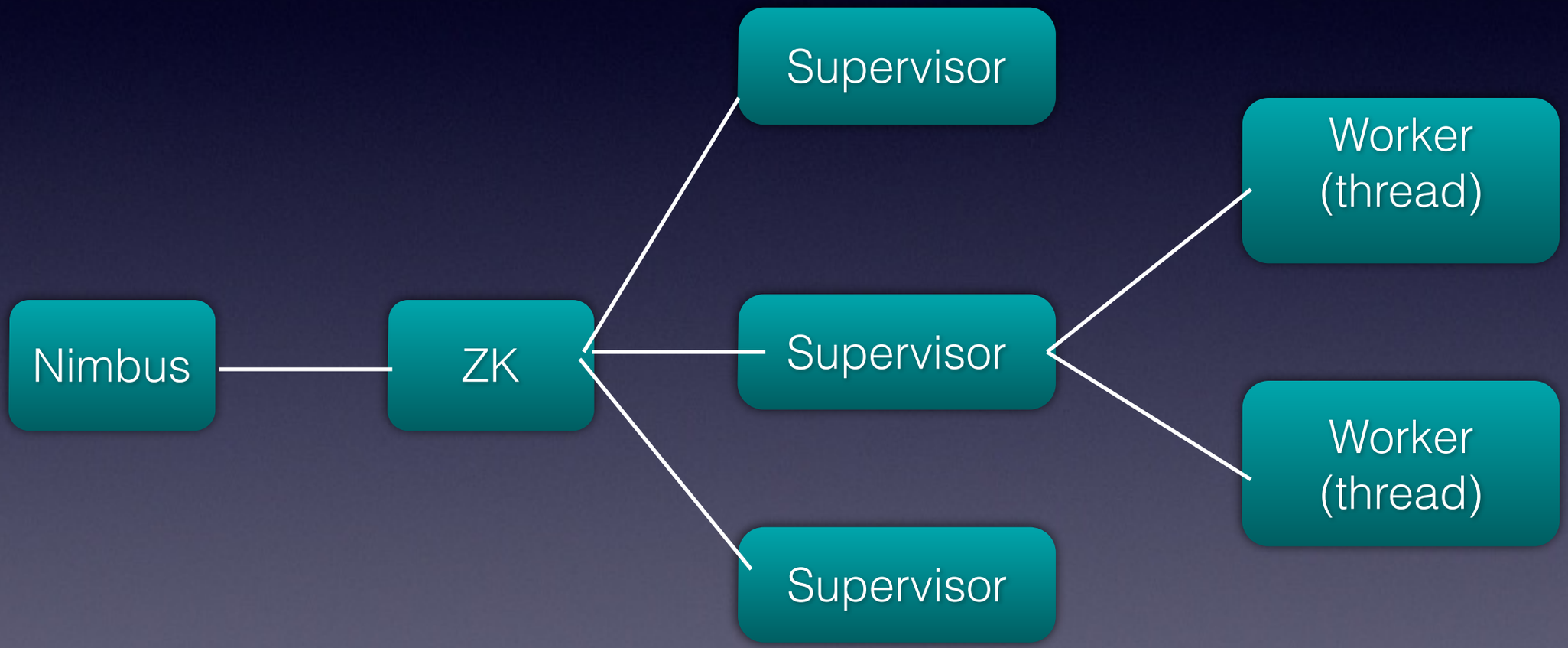
Spark Streaming原理与实践

