文章编号:1671-4598(2016)05-0202-05 DOI:10.16526/j.cnki.11—4762/tp.2016.05.057 中图分类号:TP393 文献标识码:A

一种用于云计算资源调度的改进遗传算法

刘 峰[1],毕 利[1],杨 军[2]
(1.宁夏大学 数学计算机学院,银川 750021;
2.宁夏大学 计算机网络管理中心,银川 750021)

摘要:针对轮询调度算法、遗传算法和模拟退火算法在云计算资源调度中存在收敛速度慢、易早熟和资源负载不均衡等问题,提出了一种基于模拟退火思想的改进遗传算法(simulated annealing improved genetic algorithm:SAIGA);改进算法设计了基于任务平均完成时间和负载均衡的双适应度函数和自适应的交叉变异概率函数,允许算法在退火过程中以一定概率接受劣质解从而避免早熟现象的发生,将虚拟资源上任务分配数的标准差作为选择个体的依据来实现节点的负载均衡;仿真结果表明,改进算法与上述算法相比,在任务平均完成时间、资源利用率以及收敛速度上表现得更优越,能够较快地找到资源最优调度方案,具有较好的可行性和实用性。

关键词:云计算;轮询调度;模拟退火思想;改进遗传算法;负载均衡

An Improved Genetic Algorithm for Cloud Computing Resource Scheduling

Liu Feng[1], Bi Li[1], Yang Jun[2]
(1. School of Mathematics and Computer Science, Ningxia University, Yinchuan 750021, China;
2. Network Administration Center, Ningxia University, Yinchuan 750021, China)

Abstract: For Round-Robin scheduling algorithm and genetic algorithm and simulated annealing algorithm in cloud resource scheduling having shortcomings, such as slow convergence speed, easy to premature and the imbalance of the resource load, the paper proposed the improved genetic algorithm combined with simulated annealing thought (Simulated Annealing Improved Genetic Algorithm: SAIGA). The improved algorithm gave a dual fitness function based on task average completion time and load balance and adaptive crossover mutation probability function. It allowed the algorithm in the annealing process to accept inferior solution with a certain probability to avoid prematurity phenomenon occurs. We regarded the virtual machine task allotment standard deviation as the basis of individual choice to realize the resource node load balancing. Simulation experiments showed that the improved algorithm is more superior on average task completion time, resource load balancing, and the convergence rate. It can rapidly find the optimal scheduling scheme and has good feasibility and practicability.

Keywords: cloud computing; round robin scheduling; simulated annealing thought; improved genetic algorithm; load balancing

0 引言

云计算作为继网格计算、并行计算和分布式计算之后的新兴计算模式,被高校、科研机构以及商业组织进行了大量的研究。中国网格计算、云计算专家刘鹏给出如下定义:“云计算将计算任务分布在大量计算机构成的资源池上,使各种应用系统能够根据需要获取计算力、存储空间和各种软件服务[1]”。然而,云计算服务的核心问题是资源的调度,其服务质量的好坏取决于调度效率的高低,因此资源调度一直是云计算研究领域中的重点和难点[2]。

传统资源调度算法大都是一些静态算法,如贪心算法、轮询调度算法等,这类方法对于少量任务的调度可以获得较好的调度方案。随着任务数的不断增加,云计算系统的复杂性增大,传统方法存在任务完成时间过长、节点间负载不均衡和资源利用率低等缺点[3-4]。目前,人工智能技术的日趋成熟,智能算法越来越多地被用于资源调度中,成为当前研究的热点。

