Deep Dive: Scheduler in Apache Spark

Xingbo Jiang

Spark Summit | SF | Jun 2018

About Me

- Software Engineer at databricks
- Active contributor of Apache Spark

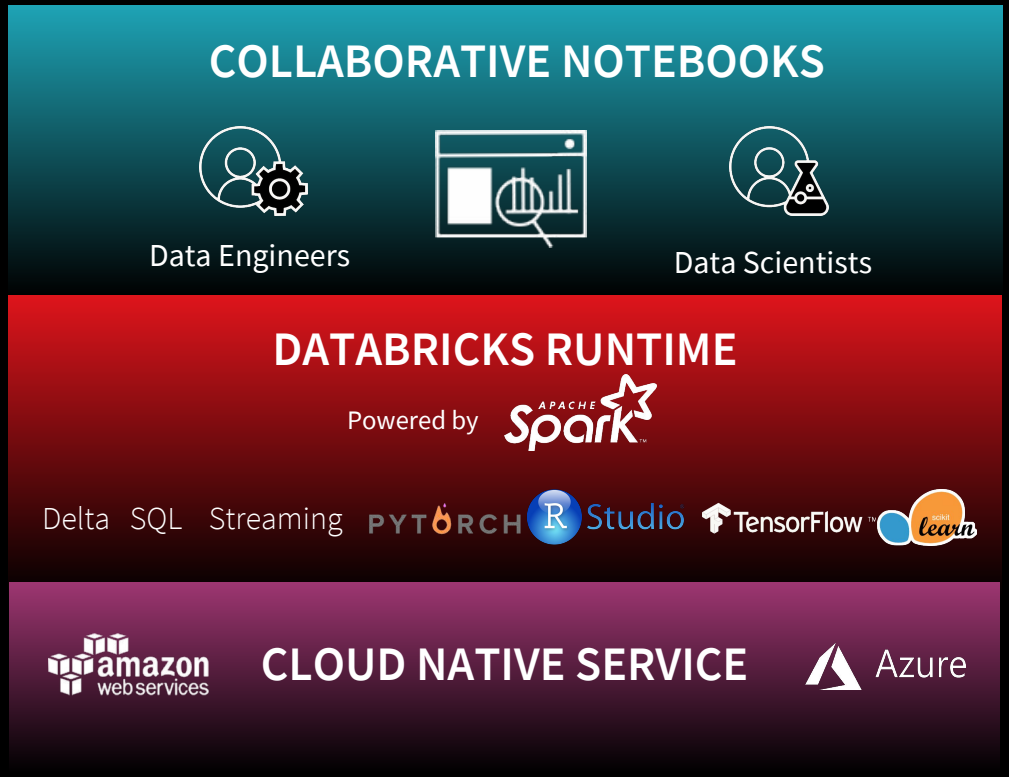
Xingbo Jiang (Github: jiangxb1987)

