基于数据库集群的动态负载均衡研究与实现*

何 骏[1],熊 伟[1],陈 荦[1],殷佳欣[2]
(1.国防科学技术大学 电子科学与工程学院,湖南 长沙 410073;
2.中国科学院 软件所,北京 100190)

摘 要:针对数据库集群负载问题,提出了一种动态负载均衡方法,并进一步设计、实现了包括CPU使用率、磁盘存储量、磁盘响应效率、网络延时、内存使用率等在内的多指标的节点负载测量和实时监控。该算法根据各节点的负载反馈信息进行任务分配,实现了负载均衡。性能分析和实验表明,该算法具有较高的负载均衡度和较低的系统开销。

关键词:动态负载均衡;数据库集群;负载测量;实时监控

中图分类号:TP393 **文献标识码**:B **文章编号**:1674-7720(2011)02-0068-04

Study and implementation of dynamic load balancing based on database cluster

He Jun[1], Xiong Wei[1], Chen Luo[1], Yin Jiaxin[2]
(1.College of Electronic Science and Engineering, National University of Defense Technology, Changsha 410073, China;
2.Institute of Software, Chinese Academy of Science, Beijing 100190, China)

Abstract: For the database cluster load problem, a dynamic load balancing algorithm is given. Multi-node load balancing measurement and real-time monitoring is designed and implemented, such as CPU utilization, storage capacity, disk response efficiency, network delay, memory utilization and so on. Through load information of the nodes, the algorithm can get load mission groups, and attains load balancing. Performance analysis and simulation results show the algorithm has lower system cost and better load balancing factor.

Key words: dynamic load balancing; database cluster; real-time monitoring

随着数据库技术的发展和各种数据库产品的产生,数据库系统在各行各业应用广泛,随着应用需求的不断增加,越来越多的用户希望能够透明地访问和处理来自多个数据库的数据。同时,电子商务和信息技术的迅猛发展使数据库管理系统(DBMS)不堪重负,将这些数据库有机地连接起来,统一管理,协调工作,不但能提高数据库的性能和可用性,而且解决了遗留系统的问题。对于这些通过网络连接起来的数据库而言,必须能够实时处理大量的用户请求,而且必须能向客户提供高质量服务。但是这些系统潜在性能的实际利用率通常仅为1%~10%[1],导致系统运行效率低下。如何才能提高由网络连接起来的数据库的响应速度、稳定性和扩展性,并且保护最初的硬件投资;如何避免大量用户请求对系统带来的冲击,负载均衡为设计者提供了一条途径,它是提高系统资源利用率的一个关键技术[2-3],它在后端数据库间分发客户请求,以达到减少系统瓶颈、增强系统响应能力。

1 负载均衡

对于一个分布式计算机系统,由于任务到达的随机性,以及各处理节点处理能力上的差异,当系统运行一段时间后,某些节点分配的任务还很多(称之为超载),而另一些节点却是空闲的(称之为轻载)。一方面,使超载节点尽可能快地完成任务是当务之急;另一方面,使某些节点空闲是一种浪费。如何避免这种空闲与忙等待并存的情况,从而有效地提高系统的资源利用率,减少任务的平均响应时间,促使负载均衡的产生与发展。

*基金项目:国家自然科学基金(40801160);国家自然科学基金(60902036);国家863项目(2007AA120402)

负载均衡设法对分配给各节点的任务进行重新调度，并通过进程迁移(又称为任务迁移)，使各节点负载大致相等。

目前普遍使用的数据库集群系统负载均衡方法包括：

(1)Random：即随机选择法。在后端中随机选择一个节点来执行用户查询请求；

(2)Round Robin：即轮询法。第一个请求发送到第一个后端，第二个请求发送到第二个后端，依此类推。不断地循环，直到请求又从第一个后端开始；

(3)Weight Round Robin：即加权轮询法。与轮询方法相同，但是给每个后端数据库分配了一个权重值。这个值决定了这个后端相对于其他后端接受负载的比例。例如，一个后端的权重值为2，那么其负载的请求数是权重为1的后端的两倍。

然而，上述负载均衡方法均没有考虑到不同计算机节点的差异性。随机选择法总是随机选择数据库节点，完全不了解后端状态也完全没有可控性。而轮询法和加权轮询法在各个后端数据库之间按顺序循环执行，这虽然可以使每个后端都有任务，但并不能做到各个后端任务的最优分配。

