文章编号:1001-506X(2010)04-0854-06

安全驱动的实时任务调度遗传算法

朱 海,王宇平
(西安电子科技大学计算机学院,陕西 西安 710071)

摘 要:对异构网格环境下的硬实时任务调度问题,不仅考虑了时间约束而且考虑了其安全性能需求,构造了相应的安全效益函数,在此基础上构建了一个安全驱动的任务调度模型。为了解该模型,设计了新的选择算子使得不满足时间约束但安全效益值大的个体也参与到进化中,从而保证种群多样性;设计了一个能够扩大搜索范围的新的杂交算子和增强搜索精度的调整算子。最后引入一个启发式算子,提出了一种搜索能力较强的安全驱动混合遗传算法(security-driven hybrid genetic algorithm, SDHGA)。仿真实验表明,在同等条件下该算法与经典的 Min-Min、SD-Min-Min、SAREC 和 QoSGA 等算法相比,在任务调度成功率、安全效益值和系统吞吐率等方面具有较好的综合性能。

关键词:任务调度;安全驱动;启发式算子;遗传算法
中图分类号:TP 393 **文献标志码**:A

Security-driven real-time task scheduling based on genetic algorithm

ZHU Hai, WANG Yu-ping
(*School of Computer Science and Technology, Xidian Univ., Xi'an 710071, China*)

Abstract: In terms of hard real-time task scheduling problem under heterogeneous grid environment, both the time constraints and the security performance are considered. Firstly, a security-efficiency function is proposed, and a new security-driven task scheduling model is constructed based on the function. Secondly, a new selection operator is proposed, where some individuals not satisfying time constraints but having a high value of security efficiency are selected. In doing so, the diversity of the population is maintained. Thirdly, a new crossover operator and a local search operator are designed, which can enhance the exploration ability. Finally, a heuristic operator is introduced, and a new security-driven hybrid genetic algorithm (SDHGA) is proposed based on all these operators. The simulation results show that the proposed algorithm is competitive in terms of success ratio, security efficiency value and overall system performance in comparison with the existing algorithms.

Keywords: task scheduling; security-driven; heuristic operator; genetic algorithm

0 引 言

随着计算机网络通信技术的发展,网格计算[1]越来越引起人们的重视。在网格环境中,任务调度系统是其重要的组成部分,它要根据任务信息采用适当的策略把不同的任务分配到相应的资源节点上去运行,实现最佳的调度策略。由于网格任务调度面临的是一个非定常多项式(non-deterministic polynomial, NP)完全问题,引起了众多学者的关注,而网格平台下的实时任务调度问题成为目前网格领域研究的一个热点[2]。

由于网格平台的开放性、动态性、异构性和地理分布性等特点,实时应用在网格平台执行时,保证结果正确的同时不仅对执行时间有要求而且必须面临网络安全对结果的不良影响,保证敏感数据在执行或传输过程中不会被泄密、篡改或冒名顶替等。传统的实时任务调度策略[3-5],虽都比较成功地应用到了实时应用系统中,但通常忽略了任务调度对安全的需求,也没有考虑到节点的开放性及动态性等特征,从而使调度系统难以在开放、动态、真实的网格环境中有效运行。鉴于此,本文对异构网格环境下的硬实时调度问题,首先考虑了实时任务在异构网格环境下执行或数据传输时其对保密性、完整性和真实性等安全性能的需求,设计了相应的安全效益函数,提出了一个安全驱动的任务调度模型。

文献[6-7]主要针对节点的行为不可靠性分别提出了

收稿日期:2008-11-05;**修回日期**:2009-04-07。
基金项目:国家自然科学基金(60873099)资助课题
作者简介:朱海(1978-),男,博士研究生,主要研究方向为网格计算、任务调度、进化算法及人工智能。E-mail:zhu_sea@163.com

