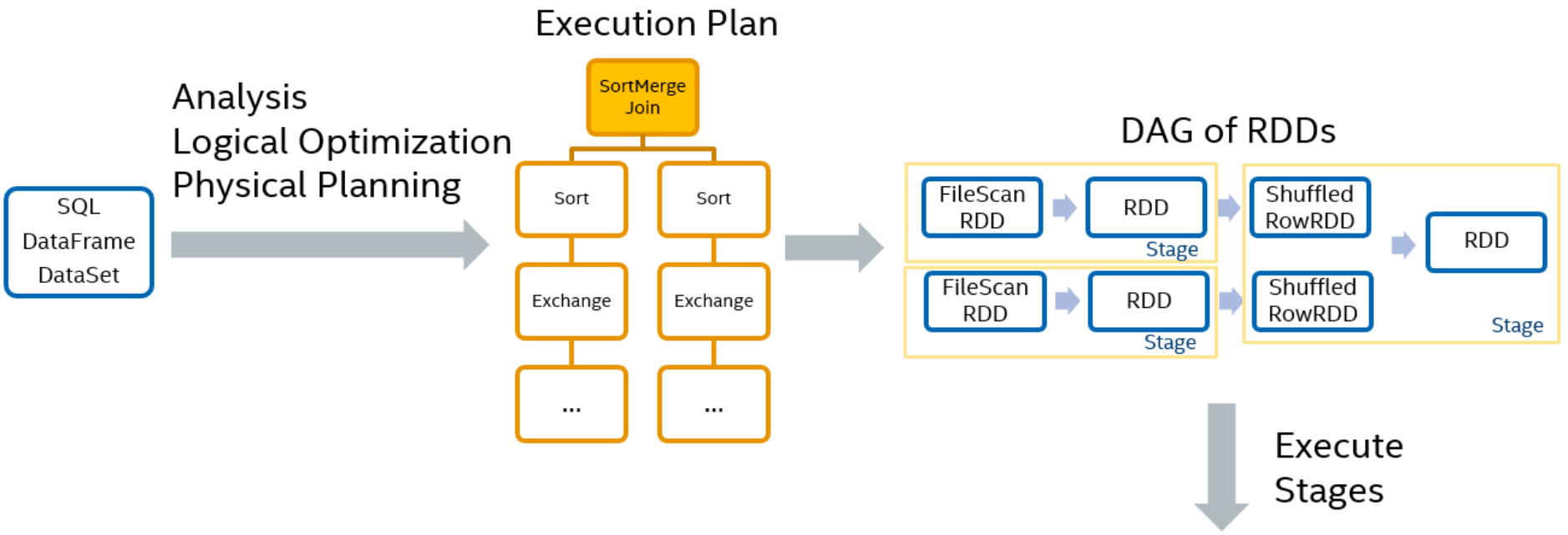
- A Join may takes intermediate results as inputs. Spark SQL may choose an inefficient join strategy if it doesn't know the exact size at planning phase.



