# 一种新型的网格资源协同分配机制

刘　颖

(空军工程大学电讯工程学院，陕西西安 710077)

**摘　要：通过给出网格资源协同分配系统模型，提出一种新型的同步排队策略，用以实现保证网格服务质量(QoS)的资源协同分配。不同于现有方法，该策略不需要资源预置，可以实现资源的超额预定，提高资源利用率，并通过仿真实验进行了验证。**

**关键词：网格计算；服务质量；协同分配；同步队列**

**中图分类号：TP393　　文献标识码：A　　文章编号：1009-3516(2006)05-0081-04**

网格计算正在成为科学计算和协同工作的一个新计算架构，它能协调地理上分布的各类资源为用户提供有效的一致访问和协同工作，以满足虚拟组织(Virtual Organizations VOs)的需要。分布、异构、动态和大规模是网格的主要特征，而提供一定的服务质量(quality of service，QoS)则是网格的主要目标之一。然而，二者本身就具有矛盾性，网格的特征决定了其QoS是难以保证的。而解决网格资源的协同分配(Co-allocation)问题是保证QoS的关键。在网格研究中资源的同步分配常称为协同分配[1]。协同分配问题在网格计算环境中比在传统的分布计算环境中显得更加复杂。在以往的分布式系统中，为了支持多个资源协同工作，通常采用资源预置(Advanced Resource Rervation)的方法[1-2]。但这种方法存在以下缺陷：一是这种模式不允许超额预定，从而导致整个系统利用率降低。二是基于资源预置的方法对用户有严格的时间限制。

## 1　问题描述

### 1.1　系统模型

假设 $t$ 是客户程序需要提交给网格处理的某个任务的集合，$t$ 由 $n$ 个子任务 $S_0, S_1, \cdots, S_n$ 组成，$n \geqslant 0$。考虑网格层调度器把不同的子任务映射到网格中不同的网格节点上的情况。网格层调度器把属于不同任务的子任务分配给特定的网格节点。本地调度器再根据本地的策略控制进一步调度子任务。

一旦子任务被分配给不同的网格节点，本地调度器将分派足够的网格资源去执行每个子任务。因为不同的本地调度器拥有不同的任务、子任务和调度策略，它们的行为也必将不同。对于有协同分配需求的任务，其子任务必须以协同的方式连续执行。其中某些子任务可能因为没有分配到足够的资源而被延迟。运行最快的和运行最慢的子任务之间的时差定义为任务协同分配的相位差。图1给出了分派给不同网格节点的同一个任务的5个子任务的协同分配相位差。阴影条纹表示本地计算机分配的任务的进程。本文所提出的SQ算法的目标是使需要协同分配所有任务的协同分配相位差最小。

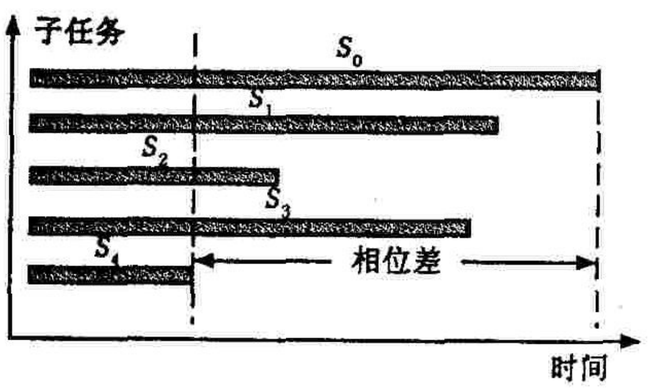


图1　分配相位差图

用 $s_i$ 和 $s_j$ 表示任务 $t$ 的2个子任务，并在调度周期 $q_0$ 触发。用 $r_{si}$ 表示子任务 $s_i$ 的权重，$W_k^{si}$ 表示子任务 $s_i$ 在调度周期 $q_k$ 的完成量。子任务的权重是预先规定的相对价值或优先权，表示子任务重要性的差异。$s_i$ 和 $s_j$ 在任何给定的调度周期 $q_k$ 是同步的，假设子任务 $s_i$ 和 $s_j$ 触发时刻相