一种信任驱动和信任增强的启发式调度算法,由于其缺乏对时间约束的表示和处理,并不适合应用于网格平台下的实时调度系统中。文献[8-9]对同构网格环境下的实时任务调度应用融入了安全因素,并提出了一种融入安全的实时任务调度启发式(security-aware real-time heuristic strategy for clusters, SAREC)算法。然而,由于调度问题为约束优化问题,SAREC 并没有在全局范围内比较它的解,特别是在大规模网格环境下很容易陷入局部最优,难以找到全局最优调度策略。鉴于此,在新模型基础上设计了新的选择算子使得不满足时间约束但安全效益值大的个体也参与到进化中,相对于传统模型只选择满足时间约束的个体而丢弃安全值大的个体来讲,保证了种群的多样性;同时设计了一个能够扩大搜索范围的杂交算子和增强搜索精度的调整算子;最后引入一个启发式算子,提出了一种搜索能力较强的安全驱动混合遗传算法(security driven hybrid genetic algorithm, SDHGA)。仿真实验表明,在同等条件下 SDHGA 与经典的 Min-Min, SD-Min-Min(security driven Min-Min)和 SAREC 等算法相比,在任务调度成功率、安全效益值和系统吞吐率等方面具有较好的综合性能。

1　任务调度模型

在提高网格任务调度安全性能的同时,往往意味着要降低时间性能,因而量化达到不同安全性能的时间花费就显得非常重要。不失一般性,本节主要考虑网格安全中广泛涉及的保密性、完整性和真实性等安全性能需求,设计了相应的时间花费及安全效益函数,在此基础上构建了安全驱动的任务调度新模型。

1.1　安全效益函数

1.1.1　保密性

保密性一般主要依赖不同加密算法来实现,其目的是预防任务在网格环境中执行或传输时暴露给未经授权的其他用户或进程,避免信息泄露。而加密算法主要是对应用任务的执行文件或数据进行加密,其时间花费由所采用的加密算法类型、需要加密的数据量及机器性能共同决定。

异构网格环境下,任一台机器可以拥有不同类型的加密算法,而加密算法根据其性能特性 μ_k^e(k 为加密算法类型标识)设定为相应的加密安全水平值 SL_k^e。加密算法的保密性安全水平值与它的性能成反比例关系,安全水平值越低其性能一般越高。这也符合一般逻辑思维,在同等条件下,不会选一个安全水平低且其性能也低的加密机制。若设任务 t_i 需加密的数据量为 l_i,其在机器 p_j 上获得的加密安全值若为 $SV^e(i,j)$,对应的加密算法性能为 $P(SV^e(i,j))$,则其相应的时间花费 $ST(SV^e(i,j))$为

$$ST(SV^e(i,j)) = l_i/P(SV^e(i,j)) = l_i/P(SL_k^e) = l_i/\mu_k^e \tag{1}$$

1.1.2　完整性

完整性服务主要确保任务在网格平台下执行时不会被非法的用户修改或篡改,一般主要通过不同的 Hash 函数来实现。异构网格环境下,任务的完整性时间花费一般由所采用的 Hash 函数类型、需要处理的数据量及机器性能共同决定。

每个 Hash 函数根据其性能特性 μ_k^g(k 为 Hash 函数类型标识)设定为相应的完整性安全水平值 SL_k^g。若设任务 t_i 需 Hash 函数处理的数据量为 l_i,其在机器 p_j 上获得的完整性安全值若为 $SV^g(i,j)$,对应 Hash 函数性能为 $P(SV^g,(i,j))$,则其相应的时间花费 $ST(SV^g(i,j))$为

$$ST(SV^g(i,j)) = l_i/P(SV^g(i,j)) = l_i/P(SL_k^g) = l_i/\mu_k^g \tag{2}$$

1.1.3　真实性

为保证所处理的任务是从合法的用户提交的,确保数据真实性或一致性,一般使用不同的数字签名技术对通讯实体身份的真实性进行鉴别。由于每一种认证技术在特定环境下的时间花费为一常量,与所处理的数据量大小无关,因此真实性的时间花费主要由所采用的不同签名技术和机器的性能决定。

