改进蚂蚁算法在网格容错调度中的研究

The research of improved ant algorithm for fault-tolerant scheduling in grid environment

梁 鸿 高元涛 张春明
(中国石油大学(华东)计算机与通信工程学院 山东东营 257061)

摘要:本文提出了一种改进的蚂蚁算法,并将其应用到网格容错调度模型中,通过选取合适的参数,并增加负载平衡因子,实现了即使在个别节点出现故障的情况下也能顺利完成任务,并能有效地提高资源的利用率。

关键词:网格计算 调度 蚂蚁算法 最优解 负载平衡

1 引言

随着 Internet 技术的飞速发展和广泛应用,一种新的计算模式——网格计算蓬勃发展起来。计算网格以其强大的计算能力[1]和良好的协同工作能力满足了诸如高能物理、气候模拟等计算密集型任务的需求。在网格这种广域分布、普遍异构的计算环境中进行协同资源共享和问题求解需要解决许多挑战性的问题。其中之一就是在资源故障较频繁的情况下,如何保证网格的可用性和可靠性。这就涉及到了容错计算技术,容错计算一般采用的技术是对失效节点进行任务迁移,以充分利用网格资源的分布性。但是网格资源动态多变,将任务迁移到哪些节点上才能达到系统资源的优化利用呢?

本文提出了一种改进的蚂蚁算法,并将其应用到了网格容错调度模型中,它充分考虑了网格环境中各计算节点的 CPU 的个数及其计算处理能力、网络带宽、磁盘容量、节点信任度等因素,并把这些因素加以综合从而确定各个节点资源的信息素浓度,并且通过增加负载平衡因子,在保证资源利用率的同时,兼顾了网格系统的负载均衡问题。

2 蚂蚁算法介绍

蚂蚁算法[2,3]由意大利学者 M. Dorigo 等人首先提出,它是一种源于大自然生物世界的新的仿生类算法。作为随机型的优化方法,它吸收了蚂蚁的行为特点,通过其内在的搜索机制,在许多困难的优化组合问题的求解中都取得了很好的成效。蚂蚁算法的原理是一种正反馈机制,即利用与环境的动态交互获得反馈信息,调整自我,以期逐步获得最佳解。由于在模拟仿真中使用的是人工蚂蚁概念,因此也被称为蚂蚁系统[4]。

目前,蚂蚁算法的应用已涉及到许多领域,如旅行商问题、二次分配问题[5]、图形着色问题[6]等,并且取得了很好的效果。蚂蚁算法是一种有效的求解 NP 类问题的新型算法,具有较强的鲁棒性和分布并行性,并且非常易于同其它方法相结合。当然,现阶段对蚂蚁算法的研究尚未提出一个完善的理论分析,大部分的工作还处于仿真阶段,但这并不能影响蚂蚁算法的广泛应用。

2.1 蚂蚁算法原理

蚂蚁算法是受真实蚂蚁群体路径寻优方式的启发而提出的。动物学家的观察表明,蚂蚁在从巢穴去食物源沿途中释放一种叫做信息素(pheromone)的化学物质,这些信息素构成信息素的迹(trail)。蚂蚁总是依概率根据沿途信息素迹的大小来选择路径,信息素迹大的路径,被选择的概率也就大。

如图 1 所示,假设蚂蚁要从巢穴 A 去食物源 D,其中 BC 为障碍物,蚂蚁可以选择经由 ABD 或者 ACD 到达食物源。最开始的蚂蚁会随机选择其中一个路径,到达 D,取得食物后返回 A,并在其经过的路径上遗留下信息素。由于蚂蚁走 ACD 路径所用的时间比较短,所以在一定时间内经过 ACD 并遗留下信息素的蚂蚁会相对比较多,并且随着时间的推移,蚂蚁会以越来越

