分布式流计算框架

摘要

该项目设计成为一个处理通用数据,分布式的,高扩展性的,容错的流数据计算框架.可方便应用在各种建立在流式数据处理的场合,提供持续的高效的流数据处理服务. 系统主要通过管道(processing element. PE)处理数据流.PE 在接到数据流后对数据进行处理:

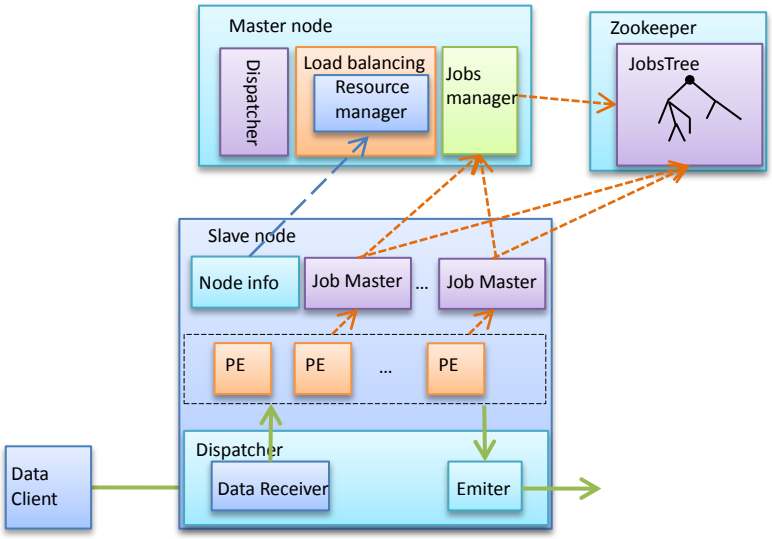
1. 生成需进一步数据的数据发送给后续 PE 模块.
2. 生成数据结果发送给相关应用程序.

该项目借鉴了基于 MapReduce 的分布式计算框架的经验.采取 primary-slave 模式,分为 Master 和 Snode (slave node)两种节点.Master 节点主要负责系统资源的调度与分配.Snode 节点主要复制具体的流数据任务的整个运行周期的管理和跟踪.

数据流处理主要通过任务 Job 的形式运行在 Snode 节点中.

1. 系统结构

1.1. Master node &Slave node&zookeeper



系统主要分为 Maseter Node,Slave Node,Client ,Zookeeper4 大部分:

Master Node:

1) Dispatcher: 负责接收客户端创建不同数据处理应用(job)的请求并指派某个 Slave

node 作为 Job-master.

2) Load balancing & resource manager: 负责接收所有 slave Node 的当前状态汇报,包括 cp,memory,job account 等.在 job-master 请求创建后续 PE 运行节点时,根据负载均衡算法分配合适的运行节点.
3) Job manager: 负责所有 Job 的基本状态维护,提供 Job 查询状态查询接口等运维相关接口.

Slave Node:

1) Node info: 负责向 master node 汇报当前节点的 cpu,memory,job account 信息
2) Job master: 负责单个 Job 的整个运行周期的维护.包括 job 中的各个 PE 的创建,PE 状态维护.
3) PE: 根据定义处理接收的数据,生成后续数据发送给后续 PE 或者提交结果.
4) Dispatcher:
   a) Data receiver: 接收客户端发送的数据,根据数据 key 发送给相应 Job 的 PE 模块处理.
   b) Emiter: 接收 PE 的发送请求,发送数据给后续 PE 或者提交结果.

Client: 创建流数据处理 Job,接收数据源发送的数据, 分发数据.

Zookeeper: 维护所有 Job 的状态信息.