异构网格平台下,每种签名技术根据性能特性 μ_k^a(k 为签名技术类型标识)设定为相应的真实性安全水平值 SL_k^a。任务 t_i 在机器 p_j 上获得的真实性安全值若为 $SV^a(i,j)$,则其对应认证技术时间花费 $ST(SV^a(i,j))$为

$$ST(SV^a(i,j)) = P(SV^a(i,j)) = P(SL_k^a) = \mu_k^a \tag{3}$$

1.1.4　安全效益函数

综上实时任务在网格平台下执行时对保密性、完整性和真实性等安全需求的考虑,可得到对任意任务 t_i 在机器 p_j 上获得的综合安全效益值为

$$SV(i,j) = \omega_1 SV^e(i,j) + \omega_2 SV^g(i,j) + \omega_3 SV^a(i,j) \tag{4}$$

式中,ω_k 是任务 t_i 对应的一组权值,满足 $\sum_{k=1}^{3}\omega_k = 1, \omega_k \geqslant 0$,$\omega_k$ 表示不同安全服务的权重,权重越大表示其相应的安全服务越重要,在提高安全值时优先考虑;$SV^r(i,j)$可由 $SL_j^r(k)$得到,详见文献[8],易知 $0 \leqslant SV^r(i,j) \leqslant 1, r \in \{e,g,a\}$。

1.2　安全驱动的任务调度模型

定义 1　任务模型为一个四元组$(\boldsymbol{T},a,f,d)$,其中 $\boldsymbol{T}$ 为含时间约束和安全需求的实时任务集,a 为任务集的到达时间,f 为完成时间,d 为最终期限。

定义 2　任务集 $\boldsymbol{T}$ 由 n 个独立任务组成,其中任务 $\boldsymbol{t}_i(i=1,2,\cdots,n)$为一个二元组,即 $\boldsymbol{t}_i=(\boldsymbol{E}_i,\boldsymbol{S}_i)$。其中,$\boldsymbol{E}_i$ 和 $\boldsymbol{S}_i$ 分别为 $\boldsymbol{t}_i$ 的执行时间向量和安全需求向量,$\boldsymbol{E}_i=(e_i^1, e_i^2,\cdots,e_i^m)$表示任务 $\boldsymbol{t}_i$ 在异构系统不同处理器上的执行时间;$\boldsymbol{S}_i=(s_i^1,s_i^2,\cdots s_i^q)$表示任务 $\boldsymbol{t}_i$ 的 q 个不同安全需求。

定义 3　处理器的性能表示处理器运行速度的快慢,用 ρ_i 表示。运行速度越高,处理器执行任务的时间就越短。在分布式异构系统中,可选择一个处理器作为标准处理器,将其执行速度定义为 1,其他处理器的执行能力计算式为

$$\rho_j = C(i,normal)/C(i,j) \tag{5}$$

式中,$C(i,normal)$为任务 $\boldsymbol{t}_i$ 在标准处理器上的执行时间,$C(i,j)$为任务 $\boldsymbol{t}_i$ 在处理器 $\boldsymbol{p}_j$ 上的执行时间。

定义 4 处理器模型可描述为一个异构处理器集 $\boldsymbol{P}=\{p_1,p_1,\cdots,p_m\}$，任一处理器 $\boldsymbol{p}_j$ 为一个二元组，$\boldsymbol{p}_j=(\rho_j,\boldsymbol{S}_j)$，其中 ρ_j 为处理器的执行能力，$\boldsymbol{S}_j$ 为处理器提供的安全服务向量，即 $\boldsymbol{S}_j=(s_j^1,s_j^2,\cdots,s_j^q)$。

定义 5 任务集 $\boldsymbol{T}$ 中所有任务在处理器集 $\boldsymbol{P}$ 上的预期计算时间可由一个 $n\times m$ 矩阵 $\boldsymbol{ET}$ 表示，其中任一元素 $ET(i,j)$表示任务 $\boldsymbol{t}_i$ 在机器 $\boldsymbol{p}_j$ 上的预期计算时间。任务 $\boldsymbol{t}_i$ 所在节点与机器 $\boldsymbol{p}_j$ 间通信时间 $UT(i,j)$计算式为

