Runtime Improvements For Flink As A Unified Engine

公司：阿里巴巴

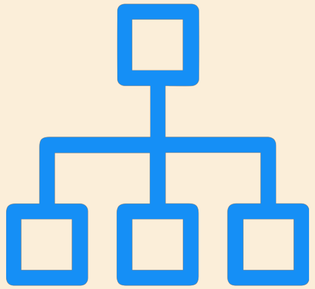
演讲者：马国维 / 陶阳宇

- Flink in Alibaba
- 阿里巴巴的FLINK技术生态
- New Flink API Stack For Unified Processing
- 新Flink API栈
- Flink Runtime Improvements
- Flink引擎的改进
- Future Plans
- 未来展望

- **Flink in Alibaba**

Flink In Alibaba

Cluster



10K Nodes

State

PetaBytes

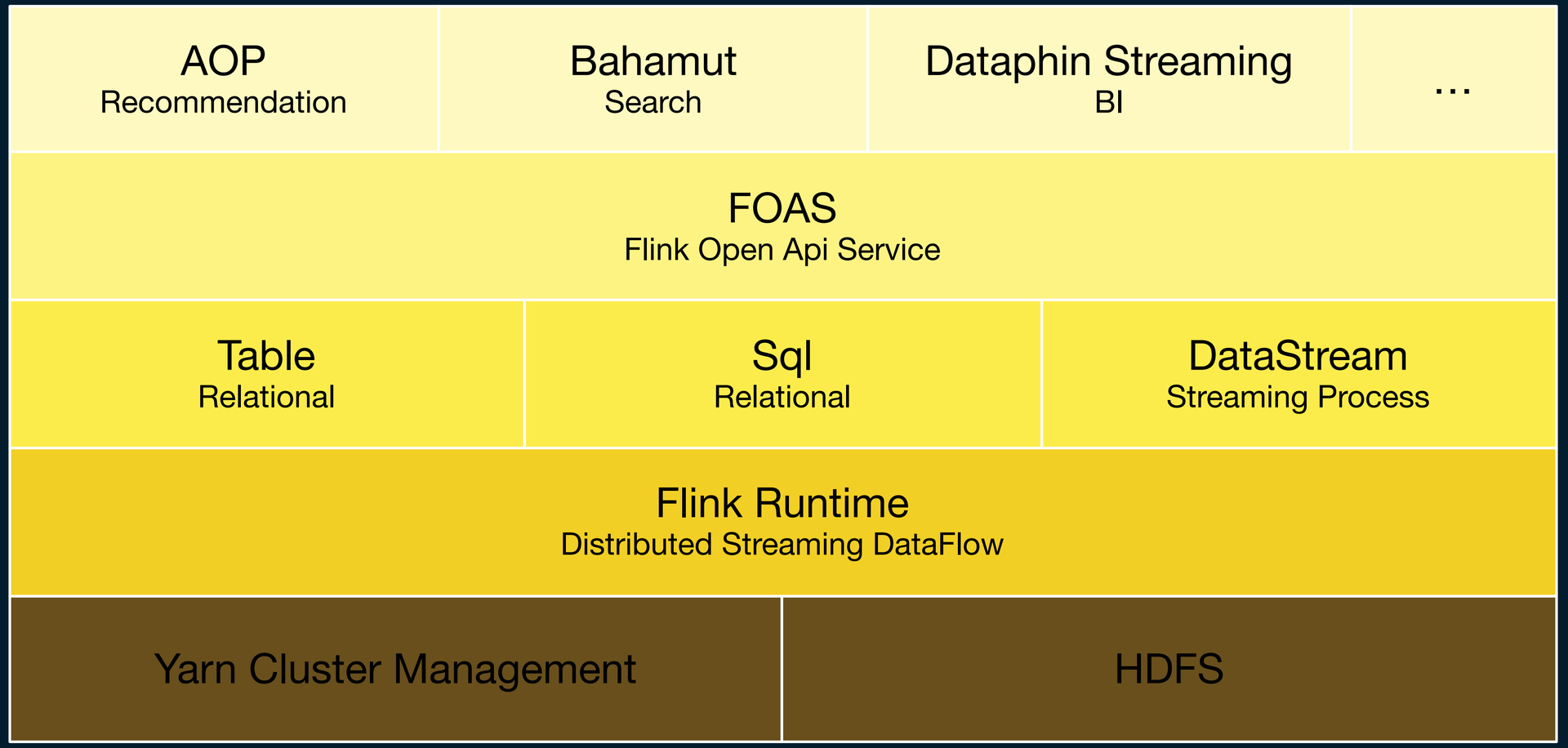
Events

Trillions/PerDay

TPS

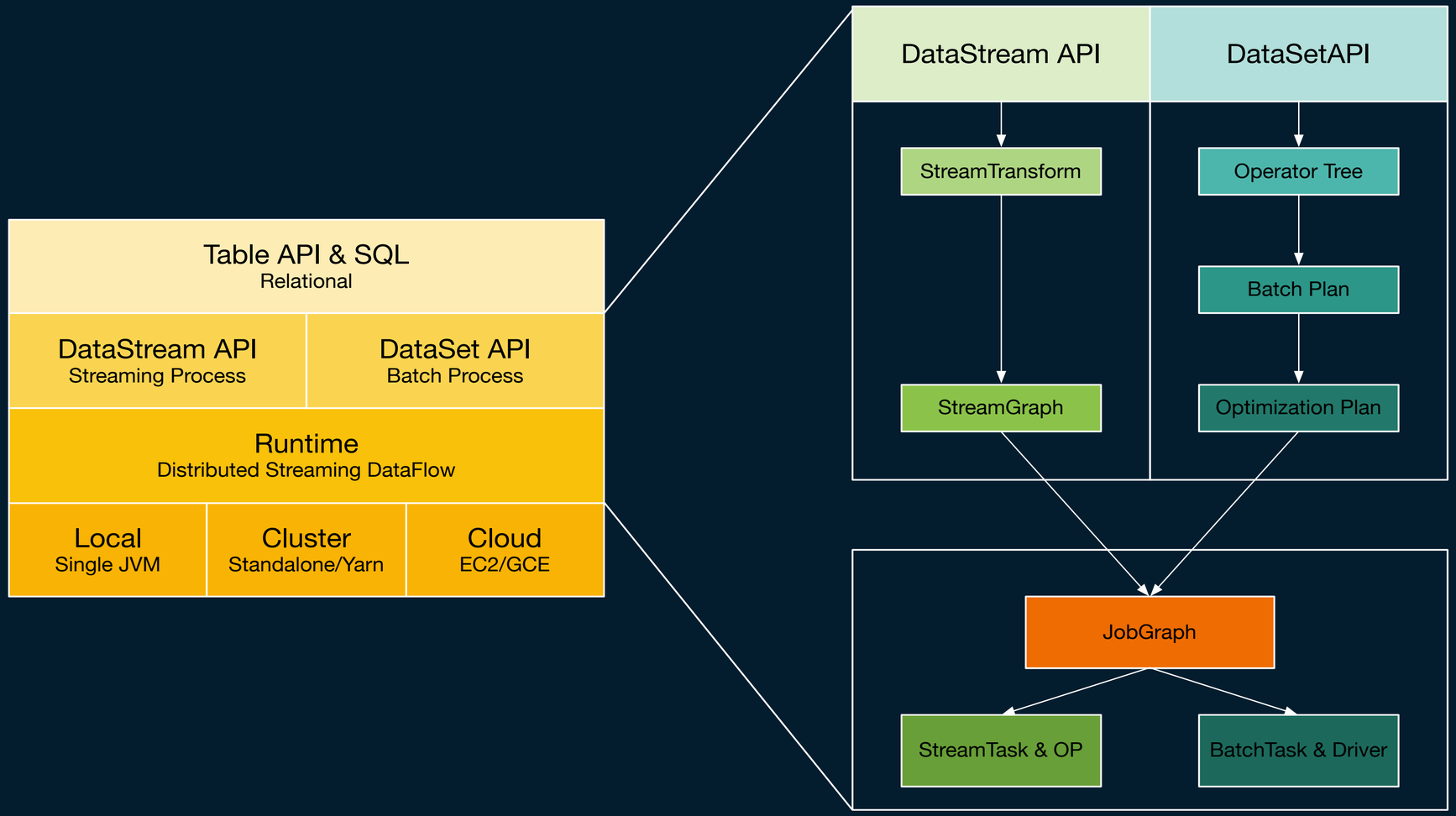
472M/Sec

Flink Computing Platform in Alibaba



- **New Flink API Stack For Unified Processing**

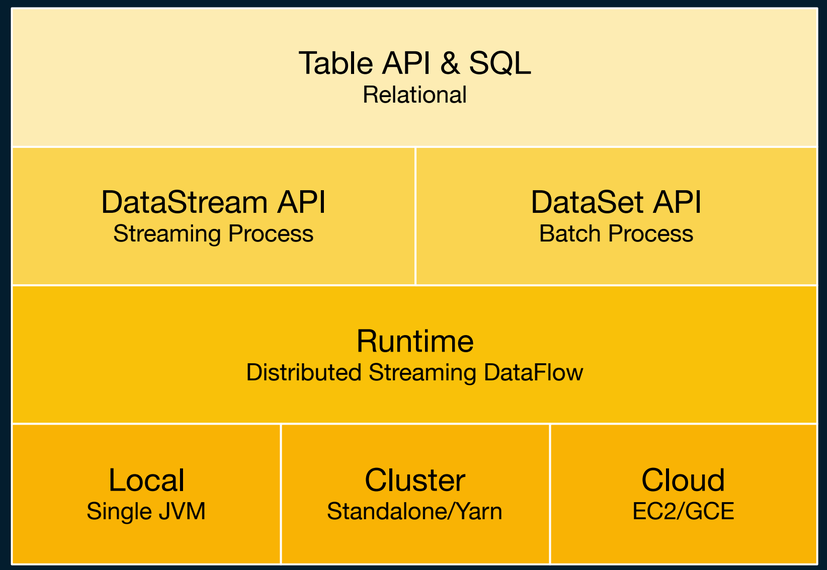
Current Flink API Stack



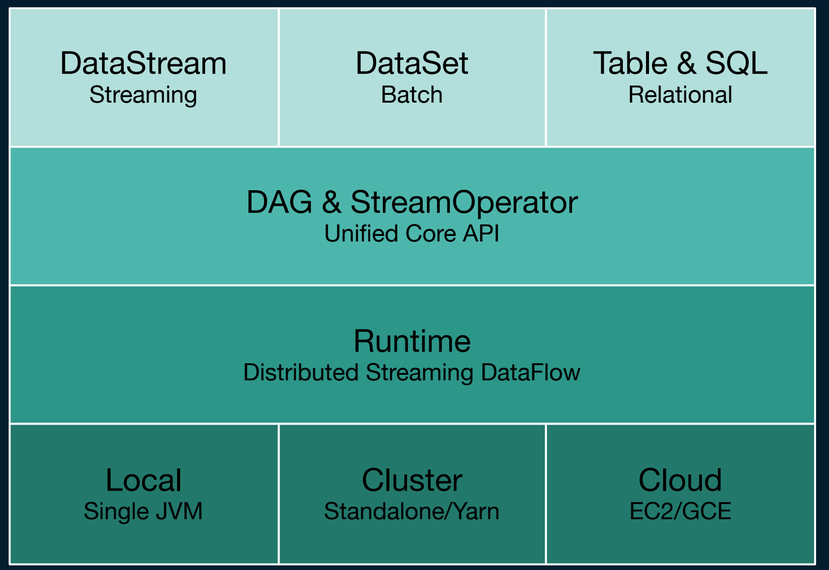
Proposed New Flink API Stack

FLINK FORWARD

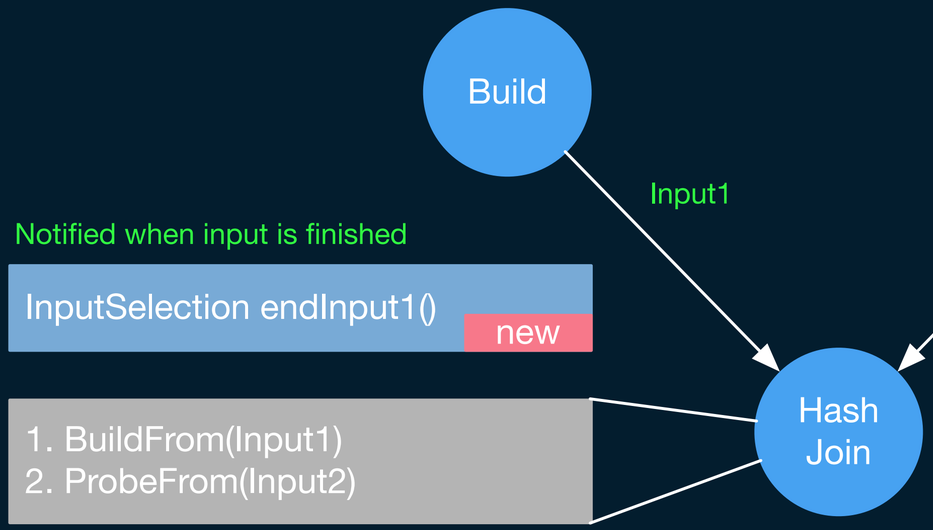
OLD



NEW



Proposed New StreamOperator API



Notified when input is finished