大的概率选择这条相对较短的路径 ACD，而路径 ABD 上的信息素由于挥发作用变得越来越少，导致这条路径最终会被废弃。

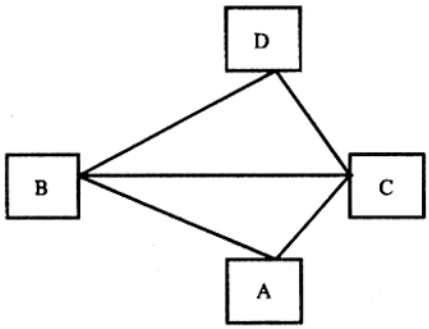

图 1　蚂蚁算法示意图

2.2 蚂蚁算法应用在网格中存在的问题

蚂蚁算法中的人工蚂蚁是一种优化的 agent，它具备真实蚂蚁所不具备的特征：记忆性——能够记忆其过去的行为信息；非盲目性——选择路径时按照一定规律寻找最短路径；蚂蚁算法还具有一个很大的优点——可扩展性。可扩展性就是指在原有的规模为 n 的问题上求出最优解后，再增加 m 个节点，可在原有解的基础上很快找到该问题的 m + n 规模的最优解[7]，因此蚂蚁算法能很好地适应网格环境中动态、不稳定的特点。

但是，我们从蚂蚁算法的图示中可以看出，一般的蚂蚁算法进行的过程中所有的蚂蚁都是通过信息素最多的路径到达目的地，所以在最优路径上出现的蚂蚁数量是巨大的。这种现象在计算网格中是不合理的，因为将失效节点的任务过多地迁移到一个最优资源节点上时，这个资源节点会因为分配的任务过多而由最优资源变为最差资源。这种现象是由于蚂蚁算法路径选择所使用的标准所决定的，蚂蚁算法的目标只有一个——寻找最优解，而没有考虑到竞争现象，最优解收敛到同一个节点上，从而使得这个节点成为整个网格的瓶颈[8]。为了解决该问题，必须对蚂蚁算法进行改进，在给失效任务寻找最优资源的同时尽可能将其分配到多个资源节点上，从而达到整个网格任务负载平衡的目的。

3 改进蚂蚁算法在容错调度模型中的应用

若想将蚂蚁算法应用到网格环境中，需要对蚂蚁算法进行研究，对算法中的参数 α、β 进行适当的改进，并通过增加负载均衡因子，以便能在网格环境下发挥蚂蚁算法的优势。下面将介绍一种改进的蚂蚁算法及其在网格容错调度模型中的具体应用。

3.1 改进蚂蚁算法

在这里，我们引入一个信任度（Tr）的概念：信任度是指网格节点资源所提供的实际服务的成功率。在本文中，信任度是通过计算节点的历史信息来获取的。这些历史信息来自于记录节点活动的日志文件。信任度的定义如下：

$$Tr = Ta/Tf \quad \text{（公式 1）}$$

其中，Ta 表示该节点已经接受的所有任务总数，Tf 表示节点成功完成所接受的任务数。信任度定义的引入为任务调度决策提供了辅助参考，可以有效减少错误率，一个计算节点的 Tr 越高，其错误率越低，通过将任务优先分配给 Tr 高的节点，可以达到提高任务完成质量的目的。

对网格中的每一个资源节点，我们都要求其提供本身的计算能力，节点的计算能力主要由下列几个参数决定：CPU 个数 n 及处理能力 p（MIPS），网络带宽 b（Mb/s）以及磁盘容量 c（M）。根据各个计算节点提供的参数，我们可以初始化该节点的信息素（每个参数对应一种信息素）。资源节点信息素的初始化如下：

$$Ts(0) = a * Tp(0) + b * Tb(0) + c * Tc(0) + d * Tr(0) \quad \text{（公式 2）}$$

其中，a + b + c + d = 1，a，b，c，d 分别表示 CPU 计算能力信息素、网络带宽信息素、磁盘容量信息素以及节点信任度在该资源节点信息素中所占的比重。

每当有新的计算节点加入到网格系统中，某计算节点出现故障，任务被分配到某个计算节点上时，资源的信息素都会随之发生如下改变：

$$Ts^{new} = \rho Ts^{old} + \Delta Ts \quad \text{（公式 3）}$$

其中为 ρ 信息素的持久性，1 − ρ 表示信息素的挥发性[9]。当任务被正常分配时，ΔTs = − K，K 是任务运算量和通信量，表示该节点为了执行任务所需消耗的资源。当任务从资源 s 成功返回时，ΔTs = Ce * K，Ce 是奖励参数，取 0.6，表示增加成功经验；当任务从

