基于 Map-Reduce 模型的云资源调度方法研究

张恒巍 韩继红 卫 波 王晋东
(解放军信息工程大学三院 郑州 450001)

摘 要 为提高 Map-Reduce 模型资源调度问题的求解效能,分别考虑 Map 和 Reduce 阶段的调度过程,建立带服务质量(QoS)约束的多目标资源调度模型,并提出用于模型求解的混沌多目标粒子群算法。算法采用信息熵理论来维护非支配解集,以保持解的多样性和分布均匀性;在利用 Sigma 方法实现快速收敛的基础上,引入混沌扰动机制,以提高种群多样性和算法全局寻优能力,避免算法陷入局部最优。实验表明,算法求解所需的迭代次数少,得到的非支配解分布均匀。Map-Reduce 资源调度问题的求解过程中,在收敛性和解集的多样性方面,所提算法均明显优于传统多目标粒子群算法。

关键词 云计算,Map-Reduce,资源调度,粒子群算法,信息熵,混沌扰动
中图法分类号 TP393.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.025

Research on Cloud Resource Scheduling Method Based on Map-Reduce

ZHANG Heng-wei HAN Ji-hong WEI Bo WANG Jin-dong
(Third Institute, PLA Information Engineering University, Zhengzhou 450001, China)

Abstract To improve the computing efficiency of Map-Reduce resource scheduling, a multi-objective resource scheduling model with QoS restriction was built. The model considers the scheduling problem of both Map and Reduce phase. A chaotic multi-objective particle swarm algorithm was proposed to solve the model. The algorithm uses the information entropy theory to maintain non-dominated solution set so as to retain the diversity of solution and the uniformity of distribution. On the basis of using Sigma methods to achieve fast convergence, chaotic disturbance mechanism was introduced to improve the diversity of population and the ability of algorithm global optimization, which can avoid the algorithm to fall into local extremism. The experiments show that the number of iteration in the algorithm obtaining solutions is little and non-dominated solutions distribute equably. It indicates that the astringency and the diversity of solution set of this algorithm are better than the traditional multi-objective particle swarm algorithm in solving Map-Reduce resource scheduling problems.

Keywords Cloud computing, Map-Reduce, Resource scheduling, Particle swarm algorithm, Information entropy, Chaotic disturbance

1 引言

云计算是在网格计算、并行计算和 P2P 等技术基础上发展起来的一种新兴计算模型,能将大量计算资源、存储资源与软件资源链接在一起,形成大规模的共享虚拟资源池,为用户和应用系统提供"召之即来,挥之即去"且似乎"能力无限"的云服务[1-3]。在云计算模式下,用户可以通过网络在任意时间和地点访问所需要的海量服务,因此合理、高效的资源调度成为影响云服务质量的关键,是云计算领域的研究重点和难点。Map-Reduce 是 Google 提出的一种云计算模型[4,5],虽然截至目前 Google 并未公开模型的实现细节,但是该模型由于具有实现简单、能高效地利用云资源等优点,受到了学术界的广泛关注和探讨。

Map-Reduce 资源调度是指:在 Map 和 Reduce 两个阶段,如何把大量任务有效地分配到相应的虚拟计算资源上,最大限度地满足任务完成时间、完成成本等要求。一方面,Map-Reduce 中的资源调度问题已被证明是一个 NP-Hard 问题;另一方面,由于云计算降低了资源准入门槛,融合了海量的网络资源形成规模巨大的候选资源池,极大地增加了问题的求解空间规模,同时云计算环境中资源按需索取的模式又对资源调度的效率提出了严峻的挑战,因此,FIFO 调度算法、公平调度算法、计算能力调度算法等传统算法因具有较高的时间复杂度而不适用于 Map-Reduce 资源调度。近年来,越来越多的智能算法被应用于云资源调度问题,但是在 Map-Reduce 资源调度方面研究成果较少。文献[6]用蚁群算法设计了 Map-Reduce 调度方案,其有效减少了任务响应时间;文

到稿日期:2014-08-28 返修日期:2015-01-11 本文受国家自然科学基金项目(61303074,61309013),国家重点基础研究发展计划("973"计划)基金项目(2012CB315900)资助。
张恒巍(1978-),男,博士生,主要研究方向为云资源管理、需求工程,E-mail:zhw11qd@126.com;韩继红(1966-),女,博士,教授,博士生导师,主要研究方向为网络资源管理、系统工程;卫 波(1990-),男,硕士生,主要研究方向为云服务组合与选择;王晋东(1966-),男,教授,主要研究方向为资源动态管理、云计算体系结构。

献[7]采用遗传算法减少了作业总完成时间和平均完成时间。但上述文献均未分别考虑 Map 阶段和 Reduce 阶段的调度过程，实用性存在局限。文献[8]分别考虑了 Map-Reduce 的两阶段调度过程，但是将多个优化目标线性加权转化为一个单目标函数，难以协调各目标函数值，无法产生多个可行解，用户没有选择余地，且未考虑 QoS 约束，存在使用局限。文献[9]将资源调度问题转化为多目标优化问题，并基于自适应领域的方法改进多目标遗传算法，用于求解网络资源调度问题，但其采用了较为复杂的遗传算法，收敛速度较慢，优化解的质量还有待提高。