$$UT(i,j)=b_{ij}^0+m\boldsymbol{t}_i/b_{ij}^1 \tag{6}$$

式中，b_{ij}^0 为链路连接建立时间；$m\boldsymbol{t}_i/b_{ij}^1$ 为数据传输时间。

定义 6 任务集 $\boldsymbol{T}$ 中所有任务在处理器集 $\boldsymbol{P}$ 上获得的安全效益时间花费可由一个 $n\times m$ 矩阵 $\boldsymbol{ST}$ 表示，其中任一元素 $ST(i,j)$表示任务 $\boldsymbol{t}_i$ 在机器 $\boldsymbol{p}_j$ 上获得安全效益时间花费，计算式为

$$ST(i,j)=\sum_{k\in\{a,e,g\}}\boldsymbol{ST}(SV^k(i,j)) \tag{7}$$

定义 7 安全驱动的实时任务调度模型是由任务模型、处理器模型和分配调度方案组成，可表示为一个六元组 $(\boldsymbol{T},a,f,d,\boldsymbol{P},\boldsymbol{MAP})$，其中 $\boldsymbol{MAP}$ 为分配调度方案的集合，$\boldsymbol{MAP}=\{map_1,map_2,\cdots,map_n\}$。分配调度方案为二元组，$map=(a,\varphi)$，其中 a 表示将任务集 $\boldsymbol{T}$ 中 n 个实时任务映射到处理器集 $\boldsymbol{P}$ 上的分配方案；φ 表示单个处理器 $\boldsymbol{t}_i$ 上的调度方案。则异构系统下实时安全调度问题即为寻找一种最优分配调度方案，将任务集 $\boldsymbol{T}$ 中所有任务合理地分配到处理器集 $\boldsymbol{P}$ 上调度执行，在满足时间约束条件下，使任务调度获得的安全效益值最大。其数学模型表示为

$$\begin{cases}\max\ \sum_{i=1}^{n}\sum_{j=1}^{m}x_{ij}f(i,j)\\ \text{s.t.}\quad x_{ij}=\begin{cases}1,\text{ 任务 }\boldsymbol{t}_i\text{ 被分配到机器 }\boldsymbol{p}_j\text{ 上执行}\\0,\text{ 其他}\end{cases}\\ \sum_{j=1}^{m}x_{i,j}=1,\ i=1,2,\cdots,n\\ \forall t_i\in p_j,\ Starttime(i,j)+CT(i,j)+ST(i,j)\leqslant d_i\\ Starttime(i,j)=a(j)+\sum_{t_k\in p_j,d_k\leqslant d_i}[CT(k,j)+ST(k,j)]\end{cases} \tag{8}$$

式中，$f(i,j)$表示实时任务 $\boldsymbol{t}_i$ 分配到机器 $\boldsymbol{p}_j$ 在满足时间约束的条件下获得的最优安全效益值，该值可通过优化式(4)中的函数 $SV(i,j)$获得，具体优化过程见算法设计部分；$CT(i,j)$表示任务 $\boldsymbol{t}_i$ 在 $\boldsymbol{p}_j$ 上的执行时间，由通信时间 $UT(i,j)$和计算时间 $ET(i,j)$之和组成。任务 $\boldsymbol{t}_i$ 的开始执行时间 $Starttime(i,j)$为机器可用时间 $a(j)$和完成分配给它的所有满足时间约束的任务所用时间(执行时间及安全效益花费时间)之和，任务 $\boldsymbol{t}_i$ 在机器 $\boldsymbol{p}_j$ 上获得安全效益值的时间花费 $ST(i,j)$可由式(7)获得。

2 算法设计

任务调度面临的是一个 NP-难问题，而基于进化理论的遗传算法[10-11]非常适合解决复杂优化问题。下面分别给出选择算子、交叉算子、启发式算子的设计及算法伪代码。

2.1 选择算子

选择算子采用改进轮盘赌的选择方式，一个调度所获得的安全效益值越大，它的适应值就越好。由于传统的轮盘赌选择方式只选择满足约束条件的个体，本文采用精英保留策略，除了选择满足时间约束的个体外，还选择一些安全效益值大但不满足时间约束的个体，增强了种群的多样性。

