OPT－Min－Min:基于 Min－Min 网格资源调度算法的优化

张忠平[1,2],温利娟[1]

[1](燕山大学 信息科学与工程学院,河北 秦皇岛 066004)
[2](河北省计算机虚拟技术与系统集成重点实验室,河北 秦皇岛 066004)
E-mail:zpzhong@ysu.edu.cn

摘　要: 网格是由大量地理上分布的异构资源组成的高性能并行计算系统,网格资源调度算法在网格资源管理中具有重要意义.在众多的启发式调度算法中,Min-Min 调度算法取得了良好的调度结果,但是 Min-Min 调度算法导致负载不平衡.本文针对 Min-Min 调度算法存在负载不均,采用重负载资源的任务分配给轻负载资源执行来均衡负载的策略,提出 OPT-Min-Min 算法,提高资源利用率,达到较小的完成时间;最后采用 Braun 等人提出的仿真模型基准来验证算法有效性.

关 键 词: 网格;资源调度;Min-Min 算法;负载平衡
中图分类号: TP311　　**文献标识码:** A　　**文 章 编 号:** 1000-1220(2014)07-1573-05

OPT-Min-Min:the Grid Resources Scheduling Algorithm Based on Load Balance

ZHANG Zhong-ping[1,2],WEN Li-juan[1]

[1](College of Information Science and Engineering,Yanshan University,Qinhuangdao 066004,China)
[2](The Key Laboratory for Computer Virtual Technology and System Integration of Hebei Province,Qinhuangdao 066004,China)

Abstract: A Grid is a high performance parallel computing system, which consisted of a large number of geographically distributed heterogeneous resources connected by network. Grid resources scheduling algorithm is of great significance in the grid resource management system. Among numerous heuristic scheduling algorithms, the Min-Min scheduling algorithm has achieved good scheduling results, but it leads to unbalanced load. According to the Min-Min scheduling algorithm in uneven load, we put forward the OPT-Min-Min algorithm. The tasks on heavy-loaded resources will be assigned to resources that need less time to load balance, raise resource utilization rate, and achieve lesser completion time. At last, we used benchmark of instance proposed by Braun et al. to prove feasibility and effectiveness of the algorithm.

Key words: grid computing;job scheduling;Min-Min scheduling algorithm;load balancing

1 引　言

网格[1]是构建在互联网上的新兴技术,它将地理上分布的大量异构高速互联网、高性能计算机、大型数据库、传感器、远程设备等融为一体,为用户提供更多的资源、功能和交互性.网格资源管理是网格研究的核心内容,主要包含资源注册、作业请求、资源发现、资源调度和作业执行等几个重要方面,一个网格资源管理系统的优劣直接影响网格系统的性能.

在并行计算平台中,资源调度是一个 NP 问题.资源调度就是在满足用户要求的前提下,将任务分配给资源,实现任务集到资源集的映射.由于网格资源具有分布性、动态性、异构性和管理多重性的特征[2],网格资源调度变得复杂.评价网格资源调度的标准主要有时间跨度(Makespan),负载平衡(Load Balancing),服务质量(QoS,Quality of Service),经济原则(Economic Principals).

在现有的启发式网格资源调度算法中,经典的 Min-Min 算法[3]取得了良好的性能.由于 Min-Min 算法采用优先调度短作业给强计算能力资源的策略,造成强计算能力资源负载过重,弱计算能力资源相对空闲,导致整体资源负载不均,资源利用率低[4].在网格计算中,网格资源调度算法的目标-提高资源利用率、最小化任务集的 Makespan.本文提出基于 Min-Min 网格资源调度的优化算法—OPT-Min-Min(Optimize-Min-Min)算法,均衡负载、提高资源利用率,最小化任务集的 Makespan,提高网格系统性能,达到用户的满意度.

OPT-Min-Min 调度算法,即基于 Min-Min 网格资源调度算法的优化.针对 Min-Min 网格资源调度算法存在负载不平衡的缺点,OPT-Min-Min 调度算法采用了二次调度的思想均衡负载.具体分为两个阶段.第一阶段:采用 Min-Min 算法进行预调度;第二阶段:根据减小 Makespan 的条件,在第一阶段预调度的基础上,将取得 Makespan 的资源的任务分配给相对空闲的资源来负载均衡,提高资源利用率,最小化 Makespan.

