

基于高效负载均衡的网络任务分配技术

苏恒阳

(广东轻工职业技术学院 实验实训管理中心, 广东 广州 510300)

摘 要: 针对传统的网格在分配任务的过程中, 当多任务发生竞争的时候, 由于多条路径的任务负载不均衡, 造成传输速度慢的问题, 提出一种基于遗传算法负载均衡的网格 workflow 算法. 通过模拟自然界的生物进化过程对任务空间进行随机化搜索, 根据预定任务的适应度函数, 并用全局并行搜索方式找到最优节点, 避免了传统方法的负载不均衡问题. 实验证明, 该算法实现了负载的快速均衡, 提高了网格 workflow 系统的工作效率.

关键词: 网格 workflow; 负载均衡; 遗传算法

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2011)07-0175-04

Based on the Efficient Load Equilibrium Network Task Allocation Technology

SU Heng-yang

(Experimental Training Management Center, Guangdong Industry Technical College, Guangzhou 510300, China)

Abstract: The problem of grid workflow technology. In traditional grid in assigned process, when multitasking competition, because the task of pathskilling transmission unbalanced load slow problem. This paper presents a genetic algorithm-based load balancing algorithm for grid work flow. By simulating the natural process of biological evolution were randomized to the task of the search space, according to the fitness function of scheduled tasks, and parallel with the global search for ways to find the best node, method is to avoid the traditional balance of the implementation of successive matching problems caused by inefficient. Experimental results show that the new algorithm is fast load balancing, grid workflow system to improve the work efficiency, and achieved satisfactory results.

Key words: grid workflow; load balancing; genetic algorithm

1 引言

目前已成为网络领域的研究热点. 网格 workflow 的负载均衡具体过程是, 首先控制网格 workflow 中每一个子任务执行的顺序, 然后在网格中所有的许多候选服务网络节点中选择适合的网络节点进行执行, 其中此具体过程的核心部分就是最佳网络节点的选取^[1-2].

传统的方法是将所有的子任务逐个分配给网格中的全部节点, 并计算每个子任务在每个网络节点中的执行时间, 执行速度最快的节点就是此子任务

的最佳节点. 当任务增多的时候, 系统会自动调用最佳路径传输任务, 这就造成了最佳路径被频繁调用, 出现信息堆积, 而其他空闲的非最佳路径没有参与任务的传递, 造成传输资源调度的负载不均衡的问题, 不能及时完成网格 workflow 中的任务.

针对传统均衡算法的缺点, 文中提出一种基于遗传算法负载均衡的网格 workflow 算法. 通过模拟自然界的生物进化过程对任务空间进行随机化搜索, 根据预定任务的适应度函数, 并用全局并行搜索方式找到最优节点, 避免了传统均衡方法的逐次大量计算带来的低效问题.

收稿日期: 2010-10-15; **修回日期:** 2010-11-03

基金项目: 学院自然科学基金项目(KJ201029)

2 网络 workflow 技术原理

网络 workflow 系统连续运行一段时间后,将会出现这样一种情况:个别网络 workflow 节点被分配给的任务非常多,但是其他的一些网络 workflow 节点被分配的任务却非常的少.前者节点也被称为重在节点,后者节点也被称为轻载节点.这种情况的产生就称之为网络负载不均衡的现象,会造成网络传输效率变低.为了使所有任务和网络中各节点的需求相平衡,以到达高效完成所有任务的目的,就此引入了负载均衡的概念^[3-4].

负载均衡最终要达到的目的是,通过根据包含大量 workflow 节点的工作流网络系统中各个网络 workflow 节点的计算性能,以及网络中各节点之间的通讯能力等多种参数,把网络中的所有任务分解成为多个部分子任务,然后分别将所有的分解子任务用最佳的分配方式,找到符合实际要求的网络 workflow 节点来完成这项子任务.可知,负载均衡的实质就是,在由 M 个需要负载均衡的任务、 N 个可以使用的网络 workflow 节点、 K 个数据储存单元构成的网络环境条件下,把 M 个负载均衡子任务 $R = \{R_1, R_2, \dots, R_M\}$ 以最佳的方式,分别负载均衡到 N 个网络 workflow 节点 $D = \{D_1, D_2, \dots, D_N\}$ 上去,从而得到尽可能小的总执行时间,最终实现利用网络 workflow 系统快速完成用户所提出的网络任务.

传统的负载均衡算法,是将任务分解为多个子任务后,逐个将子任务分配给最早可用且执行最快的节点,需要多次计算把任务分配到 N 个节点上的 N 个不同完成时间再比较选出最佳分配方式,然后将此子任务非配到相应的最佳节点上,并将此子任务从任务集合 R 中删除,继续进行下一个子任务的均衡,如此反复循环执行,直至任务集合为空,即完成了所有的负载均衡.