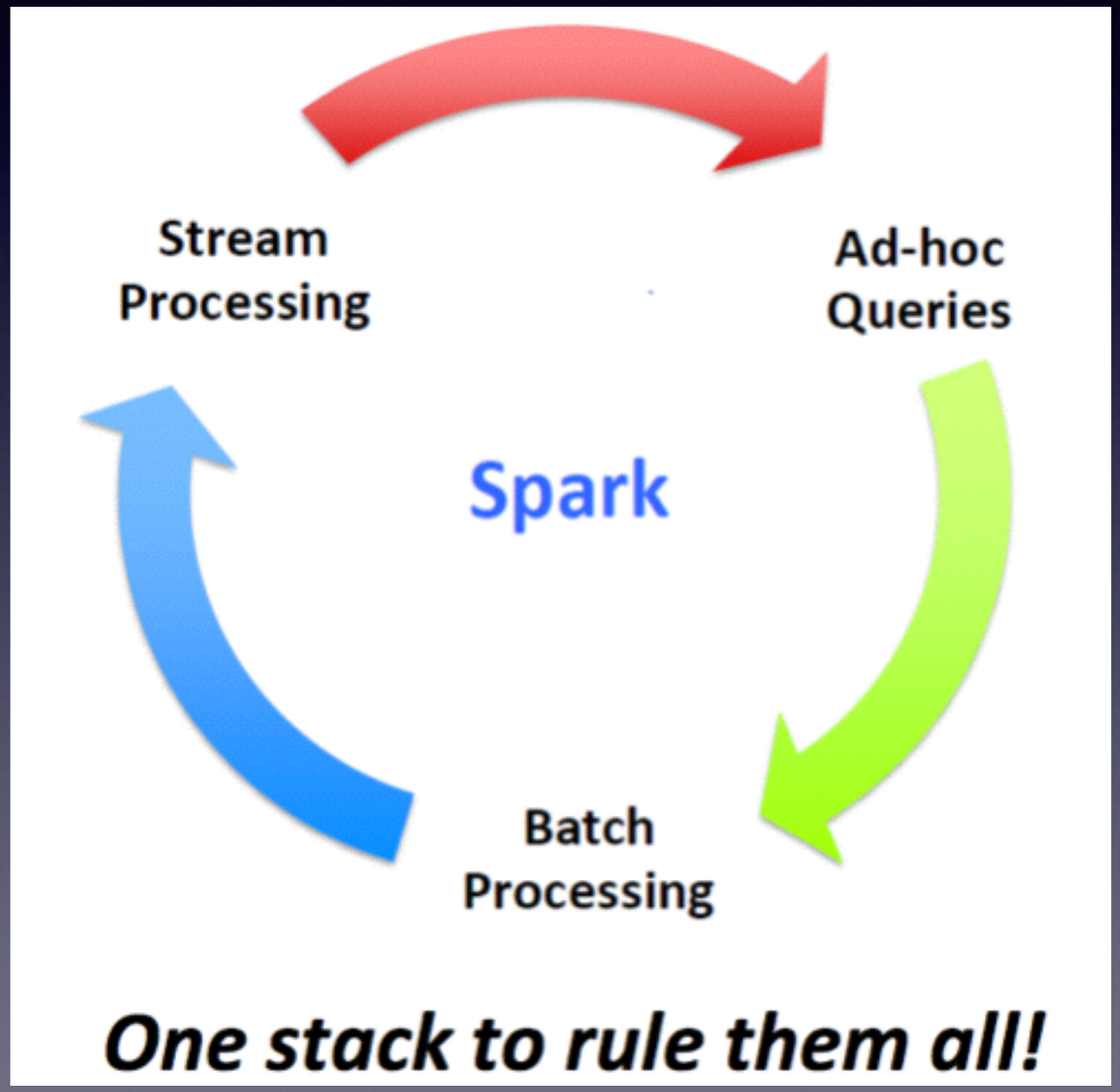
陈 超
@CrazyJvm

流式计算—你想到了什么?