刘瑜等[5]提出了基于最优跨度和负载均衡改进遗传算法的资源调度策略,考虑到云计算任务的数量大和复杂性,设计了基于资源数的编码方式,在算法接近收敛阶段自适应调节最优跨度适应度函数来提高算法的收敛速度。袁浩等[6]提出了一种基于社会力群的智能优化算法,通过模拟人群的拥挤退让行为寻找使任务总时间最小的调度方案。徐文忠[7]等提出了一种新的基于遗传算法的关于虚拟机负载均衡的调度策略,但没有考虑到任务平均完成时间指标,不能满足用户对任务响应时间的需求。李建锋等[8]设计了双适应度函数去优化资源调度过程,寻求使总任务完成时间和任务平均完成时间都较小的调度方案,并取得了不错的效果。薛玉[9]提出一种基于混沌粒子群算法的云资源调度模型,把资源的负载均衡度作为要优化的目标函数,在寻优过程中人工引入混沌机制对粒子进行扰动,通过粒子间的信息交流与共享,寻求调度最优解。邬海艳[10]提出了基于元胞自动机模型改进遗传算法用于云资源调度,其主要作用于遗传算法的选择和交叉操作中,根据演化规则自适应调节元胞与邻居元胞的状态,获取最大适应度值个体。Gao等[11]提出一种以任务完成时间、系统吞吐量为优化目标的蚁群算法构建云计算资源调度模型,在任务总完成时间上取得不错的效果,但资源的利用率不高。Zhan Zhi-Hui等[12]提出了基于Min-min和上Max-min方法的负载均衡感知遗传算法

收稿日期:2015-11-09; **修回日期**:2015-12-11。
基金项目:国家自然科学基金项目(61261001);教育部科学技术研究重点项目(212189)。
作者简介:刘 峰(1989-),男,山东菏泽人,硕士研究生,主要从事智能调度算法方向的研究。
毕 利(1968-),女,宁夏银川人,教授,硕士生导师,主要从事数据挖掘及组合优化控制方向的研究。
通讯作者:杨 军(1972-),男,宁夏吴忠人,教授,硕士生导师,主要从事云计算资源调度及无线传感器网络方向的研究。

(LAGA)，通过在适应度函数中引入时间负载均衡模型(TLB)选择优势个体，生成资源节点负载更均衡的调度方案，仅考虑了资源的负载情况，却对牺牲了算法的收敛速度。

针对以上问题，本文结合模拟退火的思想来改进遗传算法在云计算资源中的调度，提出了基于任务平均完成时间和负载均衡的双适应度函数，设计自适应的交叉和变异概率函数，使算法在寻优时以一定概率接受劣质解，减小改进算法收敛到局部最优解的可能性，同时采用基于资源节点数作为染色体的总长度，每个资源所映射的任务总数和任务编号作为基因值的编码方式，加快了算法的收敛速度，提高了算法的寻优能力。

1　云计算资源调度问题描述

在云计算平台上，管理者利用虚拟化技术把计算机的物理资源抽象为统一的虚拟资源，统一的虚拟资源又被分配到相互独立的虚拟机上，而云计算中的资源调度问题就是利用一种调度策略把大量的计算任务分配到虚拟机上，实现虚拟资源的最大化利用。具体的讲，就是如何把云计算中大量的计算任务根据一定的调度策略分配到最佳虚拟资源上，保证最少的任务完成时间和最大的资源利用率，云计算资源调度管理模型如图 1。

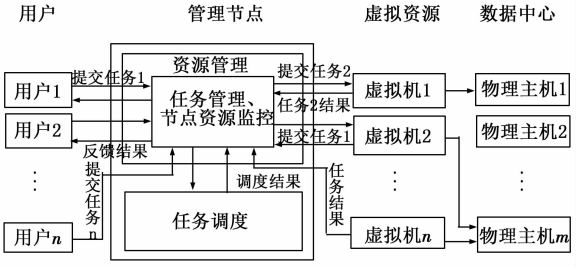

图 1　云计算资源调度管理模型

云计算资源调度包括数据中心 $D=\{d_1,d_2,\cdots,d_m\}$

虚拟资源 $V=\{vm_1,vm_2,\cdots,vm_m\}$、计算任务 $T=\{t_1,t_2,\cdots,t_n\}$ 以及它们之间的映射关系。云计算资源调度模型可描述为：

$$S=\{T,V,D,M_{tv},M_{vd}\} \tag{1}$$

式中，M_{tv} 为任务与虚拟资源之间的映射关系，M_{vd} 为虚拟机与数据中心的之间的映射关系。根据上述用户任务－虚拟资源之间的映射关系，本文用二维数组统计每个任务在每个虚拟资源上的预计完成执行时间 ETC (Excepted Time To Completion)。

$$ETC[n][m]=\begin{pmatrix} ETC_{11} & \cdots & ETC_{1m} \\ \vdots & ETC_{ij} & \vdots \\ ETC_{n1} & \cdots & ETC_{nm} \end{pmatrix} \tag{2}$$

