第46卷第3期 中国科学技术大学学报 Vol. 46, No. 3
2016年3月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Mar. 2016

文章编号:0253-2778(2016)3-0215-07

实时多任务异构云计算平台负载均衡算法

徐爱萍[1],吴 笛[1,2],徐武平,陈军[1]

(1.武汉大学计算机学院,湖北武汉 430072;2.解放军军事经济学院基础部,湖北武汉 430035)

摘要:针对组成云计算平台各节点之间软件环境存在异构性及数据分布不均匀等原因而导致云计算平台在处理大量任务时往往出现节点负载不均衡的问题,提出了解决异构云计算平台负载均衡方法与相关算法.研究首先统计云计算平台提供的各类服务的平均资源消耗,结合任务分配给指定节点后运行时长和资源占用情况,预测评估某一时刻节点上任务剩余资源消耗需求总量既剩余负载总量;各节点按周期反馈实际任务负载情况,及时修正任务负载信息;最后综合考虑节点各项性能,预测各节点负载评估值,并将待分配任务分发给最适合的节点.实验结果表明,该算法具有可行性并在实时多任务异构云计算平台负载均衡方面具有一定优势.

关键词:异构云计算平台;多任务;负载均衡;负载预测

中图分类号:TP202 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2016.03.006

引用格式:XU Aiping, WU Di, XU Wuping, et al. Real-time multitask load balance algorithm for heterogeneous cloud computing platforms[J]. Journal of University of Science and Technology of China, 2016,46(3):215-221.

徐爱萍,吴 笛,徐武平,等. 实时多任务异构云计算平台负载均衡算法[J]. 中国科学技术大学学报,2016,46(3):215-221.

Real-time multitask load balance algorithm for heterogeneous cloud computing platforms

XU Aiping, WU Di, XU Wuping, CHEN Jun

(1. *Computer School, Wuhan University, Wuhan* 430072, *China*;
2. *Department of Basic Knowledge, Military Economy Academy, Wuhan* 430035, *China*)

Abstract: A load-balancing algorithm applying in heterogeneous cloud Computing Platform handling real-time multitasks was proposed. Average hardware resource consumption of jobs running on nodes was measured. Balancing server receive load status of each node in the cluster periodically. A load status vector that reflects the quantity of resources required to finish allocated jobs of each node can be estimated according to the latest load status report and other parameters. As a request is submitted to the cluster, Balancing Server calculates the load status estimation vector of each node, and then dispatches it to the node that possesses the minimal load status estimation value. Experiment results proved that this dynamic load balancing algorithm is reasonable and effective.

Key words: heterogeneous cloud computing platform; multitask; load balance; load estimate

收稿日期:2015-08-27;**修回日期**:2015-12-01

基金项目:湖北省重大科技创新计划(2013AAA020),国家科技重大专项(2013ZX07503-001-06)资助.

作者简介:徐爱萍(通讯作者),女,1962年生,博士/教授. 研究方向:云存储、分布式数据集成方面的研究. E-mail:xuaiping@whu.edu.cn

0 引言

云计算作为大数据核心技术，近年来受到的关注，也取得了许多重要成果. 云计算是网格计算、分布式计算、并行计算、网络存储、虚拟化、负载均衡等传统计算机技术和网络技术发展融合的产物[1-2]. 云计算采用分布式存储模式，数据不再存放在单台计算机或服务器上，而是存放在多台机器组成的云计算平台中. 平台中各节点的硬件设备集合起来协同工作，共同对外提供数据存储和业务访问功能，极大地提升了系统数据处理能力，具有可扩展性强、管理维护方便等特点. 由于各节点之间软件环境具有异构性、数据的分布不均匀等特点，云计算平台在处理大量任务时往往出现节点负载不均衡的情况，影响了服务质量，因此各节点间的负载均衡成为云计算中的一个重要问题. 充分利用节点各种资源，最小化应用的执行时间，属最优化的理论范畴[3].

