

改进蚁群算法的云存储任务调度算法研究

袁恩隆, 李 飞, 唐籍涛, 赵伯听

(成都信息工程学院网络工程学院, 成都 610225)

摘 要:由于云存储环境与云计算环境中不同,若直接将云计算环境中的任务调度算法移植到云存储环境中,必然会导致任务调度的效率下降。为解决此问题,提出了一种适用于云存储环境中的改进蚁群算法。改进蚁群算法能使云计算环境中的任务调度算法更符合云存储的环境;同时,对于改进 PSO 算法在引入存在矩阵时,由于数据资源不存在而造成算法前期优化浪费引起效率低下的问题进行了有效解决。分析测试结果表明,提出的改进蚁群算法在云存储环境中的任务调度算法在保障有效解的前提下能够拥有更快的收敛速度。

关键词:云存储;任务调度;蚁群算法

中图分类号:TP393

文献标志码:A

引 言

近年来,随着信息技术的普及和应用,日积月累的数据已经变得十分庞大,用户对海量存储的需求越发明显。云存储^[1]成为了云计算后另一研究重点,云存储的核心是对大量数据的存储和管理。为了将数据安全且有效的存储,相同的数据会存储在许多不同的节点上,即产生许多冗余数据——副本。当用户向服务器提出数据请求任务时,系统会根据相应的策略为用户提供相对最优的数据副本以及路径,这就是任务调度。任务调度的优劣很大程度上决定了系统的运行处理效率。

目前,国内针对在网格计算以及云计算环境中的任务调度算法的研究较多,而单独对云存储系统的任务调度算法的研究比较少见。云存储系统本身是由云计算系统衍生而来,但是由于其主要任务不同,云计算以及网格计算主要是针对计算型任务,任务

可以被委派到任意的云节点中进行计算;而云存储大多是数据传输的任务,因此在云存储中,云中节点是否有用户请求的数据决定了云存储中的任务调度效率。虽然云存储系统与云计算和网格计算大同小异,毕竟云存储也是一个特殊的云系统,因此我们可以在借鉴前人的任务调度算法的同时应该做出适当的改进以适应云存储环境的特点。

在已有的对云存储的任务调度算法的研究中,文献[2]引入了存在矩阵以适应云存储的特点,但该文使用的方法是 PSO 任务调度算法。在 PSO 算法中引入存在矩阵会使其收敛效率低下,这是因为若存在矩阵中的矩阵项为 0 时,粒子前期的飞行计算就会被完全抛弃,浪费了时间与资源。本文研究云存储的任务调度算法,以相对成熟且被广泛应用的蚁群算法为算法基础,沿用文献[2]提出的存在矩阵,根据云存储环境的特点对其进行改进。由于蚁群在更新信息素之前可以通过存在矩阵判断出资源节点中是否有请求资源,因此在效率方面,

收稿日期:2013-09-05

基金项目:四川省科技支撑项目(2011GZ0195)

作者简介:袁恩隆(1988-),男,四川成都人,硕士生,主要从事基于云存储的计算机应用研究方面的研究,(E-mail)yuanenlong@hotmail.com

蚁群算法^[3-5]更加符合云存储环境。实验证明,改进之后的算法不仅能提高原始算法在云存储环境中的有效解,节省大量时间;同时,改进蚁群算法在效率上也比改进 PSO 算法的效率要高。

1 云存储任务调度的抽象

目前国内外所研究的任务调度^[6-7]分在线模式与批处理模式。在线模式是在任务到来的第一时间就产生映射进行调度;而批处理模式是将任务累计到一定数量,等映射事件发生后再开始映射搜集的任务。本文的研究是对于“批处理”模式进行的,即在一个单位时间内有 n 个新任务产生并等待调度。本文使用以下定义:

定义 1 $T = \{t_1, t_2, \dots, t_n\}$ 代表单位时间内等待调度的任务集。 n 是任务数。

定义 2 $N = \{n_1, n_2, \dots, n_m\}$ 代表云存储环境中的资源节点集合。 m 是节点数。对云存储环境来说, n_i 代表 n_i 上的数据。

定义 3 B 代表云存储系统的带宽矩阵。根据定义 2,云存储环境中的资源节点数为 m ,则 B 是一个 $m \times m$ 的矩阵,矩阵项 b_{ij} 为节点 i 与节点 j 之间的带宽。由于任意两个节点之间的通路可能不止一条,则带宽一般可以有多个值,这里我们取最小值即可。

定义 4 用 $V = [v_1, v_2, \dots, v_n]$ 表示任务调度向量,即一个调度方案。对云存储系统来说, v_i 代表第 i 个任务的数据由 v_i 的值代表的资源节点提供,该向量的长度即单位时间内需要调度任务的总量。用 F 表示任务调度向量的产生函数。调度向量的产生函数是一个任务调度的灵魂,该函数的不同直接导致了任务调度的效率。本文取 F 为蚁群算法。