由上述内容可知,传统的均衡算法计算量很大,需要多次执行循环访问 workflow 节点,算法的均衡效率公式^[5]为:

$$S = \frac{N}{M^{N-1}} \ln \frac{1}{1+e^x} \quad (1)$$

式中, M 为子任务个数, N 为网络中 workflow 节点数, x 为所有任务集合中的子任务访问 N 个节点次数.由式(1)可知,访问节点的次数 x 与均衡效率成反比,传统方法多次访问 workflow 节点, x 值很大,导致均衡效率低,不能快速完成网络 workflow 任务.

文中提出一种基于遗传算法负载均衡的网络工

作流算法.将子任务的可能匹配节点看作是一个个体或染色体,模拟达尔文的生物进化过程,对个体进行选择、交叉和变异等遗传学操作,通过模拟自然界的生物进化过程对任务空间进行随机化搜索,根据预定任务的适应度函数,并用全局并行搜索方式找到最优节点,避免了传统均衡方法的逐次大量计算、多次访问 workflow 节点带来的低效问题.

3 基于负载均衡的高效网络 workflow 技术

3.1 workflow 节点的负载参数选择

对于不同类型的网络 workflow 系统,各种 workflow 节点参数的重要程度有所不同.为了满足不同网络 workflow 系统运行过程中的要求,需要将各个节点参数的比例进行调整,首先为每一个节点参数给定一个特定的常量系数 π_i ,表示各个节点参数的重要程度,同时满足 $\sum \pi_i = 1$.可以得到每一个节点 i 在 j 时刻的节点负载权值公式:

$$W_{ij} = \pi_1 \cdot C_{ij} + \pi_2 \cdot M_{ij} + \pi_3 \cdot T_{ij} + \pi_4 \cdot Io_{ij} + \pi_5 \cdot P_{ij} \quad (2)$$

负载均衡预测就是对权值 W_{ij} 的预测.由研究分析知,相邻近的权值数据很容易影响下一时刻的节点负载值.通过观察在过去的一段时刻中是否大多数的预测值 W_{ij} 在误差范围之内,来判断某个 workflow 节点的负载状态是稳定态还是随机态.通过检测并删除误差较大也就是百分比误差比较大的一些时刻,防止了某些异常时刻的权值负载预测误差对系统模型所处状态的判断的影响.

文中设定网络 workflow 节点的集合 N 是包含有 m 个网络节点的集合, k 的值表示已给定的用作考察合理范围的表示, pa 为需要事先给定的百分比, ra 也是需要事先给定的一个值.将变量如下定义初始化为 $k = 200$, $pa = 5\%$, $ra = 0.05$.

对集合 N 中的任意一个节点 $n_j (j = 1, 2, \dots, m)$

(1) 计算在以前的 200 个时刻中的节点负载权值预测误差:

$$\Delta W_{ij}^t = W'_{ij} - W_{ij} \quad (t = 1, 2, \dots, 200) \quad (3)$$

将 ΔW_{ij}^t 从小到大进行排序,删除误差最大的那个时刻,余下的时间组成一个新的时刻集合 Sa .

(2) 对集合 Sa

$$R_j = \sqrt{\frac{\sum_i (W'_{ij} - W_{ij})^2}{N}} \quad (4)$$

$$i \in Sa; j = 1, 2, \dots, m$$

其中, N 为集合 Sa 中时刻的个数.

如果 $R_j < ra$ 那么 workflow 节点 n_j 的负载预测模型为稳定状态. 那么 workflow 节点 n_j 的负载预测模型为随机状态.

3.2 遗传算法的编码设计

采用自然数直接编码的形式对染色体进行编码, 设定染色体的长度与节点要均衡的任务数相等, 用 $X = \{k_1, k_2, \dots, k_i, \dots, k_n\}$ 代表染色体. 将染色体定义成一个两元组 $k_i\{s_i, p_i\} (i = 1, 2, \dots, n)$ 式中的 s_i 是与第 i 个负载子任务匹配成功的网络 workflow 节点, 根据匹配原则知道负载子任务的编号具有唯一性, 在算法的最后结束时每个负载子任务都能找到一个网络 workflow 节点与它相匹配, 并且不同的负载子任务与其匹配的资源是相同的; p_i 是每个子任务是优先权值, 形成一个符合先后次序的优先权列表.

