一种基于多维 QoS 约束的网格作业调度算法①

A Grid Task Scheduling Algorithm Based on Multi-QoS Constraints

杨长兴 周军成 (中南大学 信息科学与工程学院 湖南 长沙 410083)

摘 要: 任务调度是实现高性能网格计算的一个重要方面,然而,由于网格资源的动态性、异构性等特征,设计高性能的任务调度算法是一项非常有挑战性的工作,该问题已被证明是一个 NP 难题。文章中提出了一种新的任务调度算法,该算法根据任务 QoS 约束以及计算资源性能指标,建立任务调度的线性模型,并根据任务的需求和偏好,从线性模型中得到最优的任务分配方案。模拟实验结果表明:对大量独立任务进行调度时,该算法在满足用户需求方面优于其它算法。

关键词: 网格计算 任务调度 多维 QoS 约束 线性模型 Gridsim

1 引言

网格技术(Grid Computing)已经成为当前国内外计算机技术研究的重点和前沿领域,是当前互联网研究中的热点,被称为下一代的 Internet。任务调度作为其重要的组成部分之一,逐渐成为网格研究的重点。在网格计算中,如何有效利用动态、异构、复杂的网格资源、如何高效率、高可靠性的完成用户提交的任务,都是实现高性能网格计算的重要方面,也是任务调度所要面临的挑战。在以往的研究中,已经产生了许多的调度算法,如 Min-Min[1]算法、Max-min[1]算法、遗传算法[2]、模拟退火算法[3]等,这些算法已被广泛用于任务调度中。虽然,这些传统的算法,在提高资源利用率方面取得了不俗的成果,但它们有一个共同的特点,没有把服务质量(QoS)作为一个重要因素考虑进去,无法广泛的适用于复杂多变的网格环境。在网格环境下,如何根据任务的 QoS 要求,把任务有效的分配到合适的资源上,以及在此基础上,最大限度的充分利用有限的资源,成为当前任务调度研究中的重点内容。

本文中提出了一种基于任务 QoS 约束与用户需求偏好的算法,克服了传统算法中没有考虑任务 QoS 约束的局限性,同时,也从用户角度出发,把用户需求偏好纳入了算法的考虑范畴,既最大限度有效利用资源又很好的满足了用户需求。经过模拟实验证实,本算法具有很好的调度效果。

2 问题描述

本文研究在异构的网格计算平台下独立任务的调度问题。通过分析,我们可以知道,对任务调度有较大影响的因素主要有以下几点:(1)用户的需求,主要指用户对任务调度的一些要求,例如,用户在提交任务时,希望任务执行的成本最低,也就是说任务应该被调度到执行价格最低的计算资源上去执行,因此,我们的调度算法在对任务的调度过程中应当尽量满足用户的这一方面的需求。(2)任务的 QoS 约束需求,主要指任务对计算资源在各个方面的性能要求。(3)计算资源的 QoS 属性,主要指计算资源的性能指标。怎样把以上三个因素反映到调度模型中去是设计的难点。本文提出了综合 QoS 需求以及综合 QoS 性能两个概念。综合 QoS 需求是反映任务的 QoS 需求、用户需求的一个综合指标,对任务的调度次序起到决定作用。综合 QoS 性能是反映用户的需求、计算资源性能的一个综合指标,对计算资源的调度次序起到决定作用。

① 收稿时间:2008-07-21

通过获得综合 QoS 需求、综合 QoS 性能这两个指标，任务调度从一个多维因素问题转化为一维因素问题，降低了调度关系处理的复杂度。以上两个指标的具体计算公式在下文中有详细介绍。

图 1 给出了具有三个计算能力不同的计算节点的网格计算平台下的独立任务的调度示例。左框内表示四个任务节点，节点的权值表示该任务的综合 QoS 需求，右框内表示三个计算节点，节点的权值表示该计算节点的综合 QoS 性能，任务节点与计算节点的连线表示任务与计算节点的调度关系，连线的权值表示各任务的调度次序。从该调度示例中，我们可以看出两点：⑴综合 QoS 需求大的任务优先得到调度。⑵综合 QoS 性能高、未被分配的计算节点优先得到调度任务。以上两点是本文中所提出的调度算法的核心思想，在本文的后续部分将会更加详细的对其进行分析。