InputSelection endInput1()
new

1. BuildFrom(Input1)
2. ProbeFrom(Input2)

Notified when input is finished

InputSelection endInput2()
new

InputSeclection firstInputSelection()
new

Which Input side to read first ?

```
enum InputSelection {
    Any;
    First;
    Second;}
```
new

Which Input side to process next ?

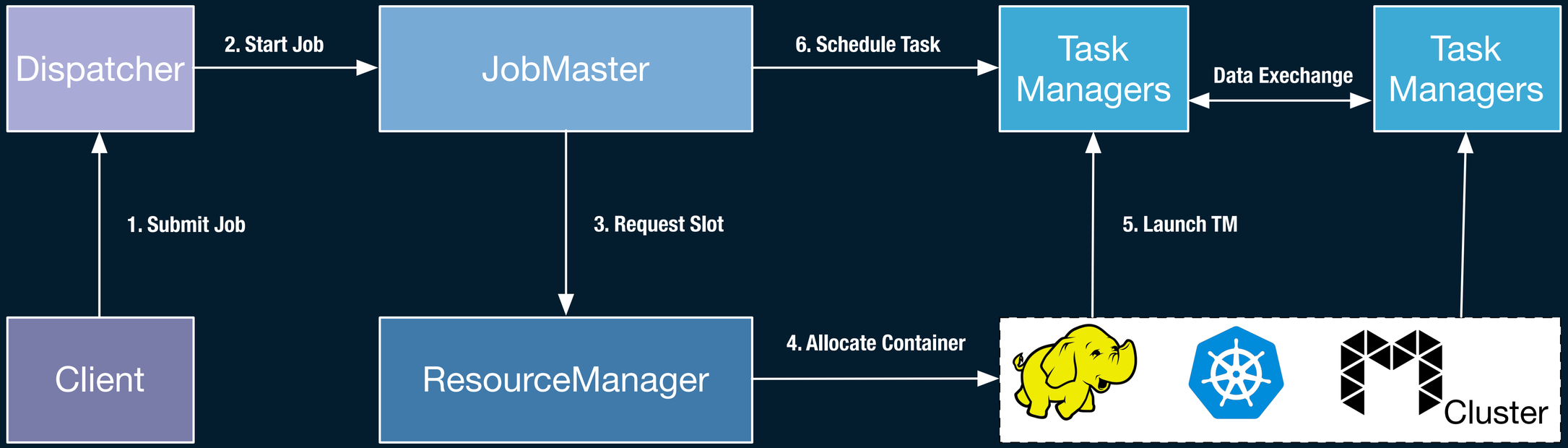
```
void processElement1(StreamRecord)
void processElement2(StreamRecord)
```
old

```
InputSelection processElement1(StreamRecord)
InputSelection processElement2(StreamRecord)
```
new

- Flink Runtime Improvements
- Flink 引擎的改进
  - Flink Architecture
  - Flink 整体架构
  - Improvements For Job Scheduler
  - 任务调度的改进
  - Pluggable Shuffle Service Architecture
  - 可插拔的数据传输服务架构
  - Improvements For Task Execution
  - 任务执行的改进
  - Improvements For Job FailOver
  - 任务容错的改进

