

网格计算中基于改进蚂蚁算法的任务调度研究

林晓娴, 王维欢

(西北师范大学 数学与信息科学学院, 甘肃 兰州 730070)

摘要:文中提出了一种以蚂蚁算法为基础的改进算法,用以解决网格环境下的任务调度问题。首先从蚂蚁算法的基本思想出发,结合网格环境下任务调度的特点,逐步改进了资源信息素的初始化处理、局部更新及全局更新方式,并针对网格环境下的任务给出信息素的概念;然后,综合考虑资源信息素和任务信息素两方面的需求,提出了一种新的任务选择资源机制;最终,提出了一种基于蚂蚁算法的、改进的网格任务调度方法。通过仿真实验的结果分析表明:基于蚂蚁算法的、改进的网格任务调度方法实现了网格环境下任务的有效调度问题,并使系统获得较好的负载平衡度。

关键词:改进的蚂蚁算法;任务调度;网格计算

中图分类号:TP393.01

文献标识码:A

文章编号:1673-629X(2011)06-0024-04

Research on Task Scheduling Based on Improved Ant Algorithm in Grid Computing

LIN Xiao-xian, WANG Wei-huan

(College of Mathematics and Information Science, Northwest Normal University, Lanzhou 730070, China)

Abstract: An improved ant algorithm has been presented in order to solve the problem of task scheduling in the grid environment. Firstly, a new style of the initialization processing, the local update and the global update of resource's pheromone has been introduced which based on the ant algorithm and the task scheduling in grid environment. And, the conception of pheromone for the task in grid environment has also been given. Secondly, a new selective mechanism of task resource has been showned by considering the pheromone of resource and task synthetically. Finally, an improved ant algorithm for the task scheduling in the grid environment has been obtained. It indicates from the simulation experiments that the effective scheduling of tasks and the load balancing of system has been implemented by the improved ant algorithm of the task scheduling in grid environment.

Key words: improved ant algorithm; task scheduling; grid computing

0 引言

网格是继因特网、Web 之后的第三次互联网浪潮,被称为第三次因特网的应用。因特网实现了计算机硬件的连通,Web 实现了网页的连通,而网格则可以实现互联网上所有资源的全面连通。网格将整个因特网整合成为一台虚拟的、能力巨大的超级计算机,实现资源共享。网格计算^[1-3]是当前互联网研究中的热点问题,也是分布和并行处理技术的重要发展方向。在网格计算中,任务管理、任务调度和资源管理是网格必须具备的三个基本功能^[4-6]。一般的,用户通过任务管理系统向网格提交任务、为任务指定所需资源、删除任务并监测任务的运行状态;用户提交的任务由任

务调度系统按照任务的类型、所需资源、可用资源等情况安排运行日程和策略;而资源管理系统则确定并监测网格资源状况,收集任务运行时的资源占用数据等。任务调度作为网格计算研究的重点问题,其主要目标就是依据最优跨度、服务质量、负载均衡、经济原则等标准对用户提交的任务实现最优调度,并设法提高网格系统的总体吞吐率。目前已有 Min-Min 算法、Max-Min 算法、Sufferage 算法、遗传算法、蚂蚁算法、模拟退火算法等多种任务调度算法被提出。

文中首先简要地介绍和分析了蚂蚁算法的基本思想,然后结合蚂蚁算法和网格计算中的任务调度问题,提出了一种基于改进蚂蚁算法的网格任务调度方法。这种方法对资源信息素的初始化处理、局部更新及全局更新进行了相应的改进。同时,对于网格环境下的任务也提出了信息素的概念。然后,通过对任务信息素进行 Min-Max 标准化处理,使资源信息素和任务信息素具有可比性,从而为资源信息素的更新提供依据。接下来在任务对资源进行选择时,综合考虑资源信息

收稿日期:2010-10-28;修回日期:2011-02-02

基金项目:甘肃省科学技术攻关计划项目(2GS047-A52-002-04)

作者简介:林晓娴(1983-),女,甘肃临洮人,硕士研究生,研究方向为分布与并行计算;导师:冯百明,教授,博士后,研究方向为计算机系统结构、网格计算。

素和任务信息素两个方面的需求,提出了一种新的任务选择资源的机制。最后仿真实验及结果分析表明,文中提出的基于改进蚂蚁算法的网格任务调度方法实现了网格环境下任务的有效调度和系统的负载平衡。

