2008年 3月 第 3期
Vol. 29 No. 3 2008

一种基于多优先级队列和 QoS的服务调度策略

姜文超, 金 海, 王述振, 章 勤, 陶文兵

(华中科技大学 集群与网格计算湖北省重点实验室,湖北 武汉 430074)
(华中科技大学 服务计算技术与系统教育部重点实验室, 湖北 武汉 430074)
(华中科技大学 计算机学院,湖北 武汉 430074)
E-mail: myheart112@ sina.com

摘 要: 针对网格服务流程的特点,在多维调度策略分析基础之上,提出基于多优先级队列和 QoS的服务调度模型,并给出划分逻辑子网的思想,为具有不同需求的不同服务分配到最佳资源并得到最优处理.实验对资源调度公平性、服务请求响应时间等指标进行了测试,充分表明该调度模型的有效性.

关 键 词: 服务调度;多优先级队列; QoS(服务质量)
中图分类号: TP393 文献标识码: A 文 章 编 号: 1000-1220(2008)03-0450-05

Service Scheduling Policy Considering Multi-level Priority Queue and QoS

JIANG Wen-chao, JIN Hai, WANG Shu-zhen, ZHANG Qin, TAO Wen-bing
(*Hubei Province Key Lab for Cluster and Grid Computing*, *Wuhan* 430074, *China*)

Abstract This paper presents a service scheduling policy based on multi-level priority queue and QoS. And the idea of logical subnet division make different grid services from different users can assign the best resources. The experiment results indicate that this model is efficient in the image processing grid.

Key words service scheduling; multi-level priority queue; QoS (quality of service)

1 引 言

网格资源管理和调度是一个非常复杂且具有挑战性的问题,目前还没有适用于所有网格应用的资源管理和调度模式.大量研究集中在寻找一种合适的资源管理调度方法[1-3],这些方法一般可分为集中式控制和本地控制两类,前者可以获得系统所有资源信息以优化资源的使用,但扩展性差且存在单点失效;后者则不能有效了解全局资源信息,无法实现具有"意识"的网格应用.

网格作业管理根据作业的资源需求和网格资源状态,对作业所要求的资源进行选择和分配,并进行任务调度和作业执行控制,其目标是实现对网格资源的优化使用,对网格用户提供更好的服务质量.然而大部分网格作业管理系统主要的调度对象是"裸"的硬件资源,针对以程序形式提交的批处理型作业,其管理的资源对象和调度目标主要面向科学计算,在基于服务的网格环境下远不能满足应用的需求:一方面,用户的使用模式从批处理为主转向带有交互性质的服务访问模式;另一方面,不同的应用对服务质量有着不同级别的需求.

在研究基于有色 Petri网的网格工作流基础上,分析网格环境中,单个服务、多个服务以及服务流程下的服务调度策略,给出基于多优先队列和 Qos的服务调度模型,并针对服务流程,在对多维调度策略分析的基础之上,提出按计算能力、网络状态以及综合性能来划分逻辑子网的思想,结合图像处理网格服务处理的特点,最终得到适合图象处理网格平台的服务调度策略.

2 CSFN与分布式资源调度

基于有色 Petri网的网格服务工作流模型[4]—有色服务流程网(Colored Service Flow Net, CSFN),可以将数据传输与网格服务功能模块分离,由服务数据的状态来决定服务执行,实现在服务运行所需数据就绪的前提下,尽早调用网格服务对作业进行处理.

分布式调度策略与算法对资源利用率和所获得的 QoS[5]有重要影响.按调度目标分为基于系统负载均衡的调度策略[6-8]和基于 QoS的调度策略[9,10].资源调度维度是指一次调度过程中选择的资源数目,这里的资源是指调度意义上的资源,其具体含义因调度问题粒度的不同而不同.按维度可分为一维调度和多维调度[11-13].资源调度模式是指调度器的构造方式和协作方式,一般包括集中式、分布式[14,15]和分级式[16]几种.