Databricks' Unified Analytics Platform

Unifies Data Engineers and Data Scientists

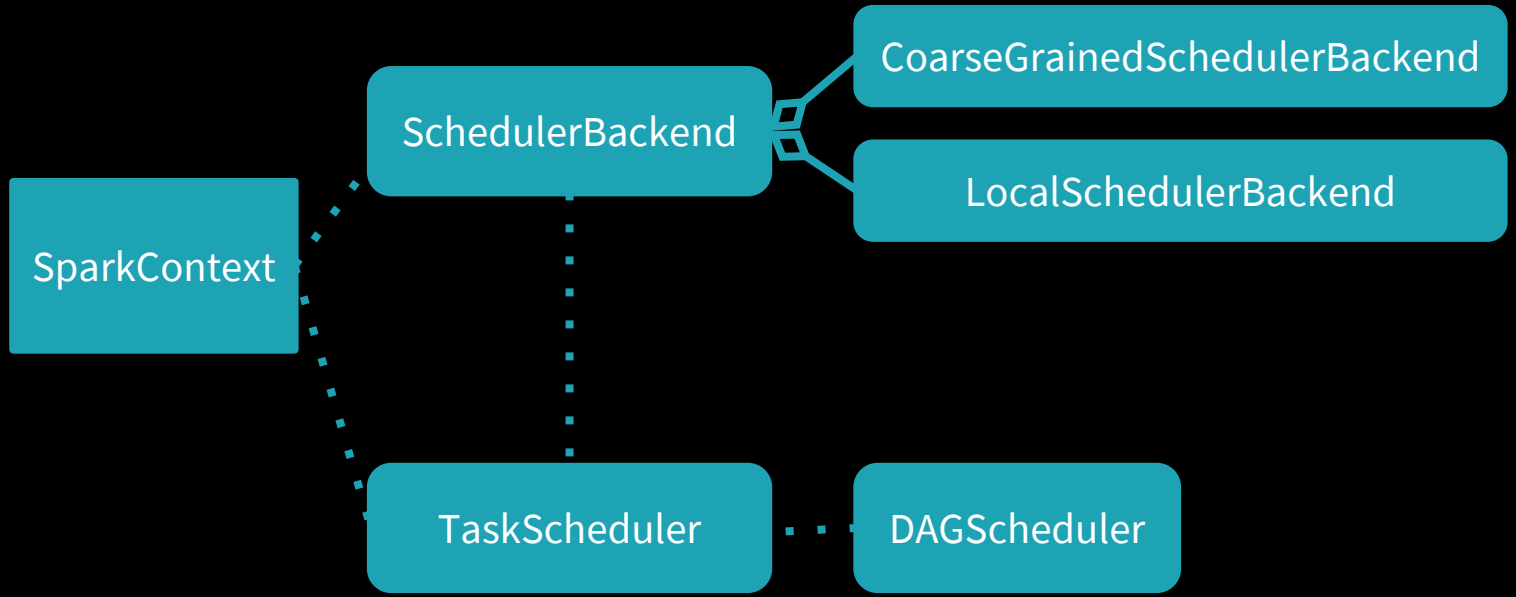
Unifies Data and AI Technologies

Eliminates infrastructure complexity



databricks

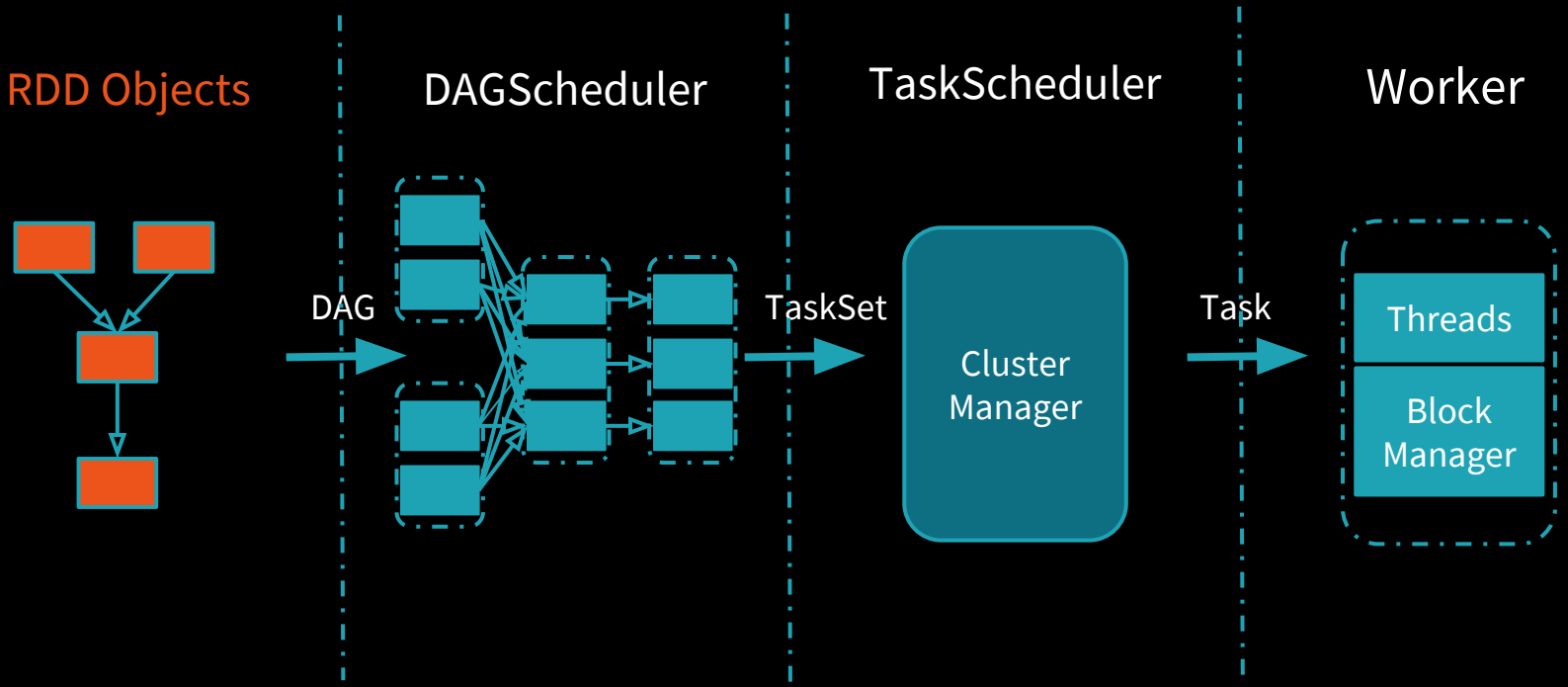
Architecture



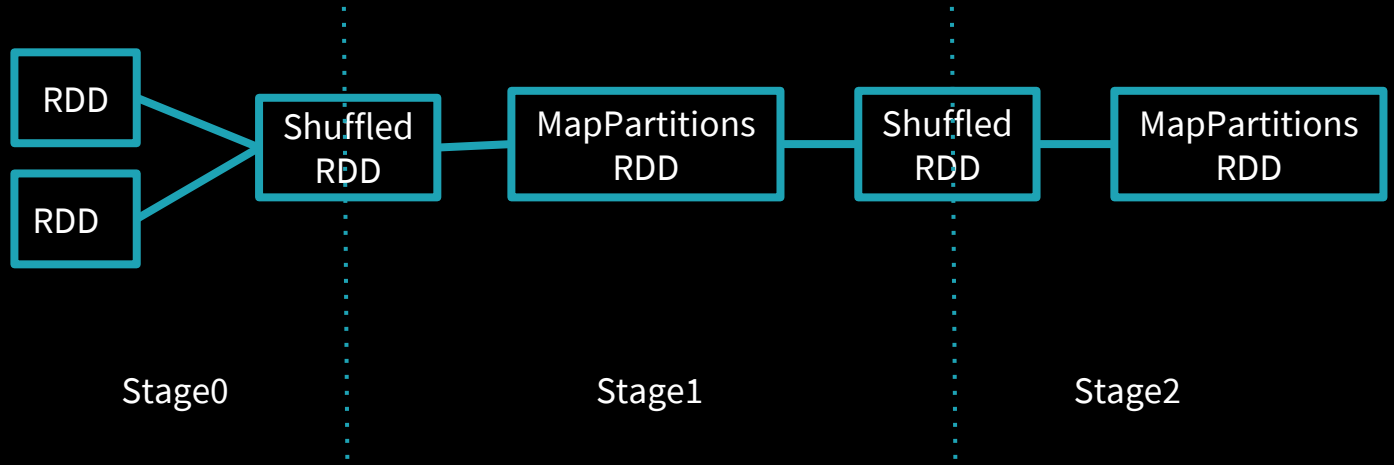
SparkContext

- Main entry point
- Create SchedulerBackend/TaskScheduler/DAGScheduler on initialization
- submitJob()/cancelJob()