2 相关工作

目前,已有大量的启发式网格资源调度算法,均是将元任务集合映射到异构的资源集合.元任务集合是指没有任何相互任务依赖的独立的任务集合.下面介绍比较常见的几种调度算法.

2.1 Opportunistic Load Balancing 算法

收稿日期:2013-05-17　收修改稿日期:2013-08-15　基金项目:河北省自然科学基金项目(F2012203087)资助;国家自然科学专项基金项目(61040023)资助;国家自然科学基金项目(61073060)资助.　作者简介:张忠平,男,1972 年生,博士,博士后,教授,CCF 高级会员,主要研究方向为网格技术、数据挖掘、半结构化数据等;温利娟,女,1987 年生,硕士研究生,主要研究方向为网格技术.

随机负载平衡调度算法 OLB(Opportunistic Load Balancing),其调度原则是将调度任务分配给下一个最早可用空闲资源,如果有多个可用空闲资源,则随机选择其中一个. OLB 并没有考虑任务在该资源上的预期执行时间 ETC(Expected Time to Compute)[5,6],导致系统的吞吐量差.

2.2 Minimum Execution Time 算法

最小执行时间算法 MET(Minimum Execution Time),属于在线模式调度,即一次仅考虑一个任务,MET 的调度策略是将任务分配给具有最小执行时间的资源,并没有考虑资源的可用性和当前负载,导致了资源负载不平衡和资源利用率低.

2.3 Minimum Completion Time 算法

最小完成时间算法 MCT(Minimum Completion Time),调度策略类似于 MET,不同之处在于完成时间而不是执行时间,完成时间是资源的可用时间和任务在资源上执行时间的和,MCT 将作业分配给具有最小完成时间的资源.

2.4 Min-Min 算法

Min-Min 调度算法不同于 MCT 和 MET,属于批处理调度模式,开始于一个未映射的任务集,首先根据 ETC 矩阵和资源的可用时间计算出每个任务在每个资源上的完成时间,其次计算出每个任务的最早完成时间,将具有全局最小的最早完成时间的任务分配给相应的资源,删除该任务,更新资源的可用时间. 在剩下的任务集中重复执行该操作直到任务全部分配.

2.5 Max-Min 算法

Max-Min 调度算法与 Min-Min 算法类似,由于 Min-Min 算法总是优先调度短任务,导致长任务的等待时间的增加,不能及时执行,在短任务的数量多于长任务时,我们可以优先调度长任务,降低长任务的等待时间. Max-Min 就是优先调度长任务给计算能力强的资源. 但是在长任务数量多的情况下,由于先调度长任务,可能增大系统完成时间.

2.6 $QPS_{Max\text{-}Min<>Min\text{-}Min}$ 算法

$QPS_{Max\text{-}Min<>Min\text{-}Min}$ 算法,即基于 QoS 的预测性 Max-Min, Min-Min 选择算法[7],由于网格中的资源大部分不是专用资源,在执行网格中任务的同时,执行自己的内部的任务,所以在网格资源调度中引入了预测机制来更好的衡量网格中任务的执行时间. 由于已知的 Min-Min, Max-Min 的调度算法分别适用于短任务少和长任务少的情况, 因此提出了 Max-Min, Min-Min 选择算法,该算法的调度策略是首先计算任务集的执行时间的标准差 SD(Standard Deviation),若任意两个作业的执行时间的差大于标准差出现在前半段,则用 Min-Min 算法,否则采用 Max-Min 算法.

2.7 Min-mean 算法

Min-mean 算法,即基于资源执行时间平均值 meanCT[8,9] 的 Min-Min 算法,该算法分为两个阶段,第一阶段采用 Min-Min 算法预调度,然后将执行时间大于 meanCT 的资源定义为重负载的资源,将重负载资源上的任务重新分配给其他资源,来平衡负载,提高资源利用率.

