(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 107196865 A
(43)申请公布日 2017.09.22

(21)申请号 201710426966.5

(22)申请日 2017.06.08

(71)申请人 中国民航大学
地址 300300 天津市东丽区津北公路2898号

(72)发明人 李国 申亚坤 丁建立 李永华 王怀超 王帅卿

(74)专利代理机构 天津市鼎和专利商标代理有限公司 12101
代理人 蒙建军

(51)Int.Cl.
H04L 12/801(2013.01)
H04L 12/803(2013.01)
H04L 12/853(2013.01)
H04L 12/26(2006.01)
H04L 29/08(2006.01)

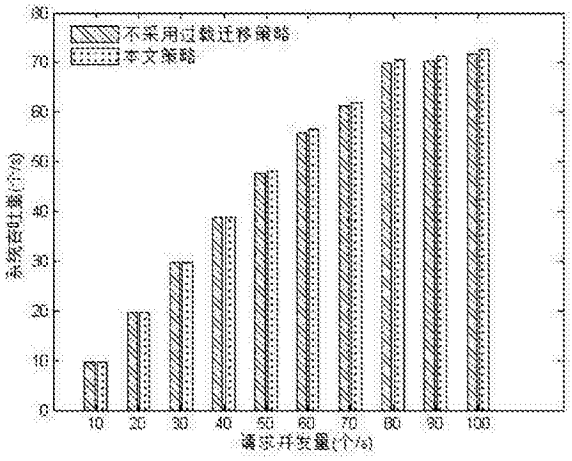
权利要求书2页 说明书7页 附图2页

(54)发明名称

一种负载感知的自适应阈值过载迁移方法

(57)摘要

本发明公开了一种负载感知的自适应阈值过载迁移方法，包括：step1初始化变量：算法开始时需要维护服务信息表，系统自动在处理机上登记进程的PCB信息；step2运行负载均衡算法：在集群系统中配置相应的负载均衡算法对用户的请求进行负载分发。step3确定是否存在需要强制迁移的处理机；step4检查服务器是否过载：定期的监测系统的运行情况，以过载阈值为基准，判定服务器是否过度负载；tep5基于负载感知的迁移服务选取：该专利根据客户端请求的速率自适应的改变过载阈值，能够有效的根据并发量实时调整各个服务器的过载阈值，并对于过载服务器选择最佳迁出方案，防止了过载服务器的超负荷工作甚至宕机。



CN 107196865 A

1.一种负载感知的自适应阈值过载迁移方法，其特征在于：包括如下步骤：

步骤101、初始化变量，开始时首先维护服务信息表，系统在处理机上登记进程的PCB信息，当服务器轻载时，D为本地处理机，S为空，此时系统不存在迁出服务；集群中的所有服务器向负载均衡控制模块发送服务状态信息，负载均衡控制模块协调汇总后生成负载状态表信息；具体步骤为：

服务登记信息表是对服务器中正在进行的服务进行记录的一种数据结构，每个服务器节点维护一个服务登记信息表；上述数据结构描述为一个向量α_i(P,D,S,T,PCB)；其中i表示集群中的第i个服务器，i∈[1,n]；P是进程标识符，描述了系统中每一个进程的ID；D是待迁移进程所属的源主机ID；S是待迁移进程所要迁往的目标主机ID；T是迁移过程中的迁移类型，如负载过重迁移、宕机迁移；PCB记录该服务CPU现场信息、堆栈信息、以及进程资源清单等相关信息，用于在目标处理机对迁移服务进行恢复；

系统负载状态表是由当前集群系统中所有服务器共同维护的，主要用于描述系统中各台服务器的忙碌程度；系统状态表用一个向量β(N,L,C)来表示；其中N表示为处理机的ID；

N∈[S_1,S_2,S_3,...,S_n]

L是当前时刻t服务器i的负载值，L详细描述了当前服务器中所有节点的闲忙程度以及可利用状态；

$$L_i(t)=L_{init}+\sum_{j=1}^{m}\partial_j.C_{ij}.T_{ij}(t)$$

其中∂_j是第j种服务类型对计算机总开销的贡献值；C_{ij}表示第i台服务器节点第j种服务类型所占的开销；$T_{ij}(t)$是在t时刻，服务器节点i接受的j服务类型的数目；

