Streaming技术原理

www.huawei.com

HUAWEI

目标

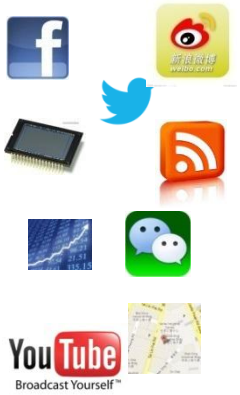
- 学完本课程后，您将能够:
  - 掌握实时流处理概念;
  - 掌握**Storm**系统架构;
  - 掌握**Storm**关键特性;
  - 掌握**CQL**基本概念。

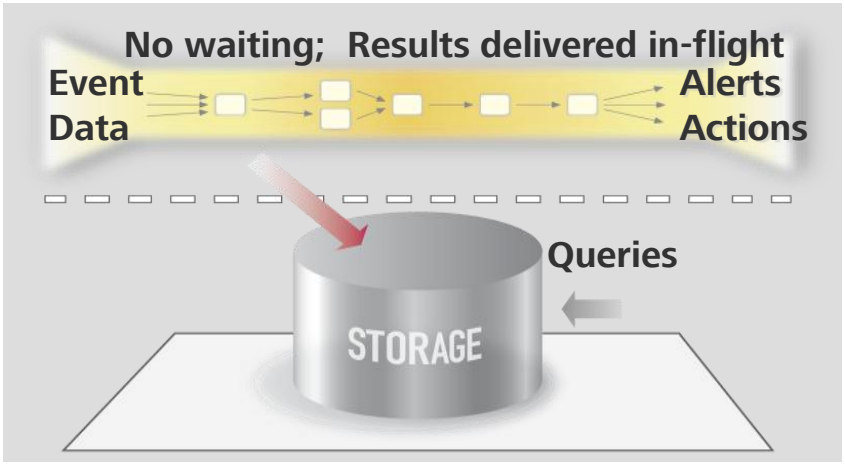
目录

Streaming定义

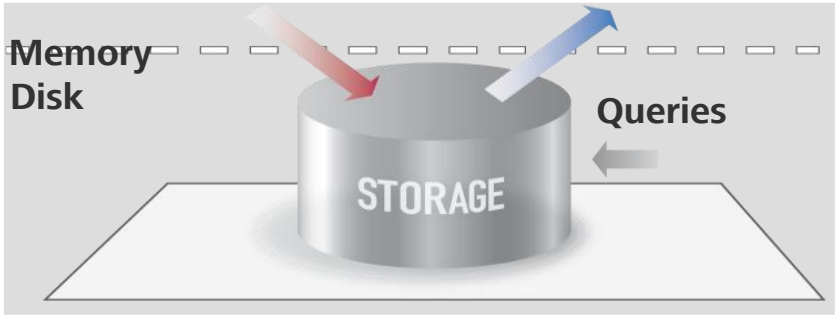
Streaming基于开源**Storm**，是一个分布式、实时计算框架。

- ✓ 事件驱动
- ✓ 连续查询
- ✓ 数据不存储，先计算
- ✓ 实时响应，低延迟





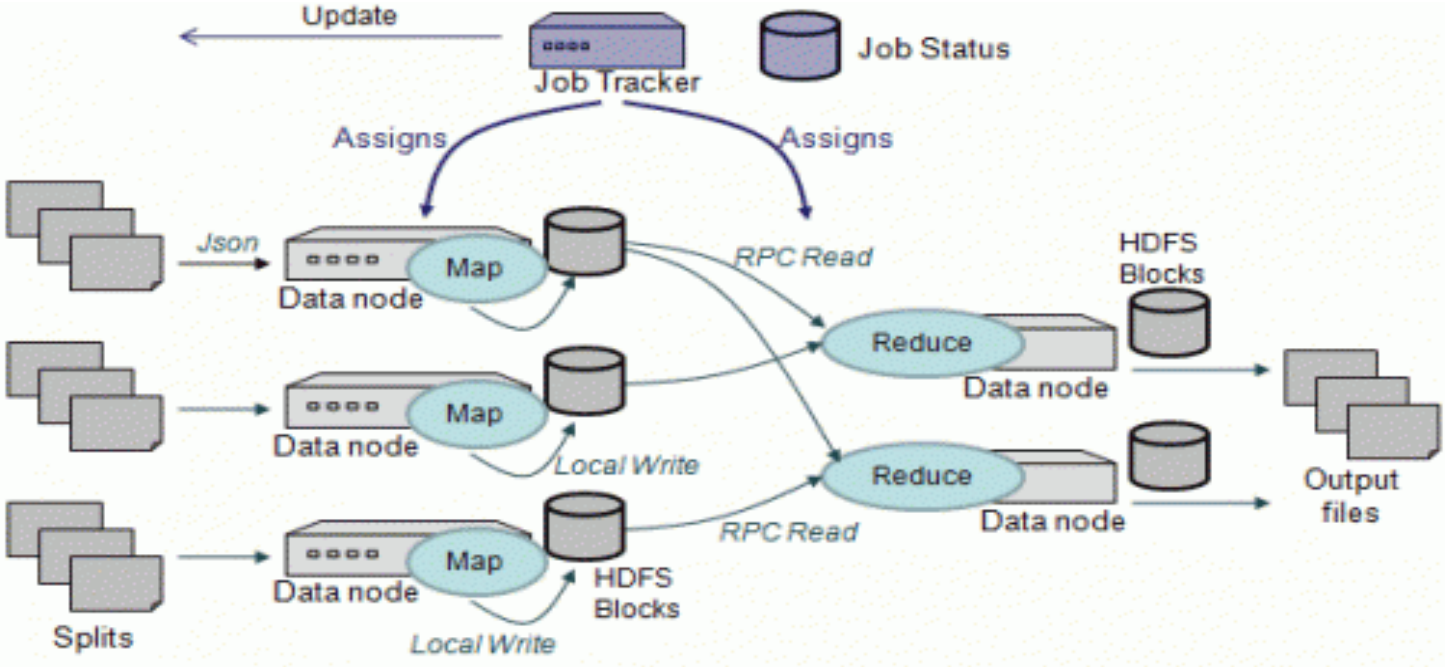
- ◆ 传统数据库技术
- ✓ 数据先存储，再查询处理



HUAWEI

Streaming定义

- **Hadoop技术**
  - ✓ 数据先写入文件系统
  - ✓ 进行批处理



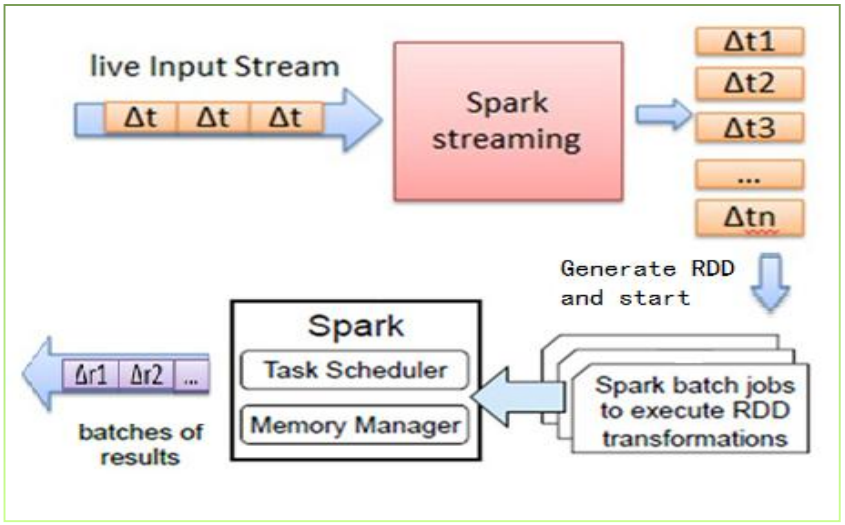
目录

应用场景

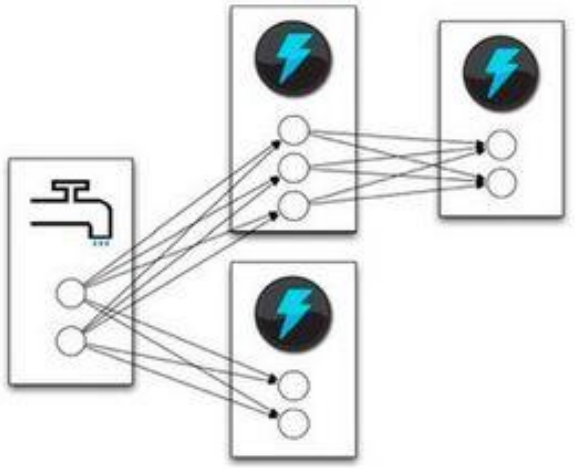
主要应用于以下几种场景：

- ✓ 实时分析：如实时日志处理、交通流量分析等
- ✓ 实时统计：如网站的实时访问统计、排序等
- ✓ 实时推荐：如实时广告定位、事件营销等

与Spark Streaming的比较



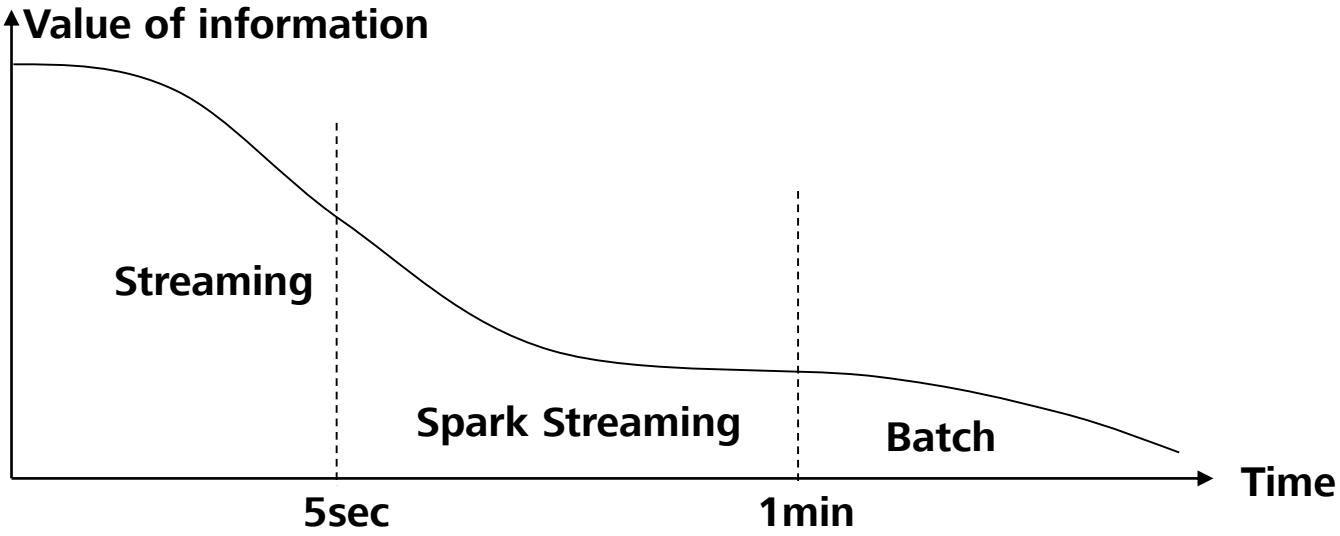
SparkStreaming微批处理流程



Streaming流处理流程

	SparkStreaming	Streaming
任务执行方式	执行逻辑即时启动，运行完回收	执行逻辑预先启动，持续存在
事件处理方式	事件需积累到一定批量时才进行处理	事件实时处理
时延	秒级	毫秒级
吞吐量	高（约为**Streaming**的**2~5**倍）	较高

