基于对等模式的虚拟计算环境下的一种任务调度算法①

张 千, 石 琎, 梁 鸿, 关新全

(中国石油大学(华东) 计算机与通信工程学院, 青岛 266580)

摘 要: 基于对等模式的虚拟计算系统体系结构,提出一种基于加权随机和反馈机制的任务调度算法. 选定的调度节点首先对资源进行按需筛选和静态量化排序, 继而对权重递减的资源进行随机选择, 然后获取相应动态信息进行过载过滤及排序得到最终资源, 最后通过反馈机制实现算法对系统负载的自适应性. 实验表明该算法有效避免了系统瓶颈, 平衡了负载, 具有较好的自适应性.

关键词: 对等模式; 加权随机; 反馈机制; 任务调度; 负载平衡

A Scheduling Algorithm Based on Peer-to-Peer Mode in Virtual Computing

ZHANG Qian, SHI Jin, LIANG Hong, GUAN Xin-Quan

(College of Computer and Communication Engineering, China University of Petroleum(East China), Qingdao 266580, China)

Abstract: Scheduling algorithm was always a hot topic in virtual computing environment.In order to elimanate system bottleneck and balance load dynamicly. A distributed scheduling algorithm based on weighted random and feedback mechanisms was proposed in this paper. At first the chosen distributed scheduling host chose resources by needs and made static quantification,and then sorted them; Secondly the algorithm chose resources from which sorted by weight randomly, And then it acquired corresponding dynamic information to make load filterate and sort the left. At last it achieved the self-adaptivity to system load through feedback mechanisms. The experiment shows that the algorithm has avoided the system bottleneck effectively and has achieved balanced load as well as self-adaptability to it.

Key words: peer-to-peer mode; weighted random;feedback mechanisms; task scheduling; load-balance

虚拟计算环境组织方式[1]主要分为集中式和对等式, 集中式计算环境实现简单, 且调度节点在了解全局资源运行状态下, 实施任务调度, 任务分配合理, 但当系统规模较大时, 过于集中的任务调度处理使得调度节点性能和稳定性成为系统瓶颈, 因此, 集中式调度适用于规模较小的计算环境; 对等式虚拟计算环境解决了集中调度模式中瓶颈问题, 支持计算节点动态加入和退出, 适合在规模较大的计算环境中采用, 但对等调度算法设计实施较难, 在无法知悉全局资源状态下, 仅根据局部信息对资源进行调度很可能达不到调度优化目的, 尽管蚁群和粒子群等对等调度算法调度执行过程趋于全局最优, 但算法实施困难, 调度过程复杂, 计算和网络传输代价较大. 实际计算环境中通常采用相对简单的调度算法, 如 OLB、MET、Min_Min、Max_Min 算法[2]. 由于 OLB 算法是随机把一个可用机器分配给一个待执行任务, 因此机器负载均衡效果较好, 但是因为没有考虑资源释放时间而导致其资源利用率较低. Min_Min 算法是将全部未分配的机器分配给在其上最早执行且具有最短资源释放时间的任务, 因此资源利用率较高, 但机器负载均衡效果不理想. MET 算法的目标是把每一个机器分配给具有最短资源释放时间的任务, 牺牲机器负载不均衡来换取较好的资源利用率. Max_Min 算法的映射策略较大地改变给定资源的可用时间状态, 因此该算法导致较低的资源利用率和不均衡的机器负载.

因此, 本文提出一种基于加权随机[3]和反馈机制的任务调度算法. 在不显著增加算法实施复杂性的前提下, 算法通过采用加权随机、过载评估和反馈策略,

① 基金项目:中央高校基本科研业务费专项资金

收稿时间:2013-02-01;收到修改稿时间:2013-03-25

© 中国科学院软件研究所 http://www.c-s-a.org.cn

保证了性能优异的节点不会超负荷运载, 同时性能一般的计算节点有执行任务的可能, 提高了系统的负载均衡性和资源利用率.