定义 5 在云计算环境中,一般的适应度函数均为最短完成时间(makespan),本文沿用此定义。云中的适应度函数为:

$$D = \min \left(\sum_i \frac{t_i}{b_{ij}} \right) \quad (1)$$

2 蚁群算法在云存储系统中的应用

2.1 标准蚁群算法

蚁群算法(ACO)具有正反馈以及较强的鲁棒性等

特点,主要用于解决不同的组合优化的问题。通过信息素的累积和更新收敛于最优路径,最终得到全局最优解。

蚂蚁在觅食的过程中会留下一种信息素,蚂蚁利用信息素与其他蚂蚁交流,找到较优路径。假如路径 (i, j) 在 t 时刻信息素强度为 $\tau_{ij}(t)$,节点 i 与节点 j 的距离为 L_k ,蚂蚁 k 在路径 (i, j) 上释放的信息素强度为 $\Delta \tau_{ij}^k$,信息素的挥发系数为 ρ , Q 为常数。则该路径上的信息素强度按下式更新:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta \tau_{ij}^k(t) \quad (2)$$

$$\Delta \tau_{ij}^k(t) = \frac{Q}{L_k} \quad (3)$$

蚂蚁 k 在 t 时刻从节点 i 到节点 j 的概率为:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in allowed_k} [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta} \quad (4)$$

当且仅当 $j \in allowed_k$, 否则 $p_{ij}^k(t) = 0$ 。其中 $allowed_k$ 为蚂蚁 k 能够到达的节点的集合; η_{ij} 为节点 i 转移到节点 j 的期望值,即启发式因子; α 和 β 分别为信息素轻度与启发式因子的重要性。

2.2 标准蚁群算法在云计算中的应用

传统的基于蚁群算法的云计算任务调度按照以上算法,跟据公式(2)和(4)初始化出任务调度解,判断结束条件,若没有结束则循环产生新的解,直至满足结束条件。

若直接把此算法直接移植到云存储环境中,不满足云存储的特殊性,即并不是每个节点都有任务所需的数据,会产生许多无效解,导致该算法在云存储环境中的效率低下。

2.3 改进蚁群算法在云存储中的应用

针对云存储环境的特殊性,由于存在有些节点不能为任务提供数据资源,本文借鉴文献[2]提出的存在矩阵(exist matrix, EM)来反映任务与资源节点之间的对应关系。矩阵中的项代表任务 i 的数据在资源节点 j 中是否存在,若存在则为 1,否则为 0。该存在矩阵可以从云存储的资源列表中生成。为了有效地解决 PSO 算法在重新产生解时浪费该算法前期的优化工作的问题,改进蚁群算法结合其自身的特点,对资源节点内资源是否存在的判断放在蚂蚁对下个节点的选择之前,即首先判

断是否有资源,若资源不存在,则直接返回上一步,只有在资源存在的情况下才会再继续执行算法的跳转部分。

在改进蚁群算法中,将矩阵项 e_{ij} 的值赋给启发式因子 η 。当 $e_{ij} = 0$, 即任务 i 的数据在节点 j 中不存在,此时 $\eta = 0$, 跳转至 j 节点的概率为:

$$P_{ij}^k(t) = 0 \quad (5)$$

当 $e_{ij} = 1$, 即任务 i 的数据存在于节点中, $\eta = 1$, 则跳转至 j 节点的概率为:

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha}{\sum_{j \in allowed_i} [\tau_{ij}(t)]^\alpha} \quad (6)$$

云存储中基于蚁群算法的任务调度算法过程如下:

(1) 初始化信息素的值、初始化 EM 矩阵、初始化带宽 b , 迭代次数 $N_c = 0$, 设置最大迭代次数 N_{max} ;

(2) 将 n 个任务任意放在 m 个节点上;

(3) 从带宽矩阵得到 b_{ij} , 计算并保存本文的适应

$$度: d = \sum_i \frac{t_i}{b_{ij}};$$

(4) 通过 EM 矩阵判断资源节点是否有请求数据, 若不存在, 返回步骤(3), 若存在, 则通过公式(6)选择节点移动, 通过公式(3)对路径上的信息素更新;

(5) $N_c = N_c + 1$;

(6) 若 $N_c < N_{max}$, 则跳至步骤(3), 若 $N_c < N_{max}$, 则继续;

(7) 调度完成, 输出一个 d 最小的任务调度向量 V_i 。

3 实验仿真及结果分析

为了验证算法的有效性, 使用云仿真工具 CloudSim 进行试验。在实验中, 带宽矩阵以及存在矩阵均由计算机随机产生, 并设置最大迭代次数 N_{max} 为 100, 资源节点数量为 10, 初始信息素的值为 5, 信息素增值为 1。仿真为任务数按照 20、40、60、80、100 递增的情况下, 依次实验。为避免多因素干扰, 将每个实验执行 20 次并取其均值, 分析算法的性能(图 1)。