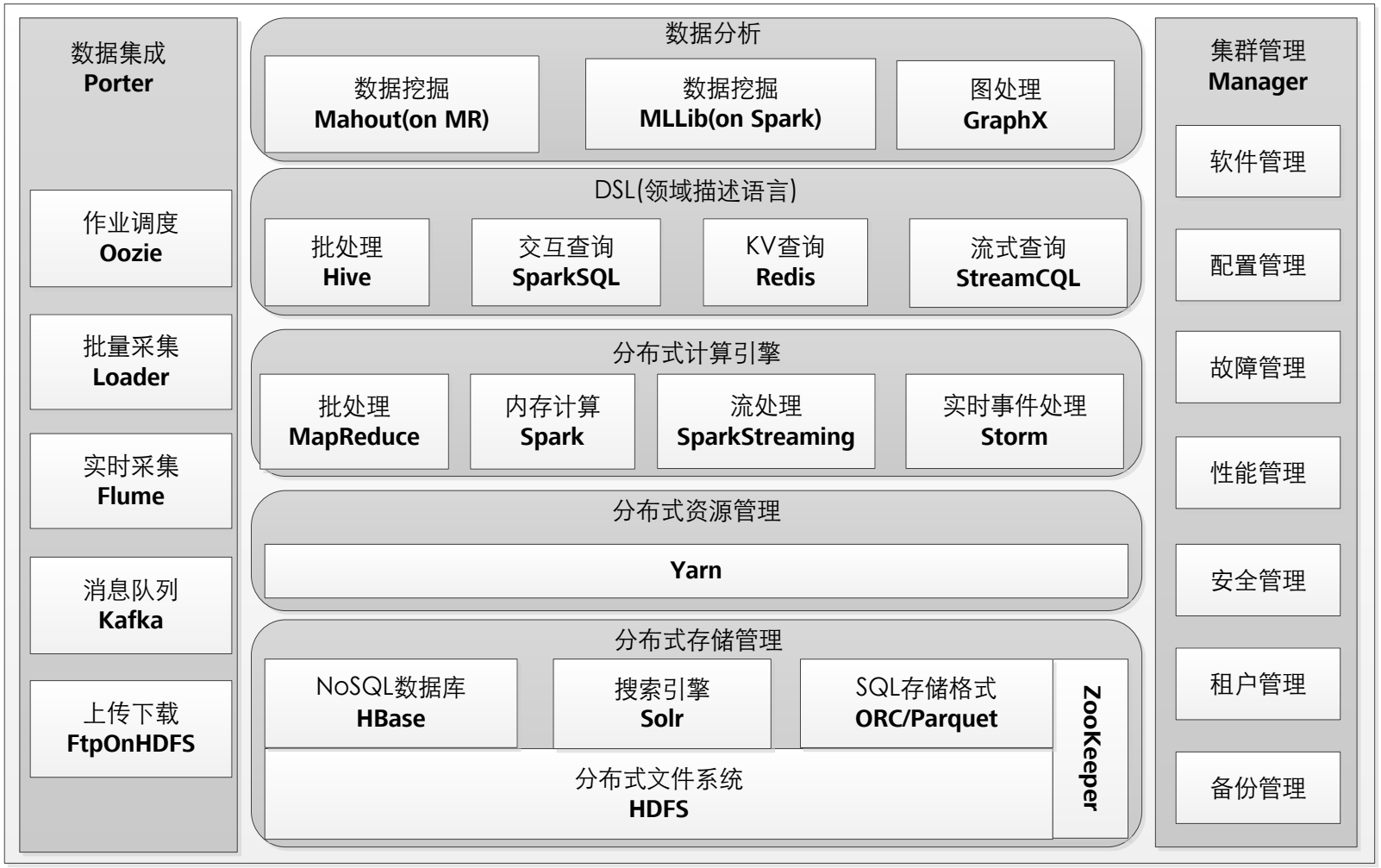
适用场景比较



- **Streaming**适用于对响应时间有严格要求的场景
- **Spark Streaming**适用于对响应时间要求不高的场景

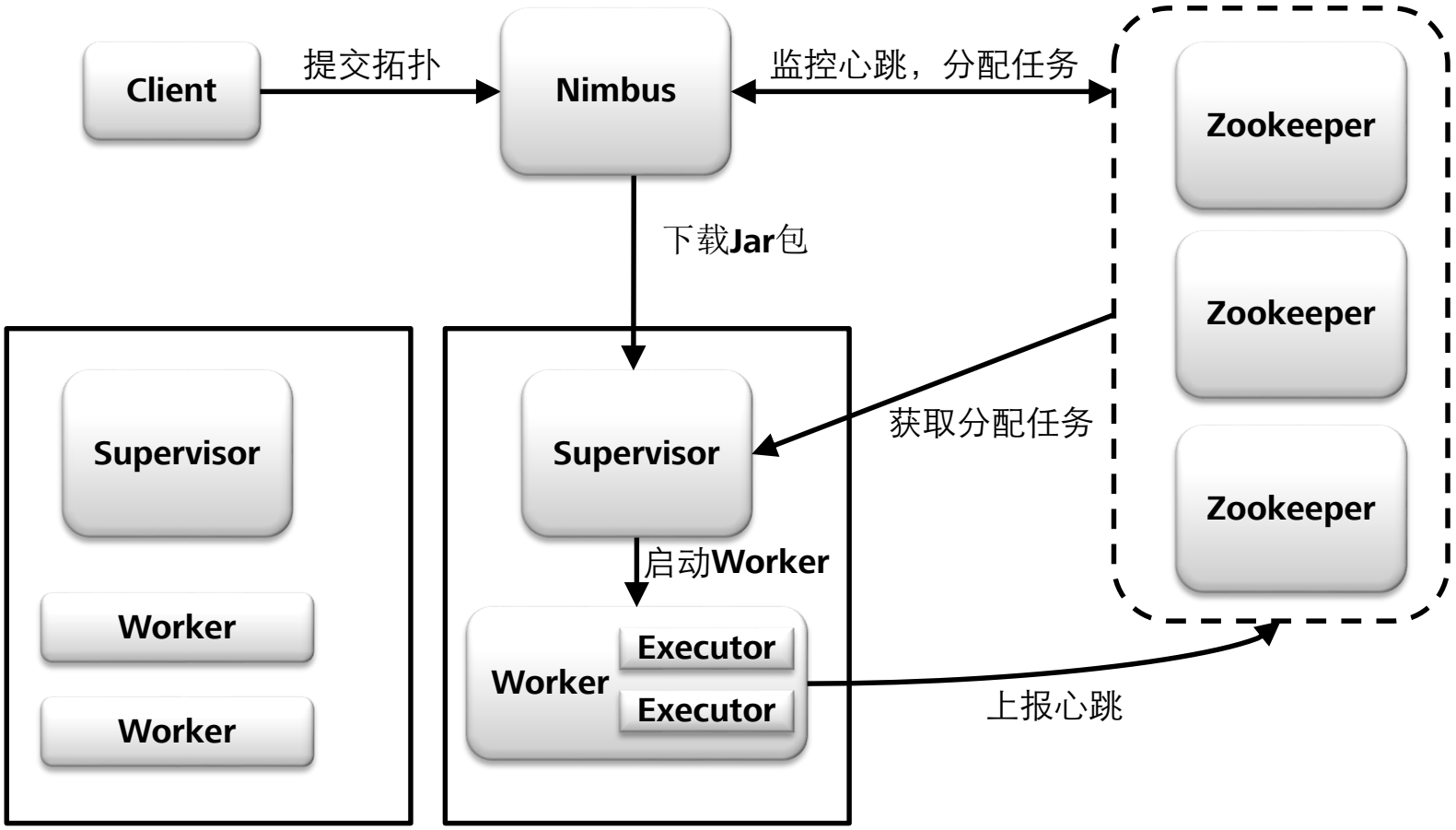
目录

Streaming在FusionInsight产品的位置



目录

系统架构

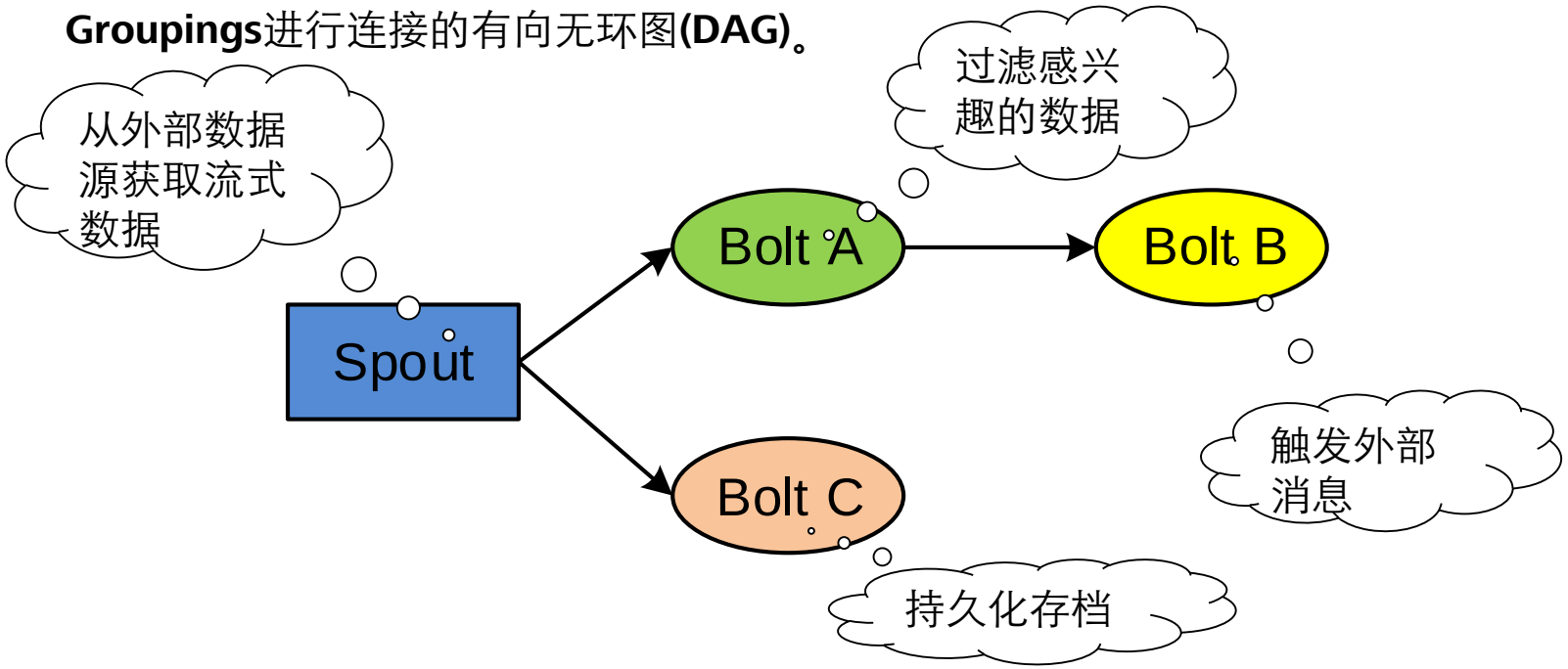


概念

- **Topology**：**streaming**中运行的一个实时应用程序.
- **Nimbus**：负责资源分配和任务调度.
- **Supervisor**：负责接受**nimbus**分配的任务,启动和停止属于自己管理的**worker**进程.
- **Worker**：运行具体处理组件逻辑的进程.
- **Spout**：在一个**topology**中产生源数据流的组件.
- **Bolt**：在一个**topology**中接受数据然后执行处理的组件.
- **Task**：**worker**中每一个**spout/bolt**的线程称为一个**task**.
- **Streams/Tuple**：一次消息传递的基本单元,一组**KeyValue**对；**Streams**是无限的**Tuple**序列