现有的负载均衡算法主要分为静态和动态两种[4-5]. 常见的静态负载均衡算法，如轮转算法、加权轮转算法等，通常以固定的概率分配任务，不考虑网络和服务器的状态信息. 动态负载均衡算法以网络状况或服务器的实时负载状态信息来决定任务的分配，如最小连接法、加权最小连接法、基于位置的最小连接法、带复制的基于位置的最小连接法等. 最小连接法每次分配请求时计算各服务器的当前连接数，将请求分配到链接数最小的服务器. 此方法在长时间运行的情况下，由于节点负载无法得到及时修正，会发生倾斜，均衡效果往往不能令人满意. 众多学者对负载均衡算法进行研究，提出了很多改进的动态负载均衡算法，许多负载均衡研究都是基于特定应用环境或针对特定目标的[6-7]，P2P 网络中的负载均衡方法[8]、天气预报等特定计算任务下的负载均衡[9]等. 通用动态负载均衡算法方面，Yun 等提出了按比例公平调度的负载均衡算法[10]，该算法考虑服务节点的性能与服务节点实际负载两方面的因素，用来指导资源分配比例，并通过动态反馈机制及时修正各节点的负载，保证系统平稳运行，但频繁收集负载信息，一方面带来了较大的额外开销，另一方面在响应时间上会受到一定影响，且该方法对异构云计算平台的支持效果并不理想. Joe-Wang 等的研究利用性能监控与任务调度中间件，解决异构网格计算中的负载均衡问题，取得了良好效果[11]. 张玉芳等提出基于负载权值的负载均衡算法[12]，该算法综合考虑节点负载和节点性能，提出基于负载权值选择分配负载的节点集合，引入负载差值计算节点分配负载的概率. 该算法的缺点是没有考虑计算任务资源需求的异构性，节点容易因为单一资源的耗尽而难以发挥最佳性能. 目前应用较广泛的 Hadoop 平台提供了先入先出、基于计算能力的调度和公平调度 3 种负载均衡策略[13]，先进先出调度过程最简单，但平台一次只运行一个任务，整体资源利用率较低. 雅虎开发的基于计算能力的调度方法通过定义多个作业队列，为各队列虚拟出指定的集群资源，队列内资源由该队列中所有作业共享，作业提交时放入相应队列，具有较好的灵活性. Facebook 开发的公平调度通过资源池(pool)组织作业，默认每个用户拥有一个独立的 pool，集群资源在 pool 之间公平分配，不管提交作业的多少都可以保证每个用户都分得均等的资源.

本文研究按周期收集节点实际负载信息，获取节点上正在运行的任务其执行情况；综合考虑任务信息与节点性能，预测某时刻节点剩余任务负载；根据待分配任务资源消耗特性和节点预测剩余负载等信息，将任务分配给相对负载较轻的节点. 研究应用于 Linux 系统下搭建的 Hadoop 云服务平台，该平台主要提供海量视频文件存储与在线处理服务. 系统总体架构如图 1 所示，云服务平台对外提供视频文件压缩、上传下载、在线播放、内容识别等服务，客户端提交任务至负载均衡服务器，服务器在线选择合适的云服务节点分配任务.

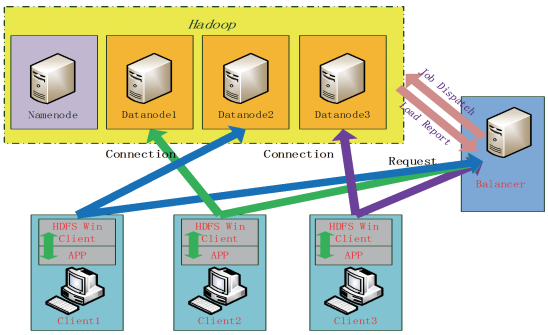


图 1 海量视频文件处理云服务平台系统架构

Fig. 1 Massive video file processing system architecture of cloud services platform

1 云计算任务分类

负载均衡算法的普遍思路是将任务分配给剩余负载较轻的节点，一般的动态分配方法只考虑了节点的综合负载情况，而对节点性能差异以及任务本身资源消耗特性关注较少. 实际上，节点在处理不同的任务请求时，CPU、内存、网络带宽等资源的消耗

情况不尽相同. 根据任务对 CPU、内存和网络带宽等资源消耗情况，可以分为偏好计算的任务、偏好网络的任务、偏好计算和网络的任务等[14].