因此，需要提供一种数据库集群系统中进行动态负载均衡的方法和系统，通过对数据库后端节点的运行状态进行测量，实时获得后端节点的响应效率评价值，当集群控制器收到用户请求时，选择效率评价值最高的后端节点来执行，以避免后端节点的过热和故障。

2 动态负载均衡设计

针对上面负载均衡方法的不足，提出了基于节点状态的动态负载均衡方法可对存储节点的即时状态进行动态测量和计算，与传统的节点状态无关的均衡方法相比，动态均衡方法可以更有效地均衡负载、防止热点和单点故障。

在如图1所示的数据库集群系统中，用户提出请求后通过集群控制器对数据库后端节点进行操作。集群控制器包含负载均衡决策组件、后端检测组件、用户请求分发组件和负载信息表，其中，负载均衡决策组件用于执行负载均衡决策过程；后端检测组件用于执行后端负载检测过程；负载信息表用于保存后端负载检测过程所得出的测量值；用户请求分发组件用于暂存用户请求、发起后端负载检测过程、发起负载均衡决策过程、并将用户请求发至决策结果所包含的数据库后端节点执行。

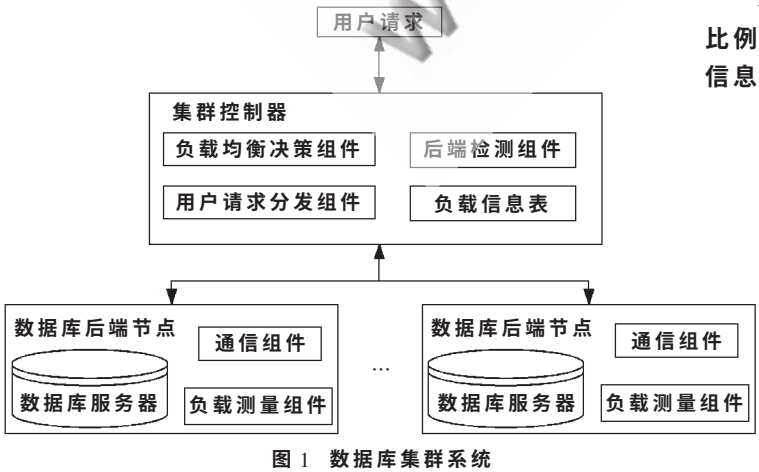

图1 数据库集群系统

数据库后端节点包含通信组件、数据库服务器和负载测量组件，其中，通信组件用于接收后端检测请求、发起负载测量、将负载测量值存入集群控制中的负载信息表中，并接受集群控制器发来的用户请求，将其发给数据库服务器执行，将得到的结果集返回至集群控制器；数据库服务器用于接收通信组件发来的用户请求并作出应答；负载测量组件用于接收通信组件发来的负载测量请求并作出应答。负载测量组件又进一步包含：CPU使用率检测模块、内存使用率检测模块、磁盘已占用空间比例检测模块、空闲磁盘大小检测模块、磁盘IO延迟检测模块和网络延迟检测模块。

动态负载均衡算法如下：

Step1：Measure the load balancing information

Create load balance table

Create load balance trigger

Step2：Compute the load balancing information

Normalize the load data

Compute the entropy of the load data

Compute the weight of the load data based on the entropy

Compute the sum of the weight to quantify the load

Step3：Select the node to execute

3 动态负载均衡实现

3.1 负载信息测量

负载信息测量主要解决节点负载评价问题，目的是为了更确切有效地对节点负载情况进行评价，以准确发现节点是否超载，同时有助于更有效地寻找空闲节点。

负载测量在动态负载均衡中起着重要的作用，只有负载信息测量准确，才能为进一步的负载决策提供条件，才可以确定节点是否超载，是否需要转移新任务。

本文中以CPU使用率、内存使用率、空闲磁盘占用比例、空闲磁盘大小、网络延迟、磁盘IO延迟作为负载信息的测量对象。

3.1.1 CPU使用率测量CU

利用Windows API函数可得知CPU空闲时间和系统时间，从而得知CPU空闲使用率，进而得知CPU使用率。CPU空闲使用率CurrentCpuIdle=SysPerfInfo.liIdleTime/SysTimeInfo.liKeSystemTime；CPU使用率CurrentCpuUsage%=100-(CurrentCpuIdle×100)。

3.1.2 内存使用率测量MU

利用Windows API函数可直接得知内存使用率，GlobalMemoryStatus(&stat)；memory_usage%=stat.dwMemoryLoad。