1 蚂蚁算法

1991年,Marco Dorigo等人提出蚂蚁算法,它是一种模拟蚂蚁群体智能行为的仿生优化算法^[7-9]。根据蚁群觅食行为的特点,蚂蚁能够在没有任何可见提示的情况下,找出从蚁穴到食物源的最短路径,并能随环境的改变搜寻新路径,做出新选择。蚂蚁在其运动过程中分泌一种特殊的化学物质——信息素,从而形成信息素轨迹,蚂蚁利用该轨迹寻找食物源或蚁穴的位置。由于蚂蚁算法具有较强的鲁棒性、优良的分布式计算机制等优点且易于与其他方法相结合,因此,国内外许多学者对其进行了大量研究并将其应用于求解二次分配问题、车辆路径问题、图着色问题以及网络路由等NP类问题及诸多优化领域,如:机器人、图像处理、数据挖掘等,且已取得相当丰富的研究成果。

2 基于改进蚂蚁算法的网格任务调度方法

2.1 资源的信息素及其初始化

通常情况下,资源信息素的初始化处理都是通过资源属性的横向计算得到^[10-15],考虑到资源属性的性质差异及量纲差别,对资源信息素的初始化处理进行一定的改进:即利用资源属性的纵向比较,得到各资源的信息素,以期更真实地反映各资源所固有的计算能力和通信能力。

假设资源集合为 $R = \{R_1, \dots, R_k, \dots, R_m\}$, 其中 m 是资源的个数; $R_k (k = 1, 2, \dots, m)$ 为第 k 个资源。各资源对应的信息素为 $Ph_R = \{Ph_{R_1}, \dots, Ph_{R_k}, \dots, Ph_{R_m}\}$, 其中 $Ph_{R_k} (k = 1, 2, \dots, m)$ 是资源 R_k 的信息素。

所考虑的资源属性主要包括以下三个方面:

(1) 各资源所包含的处理器个数 $N = \{N_{R_1}, \dots, N_{R_k}, \dots, N_{R_m}\}$ (单位:个), 其中 $N_{R_k} (k = 1, 2, \dots, m)$ 为资源 R_k 所包含的处理器个数;

(2) 各资源所包含处理器的处理能力 $Pc = \{Pc_{R_1}, \dots, Pc_{R_k}, \dots, Pc_{R_m}\}$ (单位:MIPS), 其中 $Pc_{R_k} (k = 1, 2, \dots, m)$ 是资源 R_k 所包含各处理器的处理能力;

(3) 各资源所包含处理器的通信能力 $Ps = \{Ps_{R_1}, \dots, Ps_{R_k}, \dots, Ps_{R_m}\}$ (单位:Mb/s), 其中 $Ps_{R_k} (k = 1, 2, \dots, m)$ 为资源 R_k 所包含处理器的通信能力。

若第 k 个资源 R_k 所包含的处理器个数为 N_{R_k} , 处理能力为 Pc_{R_k} , 通信能力为 Ps_{R_k} , 则资源 R_k 的信息素 Ph_{R_k} 的初始化计算为:

$$Ph_{R_k} = \alpha \times \frac{N_{R_k}}{\max \{N_{R_i}\}_{i=1}^{i=m}} + \beta \times \frac{Pc_{R_k}}{\max \{Pc_{R_i}\}_{i=1}^{i=m}} + \gamma \times \frac{Ps_{R_k}}{\max \{Ps_{R_i}\}_{i=1}^{i=m}} \quad (1)$$

其中 α 、 β 和 γ 分别是该资源所包含的处理器个数、处理能力及通信能力在初始化处理时的相对重要程度且 $\alpha + \beta + \gamma = 1$ 。

2.2 任务的信息素及其初始化

假设任务集合 $T = \{T_1, \dots, T_l, \dots, T_n\}$, 其中 n 是任务的个数; $T_l (l = 1, 2, \dots, n)$ 为第 l 个任务。各任务的信息素 $Ph_T = \{Ph_{T_1}, \dots, Ph_{T_l}, \dots, Ph_{T_n}\}$, 其中 $Ph_{T_l} (l = 1, 2, \dots, n)$ 为任务 T_l 的信息素。

