ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.11, November 2006, pp.2314−2323
DOI: 10.1360/jos172314

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于 Cache 的网格任务反馈调度方法*

袁平鹏[1,2,3+], 曹文治[1,2,3], 邝 坪[1,2,3]

[1](服务计算技术及系统教育部重点实验室,湖北 武汉 430074)

[2](集群与网格计算湖北省重点实验室,湖北 武汉 430074)

[3](华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

A Cache Based Feedback Grid Scheduling Approach

YUAN Ping-Peng[1,2,3+], CAO Wen-Zhi[1,2,3], KUANG Ping[1,2,3]

[1](Service Computing Technology and System Key Laboratory of the MOE, Wuhan 430074, China)

[2](Cluster and Grid Computing Key Laboratory of Hubei Province, Wuhan 430074, China)

[3](School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

+ Corresponding author: Phn: +86-27-87543529, E-mail: ppyuan@mail.hust.edu.cn, http://grid.hust.edu.cn/ppyuan

Yuan PP, Cao WZ, Kuang P. A cache based feedback grid scheduling approach. *Journal of Software*, 2006, 17(11):2314−2323. http://www.jos.org.cn/1000-9825/17/2314.htm

Abstract: Grid scheduling which aims at improving resource utilization and grid application performance is a key concern in grid. Currently, much research can be found about grid scheduling and some algorithms on it were proposed. However, since grid resources are autonomic, distributed and their status change over time, those scheduling algorithms did not fit for the cases well. In this paper, a cache based feedback grid scheduling (CBFS) approach is presented to capture the dynamics and impact of simultaneously co-allocated tasks in a grid. In this approach, grid scheduler utilizes recent resource performance data, such as recent task submitting time and execution time of task which are kept in cache and a feedback approach to engineer load balancing across multiple grid resources. After comparing this dynamic grid scheduling approach with previous research, it is found that CBFS is more generous than other scheduling approaches. Experimental results demonstrate that this approach diminishes latency and contributes to the overall grid load balancing, which significantly improves resource utilization and response time of tasks.

Key words: grid; scheduling; cache; feedback

摘 要: 网格调度的目标提高网格资源的利用率、改善网格应用的性能，它是网格中需着力解决的问题之一.目前,围绕着网格中的任务调度算法,国内外已做了大量的研究工作,先后提出了各种调度算法.但是,这些调度

* Supported by the National Natural Science Foundation of China under Grant No.90412010 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2004AA104280 (国家高技术研究发展计划(863)); the ChinaGrid Project of the Ministry of Education of China (国家教育部中国教育科研网格计划); the China Next Generation Internet under Grant No. CNGI-04-15-7A (中国下一代互联网示范工程); the Key Technologies R&D Program of Wuhan, China under Grant No.20061002032 (武汉市科技攻关项目)

Received 2006-06-10; Accepted 2006-08-25

算法不能很好地适应网格环境下的自治性、动态性、分布性等特征.针对目前网格调度机制存在的问题,提出了一种动态的网格调度技术——基于 Cache 的反馈调度方法(cache based feedback scheduling,简称 CBFS).该调度方法依据 Cache 中所存放的最近访问过的资源信息,如最近一次请求提交时间、任务完成时间等信息进行反馈调度,将任务提交给负载较小或性能较优的资源来完成.实验结果表明,CBFS 方法不但可以有效减少不必要的延迟,而且在任务响应时间的平滑性、任务的吞吐率及任务在调度器等待调度的时间方面比随机调度等传统算法要好.

关键词: 网格;调度;cache;反馈
中图法分类号: TP393 文献标识码: A

网格的根本目标是协调资源的共享.在网格环境中存在众多资源,这些资源的特性及资源所处的环境各不相同.网格的根本任务之一是根据各资源节点的状态、网络通信性能等参数,把不同的任务以合理的方式分配到相应的资源结点去完成,这也就是所谓的任务调度.任务调度在操作系统等领域进行过很多研究,但由于网格环境中资源的多样性、自治性和动态性,使得网格环境下的任务调度比传统环境下的调度要复杂得多.虽然随着网格技术的发展以及资源服务化,出现了基于服务的网格标准[1,2],这在某种程度上统一了资源的呈现方式,简化了任务调度的接口,但网格环境中资源的自治性和动态性依然存在.而且,由于服务所依赖的协议层次多,实际上反而增加了网格任务的响应时间[3].这就使得在高延迟、高动态、高自治的网格环境下进行调度、提高网格的吞吐率成为一个困难问题.

围绕着网格中的任务调度,国内外已做了许多研究工作,先后提出了各种调度算法.这些算法按照调度策略可以分为动态调度(dynamic scheduling)和静态调度(static scheduling)两种[4].动态调度是任务一到来就加以映射;而静态调度则是把任务收集起来,等映射事件到来后才对这些任务进行集中映射.按网格调度的度量依据主要有基于时间、经济[5]以及其他度量指标(如公平性、稳定性、健壮性等)的调度.此外,还有将多种调度准则综合在一起的调度方法,如 QoS(quality of service)[6−10]等.根据调度算法所应用于任务调度的阶段,调度算法可分为应用于预处理阶段的算法和调度及重调度阶段算法,张伟哲等人[11]提出的对资源进行分类与聚类算法就是处于预处理阶段.由于本文主要是关于网格中动态任务调度方法,因此,下面主要分析相关研究.