如果存在一条从 V_i 到 V_j 的路, 则把 V_i 称为 V_j 的前驱节点, 而对于 $(V_i, V_j) \in E$, 则将 V_i 称为 V_j 的立即前驱节点, 记为 $V_j \in iPred(V_i)$, 则将 V_i 称为 V_j 的立即后继节点, 记为 $V_j \in iSucc(V_i)$.

子任务的优先权列表由下面公式构成:

$$P(V_i) = \begin{cases} p(V_i) & \text{if } iPred(V_i) = \phi \\ \max(Q(V_j) \mid V_j \in iPred(V_i) + p(V_i)) & \end{cases} \quad (5)$$

在网格 workflow 系统中, 为了提高算法收敛的速度并改善算法的结果, 必定要求资源的均衡要有较高的实时性, 要求系统的开始建立的模型中的个体需要兼具随机性和优良性. 通过上面研究使用负载子任务的复制方法的负载均衡算法得知, 如果将复制的主要路径上的任务分配到多个网络 workflow 节点中去执行, 就能够将系统的总运行时间大大缩短, 产生初始模型的步骤如下所示:

(1) 对存在于每个关键道路中的任务, 在 $[1, 2^k - 1]$ 中产生一个随机数, 这个随机数必须满足均匀分布, 充当执行任务的时间值. 相应的, 将不是关键道路上的任务, 在 $[0, k - 1]$ 中也产生一个随机数 r , 同时也服从均匀分布, 并把 2^r 当做此项任务的执行时间值.

(2) 在 $[1, 512]$ 内随机为每个任务产生一个随机数充当此项任务的优先值, 并且此随机数也满足均匀分布, 至此就生成了多个个体.

(3) 想要产生包含多少个个体的种群, 就重复执行多少次这样的过程即可.

评价染色体优劣是用适应度函数来度量的. 设定在算法中, 均衡完成的时间也就是所有任务逐次完成所有的总时间决定了适应度函数的值, 所有子

任务完成的用时越短, 染色体就越好, 此个均衡序列就越靠前优先执行.

因为每个子任务完成的时间是一个已知的定值, 所以如果计算得到前驱网络 workflow 节点的总完成时间的值, 就能够计算得到现时正在执行的负载子任务的最短的完成时间的值. 以此类推, 就可以全部计算出自任务集合中所有子任务的最终执行时间. 设定网络 workflow 节点集合中最后的子任务 N_i 能够顺利完成所消耗的具体时间表示为 $ETC(N_i)$, 那么集合中所有的子任务最终全部执行成功并完成的时间就可以表示为 $\max(ETC(N_i))$. 并且将它的适应度函数作如下定义:

$$\text{Fitness} = 1/\max(ETC(N_i))$$

由上式可以看出, 适应函数的值越大, 所得的染色体质量就会更加优秀.

3.3 基于遗传算法的负载均衡

由上节中的内容产生染色体后, 根据算法中的选择算子选出有资格进入下一代的那些染色体, 采用“轮盘输”的选取方法, 通过比较计算后而得到的染色体适应函数的值的大小, 以此来确定现时的染色体被实验挑选选中的具体概率值. 如果现在这个染色体的适应函数的数值越大, 则可知现在的这个染色体能够被选中的概率也就更大. 染色体 r_i 被选择的概率公式如下:

$$P(r_i) = \text{Fitness}(c_i) / \sum_{j=1}^{pSize} \text{Fitness}(c_j) \quad (6)$$

在确定每个染色体个体被选中的概率值后, 网格 workflow 系统将会在 $[0, 1]$ 区间内随机的选择产生随机数组, 将此数组与此染色体的被选择的概率相比较, 如果随机数大于此染色体被选中的概率, 这样此染色体就被选中, 否则不被选中而淘汰.

基于遗传算法的交叉算法的具体的做法如下:

(1) 随机选取两个染色体个体, 并将所有的没有被选中的 workflow 节点标记;

(2) 判断如果全部的节点都被选中过, 那么转向步骤(4), 否则选取一个并没有被选中过的工作流节点并标记成已经被选中, 选取网络 workflow 节点上的任意两个子任务并且相互作比较, 如果所选取的两个 workflow 节点上的所有任务都相同, 则多次重复循环执行步骤(2), 否则将流程转向步骤(3);

(3) 在(2)中最后被选中的节点 j , 选取网络 workflow 节点中包含的任意两个染色体, 并且将它们此节点的适应函数的值作比较, 将适应函数值高的那个节点被分配的任务设为没有分配的状态, 将另一个染色体节点上分配的任务均衡到此染色体个体的

workflow 节点上;

(4) 基于遗传算法的交叉算法完成。

通过对所有 workflow 节点的均衡, 将网格中的任务分解为多个子任务, 并把这些子任务均衡到适合的 workflow 节点执行, 这样得到的执行时间最短, 保证了均衡的效率, 使得网格 workflow 系统能够快速完成网格中的所有任务。

