分布式系统节点负载动态平衡策略研究

胡 俊,洪 龙,沈春来
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:负载均衡是有关分布式系统性能和计算效率的一个关键问题。在分析了目前常用的集中式动态分配算法和分布式动态分配算法的利弊后,结合两种算法的优点,建立了一种半分布半集中式动态负载分配模型,并据此设计了相应算法。根据节点响应时间,对原有的动态分配算法和新模型进行了性能分析,详细叙述了改进算法的设计思想和实现步骤。讨论结果表明,新算法可以提高系统的客户平均响应时间,同时减少节点间交互的信息量,提高了分布式系统的资源利用率。

关键词:分布式;集中式;半分布式半集中式;负载平衡
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)02-0093-03

Research of Distributed System Node Dynamic Load Balancing Strategy

HU Jun,HONG Long,SHEN Chun-lai
(College of Computer,Nanjing University of Post & Telecommunications,Nanjing 210003,China)

Abstract:Load balancing is the key issue which is related to a distributed system performance and computational efficiency. It analyses two arithmetic's advantages and disadvantages about dynamic load which is centralized and distributed. What is more,it brings forward a new a ssigned model about dynamic load which is half distributed and half centralized based on the combining the advantages of both algorithm. According to the node's response time,the analysis of the performance of the original dynamic allocation algorithm and new assigned model was given,moreover,the improved algorithm design and the steps of the implementation have been described. The result shows that compared with the original dynamic load balancing algorithm,the new algorithm can improve the system of customer response time and reduce the amount of information interaction among nodes. Distributed system resource utilization rate can be increased.

Key words:distributed;centralized;half distributed and half centralized;load balancing

0 引 言

在分布式系统中节点可以简单看成是网络上的物理机器。由于系统中每个节点的处理能力不同,并且任务的分配具有一定的随机性,从而造成有些节点上的任务处理的较快,CPU 队列处于空闲的状态;而有些节点处理任务较慢,CPU 队列较长,处于忙状态。前者称之为节点的轻载,后者称之为节点超载。为了提高分布式系统的效率就必须保持节点间的负载平衡。

一般而言,解决分布式系统节点间的负载均衡方法有静态负载均衡方法[1,2]和动态负载均衡方法[3,4]。因为动态负载均衡可以用于任务不确定的情况,更具实用性,所以动态负载均衡比静态负载均衡更能改进系统的性能。因此文中以下出现的负载平衡指的都是动态负载平衡。目前国内外研究动态负载均衡大都从三个方面入手,即①收集各节点的负载信息;②信息决策;③节点任务的迁移[5~7]。而文中选取第一点作为研究对象,即研究各个节点间负载信息的收集。

文中简单介绍了目前常用的两种节点负载信息收集的算法—集中式和分布式算法。随后提出了一种改进的半集中半分布式的算法并分析其性能,得出优越性。

收稿日期:2011-07-05;**修回日期**:2011-10-20
基金项目:国家高技术研究发展计划(863 计划)(2009AA043303);软件开发环境国家重点实验室开放课题(SKLSDE-2011KF-04)
作者简介:胡 俊(1985-),男,安徽合肥人,硕士研究生,研究方向为分布式系统;洪 龙,教授,研究员级高级工程师,研究方向为分布式系统、非经典逻辑及应用。

1 集中式平衡算法和分布式平衡算法

1.1 基本概念与术语

服务器:实现特定服务的进程。例如,文件系统服务或数据库服务[8]。

客户:往服务器发送服务请求并等待服务器回复

的进程[8]。

节点:在分布式系统中,一个或多个功能单元与传输线路互连的一个点,节点可以看成是网络上的物理机器[9]。

1.2 集中式负载平衡算法

在集中式算法中,设置一个"负载均衡器"用于收集所有节点的负载信息并进行任务分配[10],如图1所示。

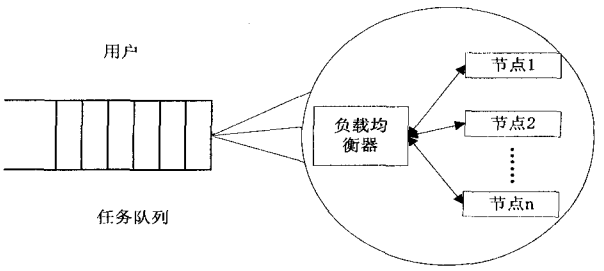

图1 集中式调度

在该系统中负载均衡器每隔一定周期 T 就会收集所有节点的负载信息,并存入列表 Node List 中。负载均衡器定时轮询每个节点的状态,更新 Node List 列表,并计算系统平均负载。节点将自身负载与系统设定的阈值做比较,如果超载则启动自身的服务器进程,将任务迁移至轻载的节点。集中式算法的优点是控制集中且简单易行。