显然，同样一个任务请求在相同的软硬件环境下，其 CPU 时间、内存消耗、网络传输量均大致相同. 在内存资源足够满足任务执行的情况下，完成任务的时间通常只与 CPU 资源和网络资源的占用情况相关，因此本文假设各节点内存资源足够使用，将任务 j 的大小用向量 $\boldsymbol{L}(j)=(\boldsymbol{L}_{\mathrm{CPU}}(j),L_{\mathrm{UL}}(j),L_{\mathrm{DL}}(j))$ 表示，其中 $L_{\mathrm{CPU}}(j)$、$L_{\mathrm{UL}}(j)$、$L_{\mathrm{DL}}(j)$ 分别代表 CPU 时间、上传总流量、下载总量流. 假设任务 j 在时刻 t_{begin} 开始执行，t_{end} 执行完成，则有：

$$\left.\begin{aligned}L_{\mathrm{CPU}}(j)&=\int_{t_{\mathrm{begin}}}^{t_{\mathrm{end}}}\mathrm{CPU}_p(j)\Delta t\\ L_{\mathrm{UL}}(j)&=\int_{t_{\mathrm{begin}}}^{t_{\mathrm{end}}}\mathrm{UL}_p(j)\Delta t\\ L_{\mathrm{DL}}(j)&=\int_{t_{\mathrm{begin}}}^{t_{\mathrm{end}}}\mathrm{DL}_p(j)\Delta t\end{aligned}\right\}\tag{1}$$

$\mathrm{CPU}_p(j)$、$\mathrm{UL}_p(j)$、$\mathrm{DL}_p(j)$ 分别表示任务 j 在某时刻的 CPU 使用率、上传带宽占用率、下载带宽占用率. 同样一个计算任务在不同性能的节点上执行时，测得的 CPU 时间是不同的，而网络流量通常相同. 选取云平台中一个节点，统计一段时间内对外提供的视频上传、视频压缩、在线播放、人物识别四类任务的资源消耗总量，结果如图 2 所示. 虽然任务实际大小各不相同，但总体而言同一类任务资源消耗分布情况相近. 观察图 1 可发现，视频压缩与人物识别任务，CPU 资源消耗较多，网络流量较少，CPU 资源往往成为瓶颈资源，属计算密集型任务；视频上传与在线播放任务，网络流量消耗较大，CPU 资源消耗较少，网络资源往往成为瓶颈，属网络密集型任务. 记 $B(j)\in\{\mathrm{CPU},\mathrm{UL},\mathrm{DL}\}$，用来标记任务 j 的瓶颈资源类型.

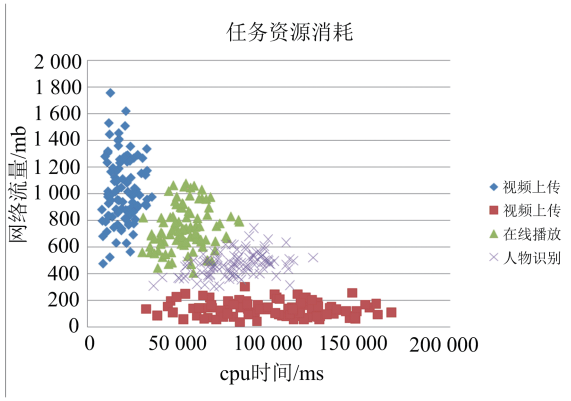


图 2　任务资源消耗统计

Fig. 2　Task resource consumption statistics

任务实际大小很难提前获得准确信息，可取任务平均资源消耗作为任务大小的估测值[15]. 向量 $\overline{\boldsymbol{L}(J)}=(\overline{L_{\mathrm{CPU}}(J)},\overline{L_{\mathrm{UL}}(J)},\overline{L_{\mathrm{DL}}(J)})$ 表示任务平均大小. 具体数值首先通过经验数据来给定，然后在算法运行过程中通过统计自学习更新.

2　负载预测算法

文献[10,12]中指出，频繁获取节点实时负载状况会加重网络和节点负担，影响任务分配响应时间. 如果能根据已知节点负载信息准确预测之后某时刻节点负载状况，将会对任务分配起到积极作用. 节点剩余负载可视为节点上正在运行的任务总资源需求量. 任务完成量与任务资源占用情况和运行时间相关，这种相关性为剩余负载预测提供了依据.

2.1　单个节点负载预测

节点负载变化需要考虑以下问题：①在周期时间内，各节点当前任务有部分已正常结束，负载减少. ②在周期时间内，每分配给节点一个请求，都会加重这一节点的负载.