1 算法描述

1.1 任务资源需求描述

本文描述的算法建立在任务程序已分析, 任务已分解的基础上, 每个任务中的子任务具有元子性, 即无优先级约束、不可跨节点运行、不可再分. 各子任务可以对执行任务的资源提出最低配置需求. 资源属性集合包括处理器能力、内存存储能力、硬盘存储能力、网络通信能力, 对应的任务配置要求定义如下:

$$R=\{R_{proc},R_{mem},R_{disk},R_{comm},R_{as}\} \tag{1}$$

任务业务处理逻辑不同, 会影响任务对计算资源某些属性有特殊要求, 例如, 计算密集型任务可能对CPU 性能和内存容量需求更高, 而存储密集型任务可能对硬盘或网络带宽要求更高, 因此, 任务对不同资源属性需求权重定义如下:

$$w=\{w_{proc},w_{mem},w_{disk},w_{comm}\} \tag{2}$$

1.2 资源属性描述

1.2.1 资源属性定义

本文称能够实施任务执行功能的计算单元为资源. 根据功能不同, 资源可分成计算资源, 存储资源和网络资源等. 首先, 对节点集合进行如下定义:

$$NS=\{N_1,N_2,\cdots,N_n\} \tag{3}$$

节点 $N_i(i=1,2,\cdots,n)$ 内影响服务能力的属性集合包括, 处理器能力、内存存储能力、硬盘存储能力、网络通信能力、可信度和队列负载, 对应的资源属性定义如下:

$$RS=\{RS_{proc},RS_{mem},RS_{disk},RS_{comm},RS_{rely},RS_{quene}\} \tag{4}$$

资源的各属性可以通过具体量值衡量, 调度算法使用属性量值作为资源性能评价依据.

RS_{proc} 包含处理器个数, 主频和运行负荷三个主要属性参数值, 定义如下:

$$RS_{proc}=\{p,k,p(t)\} \tag{5}$$

式(5)中, p 表示处理器的主频, k 表示内核数, $p(t)$ 表示在 t 时刻, 处理器负载. 如果 $k>1$, 那么 $p(t)=\{p_1(t),p_2(t),\cdots,p_k(t)\}$, 是一个集合, 并且是时间相关属性参量.

本文中, RS_{mem}、RS_{disk} 和 RS_{comm} 都包含两个参量, 分别是内存大小和负载、磁盘大小和负载、网络带宽和负载, 定义如下:

$$RS_{mem}=\{m,m(t)\},RS_{disk}=\{d,d(t)\},RS_{vomm}=\{c,c(t)\} \tag{6}$$

式(6)中, 内存负载、磁盘负载和网络负载是时间相关参量.

可信度 RS_{rely} 通过资源近期执行的若干任务成功完成概率 $r(t)$ 衡量, $r(t)$ 为执行成功的任务数目与近期执行的总任务数的比率, 定义如下:

$$r(t)=success_task(t)/recent_total_task \tag{7}$$

队列负载 RS_{quene} 通过资源任务处理队列当前已接受任务数与可容纳任务数的比 $q(t)$ 衡量, $q(t)$ 定义如下:

$$q(t)=queue_used(t)/queue_size \tag{8}$$

1.2.2 资源属性值规范化

对多目标属性决策问题中, 属性量化方式直接影响到客观评估结果. 本文在对资源评估过程中使用多个指标, 各指标存在着不同的单位和量纲, 为了更好的反映资源实际性能状态, 各指标需要在统一的度量标准下进行比较, 因此, 对各计算节点资源综合评估前, 需要将相关的指标去量纲、规范化. 本文采用向量标准化[4]方法中的几何平均数法对资源属性进行规范化, 计算简单, 并且受极值影响较小. 计算方法如下:

$$Z_{ij}=\frac{X_{ij}}{\sqrt{\sum(X_{ij})^2}} \tag{9}$$

2 算法实施

本文提出的任务调度算法基于对等模式的虚拟计算环境实现, 每个计算节点都具有完备的任务接收、资源评估、任务调度和执行功能. 调度过程中, 综合考虑全局资源运行状态后实施调度. 任务调度过程描述如图 1. 如图 1 所示, 任务调度执行流程描述如下:

1) 选择调度主机. 用户根据一定策略选择计算环境中某一台主机提交任务, 图中用户选择 N_i 提交任务, N_i 主机负责评估系统内计算资源运行状况, 调度任务到执行主机, 监控任务状态变化并反馈任务执行信息;

2) 资源静态信息筛选. 调度主机根据本地存储的静态资源信息, 使用任务资源配置需求, 淘汰不满足任务执行最低配置的资源, 使用向量标准化方法综合

© 中国科学院软件研究所 http://www.c-s-a.org.cn

评估资源的多个静态指标, 根据指标值的优劣将资源排序, 使用概率算法选取若干后备计算资源;

3) 资源动态信息获取. 通过网络获取后备计算资源实时运行信息;

4) 过载判断. 根据实时运行信息, 排除某项指标高于过载标准的资源. 将剩余计算资源使用向量标准化方法综合评估, 选取动态指标值最高的计算资源执行任务. 如果所有后备资源都被过载判断排除, 回到步骤 2, 重新对资源静态信息筛选, 但本次筛选过程中排除前一次后备计算资源, 并调节概率算法, 让更多计算资源有机会被选中. 调度结束后, 根据后备主机数目与过载判断后剩余主机数目的比值调节概率算法;

5) 调度主机将任务投递到执行节点, 监控任务执行状态, 反馈执行结果;

6) 反馈调节. 根据后备资源在过载判断过程中淘汰的比重, 可以粗略估计计算系统负载情况, 调节随机过程中使用的参数, 修改任务投递目标.

根据调度算法流程的概要描述, 可将算法实施分为以下五个步骤: ①资源筛选与静态属性量化;②后备资源获取; ③过滤过载资源; ④动态属性量化排序; ⑤负载信息反馈调节.

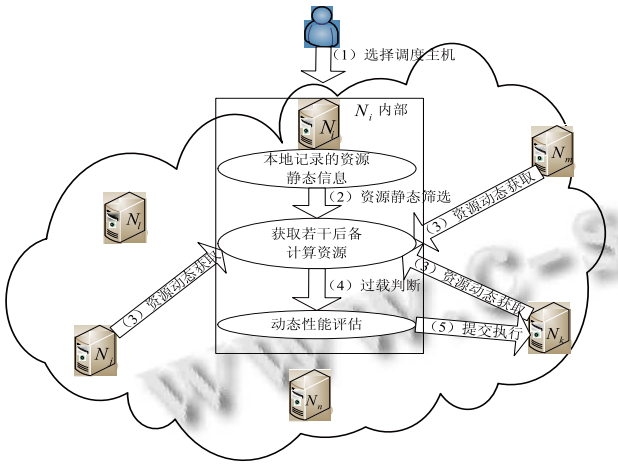

图 1 任务调度流程图

2.1 资源筛选与静态属性量化

2.1.1 按需筛选资源

调度单元接到任务请求后, 算法根据式(1)对资源进行基于静态属性的筛选, 选出满足任务最低配置需求的 u 个可用资源节点, 表述为 $N_i\left(i=\{1,2,\cdots,u\}\right)$.

2.1.2 可用资源静态属性量化

资源静态属性是资源属性指标中在系统运行期间不会发生改变的部分, 根据定义(5)、(6)单个计算节点静态属性指标定义如下:

$$S=\{p,k,m,d,c\} \tag{10}$$

其中 p 是处理器主频、k 是内核数目、m 是内存容量、d 是外存容量、c 是带宽.

