- 迟到数据，数据是否有序处理的问题。watermark
- CEP
- 事件时间（Event time）与process time的区别
- taskmanager 向 jobmanager进行注册，报告当前的taskmanager资源情况给jobmanager
- TaskManager中使用task slot来进行资源隔离，类似yarn中的container。但task slot仅是在内存方面的隔离，core是隔离不了的，多个task slot 共享cpu。一般是设置task slot和core的个数是1:1的关系，但core如果是超线程，那么slot：core可以2:1
- JobManager掌握整个集群资源情况，清楚集群资源情况对flink来说很重要
- jobmanager:资源调度、计算任务调度（这个时候触发集群进行checkpoint，jobmanager发送命令让taskmanager保存状态）
- taskmanager：资源调度，计算任务执行
- 上面两个角色是管理着计算层和资源层两件事情的，这个和spark有所区别
- 在这里配置环境变量?

```
vim ~/.bashrc
```

- 修改flink-conf.yaml
- 相同的key为一组处理数据是同一个线程
- flink提交application有两种方式：1.web ui 2.命令行

Flink集群搭建

- flink shell : 简单测试集群健康程度
- jobmanager恢复所有元数据信息，保存在hdfs. 因为是有状态计算，所有元数据信息量大，不能保存在zookeeper

```
high-availability.storageDir:hdfs://node01:9000/flink/ha
```

- 每提交一个任务，可以产生一个小型的flink集群，这是我不能理解的。yarn资源调度可以产生多个小型flink集群，每个flink集群都会有taskmanager和jobmanager的
- 果然flink集群是可以多个的，这和我们现在做的flink manager平台吻合

Flink Job 运行

- 语义：Job、Task（相当于stage）、subtask（并行度），算子链，shuffle
- 并行度就相当于spark中的partition，这玩意可以决定Task
- 多个operator chain 组成一个task
- 算子链是一个重要的概念，决定是几个Task
- 业务逻辑复杂并行度Parallelism可以设置高点
- 一个task slot是可以执行多个线程的

- Task调度规则：1.不同Task的SubTask要分发到同一个TaskSlot里面；2.相同Task的SubTask要分发到不同的TaskSlot
- TaskSlot的个数决定了未来job的并行度。eg：总共的taskslot 4，Parallelism 设置5，则运行失败
- 可以通过上下游的Partition数量（Parallelism）看出是否需要shuffle

算子

- kafka强顺序一致性，只能设置一个Partition，这样就失去分布式消息队列的效果了

DataSource

Transformation

- keyBy经常和Reduce使用，与spark reduceByKey不同，keyBy后的数据形式是这样的(k,v1),(k,v2),(k,v3)，而不是spark这样（k,[v1,v2,v3]）
- setParalism()设置很大，多个线程，对keyBy、reduce数据会有什么影响吗？-> 相同key的数据仅仅会由一个subtask（线程）去处理

Sink

checkpoint和savepoint

- at-most-once 至多一次，想象这样一种场景，shuffle的时候发生网络异常数据丢失了怎么办？或者换句话说，怎么保证exactly once?
- checkpoint：读取数据源的一条数据，这条数据让整个链路处理完，才做这条数据的持久化的checkpoint
- 数据源必须支持重复消费，不然还是不能保证exactly once
- exactly once 缺点：一条条处理是不是太慢了？引入barrier
- 引入barrier后，实际上并不能保证exactly once了，出现异常是产生了重复计算了的
- checkpoint保存的到底是什么？ 算子计算的结果、数据源的offset
- 总结：exactly once 是需要两个前提条件的：1.数据源的可重复消费；2.sink 需要幂等或者支持事务
- checkpoint是异步的
- savepoint不过就是用户以手动方式去触发checkpoint罢了

CEP

- 什么是事件（Event）？理解成一条条数据即可
- CEP是flink的一个库
- 事件（数据）的规则匹配：Pattern
- 复杂事件：事件与事件的组合
- 状态编程和窗口代替CEP？ YES

Pattern

- 创建数据流、Pattern的定义、模式检测（Pattern应用在数据流中）、选取结果

Flink训练营（Flink 1.11）

- unaligned checkpoint 对齐时间？做checkpoint需要的缓冲时间?
- watermark idleness detection
- 批计算：stage by stage
- 所谓的批计算和流计算，不过就是最小粒度数据凑成一堆再去计算， 还是每一个最小粒度数据都可以去计算出结果（每条记录触发计算，某一段时间/计算触发计算）
- 流批本质不过是计算引擎触发规则的不一样罢了，看用户需求的业务延迟是多少
- 批是流的特例：Native-Streaming
- 流是批的特例：Micro-Batching
- 两者之间计算底层的架构就不一样了
- 流计算：快！所解决的核心问题：（1）延时问题（early-fire机制：就是来一条数据触发一次计算）；（2）计算撤回；（3）容错续跑,state+checkpoint；（4）透明升级,state+savepoint；（5）乱序问题，event time + watermark；（6）正确性问题，exactly-once，At-least-once;(7)部署问题和弹性扩容
- Flink场景，事件驱动型应用，数据分析型应用，数据管道型应用（ETL）