3 基于多优先级队列和 QoS的网格服务调度策略

基于多优先级队列和 QoS的服务调度策略旨在处理不同

收稿日期: 2006-10-26 基金项目: CNGI国家发改委下一代互联网示范工程专项: 基于 IPV 6的大规模高性能网格应用(CNGI-04-15-7A)资助. 作者简介: 姜文超,男, 1977年生,博士研究生,研究方向为网格计算、生物信息技术等;金 海,男, 1966年生,教授,博士生导师, CCF高级会员,研究方向为网格计算、网络存储、集群计算、多媒体技术等.

级别用户在提交多个网格服务或网格服流程时,确保各自的执行性能,优先级队列保证所提交任务的响应速度.

3.1 多优先级队列服务调度模型

基于多优先级队列的服务调度策略如图 1所示: 各用户提交的服务请求首先进入等待队列 WQ,当等待队列中的服务所需资源可用时,依据其优先级进入就绪队列 PQ中相应的优先级就绪队列中,一旦运行窗口 RW有空闲,就根据优先级策略选取对应优先级就绪队列中的服务进行处理,处理完成或出错则进入运行结束队列 CQ中成功运行队列 SQ或运行失败队列 FQ中,若就绪队列中的服务所需资源失效,则重新进入等待队列 WQ中.其中运行窗口 RW的大小由网格节点数决定,一般地,窗口大小等于网格节点数.就绪队列中的队列的优先级从高到低依次为 PRI_0到 PRI_{r-1}.

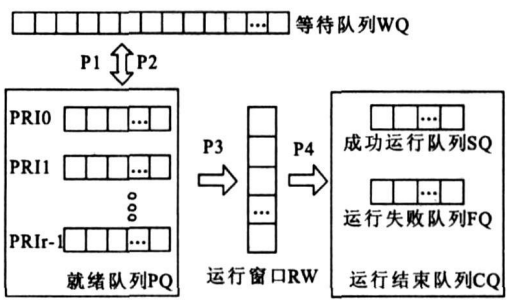


图 1　多优先级调度模型中的服务执行情况

Fig. 1　Multi-priority queue based scheduling model

为方便讨论图 1所示调度模型的服务执行情况,作以下约定:

(1) 网格服务集合为 S.

(2) 网格节点集为 N,集合大小即节点数目记为 N.num.

(3) 资源集为 R,表示计算、存储等资源信息.

(a) R(S)表示服务运行所需要的资源, R(N)表示网格节点所具有的资源.

(b) $R(s)\subseteq R(n), s\in S, n\subseteq N$,节点 n的资源满足服务 s需求,可以调度.

(c) $R(s)\subseteq R(N'), s\in S, N'\subseteq N$, 节点集 N′中所有节点的资源都满足服务 s.

(d) $R(s_1)\subseteq R(s_2), s_1, s_2\in S$, 表示服务 s_2比 s_1对资源的要求高.

(e) max(R(S))为服务集合中对资源要求最高的服务.

(4) Sche(s, n),表示服务 s调度到节点 n.记 Sche.S为已调度好的服务的集合, Sche.N表示网格中有服务运行的节点集合. Exec(s, n)表示服务 s在节点 n上运行.

(5) $C(n), n\in N$,表示节点计算能力; $\max(C(N')), N'\subseteq N$表示节点集合 N′中计算能力最强的节点.

(6) $dis(n_1, n_2), n_1, n_2\in N$,表示网格节点 n_1与 n_2之间的距离向量,其值越大,两节点间传输速率越小,主要由数据量大小与两点间网络状态 $dis(n_1, n_2)$决定.

(a) $dis(n_1, n_2)=0$表示 n_1、n_2为同一节点, $dis(n_1, n_2)=\infty$表示两节点不连通.

(b) $\min(dis(n_0, N'))$, $N'\subseteq N$,表示节点集 N′中与节点 n_0距离向量最小的节点.

(c) dis(N)表示网格中各节点的邻接距离向量矩阵.

