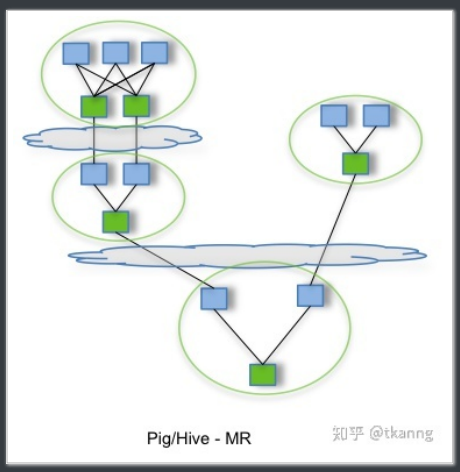
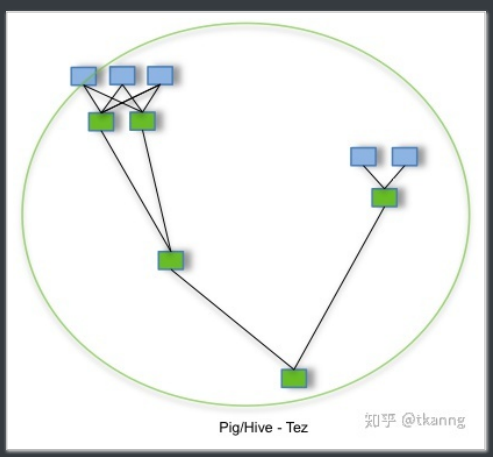
深入剖析 Tez 原理

一、产生背景

- MR 性能差，资源消耗大，如：Hive 作业之间的数据不是直接流动的，而是借助 HDFS 作为共享数据存储系统，即一个作业将处理好的数据写入 HDFS，下一个作业再从 HDFS 重新读取数据进行处理。很明显更高效的方式是，第一个作业直接将数据传递给下游作业。





- MR 默认了 map 和 reduce 阶段，map 会对中间结果进行分区、排序，reduce 会进行合并排序，这一过程并不适用于所有场景。

- 引擎级别的 Runtime 优化：MR 执行计划在编译时已经确定，无法动态调整(?)。然而在执行 ETL 和 Ad-hoc 等任务时，根据实际处理的表大小，动态调整 join 策略、任务并行度将大大缩短任务执行时间。

二、原理

2.1 DAG

https://hortonworks.com/blog/expressing-data-processing-in-apache-tez/

– Vertex： 定义了用户逻辑（如：map/reduce）与相关的资源与环境

- Edge：定义了上下游 Vertex 之间的连接方式。
- Edge 相关属性：
  - Data movement：定义了 producer 与 consumer 之间数据流动的方式。
    One-To-One: 第 i 个 producer 产生的数据，发送给第 i 个 consumer。这种上下游关系属于 Spark 的窄依赖。
    Broadcast: producer 产生的数据路由都下游所有 consumer。这种上下游关系也属于 Spark 的窄依赖。
    Scatter-Gather: producer 将产生的数据分块，将第 i 块数据发送到第 i 个 consumer。这种上下游关系属于 Spark 的宽依赖。

