Apache Flink 1.1.0和1.1.1发布，支持SQL

Apache Flink 1.1.0于2016年08月08日正式发布，虽然发布了好多天了，我觉得还是有必要说说该版本的一些重大更新。Apache Flink 1.1.0是1.x.x系列版本的第一个主要版本，其API与1.0.0版本保持兼容。这就意味着你之前使用Flink 1.0.0稳定API编写的应用程序可以直接运行在Flink 1.1.0上面。本次发布共有95位贡献者参与，包括对Bug进行修复、新特性添加以及维护性能，这些加起来有超过450个JIRA
issues。完整的issues列表参见（
http://flink.apache.org/blog/release_
1.1.0-changelog.html）。下面主要对本版本的一些重要特性进行介绍。

连接器(Connectors)

流连接器(streaming connectors)是Flink DataStream API非常重要的一部分。Flink 1.1.0版本添加了对新外部系统的支持，并且对现有的一些连接器进行了一些提升。

Continuous File System Sources

在Flink 1.0版本，社区经常收到用户希望能够监听文件夹，
并且持续地处理其中文件的特性。现在Flink 1.1通过FileProcessingMode支持这个功能了！

```
//////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2016-08-18
 Time: 23:38
 bolg: https://www.iteblog.com
 本文地址：https://www.iteblog.com/archives/1749
 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 过往记忆博客微信公共帐号：iteblog_hadoop
//////////////////////////////////////////////////////////////////////

DataStream<String> stream = env.readFile(
  textInputFormat,
  "hdfs:///file-path",
  FileProcessingMode.PROCESS_CONTINUOUSLY,
  5000, // monitoring interval (millis)
  FilePathFilter.createDefaultFilter()); // file path filter
```

上面的代码片段将会每隔5秒
监听hdfs:///file-path目录，详细请参见https://ci.apache.org/projects/flink/flink-docs-

release-1.1/apis/streaming/index.html#data-sources。

Kinesis Source and Sink

Flink 1.1添加了Kinesis connector，我们可以通过它消费(FlinkKinesisConsumer)Kinesis中的数据；同时我们也可以将产生的数据写入(FlinkKinesisProduer)到Amazon Kinesis Streams里面：

```
DataStream<String> kinesis = env.addSource(
  new FlinkKinesisConsumer<>("stream-name", schema, config));
```

详细文档请参见：https://ci.apache.org/projects/flink/flink-docs-release-1.1/apis/streaming/connectors/kinesis.html

Cassandra Sink

Apache Cassandra sink允许我们将Flink里面的数据写入到Cassandra。如果查询具有幂等性(idempotent)，Flink可以提供exactly-once保证，这意味着我们可以操作多次而结果并不会改变。

```
CassandraSink.addSink(input)
```

Table API and SQL

Table API是为关系流(relational stream)和批处理提供的类SQL表达式语言(expression language)，我们可以很容易地将它嵌入到Flink的DataSet和DataStream API中，包括Java和Scala语言。

```
//////////////////////////////////////////////////////////////////////
 User: 过往记忆
 Date: 2016-08-18
 Time: 23:38
 bolg: https://www.iteblog.com
 本文地址：https://www.iteblog.com/archives/1749
 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 过往记忆博客微信公共帐号：iteblog_hadoop
//////////////////////////////////////////////////////////////////////

Table custT = tableEnv
  .toTable(custDs, "name, zipcode")
  .where("zipcode = '12345'")
```

```
  .select("name")
```

初始版本在Flink 1.0已经可用。Flink 1.1版本社区投入了很多工作来重新对Table API进行架构，并且整合了Apache Calcite。

在这第一个版本(也就是Flink 1.1.0)，流上面的SQL(以及Table API)查询仅仅能进行select、filter以及union操作。和Flink 1.0相比，重新设计的Table API支持更多的scalar functions，支持从外部数据源读表数据，并且可以将计算好的结果写回到外部源：

```
Table iteblog = tableEnv.sql(
  "SELECT STREAM product, amount FROM Orders WHERE product LIKE '%Rubber%'");
```

更多关于Flink SQL的信息可以参见《Apache Flink：Table API和SQL发展现状概述》：https://www.iteblog.com/archives/1691

