MapReduce 框架下的 Skyline 云资源选择算法

齐玉东 何 诚 司维超
(海军航空工程学院 山东 烟台 264001)

摘 要 文中研究了 MapReduce 框架下的云资源选择算法。该算法利用概率筛选的方法计算资源节点属于 Skyline 结果集的概率,并通过阈值过滤来降低 MapReduce 中的心跳频率,以达到优化网络流量的目的。

关键词 MapReduce,Skyline,资源选择,网络优化
中图法分类号 TP391 **文献标识码** A

Cloud Resource Selection Algorithm by Skyline under MapReduce Frame

QI Yu-dong HE Cheng SI Wei-chao
(Naval Aeronautical Engineering Institute, Yantai, Shandong 264001, China)

Abstract This paper researched a cloud resource selected algorithm under the MapReduce frame, which uses a method of possibility filtrate to figure the possibility of a resource nod belonging to the Skyline results set. It filtrates information by value seted in advance, to reduce the frequency of heartbeat in the MapReduce frame eventually, to optimize the flow of network.

Keywords MapReduce, Skyline, Resource selection, Network majorization

1 引言

如何从海量的数据中筛选出“物美价廉”的、最可能满足需求的信息,过滤掉多余的、相较而言“又贵又差”的信息,成为了大数据背景下数据库领域面对的难题。在此背景下,Skyline 查询[1]应时而生。Skyline 查询能够在海量数据中针对提出的需求,利用支配与否的关系筛选出“物美价廉”的信息,为多目标决策提供直接的辅助参考。因此,Skyline 查询在目标决策[2]领域有着广泛的应用[3-4]。而云计算中虚拟资源量大、属性多的特点与 Skyline 查询十分契合,因此云计算中对 Skyline 查询的研究应用并不少见。姜罕盛等[5]研究了云环境下的数据安置策略与 Skyline 查询,王媛等[6]对了云计算环境下的容错并行 Skyline 查询算法进行了研究。近年来,主流的云计算 Hadoop 平台中采用了 MapReduce 计算框架,将数据处理过程拆分为 Map(映射)和 Reduce(简化)两步。张波良等[7]研究了 MapReduce 框架下 Skyline 的计算,史恒亮等[8]在云计算任务调度研究中研究了在低虚拟化程度时,Hadoop 架构下如何通过改进 Skyline 查询方法把多目标任务映射到具有多属性的资源节点上,以减少网络开销。

MapReduce 作业的计算工作都由 TaskTracker 完成,JobTracker 是作业的分解者,也是任务的调度中心,但不参与任务的计算。JobTracker 与 TaskTracker 之间采用了 push 通信模型[9],即 JobTracker 不会主动与 TaskTracker 通信,而是被动等待 TaskTracker 通过心跳告知 JobTracker 是否存活以及节点资源的使用情况等信息。为了保证节点资源信息的实时性,心跳的频率往往很高,而高频率的心跳所形成的大量数据会造成系统负担,性能参数数据量/总的数据传输量的比重会大幅增加。如果 JobTracker 主动将需求信息发布给 TaskTracker,然后 TaskTracker 根据自身资源的使用情况,在满足一定需求条件的情况下返回自身资源信息给 JobTracker,以筛选出最适合完成此任务的节点资源,否则默认不适合完成此任务,不返回任何信息。这样虽然会对 JobTracker 实时获取 TaskTracker 的健康信息产生影响,导致有任务失败或网络故障等不乐观情况发生时容错处理的反应变慢等问题[10],但能够有效降低心跳频率,节省网络开销。

本文基于上述思路,针对资源池化后的高虚拟化程度,以网络流量优化为研究目标,建立池化资源的属性模型,改进云环境下的 Skyline 资源选择算法,并通过网络仿真来验证算法的可靠性。