其中：$ETC_{ij}=cloudlet[i].length/vm[j].mips$，表示第 i 个任务在第 j 个资源上的预计执行时间。设 $Start(i,j)$ 为任务 t_i 在 vm_j 上的开始执行时间。则 t_i 在 vm_j 上的完成时间见式 (3)。

$$Finish(i,j)=Start(i,j)+ETC_{ij} \tag{3}$$

那么 vm_j 上全部任务的完成时间见式 (4) (5)。

$$Sum(vm_j)=\sum_{i=1}^{n}c_{ij}Finish(i,j) \tag{4}$$

$$c_{ij}=\begin{cases}1, t_iM_{tv}=vm_j\\ 0, t_iM_{tv}\neq vm_j\end{cases} \tag{5}$$

式 (5) 中，$c_{ij}=1$ 表示任务 t_i 在 vm_j 上执行，$c_{ij}=0$ 表示 t_i 不在 vm_j 上执行。

对于用户全部计算任务 $T=\{t_1,t_2,\cdots,t_n\}$ 的任务平均完成时间见式 (6)。

$$Aver(T)=\frac{1}{n}\sum_{j=1}^{m}Sum(vm_j) \tag{6}$$

对于云计算资源的调度问题，即求使式 (6) 值最小的解，因此本文的目标函数见式 (7)。

$$ObjFun(T)=\min\left\{\frac{1}{n}\sum_{j=1}^{m}Sum(vm_j)\right\} \tag{7}$$

2　遗传算法与模拟退火思想的融合设计

2.1　模拟退火思想

模拟退火思想[13-14]最早是由 Metropolis 于 1953 年提出，源于固体的退火过程，即先将温度加到很高，再缓慢降温（退火），使内能达到最低点。SA 算法的搜索过程有一种时变且概率最终趋于零的突跳性，在很大程度上可以避免陷入局部最优解。一般地，把固体内能模拟成要优化的目标函数 f，随机生成初始解 ω，计算目标函数值 $f(\omega)$，利用扰动机制产生新解 ω'，计算 $f(\omega')$ 与 $f(\omega)$ 的差值，根据 Metropolis 准则判断是否接受新解。设置温度下降函数 $T_{k+1}=\alpha T_k$，继续迭代寻优，当温度下降到最低 t_f 时或者算法在 50 代内目标值无变化，则算法结束，Metropolis 准则见式 (8)。

$$P=\begin{cases}1, f(\omega'_k)<f(\omega_k)\\ \exp\left(\dfrac{f(\omega_k)-f(\omega'_k)}{T_k}\right), f(\omega'_k)\geqslant f(\omega_k)\end{cases} \tag{8}$$

2.2　基于模拟退火思想的改进遗传算法设计

2.2.1　染色体编码与种群初始化

在任务调度的问题中，传统遗传算法通常将任务总数作为染色体基因串的长度，每个任务对应的资源 ID 作为染色体的基因值。但是云计算环境中任务的总数远远大于资源节点的个数，如果把任务总数作为染色体基因串的长度，会导致算法收敛速度慢，找到最优解的时间过长。因此本文对染色体的编码方式进行了改进，将资源节点的数量作为染色体基因串的长度，每个资源所映射的任务总数和任务编号作为染色体基因值，这种编码方式可以加快算法的收敛速度。本文采用资源——任务的间接编码方式，先对染色体进行预编码，设有 n 个任务，m 个资源节点，如下产生一条染色体。

$$\{3,1,1,2,3,\cdots,m-1,m,m-1\}$$

表示第 1、5 个任务分配到第 3 个资源上，第 2、3 个任务分配到第 1 个资源上，第 $n-2$，$n-1$ 个任务分配到第 $m-1$ 资源上，第 n 个任务分配到第 m 个资源上。接着对该染色体进行二次编码，则染色体基因串的长度为资源总数 m，染色体基因值为每个资源节点分配任务的总数和任务 ID，二次处理后的染色体见表 1。

