在线多任务异构云服务器负载均衡算法研究

徐爱萍[1] 吴 笛[1,2] 徐武平[1] 陈 军[1]
(武汉大学计算机学院 武汉 430072)[1] (解放军军事经济学院基础部 武汉 430035)[2]

摘 要 针对由于云服务器之间软件环境存在异构性及数据分布不均匀等特点而导致云服务器集群在处理大量任务时往往出现节点负载不均衡的情况,提出了解决在线多任务异构云服务器集群负载均衡的方法与相关算法。首先统计集群提供的各类服务的平均资源消耗,结合任务在服务器上已运行时长和资源占用情况,预测评估某一时刻服务器上任务剩余负载总量;然后按周期获取节点实际任务负载情况,及时修正任务负载情况;最后综合考虑节点各项性能,计算在待分配任务提交时刻各节点的预测负载评估值,并将任务分配给预测负载最轻的节点。实验结果表明,该算法具有可行性且在多任务异构云服务器集群负载均衡方面具有一定优势。

关键词 异构集群,云服务器,负载均衡,负载预测
中图法分类号 TP202 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.6.010

Research on Online Multi-task Load Balance Algorithm in Cloud Server Cluster

XU Ai-ping[1] WU Di[1,2] XU Wu-ping[1] CHEN Jun[1]
(School of Computer, Wuhan University, Wuhan 430072, China)[1]
(Department of Basic Knowledge, Military Economy Academy, Wuhan 430035, China)[2]

Abstract In this paper we presented a load-balancing algorithm applying in heterogeneous cloud server clusters. Average hardware resource consumption of jobs running on server was measured. Balancing server receives load status of each server in cluster periodically. A load status vector of each server can be estimated according to the latest load status report and other parameters. As a request is submited to cluster, balancing server calculates the load status estimation vector of each server, and then dispatches it to the server that possesses the minimal load status estimation value. Experiment results show that this dynamic load balancing algorithm is reasonable and effective.

Keywords Heterogeneous cluster, Cloud server, Load balance, Load estimate

1 引言

随着大数据时代来临,与之相关的技术成为目前最热门的研究领域,云计算作为大数据核心技术,近年来受到的关注空前,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及融合的产物[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服务器集群中。集群中各服务器的硬件设备通过云计算平台集合起来协同工作,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。由于各服务器之间的软件环境具有异构性、数据的分布不均匀等特点,云服务器集群在处理大量任务时往往会出现节点负载不均衡的情况,从而影响了服务质量。因此,各节点间的负载均衡成为云计算中的一个重要问题。负载均衡的目标是根据各服务器性能来分配与其匹配的任务,最大限度发挥集群的优势来提供更好的云服务质量,充分利用服务器的各种资源,最小化应用的执行时间,其属于最优化的理论范畴[3]。

现有的负载均衡算法主要分为静态和动态[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[6]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了研究,提出了很多改进的动态负载均衡算法,然而许多负载均衡研究都是基于特定应用环境下的[7],如张宇翔等人提出的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,国防科技大学的刘健等人提出基于动态反馈的负载均衡算法[10],该算法考虑服务

到稿日期:2015-06-22 返修日期:2015-08-03 本文受湖北省重大科技创新计划项目(2013AAA020),国家水体污染控制与治理科技重大专项(2013ZX07503-001-06)资助。

徐爱萍 教授,博士生导师,主要研究方向为云存储、分布式数据集成,E-mail:61580125@qq.com;吴 笛 博士生,主要研究方向为云计算、云存储;徐武平 博士,副教授,主要研究方向为云存储、分布式数据集成;陈 军 教授,博士生导师,主要研究方向为多媒体网络通信、云计算。

节点的性能与服务节点实际负载两方面的因素，用来指导任务的分配，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，且该方法对异构集群的支持效果并不理想。Tianyu Wo 等人的研究利用性能监控与任务调试中间件，来解决异构网格计算中的负载均衡问题，取得了良好效果[11]。张玉芳等人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，其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。

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的执行情况；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。研究应用于在 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。

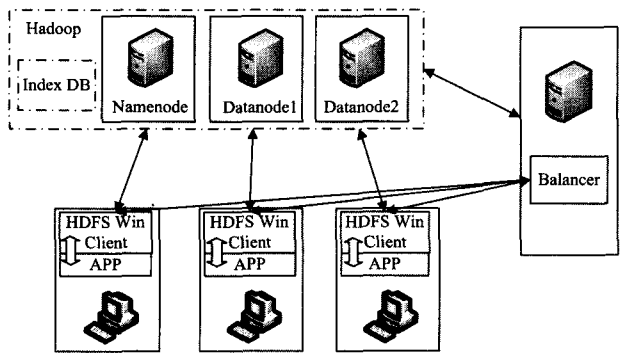


图 1　海量视频文件处理云服务平台系统架构

2　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，一般在分配任务时只考虑了服务器性能差异，而对任务本身资源消耗特性关注较少。实际上，云服务器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的消耗情况，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[13]。