对于队列处理,作以下约定:

(7) 服务队列 Q= {WQ, PQ, RW, CQ},其中 CQ= SQ+FQ;

$PQ=\bigcup_{i=0}^{r-1} PQ_i$, PQ_i表示优先级为 i的就绪队列, w(i)为级别 i对应的优先级权值.

(8) Q(s), $s\in S$,表示服务 s存在于队列 Q中; Q.len表示队列 Q中服务的个数.

(9) $s: Q1\rightarrow Q2, s\in S, Q1, Q2\subseteq Q$表示服务 s从队列 Q_1迁移到队列 Q_2.

对于提交的服务集 S,网格节点集 N,网格节点邻接距离向量矩阵 dis(N),基于多优先级队列和 QoS的调度策略算法描述如下:

(1) 等待队列: 周期性查询并将满足条件的服务转移至相应级别的就绪队列.

(1.1) $\forall s\in S\cap WQ(s), s: WQ(s)\rightarrow PQ_{i-1}, i=s.pri+1$

(1.2) wait some time, go to (1.1)

(2) 就绪队列: 周期性查询运行窗口,一旦队列不满,则从高优先级至低优先级,依次根据优先级权值来进行选取.

(2.1) e= RW.size-RW.len, if $e\leqslant 0$ go to (2.3)

(2.2) for i= 0 to r-1

if $e\leqslant 0$ go to (2.3)

$r_i=e\times w(i)\sum_{k=0}^{i} w(k)$, set a random num $t\in(0,1)$

$$r_i=\begin{cases}[r_i], (r_i\in\{1,2,3,\cdots\}) \text{ or } (t<[r_i]+1-r_i)\\ [r_i]+1, (other)\end{cases}$$

if $r_i>0, s_j: PQ_i\rightarrow RW, j\in\{0,1,2,\cdots,\min(RQ.len, r_i)-1\}$

$e=e-\min(RQ_i.len, r_i)$

end for

(2.3) sleep some time, go to (2.1)

(3) 运行窗口: 运行窗口中尚未运行的服务请求,根据资源要求从高到低的原则,利用最小计算时间、最小传输时间或最小执行时间调度策略,在未分配任务的网格节点中选取合适的节点,创建服务运行实体;然后服务运行实体开始运行,运行成功(失败)则将对应的服务从运行窗口转入成功运行(失败)队列,同时撤销该实体.运行窗口在调度并处理服务时,优先处理资源要求高的服务请求,这样可以较好地避免对资源要求较高的服务长时间得不到执行.

(3.1) if WR.len$\leqslant$ 0 go to (3.3)

(3.2) for i= 0 to WR.len-1

select s: $s=\max(R(\{s\mid WR(s)\}))$

get node n ($n\in N-Sche.N$) by MCT/MTT/MET

if n exists, i++ and start (3.4) with(s, n)

end for

(3.3) sleep some time, go to (3.1)

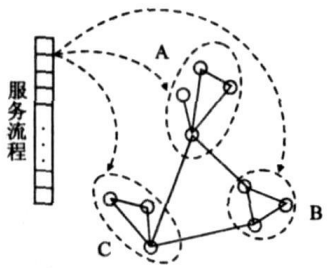
(3.4) Sche(s, n), Exec(s, n)

if succeed s: RW→SQ, else s: RW→FQ

3.2 基于QoS的多维服务调度与逻辑子网划分

针对用户可以编辑服务流程来提交包含一系列服务处理的网格作业的特点.提出基于 QoS划分逻辑子网的思想,这里QoS特指网格应用程序的执行时间.

具有相似处理性能的若干网格结点所构成的网络结构称为逻辑子网,可以依据计算能力、网络状态以及综合性能来划分逻辑子网,在服务流程中,根据队列中不同服务的资源需求,某个服务一旦被调度到某个逻辑子网,则流程后面的具有相似资源需求的各服务均在子网内进行调度,保证该子网内资源的充分利用和作业的最佳执行.如图 2所示为按计算能力划分子网的示例.



