法律声明

- 本课件包括演示文稿、示例、代码、题库、视频和声音等内容，北风网和讲师拥有完全知识产权；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或者机构不得盗版、复制、仿造其中的创意和内容，我们保留一切通过法律手段追究违反者的权利。
- 课程详情请咨询
  - 微信公众号：北风教育
  - 官方网址：http://www.ibeifeng.com/

内存计算框架Spark

大话SPARK 2.X新特性

主讲人：Gerry

上海育创网络科技有限公司

SPARK2.x版本主要新特性

- whole stage code generation
  - SparkSQL执行过程中将物理计划变成硬编码，对物理执行的多次调用转行为代码的for循环，减少执行的函数调用次数，当记录数据量比较大的时候，可以有效的提高计算性能
- SparkSession
  - 引入SparkSession概念，提供了一个统一程序的入口，现阶段主要是SparkSQL中使用
- Unifying DataFrame and Dataset
  - 统一DataFrame和Dataset的概念，SCALA和JAVA开发语言中主要使用Dataset来进行数据开发，其中DataFrame仅仅是Dataset的别名，但是在R和Python中，DataFrame还是编程的主要抽象
- Structured Streaming
  - 一种新的流式数据处理方式，利用Dataset的高性能执行API，提升了Streaming处理中的执行性能

SPARK版本比较

SPARK版本	Hadoop版本	Hive版本	Scala版本	JDK版本
1.6.x	1.x~2.x	1.2.1	2.10、2.11	1.7
2.0.x	2.x	1.2.1	2.10、2.11	1.7、1.8
2.1.x	2.x	1.2.1	2.10、2.11	1.7、1.8
2.2.x	2.6.x、2.7.x	1.2.1	2.10、2.11	1.8

- 本次课程选择Spark2.2.0版本作为授课内容
  - 官网：http://spark.apache.org/
  - 帮助文档：http://spark.apache.org/docs/2.2.0/

SPARK2.X安装

- 安装依赖环境(Java、Scala、Hadoop、Hive、Kafka等)
- 下载spark安装包
- 解压安装包进行配置安装(过程和spark1.x版本类似)

Download Apache Spark™

1. Choose a Spark release: 2.2.0 (Jul 11 2017)
2. Choose a package type: Pre-built for Apache Hadoop 2.7 and later
3. Download Spark: spark-2.2.0-bin-hadoop2.7.tgz
4. Verify this release using the 2.2.0 signatures and checksums and project release KEYS.

spark-2.2.0-bin-hadoop2.7.tgz 201
spark-2.2.0-bin-hadoop2.7.tgz.asc 201
spark-2.2.0-bin-hadoop2.7.tgz.md5 201
spark-2.2.0-bin-hadoop2.7.tgz.sha 201

```
[beifeng@hadoop-senior01 softwares]$ ls
apache-hive-1.2.1-bin.tar.gz    spark-2.2.0-bin-2.7.3.tgz
apache-maven-3.3.9-bin.tar.gz   spark-2.2.0-bin-hadoop2.7.tgz
hadoop-2.7.3.tar.gz             spark-2.2.0.tgz
jdk-8u144-linux-x64.tar.gz      zinc-0.3.11.tgz
scala-2.11.8.tgz
[beifeng@hadoop-senior01 softwares]$ tar -zxvf spark-2.2.0-bin-hadoop2.7.tgz -C
 /opt/apache2.7/
```

SPARK2.X local&standalone环境配置

- 配置和spark1.x版本基本类似

spark-env.sh | spark-defaults.conf | slaves

```
16 # See the License for the specif
17 # limitations under the License.
18 #
19 JAVA_HOME=/opt/modules/java
20 SCALA_HOME=/opt/modules/scala
```

```
30 HADOOP_CONF_DIR=/opt/apache2.7/hadoop/etc/hadoop
31 SPARK_LOCAL_IP=hadoop-senior01.ibeifeng.com
```

```
59 SPARK_MASTER_HOST=hadoop-senior01.ibeifeng.com
60 SPARK_MASTER_PORT=7070
61 SPARK_MASTER_WEBUI_PORT=8080
62 SPARK_WORKER_CORES=3
63 SPARK_WORKER_MEMORY=3G
64 SPARK_WORKER_PORT=7071
65 SPARK_WORKER_WEBUI_PORT=8081
```