节点 S_i 上某一时刻 t_0 正在运行的任务用集合 $J(S_i)=\{j_1,j_2,\cdots,j_n\}$ 表示，这些任务的开始时刻用集合 $T=\{t_1,t_2,\cdots,t_n\}$ 表示. 通过监测可获得节点上正在运行的任务 $j_k(j_k\in J(S_i))$ 在 t_0 时刻的完成情况 $L'(j_k,t_0)=(L'_{\mathrm{CPU}}(j_k,t_0),L'_{\mathrm{UL}}(j_k,t_0),L'_{\mathrm{DL}}(j_k,t_0))$，$L'_{\mathrm{CPU}}(j_k,t_0)$、$L'_{\mathrm{UL}}(j_k,t_0)$、$L'_{\mathrm{DL}}(j_k,t_0)$ 表示任务已经消耗的总 CPU 时间、总上传流量、总下载流量. 若要预测任务 $j_k(j_k\in J(S_i))$ 在 t_0 之后的某一时刻 t 的完成情况 $L'(j_k,t)=(L'_{\mathrm{CPU}}(j_k,t),L'_{\mathrm{UL}}(j_k,t),L'_{\mathrm{DL}}(j_k,t))$，根据式(1)可得

$$\left.\begin{aligned}L'_{\mathrm{CPU}}(j_k,t)&=L'_{\mathrm{CPU}}(j_k,t_0)+\int_{t_0}^{t}\mathrm{CPU}_p(j_k)\Delta t\\ L'_{\mathrm{UL}}(j_k,t)&=L'_{\mathrm{CPU}}(j_k,t_0)+\int_{t_0}^{t}\mathrm{UL}_p(j_k)\Delta t\\ L'_{\mathrm{DL}}(j_k,t)&=L'_{\mathrm{CPU}}(j_k,t_0)+\int_{t_0}^{t}\mathrm{DL}_p(j_k)\Delta t\end{aligned}\right\}\tag{2}$$

此时，任务 $j_k(j_k\in J(S_i))$ 在节点上的剩余负载 $L^*(j_k,t)=L(j_k)-L'(j_k,t)$. 由于任务实际大小并不确定，并且任务的开始时间各不相同，任务可能在 t_0 至 t 之间的任一时刻结束. 任务所属类型是已知的，因此为获得一个大致准确的剩余负载预测值，使用任务所属分类的任务平均大小作为估算任务大小，而 CPU 使用率、上传带宽占用率、下载带宽占用率使用在 t_0 时刻监测到的实际值，因此节点上的剩余负载 $L^*(j_k,t)$ 的定义如下：

$$L^*(j_k,t)=\begin{cases}\overline{L(J)}-L'(j_k,t), & \overline{L(J)}>L'(j_k,t)\\ 0, & \overline{L(J)}\leqslant L'(j_k,t)\end{cases} \tag{3}$$

将节点 S_i 上所有任务剩余负载相加得到节点 S_i 在时刻 t 的总剩余负载大小，即

$$L^*(J(S_i),t)=\sum_{j_k\in J(S_i)}L^*(j_k,t) \tag{4}$$

为了验证预测是否可行，选取云计算平台中的一个节点进行实验，该节点硬件配置为 Intel Core i3 4150 3.5GHz 双核 CPU，千兆网卡. 在 40 秒时间内随机分配 10 个视频压缩任务，根据式(3)和(4)预测节点剩余负载. 记录 1 分钟内实际 CPU 剩余负载和预测 CPU 剩余负载，结果如图 3 所示. 实验结果表明，预测 CPU 剩余负载与实际 CPU 剩余负载基本相当，预测剩余负载可在一定程度上反映节点所分配任务的剩余负载情况.

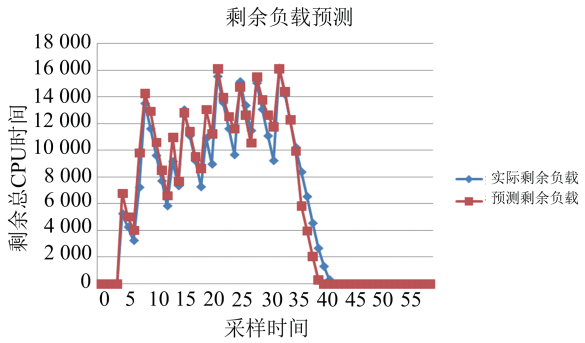


图 3 CPU 剩余负载预测

Fig. 3 CPU residual load forecast

2.2 异构节点负载预测评估

云计算平台中的节点可能存在异构的情况，一方面存在硬件上的性能差别，另一方面在软件环境上也有差异. 同一任务请求，在不同性能不同配置的节点上运行测得的 CPU 时间是不同的，因此在预测节点负载时必须考虑节点性能差异. 为了将预测结果标准化，在获取任务平均大小时统计某一个特定节点或几台性能、配置相同的节点，此类节点称为标准节点，记为 S_{std} . 将任意节点 S_i 的性能表示为