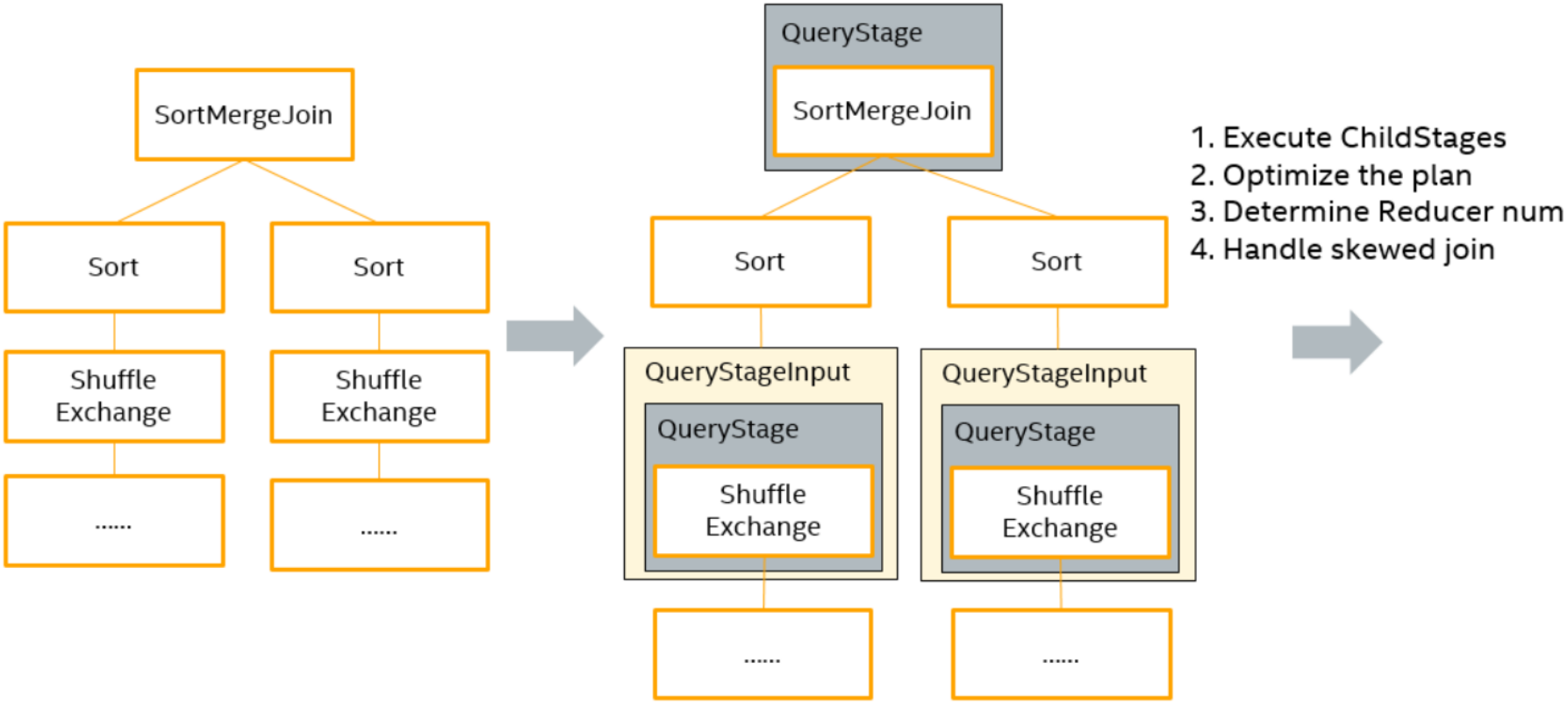
Data Skew in Join

- Data in some partitions are extremely larger than other partitions. Data skew is a common source of slowness for shuffle joins.
- Common ways to solve data skew
  - Increase shuffle partition number
  - Increase BroadcastJoin threashold
  - Add prefix to the skewed keys

Spark SQL Execution Mode



Spark SQL Adaptive Execution Mode

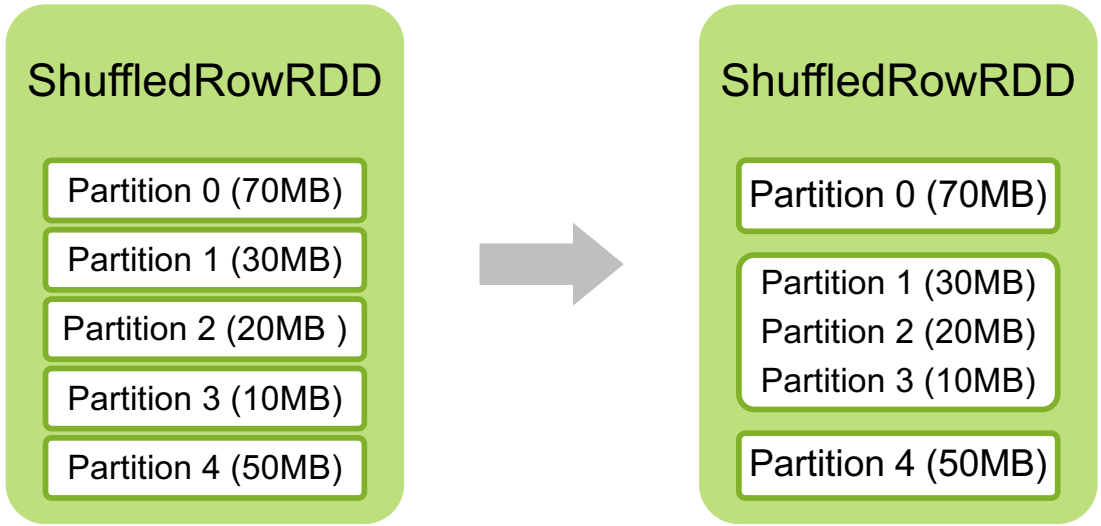


Auto Setting the Reducer Number

- Enable the feature
  - spark.sql.adaptive.enabled -> true
- Configure the behavior
  - Target input size for a reduce task
  - Min/Max shuffle partition number