静态调度算法主要有 Min-Min[12−15],Max-Min[12−14],Suffrage[12],XSuffrage[16],TCR(transfer computation ratio)[17]等.静态调度算法需要花费大量时间计算任务调度表,算法缺少灵活性.任何变化,如任务添加、删除或任务特征变化,都需要重新计算调度表.而且静态调度算法每隔一定周期进行一次调度,因此,越早到达的任务等待的时间越长,从而使得任务的响应时间过长.常见的动态调度算法有 MET(minimum execution time)[12],MCT(minimum completion time)[12],SA(switching algorithm)[12,16],KPB(K-percent best),OLB(opportunistic load balancing)[12].相比之下,动态调度算法的环境适应性好,在多种环境下操作性能良好,算法灵活.因此,动态调度算法更适合于网格环境.许多网格中间件,如 ChinaGrid 支撑平台 CGSP(ChinaGrid support platform),VEGA[18],CROWN(China research and development environment over wider-area network)[8],Globus Toolkit 等,也采用的是一种动态调度模式.

但目前的在线调度算法在如下几个方面未作有效考虑:首先,目前的调度算法均要求能够控制网格节点,这在网格环境下是很难做到的;其次,网格调度器在进行任务调度时假设资源的状态如性能、可用性等不作改变,这一点并不符合网格的实际情况.实际上,网格资源的性能与可用性均处于变化之中;第三,假设网格只有一个入口点,调度器能够获取网格上所有节点的状态或能够控制所有节点上的调度策略;最后,大多数网格调度算法未考虑网络通信延迟.这些因素使得目前的网格调度算法在调度时效性和适用性方面存在一定的缺陷.针对目前网格调度机制存在的几个问题,本文研究一种动态的网格调度技术——基于 Cache 的反馈调度方法(cache based feedback scheduling,简称 CBFS).

1 网格环境下的调度模型

在网格环境中存在很多资源(或服务),这些资源有的是同一类别的,有的是不同类别的.网格调度器可能会接受到一系列用户请求,即任务.网格调度器将用户请求分解成一系列可执行的子任务,然后将这些可执行的子任务透明地调度到具体的资源上.我们把每个任务广义地当作一个可执行服务来看待.各任务之间的关系既可以是串行关系,也可以是并行关系.由于并行任务的执行时间由最长任务来决定,为了简化讨论,我们将任务之间是并行关系的各任务视为一个执行时间最长的活动.在网格环境下,存在多种多样的资源,同一类型的资源可能存在多个,但每一资源的性能都存在差别.因此,一个任务可以有多种调度方案,每一调度均有一个执行时间值.网格环境下的任务调度的实质就是将子任务以合理的方式调度到 m 个资源节点 $NS=\{S_1,S_2,\ldots,S_m\}$的子集上去,目的是得到尽可能小的总执行时间.对于用户来说,他不关心其请求具体由哪些资源完成,而只关心他的请求是否能够尽快返回.对于网格调度器,它负责查找信息服务、获取服务信息,并调度合适的资源来完成用户的请求.如图 1 所示.

从图 1 可以看出,任务的响应时间包括本地执行时间、与信息服务通信所导致的时间、任务提交给资源节点到资源节点执行完任务返回结果的时间等.因此,网格环境下一个任务的响应时间可以定义为

$$TR = T^u + T^w + \sum_{i=0}^{\text{任务的子任务数}} (T_i^G + \alpha T_i^{G\to IS} + T_i^{IS} + \alpha T_i^{G\to S} + T_i^S) \quad (1)$$

其中,G 代表网格调度器,IS 代表信息服务,S 代表执行子任务的远程资源,T^u 代表用户请求提交到调度器的时间以及处理结果从调度器返回给用户的时间之和,T^w 是在调度器上的等待时间,T_i^G 代表任务的指令 i 在机器上执行所需的时间,$T_i^{G\to S}$ 代表第 i 条指令从网格调度器 G 提交到 S 所经过的网络时延,T_i^S 一般包括在队列中的等待时间以及处理时间.此外,由于网格调度器调度请求到某一服务机时,该服务机可能不可用.这时,调度器需要重新调度.因此,一次成功的调度可能涉及多个不成功的调度,这些不成功的调度产生的时间消耗主要在网络延迟上.为了简单起见,在成功调度的网络延迟上乘以一个系数α.α一般大于等于 1.

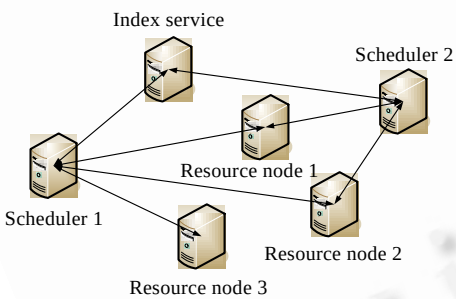

Fig.1 Grid scheduling

图 1 网格调度

在公式(1)中,$\alpha T_i^{G\to IS} + T_i^{IS}$ 是与信息服务通信所导致的时间开销.如果网格调度器不需要到信息服务器查找服务信息,那么这项开销可以大为减少.实际上,根据临近访问原则,最近刚访问的服务在不久的将来会被再次访问到.因此,若将最近访问到的服务信息保存下来,可以有效地减少与信息服务的交互次数.