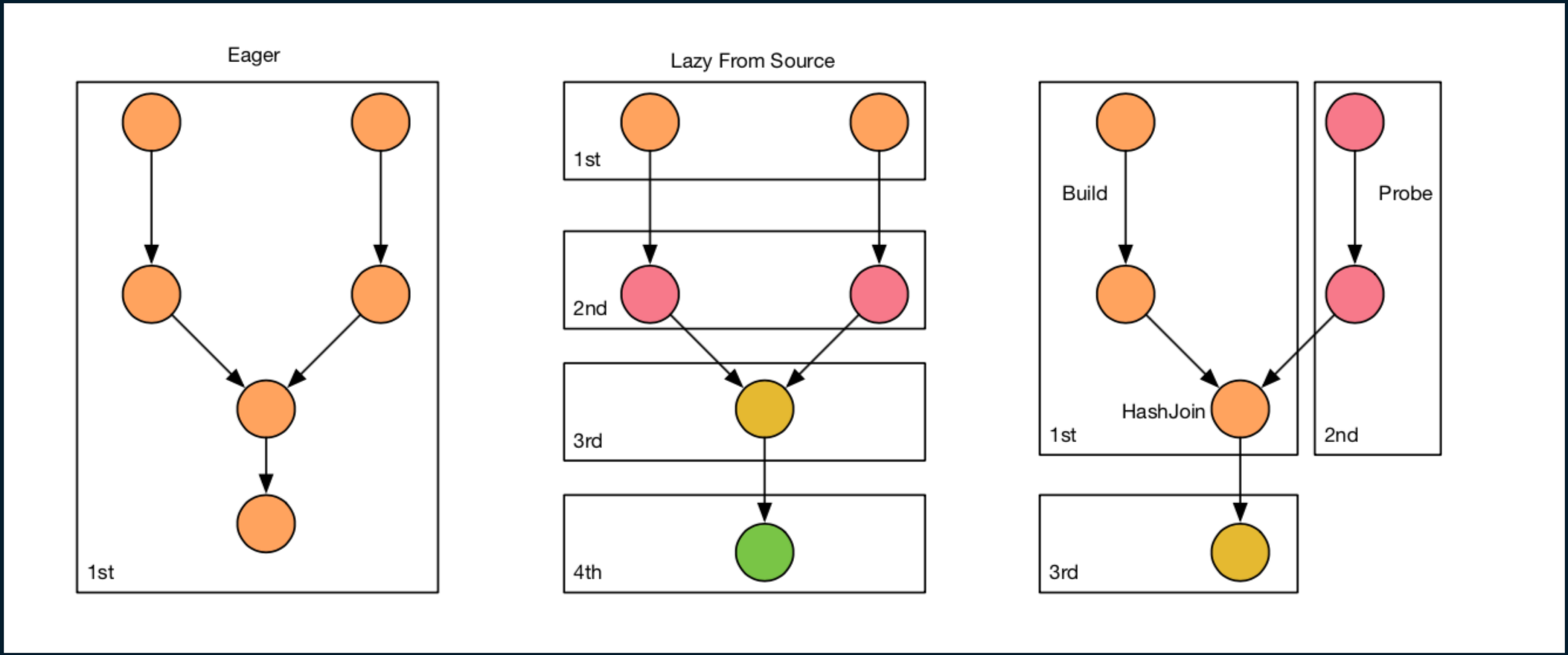
- **Flink Architecture**

Flink Architecture

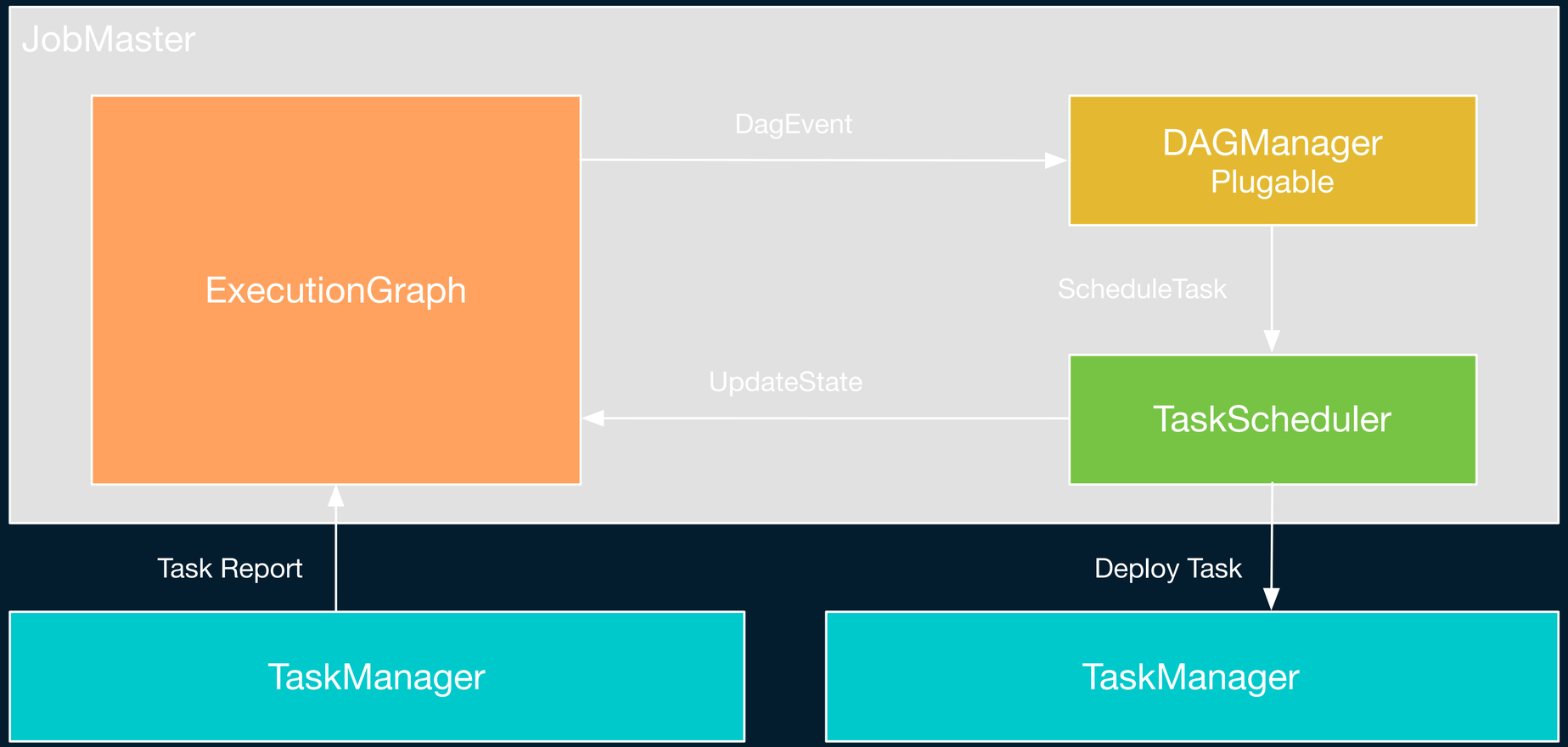


- **Job Scheduler**

Current Schedule Mode

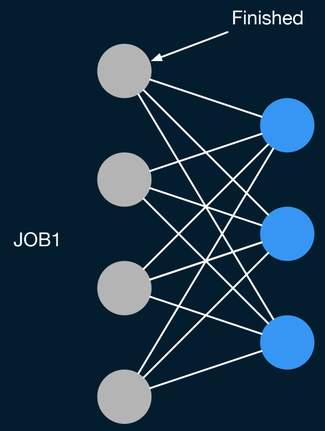


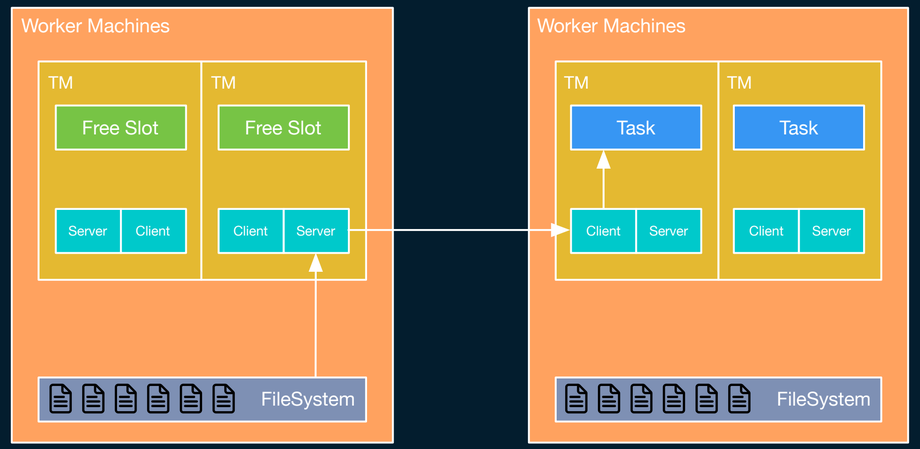
Pluggable Job Schedule Framework



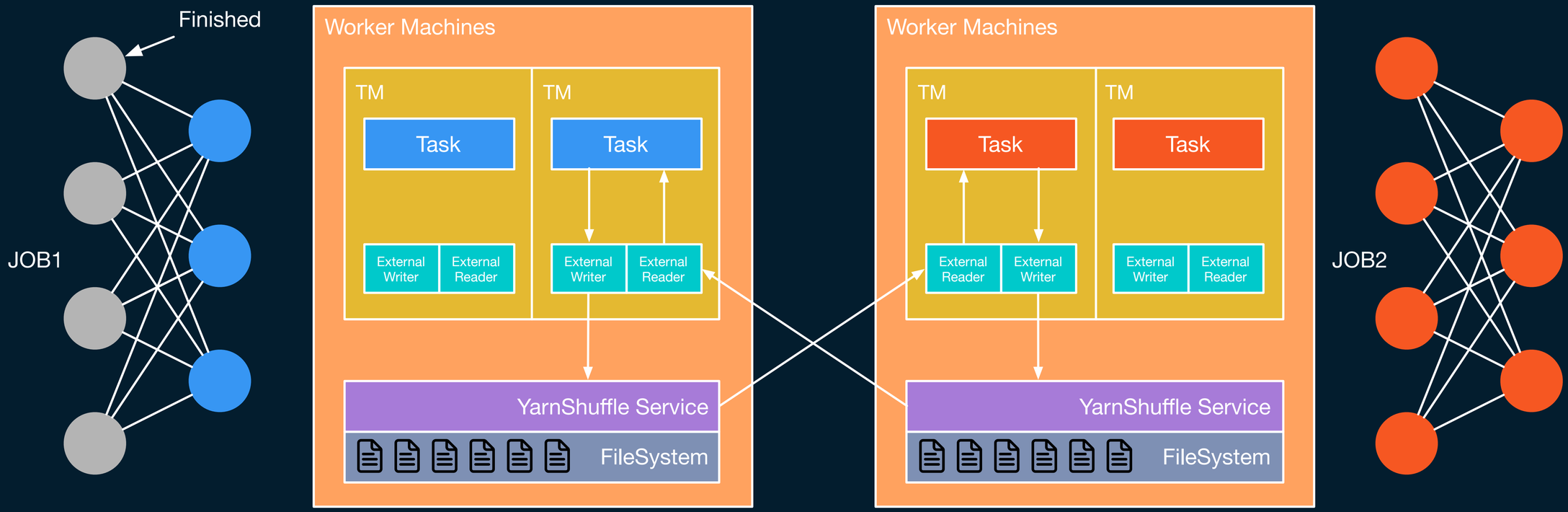
