(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 108762921 B
(45)授权公告日 2019.07.12

(21)申请号 201810486385.5

(22)申请日 2018.05.18

(65)同一申请的已公布的文献号
申请公布号 CN 108762921 A

(43)申请公布日 2018.11.06

(73)专利权人 电子科技大学
地址 611731 四川省成都市高新区(西区)西源大道2006号

(72)发明人 田文洪 叶宇飞 王金 许凌霄 匡平

(74)专利代理机构 电子科技大学专利中心 51203
代理人 邹裕蓉

(51)Int.Cl.
G06F 9/50(2006.01)
G06F 9/48(2006.01)

(56)对比文件
CN 107609141 A,2018.01.19,说明书第[0127]-[0134]段.
CN 105243155 A,2016.01.13,全文.
CN 105868019 A,2016.08.17,全文.
US 9842000 B2,2017.12.12,全文.
US 2018074852 A1,2018.03.15,全文.
杨志伟、郑烩、王嵩、杨坚、周乐乐.异构Spark集群下自适应任务调度策略.《计算机工程》.2016,第42卷(第1期),

审查员 李爽

权利要求书2页 说明书5页 附图2页

(54)发明名称

一种Spark集群系统的在线优化分区的任务调度方法及装置

(57)摘要

本发明公开了一种Spark集群系统的在线优化分区的任务调度方法,属于在线集群资源调度技术领域。本发明所述方法包括步骤:统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量以及用于拉取数据的内存比例;根据输出数据量与用于拉取数据的内存容量的大小关系,计算任务执行的轮数,并设置优化的分区数量;监测计算节点的平均CPU利用率、内存利用率,对各计算节点的资源利用水平进行评估;降序排序所有节点的资源利用水平,优先调度任务给资源利用水平最高的节点执行;重复上述步骤直到所有任务调度完成。本发明能够自动配置优化的分区数量,提高集群的资源利用率,加快Spark作业的执行速度。

CN 108762921 B

1.一种Spark集群系统的在线优化分区的任务调度方法,其特征在于,包括以下步骤:

步骤1.统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量和用于拉取数据的内存比例;

步骤2.根据输出数据量与用于拉取数据的内存容量的大小关系,计算任务执行的轮数,并设置数据的分区数量;

步骤3.监测计算节点的平均CPU利用率和内存利用率,对各计算节点的资源利用水平进行评估;

步骤4.降序排列所有节点的资源利用水平,优先调度任务给资源利用水平最高的节点执行;

步骤5.重复步骤3-步骤4,直至所有任务调度完成;

步骤1中,上游Stage输出数据量大小$S=\sum_{i=1}^{n} s_i$,其中,s_i为第i个计算节点上任务输出数据量大小,$1 \leqslant i \leqslant n$,n为计算节点的数量;参与作业执行的总CPU核数$Core_{app}=\sum_{i=1}^{n} CPU_{app(i)}$,其中,$CPU_{app(i)}$为第i个计算节点上用于应用程序的CPU核数;内存总量$Memory_{app}=\sum_{i=1}^{n} Mem_{app(i)}$,其中,$Mem_{app(i)}$为第i个计算节点上用于应用程序的内存容量;用于拉取数据的内存比例α从Spark参数配置文件中读取;

步骤2中,用于拉取数据的内存容量的大小为$Memory_{app}\times\alpha$,将输出数据量S与用于拉取数据的内存容量$Memory_{app}\times\alpha$进行大小比对:

当$S \leqslant Memory_{app}\times\alpha$时,任务执行的轮数r=1,数据的分区数为$Core_{app}$;

当$S > Memory_{app}\times\alpha$时,任务执行的轮数r=ceil(S/($Memory_{app}\times\alpha$)),其中ceil表示向上取整,数据的分区数为$Core_{app}\times r$。