$$C(S_i)=(C_{\text{CPU}}(S_i),C_{\text{UL}}(S_i),C_{\text{DL}}(S_i)).$$

令 $C(S_{\text{std}})=(1,1,1)$，节点 S_i 的性能向量 $C(S_i)$ 中的三个值分别取节点 S_i 与标准节点 S_{std} 的 CPU 性能、上传总带宽、下载总带宽比值. 同样，根据单个任务剩余负载的预测方法，任务平均大小与异构节点性能参数比值减去预估已消耗资源大小既为任务剩余负载，由式(3)和(4)推出

$$L^*_{S_i}(j_k,t)=\begin{cases}\overline{L(J)}/C(S_i)-L'(j_k,t), & \\ \overline{L(J)}/C(S_i)>L'(j_k,t) & \\ 0,\overline{L(J)}/C(S_i)\leqslant L'(j_k,t) & \end{cases} \tag{5}$$

$$L_{S_i}{}^*(J(S_i),t)=\sum_{j_k\in J(S_i)}L_{S_i}{}^*(j_k,t) \tag{6}$$

将 2.1 节实验中的节点作为标准节点，选取云平台中另一个节点作为测试节点. 该节点硬件配置为 Intel Core i5 4670 3.4GHz 四核 CPU，千兆网卡. 在 40 秒时间内分配与 2.1 节实验中相同的任务至测试节点. 根据 CPU Benchmark 评分，测试节点 CPU 性能与标准节点 CPU 性能比约为 1∶5∶1，采用式(5)和(6)预测测试节点上的剩余负载. 记录 1 分钟内实际剩余负载和预测剩余负载，并与标准节点上的结果进行对比，结果如图 4 所示.

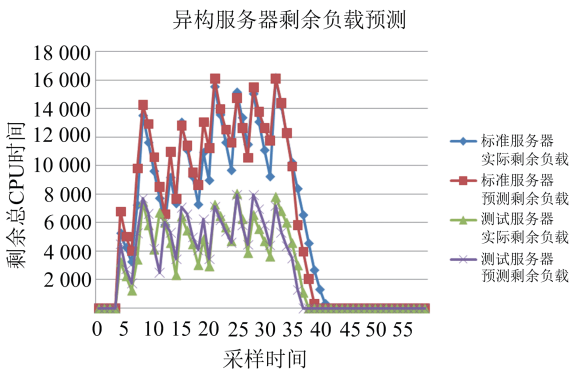


图 4 异构服务器负载预测

Fig. 4 Heterogeneous server load forecasting

由于测试节点 CPU 性能更优，无论是实际剩余负载还是预测剩余在数值上均小于标准节点，本节方法在异构节点上得到的预测剩余负载与实际剩余负载基本相当，预测剩余负载可在一定程度上反映节点所分配任务的剩余负载情况.

3 负载均衡算法

负载均衡算法的目标是使新分配至节点的任务能获得足够的系统资源. 为达到此目标，需要综合考虑节点的性能参数与节点的负载状况[16]. 根据上文的分析，通过任务资源消耗特征和节点性能可有效预测节点各种资源的剩余负载大小. 为使分配任务后各节点负载尽可能均衡，分配目标设定为将各节点平均负载方差率控制在最小. 云服务器集群由一台负载均衡服务器和 n 台提供服务的节点组成，这些节点记为集合 $S=(S_1,S_2,\cdots,S_n)$，各服务节点性能记为集合 $C=(C(S_1),C(S_2),\cdots,C(S_n))$.

任务 j 分配给节点 S_i 后，平均负载方差率为

$$\overline{\sigma(S_i,j)^2}=\Big(\sum_{S_k\in S}\Big(\frac{\overline{L_{\mathrm{CPU}}{}^*(S)}-L_{\mathrm{CPU}}{}^*(S_k)}{\overline{L_{\mathrm{CPU}}{}^*(S)}}\Big)^2+\sum_{S_k\in S}\Big(\frac{\overline{L_{\mathrm{UL}}{}^*(S)}-L_{\mathrm{UL}}{}^*(S_k)}{\overline{L_{\mathrm{UL}}{}^*(S)}}\Big)^2+\sum_{S_k\in S}\Big(\frac{\overline{L_{\mathrm{DL}}{}^*(S)}-L_{\mathrm{DL}}{}^*(S_k)}{\overline{L_{\mathrm{DL}}{}^*(S)}}\Big)^2\Big)/n \quad (7)$$

待分配任务 j 提交至负载均衡服务器后，根据式(7)计算任务分配至各节点后各节点的平均负载方差率，选取负载方差率最小的节点分配任务 j.