表 1　二次处理后染色体的编码方式

资源 ID	1	2	…	M
任务总数	2	5		2
任务 ID	{2,3}	{1,4,6,10}	{…}	{n−1,n−2}

在对染色体进行编码后，接着对种群进行初始化，本文采用随机方式产生 N 个染色体。

2.2.2　基于任务平均完成时间和负载均衡的双适应度函数设计

基于任务平均完成时间的适应度函数见式 (9)。

$$f(k)=\frac{1}{\min\left\{\frac{1}{n}\sum_{j=1}^{m}Sum^{k}(vm_j)\right\}},1\leqslant k\leqslant N \tag{9}$$

式（9）表示第 k 个染色体上所有任务在节点 j 上平均执行时间。在任务调度的过程中，我们还需要考虑资源负载均衡问题，节点负载较均衡使得资源利用率较高，避免计算能力高的节点出现负荷同时能力低的节点被闲置。采用资源节点上任务分配数的标准差来衡量节点的负载均衡问题，设任务数为 n，资源节点数为 m，则每个资源节点上平均分配的任务数为 $\frac{n}{m}$，资源节点任务分配数标准差的适应度函数见式（10）。

$$\sigma(k)=\sqrt{\frac{\sum_{i=1}^{m}(Assign\_tasks_{k,i}-Aver\_tasks)^2}{m}},1\leqslant k\leqslant N \tag{10}$$

其中：$Assign\_tasks_{k,i}$ 表示第 k 个染色体上的第 i 个资源节点所分配到的任务数。

所以，本文设计基于任务平均完成时间和负载均衡的双适应度函数见式（11）。

$$f(k)=\omega_1\frac{1}{\min\left\{\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}ETC_{ij}^{k}\right\}}+\omega_2\frac{1}{\sqrt{\frac{\sum_{i=1}^{m}(Assign\_tasks_{k,i}-Aver\_tasks)^2}{m}}},1\leqslant k\leqslant N \tag{11}$$

ω_1 和 ω_2 分别是基于最小完成时间和任务分配数标准差适应度函数的权重。用户根据自己的偏好决定每个适应度函数的权重，且满足 $\omega_1+\omega_2=1$。

2.2.3 选择算子设计

在选择算子的设计采用改进的轮盘赌方法。首先根据式（9）计算每个染色体的适应度值，然后计算该适应度值相对于整个种群适应度值总和中所占的比例，即该个体被复制到下一代中的概率。则个体 k 被选中的概率见式（12）。

$$P(k)=\frac{fitness(k)}{\sum_{k=1}^{n}fitness(k)}k=1,2,\cdots,n \tag{12}$$

2.2.4 交叉和变异算子设计

本文采用模拟退火的思想对个体进行交叉操作，种群中较优个体以较低概率进行交叉，增加种群中“弱势”个体的交叉可能性，提高种群的多样性，使算法陷入局部最优解的可能性减小。交叉概率函数见式（13）。

$$P_c=\begin{cases}\exp\left(-\frac{f_2+f_1-2f_{avg}}{\alpha\cdot T}\right),f_2>f_1\geqslant f_{avg}\\ p_1,\end{cases} \tag{13}$$

其中：f_{avg} 表示每代群体的平均适应度值，f_2，f_1 分别表示随机选择的两个个体中适应度值的较大者与较小者。α 为退火系数，T 为算法当前迭代的温度。当适应度函数值较大时（$f_2>f_1>f_{avg}$），这时让交叉概率 P_c 变小，防止较优的个体被破坏，同时较小的交叉概率能加快种群向最优解收敛。当适应度函数值较小时（$f_1<f_2<f_{avg}$），使交叉概率变大，期望重组出新的个体。交叉算子有利于在全局范围内找到较好的个体，容易出现在最优解附近徘徊的现象，对局部解空间的寻优能力较差，使用变异算子可以微调个体的基因值，提高全局的收敛精度，个体的变异概率函数见式（14）。

$$P_m=\begin{cases}\exp\left(-\frac{f'-f_{avg}}{\alpha\cdot T}\right),f'>f_{avg}\\ p_2,f'\leqslant f_{avg}\end{cases} \tag{14}$$