A B C代表三个计算能力不同的逻辑子网
弧线表示可能的调度过程

图 2 以计算能力为标准的逻辑子网划分

Fig. 2 Example of subnet-partition. A, B and C denote three subnets

4 服务调度性能测试

4.1 公平性比较

公平性是指不同优先级服务在队列中等待的长度与其优先级权重成反比,即一方面高优先级服务优先处理,同时低优先级服务不能一直处于等待状态.

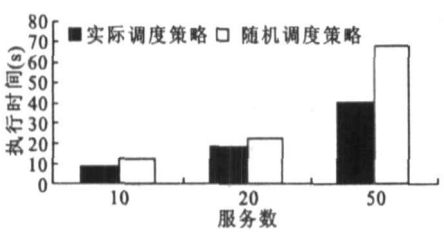


图 3 调度性能测试

Fig. 3 Comparing of scheduling performance

(1) 本文方法与随机调度策略的性能比较,针对应用中的典型服务—高通滤波变换,分别提交 10、20、50次,记录其时间开销,结果如图 3所示.可见由于本文调度策略采用了运行窗口机制,确保按照网格节点执行性能分配网格服务,当提交的服务数较多(如 50个)时,基于多优先级队列的服务调度策略明显好于随机调度策略.

(2) 基于多优先级队列的调度策略确保各优先级服务按其优先级进行处理,实验提交 20个高优先级服务与 10个低优先级服务时,测试高优先级队列中各服务开始运行时,低优先级服务已开始运行的个数,同时测试了低优先级队列中各服务开始运行时,高优先级服务已开始运行的个数,结果如图 4与图 5所示.

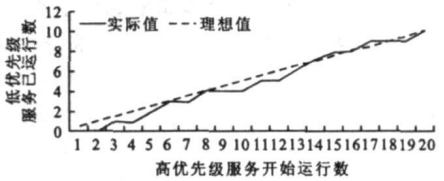


图 4 高优先级服务开始运行时低优先级服务已运行数测试

Fig. 4 The number of running services with low priority when services with high priority

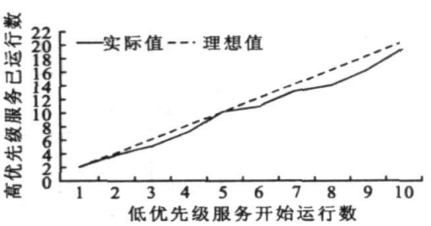


图 5 低优先级服务开始运行时高优先级服务已运行数测试

Fig. 5 The number of running services withhigh priority when services with low priority

目前平台支持两种优先级服务,高、低优先级服务的权重比为 2∶1.平台有 3个网格节点,因此开始是 2个高优先级与一个低优先级服务可以同时通过运行窗口并发处理.一旦运行窗口有空闲,一般是某一节点的服务执行完成,调度策略则按照 2/3的概率选取高优先级服务,按照 1/3的概率选取低优先级服务.从实验结果可以看出,高低优先级服务调度数目比约等于理想比值,即其权重比 2∶1.

4.2 服务请求响应时间测试

用户请求最终都被分解为服务的请求,因此服务响应时间成为网格平台的主要性能指标.由于个体服务执行时间的差异,不可能得到统一的服务响应时间,但是,通过对典型服务的响应时间测试,可以对平台整体性能与稳定性做出初步分析.

表 1 平均处理时间比较

Table 1 Comparing of mean processing time

间隔时间(s)	平均响应时间(h:m:s.ms)
1	0:0:2.58
2	0:0:1.947
3	0:0:1.950

(1) 周期性提交服务请求时各服务的响应时间,分别以 1秒、2秒和 3秒为周期向平台提交服务.结果如表 1和图 6所示.