2.根据权利要求1所述的Spark集群系统的在线优化分区的任务调度方法,其特征在于,步骤3中,为了减少计算节点CPU利用率或内存利用率的抖动误差,引入控制系统理论中的一种负反馈调节机制,CPU利用率或内存利用率的负反馈调节机制为:

$$U_i(t_j)=\begin{cases} U_i(t_{j-1})+\sqrt{U_i'(t_j)-U_i(t_{j-1})}, & j\geq 1 \\ U_i'(t_j), & j=0 \end{cases}$$

其中,与为第j个时刻,j为≥0的整数,$U_i(t_j)$为当前时刻第i个计算节点的CPU利用率或内存利用率,$U_i(t_{j-1})$为上一时刻的第i个计算节点的CPU利用率或内存利用率,$U_i'(t_j)$为当前时刻第i个计算节点的CPU或内存利用率的监测值。

3.根据权利要求2所述的Spark集群系统的在线优化分区的任务调度方法,其特征在于,步骤4中第i个计算节点的资源利用水平RL_i的计算公式为:

$RL_i=AvailableCores_i\times Speed_{cpu}\times(1-R_{i,cpu})\times(1-R_{i,mem})\times(1-E_i)$

其中,$AvailableCores_i$为第i个计算节点的可用CPU核数,$Speed_{cpu}$为计算节点CPU的主频大小,$R_{i,cpu}$为第i个计算节点的当前CPU利用率大小,$R_{i,mem}$为第i个计算节点的当前内存利用率大小,E_i为第i个计算节点的历史CPU利用率的熵值,反映CPU利用率的波动;第i个计

算节点的历史CPU利用率的熵值$E_i=-(c1\times\log_2 c1+c2\times\log_2 c2)$，其中，c1为历史CPU利用率中CPU利用率值大于等于CPU平均利用率的次数，c2为历史CPU利用率中CPU利用率值小于CPU平均利用率的次数。

4.根据权利要求3所述的Spark集群系统的在线优化分区的任务调度方法，其特征在于，任务分配时主要采取贪心策略，首先按照资源利用水平降序排列所有计算节点，然后遍历所有任务，将任务分配到资源利用水平最高的计算节点上，如果该计算节点的可用CPU核数大于每个任务需要的CPU核数，默认为1核，则在当前计算节点上分配该任务，同时更新该计算节点的可用CPU核数，任务后续将在该计算节点上上以最大的数据本地性运行；如果任务需要分配多轮，则重复步骤3-4直到所有任务分配完成。

5.一种利用权利要求1所述方法进行Spark集群系统的在线优化分区的任务调度装置，其特征在于，包括：

(1)信息收集模块：统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量和用于拉取数据的内存比例；

(2)分区优化模块：根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置数据的分区数量；

(3)节点监测模块：监测计算节点的平均CPU利用率和内存利用率，对各计算节点的资源利用水平进行评估；

(4)节点排序模块：按照资源利用水平降序或升序排列所有节点；

(5)任务分配模块：任务分配时主要采取贪心策略，优先调度任务给资源利用水平最高的节点；

(6)任务执行模块：用于按照任务分配模块决定的顺序执行任务。

一种Spark集群系统的在线优化分区的任务调度方法及装置

技术领域

[0001] 本发明属于在线集群资源调度技术领域，具体涉及一种Spark集群系统的在线优化分区的任务调度方法及装置。

背景技术

[0002] Spark是一个以可靠、高效、可伸缩的方式对大量数据进行分布式处理的内存计算框架。Spark集群主要的构件部署分为Spark Client、SparkContext、ClusterManager、Worker和Executor等，如图1所示。Spark Client用于用户提交应用程序到Spark集群，而SparkContext用于负责与ClusterManager进行通信，进行资源的申请、任务的分配和监控等，负责作业执行的生命周期管理。ClusterManager提供了资源的分配与管理，在不同的运行模式下，担任的角色有所不同。当SparkContext对运行的作业进行划分并分配资源后，会把任务发送到Worker节点上的Executor进行运行。