其中，f' 表示随机选择的个体适应度函数值，f_{avg} 表示种群适应度的平均值。当随机选择个体的适应度值大于平均适应度值时（即 $f'>f_{avg}$），根据上面的公式，f' 越大，种群中个体的变异概率则越小。这样可以在很大程度上避免较优个体的结构被破坏。

2.2.5 算法终止条件

当连续 60 代适应度函数值无变化或者进化代数达到 200 次时，终止算法的执行。

本文使用均匀交叉、多点交叉相结合的实现方式，变异方式使用基因值变异。最后基于模拟退火的改进遗传算法（SAIGA）流程图如图 2 所示。

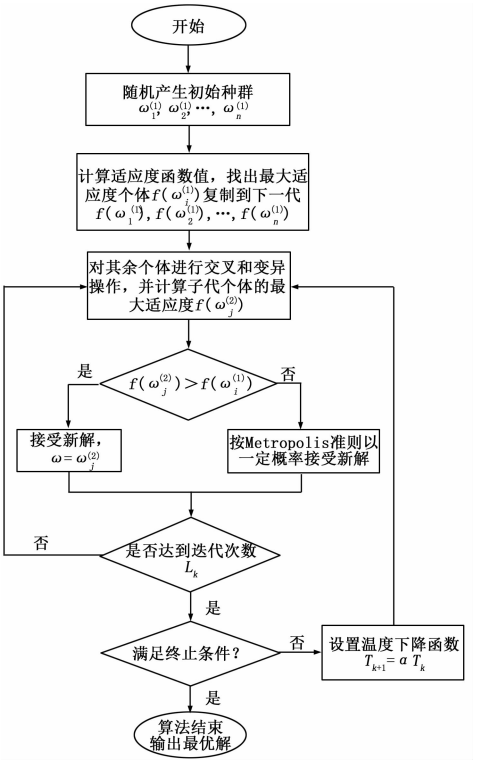

图 2 改进算法的流程图

3 仿真实验

3.1 仿真环境

在 Intel Core2 四核 2.83 GHz CPU，4 G RAM，操作系统为 Windows7 32 bit，采用 CloudSim3.0 软件对算法进行仿真，其核心是中心代理人和虚拟机调度策略，改进算法通过扩展 DatacenterBroker 和 VMAllocationPolicy 类实现云资源调度的模拟。

3.2 对比算法

本文选择轮询调度算法（RR）、遗传算法（GA）和模拟退火算法（SA）做对比实验，分析比较 4 种算法在任务总完成时间、任务平均完成时间、负载均衡以及收敛速度方面的优

劣性。GA、SA 和 SAIGA 算法的参数设置见表 2。

表 2　GA、SA 及 SAIGA 算法参数设置表

算法	参数	取值	算法	参数	取值
GA	N	100	SAIGA	N	100
	p_c	0.6		T_0	100
	p_m	0.1		α	0.9
	L	200		p_1	0.8
SA	T_0	100		p_2	0.1
	α	0.95		ω_1	0.8
	L_k	200		ω_2	0.2
				L_k	200

3.3　结果与分析

实验中模拟任务的长度 MI 取 1 000～20 000 之间，资源的运算能力 MIPS 取 500～5 000 之间，分析任务数和虚拟资源数的变化对任务总完成时间、平均完成时间以及资源负载均衡的影响。

1）将任务分配到 10 个资源节点，改变任务的数量从 10 到 200，记录任务的总完成时间（ms），结果见表 3，图 3。

表 3　资源数一定，不同任务数下的完成时间

算法	不同任务数下的总完成时间(ms)						
	10	20	50	80	120	160	200
RR	261	548	1132	2433	7826	17908	33551
GA	78	99	291	982	5440	13212	25350
SA	83	106	324	1130	5842	14620	27720
SAIGA	79	94	190	467	4199	11477	21857