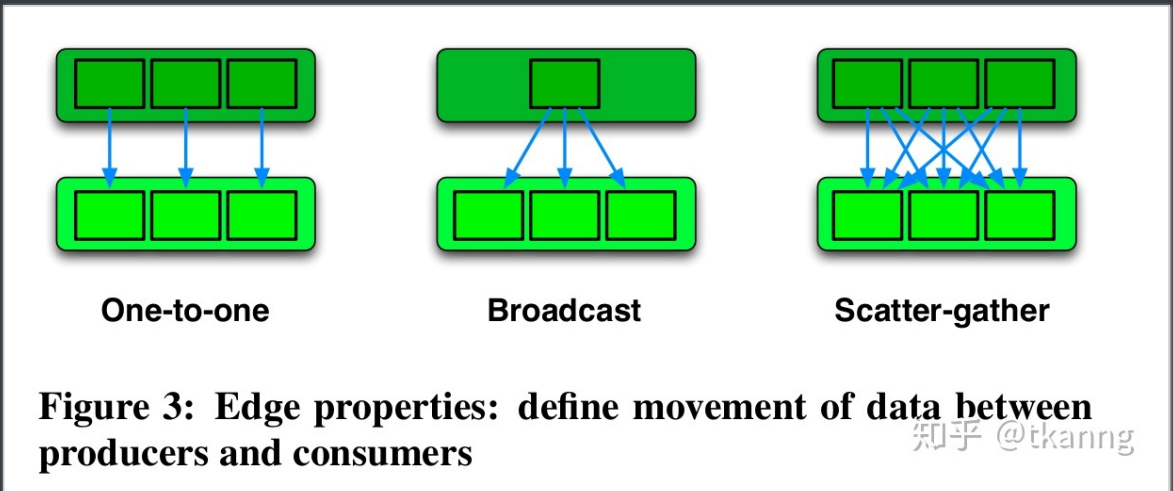


Figure 3: Edge properties: define movement of data between producers and consumers

  - Scheduling：定义了何时启动 consumer Task
    Sequential: Consumer task 需要 producer task 结束后启动，如：MR。
    Concurrent: Consumer task 与 producer task 一起启动，如：流计算。
  - Data source：定义了任务 outp 的生命周期与可靠性。
    Persisted: 当任务退出后，该任务 output 依然存在，但经过一段时间后，可能会被删除，如：Mapper 输出的中间结果。
    Persisted-Reliable: 任务 output 总是存在，比如，MR 中 reducer 的输出结果，存在 HDFS 上。
    Ephemeral: 任务输出只有当该 task 在运行的时候，才存在，如：流计算的中间结果。

- 举例——MapReduce 在 Tez 的编程模型

  一个 DAG 图中只有两个 Vertex，Map Vertex 与 Reduce Vertex。连接 Map Vertex 与 Reduce Vertex 的 Edge 有以下属性： Data movement：Scatter-Gather Scheduling：Sequential Data Source： Map Vertex 的 Data Source 为 Persisted-Reliable， reduce Vertex 的 Data Source 为 Persisted

- Tez Api 实现 WordCount

```
1: DAG dag = DAG.create("WordCount");

2: Vertex tokenizerVertex = Vertex.create("Tokenizer", TokenProcessor.class)
                            .addDataSource("Input", HdfsInitializer.class);

3: Vertex summationVertex = Vertex.create("Summation", SumProcessor.class)
                            .addDataSink("Output", HdfsCommitter.class);

4: EdgeProperty edgeProperty = EdgeProperty.create(scatter_gather,
                           KeyValueShuffleWriter.class, KeyValueShuffleReader.class);

5: dag.addVertex(tokenizerVertex).addVertex(summationVertex)
       .addEdge(Edge.create(tokenizerVertex, summationVertex, edgeProperty);
```

Figure 4: Essence of the Tez API shown via pseudo-code for the canonical WordCount example

2.2 Runtime API——Input/Processor/Output

Task 是 Tez 的最小执行单元，Vertex 中 task 的数量与该 vertex 的并行度一致。以下是 Input、Processor、Output 均需要实现的接口：

```
List<Event> initialize(Tez*Context) -This is where I/P/O receive their corresponding

handleEvents(List<Event> events) - Any events generated for the specific I/P/O will b

List<Event> close() - Any cleanup or final commits will typically be implemented in t
```

- Input: 接收上游 Output 事件，获取上游数据位置；从 physical Edge 中获取实际数据；解析实际数据，为 Processor 提供统一的逻辑试图；
- Processor: 利用 Input 获取实际数据，执行用户逻辑，最后输出；
- Output: 将 Processor 提供的数据，进行分区；向下游 Input 发送事件；