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等。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，因此本文假设服务器内存资源足够使用，将任务 j 的大小用向量 $L(j)=(L_{CPU}(j),L_{UL}(j),L_{DL}(j))$ 表示，其中 $L_{CPU}(j)$、$L_{UL}(j)$、$L_{DL}(j)$ 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：

$$\begin{cases} L_{CPU}(j)=\int_{t_{begin}}^{t_{end}} CPU_p(j)\Delta t \\ L_{UL}(j)=\int_{t_{begin}}^{t_{end}} UL_p(j)\Delta t \\ L_{DL}(j)=\int_{t_{begin}}^{t_{end}} DL_p(j)\Delta t \end{cases} \tag{1}$$

其中，$CPU_p(j)$、$UL_p(j)$、$DL_p(j)$ 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服务器上执行时，测得的 CPU 时间是不同的，而网络流量通常相同，服务器性能差异对测量任务大小的影响在下节讨论。选取集群中一台云服务器，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 4 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的资源消耗分布相对集中。观察图 2 可发现，视频压缩与人物识别任务的 CPU 资源消耗较多，网络流量较少，CPU 资源往往成为瓶颈资源，属于计算密集型任务的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，属于网络密集型任务。记 $B(j)\in\{CPU,UL,DL\}$，用来标记任务 j 的瓶颈资源类型。

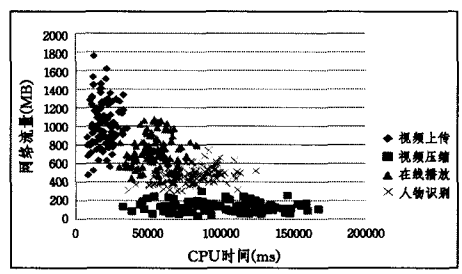


图 2　任务资源消耗统计

很难提前获得任务实际大小的准确信息，可取任务平均资源消耗作为任务大小的估测值。向量 $\overline{L(J)}=(\overline{L_{CPU}(J)},\overline{L_{UL}(J)},\overline{L_{DL}(J)})$ 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。

3　负载预测算法

研究[10,12]指出，频繁获取服务器实时负载状况会加重网络和服务器负担，影响任务分配响应时间。如果能根据已知服务器负载信息准确预测之后某时刻服务器的负载状况，将会对任务分配起到积极作用。服务器剩余负载可视为服务器上正在运行的任务总的资源需求量。如上所述，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。

3.1　单个节点负载预测评估

服务器负载变化需要考虑以下问题：1)在周期时间内，各服务器当前任务有部分已正常结束，负载减少。2)在周期时间内，每分配给服务器一个请求，都会加重该服务器的负载。

服务器 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。通过监测可获得服务器上正在运行的任务 j_k ($j_k\in J(S_i)$) 在 t_0 时刻的完成情况 $L'(j_k,t_0)=(L'_{CPU}(j_k,t_0),L'_{UL}(j_k,t_0),L'_{DL}(j_k,t_0))$，$L'_{CPU}(j_k,t_0)$，$L'_{UL}(j_k,t_0)$，$L'_{DL}(j_k,$

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 $L'(j_k,t)=(L'_{CPU}(j_k,t),L'_{UL}(j_k,t),L'_{DL}(j_k,t))$，根据式(1)可推出：

$$\begin{cases} L'_{CPU}(j_k,t)=L'_{CPU}(j_k,t_0)+\int_{t_0}^{t} CPU_p(j_k)\Delta t \\ L'_{UL}(j_k,t)=L'_{CPU}(j_k,t_0)+\int_{t_0}^{t} UL_p(j_k)\Delta t \\ L'_{DL}(j_k,t)=L'_{CPU}(j_k,t_0)+\int_{t_0}^{t} DL_p(j_k)\Delta t \end{cases} \tag{2}$$

此时，任务 $j_k(j_k \in J(S_i))$在服务器上的剩余负载 $L^*(j_k,t)=L(j_k)-L'(j_k,t)$。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，服务器上的剩余负载 $L^*(j_k,t)$的定义如式(3)所示。

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

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，选取集群中的一台云服务器进行实验，该服务器配置为 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)、式(4)预测服务器剩余负载。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。实验结果显示，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服务器所分配任务的剩余负载情况。上传网络流量与下载网络流量预测结果与 CPU 预测类似，不再另行说明。