[0003] 许多应用程序的配置参数会影响Spark的执行性能，例如任务并行性，数据压缩以及Executor的资源配置。在Spark中，输入数据或中间数据被分成多个逻辑分区，集群中的每个计算节点处理一个或多个分区。用户可以配置每个Spark作业的分区数量和数据分区方式，不理想的任务分区或选择非最佳分区方案可能会显着增加工作负载的执行时间。例如，如果分区策略在计算阶段内启动太多任务，则会导致CPU和内存资源的竞争，从而导致性能下降。相反，如果启动的任务太少，系统的资源利用率会降低，同样会导致性能下降。

[0004] Spark为用户提供了两种控制任务并行性的方法。一种方法是使用配置参数spark.default.parallelism，该参数为未指定分区数量时使用的默认任务数。第二种方法是使用重新分区API，它允许用户重新分区数据。Spark不支持在不同计算阶段之间更改数据并行性，除非用户在程序中手动通过重新分区API更改分区数。此种方法具有很大的局限性，因为分区的最佳数量可能受数据大小的影响，每次处理不同数据集时，用户都必须更改分区数并重新编译程序。

[0005] 此外，不合适的任务调度方法可能会拖慢作业的执行进度，导致某些任务的完成时间比其他任务长得多。由于数据处理框架通常在计算阶段之间具有同步屏障，所以让处于同一阶段中的所有任务同时完成是至关重要的，以避免短腿任务阻碍其他运行速度快的任务。正确的数据分区方案以及任务调度方法是从底层硬件资源中提高性能的关键，但是，找到一个能够提供最佳性能的数据分区方案以及任务调度方法并非易事。这是因为，数据分析工作流程通常涉及复杂的算法，例如机器学习和图处理。因此，随着多个计算阶段的增加，所产生的任务执行计划可能变得非常复杂。此外，考虑到每个计算阶段不同，每个阶段的最佳分区数量也可能不同，这使问题更加复杂。

发明内容

[0006] 本发明的目的是克服现有技术的缺陷，提供一种Spark集群系统的在线优化分区的任务调度方法及装置。

[0007] 本发明所提出的技术问题是这样解决的：

[0008] 一种Spark集群系统的在线优化分区的任务调度方法，包括以下步骤：

[0009] 步骤1.统计上游Stage(阶段)输出数据量大小、参与作业执行的总CPU核数、内存总量和用于拉取数据的内存比例；

[0010] 步骤2.根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置数据的分区数量；

[0011] 步骤3.监测计算节点的平均CPU利用率和内存利用率，对各计算节点的资源利用水平进行评估；

[0012] 步骤4.降序排列所有节点的资源利用水平，优先调度任务给资源利用水平最高的节点执行；

[0013] 步骤5.重复步骤3-步骤4，直至所有任务调度完成。

[0014] 步骤1中，上游Stage输出数据量大小$S=\sum_{i=1}^{n} s_i$，其中，s_i为第i个计算节点上任务输出数据量大小，$1 \leq i \leq n$，n为计算节点的数量；参与作业执行的总CPU核数$Core_{app}=\sum_{i=1}^{n} CPU_{app(i)}$，其中，$CPU_{app(i)}$为第i个计算节点上用于应用程序的CPU核数；内存总量$Memory_{app}=\sum_{i=1}^{n} Mem_{app(i)}$，其中，$Mem_{app(i)}$为第i个计算节点上用于应用程序的内存容量；用于拉取数据的内存比例α从Spark参数配置文件中读取。

[0015] 步骤2中，用于拉取数据的内存容量的大小为$Memory_{app} \times \alpha$，将输出数据量S与用于拉取数据的内存容量$Memory_{app} \times \alpha$进行大小比对：