- **Stream grouping**：消息的分组方法

Topology介绍

- 业务处理逻辑被封装进**Streaming**中的**topology**中。
- 一个**topology**是由一组**Spout**组件（数据源）和**Bolt**组件（逻辑处理）通过**Stream Groupings**进行连接的有向无环图**(DAG)**。

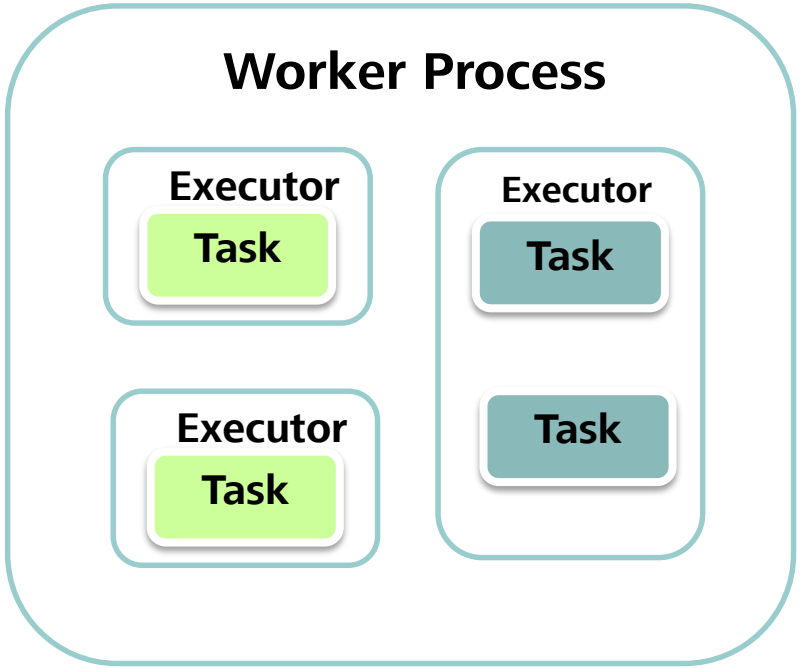


Worker介绍

✓**Worker**：一个**Worker**是一个**JVM进程**，所有的**Topology**都是在一个或者多个**Worker**中运行的。**Worker**启动后是长期运行的，除非人工停止。**Worker**进程的个数取决于**Topology**的设置，且无设置上限，具体可获得调度并启动的**Worker**个数则取决于**Supervisor**配置的**slot**个数

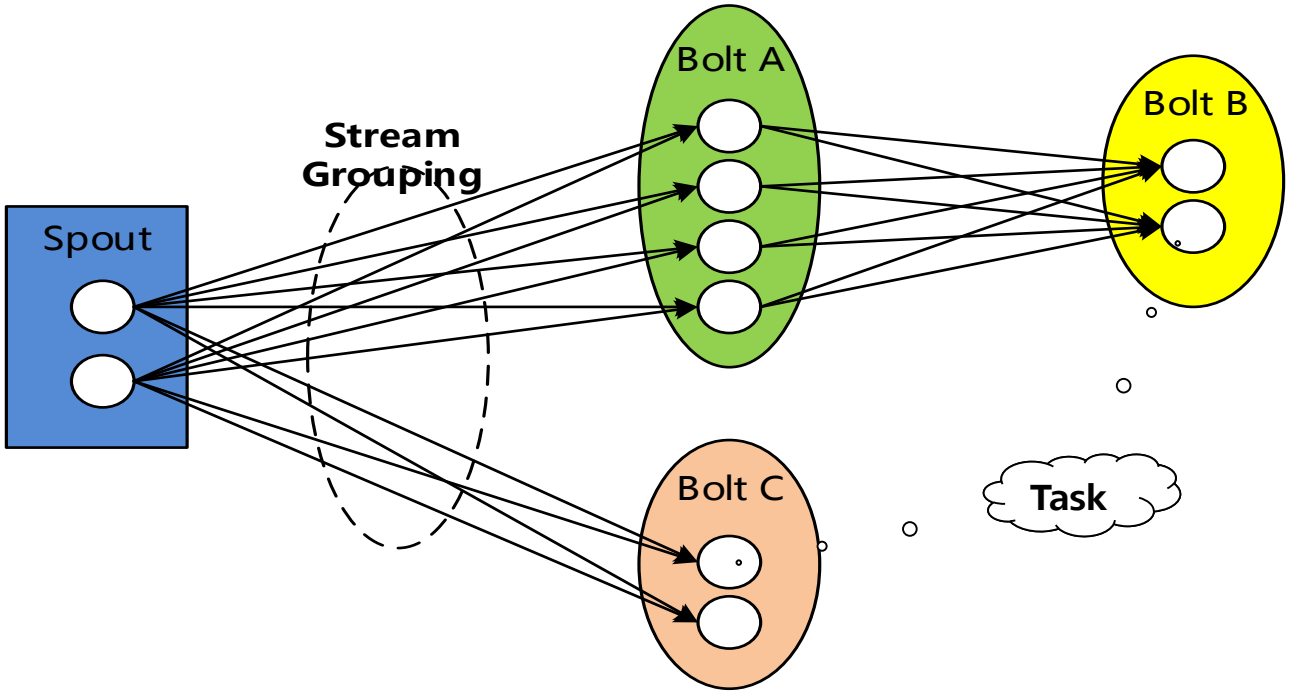
✓**Executor**：在一个单独的**Worker**进程中会运行一个或多个**Executor线程**。每个**Executor**只能运行**Spout**或者**Bolt**中的一个或多个**task**实例。