2 资源池化属性模型

定义 1 资源节点有 i 个动态属性 $A_1, A_2, \cdots, A_i$,在某个固定时刻,动态属性都有一个静态值 $a_1, a_2, \cdots, a_i$ 来表示在该时刻资源节点在各个动态属性上拥有的资源或属性,这些静态属性构成一个 i 维向量 $r=\langle a_1, a_2, \cdots, a_i\rangle$,$a_1 \leqslant a_{1\max}$,$a_2 \leqslant a_{2\max}, \cdots, a_i \leqslant a_{i\max}$,其中 $a_{1\max}, a_{2\max}, \cdots, a_{i\max}$ 分别为资源节点在动态属性 $A_1, A_2, \cdots, A_i$ 上拥有的总资源或总属性,r 能反映某时刻该资源节点的瞬时资源和属性。

定义 2 JobTracker 拆分任务时,每一个拆分任务都得到一个需求信息 Q,需求信息 Q 也是一个 i 维向量 $\langle a_1, a_2, \cdots, a_i\rangle$。

定义 3 对于任意资源节点 j 的资源和属性的向量 $r_j=\langle a_{1,j}, a_{2,j}, \cdots, a_{l,j}, \cdots, a_{i,j}\rangle$,该资源节点相对于需求 $Q=\langle a_1, a_2, \cdots, a_l, \cdots, a_i\rangle$ 的距离向量为 $d_j=\langle d_{1,j}, d_{2,j}, \cdots, d_{l,j}, \cdots, d_{i,j}\rangle$,其中 $d_{l,j}$ 为资源节点 j 在动态属性 A_l 上与需求 Q 的距离值,$d_{l,j}=a_{l,j}-a_l$,$1 \leqslant l \leqslant i$。

齐玉东(1973-),男,博士,副教授,主要研究领域为指挥信息系统;何 诚(1993-),男,硕士生,主要研究领域为网络作战辅助决策;司维超(1984-),男,博士,讲师,主要研究领域为装备信息化,E-mail:674690154@qq.com。

在此资源池化模型中，每一个资源节点的 i 个动态属性描述了节点资源的状态与能力，需求信息的维度与节点动态属性维度相匹配。在此模型下，各虚拟资源属性有了统一的描述方式，任务调度过程中通过比较各资源的静态属性与需求信息的关系，能够合理地完成资源的分配。

定义 4 资源节点 j_1 与资源节点 j_2 关于需求 Q 的距离向量分别为 d_{j1} 和 d_{j2}，如果对于 $\forall l \in N^+$，$1 \leqslant l \leqslant i$，都有 $d_{l,j_1} \leqslant d_{l,j_2}$，则称节点 j_1 支配节点 j_2，记为 $j_1 \prec j_2$。

定义 5 对于任意的需求 Q，经由 Skyline 算法计算返回的节点集 B 满足 $B=\{j_i \mid j_i \in N^+, \forall j_k \in N^+, j_k \neq j_i, j_k$ 不支配 $j_i\}$，称集合 B 为 Skyline 节点集，记为 SK。

3 资源选择算法

3.1 主动式的资源需求信息广播

在 Hadoop 架构下，任务调度问题的关键在于 JobTracker 如何能够“物尽其用”地使用云计算资源，这也是资源选择算法的目的所在。如果资源的分配不合理，JobTracker 把资源需求高、计算复杂的任务分配给现有剩余计算资源不足的 TaskTracker，则此 TaskTracker 执行 Map 或 Reduce 任务的效率会降低，单独的任务效率低会影响到整个 Job 的处理过程，这就是所谓的“短板”效应[11]；如果 JobTracker 把资源需求低、计算较为简单的任务分配给了现有剩余资源远远满足需求的 TaskTracker，这虽然不会对此任务的效率产生影响，但可能会影响一个原本最适合此 TaskTracker 的大型任务的分配，从而造成资源的浪费，成为一种“因小失大”的调度。

因此，在 MapReduce 计算框架中利用 Skyline 算法进行任务调度的资源选择算法，需要 JobTracker 对所有 TaskTracker 的资源信息和属性进行 Skyline 计算，从而得到满足需求的 SK。此算法需要高频的心跳才能保证资源的时效性，这会造成大量的网络开销。如果能够降低心跳频率，那么改进后的资源选择算法便能优化网络流量。