为此，本文基于 Map-Reduce 模型，提出一种基于混沌多目标粒子群算法(Chaotic Multi-Objective Particle Swarm Optimization，CMOPSO)的资源调度方法。将资源调度问题转化为带 QoS 约束的多目标寻优问题，通过同时优化多个目标参数，最终产生一组满足约束条件的非支配解。该算法相比 MOPSO 算法能有效避免陷入局部最优，具有更强的全局寻优能力；且收敛性、解集的多样性均得到改善，能够更好地解决云计算环境下的资源调度问题。

2 基于 Map-Reduce 的资源调度模型

Map-Reduce 模型的结构框架如图 1 所示，模型中有 3 种角色[10]，即用户、Master、资源节点。Master 是系统的主控程序，负责元数据组织、任务调度、负载均衡、容错处理等；资源节点从 Master 接收任务，进行数据处理和计算；用户需要实现 Map 和 Reduce 函数，控制计算。Map-Reduce 运行过程如下：将用户提交的一个任务划分为 M 个 Map 子任务，首先分配到 Map 阶段的资源节点上执行，产生 R 个结果并缓存到本地磁盘；其次将 Map 阶段的结果作为 R 个 Reduce 子任务，提交给 Reduce 阶段的资源节点执行；最后把得到的最终结果返回给用户。其中主控程序 Master 实现这 M 个 Map 子任务和 R 个Reduce 子任务在空闲资源节点上的调度。

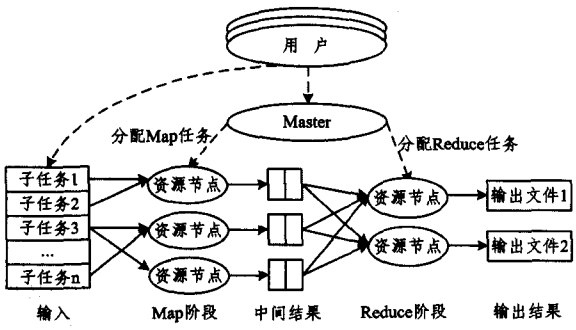

图 1 Map-Reduce 模型结构

本文参考文献[11]，将 Map-Reduce 的资源调度表示为以下的四元组：

$$M=(U,S,F,\theta)$$

其中，U 表示由用户提交的 n 个任务组成的任务集合；S 表示由 m 个资源节点组成的资源集合；F 表示云资源调度模型的目标函数；θ 则表示解决该调度模型采用的智能寻优算法。调度模型的具体特征可以描述如下：

1)任务集合 $U=(t_1,t_2,\cdots,t_n)$，由用户提交的 n 个任务组成，其中第 t_i 个任务可以拆分为 $M(i)$ 个 Map 子任务和 $R(i)$ 个 Reduce 子任务，则 U 中合计有 sn 个子任务，$sn=\sum_{i=1}^{n}(M(i)+R(i))$。

2)资源集合 $S=(s_1,s_2,\cdots,s_m)$，由 m 个资源组成，每个资源负责处理分配到该资源上的子任务。

3)用已知预测执行时间(ETC)矩阵来记录任务执行时间，$ETC(i,j)$表示第 i 个子任务在第 j 个资源节点上的执行时间。

4)用 RCT 数组表示各个资源的单位时间成本，$ETC(s)$表示第 s 个资源的单位时间成本。

可以得到任务 t_i 的执行时间如下式：

$$Time(t_i)=\max_{k=1}^{M(i)}\left(\sum_{u=1}^{Nom}WM(k,u)\right)+\max_{g=1}^{R(i)}\left(\sum_{v=1}^{Nor}WR(g,v)\right) \quad (1)$$

其中，Nom 代表第 k 个 Map 子任务被安排在资源 S 的计算队列中的位置，在它前面资源 S 的计算队列中还有($Nom-1$)个子任务；$WM(k,u)$代表资源 S 依据计算队列的顺序完成第 u 个子任务的时间，可从 ETC 矩阵获取；当 $u=Nom$时，$WM(k,u)=WM(k,k)$代表资源 S 完成第 k 个子任务的时间；$\sum_{k=1}^{Nom}WM(k,u)$表示资源 s 依据计算队列的顺序执行完第 k 个 Map 子任务所用的全部时间。同理，Nor 代表第 g 个 Reduce 子任务被安排在某个资源的计算队列中的位置，$\sum_{g=1}^{Nor}WR(g,v)$表示该资源依据计算队列的顺序执行完第 g 个 Reduce 子任务所用的全部时间。

基于上述分析可知，当任务集合 U 中用时最长的任务 t' 执行完毕时，整个任务集合最终执行完成。因此，任务集内全部任务完成总时间 T_1 等于任务 t'的执行时间，即：

$$T_1=\max_{i=1}^{n}\left[\max_{k=1}^{M(i)}\left(\sum_{u=1}^{Nom}WM(k,u)\right)+\max_{g=1}^{R(i)}\left(\sum_{v=1}^{Nor}WR(g,v)\right)\right] \quad (2)$$