[0016] 当$S \leq Memory_{app} \times \alpha$时，任务执行的轮数r＝1，数据的分区数为$Core_{app}$；

[0017] 当$S > Memory_{app} \times \alpha$时，任务执行的轮数$r = ceil(S/(Memory_{app} \times \alpha))$，其中ceil表示向上取整，数据的分区数为$Core_{app} \times r$。

[0018] 步骤3中，为了减少计算节点CPU利用率或内存利用率的抖动误差，引入控制系统理论中的一种负反馈调节机制，CPU利用率或内存利用率的负反馈调节机制为：

[0019] $$U_i(t_j)=\begin{cases} U_i(t_{j-1})+\sqrt{U_i'(t_j)-U_i(t_{j-1})}, & j \geq 1 \\ U_i'(t_j), & j = 0 \end{cases}$$

[0020] 其中，t_j为第j个时刻，j为≥0的整数，$U_i(t_j)$为当前时刻第i个计算节点的CPU利用率或内存利用率，$U_i(t_{j-1})$为上一时刻的第i个计算节点的CPU利用率或内存利用率，$U_i'(t_j)$为当前时刻第i个计算节点的CPU或内存利用率的监测值。

[0021] 计算节点的资源利用水平是随着其资源使用状况动态变化的，需要在任务开始分配时对每个计算节点的资源利用水平进行度量，以此决定调度任务到哪个节点上执行。计算节点的资源利用水平由计算节点剩余可用CPU核数、CPU的主频大小、当前CPU利用率、当前内存利用率以及历史CPU利用率的熵值共同决定，即计算节点的资源利用水平与节点硬件属性以及资源利用率高度相关，计算节点剩余可用CPU核数越大、CPU的主频越大、当前CPU利用率越低、当前内存利用率越低、历史CPU利用率的熵值越小，该计算节点的资源利用

水平越高，分配任务时具有优先分配权。

[0022] 第i个计算节点的资源利用水平RL_i的计算公式为：

[0023] $RL_i = AvailableCores_i \times Speed_{cpu} \times (1-R_{i,cpu}) \times (1-R_{i,mem}) \times (1-E_i)$

[0024] 其中，$AvailableCores_i$为第i个计算节点的可用CPU核数，$Speed_{cpu}$为计算节点CPU的主频大小，$R_{i,cpu}$为第i个计算节点的当前CPU利用率大小，$R_{i,mem}$为第i个计算节点的当前内存利用率大小，E_i为第i个计算节点的历史CPU利用率的熵值，反映CPU利用率的波动。

[0025] 第i个计算节点的历史CPU利用率的熵值$E_i = -(c1 \times \log_2 c1 + c2 \times \log_2 c2)$，其中，c1为历史CPU利用率中CPU利用率值大于等于CPU平均利用率的次数，c2为历史CPU利用率中CPU利用率值小于CPU平均利用率的次数。

[0026] 步骤4中，任务分配时主要采取贪心策略，首先按照资源利用水平降序排列所有计算节点，然后遍历所有任务，将任务分配到资源利用水平最高的计算节点上，如果该计算节点的可用CPU核数大于每个任务需要的CPU核数(默认为1核)，则在当前计算节点上分配该任务，同时更新该计算节点的可用CPU核数，任务后续将在该计算节点上上以最大的数据本地性运行；如果任务需要分配多轮，则重复步骤3-4直到所有任务分配完成。

[0027] 本发明还提供了一种Spark集群系统的在线优化分区的任务调度装置，包括：

[0028] (1)信息收集模块：统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量和用于拉取数据的内存比例；