本文认为，JobTracker 可以主动地将需求信息 Q 广播给每一个空闲的 TaskTracker，TaskTracker_j 根据自身的属性 r_j 得到距离向量 d_j。显然，如果存在一个 $d_{l,j}<0$，则该资源节点在动态属性 A_l 上不满足完成任务的要求，资源节点保持沉默，仅仅接收需求信息，不返回自身的资源信息。

如果 TaskTracker_j 的距离向量 d_j 的每一个距离值都大于或等于 0，则该资源节点的现有剩余资源能够满足完成此任务的需求。在此情况下，该资源节点是有被分配此任务的潜力的。而能够满足需求信息的资源节点往往不止一个，这些满足需求的资源节点在动态属性 A_i 上的差异，造成了各节点执行任务的合适程度不同。所有满足需求的资源节点中存在 SK，每一个属于 SK 的资源节点都各有优势，而非 Skyline 节点的资源节点被支配，属于“又贵又差”的资源。

用块嵌套循环算法 BNL、比特位图算法 Bitmap[12] 等 Skyline 查询方法计算 Skyline 节点时需要所有 TaskTracker 的现有资源和属性参与计算，这些现有资源和属性通过心跳的方式由 TaskTracker 向 JobTracker 传递，造成了大量的网络花销。TaskTracker 如果在接收到需求 Q 时，就计算自身成为 Skyline 节点的概率，那么可以设定概率阈值 P，当概率小于 P 时，TaskTracker 认为自身参与计算的性价比不高，不向 JobTracker 返回信息，这将会节省部分网络花销，且所有返回信息的节点属于部分结果集，计算 Skyline 查询的完整结果集与只计算部分结果集的代价是不同的[13]。

3.2 算法描述

由上节的资源池化属性模型可知，需求 Q 与节点 j 的属性 r_j 为 i 维向量，即 $Q=\langle a_1, a_2, \cdots, a_l, \cdots, a_i\rangle$，$r_j=\langle a_{1,j}, a_{2,j}, \cdots, a_{l,j}, \cdots, a_{i,j}\rangle$，则可以得到距离向量 $d_j=\langle d_{1,j}, d_{2,j}, \cdots, d_{l,j}, \cdots, d_{i,j}\rangle$。

定义 6 如果存在一个距离值 $d_{l,j}<0$，则该资源节点在动态属性 A_i 上不满足完成任务的要求，该资源节点保持沉默，仅仅接收需求信息，不返回自身的资源信息；如果节点 j 的距离向量 d_j 的每一个距离值都大于或等于 0，则该资源节点的现有剩余资源能够满足完成此任务的需求。

资源节点在动态属性 $A_1, A_2, \cdots, A_i$ 上拥有的静态属性的总量是固定的，在云计算正常工作下，每个节点都执行不同的任务，因此剩余资源的静态属性是随时间变化的，通过统计这种变化可以得到节点的现有剩余资源的概率密度函数。其中，节点 j 的概率密度函数为 $f_j(x_1, x_2, \cdots, x_i)$。

每个资源节点拥有的总资源为 i 维向量 $r_{j\max}=\langle a_{1,j\max}, a_{2,j\max}, \cdots, a_{i,j\max}\rangle$。

在节点接收需求 Q 的瞬间，节点的静态资源固定为 i 维向量 $r_j=\langle a_{1,j}, a_{2,j}, \cdots, a_{i,j}\rangle$。

显然，所有各类资源节点在执行任务和资源使用情况上是独立的，在资源节点 j_k 接收需求 Q 时，所有其他节点 j_l ($l \neq k$)的资源属性 j_k 并不明确，可能 j_l 支配 j_k，也可能 j_l 不支配 j_k。

j_k 不满足 Q 的情况不考虑，因此在 j_k 满足 Q 且 j_l 不满足 Q 的情况下，才有 $j_l \prec j_k$ 的可能。下面首先在二维空间上对资源选择算法进行研究，然后将其推广到高维空间中。