3.1.3 空闲磁盘占用比例测量 DFE

利用 Windows API 函数直接可得知空闲磁盘和整个磁盘的大小，它们的比值即为空闲磁盘占用比例。

```
fResult=pGetDiskFreeSpaceEx(pszDrive,
    (PULARGE_INTEGER)&i64FreeBytesToCaller,
    (PULARGE_INTEGER)&i64TotalBytes,
    (PULARGE_INTEGER)&i64FreeBytes);
```

空闲磁盘占用比例 disk_usage%=i64FreeBytes×100/i64TotalBytes。

3.1.4 空闲磁盘大小测量 DF

根据上面的 pGetDiskFreeSpaceEx 函数可直接得出空闲磁盘大小 diskfree=i64FreeBytes/(1024×1024)。

3.1.5 磁盘 IO 延迟测量 IO

磁盘 IO 延迟的原理是通过读取磁盘上的文件时间来得出磁盘 IO 延迟。

```
OutSpeed=(sysafter.wMinute-sysbefore.wMinute)×60000+
(sysafter.wSecond-sysbefore.wSecond)×1000+
(sysafter.wMilliseconds-sysbefore.wMilliseconds);
```

3.1.6 网络延迟测量 ND

网络延迟的原理是对各个后端节点执行一条 SQL 语句，SQL 语句运行的时间即为网络延迟。

```
long before=System.currentTimeMillis();
PreparedStatement pm =conn.prepareStatement ("DELETE
FROM loadstate");
pm.executeUpdate();
long now=System.currentTimeMillis();
long timegap=now-before;
```

图 2 所示为在集群管理器中负载均衡测量截图。

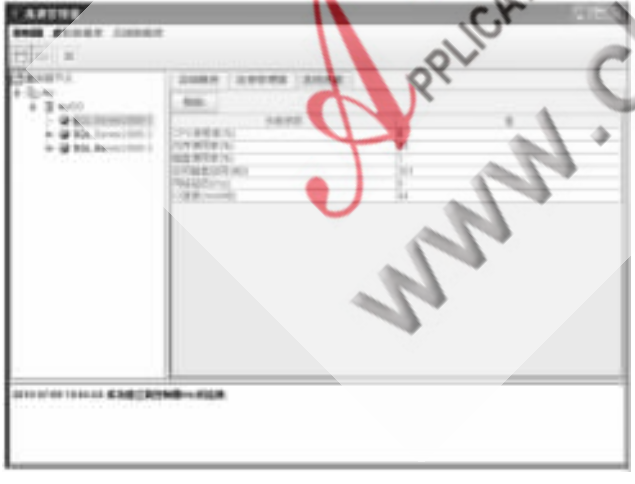

图 2 负载均衡测量截图

3.2 负载决策

节点的负载决策是负载均衡的一个重要因素，因为只有系统准确及时地对节点负载进行评价记录，节点才可以及时准确地确定本身是否超载，是否需要转移新进任务，同时忙节点也可以有效地寻找空闲节点以转移自身的超载任务，相反，如果节点负载决策不准确，则节点无法及时准确定位自己的状态，以至于忙节点在超载的状态下仍接受新的任务。

一般地，某个属性的属性值变异程度越大，信息熵越小，该属性提供的信息量越大，即该属性在方案排序中所起的作用越大，从而该属性的权重也应该越大；反之，某个属性的属性值变异程度越小，信息熵越大，该属性提供的信息量越小，即该属性在方案排序中所起的作用越小，从而该属性的权重也应该越小。

根据参考文献[4]给出一种基于信息熵的模糊多属性决策方法。最常见的属性类型有效益型和成本型。为了消除不同物理量纲对决策结果的影响，采用下列计算公式将模糊决策矩阵 $A=[a_{ij}]_{m\times n}$ 转化为规范化矩阵 $\boldsymbol{R}=[r_{ij}]_{m\times n}$。

成效益型指标计算公式：

$$r_{ij}=\frac{a_{ij}}{p_j},i\in M,j\in N \tag{1}$$

成本型指标计算公式：

$$r_{ij}=\frac{p_j'-a_{ij}}{p_j},i\in M,j\in N \tag{2}$$

其中 $p_j=\max\{a_{ij},i\in M\}$，$p_j'=\max\{p_j-a_{ij},i\in M\}$。

步骤 1：根据式(1)和式(2)将模糊决策矩阵 $\boldsymbol{A}=[a_{ij}]_{m\times n}$ 转化为规范化矩阵 $\boldsymbol{R}=[r_{ij}]_{m\times n}$。