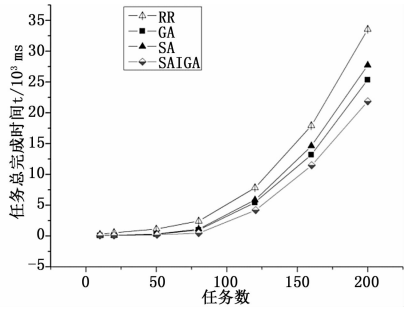

图 3　资源数一定，不同任务数下的总完成时间

从图 3 可以看出，在任务数不多时，4 种算法下任务的完成时间差别不大，这是由于初期资源节点的数量和任务的数量差别不大，节点有足够的能力去处理这些任务。随着任务数的增多，特别地当任务数大于 150 时，SAIGA 算法调度下的任务总完成时间明显优于其他 3 种算法。

2）当任务数量固定为 1 000 时，改变虚拟资源的数量，从 10 到 50，记录任务的平均完成时间如图 4。

从图 4 可以看出，在虚拟资源数较少时，SAIGA 的任务完成时间远远小于 RR、GA 和 SA。随着虚拟资源数量的增加，4 种算法下任务的完成时间都在不断减少，最后 SAIGA、GA 和 SA 算法的任务平均完成时间大致接近，这是由于随着资源节点数量的增加，其处理能力能够逐渐满足任务的执行需求，任务抢占资源的情况减少。

3）任务数为 1 000 时，将任务分配到 5 个资源节点（R1，R2，R3，R4，R5）上，设置其处理能力分别为：（1 200，900，600，1800，800）MIPS，记录 5 个资源节点上的负载情况如图 5。

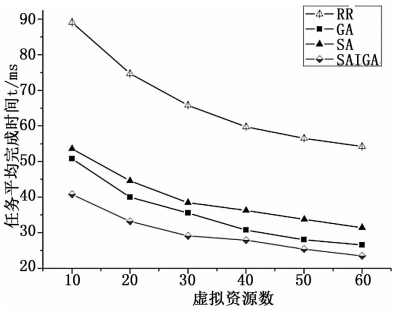

图 4　不同资源数下任务的平均完成时间

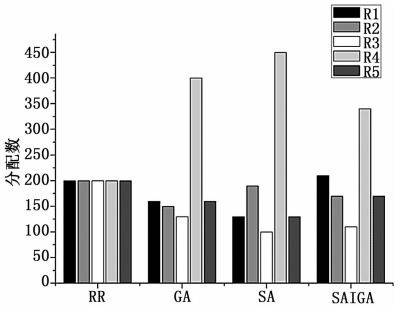

图 5　大量任务下资源节点的负载情况

从图 5 可以看出，在资源计算能力有较大差异的情况下，SAIGA 算法的分配的任务数更加均衡，而 RR 没有考虑到节点能力的差异性，采取了均分任务的方式，GA、SA 中则出现计算能力高的节点分配到任务数较多，计算能力低的节点分配到的任务数较少的现象。总之，SAIGA 算法在资源节点的负载均衡方面表现出较好的优越性。

4）任务数为 1 000 时，资源节点为 40 时，GA 和 SAIGA 算法的迭代收敛情况如图 6。

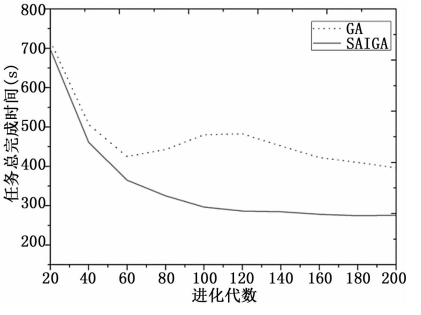

图 6　SAIGA 和 GA 算法收敛结果比较

从图 6 可以看出，在算法迭代初期 GA 和 SAIGA 性能差不多，但由于 SAIGA 采用任务总数加编号的方式作为染色体的基因值，使它后期的收敛速度加快。SAIGA 算法在迭代 120 次后开始收敛，而 GA 算法在接近 160 代时才出现收敛的趋势，在收敛精度上改进算法提高了近 130 秒。另外，GA 算法迭代大约至 60 代时出现一些适应度值超常的个体误导了种群进化的方向，但这些个体并不是算法最优解。由于 SAIGA 采用自学习的交叉概率函数，在算法接近收敛的阶段对超常适应度值的个体进行了一定概率的调整，避免误导了种群的进化方向，使得改进算法更快地收敛到最优解。