记二维空间中资源节点 j 的概率密度函数为 $f_j(x_1, x_2)$，节点的静态属性为 $r_j=\langle a_{1,j}, a_{2,j}\rangle$，我们需要计算资源节点 j 成为 Skyline 节点的概率 $P(j \in SK)$。直接利用已知的条件进行 $P(j \in SK)$的研究较为困难，从侧面考虑可知 $j \in SK$ 与 $j \notin SK$ 是对立事件，则可以得到：

$$P(j \in SK)=1-P(j \notin SK)$$

由此，问题便转化成了计算节点 j 不是 Skyline 节点，即节点 j 被支配的概率。

假设 $Q=\langle 9,5\rangle$，有节点 1,2,3,4，下面以研究资源节点 1 成为 Skyline 节点的概率为例，来论述如何计算资源节点 j 成为 Skyline 节点的概率。

设 1,2,3,4 资源节点的总节点属性分别为 $r_{1\max}=\langle 12, 15\rangle$，$r_{2\max}=\langle 14, 11\rangle$，$r_{3\max}=\langle 21, 7\rangle$，$r_{4\max}=\langle 18, 14\rangle$，如图 1 所示，则节点 1,2,3,4 所对应的距离向量 $d_{1\max}=\langle 3, 10\rangle$，$d_{2\max}=\langle 5, 6\rangle$，$d_{3\max}=\langle 12, 2\rangle$，$d_{4\max}=\langle 9, 9\rangle$，如图 2 所示。

图 1 中所示阴影部分面积 S_1，即为资源节点 1 可能存在的资源剩余情况。在节点 2,3,4,5 看来，节点 1 以概率密度函数 $f_1(x_1, x_2)$ 随机处于 S_1 内。类比可知其他节点的资源剩余信息和属性。

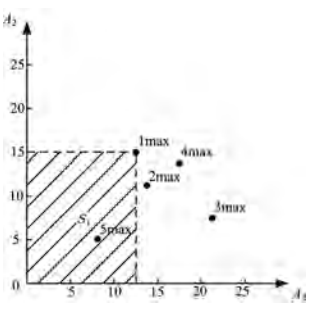

图 1 2 维节点属性图

由图 1 可知，节点 5 不满足 Q，因此不参与计算。当节点 1 接收 Q 时，节点属性为 $r_1=\langle 11,13\rangle$，则节点 1，2，3，4 在需求 Q 下的距离向量 $d_{1\max}$，$d_{2\max}$，$d_{3\max}$，$d_{4\max}$ 如图 2 所示。

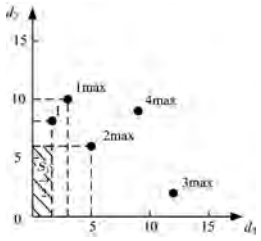

图 2　满足 Q 的 2 维节点属性图

资源节点 1 的特征向量值点与原点构成的矩形内的所有点组成了支配资源节点 1 的所有属性状态的集合 U_1；资源节点 2 的总资源特征向量值(2max)与原点构成的矩形内的所有点组成了资源节点 2 满足 Q 的所有属性状态的集合 U_2；资源节点 2 满足 Q，$2\prec 1$ 的属性状态集合为 U_3，且有：

$$U_3=U_1\cap U_2$$

在图 2 中，U_3 为矩形 $S_2\prec 1$ 中所有属性点的集合，因此只要资源节点机 2 的属性 $r_2\in U_3$，则 $2\prec 1$。$S_2\prec 1$ 的边界条件由定义可知，如下：

$$\begin{cases} d_{2,s}\leqslant d_{2,1} \\ d_{2,s}\leqslant d_{2,2} \\ d_{1,s}\leqslant d_{1,1} \\ d_{1,s}\leqslant d_{1,2} \\ d_{2,s}\geqslant 0 \\ d_{1,s}\geqslant 0 \end{cases} \tag{1}$$

可化简为：

$$\begin{cases} d_{2,s}\leqslant \min(d_{2,1},d_{2,2}) \\ d_{1,s}\leqslant \min(d_{1,1},d_{1,2}) \\ d_2\geqslant 0 \\ d_1\geqslant 0 \end{cases} \tag{2}$$