**收稿日期**：2005-12-30
**基金项目**：陕西省自然科学基金资助项目(2004F14)
**作者简介**：刘　颖(1980-)，女，江苏淮安人，博士生，主要从事网格资源管理与调度研究.

同,则给出同步状态定义如下:

$$\frac{1}{r_{si}}\sum_{m=0}^{k-1}W_k^{si}-\frac{1}{r_{sj}}\sum_{m=0}^{k-1}W_m^{sj}=0 \tag{1}$$

这是同步的理想化定义,即假设所有子任务都可以进行无穷分解。假设任务 $t$ 有 $l$ 数目的子任务:则 $t$ 在调度周期 $q_k$ 的协同分配相位差 $\Phi_k(t)$ 如下所示:

$$\Phi_k(t)=\max_{i\in[0,l-1]}\left[\sum_{m=0}^{k-1}W_m^{si}/r_{s_i}\right]-\max_{i\in[0,l-1]}\left[\sum_{m=0}^{k-1}W_m^{sj}/r_{s_j}\right] \tag{2}$$

SQ 算法的目标是使需要协同分配的所有任务的协同分配相位差最小。假定 $\Gamma$ 是一组需要协同分配的任务,然后利用 SQ 算法最小化 $\sum_{t\in\Gamma}\Phi(t)$。

### 1.2 QoS 在资源管理中的应用

随着 QoS 机制在网络以及多媒体领域的发展与成熟应用,这一概念也被借鉴到网格系统的任务调度、资源分配等领域中[3]。QoS 是用来表达为用户提供服务的整个系统的各种特性的组合,这些特性构成层次化结构,并针对用户请求的具体应用服务考察其取值,达到用户满意的目的。有关 QoS 的研究集中在:标准的制定;协商协议与方法;根据 QoS 要求的资源分配及调度;维护、监视和控制方法等方面。可以将网格中的不同类型的任务视为 QoS 体系中的用户,将网格给每个任务提供的各种网格资源视为网格为用户提供的服务。

### 1.3 网格资源管理模型

网格资源管理结构为两级式:全球网格资源管理系统(GGRMS)和本地资源管理系统(LRMS)。GGRMS 负责将子任务分配给不同本地资源。因此,GGRMS 需要掌握本地资源的特征和性能,这些特征和性能是受 LRMS 执行的资源管理策略影响的。在网格环境中,通常不同的资源拥有不同的负载管理系统和资源调度策略。在本模型里,本地资源被 LRMS 分成 3 个部分。结点自主性是因为本地资源可以任意的给本地分配选择任何负载管理策略,并且本地分配的粒度也取决于本地策略。然而,在本模型中某一资源本来的调度器被多级本地调度器代替。图 2 是本地调度器中的队列分层结构。在本网格模型中,假定应用程序预先已分解成若干子任务并且子任务间的相对权值是已知的。权值是跟据具体应用情况给定的,并且这里考虑的是相互独立的子任务之间的映射,即子任务之间没有任何通信和数据依赖。

协同分配问题是确保给某任务中所有的子任务分派足够的资源,使其能获得满意的执行进程,这些子任务运行在异构的环境中。一台本地计算机的负载来自于三类任务流即网格本地任务流、网格 QoS 任务流和网格 BE 任务流。网格 QoS 任务流可以被进一步的划分为严格的 QoS 任务流和宽松的 QoS 任务流[4]。网格 QoS 任务和网格 BE 任务由 GGRMS 分配给 LRMS。如图 2 所示不同类型的任务或子任务分配给它们的各自的队列。网格策略和队列管理程序将决定在每个调度周期给这 3 个队列分别分配多少本地资源。图 2 所示的调度器结构用于各本地资源管理系统。选择调度哪一个队列由网格中心调度器决定,在已选择的队列中具体调度哪个任务或子任务将由节点局部调度器决定。