Scheduling Process

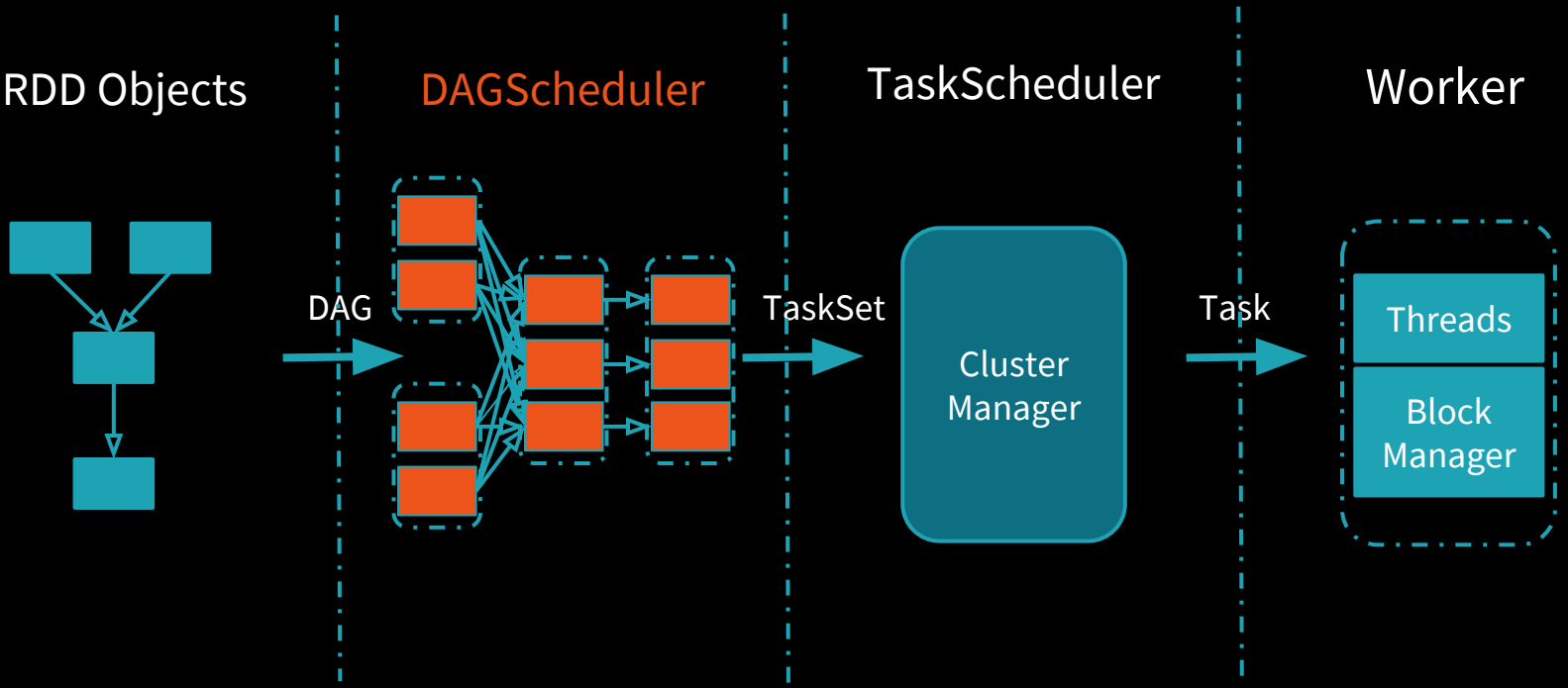


RDD → Stage



rdd1.join(rdd2).groupBy(...).filter(...)

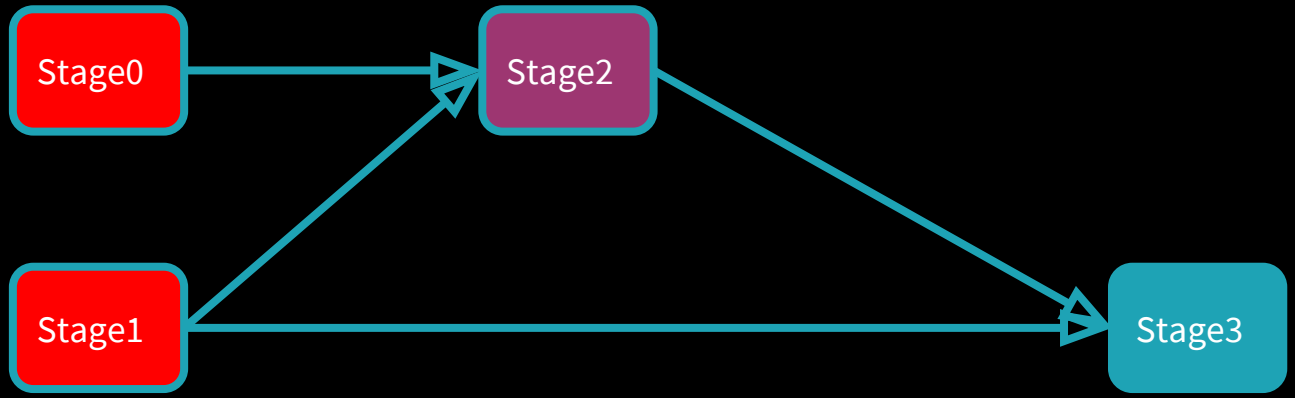
Scheduling Process



DAGScheduler

- Implement stage-oriented scheduling
  - Compute a DAG of stages for submitted job
  - Keep track of materialized RDD/Stage outputs
  - Find a minimal schedule to run the job
- Stage → TaskSet