所考虑的任务属性主要包括以下两个方面:

(1) 各任务的计算负载 $Lc = \{Lc_{T_1}, \dots, Lc_{T_l}, \dots, Lc_{T_n}\}$ (单位:MIPS), 其中 $Lc_{T_l} (l = 1, 2, \dots, n)$ 是任务 T_l 的计算负载;

(2) 各任务的通信负载 $Ls = \{Ls_{T_1}, \dots, Ls_{T_l}, \dots, Ls_{T_n}\}$ (单位:Mb), 其中 $Ls_{T_l} (l = 1, 2, \dots, n)$ 为任务 T_l 的通信负载。

假设第 l 个任务 T_l 的计算负载为 Lc_{T_l} , 通信负载为 Ls_{T_l} , 则任务 T_l 的信息素 Ph_{T_l} 的初始化计算如下:

$$Ph_{T_l} = \lambda \times \frac{Lc_{T_l}}{\max \{Lc_{T_i}\}_{i=1}^{i=n}} + \mu \times \frac{Ls_{T_l}}{\max \{Ls_{T_i}\}_{i=1}^{i=n}} \quad (2)$$

其中 λ 和 μ 分别表示该任务的计算负载和通信负载在初始化处理时的相对重要程度且 $\lambda + \mu = 1$ 。

2.3 任务对资源的选择

任务选择资源时,一般仅考虑资源的需求而不兼顾任务需求^[10-15]。综合考虑资源信息素和任务信息素两方面的需求,提出一种新的任务选择资源机制。任务 T_l 选择资源 R_k 的概率为:

$$P_{R_k}^{T_l} = \begin{cases} \frac{[Ph_{R_k}]^\zeta \cdot [Ph_{T_l}]^\xi}{\sum_{\substack{i=1,2,\dots,m \\ j=1,2,\dots,n}} [Ph_{R_i}]^\zeta \cdot [Ph_{T_j}]^\xi}, & R_k \in R \text{ and } T_l \in T; \\ 0, & \text{otherwise;} \end{cases} \quad (3)$$

其中 ζ 和 ξ 分别为资源和任务的启发式因子。

2.4 任务信息素的 Min-Max 标准化

由于资源信息素和任务信息素的初始化处理方法相同,因此资源和任务信息素的量纲也相同。另外,由于所考虑的资源属性和任务属性的数目与性质不同,所以资源信息素和任务信息素的值域也不同,即 $Ph_{R_k} \in [\min \{Ph_{R_i}\}_{i=1}^{i=m}, \max \{Ph_{R_i}\}_{i=1}^{i=m}]$, $Ph_{T_l} \in [\min \{Ph_{T_j}\}_{j=1}^{j=n}, \max \{Ph_{T_j}\}_{j=1}^{j=n}]$ 。因此,为了使资源和任务的信息素具有可比性,以便为资源信息素的更新提供依据,采用 Min-Max 标准化方法,以资源信息素的值

域为标准,对任务信息素进行 Min-Max 标准化处理,其计算方法如下:

$$Ph_{T_i}^{new} = \frac{Ph_{T_i}^{old} - \min \{Ph_{T_i}\}_{i=1}^{i=n}}{\max \{Ph_{T_i}\}_{i=1}^{i=n} - \min \{Ph_{T_i}\}_{i=1}^{i=n}}$$

$$(\max \{Ph_{R_j}\}_{j=1}^{j=m} - \min \{Ph_{R_j}\}_{j=1}^{j=m}) + \min \{Ph_{R_j}\}_{j=1}^{j=m} \quad (4)$$

其中 $Ph_{T_i}^{old}$ 和 $Ph_{T_i}^{new}$ 分别为 Min-Max 标准化处理之前、后任务 T_i 的信息素。

2.5 资源信息素的更新

任务被分配给资源、任务的成功返回或失败返回,都会对资源信息素产生影响,这就需要根据具体情况对资源信息素进行更新,从而达到调整任务选择资源的目的。资源信息素的更新主要包括局部更新和全局更新两个方面,它们由局部增量和全局增量在该资源上所执行任务的计算量和通信量来确定^[10-15]。