资源 s 失败返回时，$\Delta Ts = Cp * K$，Cp 是惩罚因子，取 −1.2。

这样，我们就可以根据各资源节点的信息素得到在该节点上分配任务的概率：

$$P_s^k(t) = \begin{cases} \dfrac{[T_s(t)]^\alpha * [\eta_s]^\beta}{\sum_u \{[T_u(t)]^\alpha * [\eta_u]^\beta\}} & s,u \text{ 为网格中可用的计算节点其他} \\ 0 \end{cases}$$

（公式 4）

其中，$T_s(t)$ 为时间 t 时计算资源 s 的信息素浓度，$\eta_s = T_s(0)$ 表示计算资源 s 的固有属性；α 表示计算资源信息素的重要性，β 表示计算资源固有属性的重要性。当 α 相对较小时，则收敛速度较快，算法容易陷入局部最优，当 β 相对较小时，算法收敛慢，易得到较好的解。

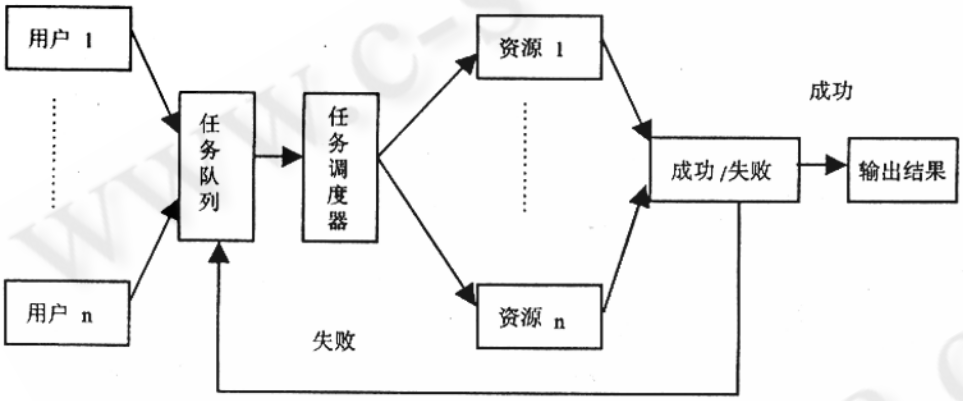

图 2　容错任务调度模型设计图

对于公式 4 中参数的选取，我们引入了动态的 α、β，在算法开始时，取 $\alpha = \alpha_0$，$\beta = \beta_0$，当信息素 Ts 达到一定的阈值后，对 α、β 进行动态调整，$\alpha = \alpha_1$，$\beta = \beta_1$，其中，$\alpha_0 > \alpha_1$，$\beta_0 > \beta_1$。通过动态调整参数 α、β 的值，可以在算法初期增大 β 以加快算法的收敛速度，而在算法后期可以减小来达到加快收敛速度的目的，从而在最短的时间内寻找出全局最优解。

为了取得负载均衡效果，我们根据资源的负载及其任务完成情况，对以后新任务的分配进行调整，同时通过把各资源的负载完成率的差值控制在一个较小的门限之内来保证负载均衡。所谓资源的负载完成率，也就是已经完成的负载和所分配任务的比值。即在每次任务完成时，将任务完成率高的资源信息素增加一些，任务完成率低的信息素降低些。这样就保证负载完成率尽快逼近同一个值，在以后分配任务的过程中，系统的负载均衡能力会得到进一步提高。

3.2　基于改进蚂蚁算法的容错调度流程

网格环境下的容错调度模型如图 2 所示。

用户提交的任务在任务队列中排序，任务调度器按照改进的蚂蚁算法对任务进行调度，在保证资源利用率的同时，提高了任务的成功执行率。一旦任务由于节点故障导致运行失败，可以通过重运行机制将失效节点的任务重新插入到任务队列中去，从而使失败任务同样按照改进蚂蚁算法进行分配，在保证其他节点正常运行的同时，最大程度上提高了整个系统资源利用率和负载均衡。

基于改进蚂蚁算法的思想和容错调度的模型，设计了基于改进蚂蚁算法的容错调度工作流程，该流程共分为：

（1）利用公式 2 初始化网格中各个计算节点的信息素。

（2）用户提交任务到任务队列，任务按优先级排序，任务调度器从任务队列取出任务，具体实施将任务分配到各个计算节点上去，具体做法见 3：

（3）利用公式：

$R(s) = [T_s(t)]^\alpha * [\eta_s]^\beta$

$T = \sum_u \{[T_u(t)]^\alpha * [\eta_u]^\beta\}$

可得概率：$p(s) = R(s)/T$

（4）按照概率 p(s) 将任务分配给相应的节点。

（5）根据公式 3：$Ts^{new} = \rho Ts^{old} + \Delta Ts$，其中，$\Delta Ts = -K$，K 是任务运算量和通信量，对分配任务后的节点的信息素进行更新。