C代表当前处理机的状态，其中L_o表示当前系统负载均值，当L_i大于L_o时，当前服务器为重载，标记为W；当L_i小于等于L_o时，表示为轻载服务器，标记为E；当服务器不可用时，标记为D。

$$L_O=\sum_{p=1}^{m}\sqrt[f]{\prod_{i=1}^{m}L_i^{f_p}}$$

其中f是跟服务器性能相关的权值，该权值采用加权集合平均数计算得到；

步骤102、运行负载均衡算法：

根据不同的业务场景部署相应的负载均衡算法：具体为：

加权轮询算法，适用于服务器性能相差不大的集群，任务队列的每个成员分配任务的概率相同；

随机算法，其中用户请求随机分发给后台的各个服务器，其中，随机函数的选取直接影响算法的好坏；

比率算法，依据各个服务器的负载能力分配，权值决定请求的分配概率，综合考虑了服务器性能的差异性；

最少连接算法，依据服务器连接数分配用户请求，忽略了请求消耗资源的不同，适用于请求类型单一的集群；

预测模式，基于应用程序的行为对处理器分配工作负载，从中选择一台服务器分配用户请求；

步骤103、确定是否存在需要强制迁移的处理机：

在系统运行过程中，当服务器自身的物理故障或者是人为原因，导致该服务器不能够正常提供服务，此时必须把用户所有的请求强制迁移出去；保证所有的请求在最短时间内迁出；

步骤104、检查服务器是否过载：

在系统运行过程中，定期的监测系统的运行情况，以服务器设定的过载阈值为基准，判定服务器是否过度负载；该阈值随服务器收到的用户并发量的变化而变化；当服务器在某一段时间接收过多的请求时，适当的调高负载阈值，当服务器较空闲时，则适当降低负载阈值；如果服务器确超过过载阈值，则进行下一步；

过载阈值的设定，如果过载阈值设定偏小，则会很容易触发过载迁移策略，导致服务器计算资源的浪费；相反，如果过载阈值设定过大，则很难触发过载迁移或者刚进行过载迁移服务器节点就变得难以恢复导致迁移代价巨大，则会导致某个节点很容易达到用户宕机的红线，所以服务器过载阈值的设定应该有区别性；具体的阈值调整方法为：

$$W_{new}=\begin{cases}k_1W_{old} & N<N_{\min}\\ k_2W_{old} & N>N_{\max}\end{cases}$$

其中W_{old}是根据系统性能不同而设定的过载阈值，W_{new}为新生成的阈值，N为监测到的用户请求分发到服务器的并发量，N_{min}，N_{max}为预先设定的过载区间；当$N<N_{min}$时，增大过载阈值，使更多的服务请求能够被接收；反之当$N>N_{max}$时需要减小过载阈值，把超过负载能力的任务迁移出去，防止服务器过于沉重或者宕机；通常k_1设置为1.2，k_2设置为0.8；

步骤105、基于负载感知的迁移服务选取：

当服务器过载时，对服务器上的服务进行迁移，究竟选取哪些服务进行迁移而保证系统负载率最小且迁移代价最小是该研究的关键；

服务选取方法是一种启发式方法，迭代的选择满足要求的服务进行迁移，具体步骤如下：

A、获取服务器上所有服务的集合V，置最小需要迁移的服务集合$V_{min}=V$；

B、对集合中的各个服务按照负载值大小进行排序，并且令索引$i=1$，置V_m为空，$j=1$；

C、若$|i+j|>|V|$，则已遍历出当前算法，算法终止；否则选择第$i+j$个服务为迁移服务，并把该服务加入到V_m中，若V在移出该服务后，服务器处于负载阈值之下转入步骤D，否则执行$j=j+1$，循环步骤C；

D、若V_m中的负载值小于V_{min}负载值，则更新V_{min}，$V_{min}=V_m$，并且$i=i+1$，返回步骤B。

一种负载感知的自适应阈值过载迁移方法

技术领域

[0001] 本发明应用于服务负载迁移领域，特别是涉及一种负载感知的自适应阈值过载迁移方法。

背景技术

[0002] 系统负载迁移是实际中经常遇到的一种问题，负载迁移策略的选择直接影响系统的效率，好的迁移策略能够增加系统的吞吐量，降低用户的相应时间，从而提高系统整体的吞吐量。

[0003] 通常常见的过载迁移策略有以下几种：基于目标处理机的选择策略、基于过载阈值的设定策略、过载迁移整体架构策略。这几种过载迁移策略各有优缺点，能够适应不同的应用场景。

[0004] 基于目标处理机的选择策略简单且比较常用的一种策略，其中比较常见的是以下几种：比如通过分析任务的特征并计算任务事件次数的数学期望和方差作为可靠性评价参数来选择目标迁移节点。将服务可迁移的目标定义为不破坏节点间的偏序关系且不产生死锁的情况下，使得迁移后的服务执行期望最大化，服务时间最小化。基于Xen虚拟机内存迭代拷贝算法，提出了通过缩短迭代拷贝的终止时间来减少虚拟机动态迁移所花费的时间，以使任务迁移时间最小化。采用首次适应算法或最佳适应算法找出第一个符合要求的节点作为目的节点，提出一种将目录迁移与目录复制相结合的负载均衡策略。这几种方法侧重点在于解决目标节点的选择问题，对阈值的选择以及对迁移节点的待迁移进程的研究稍有欠缺。