但是由于收集负载信息的任务都由负载均衡器来完成,一旦负载均衡器出现故障,则系统将无法正常运行;另一方面,所有的任务都将通过"负载均衡器",无形中加重了系统负担,造成处理任务的时间大大增加。

1.3 分布式动态负载分配

在现实生活中,流言通常是A传播给B,然后B传播给C,同时A又传播给D,之后就变成A、B、C、D四个人传播,于是很快这个流言就传播开了,在这样的传播方式下,即使后面的A、B不传播了,其他人也会慢慢知道这个流言,只是速度可能会慢点。这种方式的好处在于可以不依赖任何一个点,无中心化。

分布式动态负载分配类似流言的传播,在这个系统中所有节点通过自身的服务器进程进行负载信息交互。并建立一张负载信息列表,用于维护所有节点的负载信息。如果有节点超载,则根据负载信息表,节点自身将任务转移至轻载节点。具有很强的容错能力,如图2所示。

分布式负载均衡类似于网络中的路由算法,通过节点自身之间的交互达到负载均衡。在分布式算法中所有节点都会按照周期T向其余节点广播自身的负载。在有 n 个节点的系统中,每个节点都会发送 $n(n-1)$ 条负载信息到其余各节点。因此,若节点 n 的规模过大,则系统中节点之间的交互信息量也随之变大,交

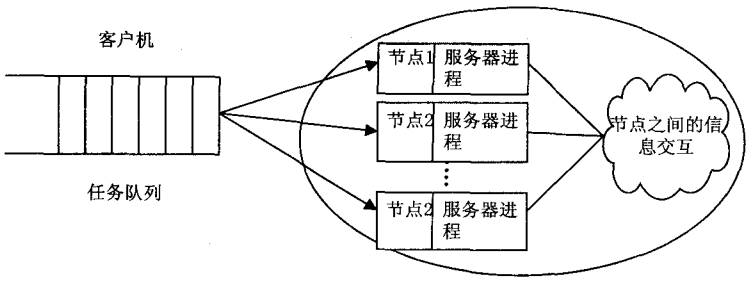

图2 分布式动态负载分配

互时间将大大延长,从而导致系统性能下降。另外,如果有多个超载节点将任务迁移至同一个轻载节点,就会造成轻节点负载急剧增大从而瘫痪,甚至有可能造成迁移抖动等新的问题[11,12]。

2 改进的动态负载均衡算法

通过前面的分析,提出一种改进的动态负载均衡算法,用于更好地提高分布式系统的运行效率。算法模型如图3所示。

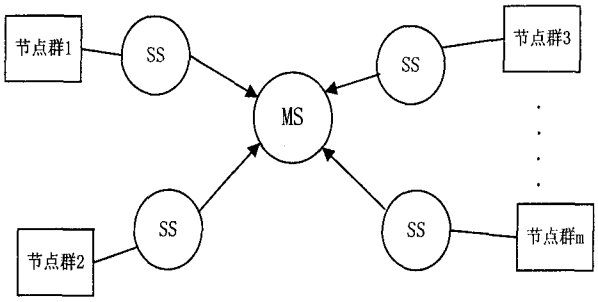

图3 半集中半分布式

图中服务器分成两类:

(1)主服务器进程(MS):用于收集各从服务器的负载信息的进程,一个。

(2)从服务器进程(SS):用分布式的方法获得子节点群的负载信息,并和MS进行负载信息交互,当超载时,自动转移负载的进程,多个。

(3)分布式系统节点:在分布式系统的每个节点中,都运行着一个从服务器的进程,用于系统的负载均衡。

在分布式系统中,考虑到现实生活中不可能保证各节点负载的绝对平衡,从而提出在一个小的范围内允许节点之间的负载存在一定的差异。这样做的好处有两点。第一点:可以简化问题的处理,设计出一个相对合理、在实际中易于实现的算法;第二点:避免节点因为轻微的负载不平衡而在节点之间频繁进行进程转移,因为频繁的进程转移和切换会增加系统的负担,反而降低了系统运行的效率。在这里假设系统中每个节点的最大负载都是 L_{max},将负载均等分为 W 个等级。而系统所允许节点之间负载差为 L_{max}/W。即如果各节