3.1　算法流程

负载均衡服务器维护一张节点任务负载表，该表记录各服务节点当前所有正在运行任务的剩余负载大小和节点报告的该任务资源占用率. 系统开始运行后，具体算法描述如下：

(Ⅰ)节点按指定周期报告最新任务负载信息，均衡服务器用接收到的实际负载信息更新节点任务负载表.

(Ⅱ)每到达一个实时任务请求，根据任务负载表中各任务情况，按照式(6)计算此时刻各节点预测剩余负载大小.

(Ⅲ)按照式(7)计算任务分配至各节点后的平均负载方差率.

(Ⅳ)将任务分配给节点平均负载方差率最小的节点.

(Ⅴ)在节点任务负载表中更新被分配任务节点的任务负载情况.

3.2　实验分析

算法的测试环境为一个 5 节点的云计算平台，其中选取 1 个节点为负载均衡服务器，其他 4 个节点提供服务. 为了检验算法对异构节点云计算平台的均衡效果，服务节点由四台性能不同的服务器组成，四个节点 CPU 性能比为 1∶1.2∶1.4∶2(按照 CPU Benchmark 平均得分计算)，而内存与网卡使用相同配置. 同时提供视频压缩(计算任务)与视频

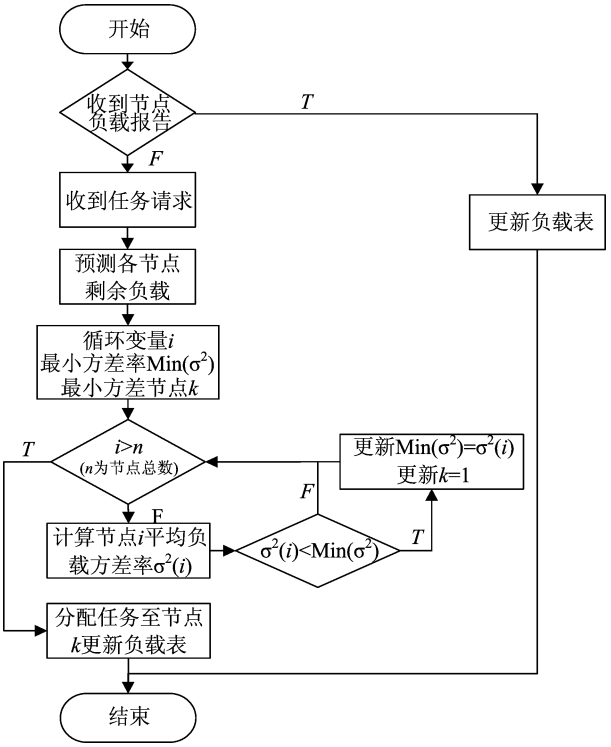
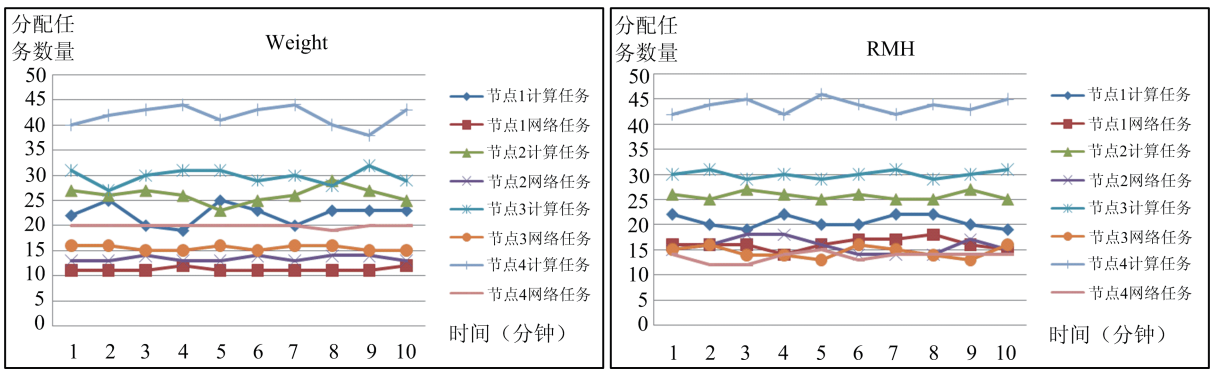