✓**Task**：是最终完成数据处理的实体单元



Task介绍

Topology里面的每一个**Component**（**Spout/Bolt**）节点都是并行运行的。在**topology**里面， 可以指定每个节点的并发度， **streaming**则会在集群里面分配相应的**Task**来同时计算，以增强系统的处理能力。



消息分发策略

分组方式	功能介绍
fieldsGrouping（字段分组）	按照消息的哈希值分组发送给目标**Bolt**的**Task**
globalGrouping（全局分组）	所有消息都发送给目标**Bolt**的固定一个**Task**
shuffleGrouping（随机分组）	消息发送给目标**Bolt**的随机一个**task**
localOrShuffleGrouping（本地或者随机分组）	如果目标**Bolt**在同一工作进程存在一个或多个**Task**，数据会随机分配给这些**Task**。否则，该分组方式与随机分组方式相同
allGrouping（广播分组）	消息群发给目标**Bolt**的所有**Task**
directGrouping（直接分组）	由数据生产者决定数据发送给目标**Bolt**的哪一个**Task**。需在发送时使用**emitDirect(taskID, tuple)**接口指定**TaskID**
partialKeyGrouping（局部字段分组）	更均衡的字段分组
noneGrouping（不分组）	当前和随机分组相同

常用接口

- **Streaming提供接口：**
  - **REST** 接口

    REST（Representational State Transfer）表述性状态转移接口。
  - **Thrift**接口

    由**Nimbus**提供。**Thrift** 是一个基于静态代码生成的跨语言的**RPC**协议栈 实现，它可以生成包括**C++, Java, Python, Ruby, PHP** 等主流语言的代码，这些代码实现了 **RPC** 的协议层和传输层功能，从而让用户可以集中精力于服务的调用和实现。