在上述网格资源调度算法中,Min-Min 算法凭借其自身的优点取得了良好的性能. Min-Min 算法优先调度任务集中最短作业给强计算能力的资源,导致分配给强计算能力资源任务多,弱计算能力资源相对空闲,导致负载不平衡. Min-mean 调度算法采用二次调度的思想均衡负载,但是 Min-mean 调度算法进行二次调度条件-资源执行时间的平均值 meanCT,设置不合理,不能很好的均衡负载. OPT-Min-Min 调度算法,首先采用二次调度的思想,其次根据最小化 Makespan 的标准设定二次调度的条件,对 Min-Min 调度算法的进行改进,性能优越于上述所有的算法.

3 OPT-Min-Min 算法

3.1 问题定义

网格资源调度就是在异构网格环境中实现任务集到资源集的映射. 由于映射机制的 NP 完备性特性,需要在合理的代价下找到可接受的解. 本文是基于 Braun 等人[10,11] 提出的仿真模型基准. 在该基准下,采用的是元任务的静态映射,一个资源只能执行一个任务,任务在资源上采用非抢占式按照分配顺序依次执行. 采用元任务静态映射机制,要求必备的前提条件-元任务以及资源的数量已知,ETC 矩阵已知.

本文提出的网格资源调度算法是基于下述的假设条件:

1)应用程序的执行是由不可再分的、任务之间无任何通信的独立任务组成,即元任务[10] ,任务没有截止时间和优先权限制.

2)异构环境中参与分配任务的可用资源的数量是已知的.

3)提交给异构环境的元任务的数量是已知的.

4)提交给异构环境的每个任务的工作量是已知的.

5)异构环境中每个资源的计算能力及网络带宽是已知的.

6)资源在执行完上一个任务到执行下一任务的就绪时间是已知的.

7)ETC(m* n)矩阵是已知的,给定 m 个任务,在 n 个资源上的执行时间. 矩阵元素 ET_{ij} 表示任务 T_i 在资源 R_j 上的预期执行时间;矩阵的行代表一个任务在所有资源上的预期执行时间;矩阵的列代表所有任务在一个资源上的预期执行时间.

在上述假设前提之下,网格资源调度问题可以描述如下:

1)提交给网格资源调度器的任务集{T_1,T_2,T_3,…,T_m}.

2)在任务提交时可用资源集{R_1,R_2,R_3,…,R_n}.

3)Makespan = max(CT_j),$CT_j = \sum_i CT_{ij}$,$CT_{ij} = r_j + ET_{ij}$,Makespan 表示任务集的完成时间;CT_j 表示资源 R_j 的执行时间;ET_{ij} 任务 T_i 在资源 R_j 上的预期执行时间,预期执行时间可以根据任务的指令和数据量以及资源的计算速率和网络带宽求得;r_j 资源完成前一个任务到可以执行下一个任务的就绪时间.

4)本文采用 Makespan 是评价调度算法优劣的主要指标之一,OPT-Min-Min 调度算法的主要目标就是减少 Makespan.

3.2 OPT-Min-Min 调度算法描述

OPT-Min-Min 算法是对 Min-Min 调度算法的优化. 针对 Min-Min 调度算法存在负载不平衡,第一个阶段采用 Min-Min 算法进行预调度,第二阶段通过二次调度将重负载资源的任务分配给轻负载资源执行来均衡负载.

OPT-Min-Min 算法分为两个阶段. 第一阶段采用 Min-Min 调度算法预调度任务.

1)找到最小任务最早完成时间及产生最小最早完 成时间的资源,将该任务分配给该资源调度,删除该任务,更新资源的可用时间.

2)在剩下的任务集中重复执行步骤(1),直到所有的任

务全部分配.

综上,Min-Min 调度算法总是优先调度短任务给强计算能力资源,导致强计算能力的资源负载重而弱计算能力的资源空闲,但是相比于其他的算法,Min-Min 算法产生较小的 Makespan.

第二阶段将重负载资源的任务分配给轻负载资源. OPT-Min-Min 算法将第一阶段采用 Min-Min 算法取得 Makespan 的资源定义为重负载的资源,其中 $Makespan = \max(CT_j)$.

1)取出分配给该重负载资源的任务集并将任务按执行时间升序排序.

2)假设将任务集中最小的任务重新分配给其它所有资源,更新该重负载资源及其它所有资源的执行时间.

①如果全部资源的最大执行时间小于 Makespan,则将该任务分配给执行时间最小的资源.