任务的平均完成时间 T_2 为：

$$T_2=\frac{1}{n}T_1=\frac{1}{n}\left(\max_{i=1}^{n}Time(t_i)\right) \quad (3)$$

可以看出，当资源充分时，随着任务集合 U 的增大，任务完成总时间 T_1 并不增加，此时平均完成时间 T_2 反而逐渐减小，甚至小于用时最短的任务的执行时间，体现了云计算的优势。

第 s 个资源完成分配到该资源上的所有子任务所需成本为：

$$cost(s)=\sum_{i=1}^{Num(s)}ECT(i,s)\times RCT(s) \quad (4)$$

其中，$Mum(s)$表示资源 s 上分配的子任务的个数。

所有任务完成的总成本为：

$$SC=\sum_{s=1}^{m}cost(s) \quad (5)$$

云资源调度的目标是对任务队列和资源节点进行最优匹配，要求在满足特定 QoS 约束的条件下，使资源调度方案的多个目标达到最优，属于带约束的多目标寻优问题。本文将全部任务完成总时间 T_1、任务的平均完成时间 T_2、完成全部任务的总成本 SC 作为 3 个目标，希望得到最小的 T_1、T_2、SC。将资源可靠性 Dep、资源信誉度 Rep 作为两个约束条件，表明调度资源所要求的最低可靠性和信誉度。一个带约束的多目标资源调度模型可描述为：

$$\begin{cases} \text{Min}T_1, \text{Min}T_2, \text{Min}SC \\ \text{s. t. } Dep(s) \geqslant Dep_0 \\ \quad Rep(s) \geqslant Rep_0 \end{cases} \tag{6}$$

一方面，由于实际应用中寻优目标、约束条件的种类很多，可以将它们看成多目标寻优问题的目标函数或约束条件，因此本模型可以对目标函数和约束条件进行推广。

另一方面，不同的目标之间具有不可公度性和矛盾性等特点，可能造成目标函数之间相互冲突，无法同时满足多个目标的最优化条件。因此，多目标寻优问题的结果一般不是单一解，而是受制于约束条件的一组非支配解[12]。

基于上述分析可知，Map-Reduce 调度问题是一个两阶段、带约束条件的多目标寻优问题，因此求解算法不但要分别考虑两阶段调度过程，还要能够在 QoS 约束的前提下进行多目标同时寻优。为此，基于多目标粒子群算法（MOPSO），针对 Map-Reduce 调度问题，提出混沌多目标粒子群算法（CMOPSO）对问题进行求解。

3 混沌多目标粒子群算法

粒子群优化算法[12]最早是由 James Kenndy 和 Russell Eberhart 在深入研究鸟类群体觅食的基础上提出的。该算法由于具有并行计算、群体寻优、实现方便、搜索速度快等优点，得到国内外学者的广泛关注，已经被广泛应用于多个领域。MOPSO 算法是 PSO 算法应用于多目标优化问题时提出的，算法设定粒子具有位置、速度和适应值 3 个基本属性，其中每个粒子的位置代表问题的一个可行解，粒子的速度决定其位置的变化方向和变化率，适应值由目标函数决定，根据适应值的大小比较粒子的优劣，再根据支配关系从初始粒子群中筛选出非支配解集。算法基本流程如图 2 所示。

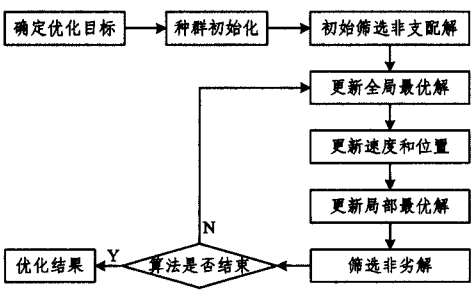

图 2 基本多目标粒子群算法流程

MOPSO 算法在应用中的主要缺点是：非支配解在解空间中的分布均匀性较差，解的多样性不足；全局寻优能力较差。针对上述问题，研究人员从不同角度对 MOPSO 进行了改进，张伟星[12]基于动态多种群优化思想对 MOPSO 进行改进，改进后的算法能对变换的帕雷托前沿进行快速跟踪，提高了最优解集的分布性。裴胜玉、周永权[13]提出利用混沌变异方法跳出局部最优解，增强 MOPSO 算法的全局寻优能力，但是没有给出种群粒子适应度的计算方法，并且采用轮盘赌方法选择种群最优个体，其准确性和效率有待提高，同时未给出混沌扰动的使用时机和具体实现。以上文献是本文提出 CMOPSO 算法的研究基础。

CMOPSO 算法的贡献在于：(1)提出利用信息熵计算粒子适应度的方法；(2)采用 Sigma 方法确定迭代中粒子的全局最优位置，提高了收敛速度；(3)设计了实施混沌扰动的判定条件和具体形式，帮助种群摆脱局部最优的束缚。