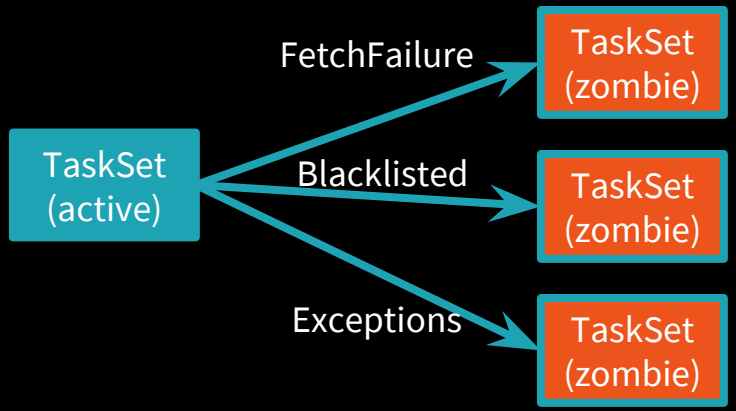
Stage Dependency



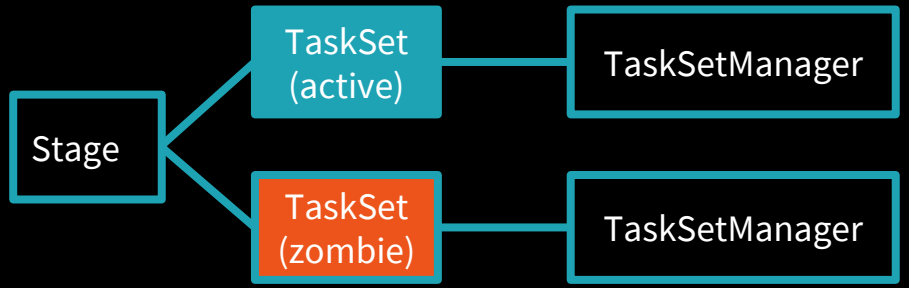
TaskSet

- A set of tasks submitted to compute missing partitions of a particular stage
- A stage can correspond to multiple TaskSets