1) 资源信息素的局部更新。

当任务 J_i 被分配给资源 R_k 时,资源 R_k 的信息素将随之发生变化,即资源 R_k 的信息素进行局部更新。引入新的分配系数 C_a 使局部更新和全局更新保持一致,资源信息素局部更新的计算公式如下:

$$Ph_{R_k}^{new} = \rho \times Ph_{R_k}^{old} - \Delta Ph_{R_k}^{local} \quad (5)$$

其中 $Ph_{R_k}^{old}$ 和 $Ph_{R_k}^{new}$ 分别表示局部更新前、后资源 R_k 的信息素; $\rho \in [0, 1]$ 为资源信息素的持久系数; $\Delta Ph_{R_k}^{local} = C_a \times Ph_{T_i}$ 为资源 R_k 的信息素 Ph_{R_k} 的局部增量,这里分配系数 $C_a \in [0, 1]$ 。

2) 资源信息素的全局更新。

当任务 J_i 从资源 R_k 返回时,资源 R_k 的信息素也会随之发生变化,即对资源 R_k 的信息素进行全局更新,其计算方法如下:

$$Ph_{R_k}^{new} = \begin{cases} \rho \times Ph_{R_k}^{old} + \Delta Ph_{R_k}^{global}, & \text{任务 } T_i \text{ 从资源 } R_k \text{ 成功返回时;} \\ \rho \times Ph_{R_k}^{old} - \Delta Ph_{R_k}^{global}, & \text{任务 } T_i \text{ 从资源 } R_k \text{ 失败返回时;} \end{cases} \quad (6)$$

其中 $Ph_{R_k}^{old}$ 和 $Ph_{R_k}^{new}$ 分别表示全局更新之前、后资源 R_k 的信息素; $\Delta Ph_{R_k}^{global}$ 是资源信息素的全局增量:

$$\Delta Ph_{R_k}^{global} = \begin{cases} C_e \times Ph_{T_i}^{new}, & \text{任务 } T_i \text{ 从资源 } R_k \text{ 成功返回时;} \\ C_p \times Ph_{T_i}^{new}, & \text{任务 } T_i \text{ 从资源 } R_k \text{ 失败返回时;} \end{cases} \quad (7)$$

这里奖励系数 $C_e \in [0, 1]$ 和惩罚系数 $C_p \in [0, 1]$ 分别表示资源执行任务的成功经验与失败经验的增加,且 $C_e - C_p = 2C_a$, $C_p < C_a < C_e$ 。

3 基于改进蚂蚁算法的网格任务调度方法的基本步骤

Step 1 根据公式(1),对资源集合中各资源的信

息素进行初始化处理;

Step 2 根据公式(2),对任务集合中各任务的信息素进行初始化处理;

Step 3 根据公式(3),计算任务集合中的任务选择资源的概率,从中找出概率最大的任务和资源,并将该任务分配到与其相对应的资源上去执行;

Step 4 根据公式(4),当某个任务被分配给某个资源时,对该任务的信息素进行 Min-Max 标准化处理;

Step 5 根据公式(5)、(6),当某个任务被分配给某个资源时,对该资源的信息素进行局部更新;

Step 6 当某个任务被分配给某个资源后,等待该任务的执行结果返回。根据公式(7)、(8),对该资源的信息素进行全局更新,并将该任务从任务集合中删除;

Step 7 重复 Step 3 ~ Step 6,直到任务集合中的所有任务都被成功执行为止。

4 仿真实验及结果分析

使用由 Rajkumar Buyya 教授领导开发的 GridSim 工具包对文中提出的网格任务调度方法进行模拟^[16]。仿真实验中的网格系统由 10 个资源组成^[10],用户分别提交 10、20、40 个任务,各任务的计算负载在 10000 ~ 100000MIPS 之间随机产生,通信负载在 10 ~ 100Mb 之间随机产生。由于资源和任务的信息素在初始化处理时,各资源所包含的处理器个数、处理能力、通信能力以及任务的计算和通信负载均同等重要,故取 $\alpha = 1/3$ 、 $\beta = 1/3$ 、 $\gamma = 1/3$ 、 $\lambda = 1/2$ 、 $\zeta = 1$ 和 $\xi = 1$ ^[7]。