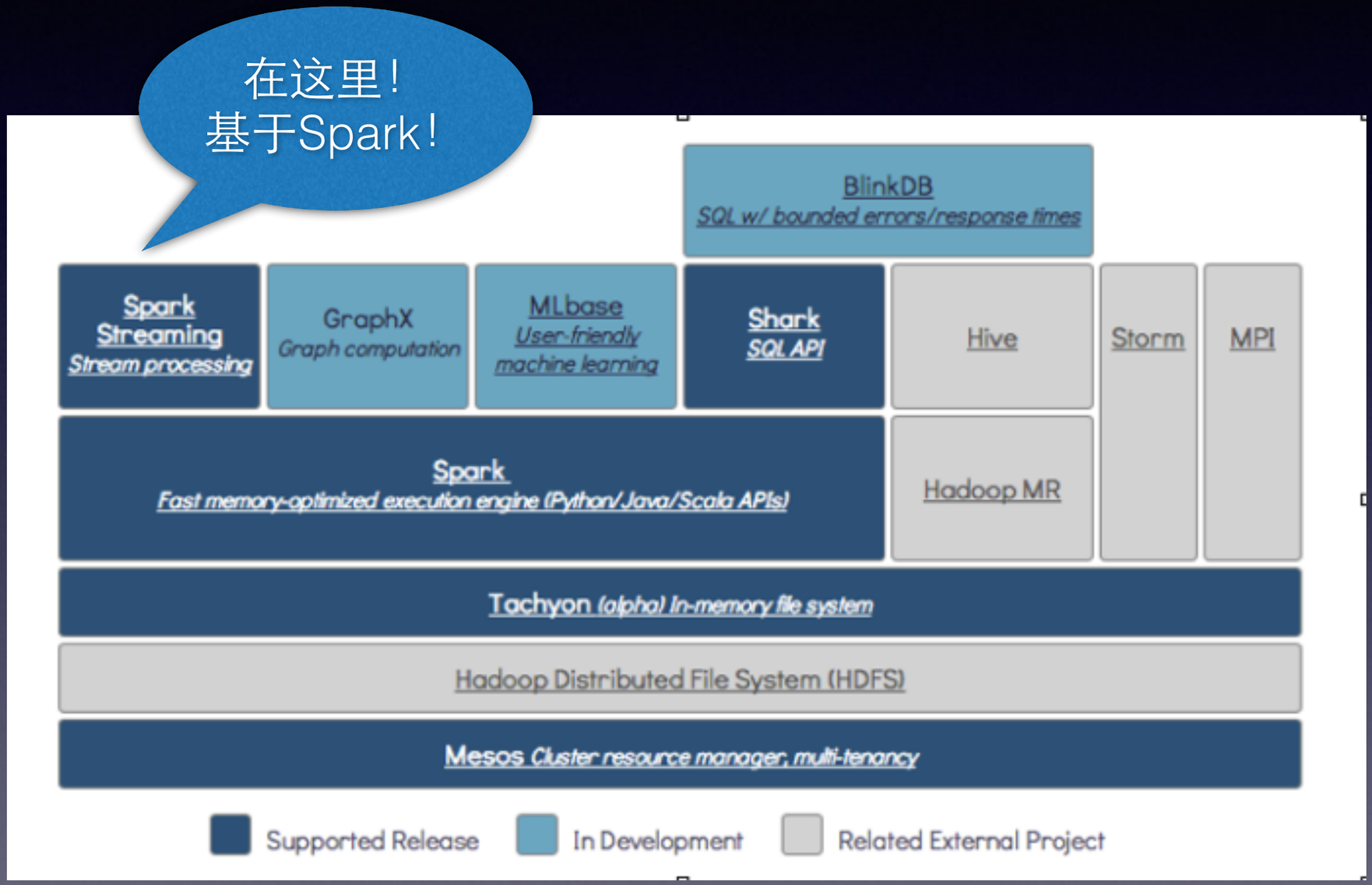
Storm???