将 $2\prec 1$ 的概率用 $P(2\prec 1)$ 表示，$r_2\in U_3$ 的概率用 $P(r_2\in U_3)$ 表示，则可以得到：

$$P(2\prec 1)=P(r_2\in U_3)=\int_{S_{2\prec 1}} f_2(x_1,x_2)\mathrm{d}s$$

同理可以得到：

$$P(3\prec 1)=\int_{S_{3\prec 1}} f_3(x_1,x_2)\mathrm{d}s$$

$$P(4\prec 1)=\int_{S_{4\prec 1}} f_4(x_1,x_2)\mathrm{d}s$$

上文已经论述过各资源节点的任务执行和资源使用情况是独立的，因而支配关系属于独立事件，即 $2\prec 1$，$3\prec 1$ 与 $4\prec 1$ 互相之间没有关联。

因此，可以得到：

$$P(2\prec 1\cap 3\prec 1)=P(2\prec 1)\times P(3\prec 1)$$

$$P(2\prec 1\cap 4\prec 1)=P(2\prec 1)\times P(4\prec 1)$$

$$P(3\prec 1\cap 4\prec 1)=P(3\prec 1)\times P(4\prec 1)$$

$$P(2\prec 1\cap 3\prec 1\cap 4\prec 1)=P(2\prec 1)\times P(3\prec 1)\times P(4\prec 1)$$

显然，$1\notin SK$ 时，资源节点 1 被其他节点所支配，因而：

$$P(1\notin SK)=P(2\prec 1\cup 3\prec 1\cup 4\prec 1)$$

由事件的概率加法公式可以得到：

$$P(1\notin SK)=P(2\prec 1)+P(3\prec 1)+P(4\prec 1)-P(2\prec 1\cap 3\prec 1)-P(2\prec 1\cap 4\prec 1)-P(3\prec 1\cap 4\prec 1)+P(2\prec 1\cap 3\prec 1\cap 4\prec 1)$$

因此，资源节点 1 成为 Skyline 点的概率为：

$$P(1\in SK)=1-P(1\notin SK)$$

将维度扩充到 i 维情况，设总的资源节点为 n，资源节点机用 j_k 表示，k 为整数且 $0<k\leqslant n$，则可以得到 $S_{j,k_1\to j,k_2}$ 的边界条件为：

$$\begin{cases} d_{1,s}\leqslant \min(d_{1,j_{k1}},d_{1,j_{k2}}) \\ d_{2,s}\leqslant \min(d_{2,j_{k1}},d_{2,j_{k2}}) \\ \quad\vdots \\ d_{i,s}\leqslant \min(d_{i,j_{k1}},d_{i,j_{k2}}) \\ d_{1,s}\geqslant 0 \\ d_{2,s}\geqslant 0 \\ \quad\vdots \\ d_{i,s}\geqslant 0 \end{cases} \tag{3}$$

资源节点 $j_{k_1}\prec j_{k_2}$ 的概率为：

$$P(j_{k1}\prec j_{k2})=\int_{s_{jk1}\prec j_{k2}} f_{j_{k1}}(x_1,x_2)\mathrm{d}s \tag{4}$$

资源节点 j_{k_i} 成为 Skyline 点的概率为：

$$\begin{aligned} P(j_{k_i}\in SK)&=1-P(j_{k_i}\notin SK) \\ &=1-\Big(\sum_{k=1,k\neq k_i}^{n} P(j_k\prec j_{k_i})-\sum_{1\leqslant j<k\leqslant n} P((j_{k_j}\prec j_{k_i})\cap(j_{k_k}\prec j_{k_i}))+\cdots+(-1)^{n-1}P((j_1\prec j_i)\cap(j_2\prec j_i)\cap\cdots\cap(j_n\prec j_i))\Big) \end{aligned} \tag{5}$$

当 $P(j_{k_i}\in SK)<P$ 时，资源节点 j_{k_i} 不返回节点属性信息，JobTracker 利用经过阈值筛选后的返回信息再经由已有的 Skyline 算法计算 SK。这种资源选择算法的改进，虽然输出为部分 SK，而且影响了任务调度时节点信息的时效性，但减少了任务调度时的输入节点集，同时降低了网络的开销，适用于网络环境较差的情况。