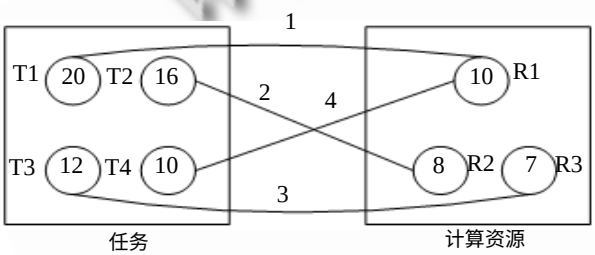

图 1　任务调度示例

3　调度模型

3.1 基本模型

我们把调度模型表示为一个由多个子集构成的集合,从而将对子任务的调度问题转化为处理子集之间的关系问题。调度模型集合的表示：G = {C，P，T，R，E，U，V}。

C：表示任务的 QoS 需求的集合，C = {C1，C2，…，Ci} (1≤i≤n)，n 表示 QoS 需求总数，Ci 表示对应 QoS 需求中的某个约束值，例如，第 i 项表示任务对资源网络带宽的要求，如果要求带宽大于或等于 2MB/s，则 Ci = 2MB/s,或要求带宽小于或等于 2MB/s,则 Ci = 1/2MB/s；

P：表示任务的 QoS 需求系数集合，P = {P1，P2，…，Pi} (1≤i≤n，　0≤P1 + P2+…+ Pi≤1)，n 表示 QoS 需求总数。集合 P 中 Pi 与集合 C 中 Ci 对应，集合中各项由用户在提交任务时输入，反映用户对任务中个约束条件的偏爱程度，Pi 越大代表用户对约束条件 Ci 越重视；

T：表示任务集合，T = {T1，T2，,…，Ti} (1≤i≤n)。Ti 由集合 C 与集合 P 构成，即 Ti ={C,P}，n 表示任务总数；

R：表示资源集合，R = {R1，R2，…，Ri} (1≤i≤m)。Ri 由集合 C 构成，Ri ={C}，m 表示计算资源节点总数；

E：表示任务调度集合，E = {(Rj ,Ti)} (1≤i≤n，1≤j≤m)，其中(Rj ,Ti) 表示任务 Ti 分配到资源 Rj 上执行，n 表示任务总数，m 表示计算资源节点总数；

U：表示任务的综合 QoS 需求的集合，U = {U1，U2，…，Ui} (1≤i≤n)。Ui 表示根据任务 Ti 的多维 QoS 需求以及 QoS 需求系数，通过相应的计算公式，转化为一维的 Ui，它是任务调度中的一项重要依据。Ui 的计算公式如下：

$Ui=\sum_{t=1}^{n}P(i,j)*(R(i,j)/\sum_{t=1}^{m}R(t,j))$，其中 P(i,j)表示任务 Ti 的第 j 项 QoS 需求系数,R(i,j)表示任务 Ti 的第 j 项 QoS 需求，m 表示任务总数，n 表示 QoS 需求总数；

V :表示计算资源综合 QoS 性能集合,V = {V(1,1)，V(1,2)，…，V(i,j)} (1≤i≤m,1≤j≤n)。V(i,j)表示资源 Ri 相对于任务 Tj 的性能，该值是一相对值，作为任务选择资源的一项依据。V(i,j)的计算公式如下：

$V(i,j)=\sum_{t=1}^{n}P(j,t)*(R(i,t)/\sum_{z=1}^{m}R(z,t))$，其中 P(j,t)表示任务 Tj 第 t 项的 QoS 需求系数，R(i,t)表示资源 Ri 第 t 项的 QoS 性能，m 表示计算资源节点总数，n 表示 QoS 需求总数；

在任务调度时，集合 C，P，T，R，U，V 是已知的，关键是如何得到任务与资源的最佳调度关系，即集合 E。要得到集合 E = {(Rj ,Ti)} (1≤i≤n，1≤j≤m)，(Rj ,Ti) 必须满足以下条件：

$$\begin{cases}(1)Rj\ 未被分配\\(2)Rj\ 满足任务Tj\ 约束条件\\(3)V(i,j)=\max(V(i,t))\ 其中1\le t\le m,Rt\ 未被分配\\(4)Ui=\max(Ut)\ 其中1\le t\le n,Tt\ 未被调度\end{cases}$$