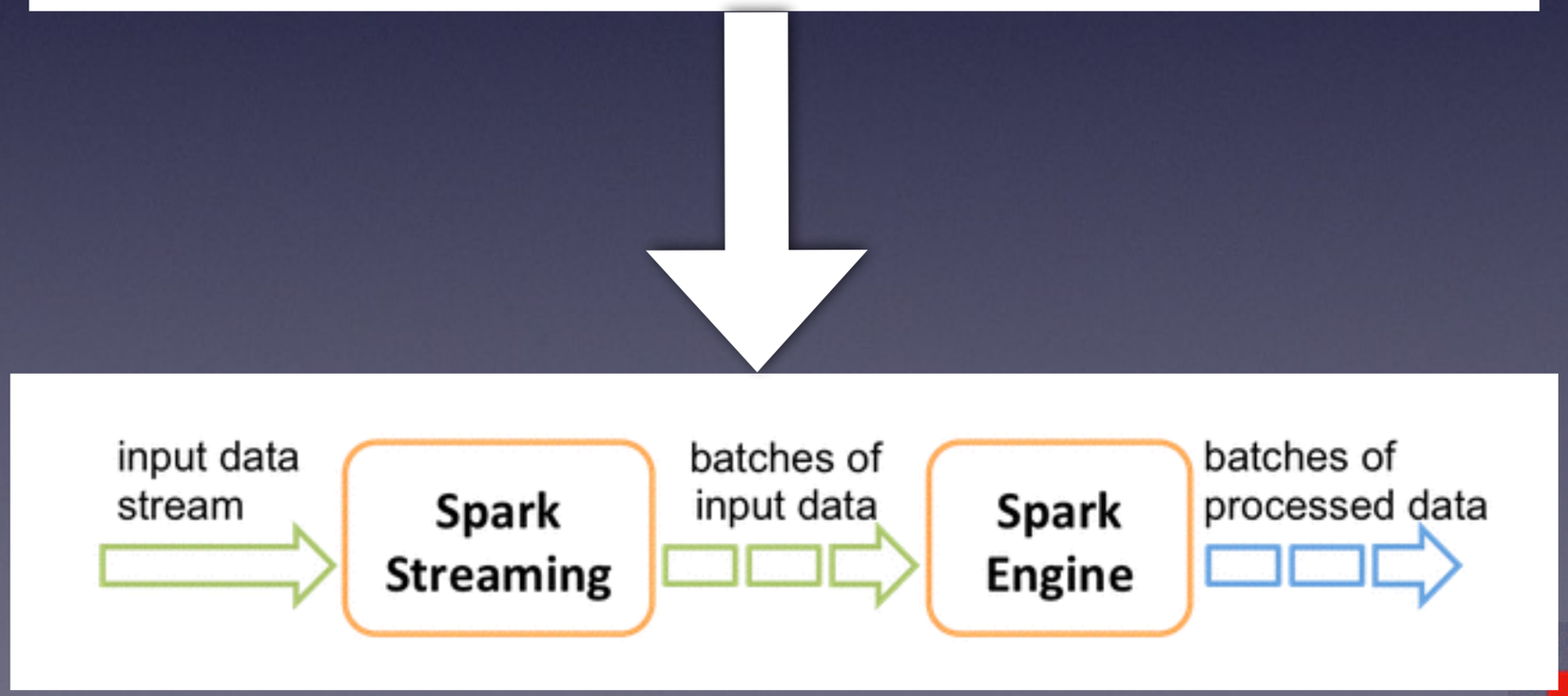
One Stack rule them all!