- **Pluggable Shuffle Service**

Current Shuffle Service

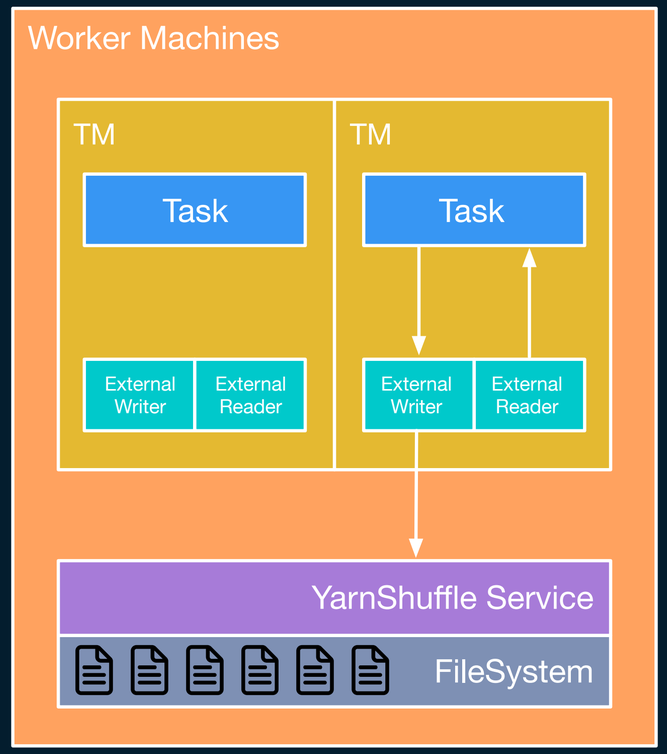


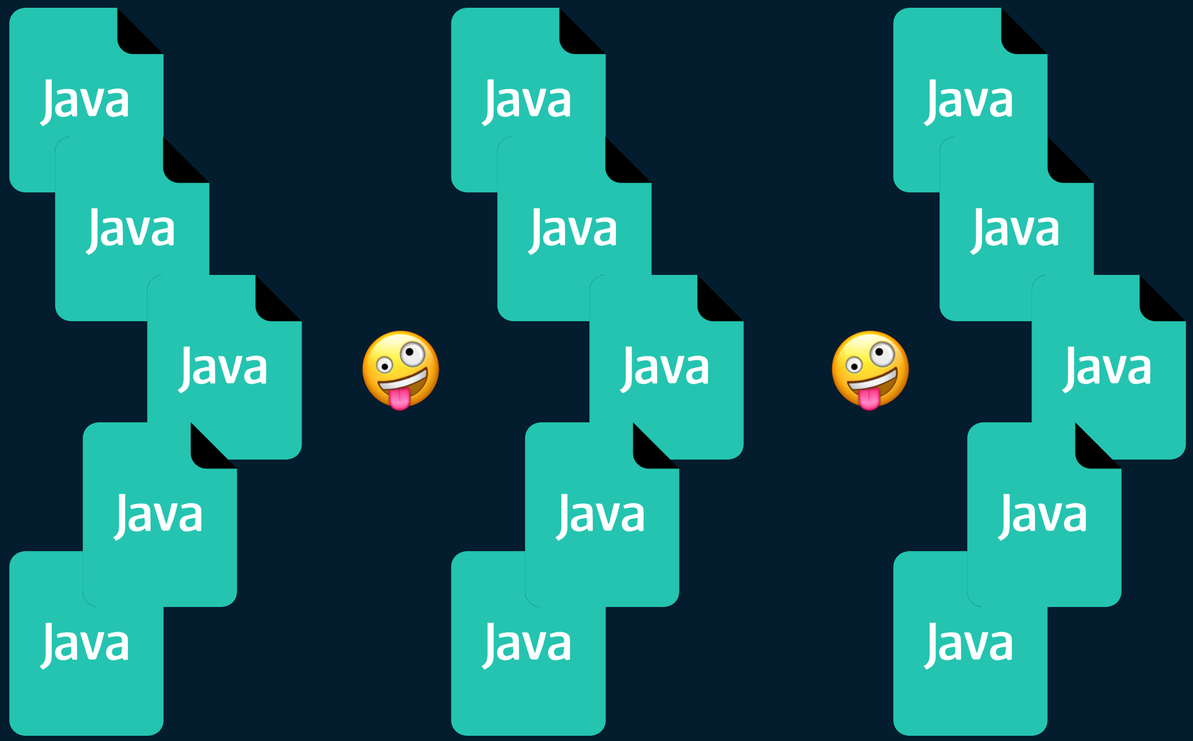


External Yarn Shuffle Service

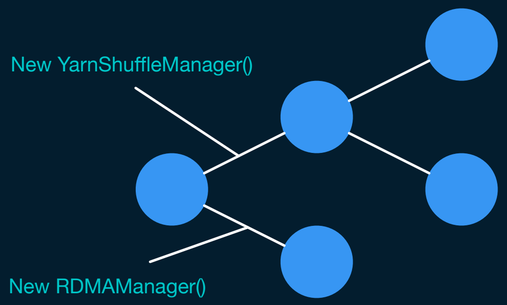


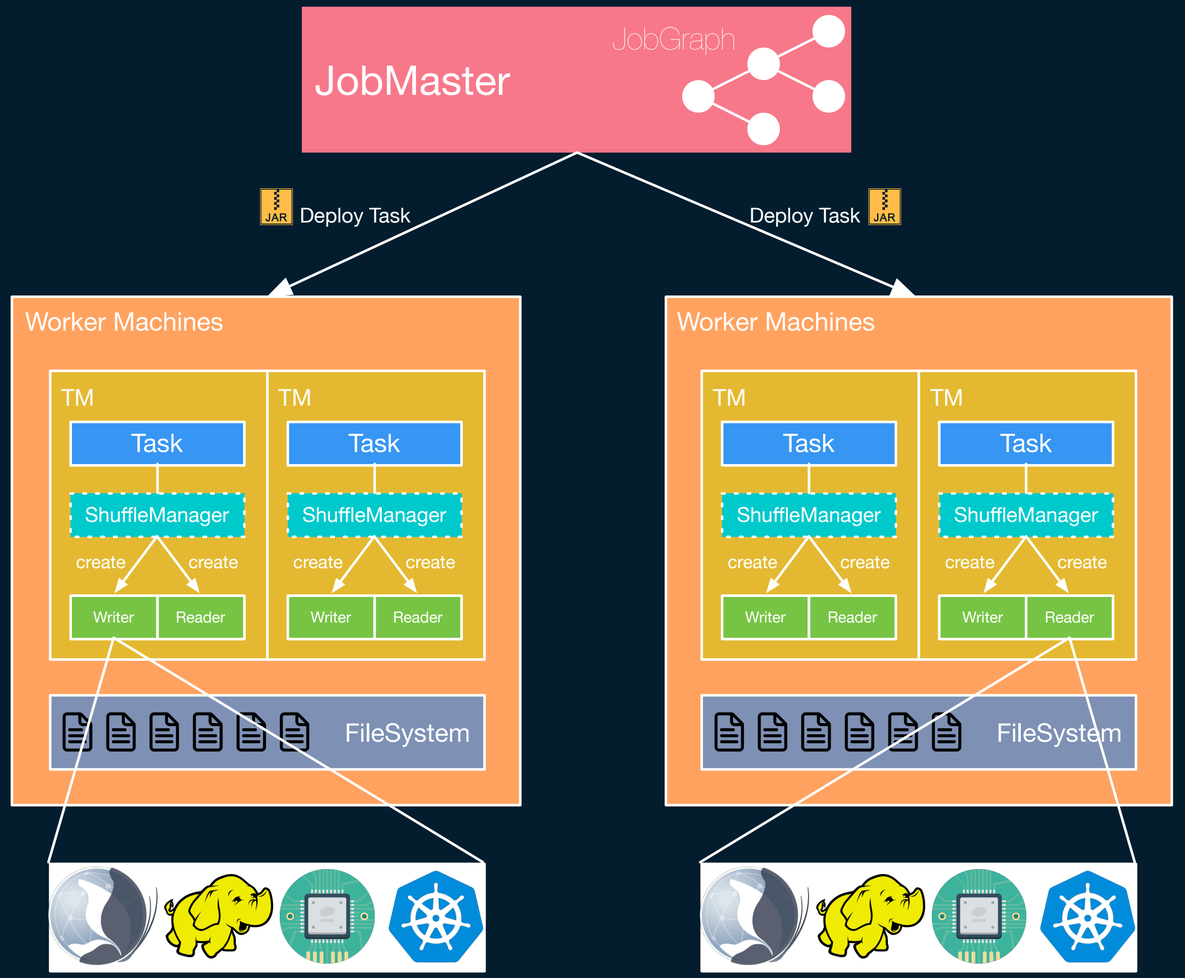
Hard to Extend New Shuffle Service





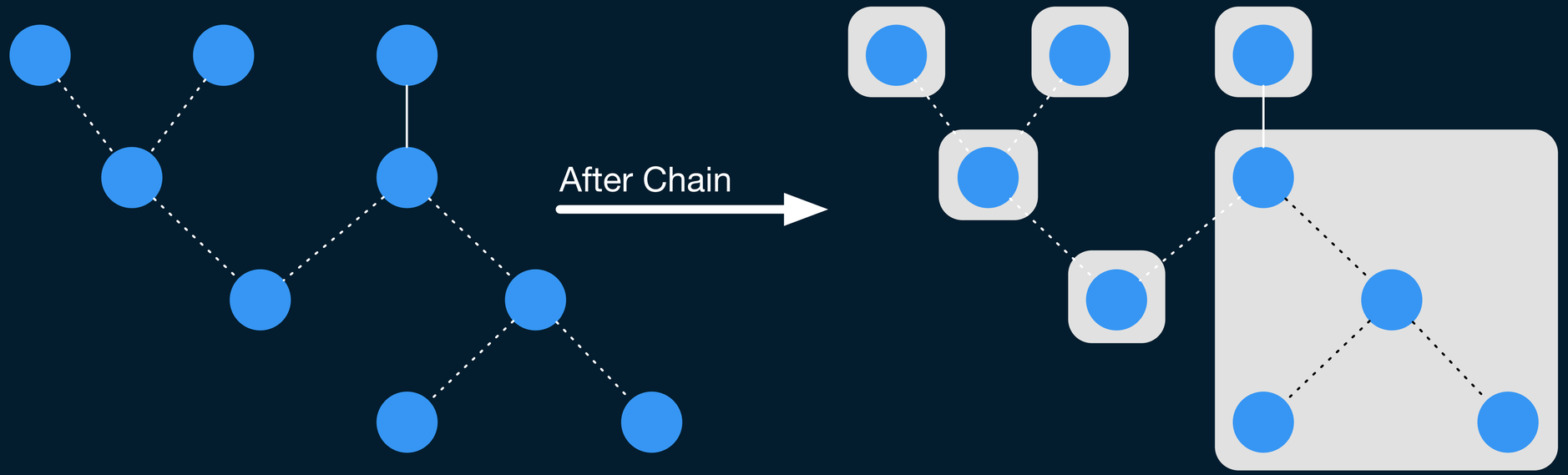