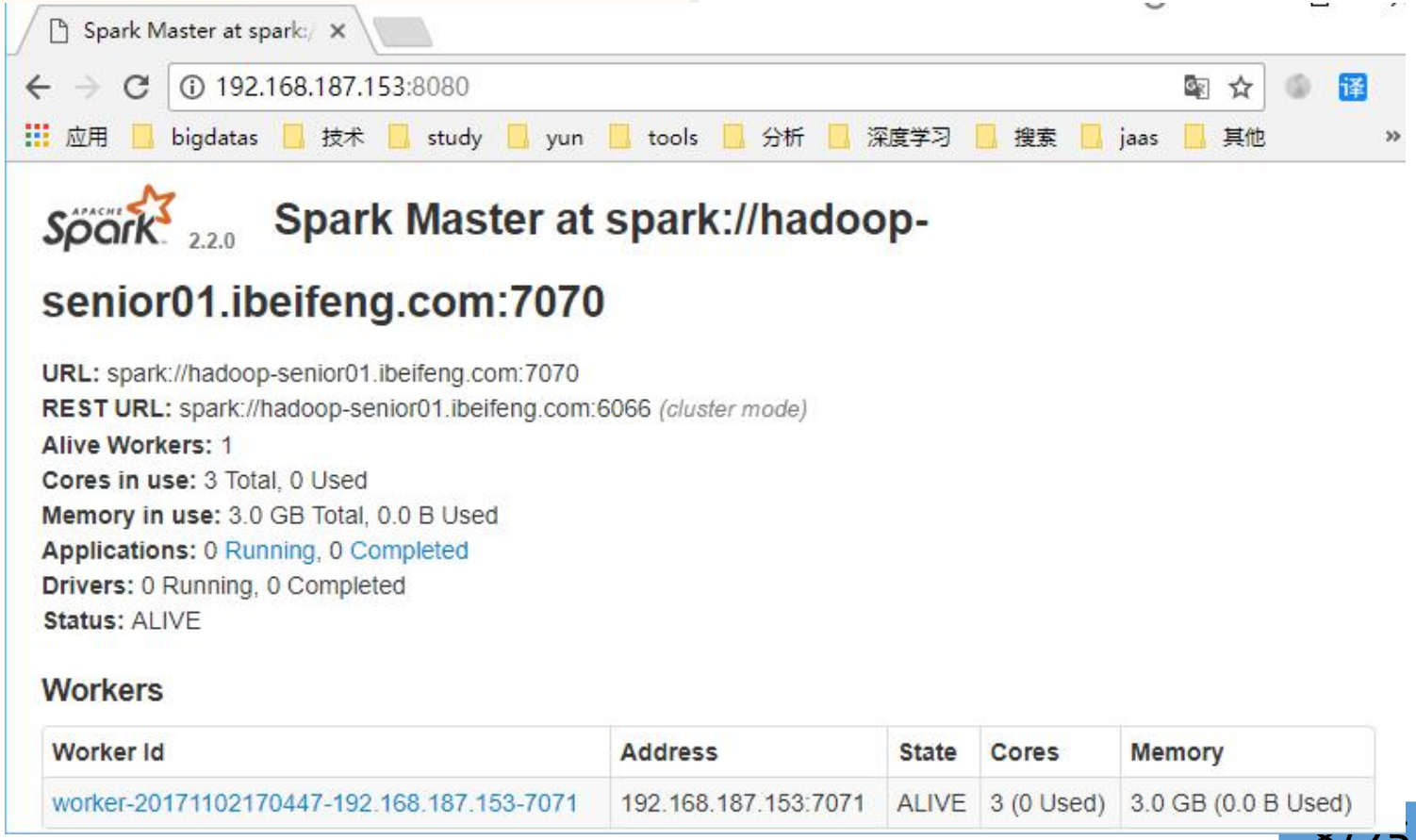
SPARK2.X local环境测试

```
[beifeng@hadoop-senior01 spark]$ bin/spark-shell
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val rdd = sc.textFile("/word.txt")
val wordCountRDD = rdd
  .flatMap(line => line.split(" "))
  .filter(word => word.nonEmpty)
  .map(word => (word,1))
  .reduceByKey(_ + _)
wordCountRDD.collect()

// Exiting paste mode, now interpreting.

rdd: org.apache.spark.rdd.RDD[String] = /word.txt MapPartitionsRDD[1] at textFi
le at <console>:24
wordCountRDD: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[5] at reduc
eByKey at <console>:29
res1: Array[(String, Int)] = Array((spark,7), (hive,2), (hadoop,1), (oozie,1),
(standalone,1), (flume,1), (mapreduce,2), (mesos,1), (hue,1), (yarn,1), (hdfs,2
```