②如果所有资源的最大执行时间大于等于 Makespan,则考虑该任务集中的其他任务,将能够分配出去的任务重新分配给其它资源.

若①②中有任务分配出去则更新所有资源的执行时间,重新取得 Makespan.

(3)重复(1)(2)过程,直到取得 Makespan 的资源任务集中的任务不能再重新分配出去为止. 此时算法结束,取得最终的 Makespan.

3.3　OPT-Min-Min 调度算法

根据 3.2 节的算法思想描述,给出 OPT-Min-Min 调度算法描述如算法 1.

算法 1. OPT-Min-Min 调度算法.

输入:所有的任务 $T_i(i=1,2,\cdots,m)$,资源 $R_j(j=1,2,\cdots,n)$,

ETC(m* n)矩阵

输出:完成时间 Makespan,任务调度表 Table

OPT-Min-Min (T_i,R_j,ETC)

BEGIN

// Min-Min 算法进行预调度

1)FOR 对于所有的任务 T_i

2)　FOR 对于所有的资源 R_j

3)　　$CT_{ij} = ET_{ij} + r_j$

4) END FOR

5)END FOR

6)DO UNTIL 所有的任务都映射

7)　对于每一个任务找到最早完成时间和达到最早完成时间的资源

8)　找到任务 T_k 具有最小的最早完成时间,并且给 T_k 分配取得最小最早完成时间的资源 R_f

9)　从列表中删除 T_k

10)　更新 R_f的准备时间

11)　对于所有的 i 更新 CT_{if}

12) END DO

// 将负载过重的资源的任务重新调度

13)将资源按照执行时间 CT_j 升序排序

14)DO WHILE 具有 Makespan 的资源 R_{maxCT}有任务可以分配出去

15)　将 R_{maxCT}上的任务按照执行时间升序排序

16)　FOR 对于分配到资源 R_{maxCT}所有的任务 T_{maxCT}

17)　　FOR 其它的 n-1 个资源

18)　　IF($CT_j + ET < Makespan$ && $CT_j + ET$ 最小)

19)　　重新调度 T 在资源 R_j 并且更新 CT,

20)　　END FOR

21)　　IF 有任务分配出去,跳出循环

22)　END FOR

23)将资源按照执行时间 CT_j 升序排序

24)END DO

25)输出 Makespan 和任务调度表 Table

END

根据上述 OPT-Min-Min 算法可得,1)-12)属于第一阶段:采用 Min-Min 算法预调度,由于 Min-Min 调度算法导致负载不均衡,资源利用率低. 采用算法中的 13)-24)进行二次调度,开始第二阶段:首先将资源按照执行时间升序排序,取出具有 Makespan 的资源 R_{maxCT}上的任务集 T_{maxCT},按照算法中的 18),如果将该任务集中的任务 T 分配到其它 n-1 资源,能减少 Makespan,则将 T 分配出去,如果有多个资源同时满足能减少 Makespan ,则选中执行时间最小的资源进行分配,然后再将资源按照执行时间再次升序排序,执行循环 14)-24),直到当前取得 Makespan 资源 R_{maxCT}上的任务不再能分配出去. 则得到最终的 Makespan.

4　实例分析

本文将通过如下实例来验证提出的 OPT-Min-Min 调度算法相对于 Min-Min 调度算法的有效性.

假设通过网格资源查找得到网格环境中可用的资源集$\{R_1,R_2,R_3\}$,提交给网格资源管理器的元任务集合$\{T_1,T_2,T_3,T_4,T_5\}$,网格调度器可以调度元任务集合中的$\{T_1,T_2,T_3,T_4,T_5\}$给资源$\{R_1,R_2,R_3\}$. 所有任务的预期执行时间可以通过任务的指令和数据量以及资源的计算速率和网络带宽获得,进而给出任务的预期执行时间-ETC 矩阵,如表 1 所示.

表 1　ETC 矩阵

Table 1　ETC matrix

	R_1	R_2	R_3
T_1	1	2	3
T_2	2	4	5
T_3	3	7	8
T_4	4	11	12
T_5	5	16	17

第一阶段:采用 Min-Min 调度算法进行预调度的结果如图 1 所示.

