Saturn作业调度算法

1.设计方案

方案的基本原理是给每个作业分片一个负载(load)值和优先执行结点(prefer
list)，当需要重新分片时，参考作业优先设定和执行结点的负载值来进行域内执行结点之间的资源分配，从而达到资源平衡。

1.1 平衡算法

前置条件：

- a. 每个作业引入负载(load)属性，由用户通过Saturn
  UI界面输入；比如一个作业分片是3，作业负载(load)是20，则每个作业分片的负载(load)都是20。
- b. 为每一个作业引入一个新的属性prefer list（优先列表，或者叫预分配列表），由管理员通过ui界面编辑。
- c. 作业引入启用状态(enabled/disabled)，用户通过saturn
  ui界面操作来修改这个状态（启用/禁用按键）；启用状态的作业会被执行结点执行，禁用状态的作业不会被执行；启用状态的作业不可编辑（不可调整分片大小，不可调整prefer list)、删除，禁用状态的作业可编辑、删除。

实施步骤：

第一步，摘取(摘掉需要重新分配的作业分片，摘取方法见1.2)
第二步，放回(将这些作业分片按照负载从大到小顺序逐个分配给负载最小的执行结点)

放回算法具体描述如下：

- a. 构造需要添加的作业分片列表，我们起名为待分配列表，长度为n，
  待分配列表按照负载(load)从大到小排序，排序时需保证相同作业的所有分片是连续的
- b. 构造每种作业类型的exetuor列表(如果有prefer list，且有存活，则该作业的executor列表就是prefer
  list)，得到一个map<jobName,executorList>
- c. 从待分配列表中依次取出第0到第n-1个作业分片jobi,对于每个jobi
- d. 从map中取出可运行jobi的exector列表listi
- e. 将jobi分配给list1中负载(load)总和最小的executor

1.2 调度器实现

调度器基于zookeeper监听以下事件类型：
executor上线，executor下线，作业启动(状态由stopped变为started)，作业停止(状态由started变为stopping)
处理完之后，将调度结果保存，并使用zk通知各个作业的scheculer进行认领（代替原有shard）。

1.2.1 executor上线

摘取：

第一步，找出新上线结点的全部可执行作业列表；对于每个作业，判断prefer
list中是否包含了新上线的结点；如果是，则摘取其全部分片；这些已经处理过的作业称为预分配作业；

第二步，从新上线结点的作业列表中减去预分配作业，然后使用以下的方法依次摘取：

- 假如上线的executor为a,它能处理的作业类型为j1,j2(已减去预分配列表)。遍历当前域下的executor列表，拿掉全部作业类型为j1,j2的分片，加上尚未分配的j1,j2作业分片列表，作为算法的待分配列表。
- 在处理每个executor结点时，每拿掉一个作业分片后判断被拿掉的负载(load)是否已超过了自身处理前总负载(load)的1/n(n为当前executor结点的总数量)，如果超过，则本执行结点摘取完成，继续处理下一个执行结点；如果不超过则继续摘取，直到超过(大于等于)为止。

放回：

使用平衡算法逐个处理待分配列表中的作业分片。

1.2.2 exectuor下线

摘取：

取出下线的exectuor当前分配到的全部作业分片，作为算法的待分配列表

放回：

使用平衡算法逐个处理待分配列表中的作业分片。

1.2.3 作业启动

摘取：

从所有executor中摘取将被启动作业的全部分片作为算法的待分配列表

放回：

使用调整后的平衡算法放回

1.2.4 作业停止

摘取：

将被停止的作业分片从各执行结点删除

放回：

无

1.3部署架构

2. 可能的问题

2.1 生效不同步

每次做完重新分片之后，不会立刻生效（生效延迟），开始生效之后不会全部作业同时生效（部分作业先生效，部分作业后生效）。
于是在分片未完全生效之前，执行结点之间可能存在比较显著的资源不均衡。如果执行结点本身负载较重，可能会导致负载过重，甚至导致crash。
解决方案：等待全部作业全部分片执行完毕才开始新的分片（可使用开关控制）

2.2 分片触发与执行并发性问题

分片事件不断触发，在应用新分片的过程中，有新的分片事件发生。
解决方案：引入更新标识新，当分片发生时设true，在应用新分片开始时设false,结束时判断为true需要立刻再做一次。