Pluggable Shuffle Architecture

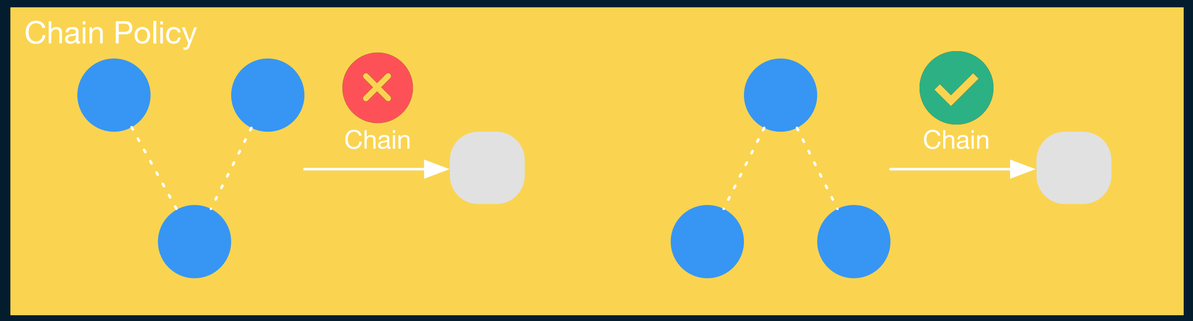




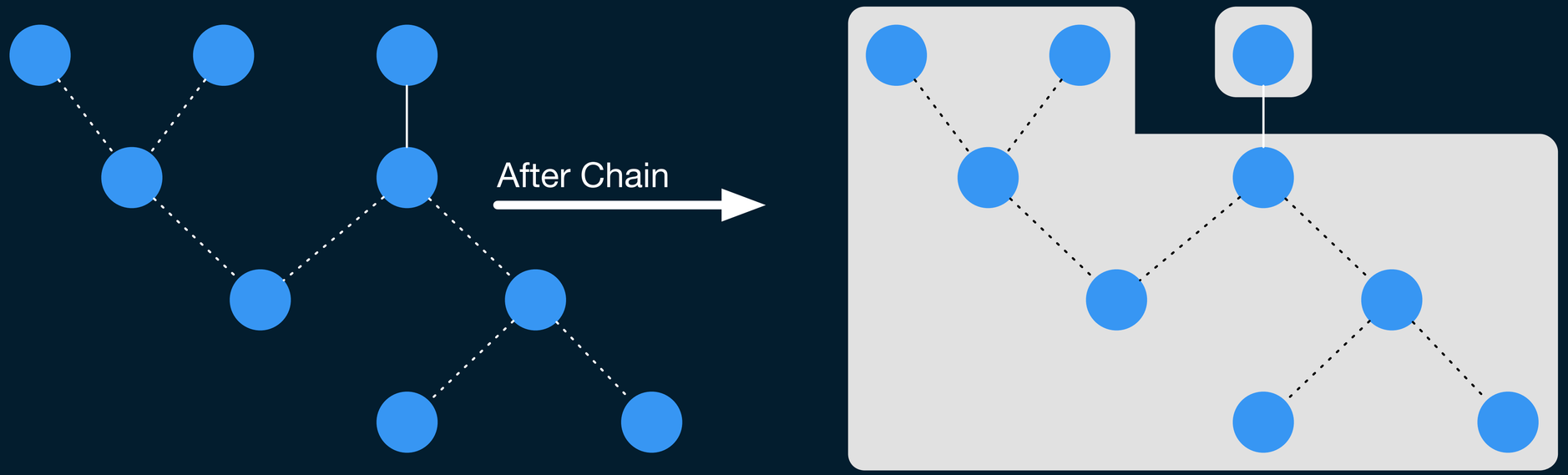
- **Task Execution**

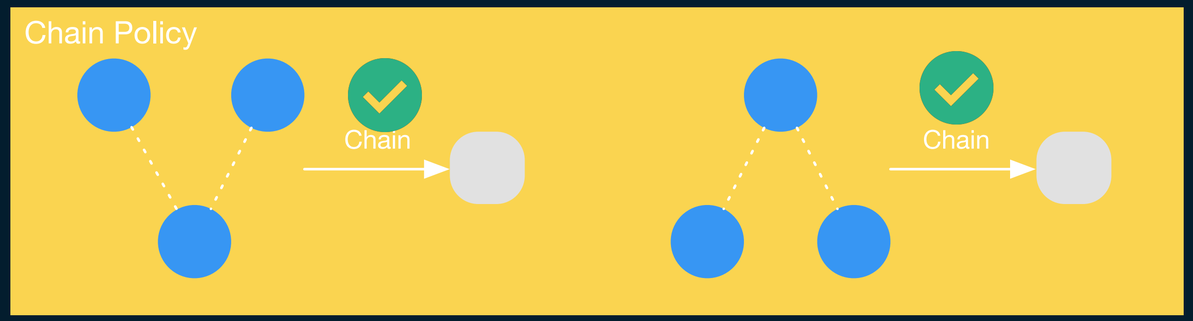
StreamTask Based on OperatorTree





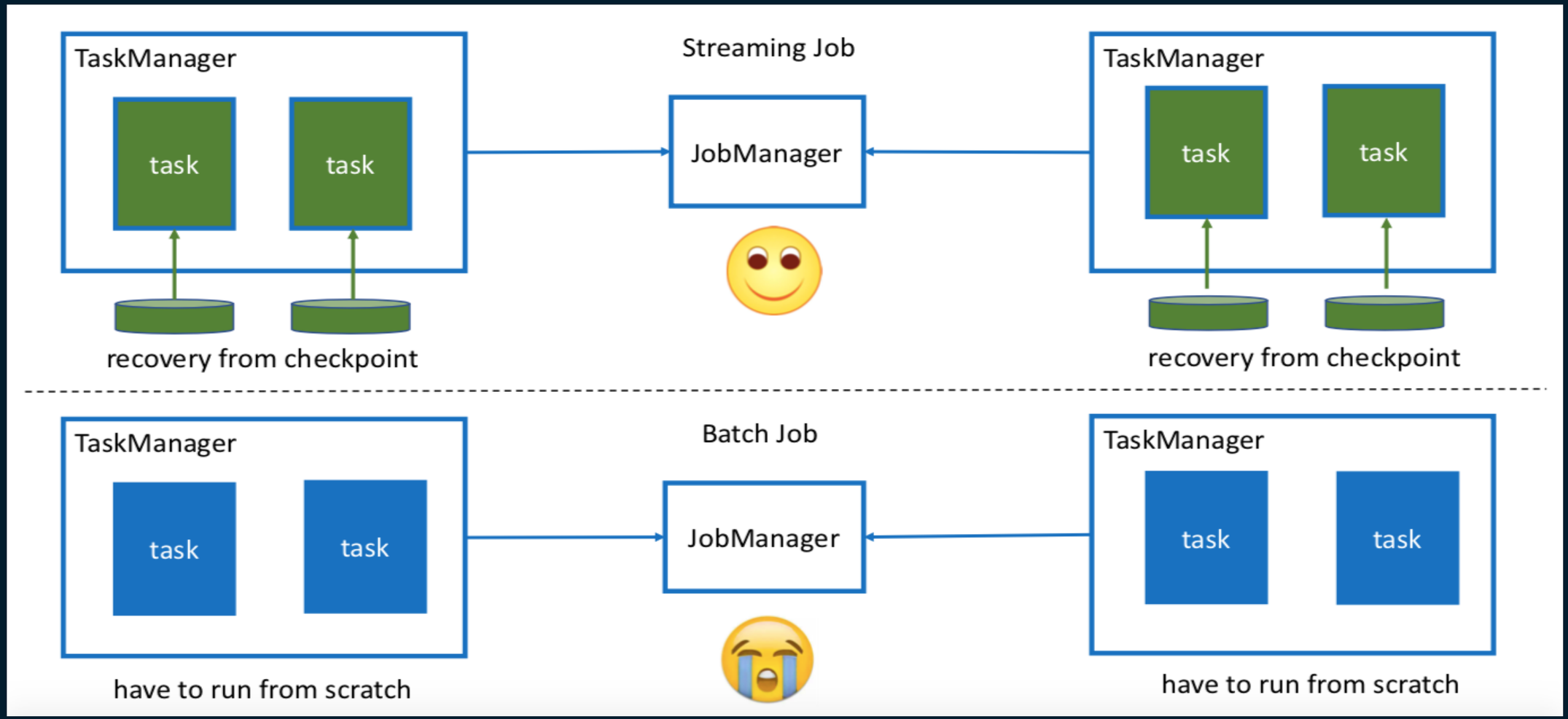
StreamTask Based on OperatorDag



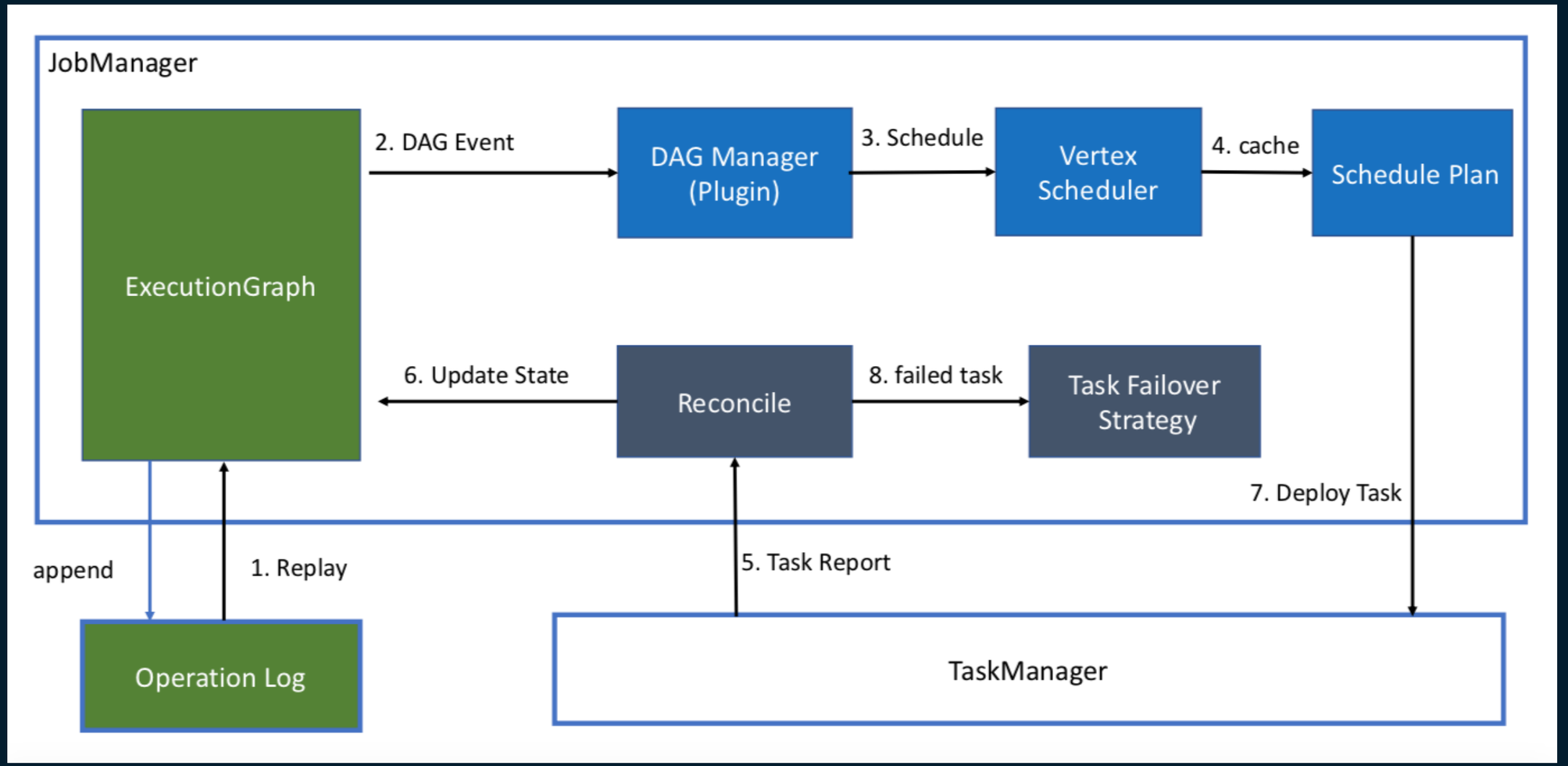


- **Job FailOver**