由实验结果可知,提交间隔大于 1秒时,平均响应时间变化不大,约为 2秒,而且其最大、最小处理时间之差约为 2秒,系统运行稳定.可以认为,当时间间隔大于一定值时,系统能够及时响应服务请求并稳定执行.对于连续服务之间执行时间的波动是因为各节点执行时间有差异,而调度总在不同的节点间进行.

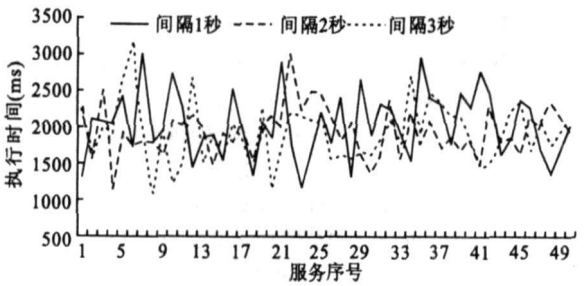


图 6　服务响应时间测试

Fig. 6　Comparing of responsing time

(2) 单位时间内平台响应服务数.在 1分钟内,分别提交 10、20、40、60、80个表 1所示的服务,测试其平均响应时间与最大响应时间,实验结果如图 7与表 2所示.

表 2　1分钟内最大响应时间测试

Table 2　The max responsing time in 1 minitute

提交服务数	最大响应时间 (h:s:m:ms)
10	0:0:2.570
20	0:0:2.359
40	0:0:2.772
60	0:0:2.795
80	0:1:2.149
100	0:1:33.952

可见,当平台每分钟处理服务数不大于 60时,平均响应时间维持在 2秒左右,当服务数大于 60时,最大响应时间急剧增加.当网格中节点数目增加时,理论上可以缓解单位时间

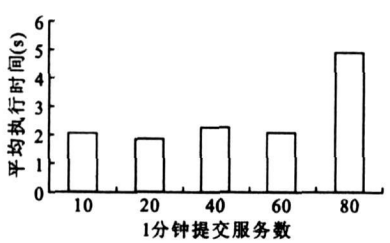


图 7　1分钟内平均响应时间测试

Fig. 7　The mean responsing time in 1 minitute

内大量服务请求时性能明显下降的情况,然而测试调度开销与各节点服务执行能力的不一致也会对响应时间造成影响.

4.3　子网划分对于实际服务 QoS的影响

为测试逻辑子网划分对平台性能的影响,在子网划分前后运行相同的服务序列,对服务执行时间进行统计,结果图 8所示.

由图 8可知,在整体性能上,子网划分后都比划分前有所提高,但也存在相同甚至性能降低的情况,主要原因是当相邻两个服务序列属于不同类型时,比如从计算密集型服务跳跃到通信密集型,需要一定的系统和通信开销,同时遇到网络性能较差时也将出现该现象,而且这种情况将更加明显.从运行

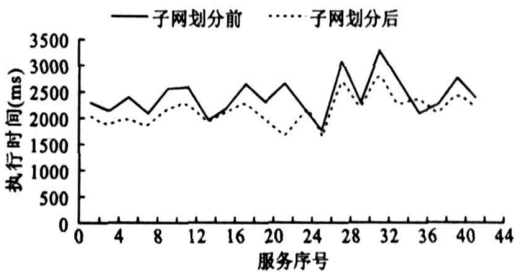


图 8　子网划分前后相同服务序列的执行时间比较

Fig. 8　Comparing of executing time of services after subnet partition

经验知,这种情况较少出现,对整体性能提高来说,子网划分仍具有较高的性能优势.

5　结　论

给出基于多优先队列和 Qos的服务调度模型,并针对服务流程,在多维调度策略分析的基础之上,提出按计算能力、网络状态以及综合性能划分逻辑子网的思想,优化不同服务请求的资源调度过程,使其得到最优处理,得到适合图像处理的服务调度策略.

在以后工作中,将继续对子网划分方法进行研究,使系统能在子网划分基础之上进一步提高任务和资源的优化匹配,从而提高调度性能.