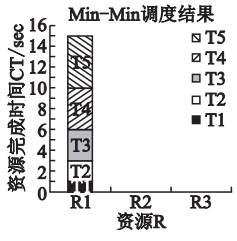


图 1　Min-Min 预调度的结果

Fig. 1　Results of Min-Min algorithm

由图 1 可知,用 Min-Min 算法进行预调度的结果是只在资源 R1 上执行所有的任务,Makespan = 15sec.

第二阶段:此时,资源 R_1 的执行时间为 15sec,R_2,R_3 的执行时间为 0sec,将资源集$\{R_1,R_2,$

R_3}按照执行时间升序排序得{R_2,R_3,R_1},取 R_1 上的任务集{T_1,T_2,T_3,T_4,T_5},根据算法 OPT-Min-Min 中的步骤 18)可以将 T_1 分配给 R_2,得到图 2 所示的结果.

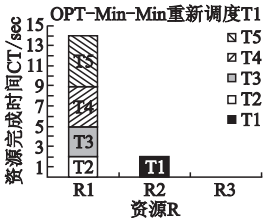


图 2 OPT-Min-Min 重新调度 T1

Fig. 2 OPT-Min-Min Algorithm Recheduling of T1

此时,资源 R_1 的执行时间为 14sec,R_2 的执行时间为 2sec,R_3 的执行时间为 0sec,将资源集{R_2,R_3,R_1}按照执行时间升序排序得{R_3,R_2,R_1},取 R_1 上的任务集{T_2,T_3,T_4,T_5},根据 OPT-Min-Min 算法中的步骤 18)可以将 T_2 分配给 R_3,得到图 3 所示的结果.

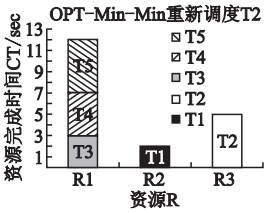


图 3 OPT-Min-Min 重新调度 T2

Fig. 3 OPT-Min-Min algorithm recheduling of T2

此时,资源 R_1 的执行时间为 12sec,R_2 的执行时间为 2sec,R_3 的执行时间为 5sec,然后将资源集{R_3,R_2,R_1}按照执行时间升序排序得{R_2,R_3,R_1},取 R_1 上的任务集{T_3,T_4,T_5},根据算法 OPT-Min-Min 中的步骤 18)可以将 T_3 分配给 R_2,得到图 4 所示的结果.

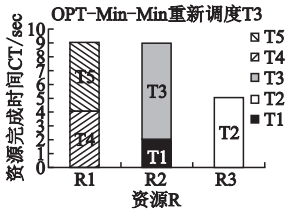


图 4 OPT-Min-Min 重新调度 T3

Fig. 4 OPT-Min-Min algorithm recheduling of T3

此时,资源 R_1 的执行时间为 9sec,R_2 的执行时间为 9sec,R_3 的执行时间为 5sec,然后将资源集{R_2,R_3,R_1}按照执行时间升序排序得{R_3,R_2,R_1},取 R_1 上的任务集{T_4,T_5},由于 R_1 上的任务不再可以分配出去,所以算法 OPT-Min-Min 结束,得到图 4 的结果,Makespan = 9sec < 15 sec.

5 实验结果和性能分析

本文采用网格资源调度算法中普遍采用的基准-Braun 等人[10,11] 提出的网格仿真模型基准. 该基准根据网格中任务与资源的异构性,ETC 矩阵一致性,模拟网格环境中 12 种情况下的 OPT-Min-Min、Min-Min、Max-Min、Min-mean 四种常见的批处理模式网格资源调度算法.

5.1 实验基准及实验环境

实验基准 12 种情况用 u-x-yyzz 表示.

1)u-均匀分布产生 ETC 矩阵

2)x-ETC 矩阵的类型,(c-一致性矩阵,s-半一致性矩阵,i-非一致性矩阵)

①ETC 矩阵是一致性矩阵,满足若资源 R_j 执行任意的任务 T_i 快于 R_k,则资源 R_j 执行所有的任务均快于 R_k.

②ETC 矩阵是半一致性矩阵,则 ETC 包含一个一致性子矩阵.

③ETC 是非一致性矩阵,满足资源 R_j 执行一部分任务快于 R_k,一部分任务慢于 R_k.

