

优化的资源配置算法在网格计算系统中的实现

周建鸿^{1,2,3}, 马懋德²

(1 中科院成都信息技术有限公司(原中国科学院成都计算所), 四川 成都 610041;

2 新加坡南洋理工大学, 电子电气工程学院, 新加坡 639798;

3 四川大学 锦城学院电子信息工程系, 四川 成都 611731)

摘 要: 资源配置算法是影响网格计算系统性能的一个最重要的因素. 本文提出了一个基于资源优化配置算法的网格计算系统, 系统包含网格用户、网格 broker、网格资源和网格信息服务中心四个部分. 其中重点介绍了资源优化配置算法, 包含了用户端的资源配置与资源端的请求排列两部分. 通过这两部分算法的优化, 可以将请求分配到最合适的资源并且按照用户要求被处理. 最后将本算法与两个已有的资源配置算法依据请求完成成功率、完成时间、平均等待时间三个参数进行对比, 体现出本系统的优越性.

关键词: 网格计算系统; 资源配置; 请求排列; 完成时间; 平均等待时间

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2016)11-0020-05

Research and Realization of an Optimized Resource Allocation Algorithm in Grid Computing System

ZHOU Jian-hong^{1,2,3}, MA Mao-de²

(1 Chengdu Information Technology of Chinese Academy of Science Co. Ltd., Chengdu

610041, China; 2 Nanyang Technological University, School of Electrical & Electronics

Engineering 639798, Singapore; 3 Jincheng College of Sichuan University, Department of electronic and information engineering, Chengdu 611731, China)

Abstract: Grid computing has become one of the most popular topics in both research community and commercial companies. Allocation policy is one of the most important factors, which affect the performance of Grid Computing system. A new Grid Computing System based on an optimized resource allocation algorithm was proposed in this paper. It contains four components: Grid User, Grid Broker, Grid Resource and Grid Information Service. The optimized resource allocation algorithm, consisting of user side resource allocation and resource side job scheduling, was emphasized in this paper. With the combination of both sides' efforts, the jobs will be allocated to the most appropriate resources and completed in the most suitable order to meet the requirements of users. At last, the experiments and results comparison with traditional particle swarm optimization and Max-min scheduling are presented, which proves that the proposed algorithm has better performance than the other two algorithms.

Key words: grid computing system; resource allocation; job schedule; completion time; average waiting time

1 引言

近年来, 网格计算作为一种全新的领域, 正在从各个方面逐步改变人们解决复杂计算问题的方式.

随着计算机的普及, 个人电脑开始进入千家万户. 越来越多的电脑处于闲置状态, 即使在开机状态下 CPU 的潜力也远远不能被完全利用. 互联网的出现, 使得连接调用所有拥有闲置计算资源的计算机

系统成为了现实. 网格计算的目的是^[1], 通过任何一台计算机都可以提供无限的计算能力, 可以接入浩如烟海的信息. 这种环境将能够使各企业解决以前难以处理的问题, 最有效地使用他们的系统, 满足客户要求并降低他们计算机资源的拥有和管理总成本. 在过去数年间, 越来越多的网格计算实验室被建立起来, 用于研究如何将网格计算应用到我们生活中的各个领域来解决那些以前受仪器系统限制而难以解决的复杂问题, 比如说: 用于解决比较复杂的数学问题、研究寻找最为安全的密码系统和治疗各种各样疾病的药物研究等等.

在网格计算系统中, 资源配置算法的优劣程度是影响整个系统网络工作效率的一个重要因素. 它决定了在用户提出要求以后, 对资源配置处理的速度快慢, 最终决定了整个系统处理问题的成功率和整个系统完成用户要求所需要的时间. 本文设计了一种优化的资源配置算法, 用于提高网格计算系统中的资源分配效率, 并且利用 GridSim 工具包^[2]实现了算法的仿真, 最后通过与目前常用的资源配置算法仿真进行性能对比, 突出了本文算法的有效性.

2 系统模型

如前所述, 网格计算涉及到要联系不同地方的电脑资源来完成一个共同的目标. 因此, 在网格计算模型中一般包含两大重要因素: 资源使用者和资源提供者. 在本文介绍的系统中, 用户即资源使用者遍及世界各地, 并且大家相互间并没有任何联系; 资源提供者也是遍及世界各地并且他们并不属于某一个公司或者机构. 为了完成用户和资源之间有效的沟通, 本系统设置了两个重要角色: 网格 broker 和网格信息服务中心. 它们之间的关系如图 1 所示.