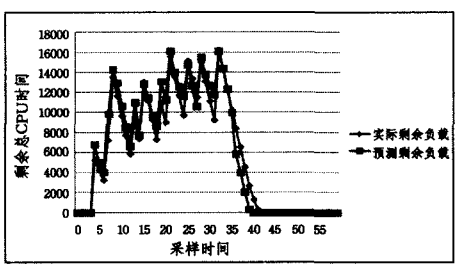


图 3　CPU 剩余负载预测

3.2　异构节点负载预测评估

服务器集群中的节点往往是异构的，一方面可能存在硬件上的性能差别，另一方面可能在软件环境上有差异。同一任务请求在不同性能、不同配置的服务器上测得的 CPU 时间是不同的，因此在预测服务器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台特定服务器或几台性能相同的服务器，此类服务器称为标准服务器，记为 S_{std}。将任意服务器 S_i 的性能表示为向量 $C(S_i)=(C_{CPU}(S_i),C_{UL}(S_i),C_{DL}(S_i))$。令 $C(S_{std})=(1,1,1)$，服务器 S_i 的性能向量 $C(S_i)$中的 3 个值分别取服务器 S_i 与标准服务器 S_{std} 的 CPU 性能、上传总带宽、下载总带宽比值。由式(3)、式(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 3.1 节实验中的服务器作为标准服务器，选取集群中另一台服务器作为测试服务器。该服务器配置为 intel core i5 4670 3.4GHz 四核 CPU，千兆网卡。在 40 秒时间内分配与 3.1 节实验中相同的任务至测试服务器。根据 CPU Benchmark 评分，测试服务器 CPU 性能与标准服务器 CPU 的性能比约为 1.5:1，采用式(5)、式(6)预测测试服务器上的剩余负载。记录 1 分钟内实际剩余负载和预测剩余负载，并与标准服务器上的结果进行对比，结果如图 4 所示。

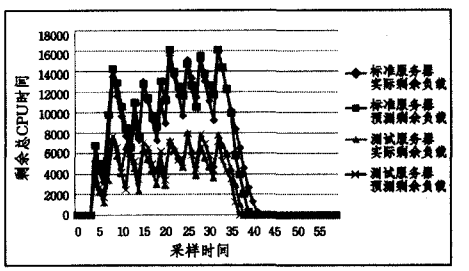


图 4　异构服务器负载预测

由于测试服务器 CPU 性能更优，无论是实际剩余负载还是预测剩余负载在数值上均小于标准服务器，本节方法在异构服务器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服务器所分配任务的剩余负载情况。

4　负载均衡算法

本研究中负载均衡算法的目标是让各服务器之间的负载量尽可能达到平衡，使新分配至服务器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。根据前面两节的分析，通过任务资源消耗特征和节点性能可有效预测服务器各种资源的剩余负载大小。对任务完成时间影响最大的往往是瓶颈资源，本文算法根据服务器资源剩余负载大小和任务瓶颈资源类型分配任务，尽可能满足任务的瓶颈资源需求。

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 在 t 时刻提交到集群中，其分配节点记为 $S(j)$，$S(j)$满足：

$$\begin{cases} S(j)\in S, B(j)=CPU & L^*_{CPU}(S(j),t)=\min(L^*_{CPU}(S_i\in S,t)), \\ S(j)\in S, B(j)=UL & L^*_{UL}(S(j),t)=\min(L^*_{UL}(S_i\in S,t)), \\ S(j)\in S, B(j)=DL & L^*_{DL}(S(j),t)=\min(L^*_{DL}(S_i\in S,t)), \end{cases}$$