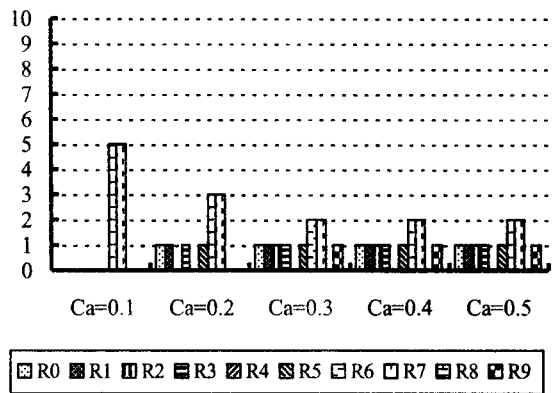


图 1 10 个任务在 10 个资源上的分配情况

从图 1 ~ 图 3 可以看出:当资源总数不变时,在任务数较少时,分配到各资源上的任务数目也较少,即分配系数的取值范围较大;而当任务数较多时,分配到各资源上的任务数目也较多,即分配系数的取值范围较小。图示表明:若分配系数的取值较小,则各任务倾向于被分配到信息素较大的资源上(例如: R_0 和 R_1);而

分配系数的取值较大时,则信息素较小的资源(例如: R_8)也有可能被分配一定的任务。根据资源的计算和通信能力以及任务的要求,可以通过调整分配系数将任务较均匀地分配到资源上去执行。