[0005] 基于过载阈值的设定策略是基于过载迁移最基本问题即过载阈值提出来的。以节点的下载量和被下载量为基础提出了推拉结合的结构化网络“热点”动态迁移策略。将过载阈值选择问题建模为马尔可夫决策过程并根据最小迁移时间原则以及最小能耗增加放置原则确定虚拟机的迁移策略。以文件为粒度在迁入端重建迁出端的相关状态结构，并根据被访问状态选择是否立即响应。对负载迁移进行建模，在迁移过程中减少网络访问次数、减少全局时间消耗以及在提高效率的同时兼顾全局的负载均衡。利用缓存和链路迁移策略，将重载节点中的剩余负载向其他轻载节点转移。在以上方法中，对过载阈值进行了建模，根据各自研究背景的不同，选取了不同的阈值设定策略。

[0006] 最后一种研究比较多的是过载迁移实现框架。有的是借助蚁群算法的思想，提出了一种面向负载均衡的自主式虚拟机动态迁移框架。通过专门的迁移协议的实现，以及专门的软件模块的实现和内核修改完成了进程迁移。基于最小化网络通信录的负载迁移策略，它主要关注负载迁移整个系统架构以及各模块所发挥的作用。综上所述，以上所有研究只是关注于负载迁移的某个环节，并没有把各个环节连贯起来，且对于待迁移服务的选择研究较少，故在前述研究的基础上提出了一种负载感知的自适应阈值过载迁移方法。

发明内容

[0007] 本发明要解决的技术问题是:本发明的目的是提供一种负载感知的自适应阈值过载迁移方法,该负载感知的自适应阈值过载迁移方法能够根据客户端请求的速率自适应的改变过载阈值,并且在待迁移服务器中待迁移服务的选择上提出了一种负载感知方法,能够有效的根据并发量实时调整各个服务器的过载阈值,并对于过载服务器选择最佳迁出方案,防止了过载服务器的超负荷工作甚至宕机。

[0008] 本发明为解决公知技术中存在的技术问题所采取的技术方案是:

[0009] 一种面向多类型服务的粒子群优化用户请求调度方法,包括以下步骤:

[0010] 步骤101、初始化变量,开始时首先维护服务信息表,系统在处理机上登记进程的PCB信息,当服务器轻载时,D为本地处理机,S为空,此时系统不存在迁出服务;集群中的所有服务器向负载均衡控制模块发送服务状态信息,负载均衡控制模块协调汇总后生成负载状态表信息;具体步骤为:

[0011] 服务登记信息表是对服务器中正在进行的服务进行记录的一种数据结构,每个服务器节点维护一个服务登记信息表;上述数据结构描述为一个向量$\alpha_i(P,D,S,T,PCB)$;其中i表示集群中的第i个服务器,$i\in[1,n]$;P是进程标识符,描述了系统中每一个进程的ID;D是待迁移进程所属的源主机ID;S是待迁移进程所要迁往的目标主机ID;T是迁移过程中的迁移类型,如负载过重迁移、宕机迁移;PCB记录该服务CPU现场信息、堆栈信息、以及进程资源清单等相关信息,用于在目标处理机对迁移服务进行恢复;

[0012] 系统负载状态表是由当前集群系统中所有服务器共同维护的,主要用于描述系统中各台服务器的忙碌程度;系统状态表用一个向量$\beta(N,L,C)$来表示;其中N表示为处理机的ID;

[0013] $N\in[S_1,S_2,S_3,\ldots,S_n]$

[0014] L是当前时刻t服务器i的负载值,L详细描述了当前服务器中所有节点的闲忙程度以及可利用状态;

[0015] $$L_i(t)=L_{init}+\sum_{j=1}^{m}\partial_j C_{ij} T_{ij}(t)$$

[0016] 其中∂_j是第j种服务类型对计算机总开销的贡献值;C_{ij}表示第i台服务器节点第j种服务类型所占的开销;$T_{ij}(t)$是在t时刻,服务器节点i接受的j服务类型的数目;

[0017] C代表当前处理机的状态,其中L_o表示当前系统负载均值,当L_i大于L_o时,当前服务器为重载,标记为W;当L_i小于等于L_o时,表示为轻载服务器,标记为E;当服务器不可用时,标记为D。

[0018] $$L_O=\sum_{p=1}^{m}\sqrt[f]{\prod_{i=1}^{m}L_i^{f_p}}$$

[0019] 其中f是跟服务器性能相关的权值,该权值采用加权集合平均数计算得到;

[0020] 步骤102、运行负载均衡算法:

[0021] 根据不同的业务场景部署相应的负载均衡算法:具体为:

[0022] 加权轮询算法,适用于服务器性能相差不大的集群,任务队列的每个成员分配任务的概率相同;