4.1　算法流程

负载均衡服务器维护一张节点任务负载表，该表记录了

在各服务节点当前所有正在运行的任务的剩余负载大小和节点报告的该任务的资源占用率。系统开始运行后，具体算法描述如下：

1)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；

2)每到达一个任务请求，根据任务负载表中各任务情况，按照第 3 节描述的方法计算此时刻集群各节点的预测剩余负载大小；

3)判断任务瓶颈资源类型，若瓶颈资源为 CPU 则转 4)，若瓶颈资源为网络上传带宽则转 5)，若瓶颈资源为网络下载带宽则转 6)；

4)将任务分配给 CPU 负载预测值最小的服务器；

5)将任务分配给网络上传带宽负载预测值最小的服务器；

6)将任务分配给网络下载带宽负载预测值最小的服务器；

7)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。

负载均衡算法流程如图 5 所示。

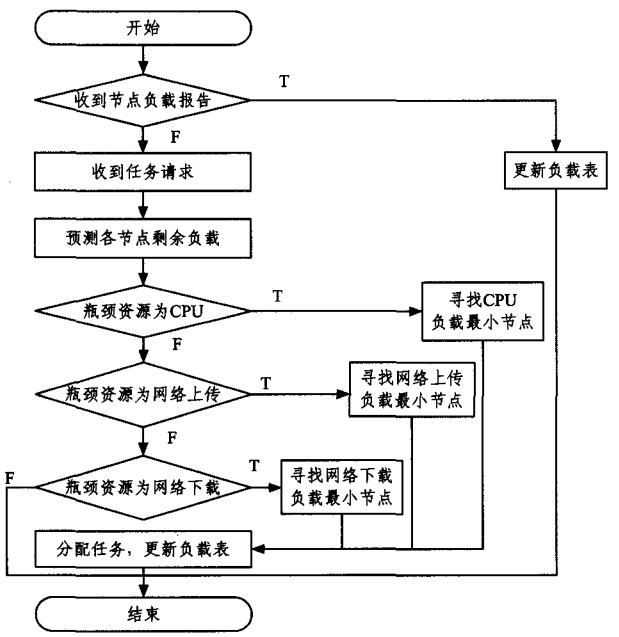
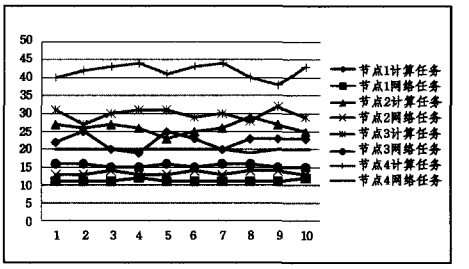


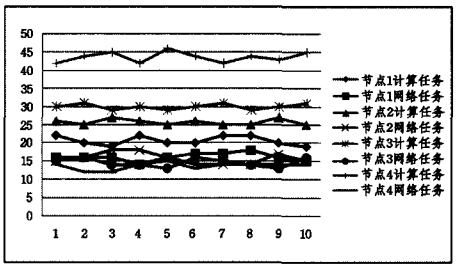
图 5　负载均衡算法流程

4.2　实验分析

对本文提出的算法进行测试，测试环境为一个 5 节点的云服务器集群，其中选取 1 个节点为负载均衡服务器，其它 4 台服务器提供服务。为了检验算法对异构节点集群系统的均衡效果，服务节点由 4 台性能不同的服务器组成，4 个节点的 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)与视频文件传输服务(网络任务)，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。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 分钟时间内随机发送 3 次任务请求，实验共进行 10 分钟。分别对文献[12]中基于负载权值的动态负载均衡算法(Weight)和本文提出的在线多任务异构负载均衡算法(OMH)进行实验。实验结果记录了两种方法在每分钟分配到各服务节点的任务量，如图 6 所示。



(a)Weight



(b)OMH

图 6　分配任务数

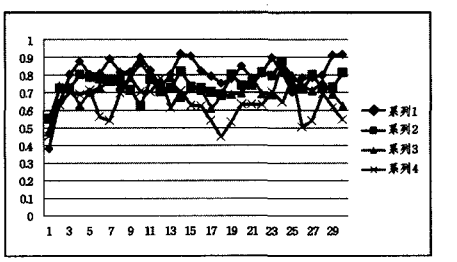
最终统计所有结点完成任务的最终时间，结果如表 1 所列。

表 1　节点完成所有任务时间

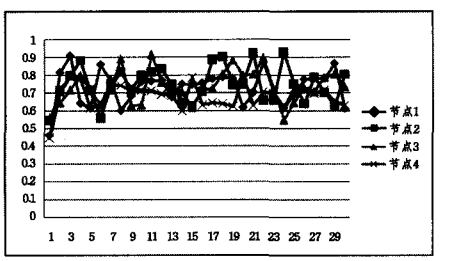
节点	节点 1		节点 2		节点 3		节点 4	
算法	Weight	OMH	Weight	OMH	Weight	OMH	Weight	OMH
完成时间	11min 26s	10min 45s	10min 39s	10min 58s	11min 09s	10min 47s	12min 09s	11min 10s

可以看到，两种算法都能区分服务器性能与负载。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。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 1 中显示节点 1 和节点 4 使用负载权值算法时完成所有任务的最终时间较长，而本文算法完成时间相对差异较小，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。