3.2 任务调度实例

考虑一个只有 3 个计算节点的网格计算环境,各计算节点的计算能力、带宽、使用资源单位价格存在差异,如表 2 所示，资源集合 R = {R1，R2，R3}，根据模型中的集合构造方法，可知 R1 = {900，9，0.5}，R2 = {800，10，1/3}，R3 = {600，10，0.5}。在该环境下提交 3 个单位任务并且假定用户提交的 QoS 需求系数为：P = {0.1，0.1，0.8}，如表 1 所示，任务只考虑计算能力、带宽、资源单位价格三个方面的 QoS 约束，任务集合 T = {T1，T2，T3}，其中 T1 = {{600，10，1/3}，P}，T2 = {{900，8，0.5}，P}，T3 = {{800，5，0.25}，P}。通过计算，可以得到集合 U = {0.315，0.443，0.241}，V={ V1，V2，V3 }，V1={0.37，0.37，0.37}，V2={0.27，0.27，0.27}，V3={0.36，0.36，0.36}。

表 1　任务属性

	CPU 处理能力（MIPS）	网络带宽 (Gbits/s)	价格 ($/s)
T1	≥600	≥10	≤3
T2	≥900	≥8	≤2
T3	≥800	≥5	≤4

表 2　计算资源属性

	CPU 处理能力（MIPS）	网络带宽 (Gbits/s)	价格 ($/s)
R1	900	9	2
R2	800	10	3
R3	600	10	2

在得到集合 R，T，P，U，V 以后，我们可以按照模型中给出的调度条件为每一个任务分配计算资源：

（1）集合 U 中当前未调度任务所对应的值中，最大值是 0.443,该值所对应的任务是 T2，则 T2 被调度，满足 T2 的 QoS 约束条件的资源只有 R1，则任务 T2 分配到资源 R1。

（2）集合 U 中当前未调度任务所对应的值中，最大值是 0.315,该值所对应的任务是 T1，则 T1 被调度，满足 T1 的 QoS 约束条件的资源有 R2、R3，而(R2，T1)对应集合 V 中的值为 0.27，(R3，T1)对应集合 V 中的值为 0.36，则任务 T1 分配到资源 R3。

（3）集合 U 中当前未调度任务所对应的值中，最大值是 0.241,该值所对应的任务是 T3，则 T3 被调度，满足 T3 的 QoS 约束条件的资源有 R1、R2，而(R2，T3)对应集合 V 中的值为 0.27，(R1，T3)对应集合 V 中的值为 0.37，则任务 T1 分配到资源 R1。

经过以上调度步骤，三个任务都已经调度完成，得到了调度关系集合 E = {(R1 ,T2) ，(R3,T1) ，(R1,T3)}。从分配的关系集合中，我们可以看出，资源 R1 中先后分配了两个任务，似乎造成了负载的不平衡，任务 T3 完全可以分配到资源 R2 上去执行，资源 R2 是满足任务 T3 的各项 QoS 约束条件的，但是，我们注意到在这次调度中，用户输入的 QoS 约束系数是：P = {0.1，0.1，0.8}，从系数中可以看出，用户对任务的执行代价是非常重视的，系数取值是 0.8，也就是说，用户对其他的因数并不是很在乎，他们关心的是任务执行的成本代价。通过计算，可以知道，按照集合 E 中的调度方式调度任务成本是最低，从这个角度出发，这次调度是非常成功的，它很好的满足了用户的需求。