[0023] 随机算法,其中用户请求随机分发给后台的各个服务器,其中,随机函数的选取直

接影响算法的好坏；

[0024] 比率算法，依据各个服务器的负载能力分配，权值决定请求的分配概率，综合考虑了服务器性能的差异性；

[0025] 最少连接算法，依据服务器连接数分配用户请求，忽略了请求消耗资源的不同，适用于请求类型单一的集群；

[0026] 预测模式，基于应用程序的行为对处理器分配工作负载，从中选择一台服务器分配用户请求；

[0027] 步骤103、确定是否存在需要强制迁移的处理机：

[0028] 在系统运行过程中，当服务器自身的物理故障或者是人为原因，导致该服务器不能够正常提供服务，此时必须把用户所有的请求强制迁移出去；保证所有的请求在最短时间内迁出；

[0029] 步骤104、检查服务器是否过载：

[0030] 在系统运行过程中，定期的监测系统的运行情况，以服务器设定的过载阈值为基准，判定服务器是否过度负载；该阈值随服务器收到的用户并发量的变化而变化；当服务器在某一段时间接收过多的请求时，适当的调高负载阈值，当服务器较空闲时，则适当降低负载阈值；如果服务器确超过过载阈值，则进行下一步；

[0031] 过载阈值的设定，如果过载阈值设定偏小，则会很容易触发过载迁移策略，导致服务器计算资源的浪费；相反，如果过载阈值设定过大，则很难触发过载迁移或者刚进行过载迁移服务器节点就变得难以恢复导致迁移代价巨大，则会导致某个节点很容易达到用户宕机的红线，所以服务器过载阈值的设定应该有区别性；具体的阈值调整方法为：

[0032] $$W_{new}=\begin{cases}k_1W_{old} & N<N_{\min}\\ k_2W_{old} & N>N_{\max}\end{cases}$$

[0033] 其中W_{old}是根据系统性能不同而设定的过载阈值，W_{new}为新生成的阈值，N为监测到的用户请求分发到服务器的并发量，N_{min}，N_{max}为预先设定的过载区间；当$N<N_{min}$时，增大过载阈值，使更多的服务请求能够被接收；反之当$N>N_{max}$时需要减小过载阈值，把超过负载能力的任务迁移出去，防止服务器过于沉重或者宕机；通常k_1设置为1.2，k_2设置为0.8；

[0034] 步骤105、基于负载感知的迁移服务选取：

[0035] 当服务器过载时，对服务器上的服务进行迁移，究竟选取哪些服务进行迁移而保证系统负载率最小且迁移代价最小是该研究的关键；

[0036] 服务选取方法是一种启发式方法，迭代的选择满足要求的服务进行迁移，具体步骤如下：

[0037] A、获取服务器上所有服务的集合V，置最小需要迁移的服务集合$V_{min}=V$；

[0038] B、对集合中的各个服务按照负载值大小进行排序，并且令索引$i=1$，置V_m为空，$j=1$；

[0039] C、若$|i+j|>|V|$，则已遍历出当前算法，算法终止；否则选择第i+j个服务为迁移服务，并把该服务加入到V_m中，若V在移出该服务后，服务器处于负载阈值之下转入

[0040] 步骤D，否则执行$j=j+1$，循环步骤C；

[0041] D、若V_m中的负载值小于V_{min}负载值，则更新V_{min}，$V_{min}=V_m$，并且$i=i+1$，返回步骤B。

[0042] 本发明具有的优点和积极效果：

[0043] 通过采用上述技术方案，该负载感知的自适应阈值过载迁移方法能够根据客户端请求的速率自适应的改变过载阈值，并且在待迁移服务器中待迁移服务的选择上提出了一种负载感知方法，能够有效的根据并发量实时调整各个服务器的过载阈值，并对于过载服务器选择最佳迁出方案，防止了过载服务器的超负荷工作甚至宕机。

附图说明：

[0044] 图1是采用本发明技术方案与传统方案所得到的第一吐量图对比表；

[0045] 图2是采用本发明技术方案与传统方案所得到的负载率对比；

[0046] 图3是采用本发明技术方案与传统方案所得到的第二吞吐量对比表；

具体实施方式

[0047] 为能进一步了解本发明的发明内容、特点及功效，兹例举以下实施例，并配合附图详细说明如下：

[0048] 请参阅图1至图3，一种负载感知的自适应阈值过载迁移方法，

[0049] 步骤101、初始化变量，算法开始时需要维护服务信息表，系统自动在处理机上登记进程的PCB信息，当服务器轻载时，D为本地处理机，S为空，此时系统不存在迁出服务。集群中的所有服务器向负载均衡控制模块发送服务状态信息，后者协调汇总后生成负载状态表信息。具体步骤为：

[0050] 服务登记信息表是对服务器中正在进行的服务进行记录的一种数据结构，每个服务器节点维护一个服务登记信息表。该结构可以描述为一个向量α_i(P,D,S,T,PCB)。其中i表示集群中的第i个服务器，i∈[1,n]；P是进程标识符，描述了系统中每一个进程的ID；D是待迁移进程所属的源主机ID；S是待迁移进程所要迁往的目标主机ID；T是迁移过程中的迁移类型，如负载过重迁移、宕机迁移；PCB记录该服务CPU现场信息、堆栈信息、以及进程资源清单等相关信息，用于在目标处理机对迁移服务进行恢复。