由于不同资源属性量纲有差异, 因此使用式(9)将资源静态属性去量纲、标准化, 如下所示:

$$\tilde{p}_i=\frac{p_i\times k_i}{\sqrt{\sum_{j=1}^{u}(p_j\times k_j)^2}}\quad \tilde{m}_i=\frac{m_i}{\sqrt{\sum_{j=1}^{u}m_j^2}}\quad \tilde{d}_i=\frac{d_i}{\sqrt{\sum_{j=1}^{u}d_j^2}}\quad \tilde{c}_i=\frac{c_i}{\sqrt{\sum_{j=1}^{u}c_j^2}} \tag{11}$$

其中, $i=\{1,2,\cdots,u\}$表示各可用计算资源. 式(11)各式分别表示单个计算节点处理器、内存、硬盘、网络四个属性的标准化结果. 根据定义(2), 任务对资源的不同属性指标需求有差异, 因此, 静态资源评估标准定义如下:

$$\eta_i=w_{proc}\times\tilde{p}_i+w_{mem}\times\tilde{m}_i+w_{disk}\times\tilde{d}_i+w_{comm}\times\tilde{c}_i \quad i=\{1,2,\cdots,u\} \tag{12}$$

通过上式即可获取可用计算节点 $N_i\left(i=\{1,2,\cdots,u\}\right)$的评估值.

2.2 后备资源获取

上一步中按需筛选出了 u 个可用资源, 如果可用资源较多, 还需从中筛选出 l 个性能相对优异的后备资源. 首先使用冒泡排序算法以式(12)求出的 η 值将资源按降序排列, 在排序过程中产生 v 个已序节点后, 排序算法结束. 此时, 排列中 l 到 v 的节点是降序排列的, 而 v+1 到 u 节点是无序的, 其中 $u>v>l$ 且:

$$\forall i\in[v+1,u],\ \text{满足}\ \eta_i<\eta_v \tag{13}$$

通过排序算法, 截去性能较差的计算节点, 之后需要在剩余性能较优的 v 个资源中使用随机算法抽取 l 个后备计算单元. 本文称 v 为性能截取窗口宽度, l 为后备资源窗口宽度.

一般认为, 经过性能截取窗口 v 截取的计算资源性能都比较优异, 将任务提交到其中的任意主机上执行, 时间效能比较相近, 因此, 使用加权随机算法从 v 个资源节点抽取 l 个后备资源, 可以满足任务在性能较优异的节点上执行, 同时也避免了所有调度节点都将任务提交到最优异的计算节点执行, 导致的负载不均衡.

加权随机采用等比加权法, 等比加权系数为 γ, 因此两个相邻有序资源的概率满足:

© 中国科学院软件研究所 http://www.c-s-a.org.cn

$$a_{i+1}=\gamma a_i,\ \text{且}\ i=\{1,2,\cdots,v-1\},\quad 0<\gamma<1 \tag{14}$$

a_i 表示静态性能排列在第 i 位的资源被选中的概率. 由于 γ 在 0 和 1 之间, 因此, 静态性能越优的资源被选中的概率越高, 概率以 γ 倍速等比下降. 一次选择中所有 v 个计算节点被选中的概率和应当等于 1, 根据等比数列求和公式, 可以求得性能最优的资源被选中的概率为:

$$a_1=(1-\gamma)/(1-\gamma^v) \tag{15}$$

结合使用式(14)和式(15)即可求出排列中第 k 位的计算资源被选中的概率 a_k.

使用均匀分布随机数生成器在(0, 1)间生成随机数 z, 使用下面方法确定排列中被选中的主机:

$$\sum_{1}^{k}a_k\le z\le\sum_{1}^{k+1}a_{k+1},\ \text{其中}\ k=\{1,2,\cdots,v-1\} \tag{16}$$

上式表明, 从排列列首开始累加概率, 直到累加值超过随机数 z 值, 则排列中的第 k 个计算单元被选中.

结合式(14)和(16)可以求得值为:

$$k=\left\lfloor\frac{\ln\left(1-\frac{z\times(1-\gamma)}{a_1}\right)}{\ln\gamma}\right\rfloor \tag{17}$$

因此, 使用加权随机算法抽取一个后备资源的过程如下:

1) 根据已知的比例系数 γ 和性能截取宽度窗口 v, 利用式(15)求出排列中最优资源被抽取的概率 a_1;

2) 使用均匀分布随机数生成器在 0 到 1 之间生成一个随机数 z;

3) 按照式(17)计算 k 值, 并将排列中第 k 位的计算主机删除, 添加到后备主机列表.