在公式(1)中,$\alpha T_i^{G\to S} + T_i^S$ 是指从任务提交给资源节点到资源节点执行完任务返回结果的时间.若资源忙或者资源失效,那么任务不得不等待较长时间或重新进行调度,因此,这部分时间均包含服务失败及请求等待响应的等待时间.而且,服务上若有较多的请求在处理或等待处理,则对服务的性能将有所影响,造成服务性能的下降,使得请求的响应时间延长.因此,为了缩短任务的完成时间,应尽可能地将任务调度到响应时间最优的节点上.但由于网格资源节点是自治的,网格调度器获知资源节点的有效方式之一是根据该资源节点处理前面任务的响应时间,即用反馈方法来推断资源状态,控制提交任务给资源的时间间隔.

基于以上分析,本文提出了基于 Cache 的反馈调度方法.本文拟考虑利用网格调度器中调用远程任务的各种历史记录来提高网格的响应时间.具体地说,网格调度器利用 Cache 记录减少访问信息服务的次数及频率;其次,利用 Cache 中记录的子任务提交时间间隔及子任务的响应时间等值判断是否该向一个服务提交新的请求.下面将详细介绍基于 Cache 的反馈调度方法.

2 基于 Cache 的反馈调度

在网格环境中,资源节点是自治的,网格调度器不能够对资源节点实现完全控制.由于网络通信的开销,如果网格调度器将一个子任务调度到一个性能下降的资源节点上执行,那么,子任务可能会等待很长时间才能得到处理.极端情况下,延迟过大,网格调度器端认为该服务不可用,从而不得不重新调度该子任务到新的资源节点上.这都会使得任务的响应时间增大.此外,任务调度需要访问信息服务器获取资源的信息,但由于信息服务器中的服务信息更新周期长,而且访问信息服务器会产生延迟,因此,网格调度器单纯地根据信息服务中的信息进行调度,可能导致某些服务负载过大,而另外一些服务利用率不高.之所以存在这些问题,主要是由于网格调度器没有有效地根据资源节点的状态信息,而是根据信息服务中的非实时服务信息来进行任务调度.事实上,网格调度器可以根据任务的处理情况推断资源节点的当前状态,由此来决定后面的任务如何调度.这种调度方法就是反馈调度.

由于信息服务器不可能实时获取服务的状态,而且服务均是自治的,因此,本文反馈调度所依赖的信息有限.为了较准确地推测服务的状态,本反馈调度方法以前次提交任务给服务的时间间隔、服务最近一次的响应时间、已提交给资源的子任务数等相关信息作为下次调度的依据.为了记录这些信息,在网格调度器中设置 Cache.通过设置 Cache,减少访问信息服务器的次数,并依据 Cache 中的信息推断资源节点的状态,从而尽可能地避免服务请求提交到性能差或无效资源节点,以确保尽可能地将服务请求交给当前性能好的服务机运行.这就是基于 Cache 的反馈调度方法的基本思想.在该调度方法里,最关键的是反馈调度以及 Cache 内容的更新.下面分别介绍这两点.为了后面叙述方便,下面先定义几个概念.

定义 **1**. 资源是一种能够提供服务的实体,可定义为六元组(*ID*,*RC*,*TR*,*TV*,*TS*,*NL*),其中 *ID* 代表资源的标识,*RC* 是等待该资源提供服务的任务个数,*TR* 是资源的最近响应时间,*TV* 为最近两次提交任务给资源的时间间隔,*TS* 为最近一次提交任务给资源的时间,*NL* 代表将任务提交到资源的网络延迟.

定义 **2**. 任务是一组可在资源上执行的子任务集合,可定义为 *T*=(*ST*,*TQ*).其中:*ST* 为任务的状态,状态包括可执行、等待等;*TQ* 为任务中的子任务序列,一系列任务又构成请求队列,由调度器进行调度来完成请求队列中的任务.

2.1 反馈调度

假设网格调度器提交任务给资源的提交时间为 *TS*,提交到同一资源的相邻提交时间间隔为 *TV*,任务处理完返回给调度器的时间为 *TC*.如果服务的执行时间大于 *TV*,那么,后续到达该服务的任务将会形成队列.此时,请求需在队列中等待一段时间,以便服务处理完前面的任务后再来响应该任务.如果服务的执行时间小于 *TV*,那么,服务处理完所有任务,等待下一任务的到来.此时,服务处在间断空闲状态.因此,理想情况下,一个服务处理完一个任务以后,后面一个任务此时恰好到达,这时任务不必等待,而服务利用率同时也达到最高,这时,任务的等待时间最短.但等待时间最短,并不能说明处理时间最短.因此,调度任务时不但要考虑到等待时间,而且也要考虑到处理时间.