[0051] 系统负载状态表是由当前集群系统中所有服务器共同维护的，它主要用于描述系统中各台服务器的忙碌程度。系统状态表用一个向量β(N,L,C)来表示。其中N表示为处理机的ID。

[0052] $N\in[S_1,S_2,S_3,\ldots,S_n]$

[0053] L是当前时刻t服务器i的负载值，它详细描述了当前服务器中所有节点的闲忙程度以及可利用状态。

[0054] $$L_i(t)=L_{init}+\sum_{j=1}^{m}\partial_j.C_{ij}.T_{ij}(t)$$

[0055] 其中∂_j是第j种服务类型对计算机总开销的贡献值。C_{ij}表示第i台服务器节点第j种服务类型所占的开销。$T_{ij}(t)$是在t时刻，服务器节点i接受的j服务类型的数目。

[0056] C代表当前处理机的状态，其中L_o表示当前系统负载均值，当L_i大于L_o是，当前服务器为重载，标记为W；当L_i小于等于L_o时，表示为轻载服务器，标记为E；当服务器不可用时，标记为D。

[0057] $$L_O=\sum_{p=1}^{m}\sqrt[f]{\prod_{i=1}^{m}L_i^{f_p}}$$

[0058] 其中f是跟服务器性能相关的权值，此处采用加权集合平均数的计算，既能够避免某些服务器负载率的突然升高，又能适合于动态平均数的计算。

[0059] 步骤102、运行负载均衡算法：

[0060] 根据不同的业务场景部署相应的负载均衡算法：如加权轮询算法，适用于服务器性能相差不大的集群，任务队列的每个成员分配任务的概率相同；随机算法，其中用户请求随机分发给后台的各个服务器，其中，随机函数的选取直接影响算法的好坏；比率算法，依据各个服务器的负载能力分配，权值决定请求的分配概率，综合考虑了服务器性能的差异性；最少连接算法，依据服务器连接数分配用户请求，忽略了请求消耗资源的不同，适用于请求类型单一的集群；预测模式，基于应用程序的行为对处理器分配工作负载，从中选择一台服务器分配用户请求。

[0061] 步骤103、确定是否存在需要强制迁移的处理机：

[0062] 在系统运行过程中，服务器自身的物理故障或者是人为原因，导致该服务器不能够正常提供服务，那么此时必须用最短的时间把用户所有的请求强制迁移出去。务必保证所有的请求在最短时间内迁出。

[0063] 步骤104、检查服务器是否过载：

[0064] 在系统运行过程中，要定期的监测系统的运行情况，以服务器设定的过载阈值为基准，判定服务器是否过度负载。该阈值随服务器收到的用户并发量的变化而变化。当服务器在某一段时间接收过多的请求时，应该适当的调高负载阈值，当服务器较空闲时，则适当降低负载阈值。如果服务器确超过过载阈值，则进行下一步。

[0065] 过载阈值的设定是非常重要的，如果过载阈值设定偏小，则会很容易触发过载迁移策略，导致服务器计算资源的浪费；相反，如果过载阈值设定过大，则很难触发过载迁移或者说刚进行过载迁移服务器节点就变得难以恢复导致迁移代价巨大，则会导致某个节点很容易达到用户宕机的红线，所以服务器过载阈值的设定应该有区别性。本文采用了一种自适应阈值处理方法，能够根据用户请求速率自适应的调整阈值的大小。具体的阈值调整方法为：

[0066] $$W_{new}=\begin{cases}k_1W_{old} & N<N_{\min}\\ k_2W_{old} & N>N_{\max}\end{cases}$$

[0067] 其中W_{old}是根据系统性能不同而设定的过载阈值，W_{new}为新生成的阈值，N为监测到的用户请求分发到服务器的并发量，N_{min}，N_{max}为预先设定的过载区间。当$N<N_{min}$时可以适当增大过载阈值，使更多的服务请求能够被接收；反之当$N>N_{max}$时需要减小过载阈值，把超过负载能力的任务迁移出去，防止服务器过于沉重或者宕机。通常k_1设置为1.2，k_2设置为0.8。

[0068] 步骤105、基于负载感知的迁移服务选取：

[0069] 当服务器过载时，此时需要对服务器上的服务进行迁移，究竟选取哪些服务进行迁移而保证系统负载率最小且迁移代价最小是该研究的关键。

[0070] 该服务选取方法是一种启发式方法，迭代的选择满足要求的服务进行迁移，具体步骤如下：

[0071] ①获取服务器上所有服务的集合V，置最小需要迁移的服务集合$V_{min}=V$。

[0072] ②对集合中的各个服务按照负载值大小进行排序，并且令索引i＝1，置V_m为空，j＝1。

[0073] ③若|i+j|>|V|，则已遍历出当前算法，算法终止；否则选择第i+j个服务为迁移服务，并把该服务加入到V_m中，若V在移出该服务后，服务器处于负载阈值之下转入第四步，否则执行j＝j+1，循环第三步。