Berkeley Data Analytics Stack



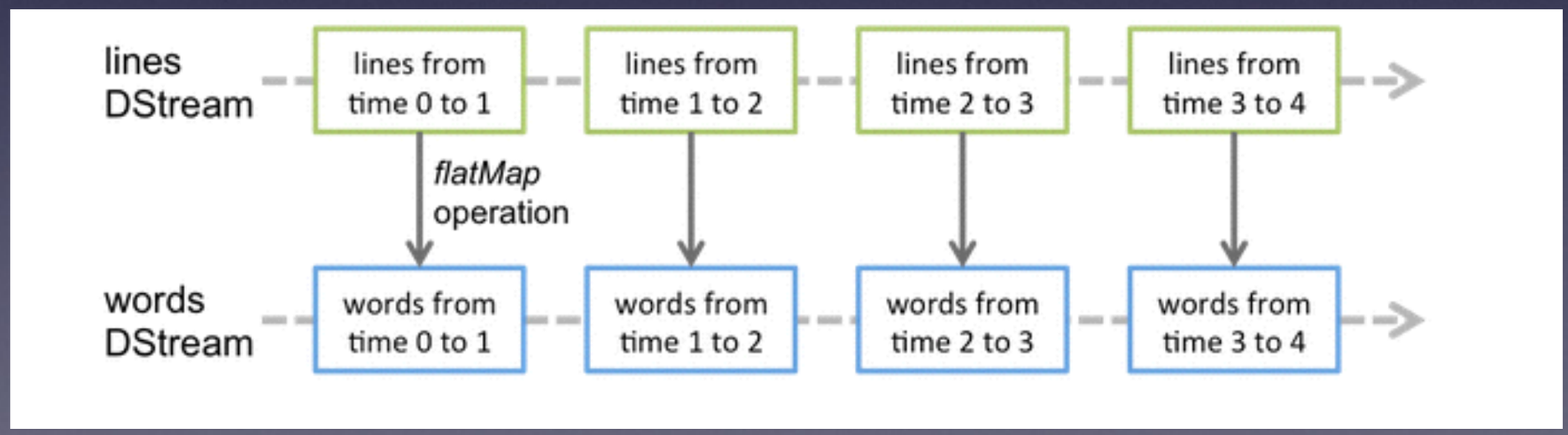
Overview



DStream

- 将流式计算分解成一系列确定并且较小的批处理作业
- 将失败或者执行较慢的任务在其它节点上并行执行
- 较强的容错能力(基于Lineage)

DStream



数据源

- Kafka
- Flume
- Twitter
- ZeroMQ
- MQTT
- TCP sockets
- Akka actor
- HDFS
- 自定义

transformation(1)

- 和Spark的语义一致

map, flatMap, filter, count, reduce, etc

groupByKey, reduceByKey, sortByKey, join, etc

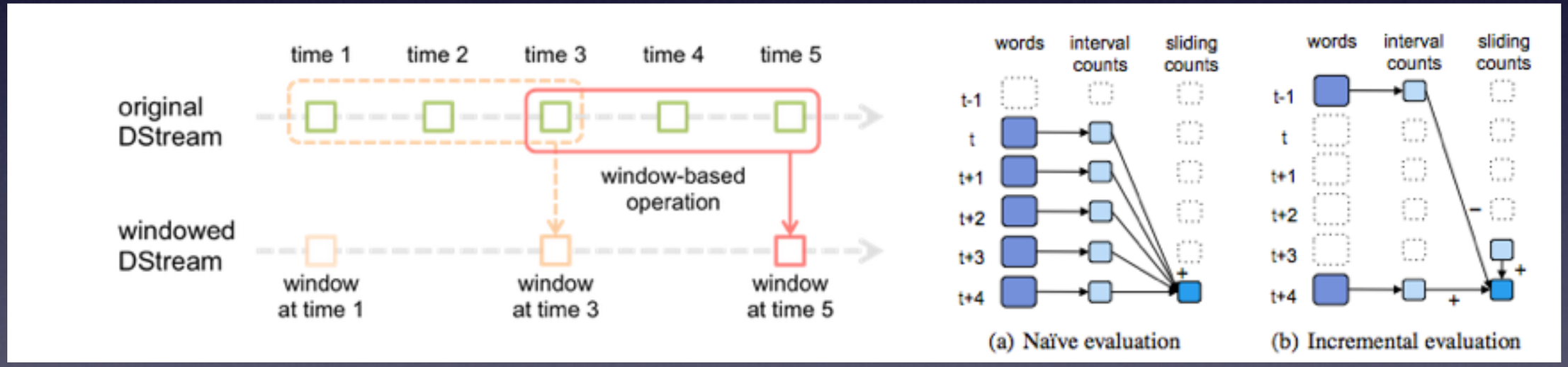
transformation(2)