[0029] (2)分区优化模块：根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置数据的分区数量；

[0030] (3)节点监测模块：监测计算节点的平均CPU利用率和内存利用率，对各计算节点的资源利用水平进行评估；

[0031] (4)节点排序模块：按照资源利用水平降序或升序排列所有节点；

[0032] (5)任务分配模块：任务分配时主要采取贪心策略，优先调度任务给资源利用水平最高的节点；

[0033] (6)任务执行模块：用于按照任务分配模块决定的顺序执行任务。

[0034] 本发明的有益效果是：

[0035] 本发明能够自动配置优化的分区数量，充分利用集群的资源，加快任务执行速度。

附图说明

[0036] 图1为Spark集群系统的架构图；

[0037] 图2为本发明所述Spark集群系统的在线优化分区的任务调度方法流程图；

[0038] 图3为本发明所述Spark集群系统的在线优化分区的任务调度装置示意图；

[0039] 图4为实施例所述方法的流程图。

具体实施方式

[0040] 下面结合附图和实施例对本发明进行进一步的说明。

[0041] 本实施例提供一种Spark集群系统的在线优化分区的任务调度方法，其流程图如图2所示，包括以下步骤：

[0042] 步骤1.统计上游Stage(阶段)输出数据量大小、参与作业执行的总CPU核数、内存

总量和用于拉取数据的内存比例；

[0043] 步骤2.根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置数据的分区数量；

[0044] 步骤3.监测计算节点的平均CPU利用率和内存利用率，对各计算节点的资源利用水平进行评估；

[0045] 步骤4.降序排列所有节点的资源利用水平，优先调度任务给资源利用水平最高的节点执行；

[0046] 步骤5.重复步骤3-步骤4，直至所有任务调度完成。

[0047] 步骤1中，上游Stage输出数据量大小$S=\sum_{i=1}^{n} s_i$，其中，s_i为第i个计算节点上任务输出数据量大小，1≤i≤n，n为计算节点的数量；参与作业执行的总CPU核数$Core_{app}=\sum_{i=1}^{n} CPU_{app(i)}$，其中，$CPU_{app(i)}$为第i个计算节点上用于应用程序的CPU核数；内存总量$Memory_{app}=\sum_{i=1}^{n} Mem_{app(i)}$，其中，$Mem_{app(i)}$为第i个计算节点上用于应用程序的内存容量；用于拉取数据的内存比例α从Spark参数配置文件中读取。

[0048] 步骤2中，用于拉取数据的内存容量的大小为$Memory_{app}\times\alpha$，将输出数据量S与用于拉取数据的内存容量$Memory_{app}\times\alpha$进行大小比对：

[0049] 当$S\leq Memory_{app}\times\alpha$时，任务执行的轮数r＝1，数据的分区数为$Core_{app}$；

[0050] 当$S>Memory_{app}\times\alpha$时，任务执行的轮数r＝ceil(S/($Memory_{app}\times\alpha$))，其中ceil表示向上取整，数据的分区数为$Core_{app}\times r$。

[0051] 步骤3中，为了减少计算节点CPU利用率或内存利用率的抖动误差，引入控制系统理论中的一种负反馈调节机制，CPU利用率或内存利用率的负反馈调节机制为：

[0052] $$U_i(t_j)=\begin{cases} U_i(t_{j-1})+\sqrt{U_i'(t_j)-U_i(t_{j-1})}, & j\geq 1 \\ U_i'(t_j), & j=0 \end{cases}$$

[0053] 其中，t_j为第j个时刻，j为≥0的整数，$U_i(t_j)$为当前时刻第i个计算节点的CPU利用率或内存利用率，$U_i(t_{i-1})$为上一时刻的第i个计算节点的CPU利用率或内存利用率，$U_i'(t_j)$为当前时刻第i个计算节点的CPU或内存利用率的监测值。

[0054] 计算节点的资源利用水平是随着其资源使用状况动态变化的，需要在任务开始分配时对每个计算节点的资源利用水平进行度量，以此决定调度任务到哪个节点上执行。计算节点的资源利用水平由计算节点剩余可用CPU核数、CPU的主频大小、当前CPU利用率、当前内存利用率以及历史CPU利用率的熵值共同决定，即计算节点的资源利用水平与节点硬件属性以及资源利用率高度相关，计算节点剩余可用CPU核数越大、CPU的主频越大、当前CPU利用率越低、当前内存利用率越低、历史CPU利用率的熵值越小，该计算节点的资源利用水平越高，分配任务时具有优先分配权。