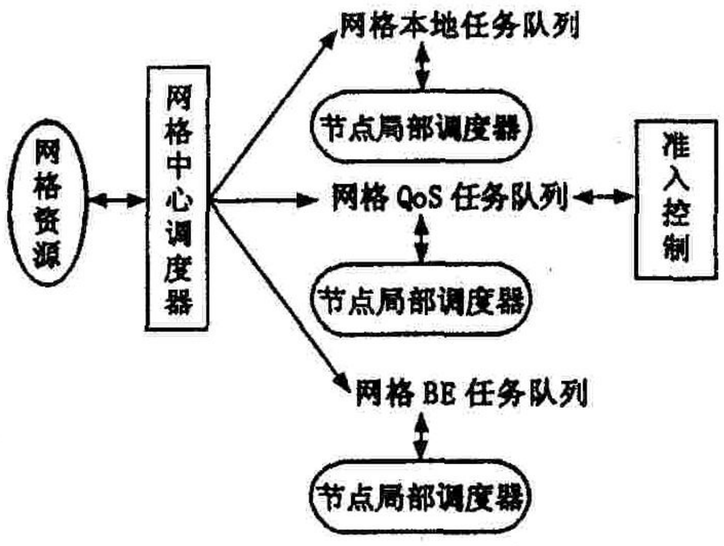


图 2 本地调度器中的队列分层结构图

本地资源会监控映射其上的子任务的进程。这些监控过程是在大于或等于一个调度周期的时间间隔里执行的。如果进程偏离预期值超过一个给定的阈值,本地资源将通知网格协同分配控制器发生偏差并告知偏差值。协同分配控制器接收来自不同的资源的偏差报告并给出必要的校正处理。

各本地资源的进程监控是由实时时钟($R_T$)和虚拟时钟($T_V$)来实施[5]。$T_R$ 时钟是运行在每个资源上的独立进程,它有一个全局统一值。假设不同的资源 $R_T$ 相位差是可计算的且误差可忽略。

假定任务 $t$ 的子任务 $s_j$ 在 $T_R$ 计时器值为 $\tau_0$ 时触发。令 $T_R=T_{RP}=T_V=\tau_0$,$T_{RP}$ 是前一调度周期结束时 $T_R$ 的值。假定一个调度周期分配给 $s_j$ 的 CPU 时间片为 $x$,但实际分配值为 $\bar{x}$。假设一个调度周期的长度为 $q_i$,则时钟值递增 $T'_R$ 及 $T'_V$ 如下:

$$T'_R=T_R+q_i \quad (3)\quad ;\quad T'_V=T_V+\bar{x}(T_R-T_{RP})/x \quad (4)\quad ;\quad T_{RP}=T_R \tag{5}$$

## 2　SQ 协同分配算法

### 2.1　平均虚拟时钟值的选定

接收来自本地计算机严格 QoS 分配子任务的进程信息前,网格控制器会选定一个$\overline{T_V}$值,$\overline{T_V} = 1/n \sum_{i=0}^{n-1} T_V$,$n$ 为 $t$ 的子任务数目,$T_{Vi}$是 $s_i$ 的虚拟时钟值。因此,从本质上讲,$\overline{T_V}$是 $t$ 的 $n$ 个子任务的平均虚拟时钟值。

### 2.2　协同分配相位差的检测

对于严格 QoS 任务,客户程序提供两种 QoS 属性:总体偏差值和异步性。总体偏差值是指任务 $t$ 能接受其子任务的延迟或加速。异步性是指任务 $t$ 可以允许的协同分配相位差,计算如下:

$$\text{Asyn} = T_{Vf} - T_{Vs} \tag{6}$$

$T_{Vf}$和 $T_{Vs}$分别是任务 $t$ 的所有子任务的最快虚拟时钟值和最慢虚拟时钟值。

### 2.3　协同分配相位差的校正

通过总体误差窗和异步窗口实施协同分配相位差的校正。总体误差窗对某任务中的所有子任务的总体进程实施 QoS 约束,可以防止任务加速或减速(没有异步性)超过允许极限;异步窗口用来实施可以接受的异步性。校正过程是,周期地查看网格协同分配控制器发送的任务 $t$ 全部子任务的报告。对于每个 $t$,计算$\overline{T_V}$值并检测它是否在总体误差窗内部。如果在总体误差窗外部,则通过增加或降低子任务的速度进行同步。如果$\overline{T_V}$值在总体误差窗区域之内,将检测异步窗口。如果最慢或最快子任务在窗口外部,则需要减速最快的子任务或加速最慢的子任务。任务 $s_i$ 的减速因子 $SF_i^d$给出如下:

$$SF_i^d = [T_{Vsi} - T_{Rsi}] r_{si}/100 \tag{7}$$

子任务 $s_i$ 传递权值 $SF_i^d$ 给初始化的虚拟子任务。$s_i$ 的权值相对权重由 $r_{si}$减去 $SF_i^d$。同理,如果要加速子任务则加上 $SF_i^u$。加速时,权值从一个虚拟子任务分配到子任务,假定虚拟子任务足够大。当减速时,如果没有虚拟的子任务,将生成一个权值等于 $SF_i^d$ 的虚拟子任务。该任务将被插入网格 QoS 任务队列中消耗 $s_i$ 等于 $SF_i^d$ 额外权值。