SPARK2.X standalone环境测试01

```
[beifeng@hadoop-senior01 spark]$ sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/apache2.7/spark
/logs/spark-beifeng-org.apache.spark.deploy.master.Master-1-hadoop-senior01.ibe
ifeng.com.out
hadoop-senior01.ibeifeng.com: starting org.apache.spark.deploy.worker.Worker, l
ogging to /opt/apache2.7/spark/logs/spark-beifeng-org.apache.spark.deploy.worke
r.Worker-1-hadoop-senior01.ibeifeng.com.out
[beifeng@hadoop-senior01 spark]$
```



SPARK2.X standalone环境测试02

```
[beifeng@hadoop-senior01 spark]$ bin/spark-shell --master spark://hadoop-senior
01.ibeifeng.com:7070
```

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

val rdd = sc.textFile("/word.txt")
val wordCountRDD = rdd
  .flatMap(line => line.split(" "))
  .filter(word => word.nonEmpty)
  .map(word => (word,1))
  .reduceByKey(_ + _)
wordCountRDD.collect()

// Exiting paste mode, now int
```

```
scala> wordCountRDD.collect
res3: Array[(String, Int)] = Array((hive
,2), (oozie,1), (mapreduce,2), (hue,1),
(yarn,1), (sqoop,1), (spark,7), (hadoop,
1), (standalone,1), (flume,1), (mesos,1)
, (hdfs,2), (hbase,4))
```

SPARK2.X第三方jar文件依赖解决方案

- 直接将第三方的jar文件放到spark根目录下的jars文件夹中即可完成第三方jar文件的依赖，另外也可以尝试使用Spark1.6版本中介绍的第三方jar文件解决方案

SPARK2.X和Hive集成方式讲解

- 将hive的配置文件hive-site.xml放到spark根目录下的conf文件夹中，然后根据hive-site.xml文件中的参数项hive.metastore.uris采用不同的集成方式；当hive.metastore.uris参数配置项为空或者默认值的情况下，将元数据的第三方jar文件添加到classpath中；否则启动hive配置的metastore服务

```
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://hadoop-senior01.ibeifeng.com:9083</value>
</property>
```

```
[beifeng@hadoop-senior01 conf]$ hive --service metastore &
[1] 5063
[beifeng@hadoop-senior01 conf]$ ls: cannot access /opt/apache2.7/spark/lib/spark-assembly-*.jar: No such file o
r directory
Starting Hive Metastore Server
```

默认情况下，hive1.2.1版本会加载spark的依赖包；但是在2.x版本中，这个包是不存在的；所以这里有异常，没有关系

SPARK2.X版本中SparkSQL和Hive集成测试