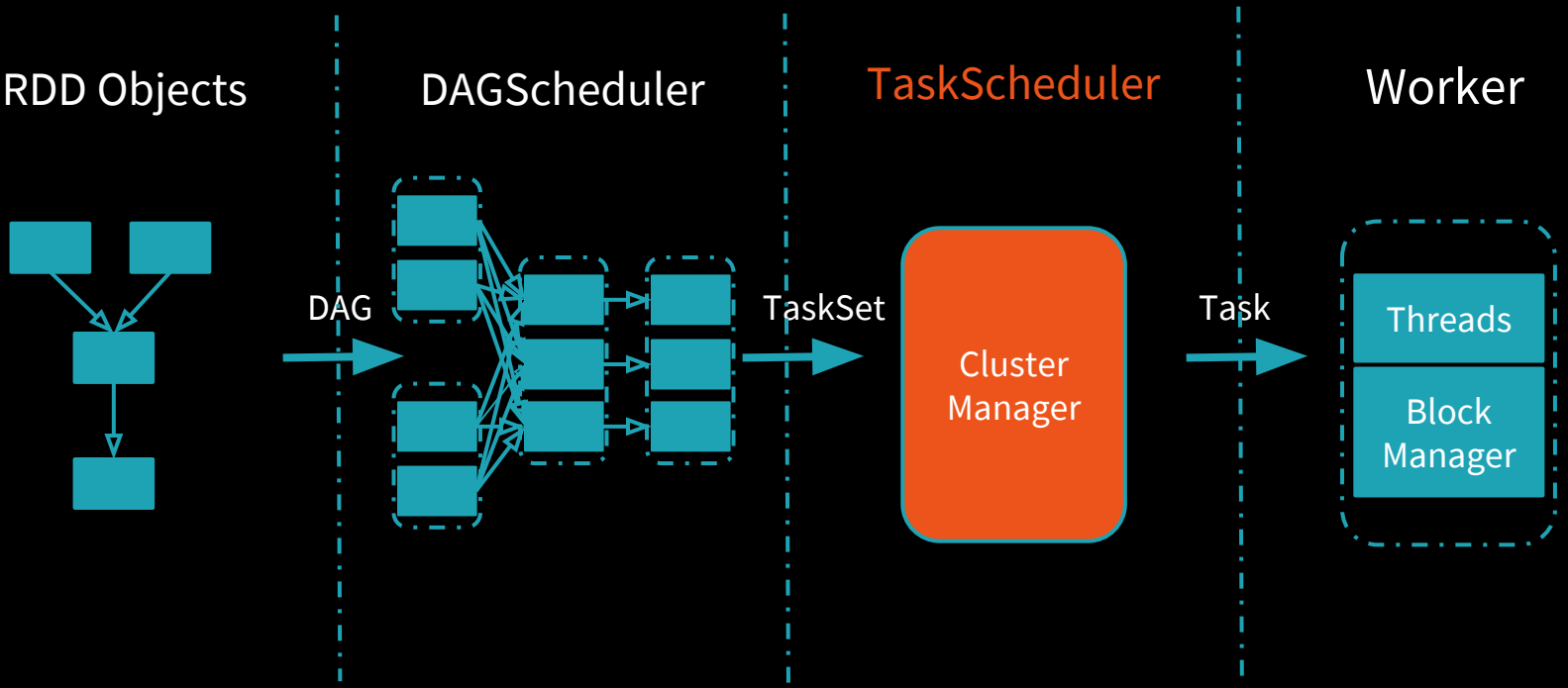
Zombie TaskSet



Stage → TaskSet



Scheduling Process



TaskScheduler

- DAGScheduler submit set of tasks to TaskScheduler
- Schedule and monitor tasks with SchedulerBackend
- Return events to DAGScheduler
  - JobSubmitted/JobCancelled
  - MapStageSubmitted/StageCancelled
  - CompletionEvent

Scheduling Tasks

- Batch scheduling approach
  - Get all available slots
  - Schedule tasks with locality preference
- Barrier scheduling approach (SPARK-24375)
  - Wait until all tasks in the same TaskSet can be scheduled at the same time
  - Retry all tasks in the TaskSet if any task fails

Scheduling TaskSets

- FIFO by default
- When multiple users run jobs on a single cluster?
  - Long running tasks may block later tasks
- Use Fair Scheduling to ensure minimal share of resources for each user

Scheduling TaskSets

- FIFO
  - FIFO between TaskSetManagers
  - Order by Priority, StageId
- Fair Scheduling
  - FS between Pools, and FIFO or FS within Pools
  - Try to launch tasks in TaskSet that is further below the minShare
  - If both TaskSets are running above minShare, order by weighted number of running tasks

Scheduling Tasks in a TaskSet

- Try to achieve better locality for each task
  - Less data transfer over network
  - Higher Performance
- Locality can have different levels
  - Process locality
  - Node locality
  - Rack locality

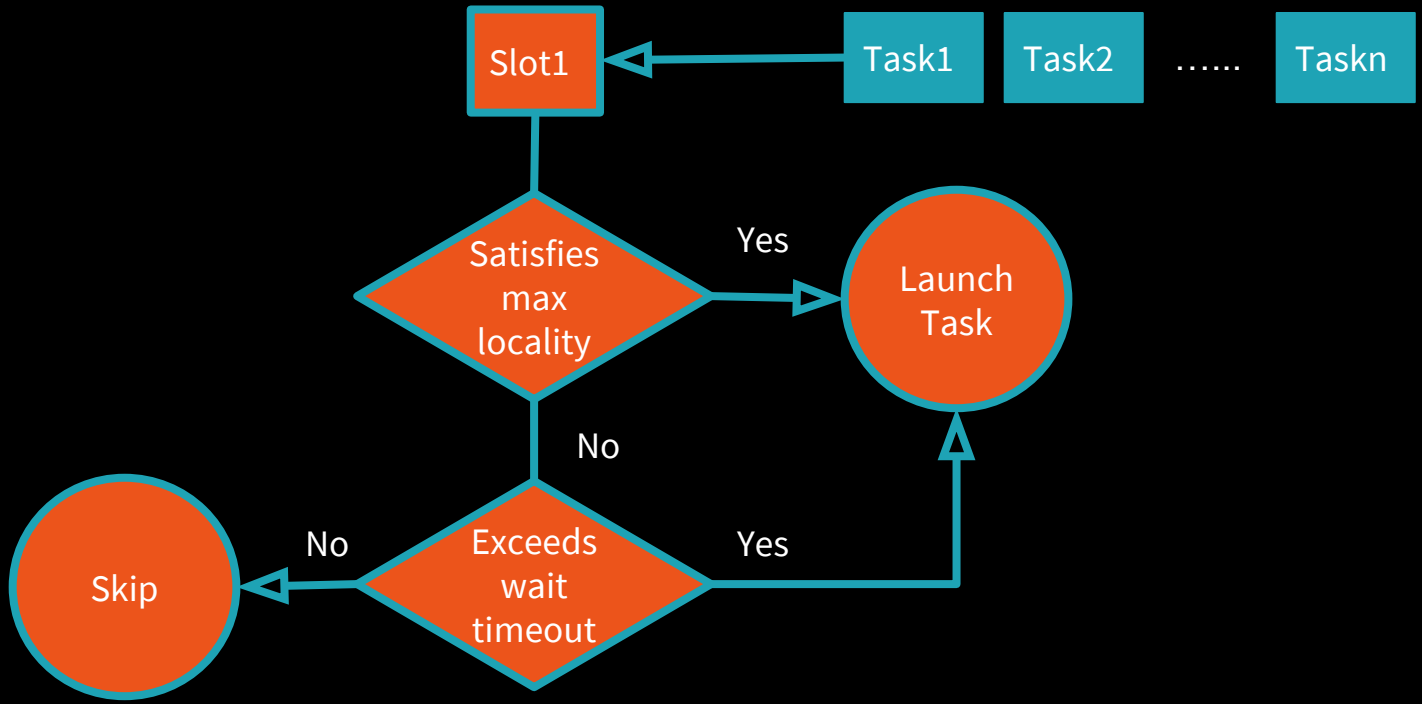
TaskSetManager

- Schedule tasks within a single TaskSet
- Implement locality-aware scheduling via delay scheduling

Delay Scheduling

- Wait for tasks to finish (vs killing running tasks) to assign slots
- Wait for a few extra time to achieve better locality