3.1 初始化粒子

粒子的位置变量代表资源调度的解（即资源调度方案）。在 Map-Reduce 资源调度模型中，设有 n 个任务，m 个资源，其中任务 t_i 被分为 $M(i)$ 个 Map 子任务和 $R(i)$ 个 Reduce 子任务。则 Map 子任务的某个调度方案 x_i 可表示为 $(x_{i1}, \cdots, x_{im}, \cdots, x_{iM})$，其中 x_{im} $(1 \leqslant m \leqslant M, M = \sum_{i=1}^{n} M(i))$ 表示执行第 t_i 个任务的第 j 个 Map 子任务的资源编号。对所有 Map 子任务采取顺序编码方式，编码的每一位可表示为：

$$m[i,j] = j + \sum_{k=1}^{i-1} M(k) \tag{7}$$

式中，$m[i,j]$ 代表第 t_i 个任务的第 j 个 Map 子任务的序号。例如，有 5 个资源，3 个任务，第 1 个任务划分为 3 个 Map 子任务，后两个任务划分为 2 个 Map 子任务，子任务的数量为 7，则 (2,3,5,1,4,3,1) 是调度模型的一个解，表示第 1 个任务的第 1 个 Map 子任务在 2 号资源上执行，第 2 个任务的第 1 个 Map 子任务的序号为 4，其在 1 号资源上执行，其余依此类推，如表 1 所列。

表 1 编码示例

任务序号	子任务序号	执行子任务的资源编号
1	1	2
	2	3
	3	5
2	4	1
	5	4
3	6	3
	7	1

同样的方法对 Reduce 子任务进行编码，则 Reduce 子任务的调度方案可表示为 $(x_{i1}, \cdots, x_{ir}, \cdots, x_{iR})$，其中 x_{ir} $(1 \leqslant r \leqslant R, R = \sum_{i=1}^{n} R(i))$ 表示执行第 t_i 个任务的第 j 个 Reduce 子任务的资源编号。

依据上述方法，每个粒子的位置用一个 n 维向量来描述，令 $n = sn = \sum_{i=1}^{n} (M(i) + R(i))$，代表有 sn 个子任务。每一维变量的取值范围为 [1, m] 上的整数，代表有 m 个资源，用区间 [1, m] 内的整数为每个资源编号。假设向量 $x_i = (x_{i1}, \cdots, x_{ij}, \cdots, x_{in})$ 为第 i 个粒子的位置，则其代表第 i 种资源调度方案，其中 x_{ij} 代表在 sn 个子任务中第 j 个子任务选用编号为 x_{ij} 的资源。

粒子群初始化质量的好坏直接影响算法的求解效果，对算法而言，初始化之后粒子在解空间里的分布越均匀效果越好。混沌序列具有典型的非线性特征以及随机性特点，在优化问题中有着广泛的应用[14]。本文借鉴混沌序列思想，利用一介 Logistic 函数构建混沌变量，对粒子群进行初始化。

$$\beta_{(k+1)j} = \mu \beta_{kj} (1 - \beta_{kj}), k = 1, 2, \cdots; \beta_j \in (0,1) \tag{8}$$

式中，k 代表函数迭代的次数；β_{ij} 是混沌变量，代表粒子在第 j 维的位置属性值。当 $\mu \geqslant 4$ 时，Logistic 方程构建的序列为混沌序列。初始化 [0,1] 区间内的 m 维随机数 $\beta_{ij} = Rand()$ $(j = 1, 2, \cdots, m)$。对于规模为 N 的粒子群，基于式 (8) 完成 $(N-1)$ 次迭代，即 $\beta_{(i+1)j} = \mu \beta_{ij} (1 - \beta_{ij})$，$i \in [1, N-1]$，$j \in [1, m]$，之后采用式 (9) 将 β_{ij} 映射为粒子的位置，并向上取整得到位置变量的整数值。

$$x_{ij}=\lceil 1+(n_j-1)\beta_{ij}\rceil \tag{9}$$

3.2 更新粒子速度和位置

在算法的迭代求解过程中，第 t 次迭代时粒子的速度更新公式如式(10)所示，位置更新公式如式(11) 所示。

$$v_{ij}^{t+1}=wv_{ij}^{t}+c_1r_1(P_{ij}^{t}-x_{ij}^{t})+c_2r_2(P_{gj}^{t}-x_{ij}^{t}) \tag{10}$$

$$x_{ij}^{t+1}=\lceil|(x_{ij}^{t}+v_{ij}^{t+1})\%n_j|\rceil \tag{11}$$

其中，w 是速度权重，c_1 和 c_2 是学习变量，通常取值为 2，r_1 和 r_2 是[0,1]区间上的随机数，设粒子在不同维上速度分量的最大值是 v_{max}($v_{max}>0$)，且当 $v_{ij}^{t}>v_{max}$ 时，$v_{ij}^{t}=v_{max}$；当 $v_{ij}^{t}<-v_{max}$ 时，$v_{ij}^{t}=-v_{max}$。利用式(11)进行位置更新时，每一维的位置变量先对 n_j 求余，然后取绝对值以保证结果非负，最后向上取整。