为此,反馈调度主要依据提交任务到资源的时间间隔以及资源的响应时间来推断资源的当前状态,从而决定是否需要向该资源提交任务.现推导它们之间的关系.假设提交一系列任务到服务 S_i 的提交时间序列为 $\{TS_1^{S_i},TS_2^{S_i},...,TS_j^{S_i},TS_{j+1}^{S_i},...\}$, 相应的任务完成时间队列为 $\{TC_1^{S_i},TC_2^{S_i},...,TC_j^{S_i},TC_{j+1}^{S_i},...\}$. 根据该序列,可得到提交给该服务的第 j 个任务的响应时间 $TR_j^{S_i}=TC_j^{S_i}-TS_j^{S_i}$,与第 j+1 个任务的提交时间间隔为 $TV_j^{S_i}=TS_{j+1}^{S_i}-TS_j^{S_i}$.按照前面的分析,$\Delta T=TC_j^{S_i}-TS_{j+1}^{S_i}-\frac{NL_j^{S_i}}{2}$,理想值应该为 0(其中 $NL_j^{S_i}$ 是调度器到节点 S_i 的双向网络延迟).此时,$TV_j^{S_i}$ 应该为第 j 个任务的执行时间 *ET*.据此可得:

$$\frac{TR_j^{S_i}}{TV_j^{S_i}}=\frac{TC_j^{S_i}-TS_j^{S_i}}{TS_{j+1}^{S_i}-TS_j^{S_i}}=1+\frac{TC_j^{S_i}-TS_{j+1}^{S_i}}{TS_{j+1}^{S_i}-TS_j^{S_i}}=1+\frac{\Delta T+\frac{NL_j^{S_i}}{2}}{TS_{j+1}^{S_i}-TS_j^{S_i}}=1+\frac{NL_j^{S_i}}{2TV_j^{S_i}} \tag{2}$$

令

$$\sigma^{S_i}=\frac{NL^{S_i}}{2TV^{S_i}} \tag{3}$$

可得 $\frac{TR^{S_i}}{TV^{S_i}}=1+\sigma^{S_i}$.

σ^{S_i} 可以作为判断资源状态的指标.在网格环境中,资源也有可能不能满足要求.此时,网格调度器不得不更加密集地提交任务给资源,资源的 $\frac{TR_j}{TV_j}$ 会偏离理想值.为了解决这个问题,允许 $\frac{TR_j}{TV_j}$ 有一个变动范围.该变动范围取为

$$\left|\frac{TR}{TV}-1\right|=(RC+1)\times\sigma \tag{4}$$

在计算式(4)过程中,需要估计本次任务的响应时间.本方法采用下述公式来估计本次任务的响应时间 TR,

$$TR=(1-\gamma)\overline{TR}+\gamma TR_r \tag{5}$$

其中:ET_c 是本次任务理想情况下的执行时间;$\overline{ET}$ 是任务的平均执行时间;ET_r 是最近一次任务的执行时间;TR_r 是资源的最近一次实际响应时间;$\overline{TR}$ 是资源的平均响应时间;γ是系数,$0<\gamma\leq 1$,当γ为 1 时,只以最近一次响应时间为依据;当γ为 0 时,则只考虑响应时间的平均值.γ值可取为 0.99.

下面给出反馈调度方法的描述.

算法 **1**. 反馈调度.

步骤:

1. 若请求处理完毕,计算相应资源的 TR,置 $RC=RC-1$ 及后继请求为可执行,调用 UpdateCache 算法;
2. 从标志为可执行的请求队列中取一个请求 Request,且该请求中的后续任务可执行;
3. 从该请求中提取最前面的一个任务;
4. 对该任务做:
5. $R=getResources(Request)$; //从 Cache 中选取与该任务相匹配的资源集合 R;
6. if $R==NULL${$UpdateCache(Request);R=getResources(Request);$}
7. $Sort(R);r=get\text{Min}TR()$;
8. 候选资源 $rs=r$;
9. $\Delta TR_j^r=TR_{j+1}^r-TR_j^r$;
10. if $\left|\frac{TR^r}{TV^r}-1\right|>(RC^r+1)\times\sigma^r$ and $\Delta TR_j^r>0$ {
11. for i in R, do {
12. $\Delta TR_{ri}=TR_{j+1}^r-TR_{j+1}^i$; $\Delta TR_j^i=TR_{j+1}^i-TR_j^i$;
13. if $(\Delta TR_{ri}-\Delta TR_j^r+\Delta TR_j^i)\leq 0$ and $\left|\frac{TR^i}{TV^i}-1\right|\leq(RC^r+1)\times\sigma^i$ 候选资源 $rs=i$;
14. }
15. }
16. 将该任务提交给资源 rs,记录提交时间 TS^{rs},并更新该资源的最新提交时间为 TS^{rs},$RS^{rs}=RC^{rs}+1$;
17. 返回步骤 1.

2.2 Cache的设置及更新

Cache 中存放最近访问过的服务信息,如资源名、资源前一次响应时间 LRT(last response time)、资源平均响应时间 ART(average response time)、上一次提交时间等信息.这些信息是反馈调度的依据,帮助调度器正确地

决策将任务提交给哪些资源来完成.由于 Cache 的大小不能无限制地增大,太大的 Cache 导致查找 Cache 的时间变长.因此,Cache 中的信息应能反映网格目前最近使用资源的状态信息或将要使用资源的状态信息.网格是动态的,随时可能有性能更优的服务注册到信息服务中,也有可能随着资源负载变化及其他因素引起资源状态改变,使得资源性能下降,甚至不可用.这就要求及时更新 Cache 中的资源信息,使得 Cache 中的服务信息能够反映网格资源的最新信息,否则,任务调度就失去了正确的依据.为此,Cache 中的信息需要进行更新.