图 1 将本文提出的改进蚁群算法与文献[2]提出的改进 PSO 算法以及原始蚁群算法在云存储中的适应度的比较。仿真实验所得到的最小适应度值即为最小执行时间(makespan)。实验表明原始蚁群算法适应度最大, 这是因为原始蚁群算法在求解中会有许多的无效解。而改进蚁群算法在云存储中的适应度更小, 即任务

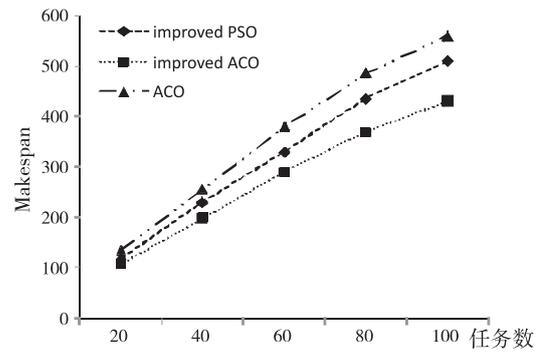


图 1 makespan 对比

总执行时间更小, 与预期结果相符。

图 2 的对比中, 改进蚁群算法在云存储中求到最优解的迭代次数明显比传统的蚁群算法的迭代次数少很多, 并随任务数的增加越发明显。改进蚁群算法的收敛速度明显优于传统蚁群算法。

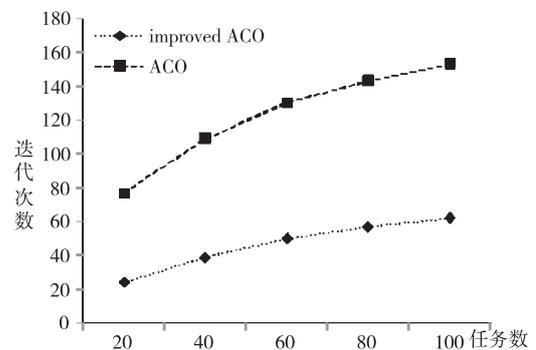


图 2 迭代次数对比

4 结束语

本文研究了云存储中的任务调度算法, 发现云存储中的改进 PSO 任务调度算法在求解任务调度解时效率不高, 因此提出了基于改进蚁群算法的云存储任务调度算法。该算法在适应云存储环境的前提下, 具有收敛更快的优势, 并通过仿真验证了其有效性。下一步的研究是在该算法的基础上, 在蚂蚁信息素更新时引入用户的 QoS 需求, 使调度任务能在有效完成的基础上最大化用户的 QoS 偏好。

参考文献:

- [1] Hayes B. Cloud computing[J]. Communications of the ACM, 2008, 51(7):9-11.
- [2] 王娟, 李飞, 张路桥. 限制解空间的 PSO 云存储任务

- 调度算法[J].计算机应用研究,2013,30(1):127-130.
- [3] 张军,胡晓明.蚁群优化[M].北京:清华大学出版社,2007.
- [4] Dorigo M,Caro G D.The ant colony optimization meta-heuristic[C]//Come D,Dorigo M,Glover F.New Ideas in Optimization.London:McGraw Hill,1999:11-32.
- [5] 李宗勇,彭霞,王智学,等.基于蚁群算法的参数相关网络任务调度算法研究[J].系统仿真学报,2007,19(14):3196-3199.
- [6] 李坤.云环境下的任务调度算法研究与实现[D].吉林:吉林大学,2012.
- [7] Martino V D,Mililotti M.Scheduling in A Grid Computing Environment Using Genetic Algorithms [C]// Proceedings of International Parallel and Distributed Processing Symposium,Florida, April 15-19,2002:235-239.
- [8] 李建锋,彭舰.云计算环境下基于改进遗传算法的任务调度算法[J].计算机应用,2011,31(1):184-186.
- [9] 王永贵,韩瑞莲.基于改进蚁群算法的云环境任务调度研究[J].计算机测量与控制,2011,19(5):1203-1211.
- [10] 魏东,吴良杰,佐丹,等.基于混合蚁群算法的网络任务调度[J].计算机工程,2010,36(3):215-217.

Research on Task Schedule Algorithm of Cloud Storage Based on Improved Ant Colony Algorithm

YUAN Enlong, LI Fei, TANG Jitao, ZHAO Boting

(College of Network Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: Due to the different between the cloud storage environment and the cloud computing environment, directly transplanting task scheduling which used in cloud computing to the cloud storage environment will inevitably lead to a decline in the efficiency of task scheduling. To solve this problem, an improved ant colony algorithm which is applicable to cloud storage environment is proposed. This improved ant colony algorithm is more suitable for cloud storage environment. At the same time, there is no data resources when the improved PSO algorithm is introduced in matrix, so that a waste of early optimizing of the algorithm is produced, which causes a problem that the efficiency is very low, the problem is solved effectively. Analysis of test results shows that the improved ant colony algorithm proposed in the cloud storage environment task scheduling algorithm has faster convergence rate under the premise to guarantee efficient solutions.

Key words: cloud storage; tasks scheduling; ant colony algorithm