Delay Scheduling



Delay Scheduling

```
when a heartbeat is received from node n:
  if n has a free slot then:
    compute maxAllowedLocality for pending tasks
    if exists task t can launch on n with locality <= maxAllowedLocality:
      launch t
      update currentLocality
    else if waitTime > maxDelayTime:
      launch t
    else:
      // Wait for next round of scheduling
  endif
```

Delay Scheduling

For more detail,

> *Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. 2010. Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling.*

Handle Failures

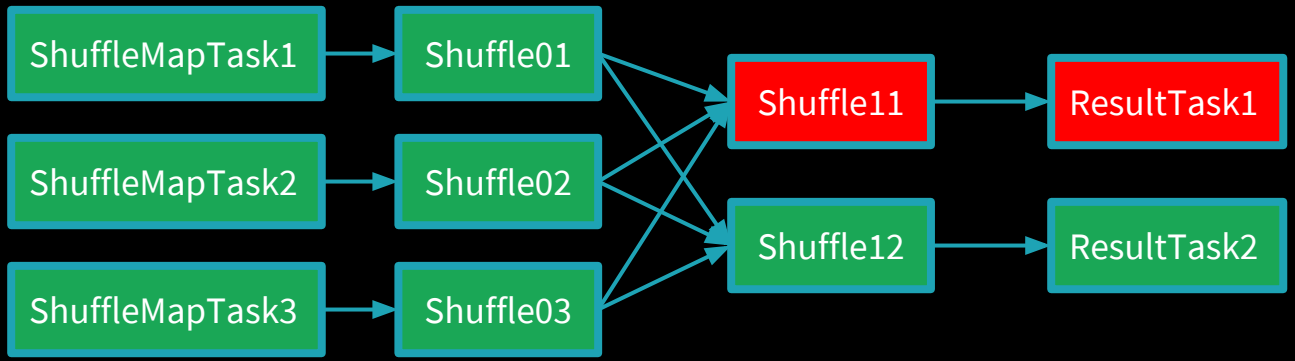
Task Failure

- Record the failure count of the task
- Retry the task if failure count < maxTaskFailures
- Abort the stage and corresponding jobs if count >= maxTaskFailures

Fetch Failure

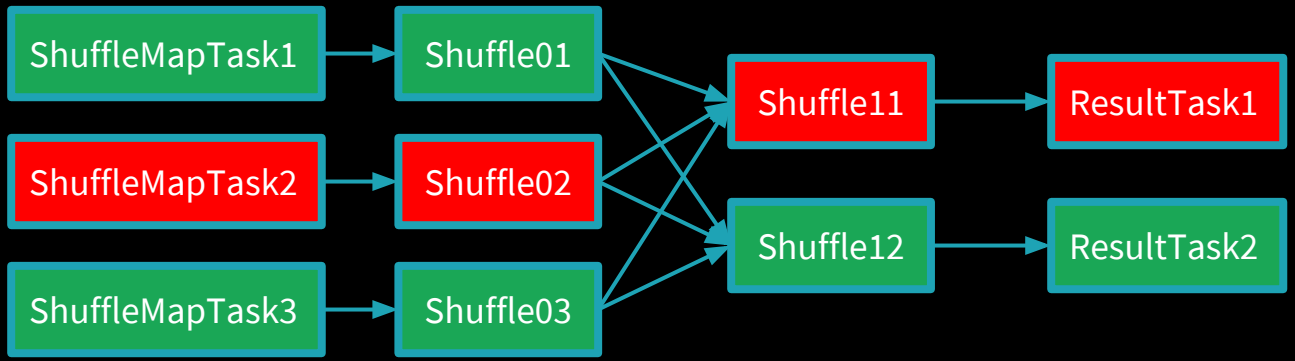
- Don't count the failure into task failure count
- Retry the stage if stage failure < maxStageFailures
- Abort the stage and corresponding jobs if stage failure >= maxStageFailures
- Mark executor/host as lost (optional)

Retry Stage



Note: Shuffle01/02/03 not on the same host with Shuffle11, don’t need to retry.