```
[beifeng@hadoop-senior01 spark]$ bin/spark-shell
```

```
val rdd = sc.textFile("/word.txt")
val wordCountRDD = rdd
  .flatMap(line => line.split(" "))
  .filter(word => word.nonEmpty)
  .map(word => (word,1))
  .reduceByKey(_ + _)
wordCountRDD
  .repartition(1)
  .map(t => s"${t._1},${t._2}")
  .saveAsTextFile("/wc_result")
```

spark.sql("create table wc(word string, count int) row format delimited fields terminated by ','").show()

spark.sql("show tables").show

spark.sql("load data inpath '/wc_result/part-00000' into table wc").show

spark.sql("select * from wc").show

```
scala> spark.sql("select * from wc").show
+----------+-----+
|      word|count|
+----------+-----+
|     spark|    7|
|      hive|    2|
|    hadoop|    1|
|     oozie|    1|
|standalone|    1|
```

SPARK2.x_SparkCore模块新特性

- SparkCore模块中主要进行更改的新特性包括以下几个部分
  - 底层RDD执行流程优化(eg: 内存、磁盘、shuffle等机制的优化)
  - hash shuffle manager被删除，只有sort shuffle manager存在了
  - RDD中新增一些相关的API， eg: cartesian(笛卡尔积)等

SPARK2.x_SparkSQL模块新特性

- SparkSQL模块是Spark 2.X版本中新特性添加比较多个一个模块，主要包括以下内容：