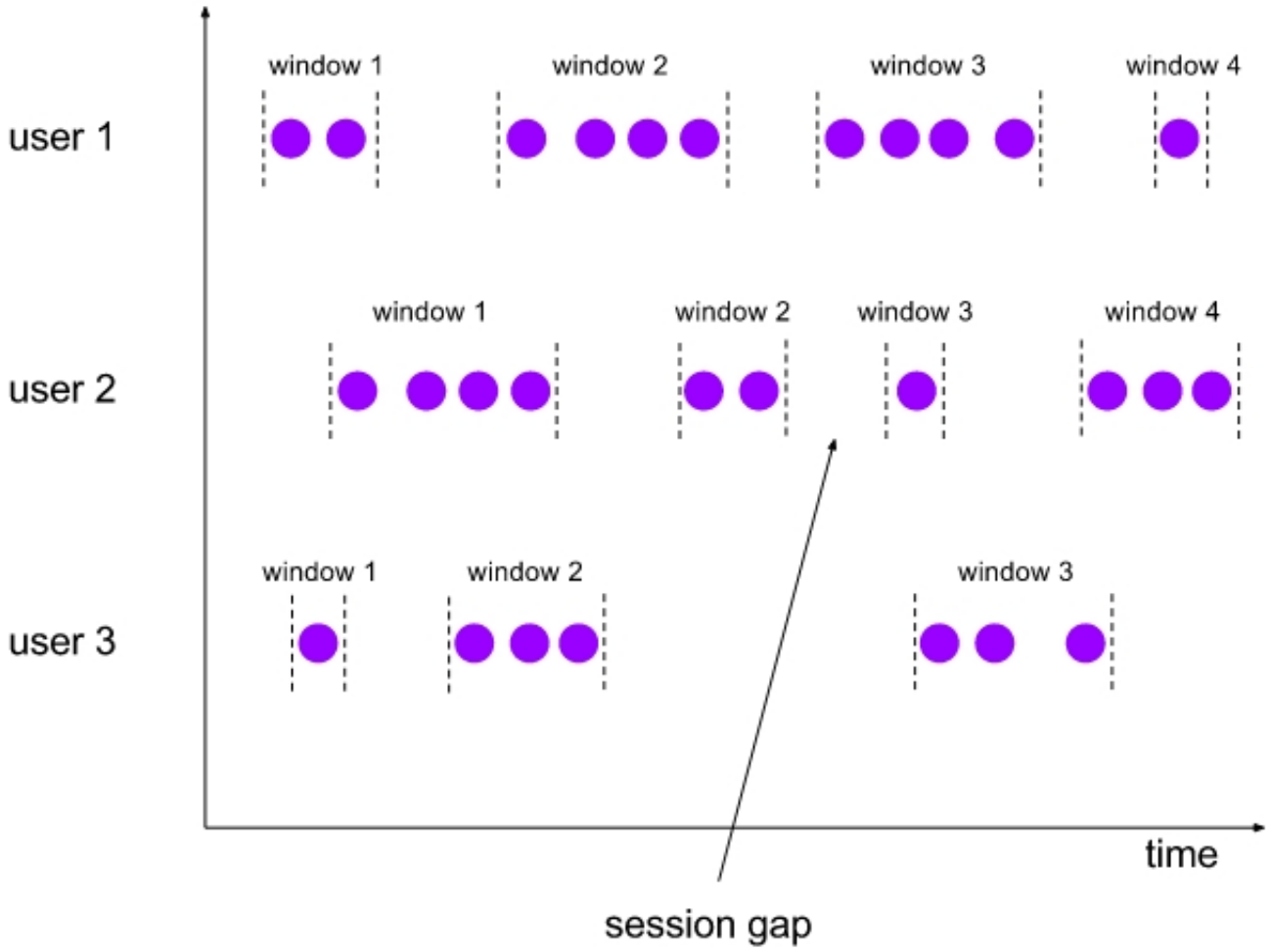
DataStream API

DataStream API现在公开了 session windows，并且允许延迟(allowed lateness)成为一等公民。

Session Windows

Session windows的应用场景是窗口范围内的数据需要根据输入的数据进行调节。这种机制使得我们拥有一个窗口，该窗口对每个key都会创建单独的点，如果一段时间内该点没有活动则会被删除。我们可以配置session的间隔，该值设定了每个session最多等待新的数据多长时间，超过了该session将会被关闭。

```
input.keyBy(<key selector>)
    .window(EventTimeSessionWindows.withGap(Time.minutes(10)))
    .<windowed transformation>(<window function>);
```



如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

Support for Late Elements

现在你可以指定windowed转换如何处理延迟的数据，以及允许延迟多长时间。该参数称为allowed lateness，指定了元素最对可以延迟多久：

```
input.keyBy(<key selector>).window(<window assigner>)
    .allowedLateness(<time>)
    .<windowed transformation>(<window function>);
```

元素只要在允许延迟的时间范围内仍然会被放入到Windows中，并且在计算结果的时候使用到。但是如果元素超过了允许延迟的时间，此元素将会被删除。

复杂事件处理(CEP)的Scala API

Flink 1.0版本为CEP增加了一些初始版本的类库。该类库的核心是Pattern API，其允许我们指定一个模式来匹配当前的事件流(event stream)。然而在Flink 1.0版本此API仅仅适用于Java；值得高兴的是，在Flink 1.1版本，社区为Scala语言提供了同样的API，允许我们以一种更加简洁的方式来指定你的事件模式(event patterns)。更多细节参见官方博客和其他相关CEP文档。

Metrics

Flink新的metrics system允许我们很容易地收集和暴露我们应用系统中的metrics到外部系统中。我们可以通过runtime context为应用程序添加counters、gauges以及histograms。

```
Counter counter = getRuntimeContext()
  .getMetricGroup()
  .counter("my-counter");
```

所有注册的metrics将会通过reporters来暴露。Flink内置支持JMX, Ganglia, Graphite以及statsD。即使是我们自定义的metrics，Flink也暴露出很多内部的metrics，比如checkpoint sizes以及JVM stats.

Apache Flink 1.1.1

Apache Flink 1.1.1发布距Apache Flink 1.1.0不到三天。其原因是Maven仓库的Apache Flink 1.1.0有一个Hadoop依赖问题。里面，没有部署Hadoop 1版本，而1.1.0artifacts却依赖了Hadoop 1。所有建议所有使用Flink 1.1.0的用户升级到1.1.1版本，如下：

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-java</artifactId>
  <version>1.1.1</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java_2.10</artifactId>
  <version>1.1.1</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-clients_2.10</artifactId>
  <version>1.1.1</version>
</dependency>
```