2.2 交叉算子

为了增强搜索的广度，设计了一种新的杂交算子，即对父代中的两个个体对应位中的值相加取模运算求余，得到一个后代。如 $n=10$，$m=4$ 的情况，交叉过程如图 1 所示。

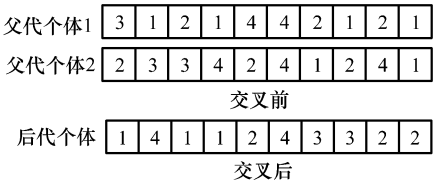

图 1 交叉算子操作前后示意图

2.3 调整算子

为了使更多任务可以满足时间约束以达到提高任务调度成功率和安全效益值的目的，设计了一个调整算子。具体的调整方式如下：首先，把该染色体映射成机器调度，计算每台机器上所分配任务的调度成功率，并将机器按照成功率的高低排序；然后选择其中成功率最低的机器 p_l，对机器 p_l 上的每一个任务，如果将该任务迁移到随机产生的一台其他机器 p_r 上执行，在满足时间约束条件下如果二者总成功率得到提高，将该任务调整到机器 p_r 上执行，得到对应的染色体编码，否则，不作调整操作。安全效益值的调整方法类似。

2.4 启发式算子

引入启发式算子的目的是为了增强算法的局部深度搜索能力，使染色体的安全效益值提升到最大。

启发式算法伪代码：

步骤 1 初始化机器累加时间 $s=0$，当前机器数 $j=1$，机器总数 machine_num，利用排序函数将每台机器上的任务按预执行时间长短排序；

步骤 2 初始化当前任务数 $i=1$，从第 j 台机器上的第 i 个任务开始执行；

步骤 3 首先，判断 if($j<$machine_num)，如果条件为假，则结束程序；否则，获取该任务的编号 k 及第 j 台机器上任务总数 task_num。其次，判断 if($i>$task_num)，如果条件为真，则 $i=1$，转到步骤 3；否则，判断 if(cost[j][k]+$s<=$timelimit[k])，如果条件为假，则当前机器数 j++，转到步骤 2，否则，则获取安全值提高一级的花费时间 sec_time。再其次，判断 if(cost[j][k]+sec_time+$s<=$timelimit[k])，如果条件为真，则该任务的安全值级别 task_sec

[k]提高一级,当前任务数 i++,机器累加时间 s+=sec_time+cost[j][k]。如果条件为假,则当前机器数 j++,转到步骤 2。

2.5　安全驱动的遗传算法(SDHGA)

SDHGA 算法伪代码:

步骤 1　采用直接整数编码的方式[12],用随机算法生成规模为 N 的初始种群;

步骤 2　对初始种群进行概率为 p_c 的交叉操作,产生交叉后代种群记为 N_c;

步骤 3　对初始种群进行概率为 p_m 的单点变异操作,产生变异后代种群记为 N_m;

步骤 4　选择当前所有后代个体集合 $N \cup N_c \cup N_m$ 中调度成功率最好个体,调用启发式算子将其安全值最优,产生一个新的个体;

步骤 5　将步骤 4 中产生的新个体加入到下一代进化中,对所有后代个体运用适应度函数 $Fitness(q_i)$,采用精英保留策略,对新种群中的个体按改进轮盘赌选择算子进行选择产生 $N-1$ 个个体,组成规模为 N 的下一代种群;

步骤 6　判终止条件,如果没有满足,转步骤 2 。

3　仿真实验及结果性能分析

本节首先给出仿真实验环境及参数设置,而后将本文提出的基于安全驱动的任务调度遗传算法 SDHGA 与经典的 Min-Min[13]、SD-Min-Min[6] 和 SAREC[8-9] 等启发式算法比较,验证本文算法在实时任务调度成功率、系统吞吐率和安全效益值等方面的有效性。另外在不同的通信/计算比(conmunication compute ratio, CCR)环境下,将本文算法与性能驱动 QoSGA[12] 相比,验证其对任务调度安全效益值的改善情况。