3.3 更新与维护非支配解集

本文通过计算粒子的信息熵来保持非支配解的多样性。假设 F_i 是粒子 x_i 的适应度，则计算步骤如下：

每一个粒子的信息熵 $H_{i,j}$ 为

$$H_{i,j}=-P\times\ln P,i=1,2,\cdots,N_s \tag{12}$$

其中，N_s 为非支配解集中粒子个数。根据粒子的信息熵可以得到不同粒子间的相似度 $\gamma_{i,j}$：

$$\gamma_{i,j}=1/(1+H_{i,j}) \tag{13}$$

粒子 x_i 在非支配解集中的密度 $D_i=m/N_s$，m 代表在非支配解集中和粒子 x_i 的相似程度 $\gamma_{i,j}\geqslant 0.9$ 的粒子的数量，则粒子 x_i 的适应度 F_i 为

$$F_i=1/D_i=N_s/m \tag{14}$$

由上述计算过程可知，相似个体的数量和个体适应度具有反比关系。如果将解集内的粒子按适应度降序排列，并依据预设阈值去除排序靠后的粒子，则可以保证解集中粒子分布的均匀性以及解的多样性。

3.4 选择最优位置

在算法的运行过程中，每次迭代时需要确定全局最优位置和局部最优位置。本文采用 Mostaghim[15] 提出的 Sigma 方法确定粒子的全局最优位置，该方法能有效提升种群的收敛速度。

假设优化问题有两个目标函数 f_1、f_2，解空间有粒子 x_i，其在两个目标函数组成的二维空间中的向量值为($f_{1,i}$，$f_{2,i}$)，定义该粒子的 Sigma 值为 α_i。

$$\alpha_i=\frac{f_{1,i}^2-f_{2,i}^2}{f_{1,i}^2+f_{2,i}^2} \tag{15}$$

当扩展到多维解空间时，Sigma 值 α_i 是一个具有 C_m^2 元素的向量，其中 m 表示该优化问题具有 m 个优化目标。向量 $\vec{\alpha_i}$ 中的每一个元素由解空间中任意两个坐标之间的所有可能组合按照式(16)计算得到，例如 3 个目标函数对应三维解空间 f_1、f_1 和 f_3，则 Sigma 值 $\vec{\alpha_i}$ 可以表示如下：

$$\vec{\alpha_i}=\begin{pmatrix}f_{1,i}^2-f_{2,i}^2\\ f_{2,i}^2-f_{3,i}^2\\ f_{3,i}^2-f_{1,i}^2\end{pmatrix}/(f_{1,i}^2+f_{2,i}^2+f_{3,i}^2) \tag{16}$$

确定粒子 x_i 的全局最优位置主要分为两步：首先依照上述方法计算非支配解中所有粒子的 Sigma 值 $\vec{\alpha_j}$；之后，计算粒子 x_i 对应的 Sigma 值 $\vec{\alpha_i}$，计算 $\vec{\alpha_i}$ 与非支配解中所有粒子对应的 Sigma 值在解空间中的距离，则距离最小的非支配解即为该粒子的全局最优位置。在双目标对应的二维空间中，种群中粒子的全局最优位置确定情况如图 3 所示。

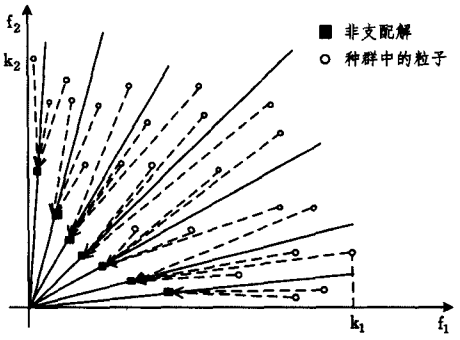

图 3 基于 Sigma 方法寻找全局最优位置示意图

当目标函数对应的值域范围不同时，需要将解空间中各维进行归一化处理。假设目标函数的值域为[0，K_1]，某个粒子对应的目函数值为 $f_1(x_i)$，则进行如下归一化处理：

$$f_1'(x_i)=f_1(x_i)/K_1 \tag{17}$$

对于粒子的局部最优位置，通过支配关系进行确定。如果粒子的当前位置对 P_i 具有支配关系，则用当前位置替代 P_i；如果 P_i 对当前位置具有支配关系，则 P_i 保持不变；若上述两种支配关系皆不成立，则从当前位置和 P_i 中任选其一作为粒子的局部最优位置。

3.5 混沌扰动

当非支配解集的多样性较差时，算法易陷入局部最优，从而失去全局寻优能力。为此，借鉴混沌理论，通过混沌扰动帮助种群摆脱局部最优的束缚，进而提高算法的全局寻优能力。具体实现过程如下：