3)yy-任务的异构性(hi-代表高异构性,lo-代表低异构性)

任务的异构性指的是一个给定资源,任务执行时间的变化

4)zz-资源的异构性(hi-代表高异构性,lo-代表低异构性)资源的异构性指的是给定一个特定的任务,在全部资源上执行时间的变化.

实验环境

台式计算机一台,系统为 Microsoft Windows XP Professional,版本 2002 Service Pack 3,Inter(R) Core(TM)2 Duo CPU E4500 @2.20GHz 2.19GHz,2.00GB 内存.

5.2 性能分析

实验是在上述 Braun 等人提出的仿真模型基准与实验环境下进行的,该基准下的 ETC 矩阵是基于 512 个任务和 16 个资源均匀分布产生的,在此模拟标准之下 OPT-Min-Min、Min-Min、Max-Min、Min-Mean 调度算法的执行结果如表 2 所示.

表 2 OPT-Min-Min 算法与其他算法对比实验结果

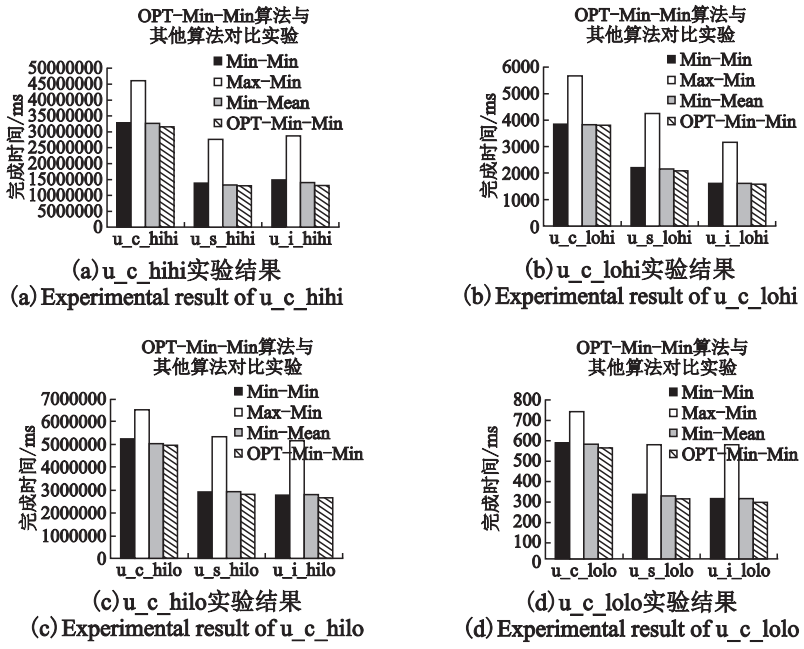
Table 2 Experimental Result of OPT-Min-Min algorithm and other algorithms

Instances	OPT-Min-Min(ms)	Min-Min (ms)	Max-Min (ms)	Min-Mean (ms)
u_c_hihi	3.1348648 ×10^7	3.2693322 ×10^7	4.583936 ×10^7	3.2260172 ×10^7
u_c_hilo	4.973211 ×10^6	5.274453 ×10^6	6.540184 ×10^6	5.037062 ×10^6
u_c_lohi	3.8217134 ×10^3	3.8765923 ×10^3	5.701304 ×10^3	3.829216 ×10^3
u_c_lolo	5.6009717 ×10^2	5.825626 ×10^2	7.3758966 ×10^2	5.761336 ×10^2
u_s_hihi	1.2843172 ×10^7	1.3765 ×10^7	2.7531366 ×10^7	1.2944868 ×10^7
u_s_hilo	2.842350 ×10^6	2.947291 ×10^6	5.334912 ×10^6	2.9298985 ×10^6
u_s_lohi	2.0907397 ×10^3	2.1972473 ×10^3	4.2852617 ×10^3	2.1578416 ×10^3
u_s_lolo	3.0420825 ×10^2	3.2438898 ×10^2	5.8201276 ×10^2	3.172402 ×10^2
u_i_hihi	1.3035405 ×10^7	1.4579022 ×10^7	2.81784 ×10^7	1.3793677 ×10^7
u_i_hilo	2.676013 ×10^6	2.797357 ×10^6	5.1531965 ×10^6	2.7932002 ×10^6
u_i_lohi	1.5806067 ×10^3	1.619184 ×10^3	3.196768 ×10^3	1.619184 ×10^3
u_i_lolo	2.912936 ×10^2	3.0633313 ×10^2	5.786073 ×10^2	3.0356866 ×10^2

根据表 2 给出的 OPT-Min-Min、Min-Min、Max-Min 与

Min-Mean 调度算法的调度结果,给出图 5.