- Tez 的事件驱动机制: Tez 中各个组件通过不同类型的 Event 进行通信。
  - 数据传输：Output 通过 ShuffleEvent 传递上游数据位置，AM 负责将 Event 路由到相应 Input 中。
  - 容错：Input 当无法获取到上游数据时，会通知框架重新调度上游任务，这也意味着任务成功完成后，仍然会被重新调度。
  - runtime 执行计划优化：根据上游 Map Stage 产生的数据大小，动态 reducer 并行度。Output 产生的事件路由到可拔插的 Vertex/Edge management module，对应 moudule 就可以对 runtime 执行计划进行调整。

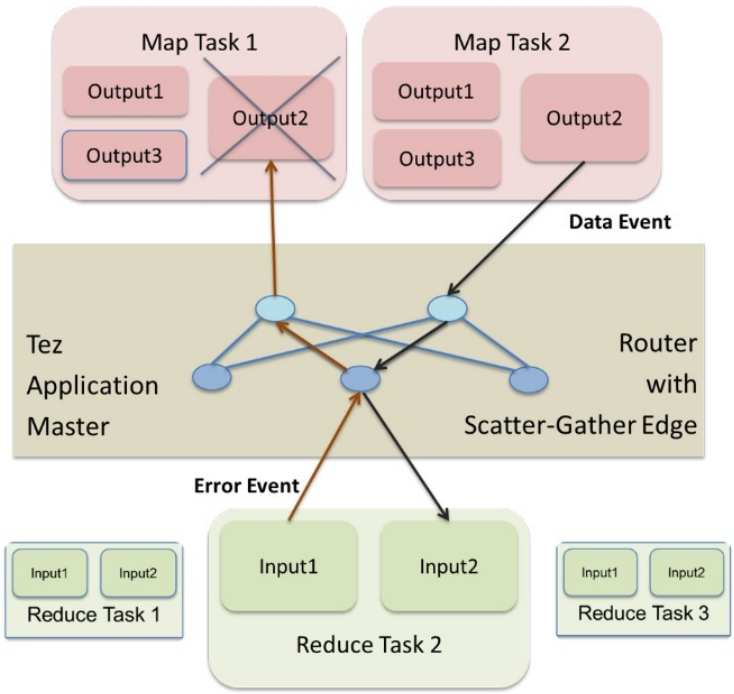
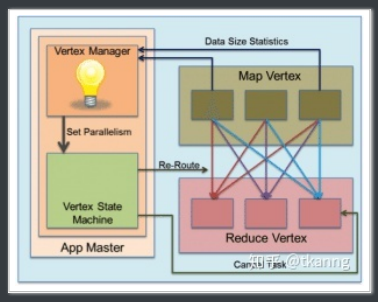


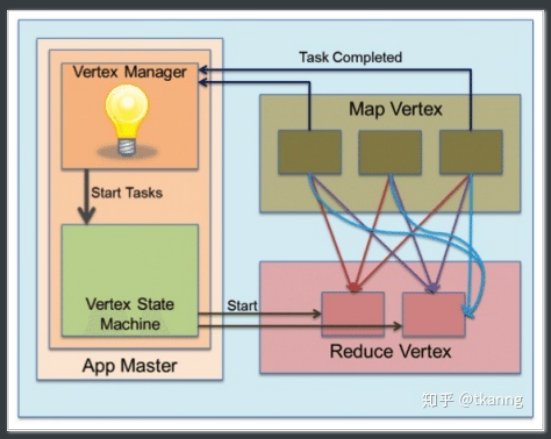
Figure 5: Events used to route metadata between application IOs and error notifications from applications to the framework

2.3 Runtime 优化

任务运行时，程序知晓更多任务相关的信息，通过这些信息，我们可以动态修改修改执行计划，比如：修改 mapper 或 reducer 数量，决定何时启动 reducer 等。在 Tez 中，不同组件通过不同事件类型，进行通信。

- 动态修改 reducer 并行度：MapTask 通过 VertexManager 类型的事件向 ShuffleVertextManager 发送信息，比如: 所处理的 partition 大小等。ShuffleVertexManager 通过所获得的信息，可以估算出所有 Task 的输出数据大小，最后来调整下游 reduce Vertex 的并行度，如下图：