图 5　负载均衡算法流程图

Fig. 5　The flowchart of load balancing algorithm

文件传输服务(网络任务). 客户端由 60 个节点组成，统一向负载均衡器发出任务请求，其中 40 台客户端发送计算任务请求，20 台客户端发送网络任务请求；每台客户端在 1 分钟内随机发送 3 次任务请求，实验共进行 10 分钟. 根据实验设定的任务分配数量，考虑到既需要及时更新修正结点负载，又不过度加重节点与网络负担，服务节点每隔 20 秒向负载均衡服务器报告自身负载率情况. 分别对文献[12]中基于负载权值的动态负载均衡算法(Weight)和本文提出的实时多任务异构负载均衡算法(RMH)进行实验比较，两种方法每分钟分配到各服务节点的任务量，如图 6 所示.

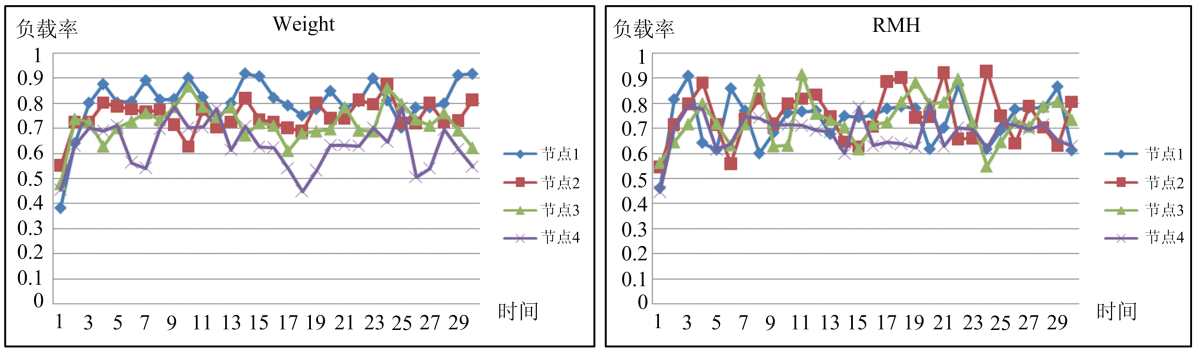


(a)Weight算法　　(b)RMH算法

图 6　分配任务数

Fig. 6　The number of assigned tasks

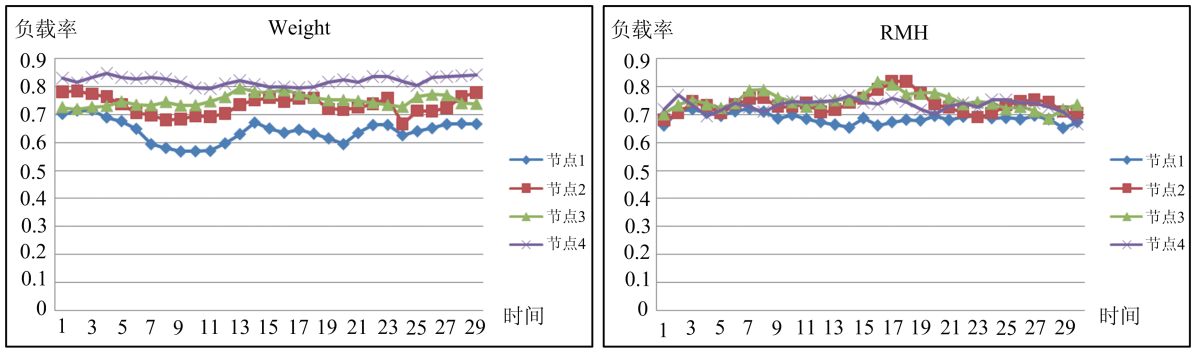
为测试任务分类对算法的影响，统计两种算法下各节点每分钟 CPU 平均负载率和网络平均负载率，结果如图 7、图 8 所示.



(a)Weight算法　　(b)RMH算法

图 7　CPU 平均负载率

Fig. 7　CPU average load rate



(a)Weight算法　　(b)RMH算法

图 8　网络平均负载率

Fig. 8　Network average load ratio

表 1　节点平均负载情况表

Tab. 1　Average node load

节点	节点 1		节点 2		节点 3		节点 4	
算法	Weight	RMH	Weight	RMH	Weight	RMH	Weight	RMH
CPU 平均负载	82.6%	71.7%	77.9%	68.9%	67.3%	66.1%	60.2%	67.4%
网络平均负载	63.2%	73.2%	71.1%	72.6%	73.8%	73.0%	81.5%	70.9%