Cache 中的信息更新包括两个方面:一方面是 Cache 中已有资源信息的更新;另一方面是将新的性能更优的网格资源信息加入到 Cache 中,或根据信息服务中的信息更新 Cache 中的信息.对于最近一段时间之内访问过的 Cache 中的资源,Cache 中资源信息的更新可根据最近一次资源的执行任务情况来更新.网格环境是一个动态、自治环境,资源可能会由于配置、程序升级等方面的变化导致资源性能的改变;此外,新的资源也有可能加入到网格环境中.这都使得网格资源处于动态变化之中.为了适应这种变化,网格调度器必须每隔一定时间访问信息服务,获取新的网格资源信息来更新 Cache.考虑到信息服务的负载以及对网格任务调度的影响,访问信息服务不能过于频繁,也不能太稀疏.从信息服务中获取的新网格资源的性能并不一定优于 Cache 中记录的资源性能,只有性能较优的网格资源才能加入到 Cache 中.比较资源的性能,必须基于统一的观察点.信息服务中所声明的网格资源性能度量观察点可能不是网格调度器,与以网格调度器作为观察点所度量的性能有一定的偏差.此外,考虑到网格服务初次启动时的开销较大.因此,对于网格调度器来说,网格资源的实际响应时间会与所声明的性能有所不同,需要对声明性能进行修正.下面给出相对于网格调度器的网格资源性能的修正公式.

$$TR_{Cache}=TR_S+NL_n+SL \tag{6}$$

其中:TR_S 是资源的声明性能,该参数从信息服务器获得;NL_n 是候补资源到网格调度器的双向网络延迟;SL 是启动延迟.

将从信息服务中获取的新资源性能进行修正后,再与 Cache 中已有资源的性能进行比较,确定新资源是否加入到 Cache 中.为了防止将过时的信息更新 Cache,每次更新都需要记载最近一次更新资源时间 RUT(recent update time).更新后的 Cache 中资源需要根据性能及最近访问时间进行排序,以便网格调度器能够有效地选取资源.如果 Cache 空间不够,需要删除 Cache 中那些在一定时间内未使用的资源信息.下面给出 Cache 更新算法.

算法 **2**. UpdateCache.

步骤:

1. if 一个资源处理完请求返回到调度器;
2. 计算 Cache 中相应资源的 *ART*,并更新 *LRT* 及 *RUT* 等;
3. if 计时器时间到或者调度器未能从 Cache 中找到所需资源;
4. 访问信息服务器,获取 Cache 中相应资源的同一类资源,并计算响应时间;
5. 如果某一资源在两倍的计时器时间间隔内未使用过,且 Cache 空间不够,则将相应的资源条目从 Cache 中删除;
6. 设置计时器中断时间间隔.

3 实验与结果分析

为了验证基于 Cache 的反馈调度算法的有效性,我们在 ChinaGrid 支撑平台 CGSP 实现了 CBFS,并与 CGSP 的调度策略进行比较.比较的指标主要有:任务的响应时间、所有任务响应时间之和、执行时间之和以及请求在调度器的处理时间之和.

单个任务的响应时间可以从微观上比较不同调度算法的调度性能,但不能很好地反映调度算法对网格处理大量任务时的性能改善.为此,本实验引入了所有任务的响应时间之和.所有任务的响应时间之和是指所有任务从任务提交到网格调度器至任务处理完返回的时间差之和,即

$$\sum_{j=0}^{\text{任务数}} TR_j = \sum_{j=0}^{\text{任务数}} (T_j^w + \sum_{i=0}^{\text{任务的子任务数}} (T_{ji}^G + \alpha T_{ji}^{G\to IS} + T_{ji}^{IS} + \alpha T_{ji}^{G\to S} + T_{ji}^S)),$$

它反映了网格的吞吐率.执行时间之和则统计所有任务从网格调度器提交到资源及从资源返回的时间差,即

$$\sum_{j=0}^{\text{任务数}} ET_j = \sum_{j=0}^{\text{任务数}} \left(\sum_{i=0}^{\text{任务的子任务数}} (T_{ji}^{G\to S} + T_{ji}^{S}) \right).$$

该指标主要度量调度器将任务提交到性能较优资源的情况.两者的差值反映了所有任务在等待调度器调度时所消耗的总时间,该时间包括访问信息服务时间及等待时间.

3.1 实验设计

本实验通过发送大量不同服务请求给调度器,测试调度器任务调度方法对任务响应时间的影响.在本实验中采用了 5 台计算机(5 台计算机的性能指标见表 1),并开发了 100 种网格服务,这 100 种服务的处理时间均不同.将 100 种网格服务分别部署在 3 台计算机 A,B,C 上,也即每台计算机上均有 3 种服务.由于机器 A,B,C 的性能不同,100 种服务中的同种服务在 3 台机器上性能也不同.经测试,服务 1 在机器 A,B,C 轻负载时的平均响应时间分别为 2 245ms,2 052ms,1 047ms.因此,100 种不同服务和 3 个不同计算机的组合,共形成了 300 种不同性能的服务.在这 300 种性能不同的服务中,响应时间最小的是 762ms,响应时间最大的是 4 002ms.在另外两台计算机上分别部署网格调度器及信息服务器.这 5 台计算机上分别部署 CGSP.部署有网格调度器的机器视测试对象的不同,确定是否加上基于 Cache 的反馈调度机制.为了模拟客户提交请求,在调度服务器上用 4 个线程模拟 4 个客户端在随机的时间间隔内发出 100 个随机的服务请求.