1)采用文献[16]提出的方法测量种群多样性 $div(t)$，并通过检测 $div(t)$ 是否超出预设阈值来判定种群是否陷入停滞。

$$div(t)=\frac{1}{N}\times\frac{1}{R}\times\sum_{i=1}^{n}\sqrt{\sum_{j=1}^{D}(x_{ij}^{t}-\overline{x_j^{t}})^2} \tag{18}$$

其中，t 表示种群的迭代次数，N 代表种群内的粒子个数，R 表示搜索空间半径的最大值，x_{ij}^{t} 表示在 t 代时第 i 个粒子的第 j 维上的值，$\overline{x_j^{t}}$ 表示种群在第 j 维上的平均值，多样性的阈值定义为 $\eta\times div$，其中 $\eta=0.4$，理论计算和实验均表明此时种群趋于停滞状态[16]。

2)将非支配解按照适应度进行降序排列，取前($N/2$)个粒子进行混沌扰动。利用式(8)得到 $\beta_{ij}^{(h)}$($h=1,2,\cdots,H$)，混沌迭代次数的最大值为 H。

3)通过混沌扰动偏差 $Bias(c_{ij})$ 控制粒子扰动的强度，将混沌扰动的影响控制在合理范围内。

$$Bias(c_{ij})=\lceil\delta(H-h)Rand\rceil \tag{19}$$

其中，σ 是比例因子，h 代表当前的迭代次数，$Rand()$ 是定义在[0,1]区间上的随机数。

4)根据式(20)计算载入混沌扰动序列后粒子的每一维的新位置变量，在每个非支配粒子附近产生 H 个邻域粒子 $c_i^{(h)*}=(c_{i1}^{(h)*},c_{i2}^{(h)*},\cdots,c_{im}^{(h)*})(h=1,2,\cdots,H)$，从中选定非支配粒子 c_i^{*}，称为经混沌扰动后新生成的粒子，由此得到($N/2$)个新生成的粒子，用这些新粒子随机替换种群中的粒子。

$$c_{ij}^{(h)*} = c_{ij} - Bias(c_{ij}) + \lceil 2Bias(c_{ij})\beta_{ij}^{(h)} \rceil \quad (20)$$

经过以上步骤，针对解空间中的稀疏粒子，利用混沌扰动在其邻近域内生成了众多新粒子，使种群多样性得到有效增强，帮助种群摆脱局部最优的束缚，提高了算法的全局寻优能力。

3.6 混沌多目标粒子群算法流程

混沌多目标粒子群算法流程如图 4 所示。

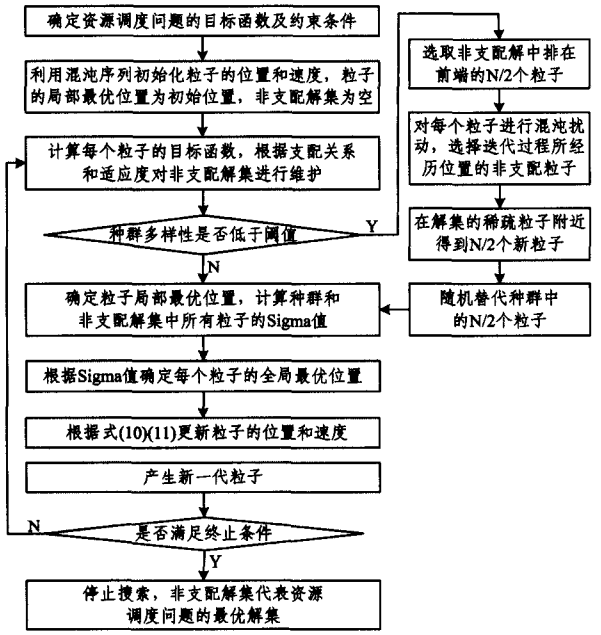

图 4 算法流程

4 仿真实验与分析

4.1 实验设计

本文采用云计算仿真平台 CloudSim[17] 进行仿真实验，通过扩展 CloudSim 的 DatacenterBroker 类，模拟采用本文提出的 CMOPSO 算法和 MOPSO 算法进行对比实验。实验用计算机配置为 Core2 处理器，2G 内存，Windows XP 操作系统，算法在 Matlab2012 下实现。多目标资源调度模型中的最小可靠性 R_0 为 0.1，最小信誉度为 2。CMOPSO 算法参数设置如下，速度权重 $w=0.75$，种群规模 $N=50$，非支配解集 $N_S=50$，混沌迭代最大次数 $H=10$，比例变量 $\delta=0.9$。实验假设任务数量为 50，每个任务被分为 20 个 Map 子任务和 10 个 Reduce 子任务。采用中国电子学会主办的 2013 届中国 WEB 服务竞赛(China Web Service Cup 2013)提供的测试数据生成器产生实验测试集[18]。由于测试集受概念数量(CN)和组合解深度(Solution Depth)两个参数的影响，本文的实验中设定 $CN=3000$，$Solution\ Depth=6$。下面分别通过实验测试 MOPSO 和 CMOPSO 算法的收敛速度和非支配解分布情况。