[0074] 若V_m中的负载值小于V_{min}负载值，则更新V_{min}，V_{min}＝V_m，并且i＝i+1，返回第二步。

[0075] 在上述优选实施例中，具体包括以下步骤：

[0076] Step1初始化变量

[0077] 算法开始需要维护进程信息表，系统自动在处理机上等级进程的PCB信息，当服务器不过载时，D为本地处理机，S为空，此时系统不存在迁出进程。集群中的所有服务器向负载均衡控制模块发送进程状态信息，后者协调汇总生成负载状态表信息。

[0078] Step2运行负载均衡算法

[0079] 在集群系统中配置相应的负载均衡算法对用户的请求进行负载。

[0080] Step3确定是否存在需要强制迁移的处理机

[0081] 在系统运行过程中，服务器自身的物理故障或者是人为原因，导致该服务器不能够正常接收请求，那么此时必须用最短的时间把用户所有的请求强制迁移出去。

[0082] 如果遇到服务器某些硬件故障导致服务器不能正常服务，那么必须尽快迁出用户的所有请求，此时选择负载状态表中负载最小的服务器进行迁移。务必保证所有的请求在最短时间内迁出。

[0083] Step4检查服务器是否过载

[0084] 在系统运行过程中，要定期的监测系统的运行情况，监测的依据就是服务器设定的阈值，该阈值是随服务器的性能变化而变化的。当服务器在某一段时间接收过多的请求时，应该适当的调高负载阈值，当服务器较空闲时，则适当降低负载阈值。如果服务器确定需要迁移，则触发下一步。

[0085] Step5基于负载感知的迁移服务选取

[0086] 当服务器过载时，此时需要对服务器上的服务进行迁移，究竟选取哪几个服务进行迁移是该研究的关键。本文提出一种基于负载感知的服务迁移策略。

[0087] 第一步：获取服务器上所有服务的集合V，置最小需要迁移的服务集合V_{min}＝V。

[0088] 第二步：然后对集合中的各个服务按照负载值大小进行排序，并且令索引i＝1，置V_m为空，j＝1。

[0089] 第三步：若|i+j|>|V|，则已遍历出当前算法，算法终止；否则选择第i+j个服务为迁移服务，并把该服务加入到V_m中，若V在移出该服务后，服务器处于负载阈值之下转入第四步，否则执行j＝j+1，循环第三步。

[0090] 第四步：若V_m中的负载值小于V_{min}负载值，则更新V_{min}，V_{min}＝V_m，并且i＝i+1，返回第二步。

[0091] 第五步：源服务器通过执行系统调用将待迁移服务迁移至目标处理机。后者在收到迁入服务之后，为其分配必要的资源，恢复期PCB信息，并插入到就绪队列。

[0092] 在该实验中，除了搭建ILink模拟环境，还需要编写两套程序，分别为客户端脚本程序和服务端服务程序。客户端脚本程序由HttpClient技术实现，通过HttpClient设定相

应的参数及请求类型向服务器发送请求，客户端脚本可以自由设定并发量，进而模拟现实中的用户请求。服务端程序则是提供几种系统中常见的不同资源消耗的模拟服务，如CPU消耗型服务、内存消耗型服务、磁盘消耗型服务等。通过在服务端的负载均衡模块配置该负载均衡策略并采集系统吞吐量和负载率等指标来验证该方法的好坏。该实验场景是按照生产环境的用户请求规律层层递进而设计的，共设计了四个实验场景：

[0093] 第一个场景：客户端脚本向服务器以[10,20,30,40,50,60,70,80,90,100](单位为(个/秒))并发量向服务器发送请求，测试服务器的连通性及负载能力。

[0094] 第二个场景：首先在第一个周期内客户端按照[10,100]的并发区间随机访问后台服务器，在第二个周期内维持在高并发区间[60,100]访问，在第三个周期内模拟用户访问减少的情况，不断降低并维持在[10,30]的并发量。第四个区间是对后台服务器进行一个随机的访问，最后综合统计后台各节点的吞吐量情况，经过大量的实验，实验稳定结果如图1。

[0095] 第三个场景：在随机并发或者低并发条件下，系统的负载率差异性不大，所以，该场景设计了持续高并发访问[60,100]考验系统的负载率指标(图2)。

[0096] 第四个场景：在该场景下模拟了机器硬盘物理故障的情况，首先在其中一台服务器内部编写一个脚本，不断创建新文件以占用磁盘空间，最终服务器空间被塞满从而服务器不能接受用户的服务，测试该条件下系统吞吐量指标(图3)。