目录

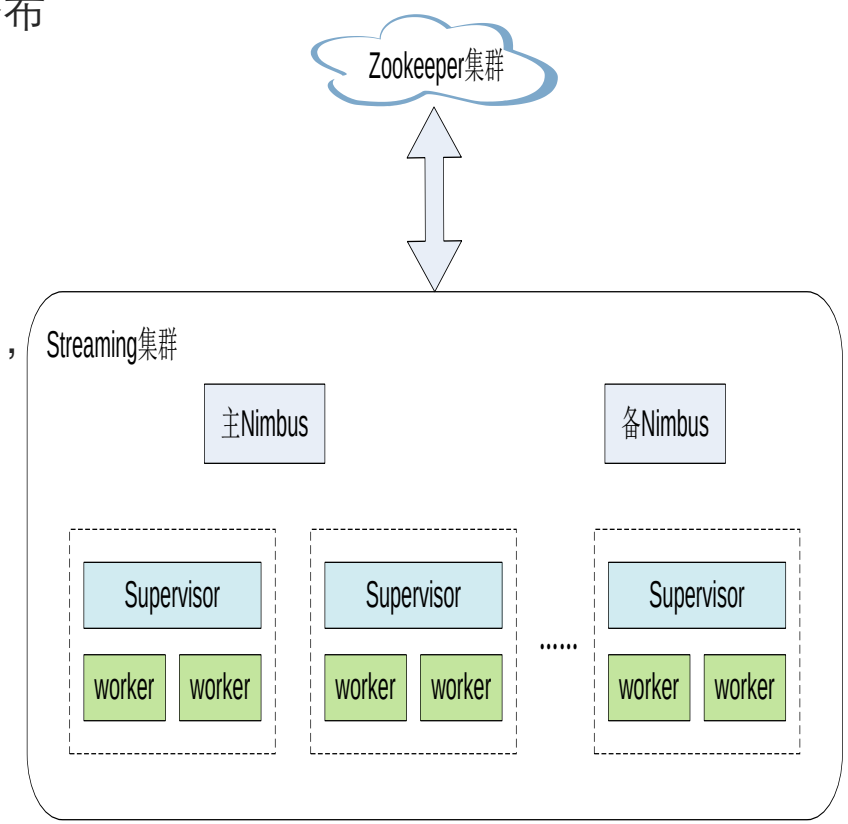
Nimbus HA

- **使用Zookeeper分布式锁**

  Nimbus HA的实现是使用**Zookeeper**分布式锁，通过主备间争抢模式完成的**Leader**选举和主备切换。

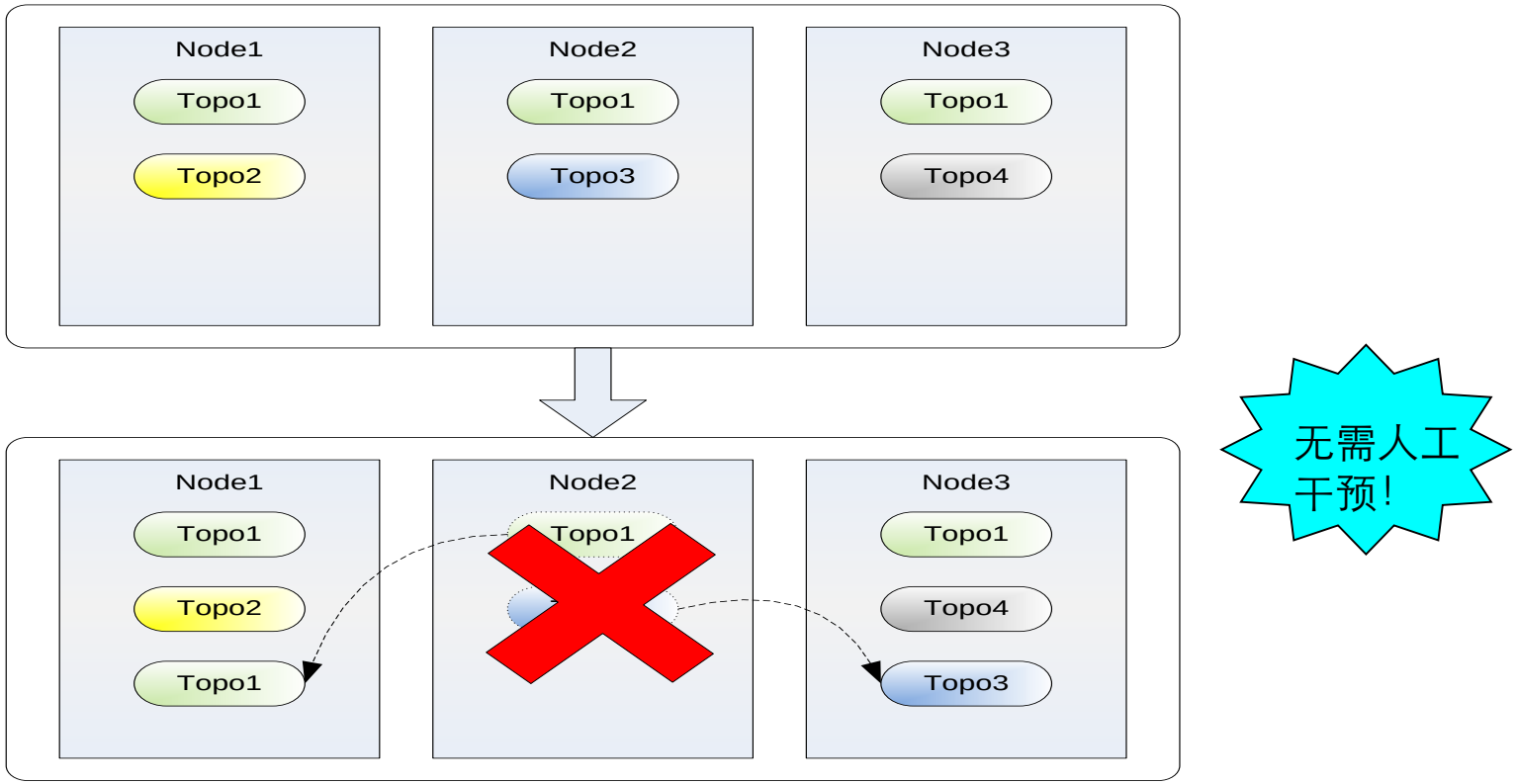
- **主备间元数据同步**

  主备**Nimbus**之间会周期性的同步元数据，保证在发生主备切换后拓扑数据不丢失，业务不受损。



容灾能力

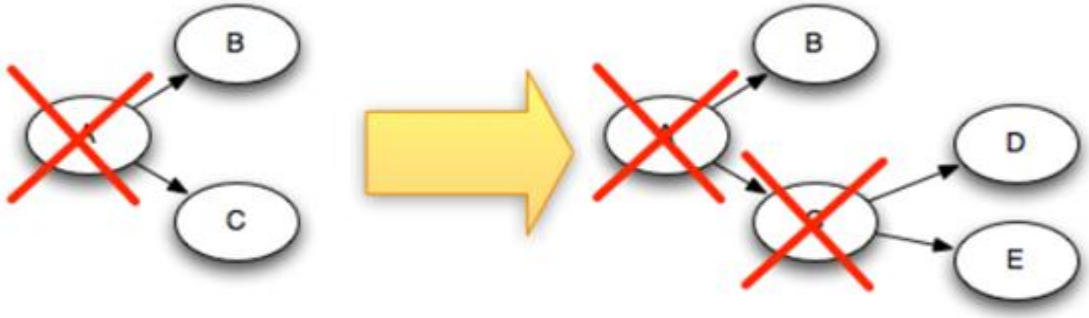
- 容灾能力：节点失效，自动迁移到正常节点，业务不中断。



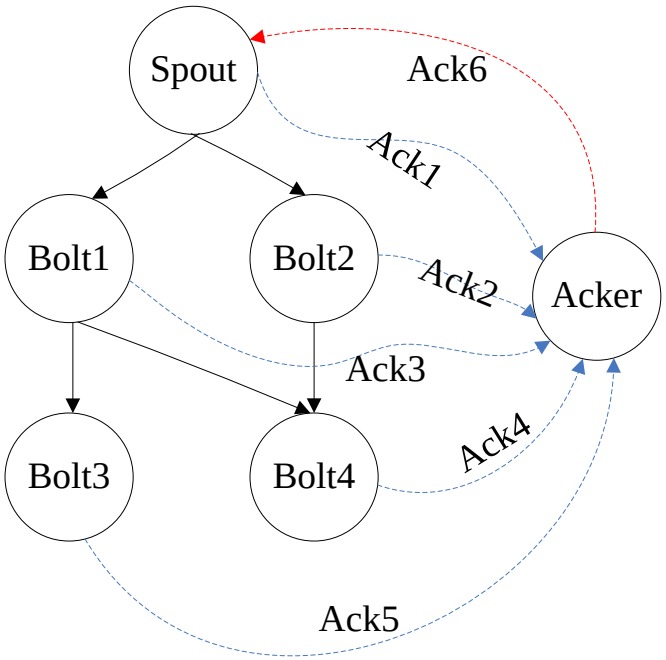
消息可靠性

可靠级别	处理机制	说明
最多一次	无	吞吐量最大，适用于消息可靠性较低的场景
最少一次	**Ack**机制	吞吐量较低，要求数据被完整处理，适用于消息可靠性要求高的场景。
精确一次	**Trident**	**Trident**是**Storm**提供的特殊的事务性**API**，吞吐量最低。