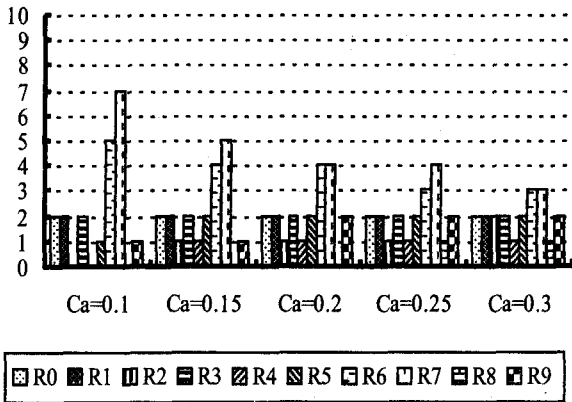


图2 20个任务在10个资源上的分配情况

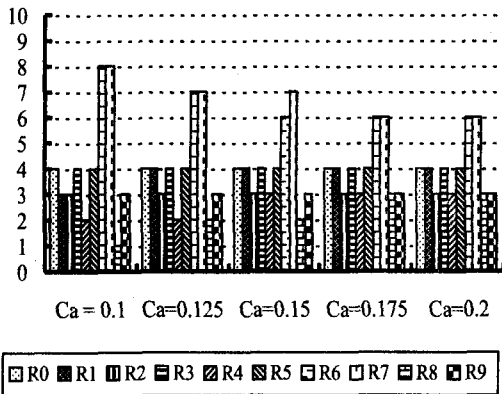


图3 40个任务在10个资源上的分配情况

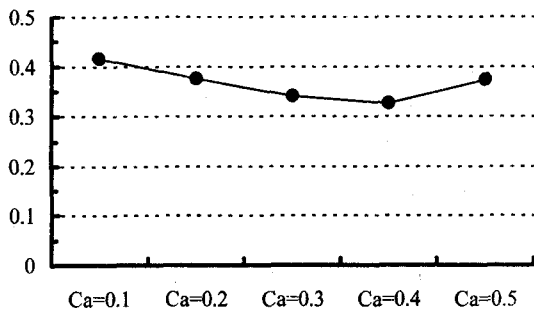


图4 10个任务在10个资源上的负载平衡度

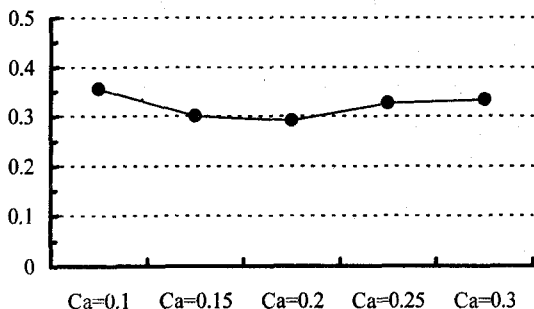


图5 20个任务在10个资源上的负载平衡度

图4~图6说明:不论任务总数如何,当分配系数

的取值较小或较大时,系统的负载平衡度都不是很好;而当分配系数的取值适中时,系统可以达到较好的负载平衡度。由此可见,分配系数的取值应符合资源的实际能力和任务的具体要求。

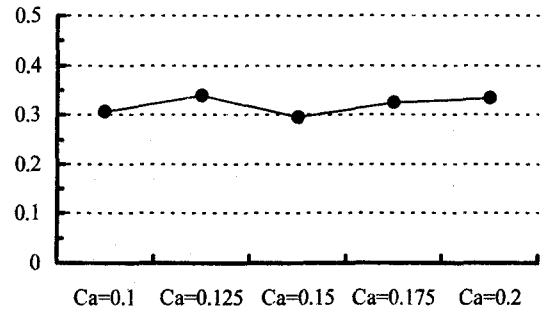


图6 40个任务在10个资源上的负载平衡度

5 结束语

针对网格环境下任务调度的特点,文中首先简要地介绍和分析了蚂蚁算法的基本思想,然后结合蚂蚁算法和网格计算中的任务调度问题,提出了一种基于改进蚂蚁算法的网格任务调度方法。在该方法中,对资源信息素的初始化处理、局部更新及全局更新进行了相应的改进,并对网格环境下的任务也提出了信息素的概念。在任务对资源进行选择时,综合考虑资源信息素和任务信息素两方面的需求,提出了一种新的任务选择资源的机制。仿真实验及结果分析表明,文中提出的基于改进蚂蚁算法的网格任务调度方法实现了网格环境下任务的有效调度和系统的负载平衡。

参考文献:

- [1] 都志辉,陈渝,刘鹏. 网格计算[M]. 北京:清华大学出版社,2002.
- [2] 徐志伟,冯百明,李伟. 网格计算技术[M]. 北京:电子工业出版社,2004.
- [3] 陈宇寒. 网格计算技术研究[J]. 计算机技术与发展, 2008,18(5):82-85.
- [4] 罗红,慕德俊,邓智群,等. 网格计算中任务调度研究综述[J]. 计算机应用研究,2005(5):16-19.
- [5] 郭创,余谅. 网格任务调度算法的研究[J]. 计算机技术与发展,2009,19(6):5-8.
- [6] 徐慧慧,石磊,陈信. 网格资源调度算法研究[J]. 计算机技术与发展,2009,19(9):76-78.
- [7] Dorigo M, Maniezzo V, Colomi A. The ant system: optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics, 1996,26(1):1-13.
- [8] 段海滨. 蚁群算法原理及其应用[M]. 北京:科学出版社,2005.
- [9] 李士勇,陈永强,李研. 蚁群算法及其应用[M]. 哈尔滨:哈尔滨工业大学出版社,2004.

(下转第31页)

越性,测试场景在添加图书信息时使用了外部 CSV 数据文件(数据格式见表1),因此场景会被实例化成3个具体的测试用例进行测试,即同一个脚本根据不同输入数据而被执行了3次,因而达到了脚本复用的目的^[12]。

5.2 测试报告

通过分析测试执行结果,测试工具自动生成基于XML格式的测试报告,报告使用XSLT(扩展样式表转换语言)进行样式格式化,并可由浏览器进行查看。报告以表格的形式分别描述了两个测试场景的执行情况,测试人员可查看预期输出网页特征检查的具体情

况以及追踪测试步路径。实验对上表测试用例“Add-Book”以3组数据进行测试,在测试步“Save1”中均出现了数据库写入错误,而测试用例“ModifyReader”中4个测试步均执行通过。