点之间负载差值在 L_{max}/W 内时,节点不进行任务迁移,超过这个范围则调用负载平衡算法。

因此根据以上的负载等级划分,在所有从服务器SS上建立一张负载信息表,在这张表中记录了运行该SS服务进程节点群的负载信息,表格样式如表1所示。

表1　SS负载信息表

节点编号	状态	等级
5	20	2
……	……	……
1	45	4

在表中"状态"表示某个节点当前需要处理的任务。"等级"表示按照升序排列MS的负载,其中MS记录的是SS下所有节点的负载之和;而SS负载信息表记录了每个节点的负载信息,也按照等级的升序排列。在主服务器进程MS中也建立一张负载信息表,保留所有SS的负载信息。表格样式如表2所示。

表2　MS负载信息表

SS编号	状态	等级
1	490	49
……	……	……
6	620	62

半分布式半集中式算法描述如下:

①SS处理算法:

If 节点负载发生变化

{

向SS汇报负载情况;

If 负载在可接受范围之内

{

在SS管理的节点群内部选择轻载节点,完成进程迁移,达到负载平衡;

}

If 负载超过上限

{

超载任务放入等待队列等待其它节点调入;

向MS汇报负载情况,MS根据信息负载表选择一个轻载SS,完成负载平衡。

}

If 负载低于下限

{

从SS中调入超载任务;

更新MS负载状态;

}

}

②主服务器进程MS处理算法:

接收来自各SS负载信息汇报;

执行从服务器进程SS算法。

在这个算法中,MS负责管理SS,而SS负载收集和管理它所在的节点区域。如果有节点超载,并且其超载的任务量小于所规定的阈值,则由SS负责在区域内部寻找负载相对较轻的节点完成进程迁移。如果节点超载的任务量大于设定的阈值,将节点负载信息交由SS,通过SS在传递给MS。最终由MS寻找负载相对较轻的SS完成任务迁移。

3　改进算法的性能分析

这里假设有 $T1 \sim T4$ 和 α 五个变量。其中:

$T1$ 表示Internet上数据传输的延迟时间。

$T2$ 表示节点中的任务排队等待处理时间。

$T3$ 表示节点间任务进程迁移时间。

$T4$ 表示节点处理任务时间和存储时间。

α 表示节点间进程迁移的概率$(0 < \alpha < 1)$。

假设所有任务都同时到达并且设总任务数为 K,在包含 N 个节点的集中式算法中,"负载均衡器"负责收集和管理所有节点的任务。则客户的平均响应时间为 $T1 \sim T4$ 总和,即:

$$T_{time-center} = T1 + T2 + T3 + T4$$
$$= T1 + (T3 * K + T4 * K/N) + T3 + T4$$
$$= T1 + (K+1)T3 + [(K/N) + 1]T4 \quad (1)$$

在分布式算法中客户的平均响应时间为:

$$T_{time-distributed} = T1 + T2 + \alpha T3 + T4$$
$$= T1 + T4 * K/N + \alpha T3 + T4$$
$$= T1 + \alpha T3 + (K/N + 1)T4 \quad (2)$$

在改进的半分布半集中式算法中,每个节点首先判断其负载的增加是否超过规定的阈值,所以客户平均响应时间为:

$$T_{time-half} = T1 + T2 + \alpha T3 + T4$$
$$= T1 + T4 * K/N + \alpha * T3 * W/L_{max} + T4$$
$$= T1 + (\alpha W/L_{max})T3 + [K/N + 1]T4 \quad (3)$$

由(1) ~ (3)式可知,$T_{time-center} > T_{time-distributed} > T_{time-half}$,改进后的算法响应时间较短。

节点间交互的信息量。在集中式算法中,每个节点间的负载信息都由"负载均衡器"收集,所以节点之间没有信息交互,即节点间交互的信息量为零。在分布式算法中,每个节点需要向整个系统中其余节点广播负载信息,所以在包含有 n 个节点的系统中,节点间交互的信息量为 $n(n-1)$。在包含 n 个节点的半分布半集中式算法中,节点的负载信息交由SS,一旦超过阈值时再由SS交给MS处理,所以节点与节点之间交互的信息量为 $2(n-1)$。

根据以上的分析对比可知,改进后的算法虽然在节点间交互的信息量要比集中式算法多,但是却比分

(下转第99页)

5　结束语

文中提出一种基于混合特征提取的人脸表情识别方法,对人脸图像提取 3 种不同的特征,并采用贡献分析法确定不同特征对各种表情的贡献权值。在分类识别时,分别采用离散的隐马尔科夫模型对每个特征进行表情识别,最后对三种特征计算得到的表情概率进行加权计算,得到识别结果。该方法对 6 种表情有着不错的识别率。

参考文献:

[1] Campadelli P, Lanzarotti R, Savazzi C. A feature-based face recognition system[C]//Proceedings of International Conference on Image Analysis and Processing. Mantova, Italy: [s. n.], 2003: 68-73.