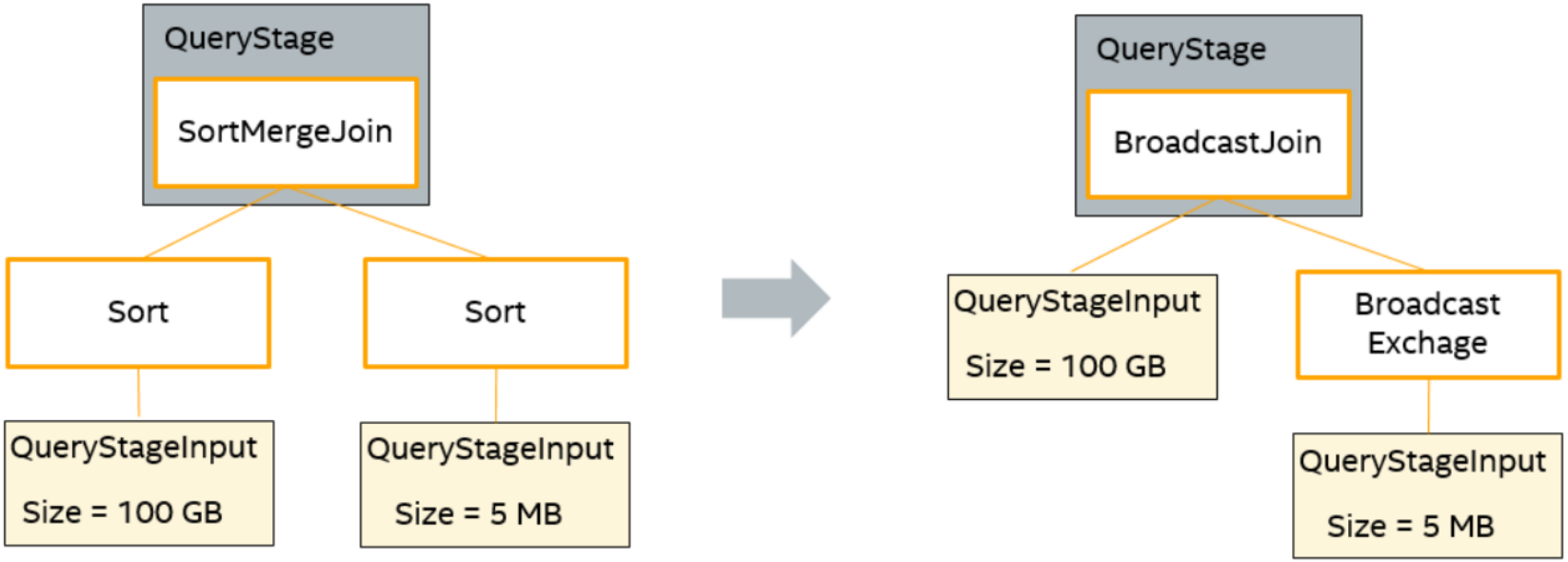
Auto Setting the Reducer Number

- Target size per reducer = 64 MB.
- Min-Max shuffle partition number = 1 to 5

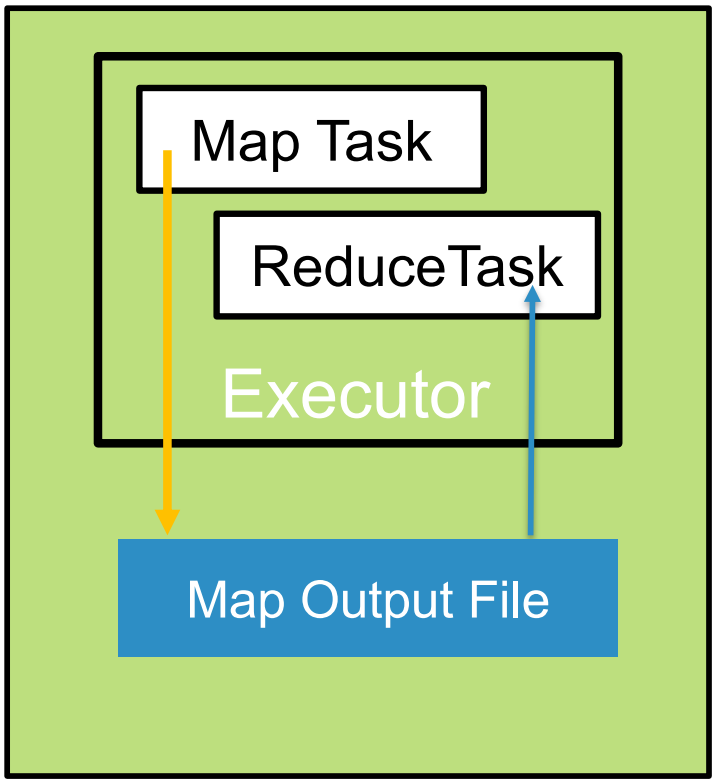


Adaptive Execution uses 3 reducers at runtime.

