一种基于参数扫描应用容错的粗粒度网格调度算法

张 磊[1],王 毅[2]

(1.河海大学水利水电工程学院,江苏 南京 210098;2.61226 部队,北京 100079)

摘要:考虑作业处理时延以及作业传输时延对调度器性能的影响,在 RR 调度算法的基础上提出了一种新的基于参数扫描应用的调度算法 PRR,并对该算法进行了理论分析,得出了该调度算法不需要作业以及处理器的相关信息,且具有比较好的容错性的结论.利用网格仿真软件 Gridsim 对 PRR 调度算法的理论分析结果进行了验证,并将该算法与 RR 调度算法的性能进行了比较.结果表明,当时延较大时,PRR 调度算法性能改善比较明显.

关键词:网格调度器;参数扫描;PRR 调度算法;RR 调度算法;容错调度器

中图分类号:TP311 **文献标识码**:A **文章编号**:1000-1980(2008)02-0258-05

调度是网格系统中的核心问题,目的是在网格资源间合理地分配负载,使系统的综合性能达到最优.目前,基于网格的调度系统各个作业互不相关的情况比较适用.实践表明,在网格环境下,如果作业之间存在数据交换,由于资源之间交互的时延和带宽的影响,所有资源协同计算总体的效率可能比单个资源计算的效率还低.作业之间没有数据交换也是比较符合实际情况的,现实生活中的许多应用都可以分解成互不相关子作业的集合.其中最典型的应用是参数扫描应用(a parameter sweep application)[1].参数扫描应用的领域十分广泛,像目前的生物信息、数据挖掘、网络仿真、电子商务模型仿真、电子 CAD、蒙特卡洛仿真等都属于参数扫描应用领域[2-3],即这些应用通过一定的手段都可以分解成具有独立参数的子作业的集合.特别是工程中的很多力学问题,比如有限元和边界元计算等,经过恰当的处理后都符合参数扫描应用要求.

目前已提出了许多针对参数扫描应用的调度算法,包括 DFPLTF[2],Suffrage-C,Min-min[4],Max-min[4] 和 WQ[5] 等.这些调度算法能够根据处理器和作业的变化动态地分配作业,但需要作业和处理器的相关信息.而在网格环境下,本地资源是动态的,所以作业和处理器的信息很难获得.针对这种情况,文献[6]提出了 RR 算法.这种调度算法的调度策略不需要任何的处理器和作业信息,总作业数较多时其调度策略与其他算法相比,相差不大[7].

上述这些调度算法都是在忽略作业提交时延和作业传输时延的情况下提出来的.很多情况下,这种假设并不符合实际情况.文献[8]对目前的一些常用的 WSRF 和 WS-Notification 的实现平台进行了测试,测试结果表明,在 X509 安全机制下,由资源的创建、资源的销毁、资源属性设置、资源属性的获取以及通知的处理组成的作业提交时延可能达到几秒.而对于一些需要传输大量数据的作业来说,传输数据的时延比作业执行的时延还要长,这些算法将不再符合实际情况.针对这种情况,本文提出了一种符合参数扫描应用要求的新的不需要处理器和作业信息的调度算法.

1 调度系统模型

网格调度器可以在一定准则下将作业分配到异构的、分布的处理器中.网格调度的关键问题是调度目标函数的确定.一般调度器目标函数的指标是生产周期(makespan),而在网格环境下,处理器的处理能力是随时间变化的.生产周期相同,处理器总的处理能力可能不一样,这样通过生产周期这个指标很难衡量一个调度算法性能的好坏.为此,文献[3]提出了一种新的标准——TPCC(total processor cycle consumption).本文以 TPCC 为指标来衡量调度器的性能.

收稿日期:2007-03-16

基金项目:国家自然科学基金(60403043);河海大学"十五"211 项目(GF1.3)

作者简介:张磊(1980—),男,天津人,博士研究生,主要从事高性能计算和网格计算研究.

处理器的处理能力指单位时间内处理指令的数目,用 $s_{p,t}$ 表示处理器 p 在 $[t,t+1)$ 时间段所处理的指令的数目.可以看出,受本地作业的影响,处理器的处理速度随时间而变.作业的大小是指完成一个作业所需要的计算机指令,考虑用 T 代表 n 个相互独立的作业的集合.这样每一个作业可以用一个三元组 (v,p,t) 代替,其中 $v\in T$,$1\leqslant p\leqslant m$(m 是处理器的个数),t 是每一个作业的开始时间.一个调度器 S 将一组作业 T 在 m 个处理器中进行调度时,三元组要满足以下 2 个条件:

R1 对于每一个 $v\in T$,都至少存在一个三元组 $(v,p,t)\in S$.