在**streaming**里面一个**tuple**被完全处理的意思是：这个**tuple**以及由这个**tuple**所派生的所有的**tuple**都被成功处理。如果这个消息在**Timeout**所指定的时间内没有成功处理,这个**tuple**就被认为处理失败了。



ACK机制



✓**Spout**发送一个**Tuple**时，会通知**Acker**一个新的根消息产生了，**Acker**会创建一个新的**tuple tree**，并初始化校验和为**0**。

✓**Bolt**发送消息时向**Acker**发送**anchor tuple**，刷新**tuple tree**，并在发送成功后向**Acker**反馈结果。如果成功则重新刷新校验和，如果失败则**Acker**会立即通知**Spout**处理失败。

✓当**tuple tree**被完全处理（校验和为**0**），**Acker**会通知**Spout**处理成功。

✓**Spout**提供**ack()**和**fail()**接口方法用于处理**Acker**的反馈结果，需要用户实现。一般在**fail()**方法中实现消息重发逻辑。

可靠性级别设置

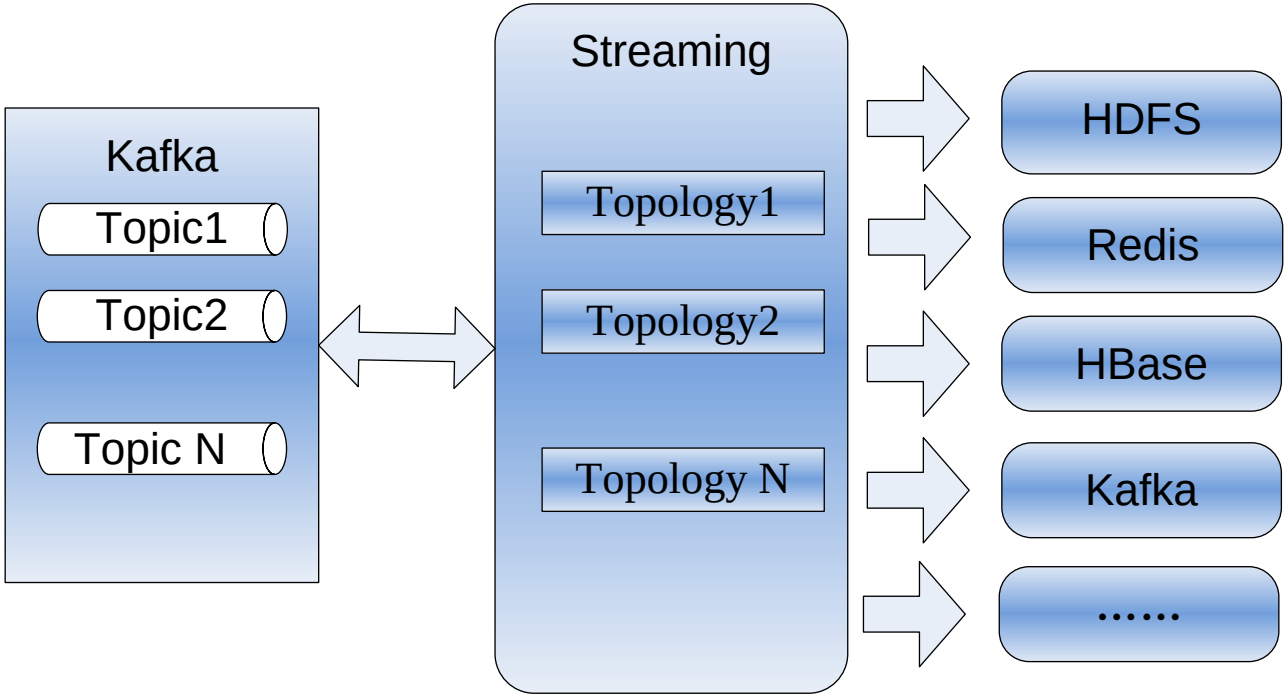
如果并不要求每个消息必须被处理（允许在处理过程中丢失一些信息），那么可以关闭消息的可靠处理机制，从而可以获取较好的性能。关闭消息的可靠处理机制意味着系统中的消息数会减半。