## 3　仿真实验和讨论

仿真研究的首要目的是评价网格环境中,SQ 算法限制某个任务的子任务之间协同分配相位差的效果。本实验通过增减网格节点来度量工作量。假定某一任务中子任务的数目和网格节点成正比。当子任务数目增加时,协同分配问题的复杂性随之提高。通过将 SQ 算法和基于严格准入控制算法作比较,来评价当不同的参数变化时这两种算法的性能。协同分配相位差最小化是 SQ 算法的首要性能指标。

在仿真中,每台本地计算机的 CPU,按权重 0.3、0.5 和 0.2 分别分配给网格本地任务、网格 QoS 任务和网格 BE 任务。在基于严格准入控制算法中,一旦容量饱和,网格 QoS 队列的准入控制器就会阻止进一步接收任务,防止超额预定,而 SQ 允许 5% 超额预定。

网格拓扑结构由一组相互间没有通信的网格节点组成。网格拓扑在(3,6,9,12)之间变化。通过 3 组任务发生器产生相应的 3 种类型任务流。任务发生器产生范围在(1 500 - 3 000)之间的随机任务,该过程服从泊松分布,参数即到达时间 λ 为(10, 100, 200,500 )。对于严格 QoS 任务,网格 QoS 发生器随机的服从均匀分布,其参数——异步性和总体偏差为(100 - 500)s。此外,一个网格任务由若干子任务组成,子任务数目随机地服从均匀分布(0,本地资源数目)。子任务执行时间服从均匀分布(1500,2500)。本地计算机 CPU 在(100 - 600)MHz 范围内随机选择,一个网格层的 CPU 标准假定为 1 GHz。每个任务通过在( 1 - 4) 范围中选择总任务权重和子任务 $s_i$ 相对权重 $r_{si}$实现资源请求。

图 3 和图 4 所示为平均协同分配相位差随着网格节点数目增加的变化情况。图 3 为 λ = 10 s 的超负载情况下的变化曲线,图 4 为 λ = 500 s 时的低负载情况下的变化曲线。当节点数目增加时,协同分配工作量也提高。从仿真结果

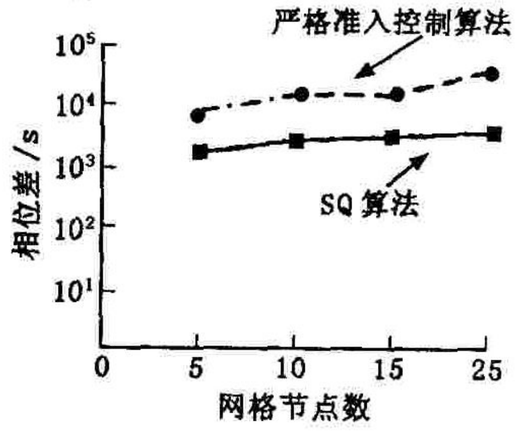

图3　λ = 10 s 时相位差变化曲线图

可以看出 SQ 优于基于严格准入控制的 QoS 方案。当 λ 提高时,平均协同分配相位差增大,这是因为:①同时使用这两种方案能放宽准入控制门限,增加准入的任务数目;②考察平均协同分配相位差只计算成功执行的任务。