Table 1 Performance of experimental computers

表 **1** 实验机器性能

Computer	CPU	Memory (MB)
Scheduler	Intel Pentium Mobile CPU 1.6GHz	768MB
Index server	Intel Pentium MMX 166MHz	48MB
Server A	Intel Pentium III CPU 938MHz	256MB
Server B	Intel Celeron 1.7GHz	256MB
Server C	AMD Athlon 1921MHz	512MB

3.2 实验步骤和结果分析

实验步骤如下:首先,测试单个服务在大量请求下的性能变化;其次,将 100 种网格服务、信息服务以及网格中间件分别部署到 5 台计算机上,客户机的 4 个线程发送随机的 400 个请求给调度器,测试 CGSP 的调度性能及基于 Cache 反馈调度机制的调度性能.目前,CGSP 的上层调度采用了如随机、轮询等调度策略.由于轮询要经常查询服务的状态,因此,需 sv 确定一个合理的轮询时间间隔,否则,调度性能可能会下降.但轮询时间间隔依赖于网络延迟,在广域网环境中不易确定.另外,即使得到一个合理的轮询时间间隔,轮询策略无论对调度器还是对服务来说开销 bn 都比较大.为了公平起见,在文中主要比较随机调度.

如图 2 所示为 50 个请求提交给一个服务以后的性能变化.图中曲线是请求的响应时间与轻负载时服务的响应时间比值.从图中可以看出:重负载时服务的性能较之轻负载时性能有较大幅度的降低,在曲线头尾部分变化较大:头部变化较大是因为请求到达有一定的时间差,早到达任务的响应时间受影响较小;曲线尾部呈下降趋势的原因是大部分任务已完成,服务的负载降低,因此响应时间缩短.从该图可知,服务性能随着服务驻留机器的负载变化,性能急剧下降.因此,静态的服务信息不能作为调度依据,只有最新的服务状态才能作为调度依据.这就需要根据目前状态推断服务未来的状态,以方便调度器作出较优的决策.

在实验的第 2 步,4 个模拟客户端分别提交 100 个请求给网格调度器,网格调度处理时间分别如表 2 和图 3 所示.表 2 列举了无 CBFS 调度机制的 CGSP 与有 CBFS 调度机制的任务响应时间之和、任务的执行时间之和及调度器处理请求的时间之和.表中还分别列举了两者之间的差值及差值与随机调度性能的百分比.从表 2 中可以看出,采用 CBFS 调度机制以后,任务的响应时间比无 CBFS 机制的任务响应时间大为提高,高出 32%以上.其所以产生这种结果,是因为实验过程中 3 台计算机上均部署了大量服务实例,400 个请求在较短时间内随机调度到这些服务实例上,会给机器带来很大的负担,从而使得服务性能下降(这一点已在如图 2 所示的实验结果分析中得出).而 CBFS 调度算法与随机调度不同,它不是随机选择服务进行提交.它会根据最近一次任务的提交时

间、服务的响应时间等,从中选择负载较轻或性能较优的服务来提交请求.CBFS 这种调度方法对机器的性能影响不会很大,使得各机器的负载趋于均衡,不会使得服务性能变化很大.因此,同一任务在两种调度方法下调度执行响应时间会有所不同,甚至会出现较大的不同.

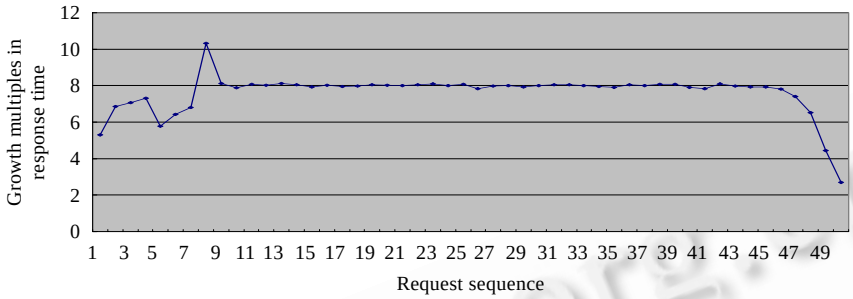

Fig.2 Relationship between response time and load

图 2 响应时间与负载的关系

Table 2 Performance comparison of two scheduling approaches

表 2 两种调度方法性能比较

Items	CBFS (ms)	CGSP (ms)	(3)−(2) (ms)	(3)/(2) (%)
$\sum_{j=0}^{N} TR_j$	3 886 437	5 759 419	1 872 982	32.5
$\sum_{j=0}^{N} ET_j$	3 842 743	5 677 934	1 835 191	32.3
$\sum_{j=0}^{N} TR_j - \sum_{j=0}^{N} ET_j$	43 694	81 485	37 791	46.4