保持状态

- 带状态

  updateStateByKey (很常用)

- window操作



window, countByWindow, reduceByWindow

countByValueAndWindow, reduceByKeyAndWindow

DStream输出

- print
- foreachRDD
- saveAsObjectFiles
- saveAsTextFiles
- saveAsHadoopFiles

持久化

- 仍然允许用户调用persist来持久化
- 默认级别为<内存+序列化>
- 对于window和stateful操作默认持久化
- 对于来自网络的数据源，每份数据会在内存中存两份

思考：何时清理呢?

```
def socketTextStream(
    hostname: String,
    port: Int,
    storageLevel: StorageLevel = StorageLevel.MEMORY_AND_DISK_SER_2
  ): DStream[String] = {
  socketStream[String](hostname, port, SocketReceiver.bytesToLines, storageLevel)
}
```

checkpoint

- 对于window和stateful操作必须checkpoint
- 通过StreamingContext的checkpoint来指定目录
- 通过DStream的checkpoint指定间隔时间
- 间隔必须是slide interval的倍数

```
// Set the checkpoint interval to be slideDuration or 10 seconds, which ever is larger
if (mustCheckpoint && checkpointDuration == null) {
  checkpointDuration = slideDuration * math.ceil(Seconds(10) / slideDuration).toInt
  logInfo("Checkpoint interval automatically set to " + checkpointDuration)
}
```

checkpoint的限制

```
private[streaming] def validate() {
  assert(rememberDuration != null, "Remember duration is set to null")

  assert(
    !mustCheckpoint || checkpointDuration != null,
    "The checkpoint interval for " + this.getClass.getSimpleName + " has not been set." +
      " Please use DStream.checkpoint() to set the interval."
  )

  assert(
   checkpointDuration == null || context.sparkContext.checkpointDir.isDefined,
    "The checkpoint directory has not been set. Please use StreamingContext.checkpoint()" +
    " or SparkContext.checkpoint() to set the checkpoint directory."
  )

  assert(
    checkpointDuration == null || checkpointDuration >= slideDuration,
    "The checkpoint interval for " + this.getClass.getSimpleName + " has been set to " +
      checkpointDuration + " which is lower than its slide time (" + slideDuration + "). " +
      "Please set it to at least " + slideDuration + "."
  )

  assert(
    checkpointDuration == null || checkpointDuration.isMultipleOf(slideDuration),
    "The checkpoint interval for " + this.getClass.getSimpleName + " has been set to " +
      checkpointDuration + " which not a multiple of its slide time (" + slideDuration + "). " +
      "Please set it to a multiple " + slideDuration + "."
  )

  assert(
    checkpointDuration == null || storageLevel != StorageLevel.NONE,
    "" + this.getClass.getSimpleName + " has been marked for checkpointing but the storage " +
      "level has not been set to enable persisting. Please use DStream.persist() to set the " +
      "storage level to use memory for better checkpointing performance."
  )

  assert(
    checkpointDuration == null || rememberDuration > checkpointDuration,
    "The remember duration for " + this.getClass.getSimpleName + " has been set to " +
      rememberDuration + " which is not more than the checkpoint interval (" +
      checkpointDuration + "). Please set it to higher than " + checkpointDuration + "."
  )
```

容错☆

- 回顾Spark容错(确定的不可变分布式数据集基于lineage恢复)
- RDD的某些partition丢失了，可以通过lineage信息重新计算恢复
- Streaming也是基于RDD计算，所以一切仍然适用