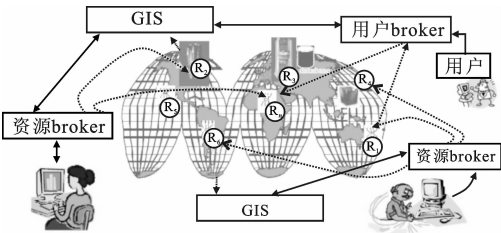


图 1 网格计算系统框架图

从图 1 可以看出整个系统大概工作框架, 首先由用户提交工作请求到 broker, 再由 broker 提交请求信息到 GIS 询问当前所有注册资源的状态信息. 获得这些信息以后, broker 再根据资源的运行速度和可用时间等属性, 基于某些资源配置算法, 将用户

请求发给最后选定的电脑进行处理. 电脑在收到用户请求以后会按照某种配置算法将用户请求放到自己的处理序列里面进行处理. 每次电脑处理完一个请求, 都会把相应的结果发回相应的用户, 同时更新自己在 GIS 里面的状态信息, 直到所有队列里面的请求都被处理完.

从以上过程可以看出, 一个优化的资源配置算法应该可以达到以下目标: (1) 保证尽量多的用户请求可以再截止时间之前完成; (2) 尽量保证完成所有请求的时间最短; (3) 尽量保证所有请求的等待时间最短.

3 资源配置算法研究

在现代应用中, 有很多种资源配置算法, 有些算法重点在于研究如何找出核实的电脑资源来满足客户的要求^[3], 而有些算法重点在于研究如何排列请求可以达到最优化的服务质量要求^[4], 这些所有算法都有一个共同的问题, 用户在提交请求以后都需要了解整个资源配置的过程, 在实际应用中, 用户并不关心资源配置的过程, 而只关心最终得到的结果. 基于以上问题, 本文提出了一种改进型的资源配置算法, 本算法主要分为用户端和资源端两部分.

3.1 用户端优化资源配置算法

3.1.1 优化模型

从用户端来看, 一个优化的算法应该尽量减少每个用户最后一个请求的完成时间和所有请求的平均运行时间. 而这样的最终结果取决于以下因素: 请求的数量、资源的数量和资源的运行能力. 从第 3 节的模型搭建可以看出, 资源配置过程应该由网格 broker 来完成, 因此 broker 需要从 GIS 收集并分析资源信息进而找出适合处理请求的资源.

假设一个单位时间间隔内, 一个用户提交了 n 个请求到 broker, 分别用 $\{J_1, J_2, \dots, J_n\}$ 表示, 它们的长度分别为 $\{W_1, W_2, \dots, W_n\}$, 并且所有这些请求间都是相互独立的. 接下来 broker 就会向 GIS 查询各资源的状态. 假设此时一共有 m 台电脑, 用 $\{M_1, M_2, \dots, M_m\}$ 表示, 它们的处理速度分别为 $\{R_1, R_2, \dots, R_m\}$, 可用的时间为 $\{T_1, T_2, \dots, T_m\}$. 如果此时 broker 将请求 i 分配给电脑 j , 那么对请求 i 的执行时间 $X_{ij} = \frac{W_i}{R_j}$, 总的完成时间应该为 $T_j + X_{ij}$, 对于电脑 j 来说, 完成所有排在它的等待队列里面的请求的时间应为

$$\sum_i (T_j + X_{ij})$$

(1)

在所有电脑里面来比,完成所有请求需要最长的时间为

$$ET = \max\{\sum_i (T_j + X_{ij})\} \quad (2)$$

每台电脑完成请求的平均时间为

$$ART = \frac{1}{n} \times \sum_j \sum_i (T_j + X_{ij}) \quad (3)$$

模型采用多目标粒子群优化算法(简称 PSO)减小 ET 和 ART.

3.1.2 多目标粒子群优化算法

多目标粒子群优化算法是一种基于群智能启发式全局随机优化算法.它是参照群体中目前处于最优位置的个体和自身曾经到达的最优位置来调整下一步的搜寻方向和大小^[5].在基本粒子群算法中,粒子群由 n 个粒子组成,每个粒子的位置 x_i 代表优化问题在 D 维搜索空间中潜在的解.每个解都可以想象成搜索空间上的一个点,称之为粒子.