Current JM FailOver - Restart All



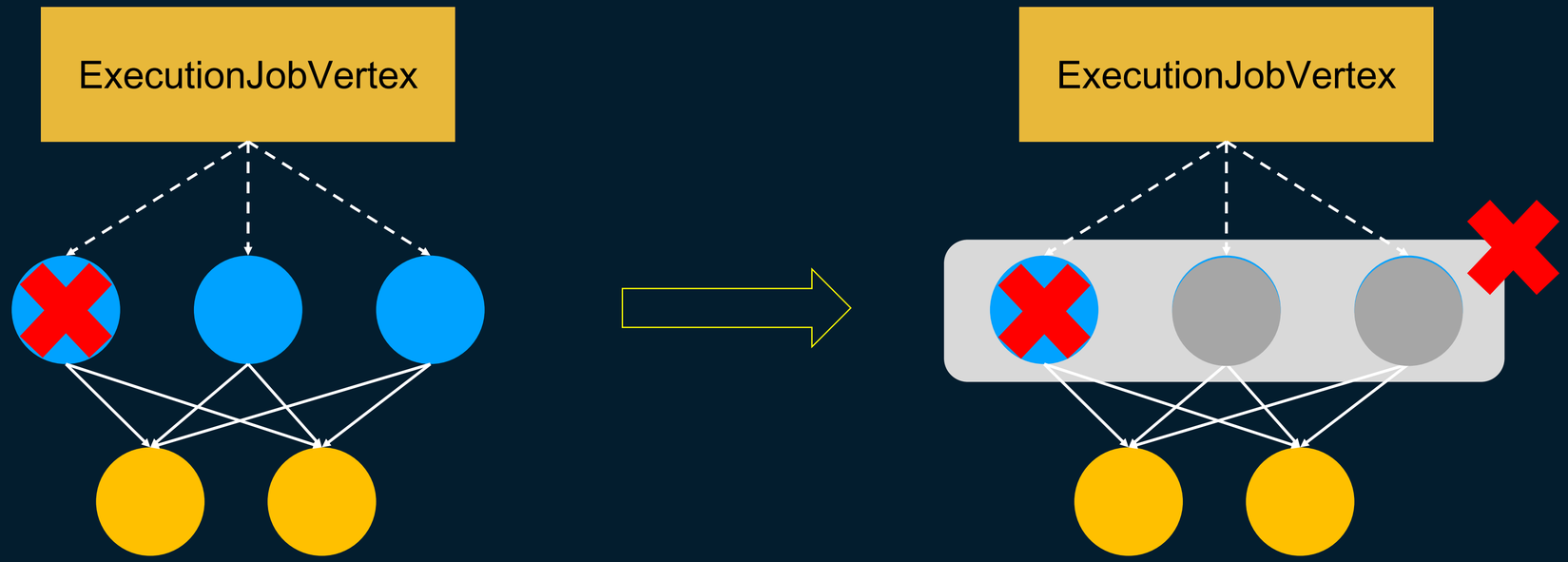
JM FailOver - Take Over



- **Failover Improvements**

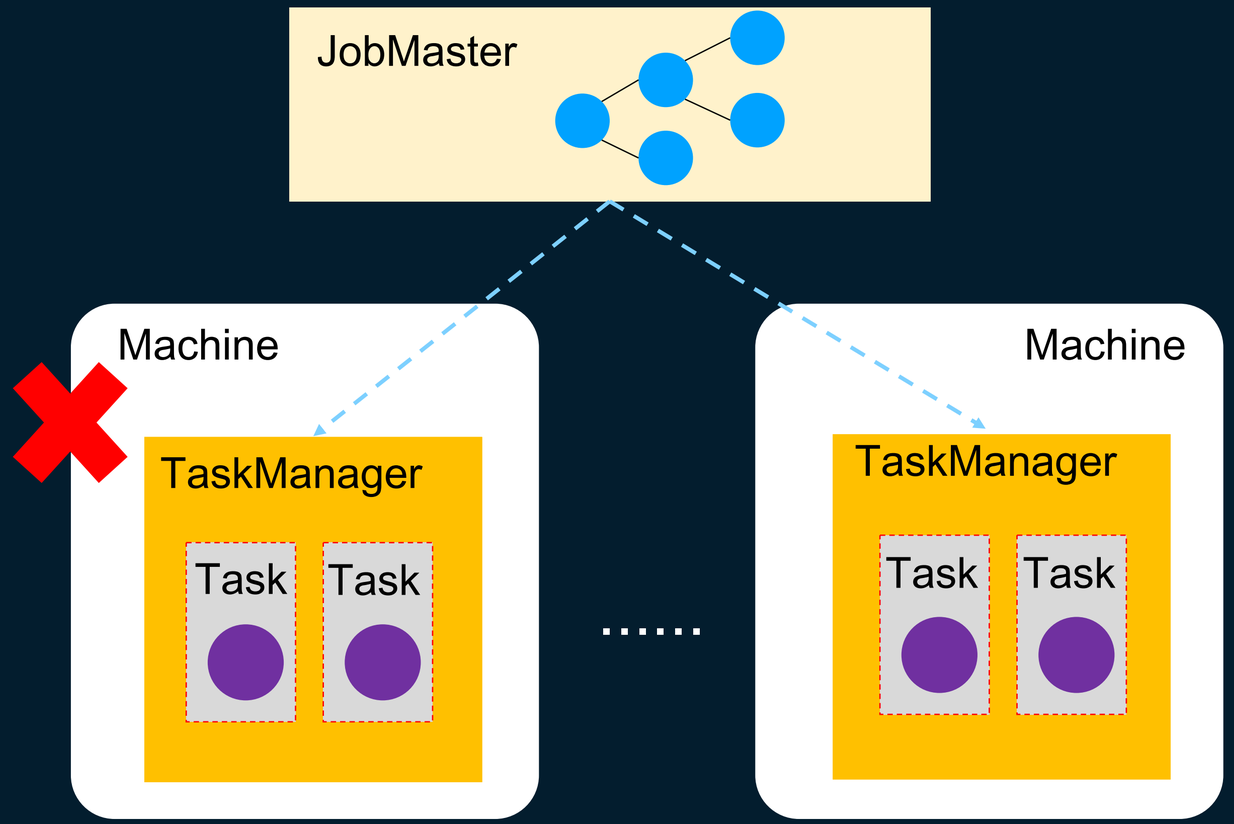
Failover Improvements [容错改进]

- Some failures are non-recoverable, such as DividebyZero where task can immediately fail.
- 有些运行时异常错误是不可恢复的例如被0除，这些情况task可以快速fail而不用多次重试.