R2 不可能同时有 2 个三元组 (v,p,t),$(v',p,t')\in S$,其中 $t<t'<t+d$,$t+d$ 表示作业 v 的完成时间.

文献[3]所采用的指标 TPCC 的表达式为

$$C=\sum_{t=0}^{M}\sum_{p=1}^{m}s_{p,t} \tag{1}$$

式中:C——处理器的 TPCC;M——调度器的工作周期(makespan).从式(1)可以看出,一个调度器的 TPCC 和这个调度器的工作周期(makespan)呈正比关系.所以,从 TPCC 的角度来衡量调度器的性能与从工作周期(makespan)的角度来衡量调度器的性能具有等价关系.本文基于调度器的 TPCC 指标,提出在网格环境下考虑时延的一种全新的调度算法——PRR 算法.考虑时延后,也就将处理时延及数据传输因素加入到了生产周期中.所以

$$M=\max_{p=1}^{m}(s_p+\delta_p+\beta_p) \tag{2}$$

式中:s_p——处理器 p 用于计算的时间;δ_p——p 处理器在作业计算前所有的时延,其中包括作业的创建、属性设置、资源属性的获取等时延;β_p——p 处理器在作业计算后的时延,包括作业的销毁、作业结果的传输等.与不考虑时延的工作周期相比,式(2)多了 δ_p 和 β_p 2 项.

2　PRR 调度算法

2.1　算法的提出

根据调度器在网格系统中的地位,将调度器分为 2 种类型:(a)在多个集群间进行作业调度的全局调度器;(b)在集群内部进行作业调度的本地调度器.基于此,PRR 算法采用考虑作业操作时延以及数据传输时延的调度策略.这个调度算法以 TPCC 为指标,通过调度策略使 TPCC 最小.

下面对本地调度器和全局调度器进行抽象.基于上述条件 R1 和 R2,可以将本地调度器抽象成一个作业的容器,且设此容器是一个 FIFO 队列.全局调度器在整个网格资源中进行作业的调度,全局调度中心也是一个 FIFO 队列,其内部作业是所有未提交的作业集合.

在算法提出之前首先定义算法的数据结构.整个数据结构分为 2 个作业等待提交队列和 1 个全局作业提交环,如图 1 所示.一个队列为全局作业等待队列,其中存放的是等待提交到本地调度队列中的作业,这些作业在网格范围内进行调度;另一个队列为本地调度器队列,其中存放的是提交到本地等待本地资源执行的作业.队列最下方代表目前正在执行的作业,其余作业代表正在等待的作业.全局作业提交环中存放的是已经由全局调度器提交到本地调度器的作业的副本,每个作业与本地调度器中的作业相对应,本地调度器中作业执行完毕也要相应地将全局作业提交环中的作业副本删除.

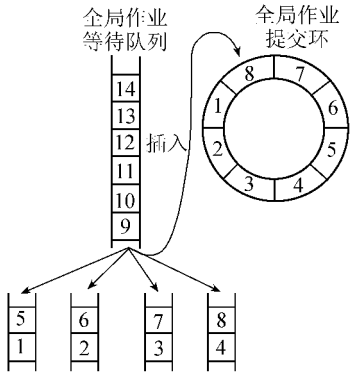

图 1　调度器数据结构

Fig.1　Data structure of scheduler

PRR 调度算法 (set T, set P):

//T 表示 $2n$ 个作业的集合.　P 表示 m 个处理器集合

//满足条件($n\gg m$)

//GQ_1 代表全局等待作业;GQ_2 代表全局作业提交环

//$LQ_0\cdots\cdots LQ_{m-1}$ 分别表示本地调度器队列

1: 将所有 T 中的作业提交到 GQ_1 中;

2: While (本地调度队列没满)

3: 提交 GQ_1 中的 m 个作业分别到 $LQ_1\cdots\cdots LQ_m$ 中,同时将作业备份到全局作业提交环 GQ_2 中;

4: While ($GQ_1\neq\varnothing$)

{

5: 等待处理器作业完成通知(处理器 x 作业 t 完成);

6: 将全局作业提交环中相应的作业 t 删除;