Retry Stage

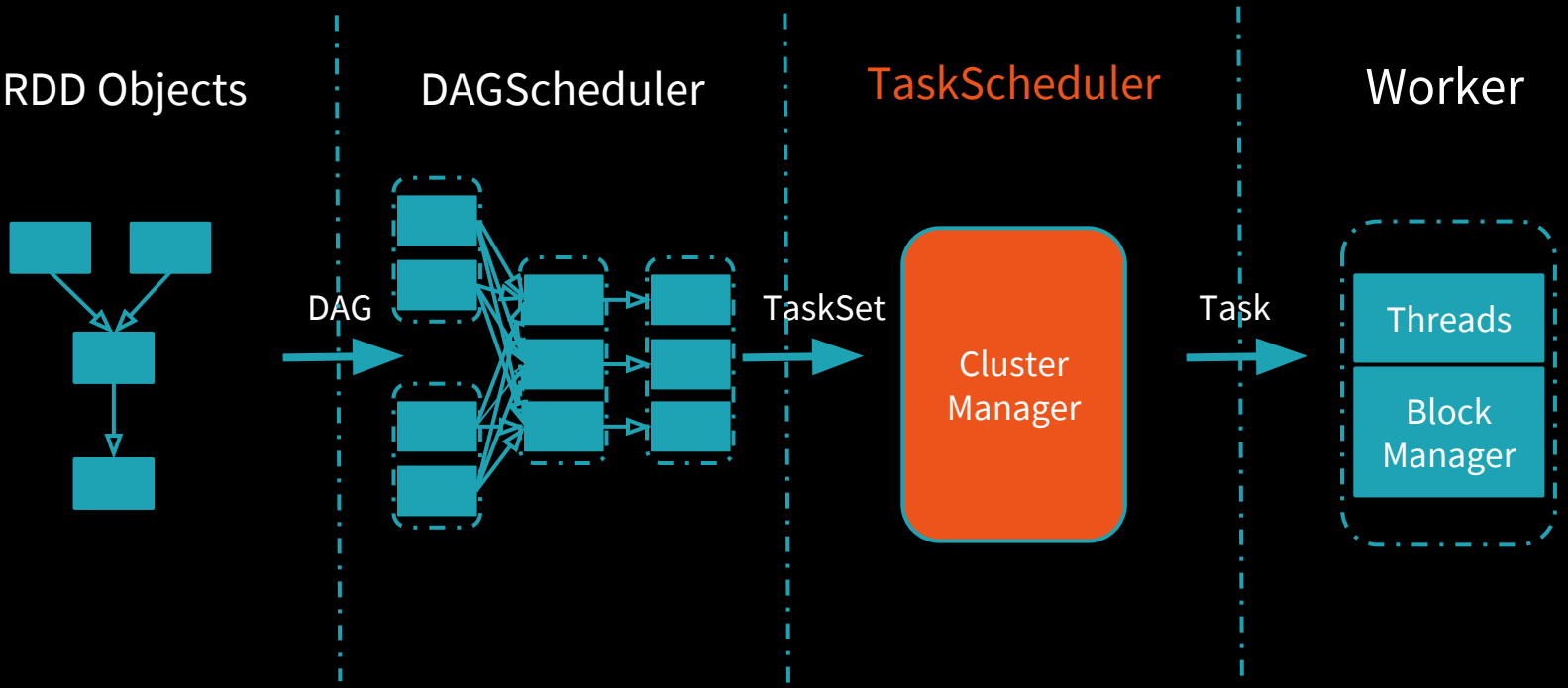


Note: Shuffle02 on the same host with Shuffle11, retry ShuffleMapTask2.

Retry Stage/TaskSet

- Retry Stage
  - Retry parent stages if necessary
  - Only retry tasks that have missing partitions
- Retry TaskSet
  - Mark current TaskSet as zombie
  - Don’t kill running tasks