References

[1] Foster I, Kesselman C. The grid, blueprint for a new computing infrastructure [M]. San Francisco: Morgan Kaufmann Publishers Inc, 1998.

[2] Czajkowski K, Foster I. A resource management architecture for metacomputing systems [A]. In: Feitelson, D. G. Rudolph. L, eds. Proceedings of the 4th Workshop on Job Scheduling Strategies for Parallel Processing [C]. LNCS 1459. Orlando: Springer-Verlag, 1998, 62-82.

[3] Sekiguchi S, Sato M. Ninf: network based information library for globally high performance computing [C/OL]. In: Proceedings of the Parallel Object-Oriented Methods and Applications (POOMA), 1996, 39-48. http://www.acl.lanl.gov/Pooma96/.

[4] Jin Hai, Wang Shu-zhen. A colored petri-net based grid workflow model [J]. Journal of Huazhong University of Science and Technology, 2006, 34(7): 39-41.

[5] He Xiao-shan, Sun Xian-he, Von Laszewski Gregor. QoS guided min-min heuristic for grid task scheduling [J]. Journal of Computer Science and Technology, 2003, 18(4): 442-451.

[6] Dattatreya G R, Venkatesh R. Static and decentralized-adaptive load balancing in a star configured distributed computing system [J]. IEEE Transactions on Systems, Man and Cybernetics, Part A, 1996, 26(1): 91–104.

[7] Grosu D, Chronopoulos A T. Algorithmic mechanism design for load balancing in distributed systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 34(1): 77–84.

[8] Ka-Po Chow, Yu-Kwong Kwok. On load balancing for distributed multiagent computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(8): 787–801.

[9] Dogan A, Ozguner F. On QoS-based scheduling of a meta-task with multiple QoS demands in heterogeneous computing [C]. Proceedings of International Parallel and Distributed Processing Symposium. Ft. Lauderdale, FL, USA. 2002. Los Alamitos, CA, USA: IEEE Comput. Soc, 2002, 50–55.

[10] He Li-gang, Jarvis S A, Spooner D P, et al. Dynamic scheduling of parallel jobs with QoS demands in multiclusters and grids[A]. In: Rajkumar Buyya ed.. Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing [C]. Pittsburgh, Pennsylvania. 2004. Los Alamitos, CA, USA: IEEE Computer Society, 2004, 402–409.

[11] Weissman J B. Scheduling multi-component applications in heterogeneous wide-area networks [C]. In: Proceedings of 9th Heterogeneous Computing Workshop, 2000. (HCW 2000). Cancun, Mexico. 2000. Los Alamitos, CA, USA: IEEE Comput. Soc, 2000, 209–215.

[12] Lipari G, Buttazzo G. Scheduling real-time multi-task applications in an open system [C]. In: Proceedings of the 11th Euromicro Conference on Real-Time Systems. York, UK. 1999. Los Alamitos, CA, USA: IEEE Comput. Soc, 1999, 234–241.

[13] Dauzere-Peres S, Pavageau C. Extensions of an integrated approach for multi-resource shop scheduling [J]. IEEE Transactions on Systems, Man and Cybernetics, Part C, 2003, 33(2): 207–213.

[14] Goel S, Sharda H, Taniar D. Distributed scheduler for high performance data-centric systems [C]. Conference on Convergent Technologies for Asia-Pacific Region, 2003, 3: 1157–1161.

[15] Peh L S, Ananda A L. The design and development of a distributed scheduler agent [C]. 1996 IEEE Second International Conference on Algorithms and Architectures for Parallel Processing (ICAPP 96). Singapore. 1996. New, NY, USA: IEEE, 1996, 108–115.

[16] Johnsson K B, Cox D C. An adaptive cross-layer scheduler for improved QoS support of multiclass data services on wireless systems [J]. IEEE Journal on Selected Areas in Communications, 2005, 23(2): 334–343.

附中文参考文献:

[4] 金 海,王述振.基于有色 Petri网的网格工作流模型 [J].华中科技大学学报, 2006, 34(7): 39–41.