7: 将 GQ_1 中的作业提交给 LQ_x,同时将其备份到 GQ_2 中;

}

8: While ($GQ_2 \neq \varnothing$)

{

9: 等待处理器作业完成通知;(处理器 x 作业 t 完成);

10: 将 $LQ_1 \cdots\cdots LQ_m$ 中的所有的作业 t 的实例都删除掉;同时在全局作业提交环 GQ_2 中将其删除;

11: 在 GQ_2 中选择一个作业备份提交到 LQ_x 中;

}

基于图1的数据结构提出的调度算法伪代码如上.此算法分3个阶段:第1阶段(1~3行)为系统初始化阶段,将作业分配到每个计算节点的本地队列中,并使本地队列排满,同时将提交的作业副本放入 GQ_2 中.第2阶段(4~7)为系统顺序执行阶段,当全局调度器接收到执行节点执行作业完毕的通知后,将 GQ_2 中的副本作业删除,分配等待队列 GQ_1 中的作业到该队列,并将这个作业备份到全局调度器的提交环 GQ_2 中.循环这个过程直到队列 GQ_1 中所有作业都分配完毕.第3阶段(8~11行)为优化阶段,该阶段主要是充分利用系统资源来减小总 TPCC.接收到某执行节点执行作业完毕的通知后,删除提交环中相应的作业副本,并且将本地调度器中这个作业的所有副本删除,由于等待队列中已经没有作业,这时提交作业提交环 GQ_2 中的作业副本到本地调度队列中.循环这个过程直到所有作业都完成.

2.2 PRR 算法分析

2.2.1 资源作业队列长度

2.1中提出的算法假设队列长度为2,下面对队列长度对性能的影响进行理论分析.首先考虑2种简单的情况:第1种情况,假设每个作业时延之和小于该作业执行时间;第2种情况,假设作业时延之和大于作业执行时间.对于这2种情况,只要队列长度为2,只要在本地资源作业队列中有一个作业在等待,即实现数据传输和作业执行的并行化.

下面针对普遍情况(即第3种情况)进行分析.第3种情况是:有些作业时延时间大于作业的执行时间,有些作业的执行时间大于作业的时延.对于这种情况,适当地增加队列长度可以提高系统的性能.这里所指的系统性能,即生产周期(这里的指标采用了生产周期而没用采用 TPCC,目的是为了便于分析).假设作业数目为 n,对于作业(v,p,t),设作业执行时间为 $s_{p,v}$,2次作业提交之间的时延 $\delta_{p,v}+\beta_{p-1,v}=\alpha_{p,v}$.由于作业的随机性,假设作业的执行时间 $s_{p,v}$ 以及作业的时延 $\alpha_{p,v}$ 分别服从负指数分布,即作业的执行时间是均值为 $1/\mu$ 的负指数分布,作业的时延是均值为 $1/\lambda$ 的负指数分布.在此假设的基础上对资源队列长度对系统性能的影响进行分析.在调度过程中,如果本地调度器队列已满,全局调度器会等待本地调度器有空余位置时,才向其进行作业提交.由于作业时延是负指数分布,根据负指数分布的无记忆性,可以假设当本地调度器队列已满时全局调度器还是继续向其提交作业,如果经过作业提交时延过后发现本地调度器作业队列还是满的,这时将该作业丢弃.

从排队论[9]和生灭过程结论可得,队列长度为 k 时的损失概率

$$\pi_k=\begin{cases}\dfrac{a^k-a^{k+1}}{1-a^{k+1}} & (a\neq 1)\\ \dfrac{1}{k+1} & (a=1)\end{cases} \tag{3}$$

式中 a 为系统供给负荷,$a=\lambda/\mu$.由式(3)可知,当 $a=1$ 时,$\pi_{k-1}-\pi_k$ 的值最大.由式(3)还可以看出,当 $a=1$ 时,$\pi_0-\pi_1=1/2$,$\pi_1-\pi_2=1/6$,$\pi_2-\pi_3=1/12$,队列的大小对性能的影响变小.因此,资源队列长度没有必要选择得太长.

2.2.2 算法特点

PRR 调度算法能够很好地消除网格环境下时延对调度器性能产生的影响.网格环境下时延一般是不能忽略的,PRR 调度算法的本质是在 RR 算法的基础上,将作业的处理时间和作业的时延在作业处理过程中并行化.