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

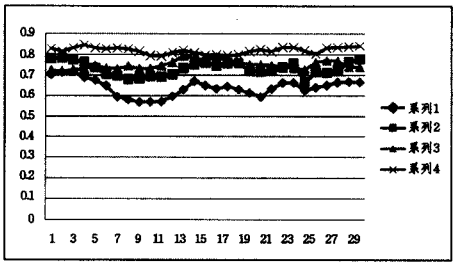


(a)Weight

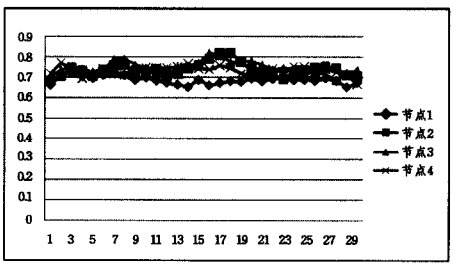


(b)OMH

图 7　CPU 平均负载率



(a)Weight



(b)OMH

图 8　网络平均负载率

统计各节点上所有任务占用的总 CPU 时间和总网络吞吐量，结果如表 2 所列。

表 2　节点总 CPU 时间与网络吞吐量

节点	节点 1		节点 2		节点 3		节点 4	
算法	Weight	OMH	Weight	OMH	Weight	OMH	Weight	OMH
总 CPU 时间(s)	577.9	504.3	544.5	481.7	477.6	462.3	399.2	471.1
总网络吞吐量(GB)	6.32	7.32	7.11	7.26	7.38	7.30	8.15	7.09

使用负载权值算法时节点 1 的 CPU 负载较重、网络负载较轻，而节点 4 的 CPU 负载较轻、网络负载较重。使用本文算法时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，增强了集群系统的稳定性和灵活性。

结束语　本文在研究常用服务器集群负载均衡算法的基础上，提出了一种在线多任务异构集群负载均衡算法。该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的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，本文中的实验规模有限，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

[1] Zhang Jian-xun, Gu Zhi-ming, Zheng Chao. A survey on CLOUD computing[J]. Application Research of Computers, 2010, 27(2): 429-433(in Chinese)
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] Chen Kang, Zheng Wei-ming. CLOUD computing: system instance and research status[J]. Journal of Software, 2009, 20(5): 1337-1348(in Chinese)
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] Chen Wei, Zhang Yu-fang, Xiong Zhong-yang. Implementation of load balancing algorithm on heterogeneous cluster dynamic feedback[J]. Journal of Chongqing University, 2010, 33(2): 73-78(in Chinese)
陈伟，张玉芳，熊忠阳. 动态反馈的异构集群负载均衡算法的实现[J]. 重庆大学学报，2010，33(2)：73-78

[4] Hu Zhi-gang, Zhang Yan-ping. Load balancing algorithm based on hierarchical dynamic constraint [J]. Application Research of Computers, 2011, 28(3): 1105-1107(in Chinese)
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] Bryhnih. A comparison of load-balancing techniques for scalable Webservers[J]. IEEE Network, 2000, 14(4): 58-64

[6] Mai Jing-jing, Gong Hong-yan, Song Chun-he. Cluster system in dynamic feedback load balancing strategy [J]. Computer Engineering, 2008, 34(16): 114-115(in Chinese)
买京京，龚红艳，宋纯贺. 集群系统中的动态反馈负载均衡策略[J]. 计算机工程，2008，34(16)：114-115

[7] Pearce O, Gamblin T. Quantifying the effectiveness of load balance algorithms[C]// Proceedings of the 26th ACM International Conference on Supercomputing. New York: ACM Press. 2012: 185-194

[8] Zhang Yu-xiang, Zhang Hong-ke. A load balancing method in structured P2P network [J]. Chinese Journal of Computers, 2010, 33(9): 1580-1590(in Chinese)
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. 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. Washington DC: IEEE Computer Society, 2010: 71-78

[10] Liu Jian, Xu Lei, Zhang Wei-ming. Load balancing algorithm based on dynamic feedback [J]. Computer engineering and Science, 2003, 25(5): 65-68(in Chinese)
刘健，徐磊，张维明. 基于动态反馈的负载均衡算法[J]. 计算机工程与科学，2003，25(5)：65-68

[11] Wo Tian-yu, Zhong Liang. A Bulletin-Board based Cooperative Load Balance Strategy for Service Grid[C]// Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid. Washington DC: IEEE Computer Society, 2007: 57-64

[12] Zhang Yu-fang, Wei Qin-lei, Zhao Ying. A load balancing algorithm based on the weight of the load [J]. Application Research of Computers, 2012, 29(12): 4711-4713(in Chinese)
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] Chen Ting-wei, Zhou Shan-jie, Qin Ming-da. Task classification method in Cloud Computing [J]. Journal of Computer Applications, 2012, 32(10): 2719-2723(in Chinese)
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