  - 程序入口更改为SparkSession，但是还是可以使用SQLContext作为程序入口
  - Dataset取代DataFrame成为SparkSQL模块中的核心抽象
  - 内嵌支持csv文件格式数据的读取
  - 支持Hive的桶表的操作
  - 在1.x版本中HQL和SQL的语法是分隔开的，在2.x版本中HQL和SQL的语法合并，可以使用同一个入口来进行操作(SparkSession)
  - SparkSQL的执行支持堆外内存的优化
  - 性能优化，eg: 更改代码生成方式、提高Parquet和ORC格式文件数据操作的性能、提升窗口分析函数的执行效率等

SPARK2.x_SparkStreaming模块新特性

- Spark Streaming模块中在Spark 2.x版本中主要新增的特性包括以下几个方面
  - 新增结构化流式数据处理模块，将SparkStreaming和SparkSQL两个模块进行一个合并，形成一个新的模块: Spark Structured Streaming
  - Spark Streaming模块最大的一个更新就是支持Kafka 0.10版本的集成

SPARK2.x_Spark Structured Streaming

- Structured Streaming is a **scalable** and **fault-tolerant** stream processing engine built on the Spark SQL engine. You can express your streaming computation the same way you woule express a batch computation on static data. You can use the DataFrame or Dataset API to process streaming data.
- Structured Streaming provides **fast**, **scalable**, **fault-tolerant**, **end-to-end exactly-once** stream processing without the user having to reason about streaming.