输入控制过程规定到达实际系统的请求流在系统负载量之内。然而,与低负荷的情况相比,系统超负荷时要完成相当大数目的任务。当 λ 提高时,基于严格准入控制算法的协同分配相位差增加。在不同资源数目的低负载情况下,相位差增加将引起给定任务中的子任务速度提高。这样又会增加协同分配相位差。对于 SQ 算法,当 λ 提高时,协同分配相位差也增加。这可以通过研究 SQ 的运用原理来解释。SQ 通过在子任务重新调整权重来减少协同分配相位差,也可以添加一个虚拟的子任务来消耗额外的资源,如果这些资源被分配给实际的子任务将会增加协同分配相位差。这是一个反馈程序,需要若干调度循环周期才能达到一个稳定的工作点。当 λ = 10 s 时,子任务在同一序列到达,系统重新配置这批子任务。当 λ = 500 s 时,子任务以稀疏的方式到达,系统处于持续的重新配置状态。只有当系统已经到达一个稳定状态时协同分配相位差为最小值,这就说明为什么 SQ 在低负载情况下执行性能不是十分理想。

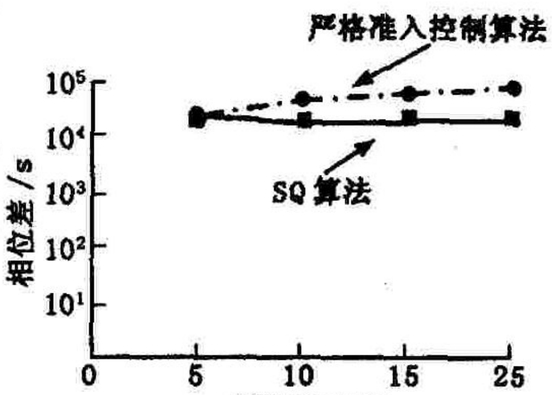

图4 λ=500 s 时相位差变化曲线图

## 4 结论

对网格资源协同分配研究目前还处于研究探索阶段,本文在深入分析网格应用环境特点的基础上,提出了一个新型的协同分配策略——SQ 算法。与现有的协同分配方法不同,SQ 不需要目标资源的资源预置,并具有下列优点:①面向存储的 QoS 性能, SQ 能记忆每个子任务在前面的调度循环周期总完成量;②同一个任务中的子任务执行时间总量相同;③对于协同分配不需要资源预置;④能提供超额预定资源的能力。

**参考文献:**

[1] Czajkowski C, Foster I, Karonis N, et al. A resource Management Architecture for Metacomputing Systems[A]. Springer – Verlag LNCS 1459. 4th Workshop on Job Scheduling Strategiesfor Parallel Processing[C]. 1998, 62 – 82.

[2] Ferrari D, Gupta A, Ventre G. Distributed Advance Reservation of Real – Time Connections [J]. ACM/Springer – Verlag Journal on Multimedia Systems , 1997, 5(3) : 187 – 198.

[3] Rajkumar. Lee C, Lehoczky J. Practical Solutions for QoS – Based Resource Allocation Problems[A]. Proceedings of the IEEE RealTime System Symposium[C]. 1998, 112 – 137.

[4] Yau D. ARC – H: Uniform CPU Scheduling for Heterogeneous Services[J]. International Conference on Multimedia Computing and System, 1999, 2(6) : 127 – 132.

[5] Zhang L. A new Traffic Control Algorithm for Packet Switching Networks[J]. IEEE Transaction on Computer Systems, 1991, 9(2) : 101 – 124.

[6] Yang Hailan , Wu Gongyi , JZhang ianzhong . On – Demand Resource Allocation for Service Level Guarantee in Grid Environment [A]. The 4th International Conference on Grid and Cooperative Computing[C]. 2005, 678 – 690.

[7] Farag Azzedin, Muthucumaru Mahesw Aaan, Neil Arnason , A Synchronous Co – Allocation Mechanism for Grid Computing Systems[J]. Cluster Computing, 2004, 7(5) : 39 – 49.

(编辑:门向生)

A Novel Resource Co -allocation Mechanism in Grids

LIU Ying

(The Communication Engineering Institute, Air Force Engineering University, Xi'an, Shaanxi 710077, China)

Abstract: Based on grid resources co -allocation systemmodel, a novel strategy called synchronous queuing (SQ) is proposed for implementing co - allocation with QoS assurances in Grids. Unlike the existing approaches, SQ does not require advanced reservation capabilities in the resources. The simulation studies performed to evaluate SQ indicate that it can improve resource utilization rate.

Key words: grid computing ; quality of service; co - allocation; synchronization queue