从 PRR 算法可以看出,调度算法本身并不需要有关的作业信息和处理器信息,这是比较符合实际情况

的.首先对处理器的实际处理能力信息的收集有困难,其次作业的信息的获取有时也比较困难,对于有些作业,很难精确确定作业的大小.如果将处理器和作业信息作为作业调度的依据,当这些信息估计不准确时反而会使系统性能下降.所以,本文提出的调度算法非常适用于网格环境.

PRR 调度算法本身具有容错性.假设在作业执行过程中一个资源节点出现故障,这时提交到这个资源上的作业结果得不到返回,但是将全局调度队列 GQ_1 中的作业提交完毕后,如果还有资源空余,这时调度算法会自动复制故障节点上的作业并且提交到其他计算资源上去执行,从而达到了容错的目的.

3　系 统 仿 真

采用网格仿真工具 GridSim[10] 对系统性能进行仿真,将 PRR 算法与文献[5]提出的 RR 算法进行不同时延情况下性能的比较.首先对仿真架构进行基本的设置.本算法根据仿真的需求,网络拓扑结构设定 2 个路由,一个路由用来连接调度器,一个路由用来连接所有计算资源,调度器到路由的带宽和资源到路由的带宽均由调度器和资源本身设定,2 个路由之间的带宽为10 MB/s.网格资源设定共享模式为 space-shared,设定网格资源中含有一个 machine,每个 machine 的处理器个数为 1,网络传输时延为 10 ms,数据包最大长度为1.5 kB.调度器作业的执行时间、输入数据长度、输出数据长度可以不同.仿真设定网格资源数量为 6,资源处理速度如表 1 所示,每个资源与路由之间的链路带宽为 1 MB/s.

表 1　网格资源处理速度

Table 1　Processing capability of grid resource

资源号	0	1	2	3	4	5
处理速度/(10^6 指令·s^{-1})	100	120	90	210	180	150

图 2 针对作业输入输出数据长度的不同,对 RR 算法和 PRR 算法性能进行了比较.作业个数设定为 200,作业大小为 2×10^9 指令/s.从图 2 仿真结果可以看出,当作业在 200 kB 以下时,随着作业数据长度的增加,PRR 算法系统总的 TPCC 变化不大,而 RR 算法系统总的 TPCC 基本上是随着作业长度的增加而线性增加.但是,当数据量过大时,PRR 算法的 TPCC 也会提高,这是因为网络已经饱和,如果想进一步提高性能,只有通过提高网络带宽来实现.

图 3 对作业大小不变而作业个数变化时算法的性能进行了比较.作业大小在 $1\times10^9\sim2\times10^9$ 指令之间,作业的输入输出数据的平均长度为 50 kB,变化范围为 ±10%.从图 3 仿真结果可以看出,随着作业个数的增加,相对于 RR 算法性能的改善程度,PRR 算法性能的改善程度要大些,这是因为当作业个数增加时,PRR 算法所节省的作业时延也相应增加,从而节省了 TPCC.

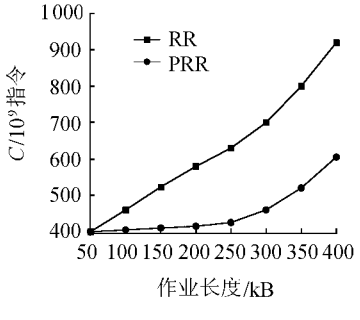

图 2　作业数据长度对 C 的影响

Fig.2　Influence of task data length on C

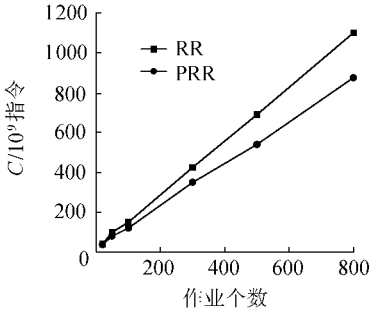

图 3　作业个数对 C 的影响

Fig.3　Influence of task number on C

表 2 为本地调度队列长度对系统性能影响的仿真结果.网格资源与图 3 的资源设置相同,调度器与路由链路带宽为 10 MB/s,作业总数为 200.其中:第 1 种情况设定作业大小为 20×10^9 指令,作业的输入输出数据的平均长度为 2 MB,变化范围为 ±10%;第 2 种情况设定作业大小为 1×10^9 指令,作业的输入输出数据的平均长度为 10 MB,变化范围为 ±10%;第 3 种情况设定作业大小为 2×10^9 指令,作业的输入输出数据的平均长度为 8 MB,变化范围为 ±80%.从仿真

表 2　3 种情况不同队列长度的 C 比较

Table 2　Comparison of C of different queue lengths in 3 different cases

情况	$C/10^{12}$指令			
	队列长度为 1	队列长度为 2	队列长度为 3	队列长度为 4
1	31.1	22.4	22.3	22.3
2	28.2	14.1	13.0	13.1
3	26.0	14.6	12.6	12.1

注:3 种情况如 2.2.1 所述.