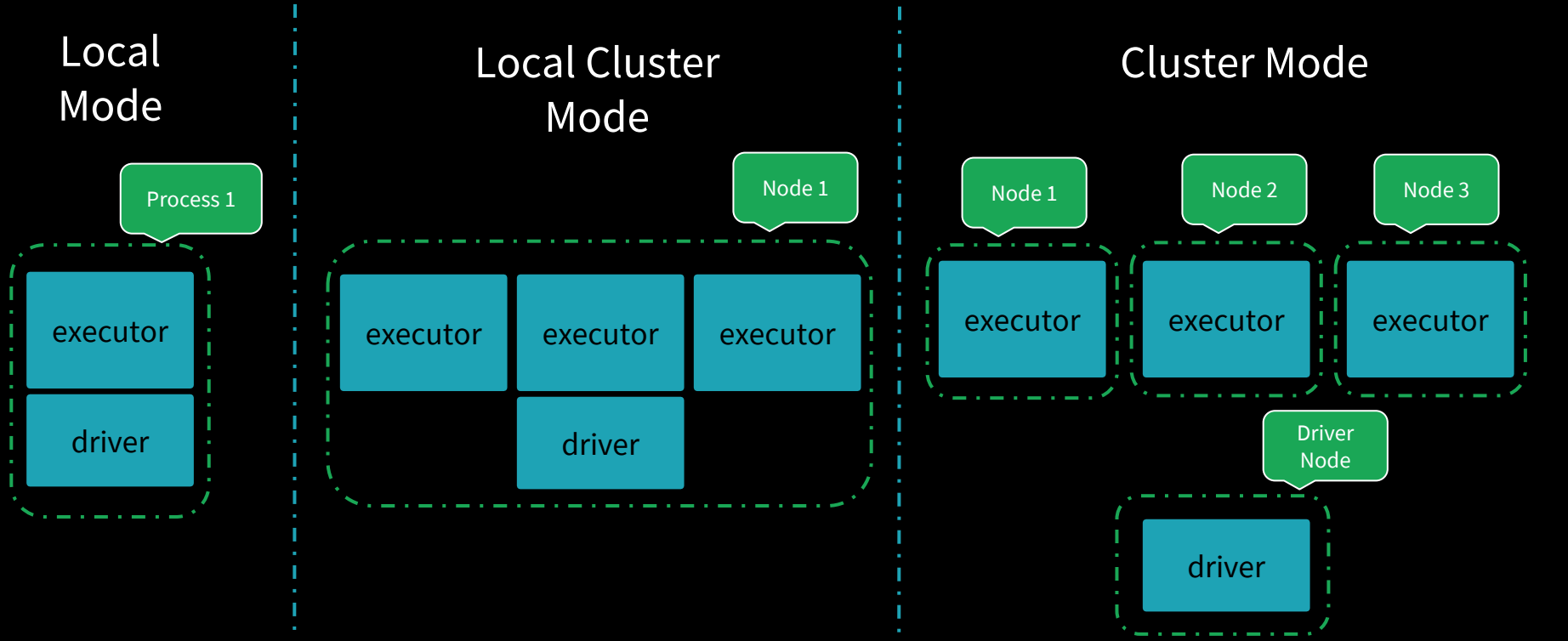
Scheduling Process



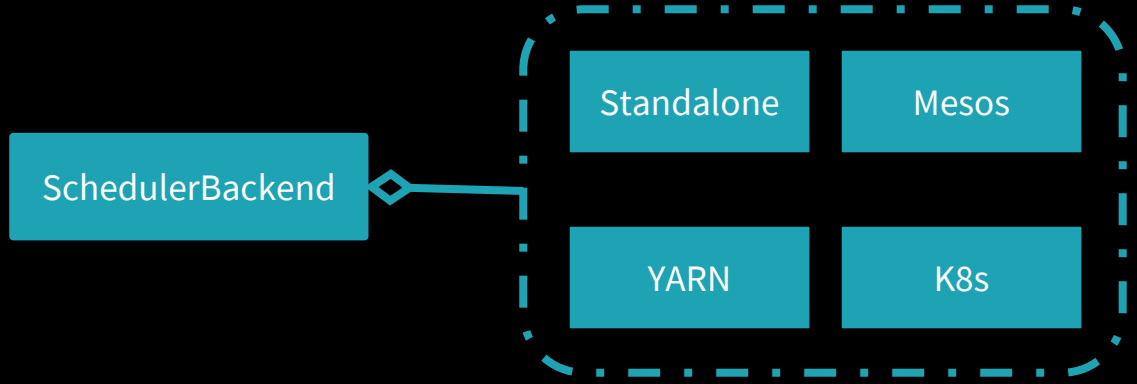
SchedulerBackend

- Manage resources to schedule tasks on
- Running mode
  - Local mode
  - Local cluster mode
  - Cluster mode

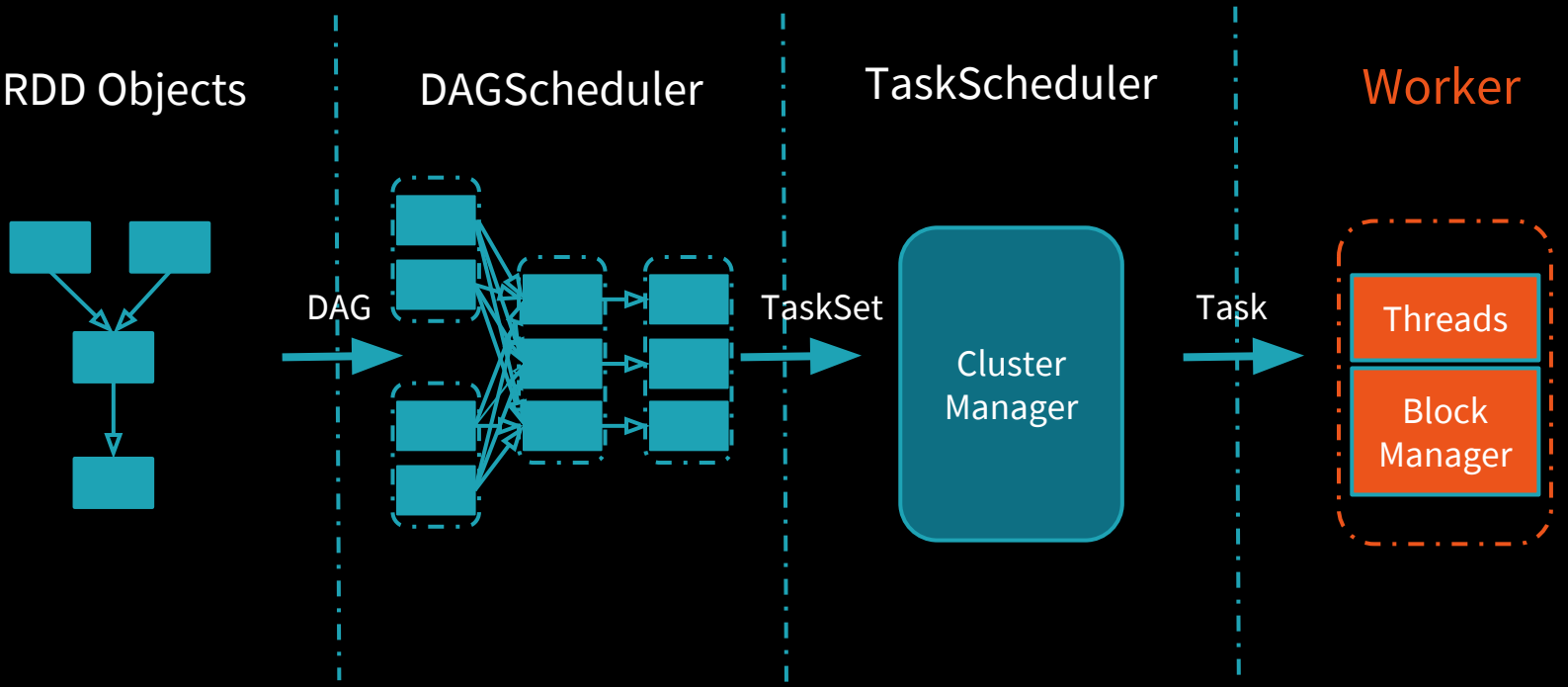
SchedulerBackend



SchedulerBackend



Scheduling Process



Worker

- A container for executors
- Executors provide
  - Threads to run Tasks
  - BlockManager to store/serve blocks
- ExternalShuffleService (optional)
  - Serve blocks even the executor has exited

Improve Job Performance

- Break long-running tasks into simple/short tasks
  - More chances to have available slots
  - Less cost recovery from failure
- Broadcast small hot input files
  - Multiple jobs/tasks need to read the same data file
  - Small data files only present on a small fraction of nodes
  - Broadcast to achieve better locality

Thank you

Xingbo Jiang (xingbo.jiang@databricks.com)

databricks