4　仿真实验及结果分析

为模拟需求信息和资源节点属性的数据包传递，本文利用 OPNET 网络仿真软件[14]搭建了网络模型，模拟算法在各 TaskTracker 上的运行，通过捕捉信道利用率来分析网络中流量的优化情况。

在仿真中，设置 Q 为 6 维需求，对应资源节点的动态属性为 A_1，A_2，A_3，A_4，A_5，A_6。动态属性与物理资源、属性的对应关系如表 1 所列。

表 1　属性的对应关系表

动态属性	物理资源、属性
A_1	内存大小(memo_spac)
A_2	硬盘读写速度(disk_spid)
A_3	硬盘存储容量(disk_capty)
A_4	CPU 频率(cpu_phse)
A_5	安全性(security)
A_6	处理器核心数(cpu_numb)

在 OPNET 中编写并建立 1 个 JobTracker 和 4 个 TaskTracker 节点模型，并用 C++编写模型的执行代码，最终建立如图 3 所示的网络拓扑。JobTracker 命名为 hub，执行的功能是以 8 s 为间隔产生任务的需求信息 $Q=\langle a_1,a_2,a_3,a_4,a_5,a_6\rangle$，其中 a_1 由正态分布 N(4，1)随机产生，模拟任务对内存的需求；a_2 由正态分布 N(24，5.5)随机产生，模拟任务对

硬盘读写速度的需求；a_3 由正态分布 N(11,2.5)随机产生，模拟任务对硬盘存储容量的需求；a_4 由正态分布 N(1.8,0.5)随机产生，模拟任务对 CPU 频率的需求；a_5 由正态分布 N(0.5,0.1)随机产生，模拟任务对安全性的需求；a_6 是由正态分布 N(6,1.5)随机产生的整数，模拟任务对处理器核心数量的需求。

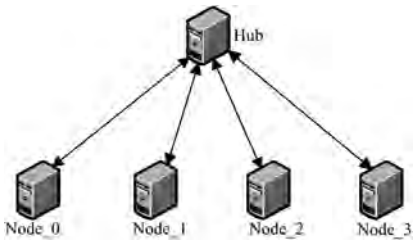

图 3 仿真拓扑图

将 TaskTracker 分别命名为 node_0，node_1，node_2，node_3。以 node_2 为例，其执行的功能是：TaskTracker 接收到需求信息 Q 后以正态分布 N(15.8,5.4)随机产生 A_1 的静态属性 a_{1_node2}，模拟当前情况下 TaskTracker 的内存资源剩余情况；以正态分布 N(45.6,8.8)随机产生 A_3 的静态属性 a_{3_node2}，模拟当前情况下 TaskTracker 的硬盘资源剩余情况；A_6 的静态属性 a_{6_node2} 是以正态分布 N(54,7.8)随机产生的整数，模拟当前情况下处理器的核心数资源；将静态属性 a_{2_node2}，a_{4_node2} 和 a_{5_node2} 固定后看作 TaskTracker 硬盘读写速度、CPU 频率和安全性的固态属性。

仿真捕捉 TaskTracker 与 JobTracker 之间的信道使用率，仿真结果以 node_2 为例，如图 4 所示。

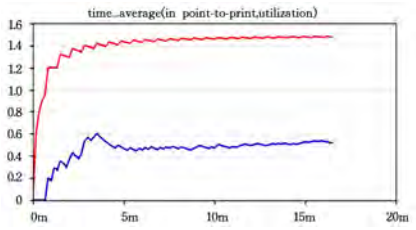

图 4 node_2 的仿真结果

仿真结果图 4 中上方线条数据代表 node_2 在接收需求 Q 时不经过阈值 P 筛选，直接返回资源状态和属性的信道使用率，这等同于心跳频率为 0.125 Hz 时的网络流量状况。图 4 中下方线条数据代表 node_2 以阈值概率 $P=0.5$ 筛选后得到的信道使用率。显然，经过改进后的模型的信道使用率有所降低，意味着改进后的 Skyline 算法模型能够优化 JobTracker 与 TaskTracker 之间传递信息时的网络流量。