（6）任务分配后，等待任务结果的返回。如果任务成功返回，则将此次执行成功结果记录下来，并利用公式 1

$Tr = Ta/Tf$

对该节点的信任度进行实时更新，同时，按照公式 3

$Ts^{new} = \rho Ts^{old} + \Delta Ts$，$\Delta Ts = Ce * K$

对该节点的信息素进行更新；

如果任务执行失败，则将任务放回任务队列中，等待任务调度器的重新分配，并将此次失败的结果记录下来，利用公式 1

$$Tr = Ta/Tf$$

对该节点的信任度进行实时更新，同时，按照公式 3

$$Ts^{new} = \rho Ts^{old} + \Delta Ts, \Delta Ts = Cp * K$$

对该节点的信息素进行更新。

(7) 重复 3 – 6，直到所有的任务都被成功执行，返回正确的结果为止。

4 实验结果

基于上述网格容错调度流程，我们在实验平台 GridSim 下进行仿真实验，算法编程用 Matlab 语言实现，对于蚂蚁算法中参数的选取如下：取 $\alpha_0 = 1, \beta_0 = 5$，$\alpha_1 = 0.5, \beta_1 = 2$，信息素阈值 Ts 取 1000，模拟产生 8 个计算节点并对各个节点的资源情况进行初始化，各个计算节点的资源情况如表 1 所示：

表 1　节点资源情况列表

节点 ID	处理器个数(N)	处理器计算能力(MIPS)	链路带宽(Mb/s)	磁盘容量(MB)
N1	4	377	20	80000
N2	4	377	30	100000
N3	4	380	20	120000
N4	2	410	40	80000
N5	2	380	10	60000
N6	6	515	20	80000
N7	8	410	20	80000
N8	16	410	40	120000

我们随机产生 800 个任务，每个任务的计算量控制在 3000MI 到 10000MI 之间，数据通信量在 10M 到 100M 之间。

5 结论

针对网格计算环境中资源异构、复杂多变的特点，将改进的蚂蚁算法应用到容错任务调度模型，使得该模型能综合考虑计算网格中各个节点资源的参数，正确评估节点性能，按照各节点性能优劣进行任务分配，即使出现节点失效的情况，利用带负载均衡的蚂蚁算法来对任务进行迁移，也能保证任务的总体执行代价最小并尽量达到系统的负载平衡。下一步研究的重点是将蚂蚁算法与其他启发式算法结合起来，其中一个就是将遗传算法跟蚂蚁算法相结合，利用遗传算法在初期收敛速度快的特点，来弥补蚂蚁算法初期信息素不足的缺陷，从而加快算法的收敛速度，提高蚂蚁算法的运行速度，减少收敛时间。

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 Marco Dorigo, Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem [J], IEEE Transaction on Evolutionary Computation, 1997, 1(4):53 – 56.

3 Marco Dorigo, Maniezzo V, Colorni A. The ant system: optimization by a colony of cooperating agents [J], IEEE Transaction on systems, Man and Cybernetics, 1996, 26(1):29 – 41.

4 赵霞，MAX – MIN 蚂蚁系统算法及其收敛性证明[J]，计算机工程与应用，2006(8):70 – 72.

5 Gambardella L M, Tailard E D, Dorigo M. Ant Colonies for the Quadratic Assignment Problem [J]. Journal of the Operational Research Society, 1999, 50(2):167 – 176.

6 Ibarra O H, Kim C E, Heuristic algotithms for scheduling independent tasks on non – identical processors[J] 1997, 24(2):280 – 289.

7 Costa D, Hertz A. Ants can Color Graphs[J], Journal of the Operational Research Society, 1997, 48: 295 – 305.

8 侯向丹，蚂蚁算法扩展性及应用研究[J]，河北工业大学学报，2002(3):15 – 16.

9 Macro Dorigo, Eric Bonabeau, Guy Theraulaz. Ant algorithms and stigmergy [J] Future Generation Computer Systems, 2000, 16(8):851 – 871.