由表 1 可以看到，两种算法都能区分服务器性能和负载. 负载权值算法不区分任务类型，网络任务的分配比例与计算任务分配比例相当. 而本文算法对每个节点分配的网络任务基本相当，而计算任务与节点 CPU 计算能力成正比. 表中显示节点 1 使用负载权值算法时 CPU 负载率较高而网络负载率较低、节点 4 则 CPU 负载率较低而网络负载率较高；本文算法各项资源负载率相对差异较小，原因是负载权值算法不区分任务类型，使节点 1 分配了过多的计算任务，而节点 4 分配了过多的网络任务. 本文算法各节点负载相对更均衡，较好地解决了负载分配不均的问题，提高了负载均衡效率.

4　结论

本文在研究常用云计算平台负载均衡算法的基础上，提出了一种实时多任务异构云计算平台负载均衡算法. 算法的主要特点包括：任务按照资源需求特点进行分类；考虑了节点剩余资源与任务资源需求的匹配；结合节点性能差异，预测任务分配后节点负载情况，任务能在较短响应时间内分配到节点；节点定期报告自身负载情况，及时修正负载预测偏差. 实验表明，本文提出的算法基本可行并优于常用负

载均衡算法,但在任务分配给节点后节点负载预测准确性还有提升空间,有待于进一步改进.由于受硬件资源的限制,本文的实验规模有限,算法的有效性与可靠性还有待进一步的实际检验.

参考文献(References)

[1] 张建勋,古志民,郑超.云计算研究进展综述[J].计算机应用研究,2010,27(2):429-433.

[2] 陈康,郑纬民.云计算:系统实例与研究现状[J].软件学报,2009,20(5):1337-1348.

[3] BONALD J, ROBERTS J. Multi-resource fairness: Objectives, algorithms and performance[J]. ACM SIGMETRICS Performance Evaluation Review, 2015, 43(1): 31-42.

[4] 胡志刚,张艳平.基于目标约束的分层动态负载均衡算法[J].计算机应用研究,2011,28(3):1105-1107.

[5] BRYHNI H, KLOVNING E, KURE O. A comparison of load-balancing techniques for scalable Web servers[J]. IEEE Network: The Magazine of Global Internetworking, 2000, 14(4): 58-64.

[6] KELLER M, KARI H. Response time-optimized distributed cloud resource allocation[C]// Proceedings of the ACM SIGCOMM workshop on Distributed cloud computing. Chicago, USA: ACM Press, 2014:47-52.

[7] PEARCE O, GAMBLIN T, DE SUPINSKI B, et al. Quantifying the effectiveness of load balance algorithms [C]// Proceedings of the 26th ACM International Conference on Supercomputing. Venice, Italy: ACM Press, 2012: 185-194.

[8] 张宇翔,张宏科.一种层次结构化 P2P 网络中的负载均衡方法[J].计算机学报,2010,33(9):1580-1590.

[9] RODRIGUES E R, NAVAUX P O A, PANETTA J, et al. A comparative analysis of load balancing algorithms applied to a weather forecast model[C]// Proceeding of 22nd International Symposium on Computer Architecture and High Performance Computing. Petrópolis, Brazil: IEEE Press, 2010: 71-78.

[10] YUN S Y, PROUTIERE A. Distributed proportional fair load balancing in heterogeneous systems[J]. ACM Sigmetrics Performance Evaluation Review, 2015, 43(1): 17-30.

[11] JOE-WANG C, SEN S, LAN T, et al. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework [J]. IEEE/ACM Transactions on Networking, 2013, 21(6): 1785-1798.

[12] 张玉芳,魏钦磊,赵膺.基于负载权值的负载均衡算法[J].计算机应用研究,2012,29(12):4711-4713.

[13] ZAHARIA M. Job scheduling with the fair and capacity schedulers[J]. Hadoop Summit, Santa Clara, USA, 2009: 9-18.

[14] 陈廷伟,周山杰,秦明达.面向云计算的任务分类方法[J].计算机应用,2012,32(10):2719-2723,2727.

[15] SHI W J, ZHANG L Q, WU C, et al. An online auction framework for dynamic resource provisioning in cloud computing[J]. ACM SIGMETRICS Performance Evaluation Review, 2014, 42(2): 71-83.

[16] BEAUMONT O, MARCHAL L. Analysis of dynamic scheduling strategies for matrix multiplication on heterogeneous platforms[C]// Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing. Vancouver, Canada: ACM Press. 2014: 141-152.