4.2 算法收敛性对比实验

实验 1 设定资源数量为 30，图 5 给出两种算法的迭代收敛过程。可以看出，两种算法都具有向最优解收敛的趋势，在迭代 50 次之后，两种算法都达到完全收敛。CMOPSO 在第 35 代完成算法收敛，而 MOPSO 在第 44 代完成算法收敛。CMOPSO 在收敛速度上明显优于 MOPSO 算法。

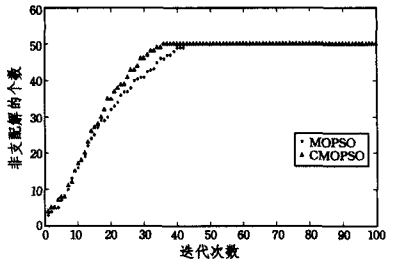

图 5 算法迭代收敛性的比较

将资源数量设定为 50、70、100、120，分别进行实验，结果如表 2 所列。

表 2 不同资源数量下算法收敛性对比

	资源数量	算法收敛所需的迭代次数	
		MOPSO 算法	CMOPSO 算法
实验 1	30	44	35
实验 2	50	54	41
实验 3	70	65	46
实验 4	100	78	53
实验 5	120	93	61

由图 6 可以看出，随着资源数量的增加，求解空间变大，两种算法达到收敛的次数都随之增大。但是 CMOPSO 算法增长较慢，且在所有实验中收敛速度都优于 MOPSO 算法。

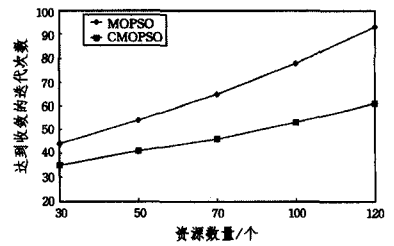

图 6 不同资源规模的算法收敛速度对比

4.3 算法非支配解分布性对比实验

设定资源数量为 30，图 7 和图 8 分别给出两种算法非支配解的分布情况。通过比较可以看出，CMOPSO 算法搜索到的非支配解的分布均匀性更好，具有更强的多样性。

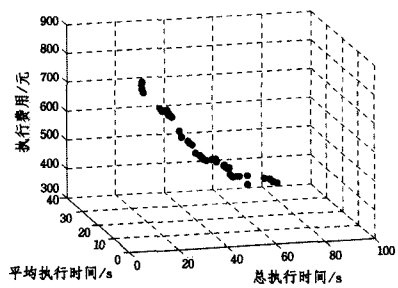

图 7 MOPSO 算法非支配解的分布情况

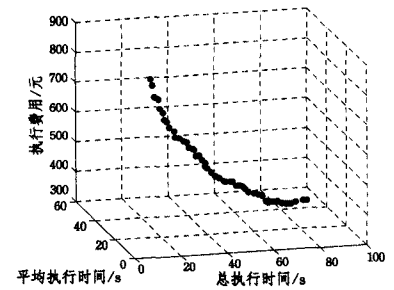

图 8 CMOPSO 算法非支配解的分布情况

为定量评价非支配解的分布均匀性，参考文献[13]，定义非支配解的间距 S 和最大散布范围 D 如下，公式中参数的具体含义详见文献[13]。

$$S=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d}-d_i)^2},\ D=\sqrt{\sum_{p=1}^{3}(\max_{i=1}^{n}f_p^i-\min_{i=1}^{n}f_p^i)^2}$$

在资源数量分别为 30、50、70、100、120 时进行实验，得到 MOPSO 和 CMOPSO 算法的最大间距 S、最大散布范围 D，结果如表 3 所列。可以看出，CMOPSO 算法得到的非支配解之间的最大间距较小，解集整体的散布范围更大。

表 3　不同资源数量下算法非支配解的分布均匀性对比

	资源数量	最大间距 S		最大散布范围 D	
		MOPSO	CMOPSO	MOPSO	CMOPSO
实验 1	30	0.213	0.098	851	1133
实验 2	50	0.301	0.071	877	1221
实验 3	70	0.195	0.082	1643	2012
实验 4	100	0.502	0.176	1462	1873
实验 5	120	0.275	0.184	1496	1520

通过上述分析可知，混沌多目标粒子群算法的全局寻优能力更强，向非支配解集收敛的速度更快，得到的优化解的多样性更好且分布更加均匀，能够提供给用户一个满足约束条件的非支配解集，便于用户依据应用需求和实际限制选择合适的优化解。

结束语　为了在云计算环境中实现资源的高效调度，有力地保证云服务质量，本文基于 Map-Reduce 模型，将资源调度问题转化为带 QoS 约束的多目标寻优问题，提出了一种基于混沌多目标粒子群算法的求解方法。算法通过计算粒子的信息熵保持了解集的多样性，采用 Sigma 方法加快算法收敛速度，并在算法执行过程中种群多样性丢失时，引入混沌扰动机制在解空间的稀疏粒子附近产生新粒子，引导种群向优势空间进化，避免算法进入局部最优。实验表明，与 MOPSO 比较，CMOPSO 算法的收敛速度和非支配解的多样性均显著提高，其能够更好地实现云计算中的资源高效调度。