Optimize Join Strategy at Runtime



Optimize Join Strategy at Runtime

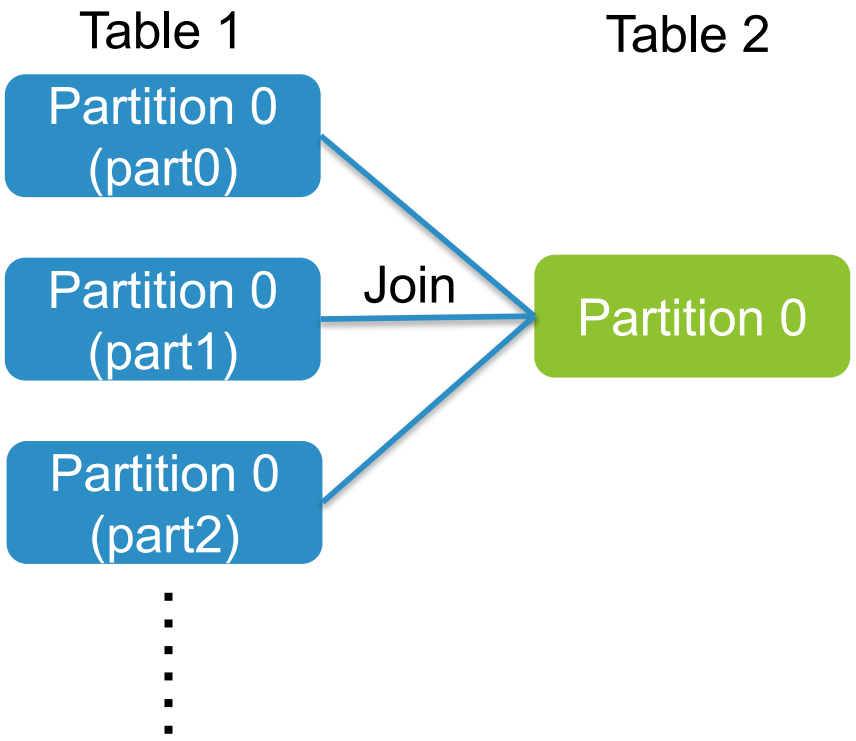


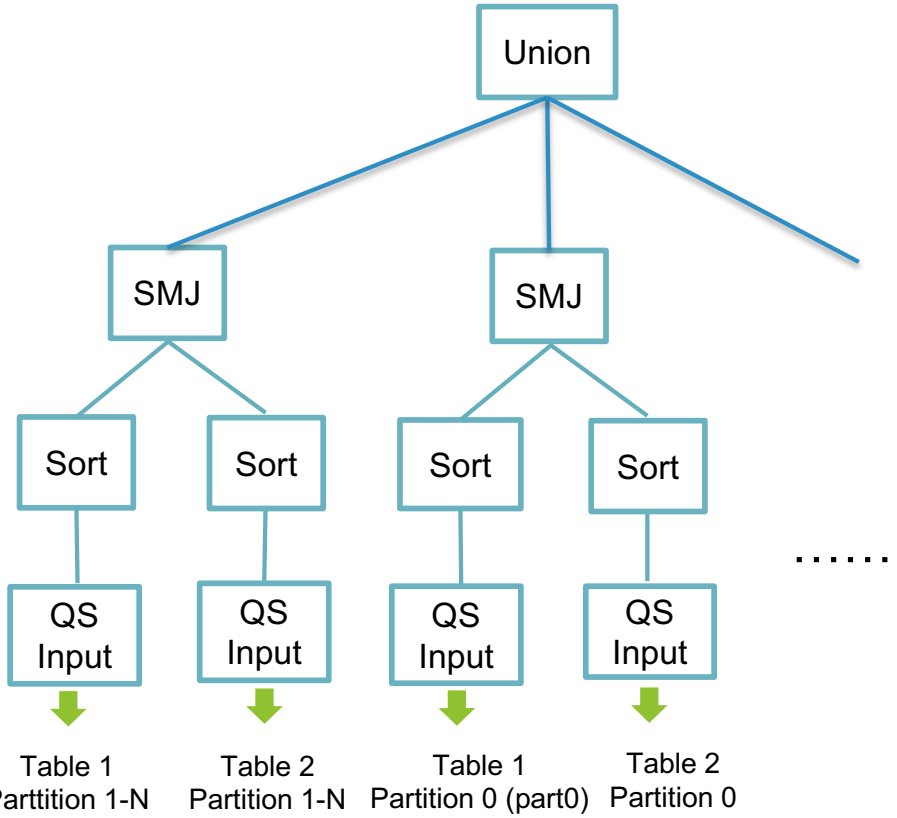
- After optimizing SortMergeJoin to BroadcastJoin, each reduce task local read the whole map output file and join with the broadcasted table.

Handle Skewed Join at Runtime

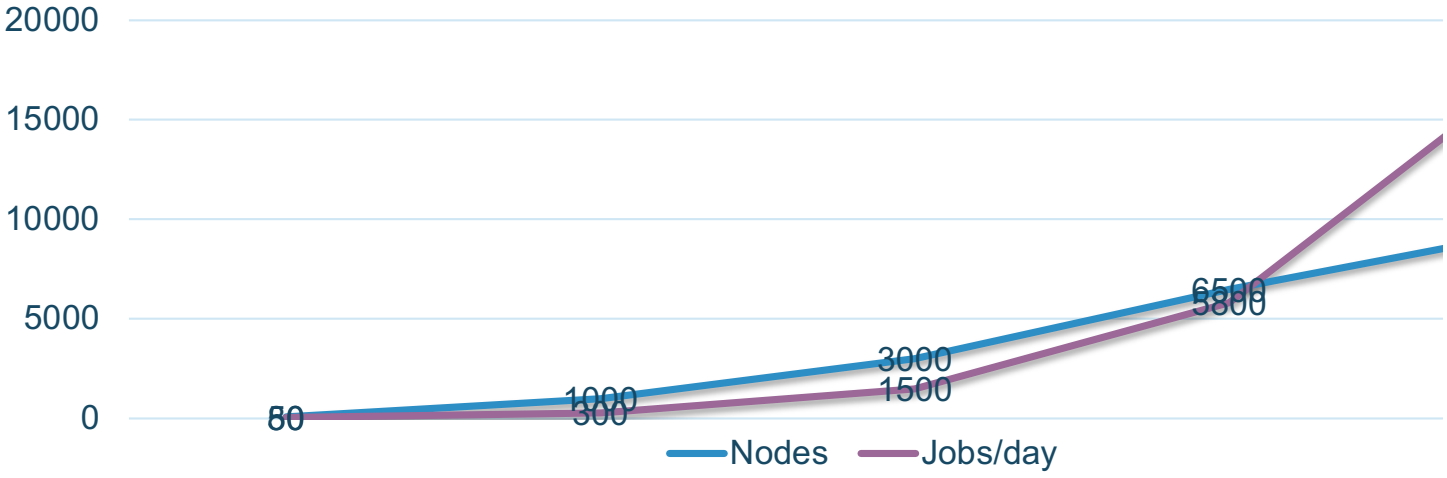
- spark.sql.adaptive.skewedJoin.enabled -> true
- A partition is thought as skewed if its data size or row count is N times larger than the median, and also larger than a pre-defined threshold.