下面对该算法进行数学描述,算法的目标就是要通过一系列适应性算法可以获得粒子的最佳位置.假设每个粒子 i 包含一个 D 维的位置向量 $X_i = (X_{i1}, X_{i2}, \dots, X_{iD})$,每次计算都有两个最佳位置, p_{best} 和 n_{best} , p_{best} 是指粒子可能被发现经历的最优位置, n_{best} 表示每次迭代初,根据自身惯性和经验及群体最优所经历的位置,据此来调整自己的速度向量已调整自身位置.因此,每个粒子的位置变化按如下公式进行:

$$V_i^{t+1}(j) = V_i^t(j) + c_1 r_1 (p_{best_i}^t(j) - X_i^t(j)) + c_2 r_2 (n_{best_i}^t(j) - P_i^t(j)) \quad (4)$$

$$X_i^{t+1}(j) = X_i^t(j) + V_i^{t+1}(j) \quad (5)$$

上面等式中, V 代表每个粒子的速度; j 代表第 j 维度; i 代表第 i 个粒子, t 代表第 t 步运算. c_1 和 c_2 为正常数,称之为加速因子. r_1 和 r_2 为 $[0, 1]$ 中均有分布的随机数.经过多次迭代以后,算法会判断是否满足程序终止条件,一旦满足条件,即刻终止.

假设以 1 或者 0 表示粒子存在某个位置的潜在可能性,则每个粒子更新的位置向量可以如下式表达:

$$P_i^{t+1}(j) = \begin{cases} 1 & \text{if } \text{sig}(V_i^{t+1}(j)) \geq r_j \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$\text{sig}(V_i^{t+1}(j)) = \frac{1}{1 + \exp(-V_i^{t+1}(j))} \quad (7)$$

3.1.3 资源配置算法中的粒子模型

在本算法中,将所有即将分配的请求对应为 PSO 中的粒子.假设有 m 台电脑,一共用于处理 n 个请求,那么每个粒子的位置就是一个 m 维的向量,

所有粒子的位置信息就是一个 $m \times n$ 的矩阵,用 P_k 表示,矩阵中的元素为 0 或 1.例如

$$P_R = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \quad (8)$$

其对应资源配置图如图 2 所示,其中 $\{J_1, J_2, J_3, J_4\}$ 表示一组请求, $\{M_1, M_2, M_3\}$ 表示电脑.

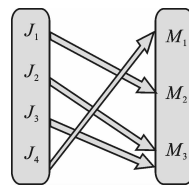


图 2 资源配置图

3.1.4 资源配置算法中的速度模型

PSO 中的速度向量在本算法中也为 $m \times n$ 的矩阵,其各元素的数值范围应该为 $[-V_{\max}, V_{\max}]$. $V_{\max}$ 是每个粒子可移动速度的上限,考虑到移动方向,其值应该为电脑总数的 2 倍,即 $2m$. 同样, n_{best} 和 p_{best} 作为全局最优和局部最优同样以 $m \times n$ 的矩阵来表示.因此算法可以表达为:

$$V_k^{t+1}(j, i) = V_k^t(j, i) + c_1 r_1 (p_{best_k}^t(j, i) - V_k^t(j, i)) + c_2 r_2 (n_{best_k}^t(j, i) - P_k^t(j, i)) \quad (9)$$

$$P_k^{t+1}(j, i) = \begin{cases} 1 & \text{if } V_k^{t+1}(j, i) \text{ is the } \max\{V_k^{t+1}(j, i)\} \text{ for } j \in \{1, 2, \dots, m\} \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

由于速度向量其实表达的是粒子可能转移的位置的可能性,因此这个可能性越大,响应粒子出现在这个新位置的可能性就越大,因此可以用速度向量的最大值来决定粒子新的位置.

3.1.5 资源配置算法中 p_{best} 和 n_{best} 的计算

在本算法中全局最优值 n_{best} 与局部最优值 p_{best} 的计算采用如下适应度函数计算:

$$f = \alpha * ET + (1 - \alpha) * ART \quad (11)$$

α 是一个介于 $[0, 1]$ 之间的数,用来平衡参数 ET 和 ART 对于最终结果的影响的一个加权参数,它的取值应该根据用户的需求和最优解条件来作调整.

整个算法采用多次迭代法,每一次迭代通过计算每个粒子的 f 的值并作比较以后来确定一次 p_{best} 和 n_{best} 的值,迭代中止的条件有两个:(1) 迭代次数达到上限;(2) 多次迭代后 p_{best} 和 n_{best} 的值无明显变化.

3.2 资源端的请求排队算法

介绍一种在资源端的请求排队算法,该算法将

同时满足最小化平均等待时间与请求截止时间的要求.

3.2.1 最小化平均等待时间

假设有 N 个请求在队列中等待处理,每个请求的长度为 $\{W_1, W_2, \dots, W_N\}$,资源的计算速度为 R . 那么第 i 个请求的处理时间即为 $T_i = \frac{W_i}{R}$,则平均等待时间取决于每个请求的处理时间 T_i .