由于云计算的资源调度分为任务与虚拟机的匹配以及虚拟机与主机的映射，而本文只考虑了任务与虚拟机之间的调度，因此如何高效地解决虚拟机与主机的映射问题是下一步研究的重点。

参 考 文 献

[1] Buyya R, Yeo C S, Venugopal S, et al. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility [J]. Future Generation Computer Systems, 2009, 25(6): 599-616

[2] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50-58

[3] 罗军舟，金嘉晖，宋爱波，等. 云计算：体系架构与关键技术[J]. 通信学报，2011，32(7)：3-21
Luo Jun-zhou, Jin Jia-hui, Song Ai-bo. et al. Cloud computing: architecture and key technologies [J]. Journal on Communications, 2011, 32(7): 3-21

[4] 王鹏. 云计算的关键技术与应用实例[M]. 北京：科学出版社，2010：11-13
Wang Peng. Key technology and application example of cloud computing [M]. Beijing: Science Press, 2010: 11-13

[5] Dean J, Ghemawat S. MapReduce: simplified data processing on large cluster[C]// Proc of the 6th Conference on Symposium on Operating System Design and Implementation (SOSDI 2004). New York: ACM Press, 2004: 137-150

[6] 张春艳，刘清林，孟珂. 基于云计算环境的蚁群优化计算资源分配[J]. 计算机应用，2012，32(5)：1418-1420
Zhang Chun-yan, Liu qing-lin, Meng Ke, et al. Task allocation based on ant colony optimization in cloud computing [J]. Journal of Computer Applications, 2012, 32(5): 1418-1420

[7] 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
Li Jian-feng, Peng Jian. Task scheduling algorithm based on improved genetic algorithm in cloud computing environment[J]. Journal of Computer Applications, 2011, 31(1): 184-186

[8] 孙黎阳，林剑柠，毛少杰. 基于改进粒子群优化算法的网络化仿真任务共同体服务选择[J]. 兵工学报，2012，33(11)：1393-1403
Sun Li-yang, Lin Jian-ning, Mao Shao-jie, et al. Service Selection of Network Simulation Task Community Based on Improved Particle Swarm Optimization Algorithm[J]. Acta Armamentarii, 2012, 33(11): 1393-2002

[9] 梁静，许波，葛宇. 基于改进蛙跳策略的 Map-Reduce 作业调度算法[J]. 计算机应用研究，2013，30(7)：1999-2002
Liang Jing, Xu Bo, Ge Yu. Map-Reduce job scheduling algorithm based on improved shuffled frog leaping strategy[J]. Application Research of Computers, 2013, 30(7): 1999-2002

[10] 陆路. 云环境下作业调度算法的研究[D]. 南京：南京理工大学，2013
Lu Lu. Research of job scheduling algorithms under cloud computing environment[D]. Nanjing: Nanjing University of Science and Technology, 2013

[11] 孙大为，常桂然，李凤云，等. 一种基于免疫克隆的偏好多维 QoS 云资源调度优化算法[J]. 电子学报，2011，39(8)：1824-1831
Sun Da-wei, Chang Gui-ran, Li Feng-yun, et al. Optimizing multi-dimensional QoS cloud resource scheduling by immune clonal with preference[J]. Acta Electronica Sinica, 2011, 39(8): 1824-1831

[12] 张伟星. 基于粒子群优化算法的动态多目标优化算法研究及应用[D]. 郑州：郑州大学，2013
Zhang Wei-xing. The Research and Application of Dynamic Multi-objective Optimization Based on Particle Swarm Optimization[D]. Zhengzhou: Zhengzhou University, 2013

[13] 裴胜玉，周永权. 一种基于混沌变异的多目标粒子群优化算法[J]. 山东大学学报(理学版)，2010，45(7)：18-23
Pei Sheng-yu, Zhou Yong-quan. A mult-objective particle swarm optimization algorithm based on the chaotic mutation [J]. Journal of Shandong University (Natural Science), 2010, 45(7): 18-23

[14] 郑金华. 多目标进化算法及其应用[M]. 北京：科学出版社，2007：4-6
Zheng Jin-hua. Multi-objective evolution algorithm and its applications[M]. Beijing: Science Press, 2007: 4-6

[15] Mostaghim S, Teich J. Strategies for finding good local guides in multi-objective particle swarm optimization[C]// Proceeding of the 2003 IEEE Swarm Intelligence Symposium Indianapolis. Dallas: ACM Press, 2003: 26-33

[16] Fang Yi-qiu, Wang Fei, Ge Jun-wei. A task scheduling algorithm based on load balancing in cloud computing[C]// Proceedings of the 2th International Conference on Web Information Systems and Mining. Berlin, Germany: Springer-Verlag, 2010: 156-162

[17] The Cloud Lab. Cloudsim[EB/OL]. 2011-08-15. http://www.cloudbus.org/cloudsim

[18] 2013 China Web Service Cup. Experiment Data Lab 2012. 10. 21. [EB/OL]. http://debs.ict.ac.cn/2013