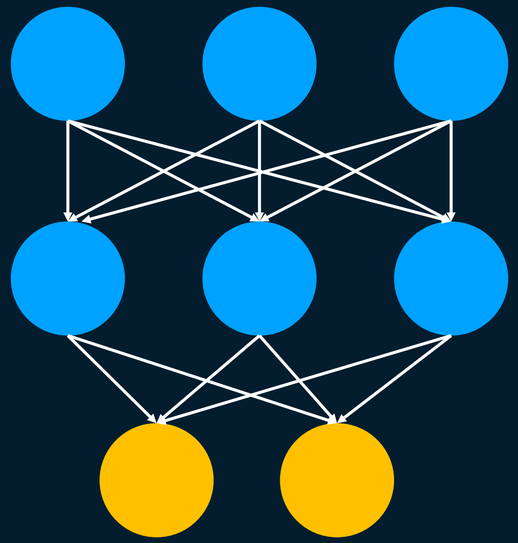
Failover Improvements [容错改进]

- There are some machines with hardware issues that we could detect and avoid to schedule to that machines.
- 有些机器可能存在硬件故障，我们可以检测出病在调度时规避.

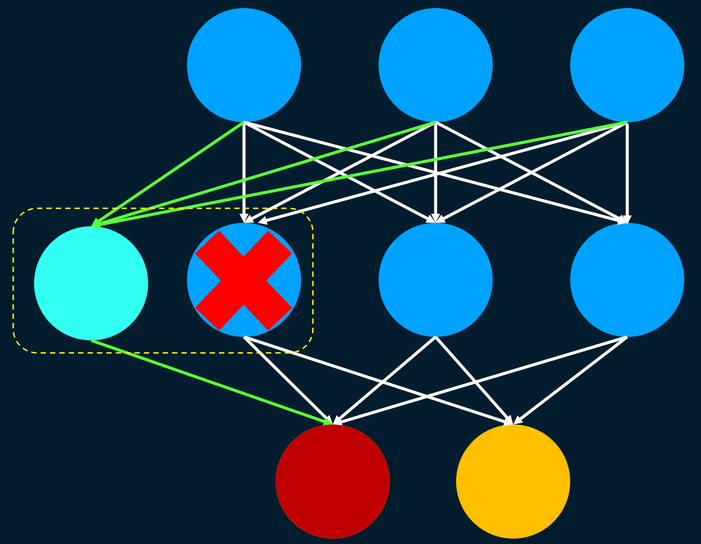


Failover Improvements [容错改进]

- We could rerun upstream vertex to re-generate data for missing task inputs..



- 当发现task的输入数据有部分丢失时，我们通过重新运行对应的上游节点，重新生成数据.



Failover Improvements [容错改进]

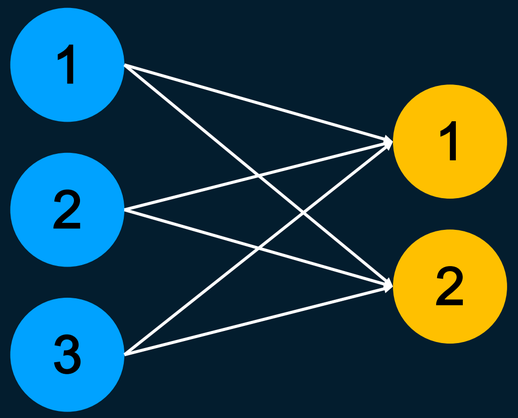
- Umbrella JIRA: FLINK-10288
  - Classify Exceptions to different categories for different strategies. [FLINK-10289]
  - Enable Per-job level failover strategy. [FLINK-10572]
  - Support task revocation. [FLINK-10573]
  - Pluggable failover strategy – can handle machine hardware issue

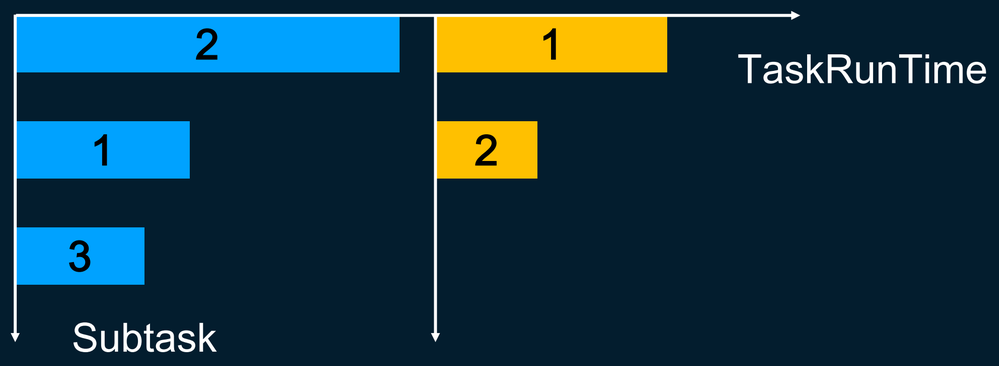
- 容错改进总体JIRA：FLINK–10288
  – 将运行出现的异常分成不同的类型，以便于识别不同的错误而采取不同的容错方法.
  – 支持job级别而不是集群cluster级别地配置容错方法.
  – 支持任务重运行，用在数据丢失时可以重运行上游任务重新生成数据.
  – 可插拔的容错机制，比如用来处理机器硬件问题.

- **Speculative Execution**

Speculative Execution [推测执行]

- Hardware problems such as accident I/O busy or high CPU load can cause the running tasks to be long tail, which thus can slow down Job running time.
- 机器硬件问题例如突发IO busy或者CPU load飙高都会导致其上运行的task速度变慢，称之为长尾，会拖慢整个作业的运行时间.

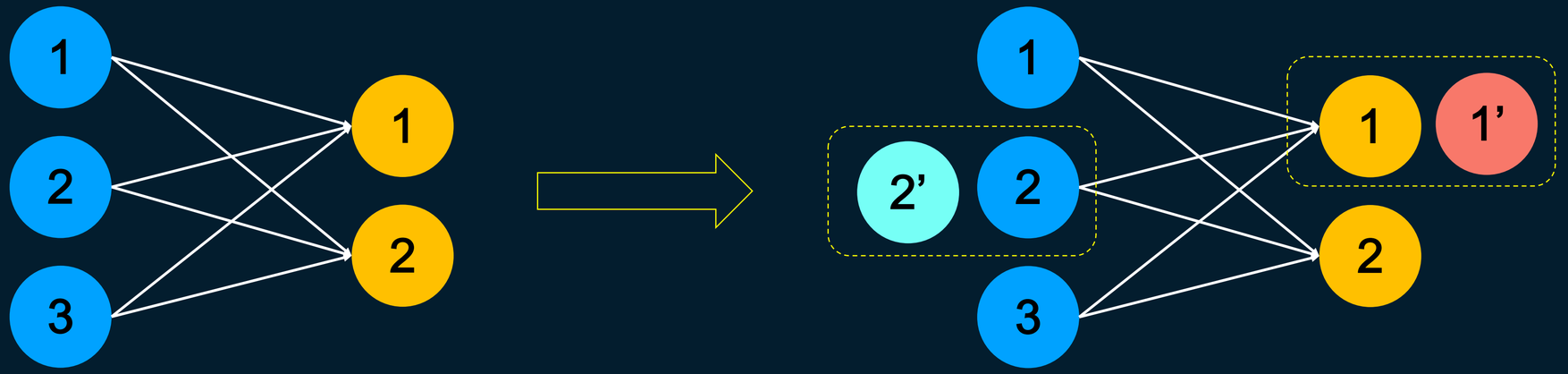




Speculative Execution [推测执行]

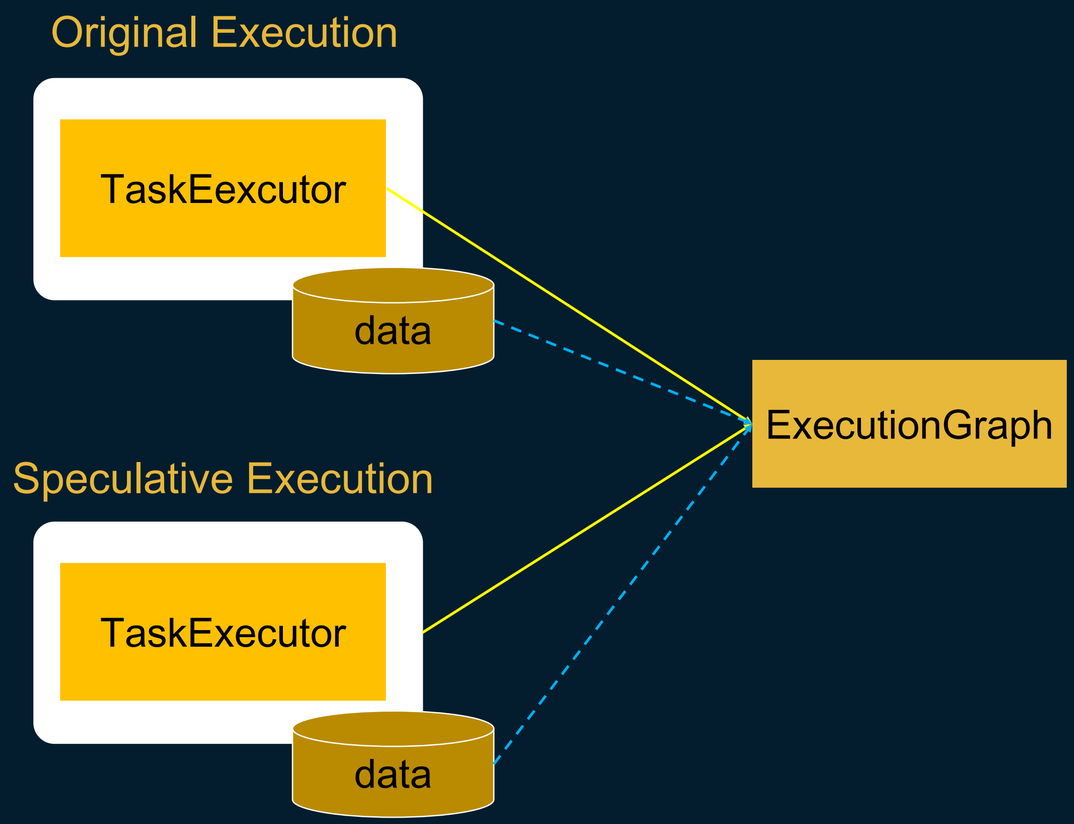
- We propose the speculative execution to handle the problem whose basic idea is to run a copy of task on another machine, and take the output result of which finishes first. [FLINK-10644]

- 我们提出推测执行的方法来解决这个问题，基本思想在另一台机器上同时运行一份备份task，谁先结束取谁的输出结果.