[0055] 第i个计算节点的资源利用水平RL_i的计算公式为：

[0056] $RL_i=AvailableCores_i\times Speed_{cpu}\times(1-R_{i,cpu})\times(1-R_{i,mem})\times(1-E_i)$

[0057] 其中，$AvailableCores_i$为第i个计算节点的可用CPU核数，$Speed_{cpu}$为计算节点CPU的主频大小，$R_{i,cpu}$为第i个计算节点的当前CPU利用率大小，$R_{i,mem}$为第i个计算节点的当前内存利用率大小，E_i为第i个计算节点的历史CPU利用率的熵值，反映CPU利用率的波动。

[0058] 第i个计算节点的历史CPU利用率的熵值$E_i=-(c1\times\log_2 c1+c2\times\log_2 c2)$，其中，c1为历史CPU利用率中CPU利用率值大于等于CPU平均利用率的次数，c2为历史CPU利用率中CPU利用率值小于CPU平均利用率的次数。

[0059] 步骤4中，任务分配时主要采取贪心策略，首先按照资源利用水平降序排列所有计算节点，然后遍历所有任务，将任务分配到资源利用水平最高的计算节点上，如果该计算节点的可用CPU核数大于每个任务需要的CPU核数(默认为1核)，则在当前计算节点上分配该任务，同时更新该计算节点的可用CPU核数，任务后续将在该计算节点上上以最大的数据本地性运行；如果任务需要分配多轮，则重复步骤3-4直到所有任务分配完成，如图4所示。

[0060] 本实施例还提供了一种Spark集群系统的在线优化分区的任务调度装置，其示意图如图3所示，包括：

[0061] (1)信息收集模块：统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量和用于拉取数据的内存比例；

[0062] (2)分区优化模块：根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置数据的分区数量；

[0063] (3)节点监测模块：监测计算节点的平均CPU利用率和内存利用率，对各计算节点的资源利用水平进行评估；

[0064] (4)节点排序模块：按照资源利用水平降序或升序排列所有节点；

[0065] (5)任务分配模块：任务分配时主要采取贪心策略，优先调度任务给资源利用水平最高的节点；

[0066] (6)任务执行模块：用于按照任务分配模块决定的顺序执行任务。

[0067] 本领域普通技术人员可以理解实现上述实施例方法中的全部或部分流程，是可以通过计算机程序来指令相关的硬件来完成，所述的程序可存储于一台计算机可读取存储介质中，该程序在执行时，可包括如上述各方法的实施例的流程。其中，所述的存储介质可为磁碟、光盘、只读存储记忆体(Read-Only Memory，ROM)或随机存储记忆体(Random Access Memory，RAM)等。