Handle Skewed Join at Runtime





Spark in Baidu



2014	2015	2016	2017
• Spark import to Baidu • Version: 0.8	• Build standalone cluster • Integrate with in-house FS\Pub-Sub\DW • Version: 1.4	• Build Cluster over YARN • Integrate with in-house Resource Scheduler System • Version: 1.6	• SQL\Graph Service over Spark • OAP • Version: 2.1

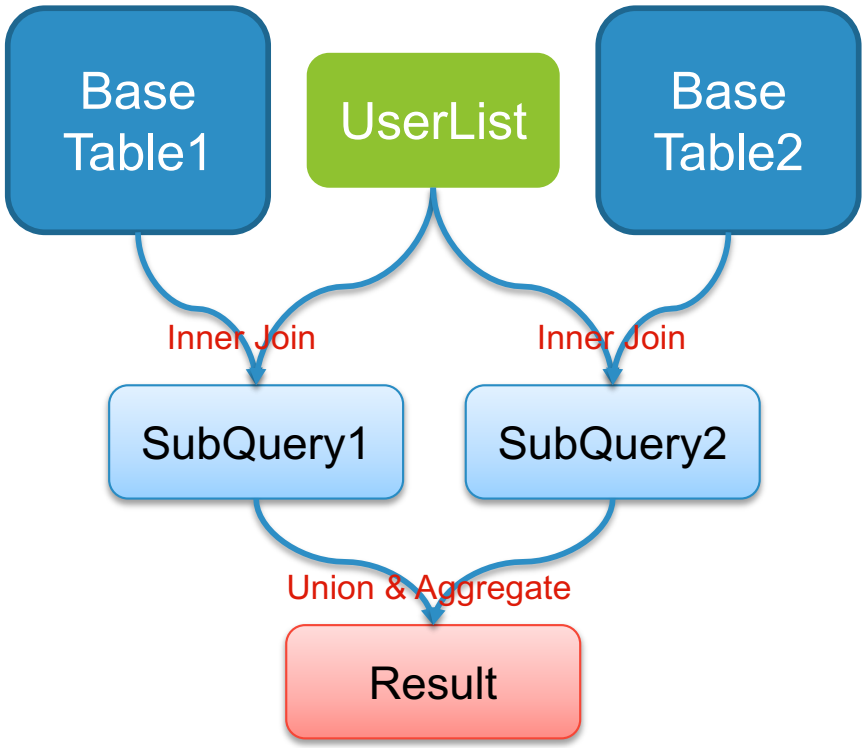
AE Boosting Scenario in Baidu

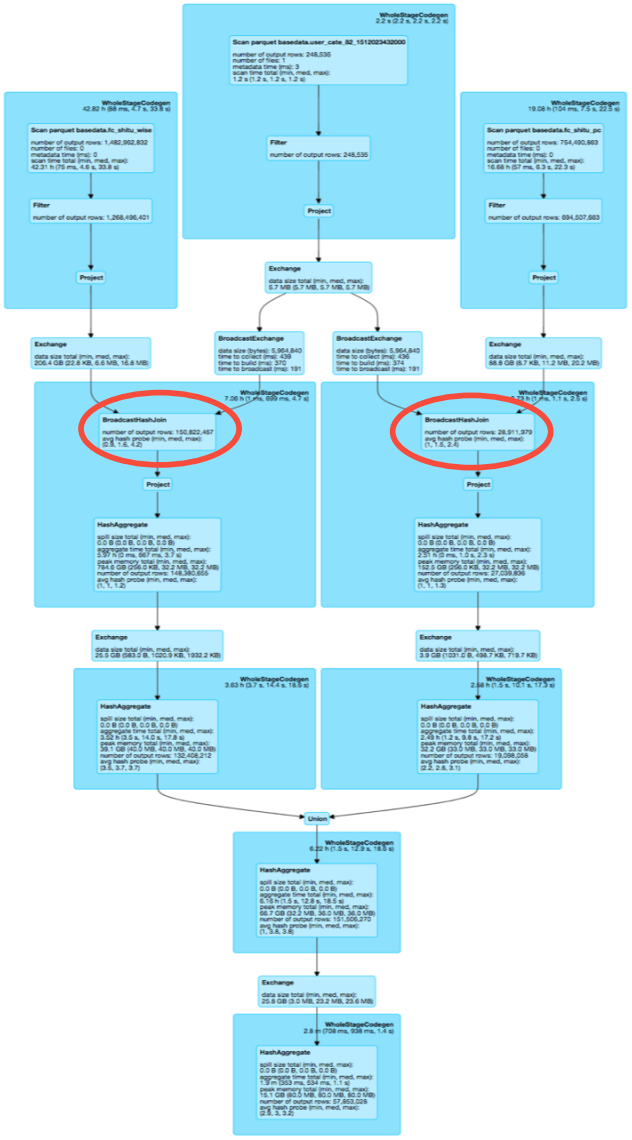
- Specific user scene(SortMergeJoin -> BroadcastJoin)
- Long running application or use Spark as a service
- Graph & ML