有三种方法可以关闭消息的可靠处理机制：

- 将参数**Config.TOPOLOGY_ACKERS**设置为**0**。
- **Spout**发送一个消息时，使用不指定消息**messageID**的接口进行发送。
- **Bolt**发送消息时使用**Unanchor**方式发送，使**Tuple**树不往下延伸，从而关闭派生消息的可靠性。

与离线系统集成

- **HDFS, HBase, Kafka…**



✓ 整合**HDFS/Hbase**等外部组件，易于实时结果供离线分析

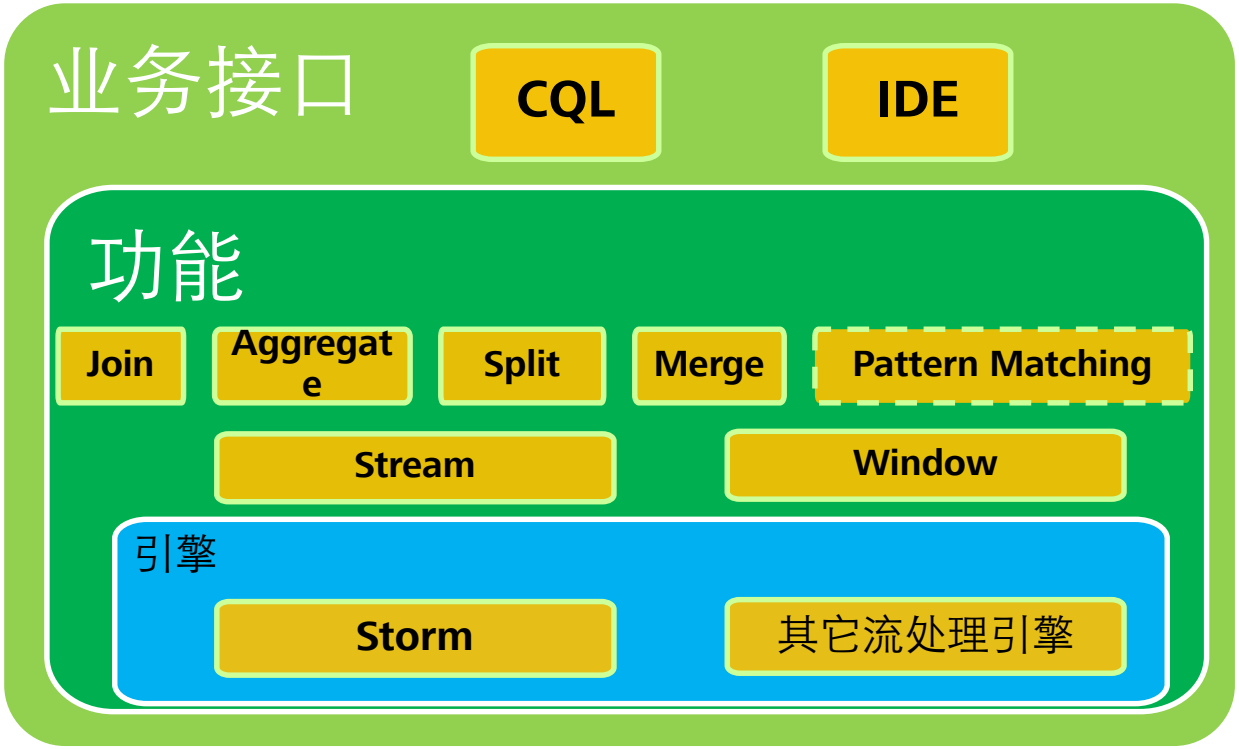
目录

StreamCQL简介

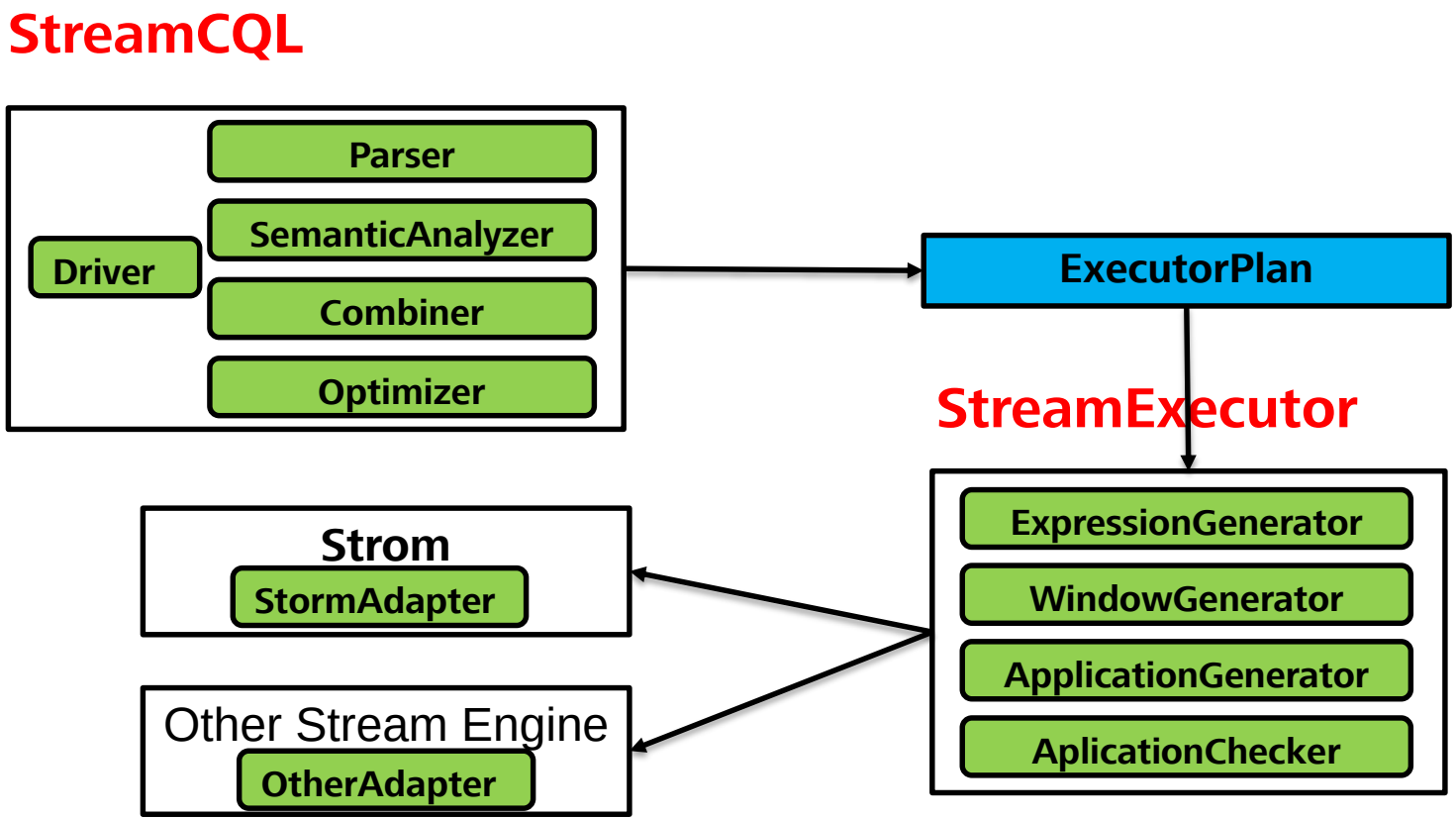
StreamCQL(Stream Continuous Query Language)是建立在分布式流处理平台基础上的查询语言**(CQL)**，架构支持构建在多种流处理引擎之上，目前主要适配**Storm**。