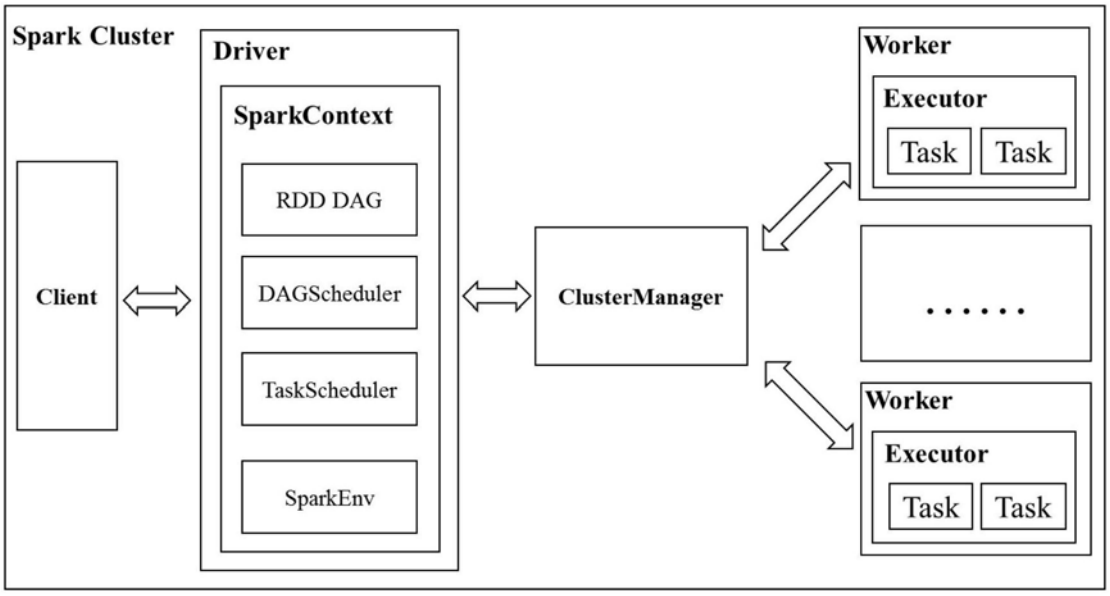
Spark Cluster
Driver
SparkContext
RDD DAG
DAGScheduler
TaskScheduler
SparkEnv
Client
ClusterManager
Worker
Executor
Task
Task
.....
Worker
Executor
Task
Task

图1

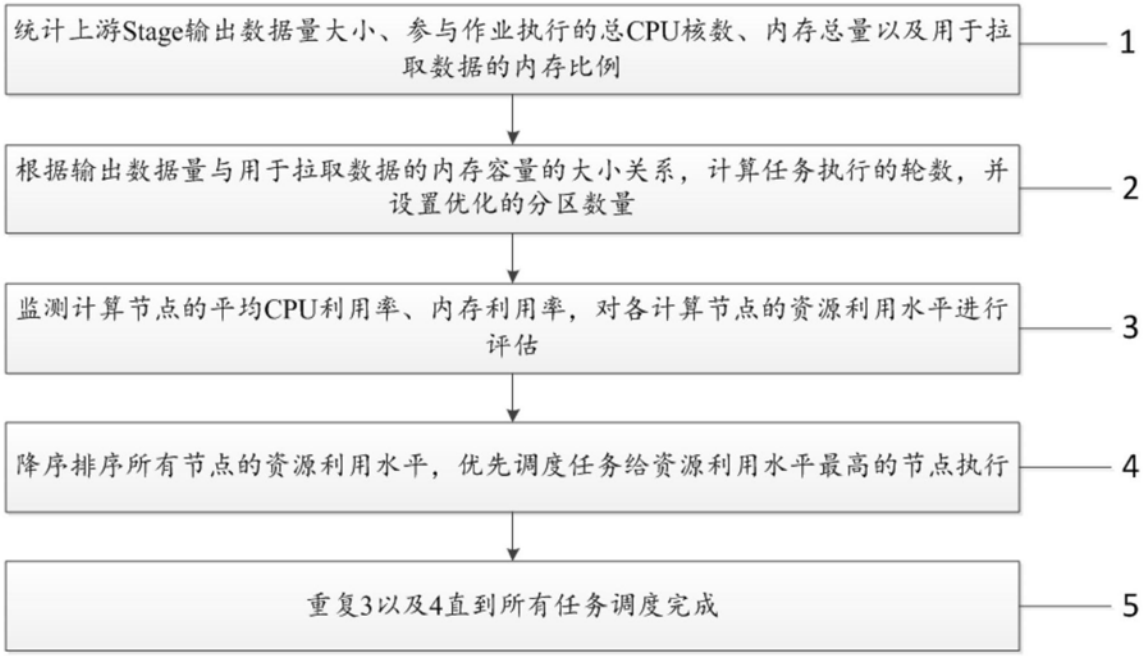
统计上游Stage输出数据量大小、参与作业执行的总CPU核数、内存总量以及用于拉取数据的内存比例
1
根据输出数据量与用于拉取数据的内存容量的大小关系，计算任务执行的轮数，并设置优化的分区数量
2
监测计算节点的平均CPU利用率、内存利用率，对各计算节点的资源利用水平进行评估
3
降序排序所有节点的资源利用水平，优先调度任务给资源利用水平最高的节点执行
4
重复3以及4直到所有任务调度完成
5