3.1　实验环境及算法参数设置

为了评估本文提出的算法,基于 C++ 语言设计开发了一个由组件 GridGenerator 和组件 TaskGenerator 构成的网格仿真器。其中组件 GridGenerator 负责模拟网格环境,能随机产生具有不同性能的异构网格节点和节点间链路;而组件 TaskGenerator 能随机产生具有不同大小的任务和相应的最长期限。仿真实验的硬件环境为一台 PⅣ 3.0 GHz的 CPU 和 1 GB 内存,软件环境为 Windows XP Professional 操作系统。

根据多次实验结果,SDHGA 的运行参数设置如下:种群规模大小为 100,交叉概率为 0.8,变异概率为 0.01,算法最大迭代次数为 500。为了使仿真环境和真实的网格环境更接近,必须扩大实验规模,加大仿真环境下任务和机器的数量级,这样才能使仿真实验与真实网络环境更接近。因此,在下面仿真实验中,将网格节点数和任务数分别扩大到 10^2 和 10^3 数量级。

3.2　仿真结果分析

本小节从实时任务调度成功率、系统总安全效益值、平均安全效益值和系统吞吐率等性能指标对本文提出的 SDHGA 算法和其他启发式算法进行综合评价。实时任务调度成功率指在一个调度中满足时间约束关系而被成功调度的任务个数;总安全效益值是从系统的角度考虑,主要指对整个系统中所有成功调度的任务请求所获得最大安全值之和;而平均安全效益值是从用户角度考虑,指所有被成功调度的服务请求所获得的安全效益平均值;吞吐率为单位时间内系统处理的服务请求数,定义为成功调度的服务请求数与最终服务期的值之比。

(1)仿真实验一主要考察在网格环境稳定不变(200 个网格节点),任务数变化(1 000 ～ 5 000 个子任务)情况下,不同性能指标的变化情况,如图 2 所示。

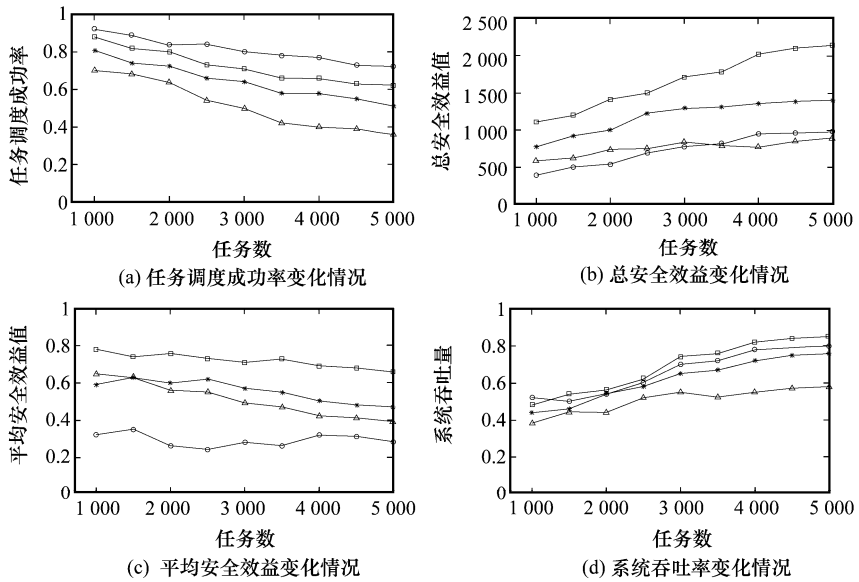

图 2　节点数固定、任务数改变情况下的性能比较

网格环境不变(假定 CCR=1),从图 2(a)可以看出随着任务数的增加,4 种算法任务调度成功率都有所降低。其中,Min-Min 算法由于不考虑安全因素,其成功率最高;SDHGA 的成功率次之,因为该算法是从全局的角度考虑任务分配与调度。从图 2(b)可以看出,随着任务数的增加,4 种算法的总安全效益值都有所增大,其中 SDHGA 总安全效益值最大并且增速较快,说明其可扩展性能最好。从图 2(c)可以看出,随着任务的增多,每种算法的平均安全效益值都有所下降,其中 SDHGA 平均安全效益值最大且下降速度最缓,而不考虑安全的 Min-Min 算法的平均安全效益值最小。从图 2(d)可以看出,随着任务数的增加系统吞吐率都有所提升,其中,Min-Min 算法系统吞吐率最大,SDHGA 和 SD-Min-Min 算法次之,主要是因为 Min-Min 算法不考虑安全因素,其满足时间约束而成功调度的任务个数最多。