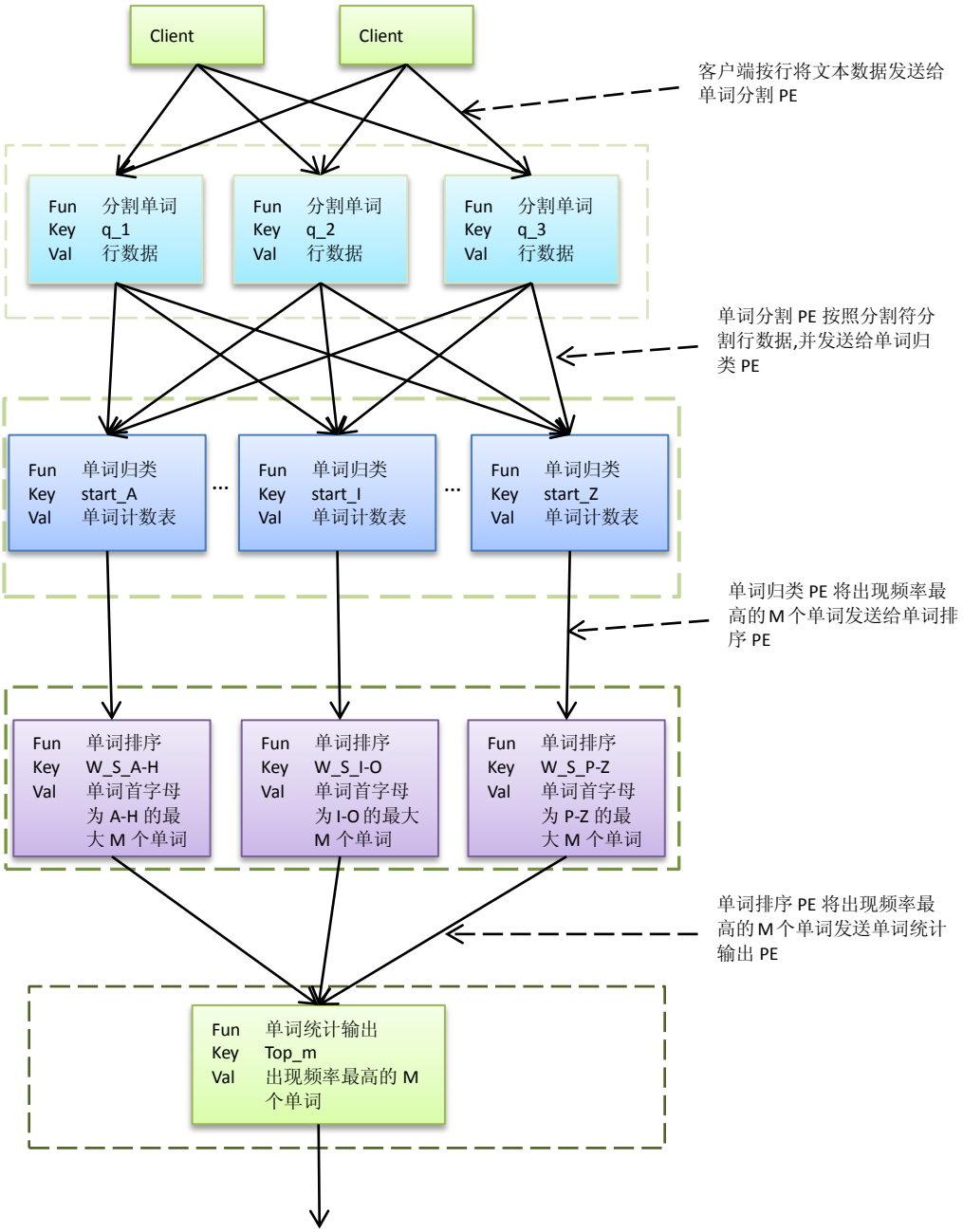
1.2. Job, Job-master,PE(Task)

流数据处理分析是通过 Job 的方式运行,Job 由 Job-master 模块和一个或多个 PE 模块组成.每个 Job 在运行周期内由 Job-master 模块进行管理,跟踪,调度,容错,由各个 PE 模块组成数据处理 chain 处理所有数据.Job 能显示的被开启,暂停,结束.Job 的状态包括,prepare,running, succeded,failed,killed.

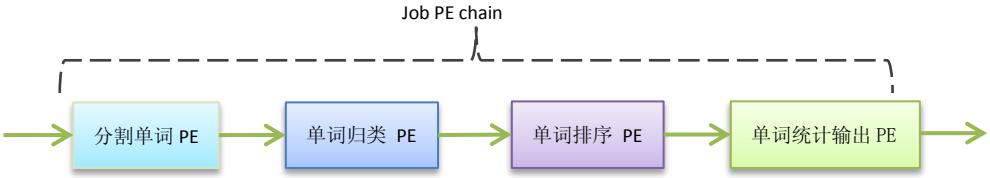
1) Prepare: job-master 根据客户端请求的流数据处理任务生成 PE 链.
2) Running: PE 开始处理流数据.
3) succeded,failed,killed: Job 终结时的运行结果.

例如单词统计场景,即分析某段文本中 N 个单词的出现频率最高的 M 个单词的统计任务:

1) 客户端按行依次将整段文本发送给单词分割 PE
2) 单词分割 PE 解析行数据,分割单词数据,发送给单词归类 PE
3) 单词归类 PE 根据单词的首个字母发送给单词排序 PE
4) 单词排序 PE 定时将出现次数最高的 M 个单词发送给单词统计 PE
5) 单词统计 PE 接收数据,统计单词出现次数最高的 M 个单词,并输出结果.

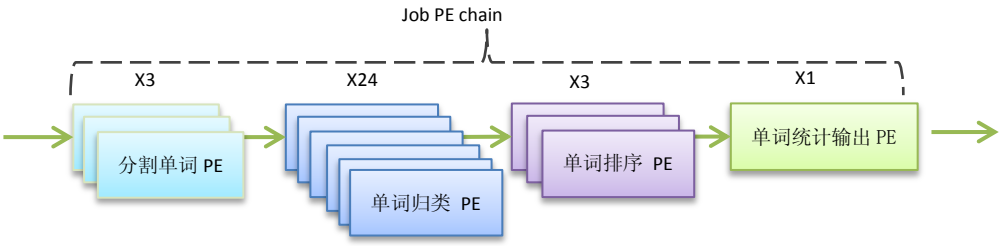


在整个单词统计过程中,整个 Job 包含一条 PE 链:分割单词 PE,单词归类 PE,单词排序 PE,单词统计输出 PE. 数据流依次通过该 PE 链,最终由单词统计输出 PE 输出结果.