步骤 2：计算属性 u_j 输出的信息熵

$$E_j=-\frac{1}{\ln m}\sum_{i=1}^{m}r_{ij}\ln r_{ij},j\in N \tag{3}$$

当 $r_{ij}=0$ 时，规定 $r_{ij}\ln r_{ij}=0$；

步骤 3：计算属性权重向量 $\omega=(\omega_1,\omega_2,\cdots,\omega_n)$，其中

$$\omega_j=\frac{1-E_j}{\sum_{k=1}^{n}(1-E_k)},j\in N \tag{4}$$

步骤 4：利用 $z_i=\sum_{j=1}^{n}r_{ij}\omega_j(i\in M)$ (5)

计算节点 p 上的模糊效用值 z_i；

步骤 5：根据 $z_i(i\in M)$ 对节点进行排序和择优，z_i 值最大的为最优节点。

下面通过一组测得的负载信息数据来说明整个负载决策的流程。

从 3 个节点采集到的负载决策矩阵如表 1 所示。

表 1 负载决策矩阵

	CU	MU	DFE	DF	IO	ND
SQL Server2005-1	1	52	46	19 101	9	13
SQL Server2005-2	4	45	3	1 551	5	6
SQL Server2005-3	2	26	9	4 054	17	2

步骤 1：根据式(1)、式(2)规范化决策矩阵，得到规范化矩阵 $\boldsymbol{R}$ 如表 2 所示。

步骤 2：根据式(3)计算各个指标的熵值，如表 3 所示。

表 2 规范化矩阵

	CU	MU	DFE	DF	IO	ND
SQL Server2005-1	1	0	1	1	0.667	0
SQL Server2005-2	0	0.269	0.065	0.081	1	0.636
SQL Server2005-3	0.667	1	0.196	0.212	0	1

表 3 指标的熵值

CU	MU	DFE	DF	IO	ND
0.246	0.322	0.452	0.485	0.246	0.262

步骤 3：根据式(4)计算各个指标的权重向量，如表4 所示。

表 4 指标的权重向量

CU	MU	DFE	DF	IO	ND
0.189	0.170	0.137	0.129	0.189	0.185

步骤 4：根据式(5)计算带权重的归一化矩阵，如表5 所示。

表 5 带权重的归一化矩阵

	模糊效用值
SQL Server2005-1	0.581
SQL Server2005-2	0.372
SQL Server2005-3	0.535

步骤 5：根据表格 5 中模糊效用值对节点进行排序和择优，z_i 最大的为最优节点，即第一个节点为最优节点，这时将把任务分配给第一个节点。

4 性能测试与分析

为了验证该负载均衡算法的有效性和优越性[5-6]，通过实验进行了验证。集群由 4 台电脑组成，它们由局域网连接而成，网络带宽 100 Mb/s，4 台电脑配置相同：CPU 是 Intel(R) Xeon(R) E5405 2.00 GHz，3G 内存，Windows XP SP3 操作系统。在其中的一台电脑上配置集群控制器，另外 3 台电脑上装有 SQL Server2005 数据库。

实验对随机法、轮询法、加权轮询法和动态负载均衡算法的效率进行对比，分别让集群中几个节点的状态不同，首先在 3 个节点轻载的情况下对 4 种负载均衡方法进行对比；然后使其中一个节点超载对 4 种负载均衡进行对比；使其中两个节点超载再进行对比。

根据实验数据得出以上三种实验的对比图如图 3、图 4、图 5 所示。

根据实验可以得出：在节点都很空闲(轻载)的情况下，动态负载均衡不一定是最优的，反而有时因为决策运算会花费时间导致其查询的时间会更长。在节点轻载和超载不一的时候，动态负载均衡体现出它的优越性，是几种负载策略中效率最高的；加权轮询法次之；随机法和轮询法效率较低。