SortMergeJoin -> BroadcastJoin

- Common features in the scenario:
  - Small table join big table in sub query
  - Small table generated by sub query
- Key Point:
  - Identify & determine ‘small’ table
- Acceleration ratio:
  - 50%~200%

SortMergeJoin -> B





m(t.num2),

num1, sum
ge_table1 t1
E (event_da
c2, c3, c4

num1, sum
ge_table2 t1
E (event_da
c2, c3, c4

Long Running Application

- Including scenario:
  - Long running batch job(> 1 hour)
  - Using Spark as a service
    - (Livy\Baidu BigSQL\Spark Shell\Zeppelin)
  - Spark Streaming
- Key Point:
  - Adaptive parallelism adjustment
- Acceleration ratio:
  - 50%~100%

Long Running Application

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
8	parquet at [redacted]15DayBaseJoin.scala:114 +details	2018/05/10 13:14:31	15 min	2000/2000 (18 killed:			1177.0 GB	
7	parquet at [redacted]15DayBaseJoin.scala:114 +details	2018/05/10 12:50:22	24 min	10000/10000 (4 killed:			1317.8 GB	1177.0 GB
6	parquet at [redacted]15DayBaseJoin.scala:114 +details	2018/05/10 12:41:42	5.8 min	10000/10000 (80 killed:			276.0 GB	244.0 GB
5	parquet at [redacted]15DayBaseJoin.scala:114 +details	2018/05/10 12:40:15	4.8 min	7000/7000 (96 killed:	870.0 GB			1073.8 GB
4	rdd at [redacted]15DayUtils.scala:172 +details	2018/05/10 12:40:15	21 s	2000/2000 (110 killed:	51.0 GB			64.7 GB
3	rdd at [redacted]15DayUtils.scala:172 +details	2018/05/10 12:40:15	1.4 min	2000/2000 (79 killed:	140.8 GB			211.2 GB

Duration: 52min
100 instance
10G executor.mem
4 executor.cores

AE enable False

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
19	parquet at [redacted]15DayBaseJoin.scala:114 +details	2018/05/10 10:51:40	8.0 min	2000/2000 (2 running)			1170.9 GB	
13	run at Executors.java:511 +details	2018/05/10 10:37:25	14 min	3334/3334 (289 killed:			1287.0 GB	1170.9 GB
8	run at Executors.java:511 +details	2018/05/10 10:32:24	4.0 min	7000/7000 (69 killed:	870.0 GB			1073.8 GB
7	run at Executors.java:511 +details	2018/05/10 10:32:24	2.3 min	556/556			276.0 GB	213.2 GB
4	run at Executors.java:511 +details	2018/05/10 10:31:14	41 s	2000/2000 (23 killed:	140.8 GB			211.2 GB
3	run at Executors.java:511 +details	2018/05/10 10:31:13	33 s	2000/2000 (134 killed:	51.0 GB			64.8 GB