[2] Alekaic P S, Katsaggelos A K. Automatic facial expression recognition using facial animation parameters and multistream HMMs[J]. IEEE Transactions on Information Forensics and Security, 2006, 1(1): 3-11.

[3] 卢春雨,张长水,闻　芳,等. 基于区域特征的快速人脸检测算法[J]. 清华大学学报(自然科学版), 1999, 39(1): 101-105.

[4] Akakin H C, Sankur B. DCT Based Facial Feature Extraction [C]//Proceedings of the 14th Nation Signal Processing and Applications Conference. Antalya, Turkey: [s. n.], 2006: 1-4.

[5] 王力维. 表情识别中预处理与人脸特征提取算法的研究[D]. 南京:东南大学, 2006.

[6] Buci U, Kotropoulos C, Pitas I. ICA and Gabor representation for facial expression recognition[C]//ICIP: 2003 International Conference on Image Processing. Washington, DC: IEEE Press, 2003: 855-858.

[7] 叶吉祥,胡秀丽. 基于 Gabor 小波与分形维的人脸感情特征提取[J]. 计算机工程, 2008, 34(4): 226-230.

[8] 王　磊,邹北骥,彭小宁,等. 一种改进的提取人脸面部特征点的 AAM 拟合算法[J]. 电子学报, 2006, 34(8): 1424-1427.

[9] Loh May-Ping, Wong Ya-Ping, Wong Chee-Onn. Facial expression recognition for e-learning systems using Gabor wavelet & neural network[C]//Proceedings of the Sixth International Conference on Advanced Learning Technologies. The Netherlands: IEEE Computer Society, 2006: 523-525.

[10] Abbound B, Davoine F, Dang M. Facial expression recognition and synthesis based on an appearance model[J]. Signal Processing: Image Communication, 2004, 19(8): 723-740.

[11] 詹永照,曹　鹏. 语音情感特征提取和识别的研究与实现[J]. 江苏大学学报, 2005, 26(1): 72-75.

[12] 王　晔,吴小俊,王士同,等. 基于改进主动形状模型的人脸表情识别[J]. 计算机工程与应用, 2009, 45(6): 197-200.

(上接第 95 页)

布式算法传输的信息量要小的多。并且由于其客户响应时间在三者中最短,因此该算法总体上要比分布式算法和集中式算法整体效率要高。

4　结束语

文中先是说明了动态负载平衡算法要解决的三个主要问题,并且选择其中之一即收集节点间的负载信息进行研究。在对比分析常用的算法即分布式算法和集中式算法之后,提出一种改进的半集中式半分布式算法。并且根据客户响应时间和节点间交互的信息量两个方面对改进后的算法进行性能分析,得出其优越性。通过分析可以得出这种算法具有良好的稳定性、可靠性。

参考文献:

[1] Efe k. Heuristic models of task assignment scheduling in distributed systems[J]. IEEE Computer, 1982, 15(6): 50-56.

[2] Chen Huaping, Li Jing, Chen Guoliang. Task scheduling in parallel distributed computing[J]. Computer Science, 1997, 24(2): 23-27.

[3] Diekmann R. Efficient schemes for nearest neighbor load balancing[J]. Parallel Comput, 1999, 25(7): 789-812.

[4] Yu K. Load Balancing Algorithm Using Production[C]//Aizu International Symposium on Parallel Algorithms/Architecture Synthesis. [s. l.]: [s. n.], 1997: 159-165.

[5] Legrand A. Mapping and load-balancing iterative computations[J]. Parallel and Distributed Systems, 2004, 6(3): 546-558.

[6] Barker K. A load balancing framework for adaptive and asynchronous applications[J]. Parallel and Distributed Systems, 2004, 2(1): 183-192.

[7] Devine K D, Boman E G, Heaphy R T, et al. New challenges in dynamic load balancing[J]. Applied Numerical Mathematics, 2005, 52(2-3): 133-152.

[8] Tanenbaum A S. Distributed Systems Principles and Paradigms [M]. 北京:清华大学出版社, 2008.

[9] 中国电子学会,电子计算机学会. 英汉计算机辞典[M]. 北京:人民邮电出版社, 1984.

[10] 林　昊. 分布式 java 应用基础与实践[M]. 北京:电子工业出版社, 2010: 228-236.

[11] Xu C, Lau F C M. Iterative dynamic load balancing in multicomputers[J]. Journal of the Operational Research Society, 1994, 45(7): 786-796.

[12] Cybenko G. Dynamic load balancing for distributed memory multiprocessors[J]. J Par Distr Comp, 1989, 7(2): 279-301.