在实验一其他条件不变的前提下,对 CCR 分别为 0.1,0.5,1,5 和 10 的应用任务进行调度,得出了新的结果。表 1显示了在 5 种不同 CCR 下,安全驱动 SDHGA 与性能驱动 QoSGA 在任务调度成功率、平均安全效益值方面增加和减少的百分比。

表 1 不同 CCR 下 SDHGA 和 QoSGA 不同性能变化率比较(实验一)

CCR	调度成功率变化率/(%)	安全效益值变化率/(%)
0.1	31.388 5	21.673 3
0.5	29.174 3	21.904 4
1.0	17.543 7	38.245 5
5.0	14.251 3	46.726 1
10.0	13.912 4	68.235 4

(2)仿真实验二主要考察任务数固定(2 000 个任务)、网格环境变化(100 ~ 500 个网格节点)情况下(其中在不同网格环境下假定 CCR=1),不同性能指标的变化情况,如图 3 所示。

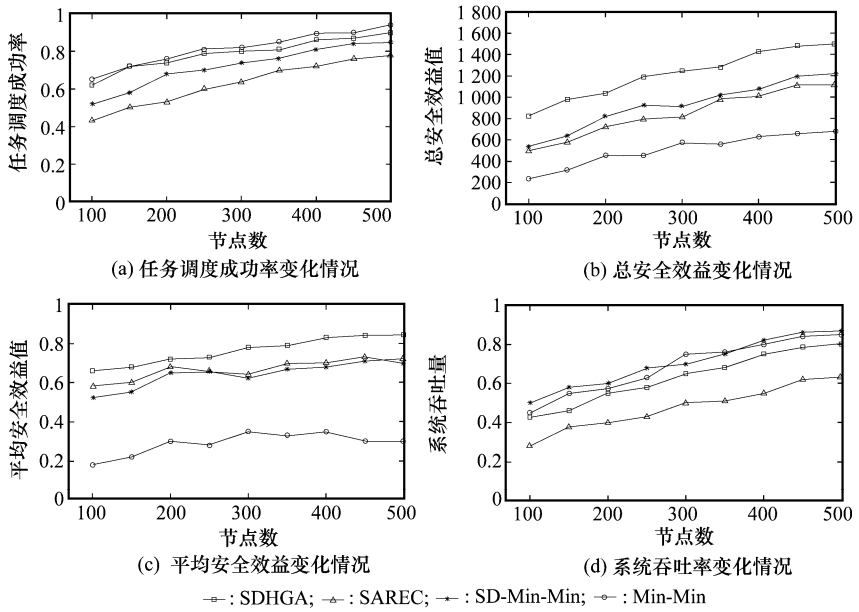

图 3 任务数固定、节点数改变情况下的性能比较

从图 3(a)可以看出随着节点数的扩充,4 种算法的任务调度成功率都有所提高,其中,Min-Min 算法不考虑安全因素,其满足时间约束而成功调度的任务最多;SDHGA 算法从全局的角度考虑任务分配与调度也能获得比较高的任务调度成功率;启发式的 SAREC 最低。从图 3(b)和图 3(c)可以看出,随着节点数的增加,安全驱动的 3 种算法安全效益值都有所增大,其中,不考虑安全因素的 Min-Min 算法最低,并且在同种情况下 SDHGA 安全效益值明显高于其他算法,说明其系统可扩展性较好。从图 3(d)可以看出,随节点数的增加系统的吞吐率都有所提升,其中,不考虑安全因素的 Min-Min 算法成功调度的任务个数最多,系统吞吐率最大,SDHGA 和 SD-Min-Min 算法次之,启发式的 SAREC 最低。

在实验二其他条件不变下,对 CCR 分别为 0.1,0.5,1,5 和 10 的应用任务也进行调度,得出了新的结果。表 2 显示了在 5 种不同 CCR 下,SDHGA 与 QoSGA 在任务调度成功率、平均安全效益值方面增加和减少的百分比。