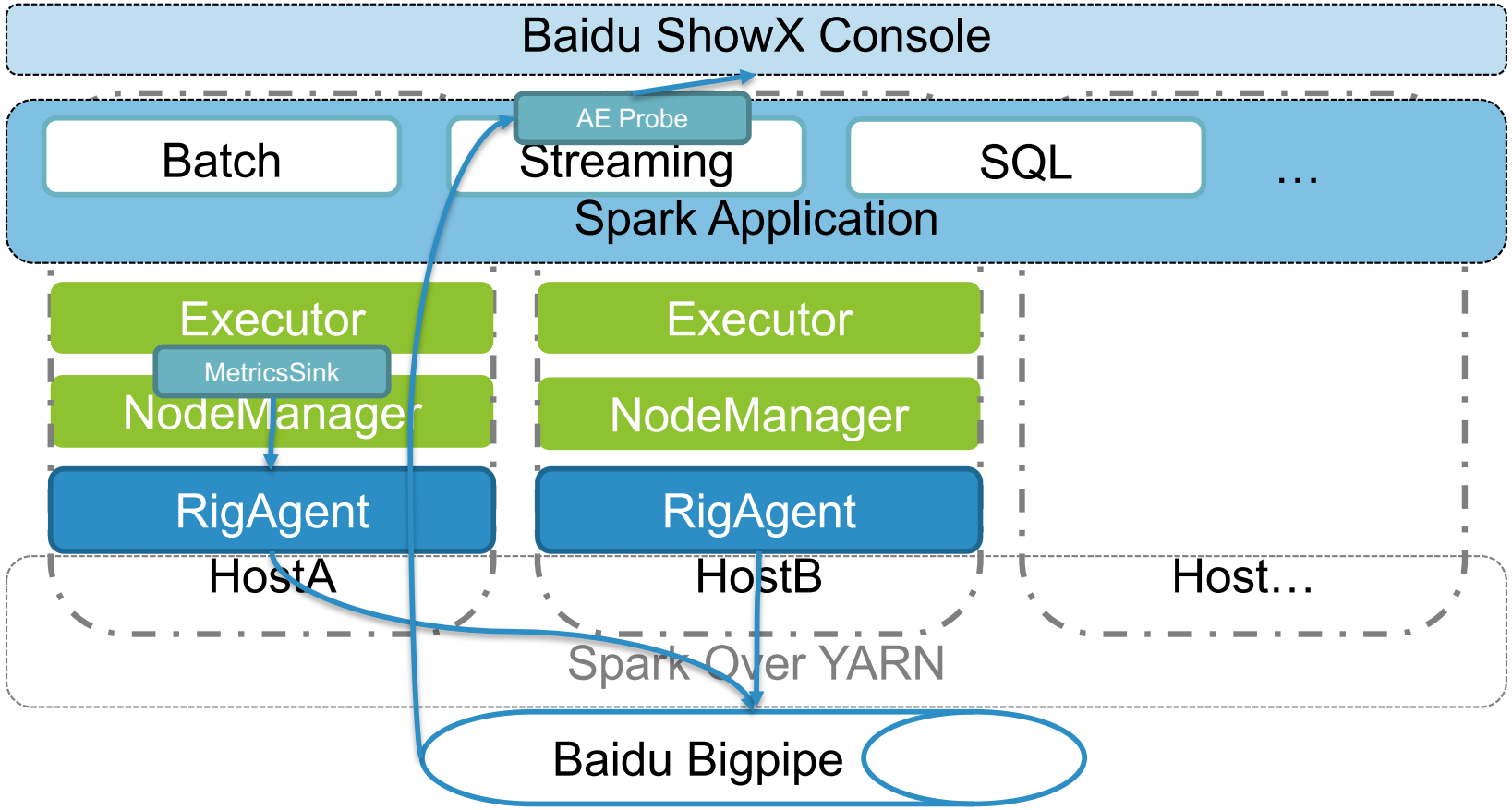
Duration: 30min
100 instance
10G executor.mem
4 executor.cores

AE enable True
Min/MaxNumPostShufflePartitions 400/10000
targetPostShuffleInputSize 512M