- reducer"慢" 启动 (预先启动)： 上游 MapTask 通过事件不断向 ShuffleVertexManager 汇报任务完成情况，ShuffleVertexManager 通过这些信息，可以判断何时启动下游 reduceTask 与需要启动的 reduceTask 数量。

2.4 从逻辑执行计划到物理执行计划

- 从逻辑 DAG 到最后物理执行计划示意图：

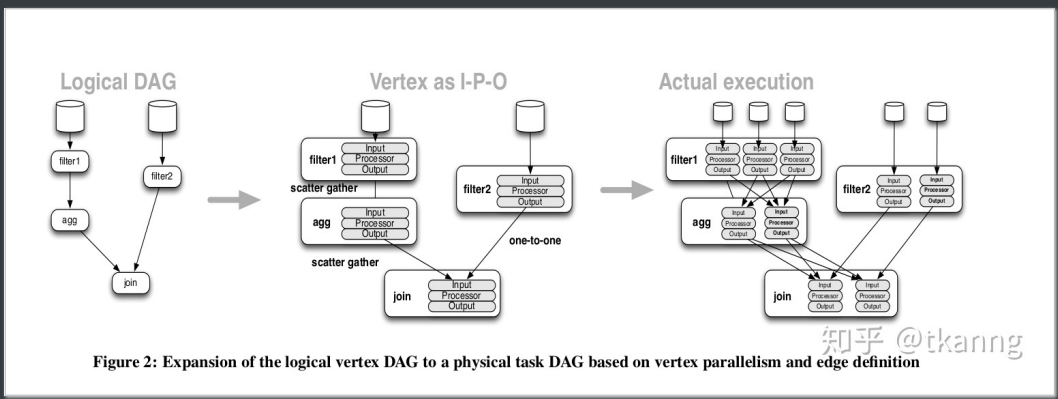
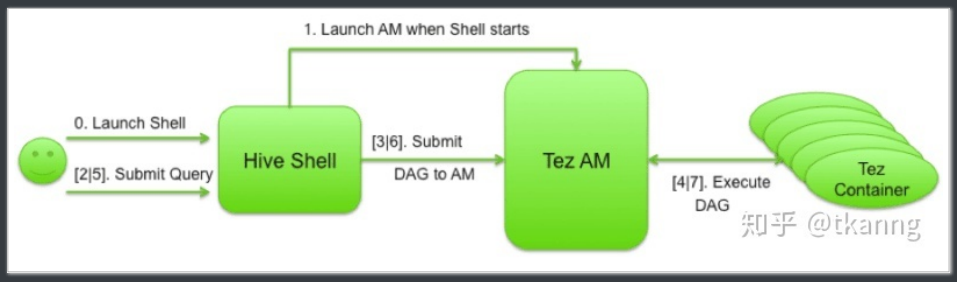