Worker节点失败

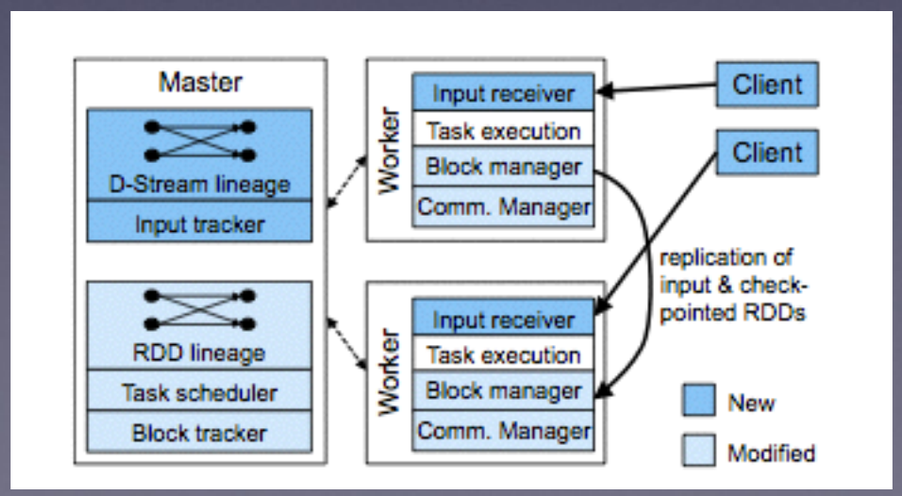
- 数据源来自外部文件系统,如HDFS

  一定可以通过重新读取数据来恢复，绝对不会有数据丢失。

- 数据源来自网络

  默认会在两个不同节点加载数据到内存，一个节点fail了，系统可以通过另一个节点的数据重算

  假设正在运行input receiver的节点fail了，可能会丢失一部分数据(why？没来得及复制)



Driver节点失败

- 会定期将元数据写到指定的checkpoint目录
- Driver失败后可以通过读取元数据重新启动Driver

```
val context = StreamingContext.getOrCreate(checkpointDirectory, functionToCreateContext _)
```

- Spark 0.9.0开始已经提供自动重启功能

```
./bin/spark-class org.apache.spark.deploy.Client launch
   [client-options] \
   <cluster-url> <application-jar-url> <main-class> \
   [application-options]
```

注意：假设代码重新编译过，必须要生成新的context，假设使用了getOrCreate，必须确保每次重新编译后清空 checkpoint目录。

- 避不开的性能问题！— 很多与Spark想通，但也有差异。

合适的并行度

- 回忆Spark调优。过多的reducer会怎样？过少呢？

减少任务启动开销

- 使任务更小(更好的序列化)
- 在Standalone及coarse-grained模式下的任务启动要比fine-grained省时

选择合适的batch size

- 没有最好的size，只有最合适的size，一切以系统反馈的数据说话
- 原则：要来得及消化流进系统的数据
- 可以从Log4j或者StreamingListener获取反馈

```
/**
 * A listener interface for receiving information about an ongoing streaming
 * computation.
 */
trait StreamingListener {
  /**
   * Called when processing of a batch has completed
   */
  def onBatchCompleted(batchCompleted: StreamingListenerBatchCompleted) { }

  /**
   * Called when processing of a batch has started
   */
  def onBatchStarted(batchStarted: StreamingListenerBatchStarted) { }
}
```

内存永远是个大问题

- 默认序列化后放入内存
- 清理缓存的RDD(LRU不够，为什么？)
- 在spark.cleaner.ttl之前缓存的RDD都会被清除掉

MetadataCleaner
spark.cleaner.ttl

- 设置spark.streaming.unpersis，系统为你分忧
- CMS (暂停时间短，但吞吐率不高，并且会引起内存碎片)
- JVM还有另一个参数:-XX:CMSFullGCsBeforeCompaction

Spark Streaming 正在变得变得越来越稳定，越来越高效。

Let’s have a try!

Demo 1

从网络上获取数据并处理

Demo 2

指定目录处理(本地和HDFS均可)

Demo 3

带状态的处理stateful

Demo 4

Window操作

谢谢

小象学院
CHINA HADOOP