首先,将所有队列中的请求依据其长度进行排序. 假设最短的请求处理时间为 T_1 ,最长的请求处理时间为 T_N . 那么对于请求 i 来说的等待时间为

$$WT_i = T_1 + T_2 + \dots + T_{i-1} \tag{12}$$

对所有请求来说总的等待时间为

$$WT_{total} = 0 + T_1 + (T_1 + T_2) + (T_1 + T_2 + T_3) + \dots + (T_1 + T_2 + \dots + T_{N-1}) \tag{13}$$

将多项式重新排列为

$$WT_{total} = (N-1)T_1 + (N-2)T_2 + \dots + 2T_{N-2} + T_{N-1} \tag{14}$$

从式(14)可以看出,在决定等待时间的时候,短的请求比长的请求占有更大的权重值. 假设 $T_i < T_j$,那么对于着两个请求来说等待时间为 $\alpha T_i + \beta T_j$,其中 α 比 β 大. 如果我们改变 i 和 j 的顺序,则总的等待时间会变成 $\beta T_i + \alpha T_j$. 很明显, $\alpha T_i + \beta T_j$ 比 $\beta T_i + \alpha T_j$ 要小,因此可以看出任何其他的排列方式都比这种排列方式的平均等待时间要长.

3.2.2 截止时间

任何请求都需要在截止时间之前完成,因此在考虑最短平均等待时间这个要求的同时,也需要考虑截止时间的要求. 因此在将所有请求按照其长度排序以后,再开始逐个计算检查每个请求的截止时间. 如果依据排序所需要的完成时间点 in 截止时间以后,那么就将请求往前挪 1 位,再重新计算该请求所需要的完成时间,重复以上过程,直到计算出可以在截止时间之前完成为止.

4 实验仿真与结果分析

为了验证本算法性能的优越性以及今后研究的延续性,本系统采用一个基于 GridSim 和 JAVA^[6] 的仿真器对以上算法进行仿真.

4.1 仿真实验参数设置

在本次仿真实验中,首先设置了 8 个资源用来模拟 8 台电脑,这 8 个资源的计算能力依据泊松分布随机产生,产生的计算能力如表 1 所示.

其次设置了 10 个用户,每个用户有 10~15 个

表 1 仿真计算能力

Resource index	1	2	3	4	5	6	7	8
Resource speed (MIPS)	78	87	96	100	112	114	126	133

请求需要处理,每个请求的长度随机产生. 所有请求的平均长度位 527MI. 开始对本文所设计的资源配置优化算法进行仿真,同时将结果分别与粒子群优化算法^[7]和 Max-Min 排列算法^[8]进行比较.

4.2 仿真结果讨论

接下来将通过以下三个参数对仿真结果进行讨论:请求处理成功率、完成时间和平均等待时间.

针对以上提到的三种算法,相应的请求处理成功率对比图如图 3 所示. 从图上可以看出,本文所设计的资源配置优化算法的成功率达 96%,高于粒子群优化算法的 74%和 Max-Min 排列算法的 87%. 对于粒子群优化算法来说,只考虑了最小化平均等待时间,没有考虑截止时间的要求,因此一些非常紧急的请求可能会被延迟错过它们的截止时间. 对于 Max-Min 排列算法来说,考虑了每个请求的截止时间,但是没有在用户端进行对资源配置,因此可能导致有的资源被分配了太多的请求堵塞导致一些截止时间早的请求可能会错过它们的截止时间.

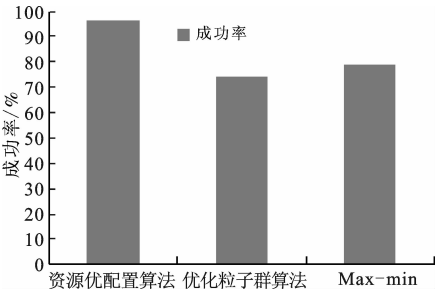


图 3 三种算法请求处理成功率对比图

三种算法的请求完成时间对比如图 4 所示. 在粒子群优化算法种,由于有些请求错过了它的截止时间,将不被处理导致该算法的完成时间比本文提出的资源优化配置算法要短一点. 但是和 Max-Min 排列算法相比较,本文提出的资源配置优化算法的时间就要短得多. 原因在于 Max-Min 排列算法没有考虑资源的优化配置,导致资源阻塞的情况更容易出现,一旦阻塞情况出现,那么完成时间将被延迟.