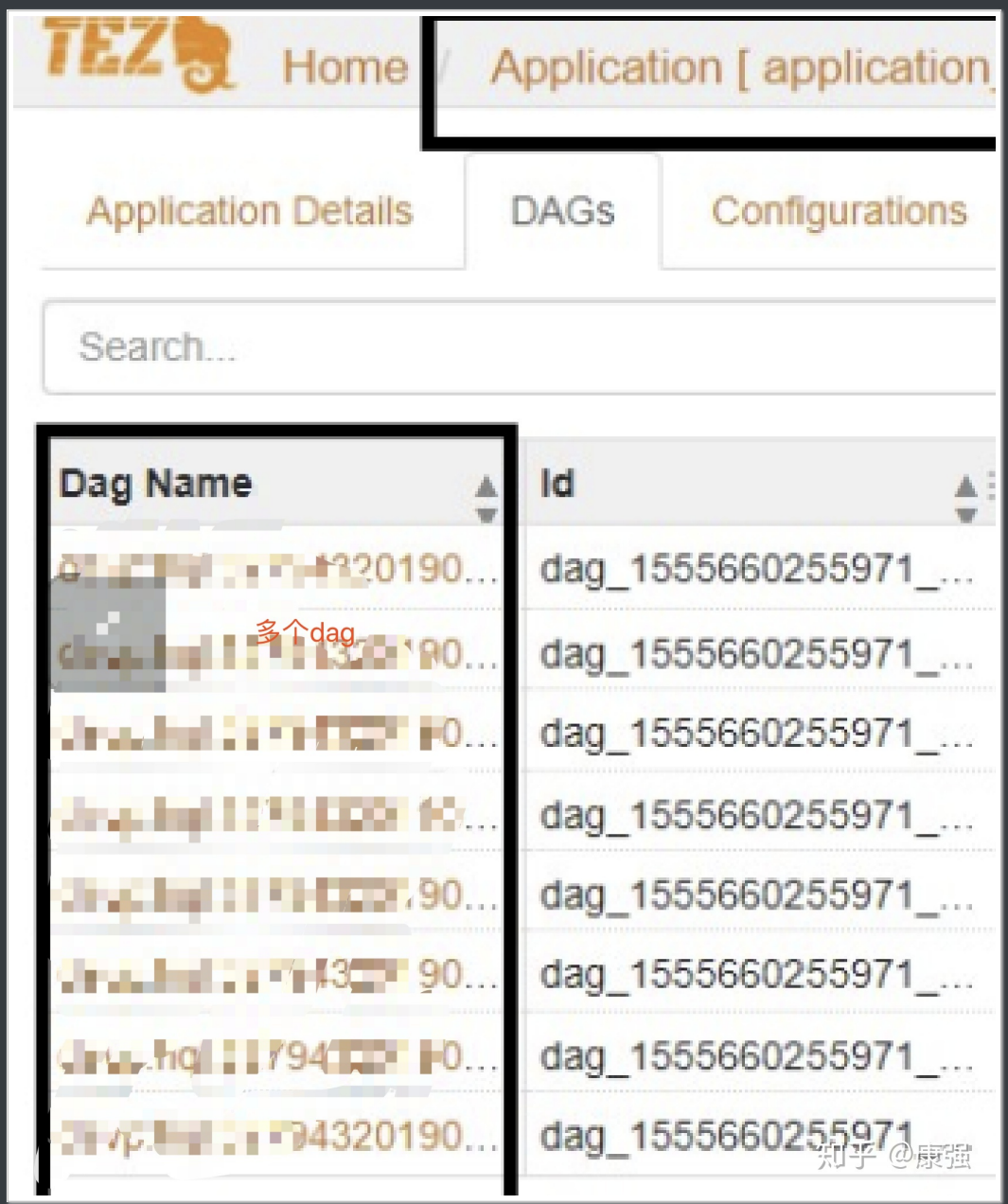
Figure 2: Expansion of the logical vertex DAG to a physical task DAG based on vertex parallelism and edge definition

2.4 其他优化措施

- Tez Session: 与数据库 session 相似，在同一个 Tez Session 中，可串行执行多个 Tez Dag。Tez Session 避免了 AM 的多次启动与销毁，在有多个 DAG 图的 Tez 作业（HQL 任务）中大大减小了任务执行时间。



这也是为什么在 Tez-UI 中，一个 HQL 任务，**只有一个 Application**，却有**多个 DAG**(MR 中一个 HQL 任务，有多个 Application)。



Tez 相关参数：

tez.am.session.min.held-containers	0	Int value. The minimum number of containers that will be held in session mode. Not active in non-session mode. Enables an idle session (not running any DAG) to hold on to a minimum number of containers to provide fast response times for the next DAG.
tez.am.mode.session	false	Boolean value. Execution mode for the Tez application. True implies session mode. If the client code is written according to best practices then the same code can execute in either mode based on this configuration. Session mode is more aggressive in reserving execution resources and is typically used for interactive applications where multiple DAGs are submitted in quick succession by the same user. For long running applications, one-off executions, batch jobs etc non-session mode is recommended. If session mode is enabled then container reuse is recommended.