结果可以看出:前2种情况,队列长度为2基本能满足要求;第3种情况,队列增加后系统性能会有一定的改善;在第2种情况下,队列长度为3比队列长度为2的性能稍好些,这是由于结果输出时连接资源的路由的最大传输单元瓶颈造成的.

4 结 束 语

本文针对能够分解成多个独立作业的参数扫描应用问题,在RR算法的基础上提出了一种能克服作业网络时延以及大数据量文件传输对调度器性能影响的调度算法.该算法不需要处理器和作业的相关信息,具有天然的容错性.本文还对该算法的性能和限制条件进行了理论分析,并利用仿真软件Gridsim对该算法进行了仿真验证.PRR算法与RR算法的性能比较结果表明,在有实际时延的条件下,PRR算法与RR算法相比,性能有较大程度的改善.

参考文献:

[1] CASANOVA H, LEGRAND A, ZAGORODNOV D, et al. Heuristics for scheduling parameter sweep applications in grid environments [C]//9th Heterogeneous Computing Workshop. Washington: IEEE Computer Society, 2000: 349-363.

[2] PARANHOS D, CIRNE W, BRASILEIRO F. Trading cycles for information: using replication to schedule bag-of-tasks applications on computational grids[J]. Lecture Notes in Computer Science, 2003, 2790: 169-180.

[3] BUCUR I D, EPEMA H J. Local versus global schedulers with processor co-allocation in multicluster systems[J]. Lecture Notes in Computer Science, 2002, 2537: 184-204.

[4] MAHESWARAN M, ALI S, SIEGEL H J, et al. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems[C]//8th IEEE Heterogeneous Computing Workshop. Washington: IEEE Computer Society, 1999: 30-44.

[5] GRAHAM R L. Bounds for certain multiprocessing anomalies[J]. Bell System Technical Journal, 1996, 45: 1563-1581.

[6] FUJIMOTO N, HAGIHARA K. Near-optimal dynamic task scheduling of precedence constrained coarse-grained tasks onto a computational grid[C]//The 2nd International Symposium on Parallel and Distributed Computing. Washington: IEEE Press, 2003: 80-87.

[7] FUJIMOTO N, HAGIHARA K. A comparison among grid scheduling algorithms for independent coarse-grained tasks[C]//SAINT 2004 Workshop on High Performance Grid Computing and Networking. Washington: IEEE Press, 2004: 674-680.

[8] MARTY H, JAREK G, JOE B, et al. State and events for web services: a comparison of five ws-resource framework and ws-notification implementations[C]//The 14th IEEE International Symposium on High Performance Distributed Computing. Washington: IEEE Computer Society, 2005: 24-27.

[9] ALLEN A. Probability, statistics and queuing theory with computer science[M]. Toronto: Application Academic Press, 1978.

[10] BUYYA R, MURSHED M. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing[J]. The Journal of Concurrency and Computation, 2002, 14: 13-15.

A fault tolerant grid scheduling algorithm for coarse-grained tasks based on parameter sweep applications

ZHANG Lei[1], WANG Yi[2]

(1. *College of Water Conservancy and Hydropower Engineering, Hohai University, Nanjing* 210098, *China*;
2. 61226 *Military Force, Beijing* 100079, *China*)

Abstract: Considering the effect of delay in task submitting and data transferring on scheduler performance, a new scheduling algorithm PRR based on parameter sweep application was presented based on RR scheduling algorithm. The theoretical analysis shows that the new algorithm has high fault tolerance and does not need the related information of tasks and processors. With the simulation software Gridsim, the theoretical result of the PRR scheduling algorithm was verified, and the performance of the new scheduling algorithm at long delay was obviously improved as compared with that of the scheduling algorithm RR.

Key words: grid scheduler; parameter sweep; PRR scheduling algorithm; RR scheduling algorithm; fault tolerant scheduler