上述算法, 在排序资源列表中使用概率抽取一个后备计算资源, 当选中的计算主机 k 被删除后, 排列主机样本空间的数目将变成 v-1, 此时, 从性能截断窗口截除的可用计算资源中(即: v+1 到 u)抽取 η 值最高的计算主机加入到排列末尾, 使用上述算法第 2、3 两步可用获取第二个随机抽取的后备计算节点, 重复上述过程, 直至 l 个后备计算资源选取完毕.

等比加权只是加权随机中的一种较为简单的加权方式, 适合选择一组个体性能差别比较明显的资源, 若一组资源内某些资源性能相近, 则可以先对资源进行按性能分组(模糊聚类[5]), 组间采用等比加权方式确定选择概率, 组内性能相近的资源平分组级概率, 后文的实验就是采用这种方式进行的, 加权随机出备选组后, 组内进行等概率随机得到最终资源.

2.3 过滤过载资源

通过按需筛选、静态量化和加权随机筛选后, 选出的个后备计算资源是静态属性较好的计算资源. 调度主机需要评估后备计算资源的实时运行信息, 选择当前运行状况最好的计算主机提交执行任务. 通过信息服务组件, 调度服务主机获取后备资源的动态运行信息, 根据式(5)、(6)、(7)、(8)的定义, 各计算资源动态信息表述如下:

$$D=\{p(t),m(t),d(t),c(t),r(t),q(t)\} \tag{18}$$

动态信息分别为, 处理器负载、内存负载、硬盘负载、网络负载、可信度和任务队列负载, 根据定义, 所有负载都满足在(0, 1)区间之内.

负载超重标准定义如下:

$$L=\{L_p,L_M,L_d,L_c,L_r,L_q\} \tag{19}$$

负载超重标准指标分别表示为, 处理器、内存、硬盘、网络、可信度和任务队列负载上限, 本文认为, 如果计算资源的某项属性指标超过相应指标的上限, 则任务资源已经超负荷运行, 不适合任务的继续提交. 很明显, 超重指标区间范围在 0 和 1 之间.

在过载评估前, 需要预处理动态资源部分数据, 例如计算主机可能存在多个内核, 因此, $p(t)$ 就是一个集合, 使用平均数方法取处理器内核负载平均值, 如下所示:

$$\overline{p}(t)=\frac{\sum_{i=1}^{k}p_k(t)}{k} \tag{20}$$

使用式(18)中的 $m(t)$、$d(t)$、$c(t)$、$r(t)$、$q(t)$ 和式(20)中的 $\overline{p}(t)$ 结合式(19)中的过载判断标准, 如果资源的某项指标超过过载判断标准, 那么称这样的主机为过载主机, 不适合继续提交执行任务.

2.4 动态属性量化排序

过载评估将资源运行负载较重的计算资源排除, 剩余若干个主机是系统中性能较好, 且负载相对较轻的计算节点. 算法最后使用动态评估从中选出最优计算节点提交任务.

在动态评估前, 需要对计算节点动态信息标准化处理, 由于, 每个计算主机处理器可能存在多个内核,

© 中国科学院软件研究所 http://www.c-s-a.org.cn

对处理器指标进行如下处理:

$$p'_i(t)=\sum_{i=1}^{k_i}\left[\left(1-p_i(t)\right)\times p_i\right] \tag{21}$$

其中 k_i 表示主机 N_i 的内核数.

结合式(9)、(11)和(21)求出单个计算资源动态属性指标标准化值, 表示如下:

$$\tilde{p}_i(t)=\frac{p'_i(t)}{\sqrt{\sum_{j=1}^{g}(p'_j(t))^2}}\ 、\ \tilde{m}_i(t)=\frac{(1-m_i(t))\times m_i}{\sqrt{\sum_{j=1}^{g}[(1-m_j(t))\times m_j]^2}}$$
$$\tilde{d}_i(t)=\frac{(1-d_i(t))\times d_i}{\sqrt{\sum_{j=1}^{g}[(1-d_j(t))\times d_j]^2}}\ 、\ \tilde{c}_i(t)=\frac{(1-c_i(t))\times c_i}{\sqrt{\sum_{j=1}^{g}[(1-c_j(t))\times c_j]^2}} \tag{22}$$

其中 g 表示 4.3 过载筛选后, 后备计算节点剩余数目.