4 实验结果与仿真

文中提出的基于遗传算法负载均衡的网格 workflow 算法, 通过模拟自然界的生物进化过程对任务空间进行随机化搜索, 根据预定任务的适应度函数, 并用全局并行搜索方式找到最优节点, 最终达到网格 workflow 系统中各 workflow 节点负载均衡的目的, 同时提高了网格 workflow 系统的工作效率。具体的流程图如图 1 所示。

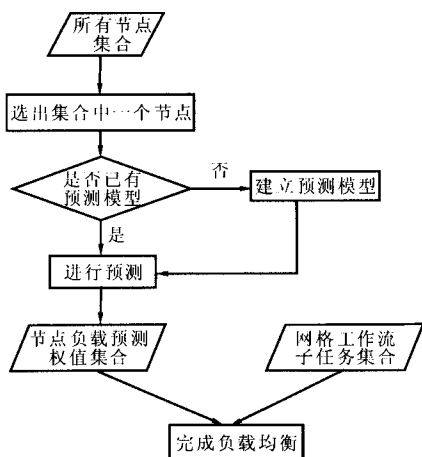


图1 基于负载均衡的均衡算法流程图

针对一个具体的产品开发问题, 将几个公司建立网格联系起来, 每个公司即一个 workflow 节点, 分别用传统算法和文中算法进行负载均衡, 对访问各个节点的次数进行统计, 如图 2 所示。

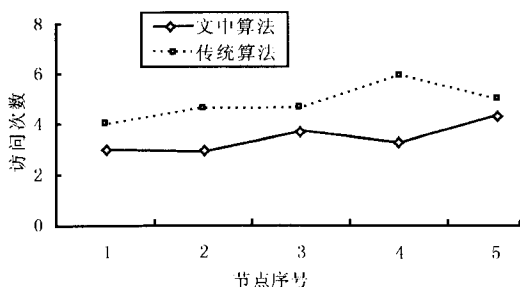


图2 对各节点访问次数统计波形图

由图 2 可以看出, 传统方法访问节点的次数明显多于基于遗传算法的访问节点次数, 根据均衡效

率公式 $S = \frac{N}{M^{N-1}} \ln \frac{1}{1+e^x}$ 知, 传统算法的 x 值大于基于遗传算法的 x 值, 显然由公式可知传统的均衡算法对负载的均衡效率较低。计算两种算法的负载均衡效率, 并统计两种方法完成此项产品开发所用的时间, 统计结果如表 1 所示。

表1 两种方法均衡效率及耗时比较

算法	均衡效率	总耗时
Min_min 算法	64%	700 s
文中算法	98%	253 s

由表 1 对两种算法的比较结果可知, 传统的 Min_min 算法由于需要逐个将子任务集合中的子任务分别访问各节点找到最佳匹配节点, 使得负载均衡的效率很低。而基于遗传算法的网格 workflow 系统, 通过模拟自然界的生物进化过程对任务空间进行随机化搜索, 根据预定任务的适应度函数, 并用全局并行搜索方式找到最优节点, 避免了传统均衡方法的逐次大量计算、多次访问 workflow 节点带来的低效问题, 最终总耗时远小于传统 Min_min 算法。

5 结束语

文中提出了一种基于遗传算法负载均衡的网格 workflow 算法, 通过模拟自然界的生物进化过程对任务空间进行随机化搜索, 根据预定任务的适应度函数, 并用全局并行搜索方式找到最优节点, 避免了传统均衡方法的逐次大量计算带来的低效问题。实验证明, 这种新算法实现了负载的快速均衡, 提高了网格 workflow 系统的工作效率。

参考文献:

- [1] 余波, 周龙骧, 钟锡昌. 网络 workflow 技术综述[J]. 计算机工程, 2006, 32(2): 4-8.
- [2] 曹怀虎. 网络环境中任务调度算法的研究[J]. 计算机工程与应用, 2004, 40(5): 87-90.
- [3] Fan J, Yao Q. Nonlinear Time Series (Nonparametric and Parametric Methods)[M]. 北京: 科学出版社, 2006.
- [4] 范军涛, 李国庆, 康琳. 图像处理 workflow 内部的数据存储[J]. 计算机仿真, 2007, 8(50): 182-184.
- [5] 王生发, 潘敏, 郭剑锋. 基于模糊决策的 workflow 互操作研究[J]. 微电子学与计算机. 2008, 25(7): 100-102

作者简介:

苏恒阳 男, (1971-) 实验师, 主要研究方向: 计算机网络、数据库技术、电子技术。