6 结束语

文中给出了测试执行引擎的工作流程,并描述了其重要组成部分,即测试脚本的设计。此外,还实现了自动化测试工具中脚本生成、解析和执行的方法,最后用一个具体测试场景描述本测试引擎的应用过程。

表1 测试场景测试输入数据

| BARCODE | BOOKNAME | TYPEID | AUTHOR | TRANSLATOR | ISBN | PRICE | PAGE | BOKKCASEID |
|---------|----------|--------|--------|------------|-------|-------|------|------------|
| 13746 | 实用软件测试 | 2 | 飞思科技 | 飞思科技 | 7-115 | 45 | 444 | 52 |
| 24157 | 自动化软件测试 | 2 | 陆璐 | 陆璐 | 7-115 | 40 | 359 | 52 |
| 48791 | NetBeans | 5 | Peter | Peter | 7-111 | 60 | 510 | 47 |

参考文献:

- [1] 许 蕾,徐宝文. 应用测试框架研究[J]. 东南大学学报, 2004, 34(6): 751-755.
- [2] 周晓宇,许 蕾,徐宝文,等. Web 应用的自动测试[J]. 计算机科学, 2005, 32(1): 125-127.
- [3] 朱经纬. XML 技术在软件测试自动化中的应用[J]. 计算机工程, 2005, 31(2): 94-95.
- [4] Andrews A A, Offutt J, Alexander R T. Testing Web applications by modeling with FSMs[J]. Softw Syst Model, 2005, 4: 326-345.
- [5] Yang Ji-tzay, Huang Jiun-long, Wang Feng-jian, et al. Constructing an Object-Oriented Architecture for Web Application Testing[J]. Journal of Information Science and Engineering, 2002, 18: 59-84.
- [6] 朱 菊,王志坚,杨 雪. 基于数据驱动的软件自动化测试框架[J]. 计算机技术与发展, 2006, 16(5): 68-70.
- [7] Jia Xiaoping, Liu Hongming, Qin Lizhang. Formal Structured Specification for Web Applications Testing [C]//2003 Midwest Software Engineering Conference. Chicago, USA: [s. n.], 2003.
- [8] 赵剑英,王 镭. Web 的自动化测试[D]. 天津:天津大学, 2006.
- [9] Liu Chien-Hung, Kung D C, Hsia P, et al. Structural Testing of Web Applications [C]//11th International Symposium on Software Reliability Engineering. [s. l.]:[s. n.], 2000.
- [10] 路晓丽,郝克刚. Web 应用软件的测试技术研究[D]. 西安:西北大学, 2004.
- [11] 许 蕾,徐宝文,陈振强. Web 测试综述[J]. 计算机科学, 2003, 30(3): 100-105.
- [12] 贾晓霞,刘 和,吴 际,等. 一个基于下推自动机的 Web 测试自动执行器[J]. 计算机科学, 2004, 33(4): 269-273.
- [10] 许智宏,孙济洲. 基于蚂蚁算法的网格计算任务调度方法设计[J]. 天津大学学报, 2004, 37(5): 414-418.
- [11] 许智宏,孙济洲. 用蚂蚁算法进行网格任务调度的研究[J]. 计算机应用, 2005, 25(10): 2236-2237.
- [12] 赵晨阳,杨 裔,刘 礼,等. 基于蚂蚁算法的网格作业调度研究[J]. 微电子学与计算机, 2007, 24(9): 78-81.
- [13] 陈 歆,罗四维. 基于蚂蚁算法的网格任务分配算法研究[J]. 计算机技术与发展, 2006, 16(3): 98-100.
- [14] Yan Hui, Shen Xue-qin, Li Xing, et al. An improved ant algorithm for job scheduling in grid computing [C]//Proceedings of 2005 International Conference on Machine Learning and Cybernetics. [s. l.]:[s. n.], 2005: 2957-2961.
- [15] Chang Ruay-Shiung, Chang Jih-Sheng, Lin Po-Sheng. An ant algorithm for balanced job scheduling in grids[J]. Future Generation Computer Systems, 2009, 25(1): 20-27.
- [16] Buyya R, Murshed M. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing [J]. Concurrency and Computation: Practice and Experience, 2002, 14: 1175-1220.

(上接第27页)