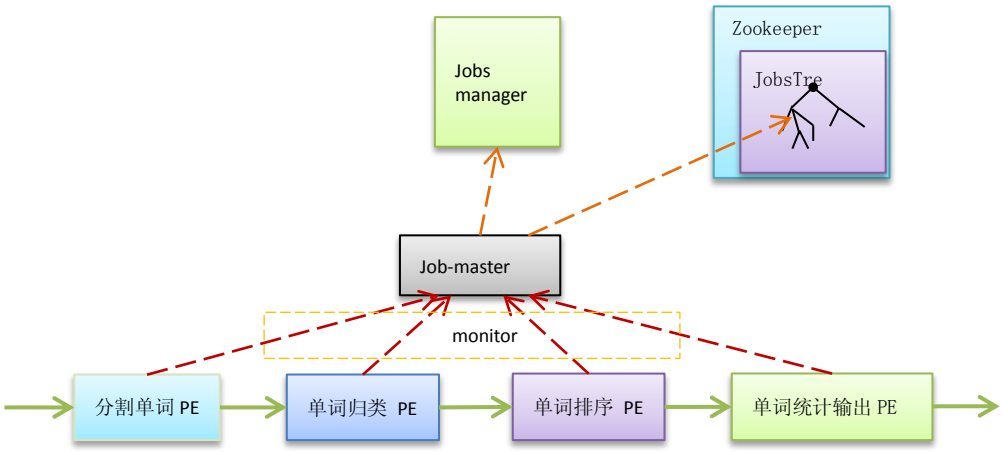


在 Job 运行过程中,PE 链中的任意一类型 PE 都能分布式运行多个实例,数据流通过简单或复杂的均衡算法从客户端到首个 PE 或者从 PE 到下个 PE.例如,在上述单词统计 Job 中:

1) 作为首个从客户端接收数据的 PE(分割单词),考虑到大量文档的分割需要较多的 CPU 资源,Job 可创建多个分割单词实例分布式运行在多个 node 中.客户端在 Job 开始运行前就已获取到了分割单词 PE 的列表,客户端可通过简单的随机分配算法将文档发送给不同的分割单词 PE.
2) 作为单词归类 PE,考虑到单词的总量是极其庞大的,Job 可通过简单的根据单词首个字母进行归类,总共创建 24 个单词归类 PE.分割单词 PE 的各个实例,在将文档中的单词分割完成后,将单词发送给相对应的单词归类 PE,由单词归类 PE 进行计数和维护.
3) 作为单词排序 PE,考虑到排序操作将耗费系统资源,Job 可根据单词首字母创建几个单词排序实例,例如实例 W_S_A-H 接收所有单词首字母从 A 到 H 的单词归类 PE 的各个实例,单词归类 PE 的各个实例定时向特定的单词排序 PE 实例发送 Top M 个单词序列.单词排序 PE 接收数据后,合并后排序.
4) 作为单词统计输出 PE,接收所有单词排序 PE 实例的数据,并排序输出 Top M 个单词的序列.

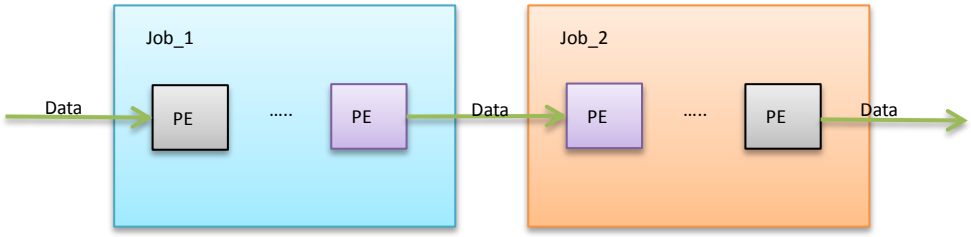


在整个 Job 运行周期中,所有 PE 的运行状态由 Job-master 统一管理和分配.Job-master 对所有 PE 进行监控和跟踪.Job-master 将本 Job 中的各个 PE 实例树的分布状况定时汇总到 zookeeper 中,并向 Jobs-manager 汇报总体状态.



通常 Job 中的数据沿着 PE chain 流动,各个 PE 独立完成相关功能,但有时数据需要多个 Job

一起完成指定分析处理,这就需要通过创建相应的 PE 传递数据.



1.3. 容错

系统提供多种特性支持数据流有效的运行:

1. 对于输入的每一份数据,可通过配置来指定是否需要保证该数据运行的完整性和正确性.

2.

1.4. Other

2. 数据结构