根据任务对资源渴求程度定义(2), 结合(22), 单个资源动态评估值定义如下:

$$\eta_i(t)=w_{proc}\times\tilde{p}_i(t)+w_{mem}\times\tilde{m}_i(t)+w_{disk}\times\tilde{d}_i(t)+w_{comm}\times\tilde{c}_i(t) \tag{23}$$

从剩余后备计算节点中选取 $\eta_i(t)$ (其中 $i=\{1,2,\cdots,g\}$)值最大的计算资源, 并将任务调度投递到该节点上执行.

2.5 负载信息反馈调节

资源按需筛选和进行静态属性量化后, 使用性能截取窗口 v, 将性能较差的计算节点排除, 之后在 v 个性能较好的资源中, 使用随机加权抽取 l 个后备计算资源, 后备资源抽取过程中, 比例系数 γ 直接影响到资源抽取的概率分布. 当 γ 趋近于 1 时, v 个排序的后备计算资源被抽到的概率相近, 当 λ 远离 1 时, 排在前面较优异的计算节点被抽中的概率将升高. 例如, 当 v 取 4, λ 取 0.5 时, 可得 4 个排序的计算资源各自被抽中的概率为:

$$a_1=\frac{8}{15}、a_2=\frac{4}{15}、a_3=\frac{2}{15}、a_4=\frac{1}{15}$$

上例中, 静态属性最优异的资源被选中的概率大于 0.5.

后备计算节点选择结束后, 算法评估资源动态属性, 使用过载判断排除运行繁忙的计算节点, 并最终剩余 g 个可继续投递任务的资源. 最终较空闲资源数 g 与系统后备计算资源窗口宽度 l 的比值, 从一定程度上能够反映系统繁忙程度. 在此基础上定义系统负载如下:

$$e=g/l \tag{24}$$

当整个系统运行比较空闲时, 抽取的 l 个性能优异的计算节点会全部通过过载评估, 成为最终可选的投递主机, 此时 e=1, 说明整个虚拟计算环境运行较空闲, 应当将计算任务尽量提交到性能特别优异的资源上执行, 因此, 可以缩减性能截取窗口宽度 v 值, 降低比例系数 γ 值, 调度系统将会更多的考虑将任务投递到整个系统中性能最优异的几个计算节点上运行; 当 0.5<e<1 时, 说明计算系统运行较为空闲, 因此, 可以不对 v 和 γ 值修改; 当 $e\le 0.5$ 时, 表明计算系统中性能较好的计算资源多数比较繁忙, 应当扩宽截取窗口宽度 v 值, 提升比例系数 γ 值, 因此, 调度系统将充分考虑将任务向性能一般的计算节点上提交, 从而减轻了优异计算主机的计算压力, 也达到了为任务选择较好主机执行的目的.

系统运行初期可以根据实际情况设定较小的性能截取窗口宽度 $v_{\min}$ 和比例系数 $\gamma_{\min}$ ($\gamma_{\min}$ 不宜选择过小, 应当在 0.5 以上, 否则达不到平衡负载的目的). 调度结束后, 系统使用下列方式调节 v 和 γ 值大小:

当 e=1 时, 说明系统运行空闲, 应当将任务尽量提交到更好的计算资源上执行, 因此, 需要缩小 v 值, 降低 γ 值. 新值获取方式如下:

$$v'=v/2,\ \ \gamma'=(3\gamma-1)/2 \tag{25}$$

若 v'<$v_{\min}$, 则性能截取窗口宽度 v 设置为 $v_{\min}$, 否则, 设置为 v'.

若 $\gamma'<\gamma_{\min}$, 则比例系数 γ 设置为 $\gamma_{\min}$, 否则, 设置为 γ'.

当 0.5<e<1 时, 表明系统运行较空闲, 无需调节窗口宽度和比例系数.