针对分布式数据库集群关键技术的负载均衡进行研究，提出了一种动态负载均衡策略，对于缩短事务平均响应时间和提高整个系统资源利用率起了很好的作

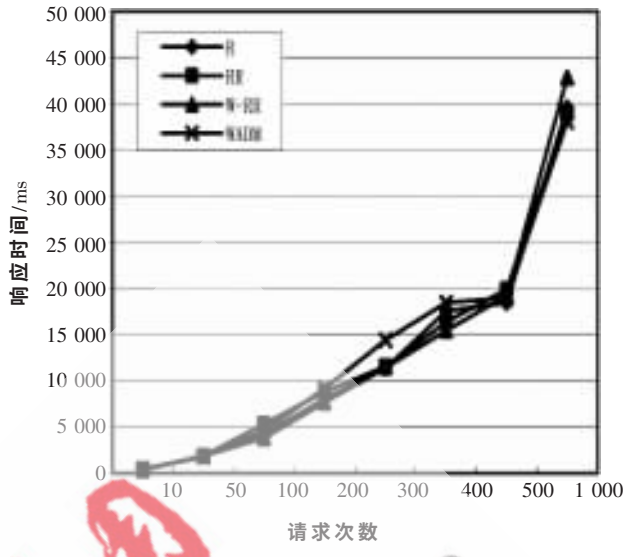

图 3 3 个节点轻载的情况下不同负载均衡算法比较

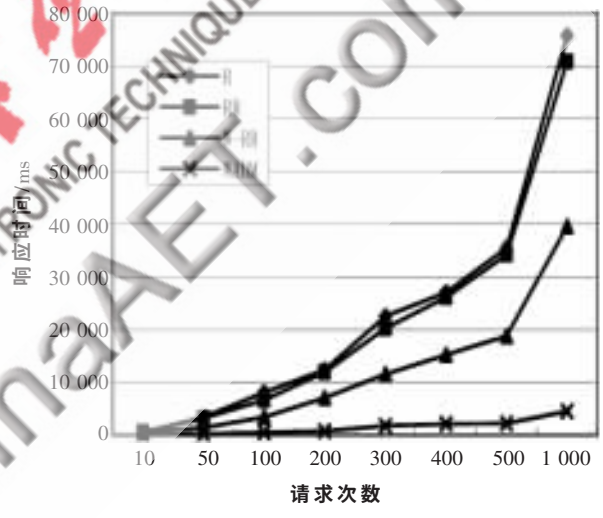

图 4 一个节点超载的情况下不同负载均衡算法比较

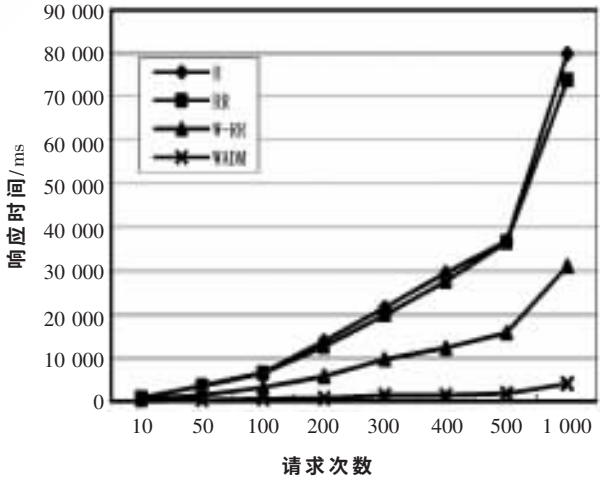

图 5 两个节点超载的情况下不同负载均衡算法比较

用，并通过实验和其他负载均衡策略进行对比，证明了它的有效性和高效性。在今后的工作中，将进一步完善本文提出的负载均衡这一数据库集群关键技术，进一步提升分布式集群数据库系统的可靠性、稳定性以及高效性。

参考文献

[1] YANG X J, DOU Y, HU Q F. Progress and challenges in high performance computer technology[J]. J Comput Sci & Technol, 2006, 21(5): 674-681.

[2] 蒋江，张民选，廖湘科. 基于多种资源的负载均衡算法的研究[J]. 电子学报，2002，30(8)：1148-1152.

[3] ZHENG G B. Achieving high performance on extremely large parallel machines: Performance Prediction and Load Balancing[D]. Urbana: UIUC, 2005.

[4] 徐泽水. 不确定多属性决策方法及应用[M]. 北京：清华大学出版社，2004.

[5] 陈勇. 一种高效的分布式反馈流量负载均衡算法[J]. 计算机工程，2009，35(2)：98-102.

[6] 谷凤娜，张志斌，王丽宏. 基于分布式入侵检测系统的负载均衡算法的比较[J]. 计算机科学，2008，35(11)：63-73.

（收稿日期：2010-07-28）

作者简介：

何骏，男，1986年生，硕士研究生，主要研究方向：地理信息系统与数据库技术。

熊伟，男，1976年生，博士研究生，副教授，主要研究方向：地理信息系统与数据库技术。

陈荦，男，1973年生，博士研究生，副教授，主要研究方向：地理信息系统与数据库技术。

APPLICATION OF ELECTRONIC TECHNIQUE
www.chinaAET.com