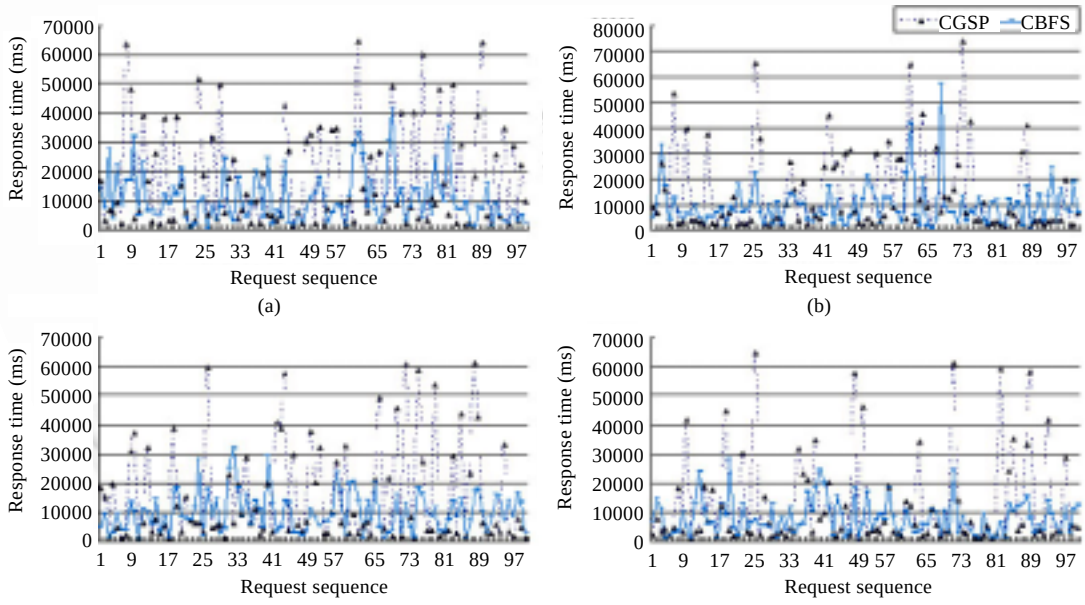

Fig.3 Comparison of response time of tasks

图 3 任务响应时间比较

表 2 最后一行反映了请求在调度器中的等待时间之和.对于 CBFS 来说,该值主要包括请求等待时间、查询 Cache 时间及查询 Cache 失败访问信息服务的时间.对于 CGSP 随机调度策略来说,该值主要包括访问信息服务

时间及请求等待时间.两者之差反映了可能的最大查询信息服务开销.在本实验中,这个开销与响应时间之差或执行时间之差相比都微不足道.这说明 Cache 节省的信息服务查询时间对网格的吞吐率作用甚小.这是由于:首先,实验环境分布在校园网内两个局域网上,网络延迟不大,因此查询信息服务所导致的开销并不大.这反过来也说明,Cache 在广域网环境中所起的作用更为明显;其次,本实验中,查询信息服务的延迟约为服务的最小响应时间的 1/5,因此服务响应时间远大于查询信息服务开销,这时,Cache 对服务响应时间的影响不会很明显.如果在广域网环境下或任务的执行时间很短的情况下,Cache 将会明显地影响网格的吞吐率.

为了显示 CBFS 能够较好地均衡负载,图 3 将 400 个请求在两种调度方法下的响应时间表示出来.图 3 中图 3(a)~图 3(d)分别是在随机调度、CBFS 调度下 4 个请求客户端分别随机发出 100 个请求的响应时间的对照.4 个子图中的横坐标均相对于请求发出客户端的请求编号.

从图 3(a)~图 3(d)可以看出,CBFS 调度机制的任务响应时间表现得较为平滑,而随机调度机制的响应时间变化剧烈.并且,CBFS 调度机制的任务最大响应时间比随机调度机制的最大任务响应时间小得多.此外,从图中可以看出,在某些情况下,随机调度请求的响应时间小于 CBFS 调度下的响应时间.这是因为:1) CBFS 调度之前先判断服务的状态,根据判断结果决定是否提交请求给服务,因此不会明显加重服务的负担,服务状态变化较为平稳.而随机调度则不考虑服务的状态,因此有可能往负载较重的服务提交过多请求,使得服务性能急剧下降;2) 准确预测服务状态比较困难,因此有时预测会发生错误,导致响应时间延长.而随机调度则是有可能调度到状态较优的服务上执行.

从以上实验结果可以看出,CBFS 调度方法在任务响应时间的平滑性、任务的吞吐率及任务在调度器等待调度的时间方面都要优于随机调度方法.

4 结 论

本文对网格环境下的任务调度算法进行了分析和研究,指出这些算法所存在的缺陷,并提出了一种基于 Cache 的反馈调度方法.该调度方法收集最近访问到的资源信息,如资源前一次响应时间、资源平均响应时间、任务提交时间等信息.网格调度器以这些实际收集的资源状态数据为依据,利用资源节点处理任务的历史数据来推断资源节点的状态进行反馈调度,将任务调度到负载较轻或性能较优的服务上执行,以达到提高网格吞吐率、均衡各资源负载的目的.该调度方法既不需要每隔一段时间采集资源的状态,也不需要频繁访问信息服务;而且,该方法不会对资源的性能产生额外影响,也不会增加额外的网络通信负担;同时,该方法减少了与信息服务交互所导致的网络延迟,缩短了任务的响应时间.实验表明,该方法具有很好的效果.