表 2 不同 CCR 下 SDHGA 和 QoSGA 不同性能变化率比较(实验二)

CCR	调度成功率变化率/(%)	安全效益值变化率/(%)
0.1	42.234 7	27.527 1
0.5	41.975 5	27.148 5
1.0	23.312 9	41.215 4
5.0	17.253 1	57.231 5
10.0	17.018 4	71.402 8

总之,仿真实验证明,本文提出的安全驱动任务调度算法在同等条件下,与经典的性能驱动 Min-Min 算法和考虑安全因素的 SD-Min-Min、SAREC 等其他算法相比,在任务调度成功率、安全效益值和系统吞吐率等方面具有较好的综合性能;特别在 CCR 高的网格环境下,与性能驱动的 QoSGA 相比,本文算法以牺牲较小的任务调度成功率换取了更大的安全性保障。

4　结束语

安全因素在网格平台应用中扮演的角色越来越重要,本文对异构网格环境下的硬实时调度问题,分析了其对保密性、完整性和真实性等安全性能的需求,并设计了相应的安全效益函数,构造了一个安全驱动任务调度新模型。针对当前存在任务调度的启发式算法难以在大规模环境下找到最优解的缺陷,在新模型基础上,通过设计新的选择算子保证了种群的多样性,设计新的杂交算子和调整算子分别增强搜索的广度和精度,以及引入了一个启发式算法,从而提出了一种搜索能力较强的安全驱动任务调度遗传算法。下一步的研究方向是还需要考虑网格节点之间通信链路的可靠性问题,以及不同用户的多种性能需求,开发基于多目标优化的可信动态任务调度算法。

参考文献:

[1] Doulamis N D, Doulamis A D, Varvarigos E A, et al. Fair scheduling algorithms in grids[J]. *IEEE Trans. on Parallel and Distributed Systems*, 2007, 18(11):1630 - 1648.

[2] Xie T, Qin X. Security-aware resource allocation for real-time parallel jobs on homogeneous and heterogeneous clusters[J]. *IEEE Trans. on Parallel and Distributed Systems*, 2008, 19(5): 682 - 697.

[3] Abdelzaher T F, Atkins E M, Shin K G. QoS negotiation in real-time systems and its application to automated flight control [J]. *IEEE Trans. on Computers*, 2000, 49(6): 1170 - 1183.

[4] Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real-time environment [J]. *Journal of the ACM*, 1973, 20(1): 46 - 61.

[5] 罗威,阳富民,庞丽萍,等. 异构分布式系统中实时周期任务的容错调度算法[J]. 计算机学报, 2007,30(10):1740 - 1749.

[6] 张伟哲,刘欣然,胡铭曾,等. 信任驱动的网格作业调度算法[J]. 通信学报, 2006, 27(2):73 - 79.

[7] 袁禄来,曾国荪,姜黎立,等. 网格环境下基于信任模型的动态级调度[J]. 计算机学报, 2006, 29(7): 1217 - 1224.

[8] Xie T, Qin X. Scheduling security-critical real-time applications on clusters[J]. *IEEE Trans. on Computers*, 2006, 55(7): 864 -879.

[9] Xie T, Qin X, Andrew S. SAREC: a security-aware scheduling strategy for real-time applications on clusters[C]// *Proc. for ICPP Conference*, Norway, 2005: 5 - 12.

[10] Song S S, Hwang K, Kwok Y K. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling[J]. *IEEE Trans. on Computers*, 2006, 55(6): 703 - 719.

[11] Song S S, Hwang K. Trusted grid computing with security assurance and resource optimization[C]// *Proc. of the 17th International Conference on Parallel and Distributed Computing Systems*, San Francisco, 2004: 15 - 17.

[12] Tracy D B, Howard J S, Noah B. 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.

[13] Fujimoto N, Hagihara K. A comparison among grid scheduling algorithms for independent coarse-grained task[C]// *Proc. of the Symposium on Applications and the Internet-Workshops*, Washington, 2004: 674 - 680.