由以上的实验结果可以看出,在批处理调度模式中,Min-Min 调度算法的性能明显优越于 Max-Min 调度算法,但由于 Min-Min 调度算法存在负载不平衡的弊端.



(a) u_c_hihi实验结果
(a) Experimental result of u_c_hihi

(b) u_c_lohi实验结果
(b) Experimental result of u_c_lohi

(c) u_c_hilo实验结果
(c) Experimental result of u_c_hilo

(d) u_c_lolo实验结果
(d) Experimental result of u_c_lolo

图 5　OPT-Min-Min 算法与其他算法对比实验结果

Fig. 5　Experimental Result of OPT-Min-Min algorithm and other algorithms

本文提出的 OPT-Min-Min 调度算法针对 Min-Min 调度算法存在的负载不平衡,采用了二次调度的思想;通过将重负载资源的任务分配给轻负载资源执行来均衡负载,取得了更小的完成时间.

6　总　结

在众多网格资源调度算法的研究中,Min-Min 调度算法基于其比较简单的特性,在网格中取得了良好的调度结果,但是 Min-Min 调度算法存在负载不平衡的缺点. Min-Mean 调度算法采用了二次调度的思想,性能优越于 Min-Min 调度算法. 本文提出的 OPT-Min-Min 调度算法,在采用二次调度的基础上,将重负载资源的任务分配给轻负载资源执行来均衡负载,取得更小的完成时间. 在理论和实验的基础上证明,OPT-Min-Min 调度算法性能优越于 Min-Min、Max-Min、Min-Mean 调度算法. 未来的研究将考虑更多的网格中的因素,例如服务质量、经济原则等,并在此基础上进一步优化算法.

References:

[1] Ian Foster, Carl Kasselman, Steven Tuecke. The anatomy of the grid [J]. Intl J. Supercomputer Application, 2001, 11(3): 181-205.

[2] Zhan Xiao-su, Zhang Shao-hua (Translated). Gird computing [M]. Beijing: Tsinghua University Press, 2005: 21-36.

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

[4] Kokiavani T, Dr D l. George amalarethinam. load balanced Min-Min algorithm for static meta-task scheduling in grid computing [J]. International Journal of Computer Application, 2011, 20(2): 43-49.

[5] Armstrong R, Hensgen D, Kidd T. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions [C]. In 7th IEEE Heterogeneous Computing Workshop (HCW'98), 1998: 79-87.

[6] Freund R F, Siegel H J. Heterogeneous processing [C]. IEEE Computer, 1993, 26(6): 13-17.

[7] Singh M, Suri P K. $QPS_{Max-Min<>Min-Min}$: a QoS based predictive Max-Min, Min-Min switcher algorithms for job scheduling in a grid [J]. Information Technology Journal, 2008, 7(8): 1176-1181.

[8] Kamalam G K, Murali Bhaskaran V. A new heuristic approach: min-mean algorithm for scheduling meta-tasks on heterogeneous computing systems [J]. International Journal of Computer Science and Network Security International Journal of Computer Science and Network Security, 2010, 10(1): 24-31.

[9] Kamalam G K, Murali Bhaskaran V. New enhanced heuristic min-mean scheduling algorithm for scheduling meta-tasks on heterogeneous grid environment [J]. European Journal of Scientific Research, 2012, 70(3): 423-430.

[10] Tracy D Braun, Howard Jay Siegel, Noah Beck. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems [J]. Journal of Parallel and Distributed Computing, 2001, 61(6): 810-837.

[11] Braun T, Siegel H, Beck N, et al. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing system [C]. In 8th IEEE Heterogeneous Computing Workshop (HCW'99), 1999: 15-29.

附中文参考文献:

[2] 战晓苏, 张少华, 译. 网格计算 [M]. 北京: 清华大学出版社, 2005: 21-36.