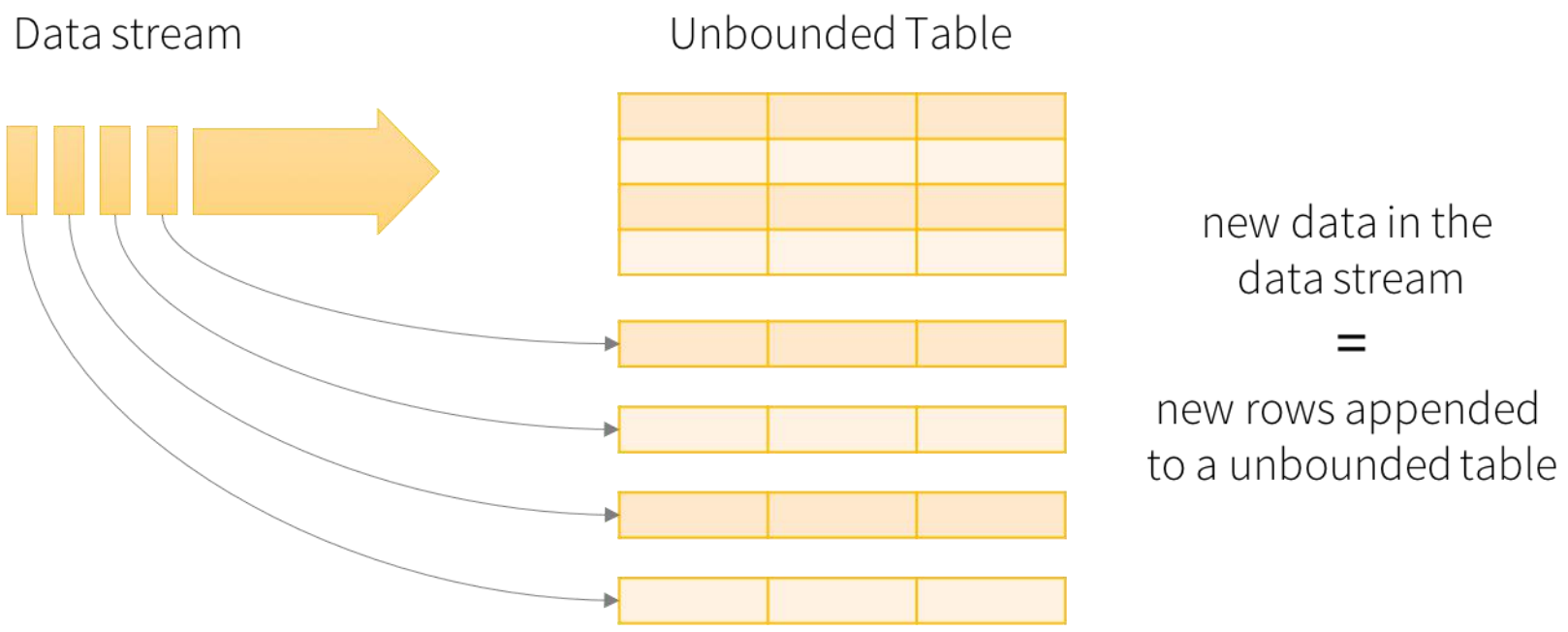
SPARK2.x_Structured Streaming Programming Model

- The key idea is Structured Streaming is to **treat to a live data stream as a table that is being continuously appended**. The leads to a new stream processing model that is very similar to batch processing model. You will express your streaming computation as standard batch-list query as on a static table, and Spark runs it as an **incremental** query on the **unbounded** input table.

SPARK2.x_Structured Streaming Programming Model

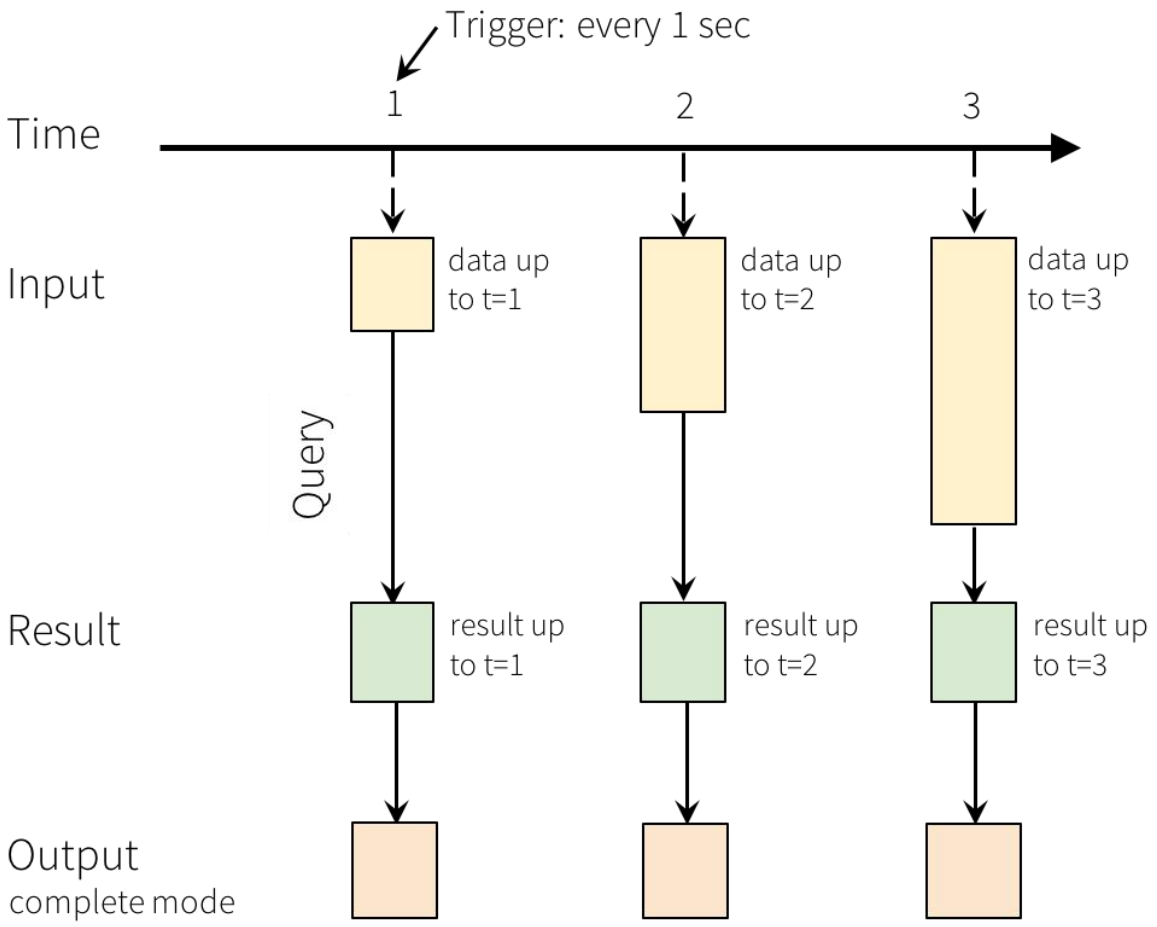
- 结构化流数据处理的核心是将动态数据流视作连续添加的一个表。这就导致新的数据流处理模型非常类似批处理模型。你可以在将流上的批次操作当做是静态表上进行操作，但是spark在运行你给定的流批次处理操作的时候，是将其当做一个运行在**无界表**上的**增量查询**操作。
- 核心：**无边界的表**以及**增量查询**

SPARK2.x_Structured Streaming Programming Model



Data stream as an unbounded table

SPARK2.x_Structured Streaming Programming Model

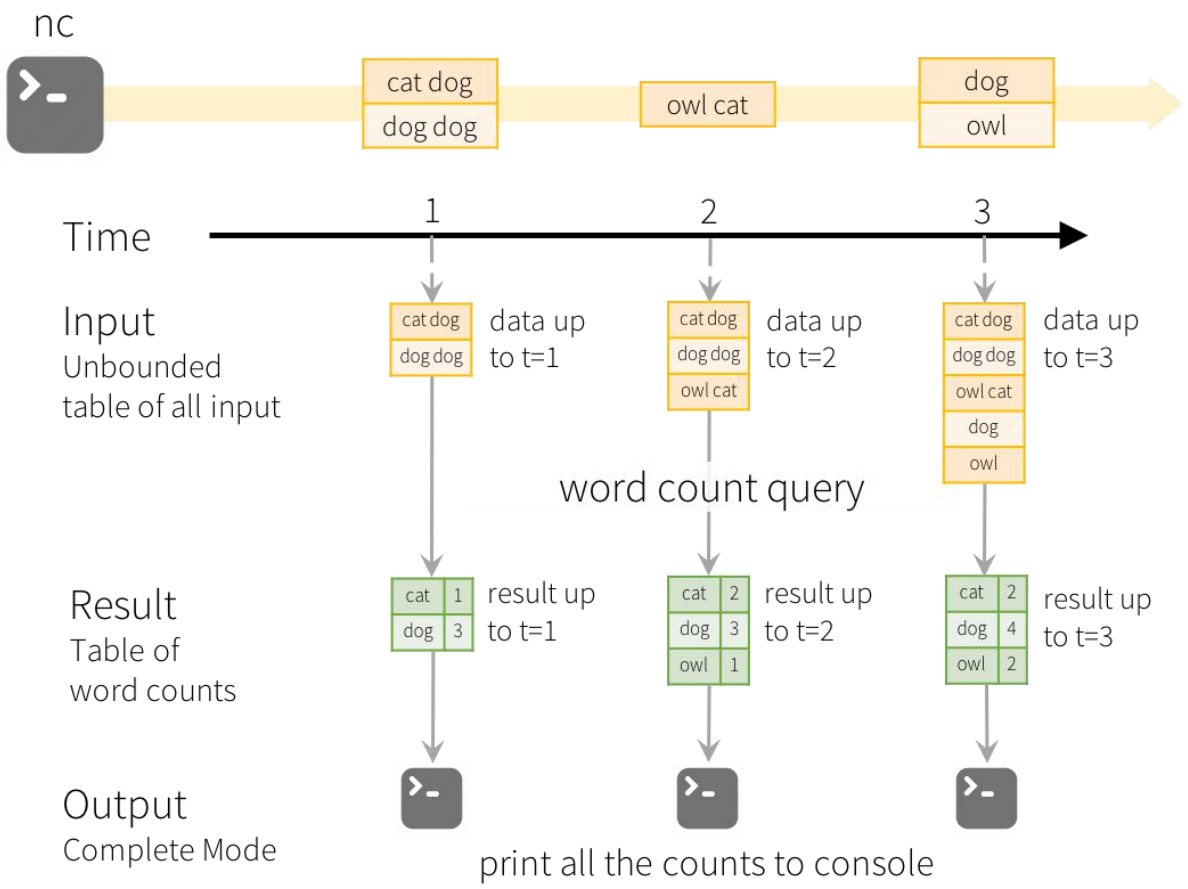


Programming Model for Structured Streaming

SPARK2.x_Structured Streaming Programming Model