Speculative Execution [推测执行]

- Workflow
  - TaskExecutors collect and report running statistics.
  - JobMaster detects long tail task.
  - Schedule speculative execution attempt.
  - Commit output data.
- 工作流程
  - Task Executor负责收集运行时的统计指标并上报.
  - Job Master负责检测发现长尾的task.
  - 然后调度启动备份task attempt.
  - task的两份attempt会向Job Master提交结果数据，尽早完成者成功.



Speculative Execution [推测执行]

- Collect running statistics
  - TaskExecutor accumulates total input size and currently processed data size in InputGate across all InputChannel, and report to JobMaster by heartbeatPayload.
  - ExecutionGraph extends ExecutionVertex to require resource and maintain multiple Attempts.

- 收集运行指标
  – TaskExecutor在InputGate类中，遍历所有InputChannel，累计统计出总输入数据量，以及当前已经处理的数据量，通过心跳的payload信息上报给JobMaster.
  – ExecutionGraph则扩展了ExecutionVertex，使之可以为备份attempt申请额外的资源，并调度管理一个task的多个attempt.

Speculative Execution [推测执行]

- Decide the long tail task [决定长尾task]
  - slow processing throughput defined as

$$s_i \le \frac{1}{\alpha} * (1 - \frac{c_j}{d_i}) * S$$

[依据这个公式来定义长尾task，即当前task的处理速度明显低于其他已结束task的平均速度]

$S = \sum_i d_i / \sum_i t_i$ is average throughput of terminated tasks

α is configurable penalty factor (>=1)

[其中S是已结束task的平均处理速度，a是可配置的惩罚因子，大于1]

Speculative Execution [推测执行]

FLINK FORWARD

Speculation = true

2018-12-14, 15:52:47	2018-12-14, 15:52:47	206ms	CHAIN DataSource (at main(WordCount.java:74) (org.apache.flink.api.java.io.TextInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:84)) -> Combine (SUM(1), at main(WordCount.java:87))	0 B	46,800	0 B	48,094	2	0 0 0 2 0 0 0	FINISHED
2018-12-14, 15:52:47	2018-12-14, 15:52:48	849ms	Reduce (SUM(1), at main(WordCount.java:87))	27.4 KB	1,294	0 B	1,294	2	0 0 0 2 0 0 0	FINISHED

Subtask Index	Start Time	End Time	Duration	Bytes received	Records received	Bytes sent	Records sent	Attempt	Host	Status
1	2018-12-14, 15:52:47	2018-12-14, 15:52:48	850ms	0 B	0	0 B	0	1	rs3f13043:52790	CANCELED
1	2018-12-14, 15:52:48	2018-12-14, 15:52:48	74ms	13.6 KB	628	0 B	628	2	rs3f13043:52790	FINISHED
2	2018-12-14, 15:52:47	2018-12-14, 15:52:48	105ms	13.9 KB	666	0 B	666	1	rs3f13043:52790	FINISHED

2018-12-14, 15:52:48	2018-12-14, 15:52:48	11ms	DataSink (collect())	27.4 KB	1,294	0 B	0	2	0 0 0 2 0 0 0	FINISHED

Speculation = false

2018-12-14, 15:42:02	2018-12-14, 15:42:03	270ms	CHAIN DataSource (at main(WordCount.java:74) (org.apache.flink.api.java.io.TextInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:84)) -> Combine (SUM(1), at main(WordCount.java:87))	0 B						
2018-12-14, 15:42:03	2018-12-14, 15:42:13	10s	Reduce (SUM(1), at main(WordCount.java:87))	27.4 K						

Subtask Index	Start Time	End Time	Duration	Bytes received	Records received					
1	2018-12-14, 15:42:03	2018-12-14, 15:42:13	10s	13.6 KB	628	0 B	628	1	rs3d07044:62070	FINISHED
2	2018-12-14, 15:42:03	2018-12-14, 15:42:03	115ms	13.9 KB	666	0 B	666	1	rs3d07044:62070	FINISHED

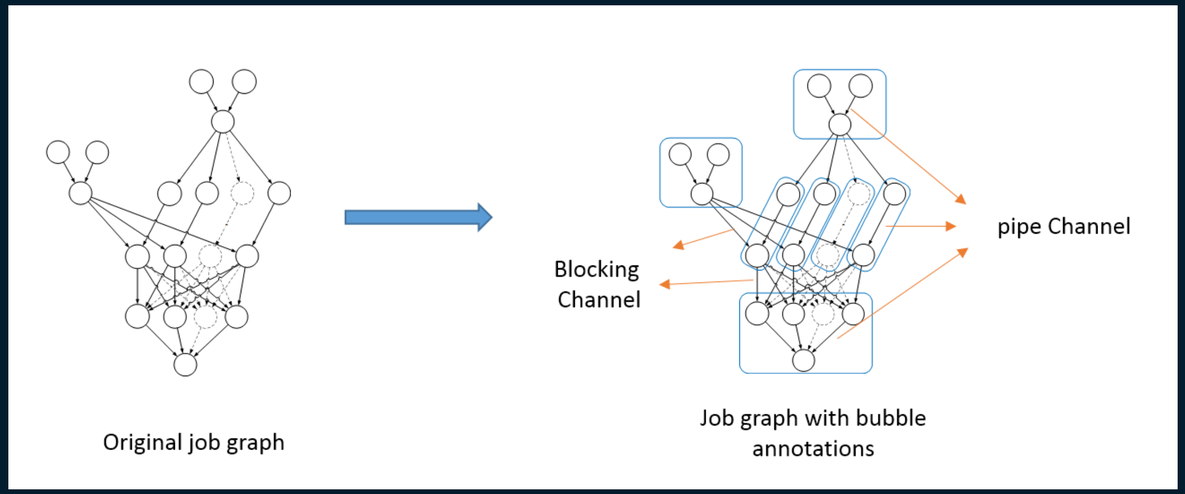
2018-12-14, 15:42:13	2018-12-14, 15:42:13	15ms	DataSink (collect())	27.4 KB	1,294	0 B	0	2	0 0 0	FINISHED

- **DAG Bubble Scheduling**

DAG Bubble Scheduling [DAG子图调度]

- Motivation
  - flexibility, efficiency, reliability.
- Prototype Design
  - divide the DAG into several sub-graphs according to edge cost function.
  - intra- pipeline, inter- blocking

- 设计目的：灵活、高效、可靠.
- 设计思想：
  - 依据边上的代价准则将DAG分成若干个小的子图.
  - 子图内部采用pipeline channel，而子图之间采用blocking channel.



DAG Bubble Scheduling [DAG子图调度]

- Prototype Test

  - TPC-H Queries can run through under limited resource budget, while may hang using the LAZY mode.

  - For TPC-H Q9, 10G data, 50 slots(half resource budget of LAZY mode) will reach 70% performance of the LAZY mode.

- 原型测试

  – TPC–H测试集可以在给定的有限资源中跑过，而通常的LAZY则会hang住.

  – 例如Q9,10G数据，50个slot，只需要LAZY模式一半的资源，性能可以达到70%.

DAG Bubble Scheduling [DAG子图调度]

- Bubble Generation [子图的生成过程]

```
Input: Slot budget, JobVertexes sorted topologically
Output: Bubbles
/* Traverse job vertexes topologically */
foreach job vertex do
    if (has no pipelined inputs) then
        Build a new bubble;
        continue;
    end
    foreach input do
        if (input's source result type is pipelined) and
            (slot budget constraint) then
            Merge the producer and consumer bubbles;
        end
    end
end
```

Fig. Bubble Generation Algorithm

- **Future Plans**

Future Plans

- Deployment 部署
  - Unified Elastic Session For Flip6
  - 统一动态session
- Job Schedule 任务调度
  - Dynamic Update of JobGraph for Batch Job
  - 为批任务动态更新Job拓扑
  - HotUpdate of JobGraph for Streaming Job
  - 为流任务热升级Job拓扑
- Network Stack 网络栈
  - More kinds of external Shuffle Service
  - 更多类型的外部数据传输服务
  - RDMA Shuffle
  - 基于RDMA的数据传输服务
- Checkpoint & StateBackend 检查点和状态管理
  - New kind of Heap-Based StateBackend
  - 一种新的基于堆的状态管理
  - Checkpoint for Batching Process
  - 为批处理增加检查点


THANKS