tez.session.am.dag.submit.timeout.secs	300	Int value. Time (in seconds) for which the Tez AM should wait for a DAG to be submitted before shutting down. Only relevant in session mode.
tez.session.client.timeout.secs	120	Int value. Time (in seconds) to wait for AM to come up when trying to submit a DAG from the client. Only relevant in session mode. If the cluster is busy and cannot launch the AM then this timeout may be hit. In those case, using non-session mode is recommended if applicable. Otherwise increase the timeout (set to -1 for infinity. Not recommended)

- Container 复用

问题：

- container 的资源兼容？ 被先后调度到同一个 container 的多个 task 所需要的资源，必须与 container 的资源相互兼容。也就是说，container 拥有的资源，如：jar 包，Memory，CPU 等，需要是 task 所需资源的“超集”。
- 怎么调度？ 进行 container 复用时，Tez 对 Task 进行调度。Tez 会依据：任务本地性、任务所需资源、pending 任务的优先级等因素，进行任务调度。

优点：

- 减少作业执行过程中 JVM 的创建与销毁带来的开销
- 减小对 RM 的请求压力
- 运行在同一 container 上 task 之间的数据共享。比如，MapJoin 中可以通过共享小表数据的方式，减少资源消耗。

相关参数：

tez.am.container.idle.release-timeout-max.millis	10000	Int value. The maximum amount of time to hold on to a container if no task can be assigned to it immediately. Only active when reuse is enabled. The value must be +ve and >= TezConfiguration#TEZ_AM_CONTAINER_IDLE_RELEASE_TIMEOUT_MIN_MILLIS. Containers will have an expire time set to a random value between TezConfiguration#TEZ_AM_CONTAINER_IDLE_RELEASE_TIMEOUT_MIN_MILLIS && TezConfiguration#TEZ_AM_CONTAINER_IDLE_RELEASE_TIMEOUT_MAX_MILLIS. This creates a graceful reduction in the amount of idle resources held
tez.am.container.idle.release-timeout-min.millis	5000	Int value. The minimum amount of time to hold on to a container that is idle. Only active when reuse is enabled. Set to -1 to never release idle containers (not recommended).
tez.am.container.reuse.enabled	true	Boolean value. Configuration to specify whether container should be reused across tasks. This improves performance by not incurring recurring launch overheads.
tez.am.container.reuse.locality.delay-allocation-millis	250	Int value. The amount of time to wait before assigning a container to the next level of locality. NODE -> RACK -> NON_LOCAL. Delay scheduling parameter. Expert level setting.
tez.am.container.reuse.non-local-fallback.enabled	false	Boolean value. Whether to reuse containers for non-local tasks. Active only if reuse is enabled. Turning this on can severely affect locality and can be bad for jobs with high data volume being read from primary data sources.
tez.am.container.reuse.rack-fallback.enabled	true	Boolean value. Whether to reuse containers for rack local tasks. Active only if reuse is enabled.

三、优缺点

- 优点：
  - 避免中间数据写回 HDFS，减小任务执行时间
  - vertex management 模块使 runtime 动态修改执行计划变成可能
  - input/processor/output 编程模型，大大提高了任务模型的灵活性
  - 提供 container 复用机制与 Tez Session，减少资源消耗
- 缺点：
  - 出现数据重复问题等数据质量问题
  - Tez 与 Hive 捆绑，在其他领域应用较少
  - 社区不活跃

四、Hive On Tez：

1. Hive On Tez 初始并行度算法
2. Hive 调节 mapper 与 reducer 数量
3. Tez Configuration
4. Hive on Tez 配置参数

参考链接： 1. https://hortonworks.com/blog/apache-tez-a-new-chapter-in-hadoop-data-processing/ 2. http://web.eecs.umich.edu/~mosharaf/Readings/Tez.pdf