图2


信息收集模块
分区优化模块
节点监测模块
节点排序模块
任务分配模块
任务执行模块

图3

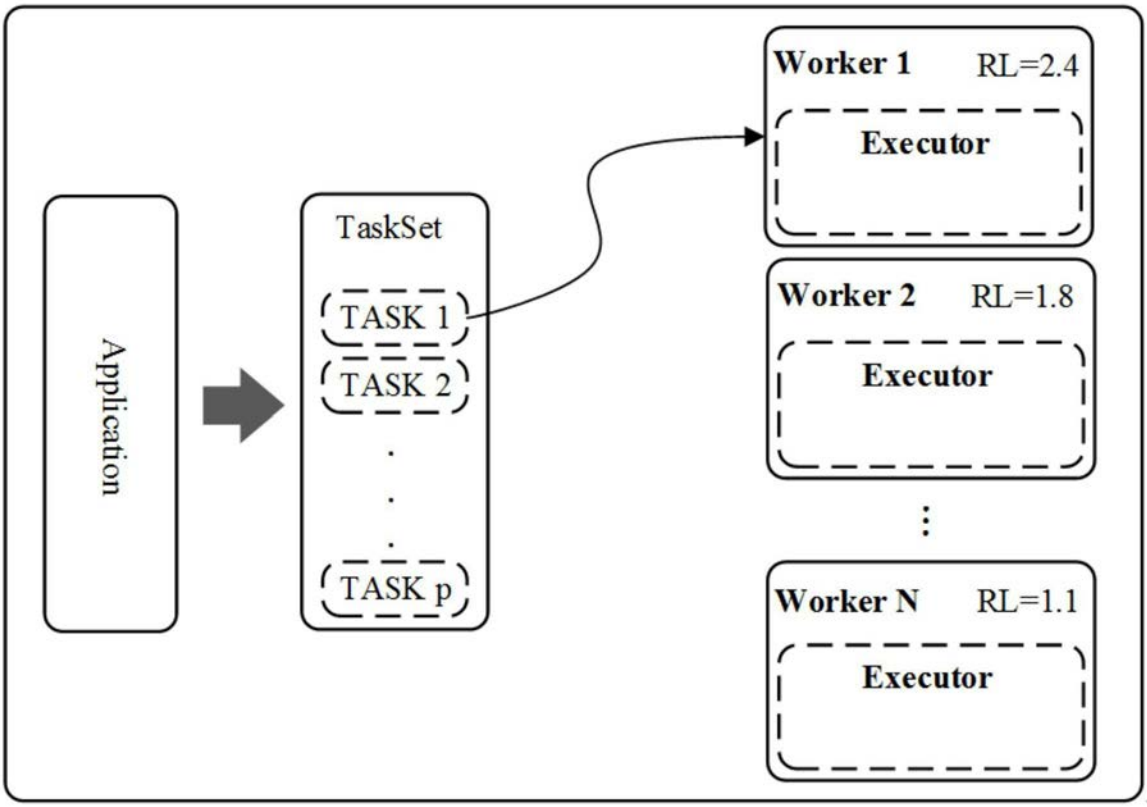
Application
TaskSet
TASK 1
TASK 2
TASK p
Worker 1 RL=2.4
Executor
Worker 2 RL=1.8
Executor
Worker N RL=1.1
Executor

图4