4　算法描述

经过前面对调度模型的讨论，我们已经能够通过对任务信息以及计算资源信息的收集，建立起一维的线性模型，从中得到最适合用户的调度关系。在最大程度满足用户需求的前提下，我们还应该考虑两点：1.怎样最大程度的利用现有的计算资源；2.怎样保证任务在调度中的公平性。例如，有 3 个独立任务，以及 6 个计算资源节点，我们按上文讨论的调度方法，将 3 个任务调度到其中的 3 个资源节点上执行,这样，其余的 3 个计算资源节点就没有被利用。为了充分利用资源，提高任务调度性能，在算法中，融入了 RR[4] 算法中的思想。RR 算法的基本思想是：把所有的任务看成一个首尾相连的循环链表,只要当前存在未被分配的计算资源，则循环调度链表中的任务,一旦某任务

执行完毕,返回结果,则把该任务从链表中删除,并且终止该任务在其它计算资源节点上的执行,释放该任务所占有的计算资源。同样，为了解决任务调度过程中的公平性问题，我们对任务实行等级制原则，具体有如下两点：①任务在提交上来时，系统赋予其初始等级;②任务在调度中，如果未发现合适的资源，表示任务本次调度没有成功，任务的等级提高一个等级，否则，表示任务调度成功，任务等级降低一个等级。算法的具体描述如下：

While(任务链表 T 不为空) {

（1）根据任务的等级、任务所对应的 U 集合(U 集合在前文中定义)中的值按从大到小顺序排列任务。

（2）从任务链表中取出头节点所表示的任务 Ti，任务链表头指针指向下一节点。

（3）如果该任务未被调度过，或被调度过，但该任务允许复制调用，则执行 4，否则执行 7。

（4）根据任务 Ti 得到集合 V(j,i)= { V(1,i)，V(2,i)，…， V(m,i)(V 集合在前文中定义)，其中 m 为当前计算节点总数。

（5）如果存在未被分配的计算节点 Rt 满足任务 Ti 的约束需求，并且 V(t,i)等于 max(V(j,i))，其中 1≤j≤m，Rj 未被分配，则执行 6，否则，将任务 Ti 等级提高一个等级，执行 7。

（6）把任务 Ti 分配给节点 Rj，将任务 Ti 等级降低一个等级，资源 Rj 标志成已分配。

（7）如果已经完成对任务链表完成遍历，则执行 8，否则执行 2。

（8）收集任务信息以及计算资源信息，如果有任务完成，则把该任务从任务链表中删除，并释放该任务所占有的计算节点。更新任务链表与资源信息表。

}

5　实验结果与分析

本文采用 Gridsim 和 eclipse 来模拟和实现上文中提出的算法。本次模拟实验中，我们主要做了以下工作：①搭建一个具有 500 个计算节点的网格计算环境。②实现 100 至 1000 个独立任务的调度。③考察任务的 3 个 QoS 约束需求：CPU 处理能力、带宽、资源使用价格。④以实验结果数据为依据，对算法做综合性评价。 实验结果如图 2、图 3 所示。

由实验测试结果（如图 2-图 3 所示）可以得出下面的结论：①当用户在提交任务时，对任务的 CPU 处理能力、带宽、资源使用价格取不同的系数时，对任务的调度影响非常大。图 2 和图 3 中，P(0.8,0.1,0.1)表示用户对 CPU 处理能力要求较高，它的系数是 0.8，而带宽、资源使用价格的 QoS 需求系数分别为 0.1、