[0097] 如图1所示在并发量60(单位(个/s))之前，由于系统完全有能力接受该并发范围的服务请求量，所以很少甚至没有触发过载迁移策略，两者的系统吞吐量几乎相同，但是，当服务请求维持在较高阶段[70,80,90,100]时，系统中单核主机逐渐不能够满足该请求规模，遇到了瓶颈，此时触发过载迁移策略，相应的，采用了本文策略的吞吐量相对于没有过载迁移的策略平均提高了约1.2%。

[0098] 如图2所示，负载率指标衡量了整个系统的忙碌程度，在并发量50(单位(个/s))之前，两种策略负载率都是稳步提升，这是因为在客户端采用了逐步递增并发量的原因导致的。在此，过载阈值从60改变到了50左右，这是因为过载阈值随并发量的改变发生了变化。当系统达到过载阈值以后，采用过载迁移策略的负载率提高较慢，这是因为在该过程中触发了过载迁移，对负载任务进行了分摊使各个服务器均处于忙碌状态，提高了系统的负载率约2.3%，这也是过载感知策略优化的目的。

[0099] 如图3所示，模拟了服务器硬件故障的情况，当用户请求并发量较少且系统硬件没有故障时，两种策略几乎没有区别，系统吞吐量相同；但是一旦超过了55(单位(个/s))以后，系统中损坏了一台服务器，所以导致系统整体的服务能力与之前相比会有所下降，但是不采用过载迁移策略的吞吐量程线性下降，而采用了过载迁移策略的吞吐量则是少量下降，这也说明了客户的请求有相当一部分被迁移成功，且当请求并发量持续增高以后，本文策略是占有一定优势的。

[0100] 以上对本发明的实施例进行了详细说明，但所述内容仅为本发明的较佳实施例，不能被认为用于限定本发明的实施范围。凡依本发明申请范围所作的均等变化与改进等，均应仍归属于本发明的专利涵盖范围之内。