当前多数流处理平台仅提供分布式处理能力，业务逻辑开发复杂，流计算业务功能较弱，存在业务逻辑重用性不高、重复开发、开发效率低下等问题。**StreamCQL**提供了较丰富的分布式流计算功能，除了具有过滤、转换等传统的**SQL**基本能力之外**, StreamCQL**引入基于流的时间窗口的计算，提供窗口数据的统计、关联等能力，以及流数据的拆分、合并等功能。

StreamCQL与流处理平台



StreamCQL架构



StreamCQL带来Storm开发方式的转变

Storm原生API

```
//Def Input:
public void open(Map conf, TopologyContext
context, SpoutOutputCollector collector) {…}
public void nextTuple() {…}
public void ack(Object id) { …}
public void
declareOutputFields(OutputFieldsDeclarer declarer)
{…}
//Def logic:
public void execute(Tuple tuple,
BasicOutputCollector collector) {…}
 public void
declareOutputFields(OutputFieldsDeclarer ofd) {…}
//Def Output:
 public void execute(Tuple tuple,
BasicOutputCollector collector) {…}
 public void
declareOutputFields(OutputFieldsDeclarer ofd) {…}
//Def Topology:
public static void main(String[] args) throws
Exception {…}
```

StreamCQL

```
--Def Input:
CREATE INPUT STREAM S1 …


--Def logic:
INSERT INTO STREAM filterstr SELECT * FROM
S1 WHERE name="HUAWEI";


--Def Output:
CREATE  OUTPUT STREAM S2…


--Def Topology:
SUBMIT APPLICATION test;
```

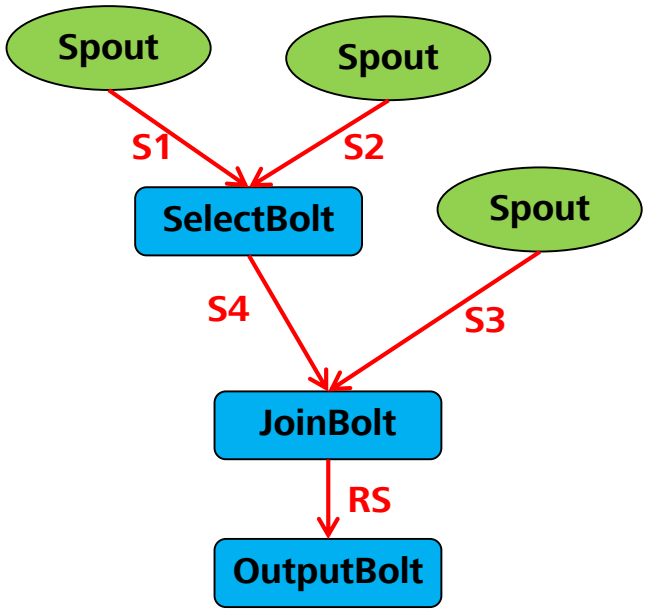
StreamCQL生成Storm拓扑

```
CREATE INPUT STREAM S1
(...)
SOURCE KafkaInput PROPERTIES(...);

CREATE INPUT STREAM S2...;
CREATE INPUT STREAM S3...;

CREATE OUTPUT STREAM RS(...)
SINK kafkaOutput PROPERTIES(...)

INSERT INTO S4 SELECT *,1 FROM S1;
INSERT INTO S4 SELECT *,2 FROM S2;
INSERT INTO RS
 SELECT * FROM S4[ROWS 10 BATCH]
  INNER JOIN S3[RANGE 3 HOURS
SLIDE]
   ON S4.id=S3.type
   WHERE S4.id > 5;
```



总结

- **Streaming**定义
- 描述了**Streaming**的应用场景
- **Streaming**在**FusionInsight**产品的位置
- **Steaming**的系统架构
- **Streaming**的关键特性介绍
- **StreamCQL**介绍

习题

- 判断题

  1. 一个套接字由**5**元组组成，分为源套接字和目的套接字。源套接字是指：源**IP**地址 + 源端口 + 目的**IP**地址。 （**T or F)**

  2. **IP**报文头中的协议（**Protocol**）字段标识了其上层所使用的协议。当上层为**TCP**协议时该字段值为**6**，当上层为**UDP**协议时该字段值为**17**。 （**T or F)**

- 单选题

  1. 在**TCP**建立连接的三次握手中，对于报文**SYN(seq=b,ack=a+1)**，下列说法正确的有？ （ ）

  A.对序号为**b**的数据包进行确认

  B.对序号为**a+1**的数据包进行确认

  C.下一个希望收到的数据包的序号为**b**

  D.下一个希望收到的数据包的序号为**a+1**

习题

1. [单选]以下对与**Supervisor**的描述正确的是

   A. **Supervisor**负责资源分配和任务调度。

   B. 负责接受**Nimbus**分配的任务,启动和停止属于自己管理的**worker**进程。

   C. 运行具体处理组件逻辑的进程。

   D. 在一个**Topology**中接受数据然后执行处理的组件。

2. [单选] **Streaming**集群在运行期间，直接依赖于下面那些组件

   A. **HDFS**

   B. **ZooKeeper**

   C. **Hbase**

   D. **DBService**

习题

3. [多选]关于**Streaming**的容错机制，描述正确的有

   A. **Nimbus** 主备部署，解决**Nimbus**单点问题，支持主从热切换

   B. **Supervisor**失效后能够自动恢复，并且不影响正在运行的业务

   C. **Worker**失效后能够自动恢复，继续运行

   D. 集群内节点失效后在该节点上的任务会被重分配到其他正常节点

4. [多选]下面哪些关键词是**Streaming**的特点

   A. 高容错

   B. 高性能

   C. 可扩展

   D. 批量处理

思考题

5. Streaming是如何保障消息可靠性?

Thank you

www.huawei.com