References:

[1] Foster I, Kishimoto H, Savva A, Berry D, Djaoui A, Grimshaw A, Horn B, Maciel F, Siebenlist F, Subramaniam R, Treadwell J, Von Reich J. The open grid services architecture. Version 1.0. 2005. http://forge.gridforum.org/projects/ogsa-wg

[2] Czajkowski K, Ferguson DF, Foster I, Frey J, Graham S, Sedukhin I, Snelling D, Tuecke S, Vambenepe W. The WS-resource framework. Version 1.0. 2004. http://www.globus.org/wsrf/

[3] Taiani F, Hiltunen M, Schlichting R. The impact of web services integration on grid performance. In: Proc. of the 14th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC-14). Piscataway: IEEE, 2005. 14−23.

[4] Fujimoto N, Hagihara K. A comparison among grid scheduling algorithms for independent coarse-grained tasks. In: Proc. of the 2004 Symp. on Applications and the Internet-Workshops. Washington: IEEE Computer Society Press, 2004. 674−680.

[5] Buyya R. Economic-Based distributed resource management and scheduling for grid computing [Ph.D. Thesis]. Melbourne: School of Computer Science and Software Engineering, Monash University, 2002.

[6] Jin H, Chen HH, Chen J, Kuang P, Qi L, Zou DJ. Real-Time strategy and practice in service grid. In: Proc. of the 28th Annual Int'l Computer Software and Applications Conf. Washington: IEEE Computer Society Press, 2004. 161−166.

[7] He XS, Sun XH, Von Laszewski G. A QoS guided scheduling algorithm for grid computing. Journal of Computer Science and Technology, 2003,18(4):442−451.

[8] Hu CM, Huai JP, Wo TY, Lei L. A service oriented grid architecture with end to end quality of service. Journal of Software, 2006,17(6):1448−1458 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1448.htm

[9] Jin H, Chen HH, Ning XM. QoS optimizing model and solving for composite service in CGSP job manager. Chinese Journal of Computers, 2005,28(4):578−588 (in Chinese with English abstract).

[10] Ding J, Chen GL, Gu J. A unified resource mapping strategy in computational grid environments. Journal of Software, 2002,13(7):1303−1308 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1303.pdf

[11] Zhang WZ, Hu MZ, Liu KP. Computing grid hosts clustering based on network performance. Journal of Computer Research and Development, 2004,41(12):2135−2140 (in Chinese with English abstract).

[12] Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: Proc. of the 8th Heterogeneous Computing Workshop (HCW'99). Washington: IEEE Computer Society Press, 1999. 30−44.

[13] Maheswaran M, Ali S, Siegel HJ, Hensgen D, Freund RF. A comparison of dynamic strategies for mapping a class of independent tasks onto heterogeneous computing systems. Technical Report, School of ECE, Purdue University, 1999.

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

[15] Wei TY, Zeng WH, Huang BB. Scheduling algorithm based on modified Min-Min in grid. Computer Applications, 2005,25(5): 1190−1192 (in Chinese with English abstract).

[16] Casanova H, Legrand A, Zagorodnov D, Berman F. Heuristics for scheduling parameter sweep applications in grid environments. In: Proc. of the. 9th Heterogeneous Computing Workshop. Washington: IEEE Computer Society Press, 2000. 349−363.

[17] Zha L, Xu ZW, Lin GZ, Liu YS. Scheduling algorithm for hybrid data and computation intensive metatask in grid. Computer Engineering and Design, 2003,24(10):1−4 (in Chinese with English abstract).

[18] Xu ZW, Li W. Research on VEGA architecture. Journal of Computer Research and Development, 2002,39(8):923−929 (in Chinese with English abstract).

附中文参考文献:

[8] 胡春明,怀进鹏,沃天宇,雷磊.一种支持端到端QoS的服务网格体系结构.软件学报,2006,17(6):1448−1458. http://www.jos.org.cn/1000-9825/17/1448.htm

[9] 金海,陈汉华,吕志鹏,宁小敏.CGSP 作业管理器合成服务的 QoS 优化模型及求解.计算机学报,2005,28(4):578−588.

[10] 丁箐,陈国良,顾钧.计算网格环境下一个统一的资源映射策略.软件学报,2002,13(7):1303−1308. http://www.jos.org.cn/1000-9825/13/1303.pdf

[11] 张伟哲,胡铭曾,刘凯鹏.基于网络性能的计算网格主机聚类.计算机研究与发展,2004,41(12):2135−2140.

[15] 魏天宇,曾文华,黄宝边.基于 Min-Min 改进后的网格调度算法.计算机应用,2005,25 (5):1190−1192.

[17] 查礼,徐志伟,林国璋,刘玉树.数据和计算密集混合元任务的网格调度算法.计算机工程与设计,2003,24(10):1−4.

[18] 徐志伟,李伟.织女星网格的体系结构研究.计算机研究与发展,2002,39(8):923−929.

袁平鹏(1972－),男,湖北大冶人,博士,副教授,主要研究领域为 CSCW,网格计算,分布式计算技术和系统,语义网.

邝坪(1975－),男,博士生,主要研究领域为网格计算,实时容错分布式系统,仿真网格.

曹文治(1983－),男,博士生,主要研究领域为网格计算,分布式计算技术和系统.