- Output is defined as what gets written out to the external storage，The output can be defined in a different mode:
  - **Complete Mode**: The entried updated Result Table will to written to the external storage.
  - **Append Mode**: Only the new rows appended in the Result Table since the last trigger will be written to the external storage.
  - **Update Mode**: Only the rows that were updated in the Result Table since the last trigger will be written to the external storage.(>2.1.1)

SPARK2.x_Structured Streaming Programming Model



Model of the Quick Example

SPARK2.x_Structured Streaming Kafka

- Structured Streaming集成Kafka要求Kafka版本至少是0.10版本，相关参数详见链接：http://spark.apache.org/docs/2.2.0/structured-streaming-kafka-integration.html
- 除此之外，Structured Streaming只支持File和Socket两种数据源的输出，其它的均不支持。

SPARK2.x_Structured Streaming Unsupported Operations

- Structured Streaming是一个处于发展中的模块，还有很多不支持的功能，主要如下：
  - 多数据流的合并
  - Limit或者获取前N条数据的操作
  - Distinct去重操作
  - 在操作过程中不支持sort操作
  - 外连接在流数据和静态的Dataset对象之间是有条件的支持操作
  - count、foreach、show等API不支持

北風网 IBEIFENG.COM

THANK YOU

上海育创网络科技有限公司