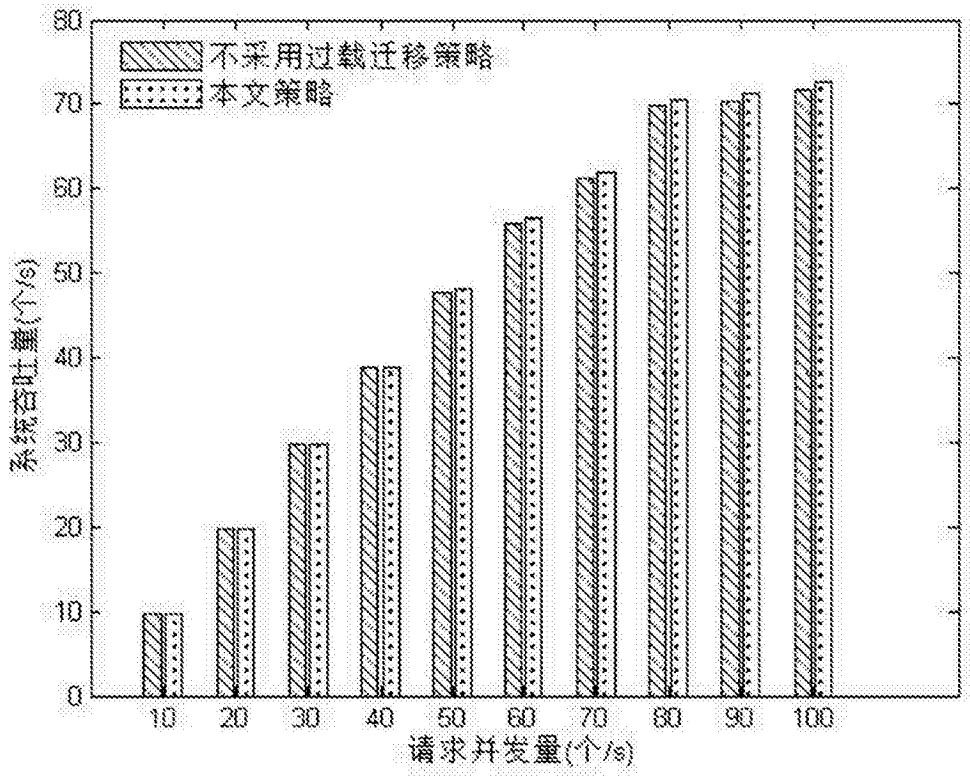

不采用过载迁移策略
本文策略
系统吞吐量(个/s)
请求并发量(个/s)
0
10
20
30
40
50
60
70
80
90
100

图1

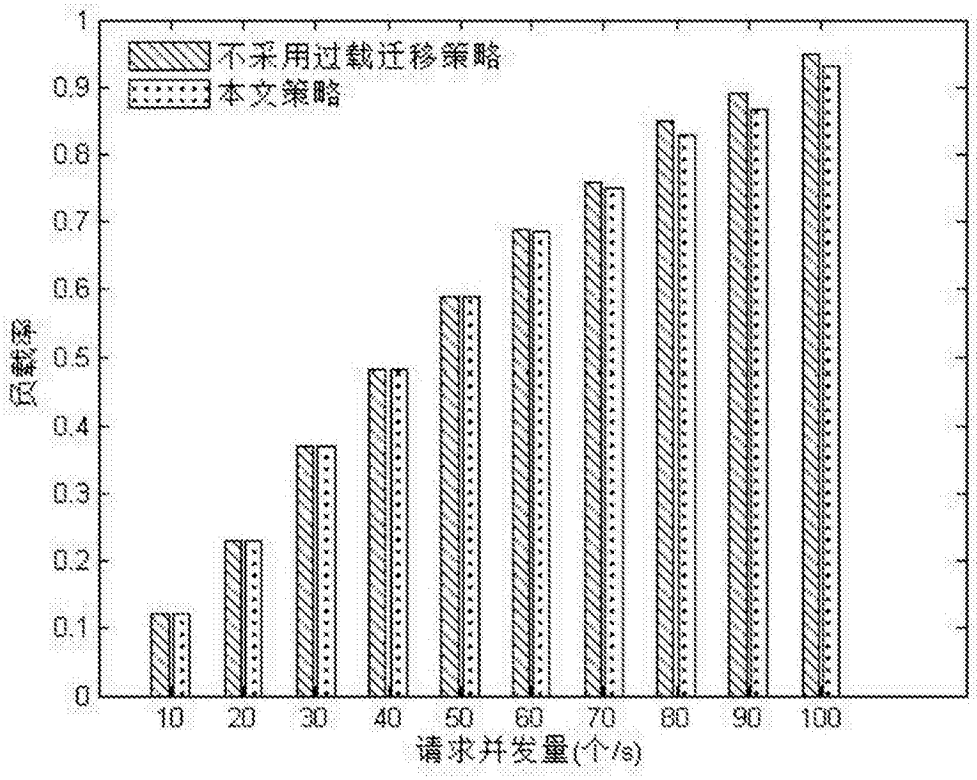

不采用过载迁移策略
本文策略
负载率
请求并发量(个/s)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
10
20
30
40
50
60
70
80
90
100

图2

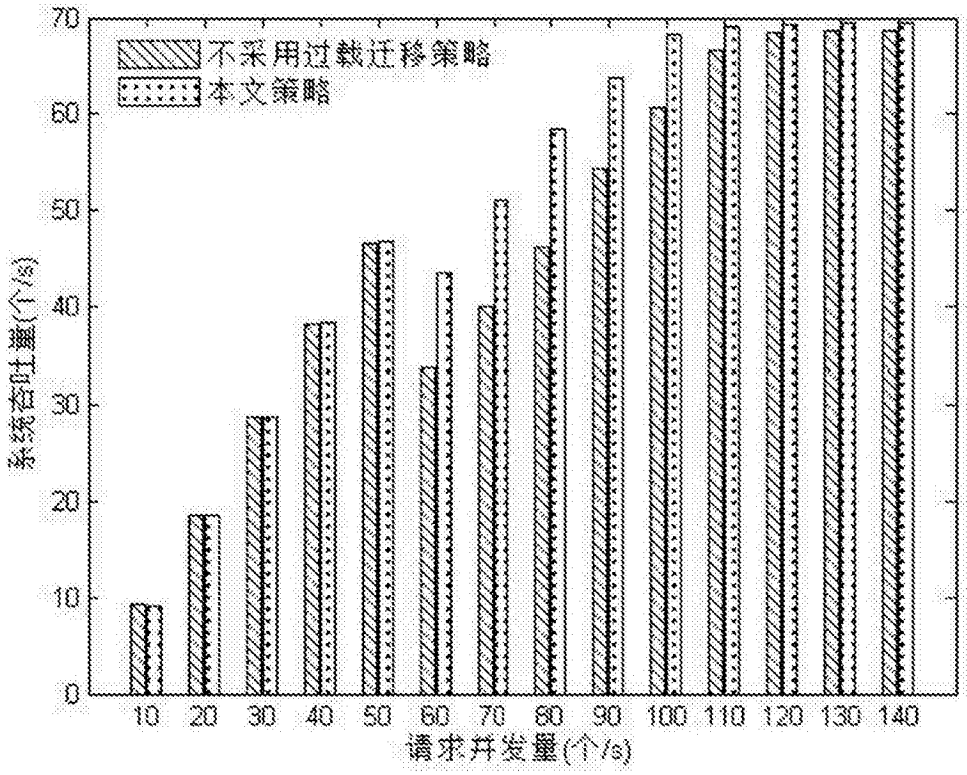

不采用过载迁移策略
本文策略
系统吞吐量(个/s)
请求并发量(个/s)
0
10
20
30
40
50
60
70
10
20
30
40
50
60
70
80
90
100
110
120
130
140

图3