三种算法的平均等待时间对比图如图 5 所示,从图上可以看出平均等待时间最短的是 Max-Min 排列算法,原因是因为该算法没有考虑最长请求所需要的等待时间. 一旦这个等待时间过长就会被系

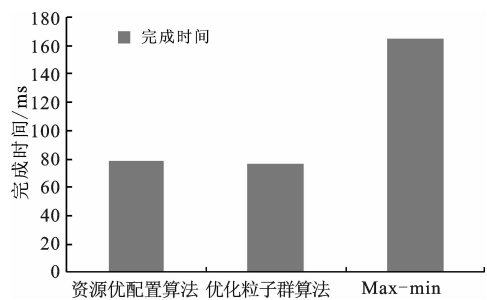


图 4 完成时间对比图

统自动放弃处理,从而导致请求处理的成功率降低.对于粒子群优化算法来说,由于没有考虑资源优化配置,所以其平均等待时间比另外两种算法要长的多.

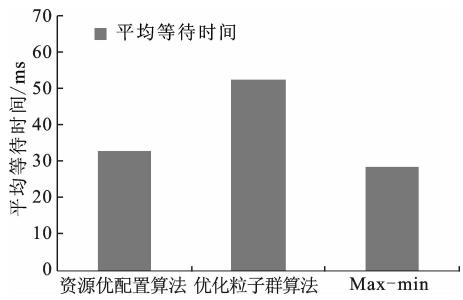


图 5 平均等待时间对比图

综上所述,尽管针对某些因素来说,本文所设计的资源优化配置算法并不是性能最好的,但是在综合考虑所有因素来说,它的性能比另外两种算法性能要优越^[9-10].

5 结束语

本文提出了一个基于资源优化配置算法的网格计算系统.首先,介绍了该网格计算系统中的四个重要部分,分别为网格用户、网格 broker,网格资源和网格信息服务中心.其次,对系统中的核心部分—资源优化配置算法进行了详细的介绍,该算法包含两个部分:用户端利用粒子群优化算法对资源进行优化配置和资源端对请求进行排队处理.最后,对系统进行了仿真并将结果与两种已有的算法系统进行比较,体现了本文提出的网格计算系统的优越性.

参考文献:

[1] Mo C, Liu B. Research and realization of an anti-noise au-

to-focusing algorithm[C]// Intelligent Human-Machine Systems and Cybernetics (IHMSC). China, Hangzhou, 2013 5th International Conference on. 2013; 255 - 258.

[2] 刘玉玲, 杜瑞忠, 田俊峰, 等. 基于信任力矩的网络资源选择模型[J]. 通信学报, 2012, 33(4): 85-92.

[3] 张忠平, 温利娟. OPT-Min-Min: 基于 Min-Min 网格资源调度算法的优化[J]. 小型微型计算机系统, 2014, 3(7): 1573-1577.

[4] Kumar M S, Purusothaman T. Explicit allocation strategy with deadline and budget constraint algorithm in bag of tasks grid [J]. Journal of Computer Science, 2012, 8(7):110-117.

[5] Aghazarian V, Delavar A G, Motlagh N G, et al. RQSG-I: an optimized real time scheduling algorithm for tasks allocation in grid environments[C]// Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. China, Xi'an IEEE, 2011;205-210.

[6] 薛洪波, 伦淑娴, 粒子群算法在多目标优化中的应用综述[J], 渤海大学学报: 自然科学版, 2009, 30 (3): 265-269.

[7] Altmann M Q D J. Resource allocation algorithm for light communication grid-based workflows within an SLA context [J]. International Journal of Parallel Emergent & Distributed Systems, 2009, 24(1):31-48.

[8] Zheng Z, Zhao T, Zhang Y. et al. Optimization of grid resource allocation using improved particle swarm optimization algorithm [C]//2010 International Forum on Information Technologies and Application. China, Qingdao, 2010

[9] DoulamisNikolaos D, DoulamisAnastasios D, Varvarigos Emmanouel A, et al, Fair scheduling algorithms in grids [J]. IEEE Transactions on Parallel and Distributed Systems, 2012,2(3): 1631-1647.

[10] 乔百友, 邓增安, 王秋杰, 等. 一种基于网格索引的空间连接查询处理优化算法[J]. 小型微型计算机系统, 2014, 4(10): 2243-2248.

作者简介:

周建鸿 女,(1984-),博士. 研究方向为计算机软件与无线网络. E-mail:zhou_jianhongs@sina. com.

马懋德 男,(1957-),博士,教授. 研究方向为无线网络安全与光学网络等.