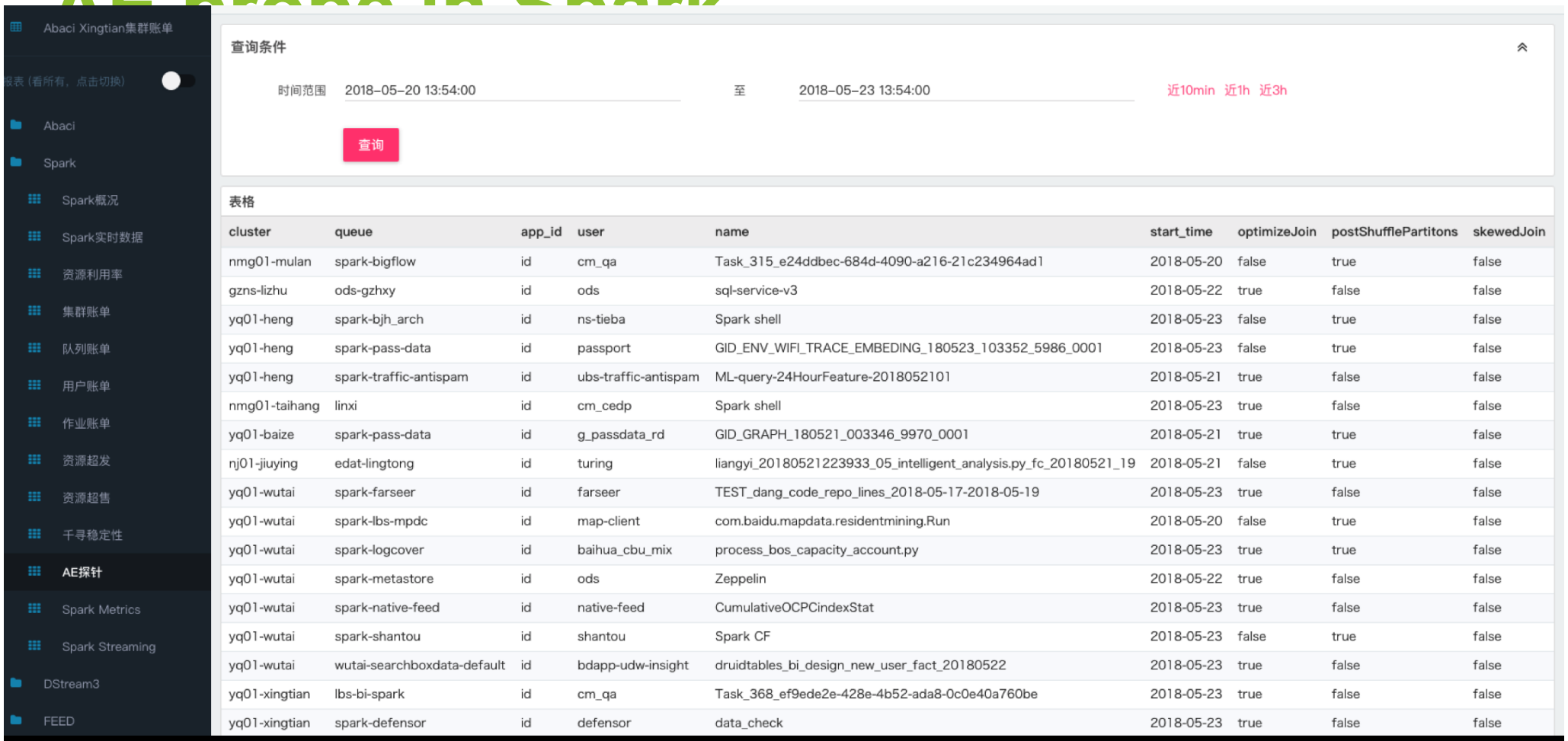
GraphFrame & MLlib

- Including scenario:
  - GraphFrame APP
  - MLlib
- Key Point:
  - Adaptive parallelism adjustment
- Acceleration ratio:
  - 50%~100%

AE probe in Spark



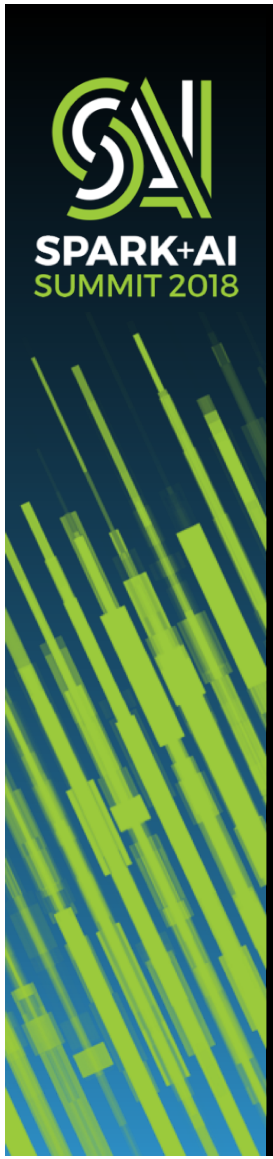
AE probe in Spark



cluster	queue	app_id	user	name	start_time	optimizeJoin	postShufflePartitons	skewedJoin
nmg01-mulan	spark-bigflow	id	cm_qa	Task_315_e24ddbec-684d-4090-a216-21c234964ad1	2018-05-20	false	true	false
gzns-lizhu	ods-gzhxy	id	ods	sql-service-v3	2018-05-22	true	false	false
yq01-heng	spark-bjh_arch	id	ns-tieba	Spark shell	2018-05-23	false	true	false
yq01-heng	spark-pass-data	id	passport	GID_ENV_WIFI_TRACE_EMBEDING_180523_103352_5986_0001	2018-05-23	false	true	false
yq01-heng	spark-traffic-antispam	id	ubs-traffic-antispam	ML-query-24HourFeature-2018052101	2018-05-21	true	false	false
nmg01-taihang	linxi	id	cm_cedp	Spark shell	2018-05-23	true	false	false
yq01-baize	spark-pass-data	id	g_passdata_rd	GID_GRAPH_180521_003346_9970_0001	2018-05-21	true	true	false
nj01-jiuying	edat-lingtong	id	turing	liangyi_20180521223933_05_intelligent_analysis.py_fc_20180521_19	2018-05-21	false	true	false
yq01-wutai	spark-farseer	id	farseer	TEST_dang_code_repo_lines_2018-05-17-2018-05-19	2018-05-23	true	false	false
yq01-wutai	spark-lbs-mpdc	id	map-client	com.baidu.mapdata.residentmining.Run	2018-05-20	false	true	false
yq01-wutai	spark-logcover	id	baihua_cbu_mix	process_bos_capacity_account.py	2018-05-23	true	true	false
yq01-wutai	spark-metastore	id	ods	Zeppelin	2018-05-22	true	false	false
yq01-wutai	spark-native-feed	id	native-feed	CumulativeOCPCindexStat	2018-05-23	true	false	false
yq01-wutai	spark-shantou	id	shantou	Spark CF	2018-05-23	false	true	false
yq01-wutai	wutai-searchboxdata-default	id	bdapp-udw-insight	druidtables_bi_design_new_user_fact_20180522	2018-05-23	true	false	false
yq01-xingtian	lbs-bi-spark	id	cm_qa	Task_368_ef9ede2e-428e-4b52-ada8-0c0e40a760be	2018-05-23	true	false	false
yq01-xingtian	spark-defensor	id	defensor	data_check	2018-05-23	true	false	false

Takeaways

- Three main features in our adaptive execution
  - Auto setting the shuffle partition number
  - Optimize join strategy at runtime
  - Handle skewed join at runtime
- For more information about our implementation:
  - https://issues.apache.org/jira/browse/SPARK-23128
  - https://github.com/Intel-bigdata/spark-adaptive



Thank you!

Carson Wang carson.wang@intel.com
Yuanjian Li liyuanjian@baidu.com

#Exp5SAIS