当 $e\le 0.5$ 时, 说明系统运行很繁忙, 需要增加性能截取窗口宽度, 提升 γ 值, 任务有机会提交到性能一般的主机上执行, 缓解性能优异资源计算压力. 新值获取方式如下:

$$v'=2v,\ \ \gamma'=(1+\gamma)/2 \tag{26}$$

性能窗口宽度 v 设置为 v', 比例系数 γ 设置为 γ'.

反馈机制根据计算环境负载情况调节等比加权系数, 系统运行较空闲时, 将任务主要投递到性能优异的计算节点上完成, 当系统性能优异的计算节点运行比较繁忙时, 通过反馈机制将任务分配到其它性能较好的计算资源上完成. 本文使用负载反馈机制提高了系统资源利用率, 动态平衡了系统负载.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

3 实验结果与性能分析

测试环境采用 4 个 Dell 服务器节点、4 个高性能计算单机节点、4 个普通测试节点进行对等式结构搭建, 环境无中心节点, 每个节点都具有完备的功能, 可进行独立调度. 每个 Dell 服务器节点由两个 CPU 组成, 每个 CPU 有四个核心, 使用超线程技术(Hyper-Threading), 每个核心提供两个计算线程, 因此, 每个服务器节点由 2×4×2 个逻辑 CPU 组成.

表 1　虚拟计算环境主机列表

主机名	CPU	Mem	OS
Server1	Xeon E5520 2.26GHz(2×4×2)	8.169GB	Linux RHEL 5.1
Server2	Xeon E5520 2.26GHz(2×4×2)	8.375GB	Linux RHEL 5.1
Server3	Xeon E5520 2.26GHz(2×4×2)	8.375GB	Linux RHEL 5.1
Server4	Xeon E5520 2.26GHz(2×4×2)	5.969GB	Linux RHEL 5.1
HP1	Pentium E5300 2.6GHz(1×2)	1.918GB	Linux RHEL 5.1
HP2	Pentium E5300 2.6GHz(1×2)	2.086GB	Linux RHEL5.1
HP3	Pentium E5300 2.6GHz(1×2)	1.918GB	Linux RHEL5.1
HP4	Pentium E5300 2.6GHz(1×2)	2.086GB	WinXP SP 3
Norm1	Pentium E2140 2.6GHz(1×2)	1.027GB	Linux FC6
Norm2	Pentium4 2.6GHz(1×1)	1.034GB	Linux FC6
Com1	Pentium4 2.6GHz(1×1)	0.545GB	Linux FC6
Com2	Pentium4 2.6GHz(1×1)	0.478GB	Linux FC6

实验一分别采用 MET 算法与本文算法调度 20 个轻量子任务, 其中后者实施的过程中根据节点性能情况将资源分为四组, 对组进行加权随机, 即 v 为 4, 又取 γ 初值为 0.6, 且环境中所有节点均满足任务需求, 最终的资源利用情况如表 2 所示:

表 2　各分组最终接收任务数

	Server(4)	HP(4)	Norm(2)	Com(2)	rate
MET	16	4	0	0	8/12
随机加权	9	6	3	2	12/12
加权理论值	9.2	5.5	3.3	2.0	

由表 1 可以看出加权随机选择的确较之 MET 有着更好的资源利用率,系统负载也更为均衡.

实验二采用本文算法顺序调度 20 个高负载子任务, 任务的执行时间均大于 20 分钟, 我们每隔 5 分钟提交一个任务, 分组数 v 依然为 4, 总资源数为 12, 采用本算法对资源按需筛选和进行静态属性量化后, 使用性能截取窗口 v, v 值为 4, 将性能较差的 4 个计算节点排除, 之后在 8 个性能较好的资源中, 使用随机加权抽取 l 个后备计算资源, 取后备资源窗口大小 l 为 4, 由于环境规模有限任务量也不大, 已被分配任务的资源不从样本空间中删除, 即取 v 值恒定. 等比加权系数 γ 初值为 0.7. 空闲比 e 与等比加权系数 γ 随时间变化情况如表 3:

表 3　γ 反馈机制

时间 t(min)	负载 e	调度前 γ	调度后 γ
5	1	0.7	0.7
10	1	0.5	0.5
15	1	0.5	0.5
20	0.75	0.5	0.5
25	1	0.5	0.5
30	0.75	0.5	0.5
35	0.75	0.5	0.5
40	0.5	0.5	0.5
45	0.25	0.5	0.75
50	0.5	0.75	0.75
55	0.25	0.75	0.875
60	0.5	0.875	0.875
65	0.5	0.875	0.875
70	0.75	0.875	0.875
75	0.5	0.875	0.875
80	0.75	0.875	0.875
85	0.75	0.875	0.875
90	1	0.875	0.813
95	1	0.813	0.719
100	0.75	0.719	0.719

由表 2 可以看出, 45min 的时候系统负载已超过反馈调节的阈值, γ 按式(26)的约束扩大, 通过增大性能稍差但空闲的资源被选中的权值, 反馈调节负载, 由表 6-3 可以看出 60min 以后系统基本达到了负载的动态平衡.

© 中国科学院软件研究所 http://www.c-s-a.org.cn

4 结语

本文基于对等模式的虚拟计算环境提出了一种加权随机调度算法, 将资源属性分成静态和动态两部分分别评估. 算法在保证将任务优先提交到性能最佳的资源基础上, 使用加权随机策略、过载评估策略和反馈策略等, 保证性能优异的节点不会超负荷运载, 同时性能一般的资源有执行任务的可能, 算法有效平衡了负载, 具有较好的自适应性.

参考文献

1 张海宾,唐琳莎,刘立祥.网格调度综述.计算机工程与设计,2009,30(9):2151−2153,2190.

2 王琴,曾文华.网格资源调度算法的负载均衡及性能分析,2006,23(10):201−203.

3 王兴柱,曾庆怀.虚拟网格服务工作流的调度算法研究.计算机工程与设计,2009,10:2371−2374.

4 苏维华.多指标综合评价理论与方法问题研究[学位论文].厦门:厦门大学,2000.

5 张敏,于剑.基于划分的模糊聚类算法.软件学报,2004,15(6):858−869.

6 Eiben AE, Raué PE, Ruttky ZS. Genetic algorithms with multiparent recombination: Proc.of the 3rd Conference on Parallel Problem Solving from Nature: Springer-Verlag, 1994:78−87.

7 Colorni A, Dorigo M, Maniezzo V, et al. Distributed Optimization by Ant Colonies: Europen conference on artificial life paris frence,1991:134−142.

8 Granville V, Kricanek M, Rasson JP. Simulated annealing: A Proof of convergence: IEEE Transactions on Pattern Analysis and Machine Intelligence, Washington, DC, USA:1994:652−656.

(上接第 163 页)

参考文献

1 Jeroen P, Van DB. Analytic Expressions for the Optimal Dwell Point in An Automated Storage/Retrieval System.International Production Economics, 2002,76(1):13−25.

2 Hu YH, Huang S, Chen CY, et al. Travel Time Analysis of a New Automated Storage and Retrieval System. Computers & Operations Research,2005,32(6):1514−1544.

3 张虹娟.基于蚁群算法的自动化立体仓库作业调度研究[学位论文].天津:天津理工大学,2008.

4 周永权,谢竹诚.求解 TSP 的改进人工鱼群算法.系统工程与电子技术,2009,31(6):1458−1461.

5 李晓磊,邵之江,钱积新.一种基于动物自治体的寻优模式:鱼群算法.系统工程理论与实践,2002,22(11):32−38.

6 李晓磊,路飞,田国会.组合优化问题的人工鱼群算法应用.山东大学学报,2004,34(5):64−67.

7 田国会,张攀.基于混合遗传算法的固定货架拣选优化问题研究.机械工程学报,2004,40(2):141−144.

8 田伟,田国会,张攀.应用改进 LK 算法求解固定货架拣选优化问题.计算机应用,2004,24(6):167−170.

© 中国科学院软件研究所 http://www.c-s-a.org.cn