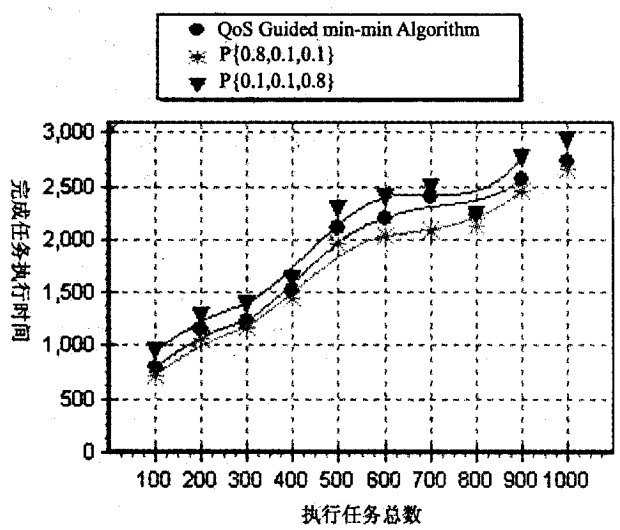

图 2 与 QoS Guided min-min 算法的完成时间比较

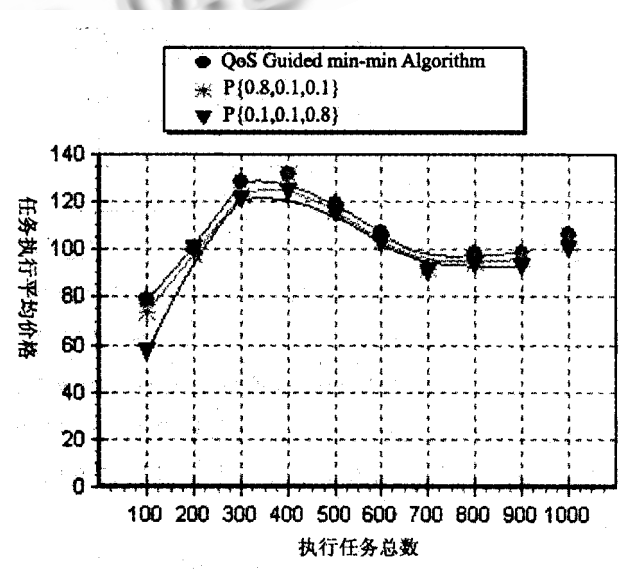

图 3 与 QoS Guided min-min 算法的平均价格比较

0.1。P(0.1,0.1,0.8)表示用户对任务使用资源的价格非常关心，它的系数是 0.8。用户的需求差异，也使任务调度的结果产生了差异。而这种差异完全是符合用户需要的，由蓝、绿两条曲线我们可看出，用户对 CPU 处理能力的重视可以加快任务完成的时间，对资源使用价格的重视可以使任务执行的成本降低，也就是说任务调度的结果很好的反映了用户的需求。②无论从性能还是从满足用户需求的角度来看，本文中提出的任务调度模型都要优于 QoS Guided min-min[5] 调度模型，如图 2 和图 3 所示，QoS 需求系数取 P{0.8,0.1,0.1}时，从时间开销方面来说，本文中提出的算法明显要优于 QoS Guided min-min 算法；QoS 需求系数取 P{0.1,0.1,0.8}时，从任务执行的成本方面来说，本文中提出的算法要优于 QoS Guided min-min 算法。

6 结束语

本文中提出了一种基于用户需求和任务多维 QoS 约束的调度算法，该算法把用户需求和任务 QoS 约束纳入算法考虑范畴，很好的满足了用户需求和适应动态的计算网格环境。通过使用 Gridsim 模拟实现了文中提出的算法，证实有很好的调度效果。在下一步的研究工作中，我们会把该调度模型运用于实际的网格计算环境中，在实际运用中做进一步的检验，同时，突破任务独立性的局限，把任务之间存在数据依赖关系的因素考虑进来，进一步完善算法。

参考文献

1 Wu MY,Shu W,Zhang H.Segmented Min-Min:A static mapping algorithm for meta-tasks on heterogeneous computing systems.// Proc.of the 9th IEEE Heterogeneous Computing Workshop.Washington:IEEE Computer Society Press,2000:375–385.

2 Goldberg D E.Genetic Algorithm in Search,Optimization and Machine Learning.New York:Addison-Wesley,1989:1–55.

3 康立山.非数值并行算法(第 1 册)—模拟退火算法.北京:科学出版社,1997:22–55.

4 Fujimoto N,Hagihara K.A comparison among grid scheduling algorithms for independent coarse-grained tasks.// Proc of the 2004 Symp on Applications and the Internet-Workshops(SAINT 2004).Los Alamitos,CA:IEEE Computer Society Press,2004:674–680.

5 He XS,Sun XH,Gregor von Laszewski.A Qos Guided Scheduling Algorithmn for Grid Computing.Journal of Computer Science and Technology,2003,18(4):442–451.

6 DeFanti T, Foster I, and Kuhfuss T. Overview of the I-WAY: Wide AreaVisual Supercomputing. International Journal of Supercomputing Applications and High Performance Computing,2006,10(2):123–130.