继续延长仿真时间，得到了 1000 个需求 Q 和与之对应的各资源节点的状态，node_2 的统计结果如表 2 所列。

表 2 node_2 的统计结果表

	次数	成为 Skyline 的个数
不满足 Q	535	—
满足阈值 $P=0.5$	312	231
不满足阈值 $P=0.5$	153	22

满足阈值 $P=0.5$ 时，node_2 返回的资源信息和属性的次数为 $Z_1=312$，Z_1 与总实验次数 Z 的比值 $Z_1/Z=0.312$，利用阈值 $P=0.5$ 筛选后的信道使用率约为直接返回资源状态和属性的信道使用率的 1/3，与图 4 仿真结果近似，从而验证了仿真结果的准确性。

在满足阈值 $P=0.5$ 的资源信息和属性次数 Z_1 中，有 $Z_2=231$ 次成为了 Skyline 点，$Z_2/Z_1=0.74$，说明经过阈值 P 筛选后，该资源节点有 74%的可信度成为 Skyline 点，可信度较高。

在不满足阈值 $P=0.5$ 的资源信息和属性次数 $Z_3=153$ 中，有 $Z_4=22$ 次成为了 Skyline 点，$Z_4/Z_3=0.14$，说明经过阈值 P 筛选后，该资源节点有 14%的概率错误地舍弃了真结果，有 86%的概率正确地过滤了假结果，过滤的可信度较高。

综上所述，改进后的 Skyline 算法模型经过阈值 $P=0.5$ 过滤后，正确过滤非 Skyline 节点的概率较高，算法模型的可信度好。

结束语 本文提出的这种云计算环境下的资源选择算法减少了 MapReduce 框架的 JobTracker 与 TaskTracker 之间传递信息时的网络流量，能够有效降低心跳频率。虽然任务调度时节点信息的时效性受到影响，但因为输出为 Skyline 查询的部分结果集，减少了任务调度时的输入节点，同时降低了网络的开销，其在网络环境恶劣情况下更为有效。

参 考 文 献

[1] 朱琳，关佶红，周水庚. Skyline 计算研究综述[J]. 计算机工程与应用，2008，44(6)：160-165.

[2] BORZSONYI S，KOSSMANN D，STOCKER K. The Skyline operator[C]// IEEE International Conference on Data Engineering(ICDE). 2001：421-430.

[3] 魏小娟，杨婧，李翠平，等. Skyline 查询处理[J]. 软件学报，2008，19(6)：1386-1400.

[4] PAPADIAS D，TAO Y，FU G，SEEGER B. ProgressiveSkyline computation in database system[J]. ACM Transcations on Database System，2005，30(1)：41-82.

[5] 姜罕盛. 云环境下数据安置策略与 Skyline 查询研究[D]. 天津：天津理工大学，2014.

[6] 王媛，王意洁，邓瑞鹏，等. 云计算环境下的容错并行 Skyline 查询算法研究[J]. 计算机科学与探索，2011，5(9)：804-814.

[7] 张波良，周水庚，关佶红. MapReduce 框架下的 Skyline 计算[J]. 计算机科学与探索，2011，5(5)：385-397.

[8] 史恒亮. 云计算任务调度研究[D]. 南京：南京理工大学，2012.

[9] 范东来. Hadoop 海量数据处理技术详解与项目实战[M]. 北京：人民邮电出版社，2016.

[10] 关国栋. Hadoop 容错调度技术研究[D]. 成都：西南交通大学，2016.

[11] DEAN J. Sanjay Ghem awant. MapReduce：simplified data processing on large clusters[M]. ACM，2008：1-27.

[12] TAN K L，ENG P K，OOI B C. Efficient progressive Skyline computation[C]// Proc of VLDB. 2001：301-310.

[13] 陈敏. OPNET 网络仿真[M]. 北京：清华大学出版社，2004.

[14] DESHPANDE A，GUESTRIN C，HONG W，et al. Exploiting correlated attributes in acquisitional query processing[C]// 21st International Conference on Data Engineering. Washington，USA，2005：143-154.