4 结束语

针对云资源调度的动态性、实时性等特点，提出了一种基于模拟退火思想的改进遗传算法 SAIGA。该算法比 GA、SA 和 RR 在任务平均完成时间上分别缩短了 11%，25%和 56%，在算法的收敛精度和速度上都有显著的提高，收敛速度上提高了 40 代左右，能够有效避免超常个体误导种群的进化方向和算法陷入局部最优，同时改进算法能够较均衡地分配任务到各个节点上，提高了资源的利用率。因此，改进算法更加适合云环境下的资源调度，具有较好的实用性。

参考文献：

[1] 刘 鹏. 云计算 [M]. 北京：电子工业出版社，2011.

[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] Caballer M，Blanquer I，Moltó G，et al. Dynamic Management of Virtual Infrastructures [J]. Journal of Grid Computing，2015，13 (1)：53－70.

[4] 林伟伟，齐德昱. 云计算资源调度研究综述 [J]. 计算机科学，2012，39 (10)：1－6.

[5] 刘 愉，赵志文，李小兰，等. 云计算环境中优化遗传算法的资源调度策略 [J]. 北京师范大学学报 (自然科学版)，2012，48 (4)：378－384.

[6] 袁 浩，李昌兵. 基于社会力群智能优化算法的云计算资源调度 [J]. 计算机科学，2015，42 (04)：206－208.

[7] 徐文忠，彭志平，左敬龙. 基于遗传算法的云计算资源调度策略研究 [J]. 计算机测量与控制，2015，23 (5)：1653－1656.

[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] 薛 玉. 云计算环境下的资源调度优化模型研究 [J]. 计算机仿真，2013，30 (5)：362－365.

[10] 邬海艳. 基于云计算环境下资源调度算法研究 [D]：赣州：江西理工大学，2012.

[11] Gao Y，Guan H，Qi Z，et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing [J]. Journal of Computer and System Sciences，2013，79 (8)：1230－1242.

[12] Zhan Z H，Zhang G Y，Ying L，et al. Load Balance Aware Genetic Algorithm for Task Scheduling in Cloud Computing [A] . In：Dick G，Browne W，Whigham P，Zhang M，Bui L，Ishibuchi H，Jin Y，Li X，Shi Y，Singh P，Tan K，Tang K，editors. Simulated Evolution and Learning [C] . Springer International Publishing，2014：644－655.

[13] Gall J，Rosenhahn B，Seidel H P. An Introduction to Interacting Simulated Annealing [A] . In：Rosenhahn B，Klette R，Metaxas D，editors. Human Motion [C] . Springer Netherlands，2008：319－345.

[14] S K，P VM. Optimization by simmulated annealing [J]. science，1983，220 (4598).

(上接第 201 页)

x = Math. cos(lat) Math. cos(lon)；

y = Math. cos(lat) Math. sin(lon)；

z = Math. sin(lat)；

那么当我们求夹角 AOB 时，只需要做一次点乘操作。比如求 (lon1，lat1) 和 (lon2，lat2) 的夹角，只需要计算 $x1x2 + y1y2 + z1 * z2$，这样避免了大量三角函数的计算。

在得到夹角之后，还需要执行 arccos 函数，将其转换成角度，AB 弧长＝角 $AOB * R$ (R 是地球半径)。上面的这种方法成为坐标转换方法。

坐标转换方法和基于三次多项式拟合三角函数方法性能都非常高，相比 lucene 使用的 Haversine 算法大大提高了计算效率，然而坐标转换方法存在一些缺点：

1) 坐标转换后的数据不能被直接用于空间索引。lucene 可以直接对经纬度进行 geohash 空间索引，而通过空间转换变成三维数据后不能直接使用。我们的应用有附近范围筛选功能 (例如附近 5 km 的团购单子)，通过 geohash 空间索引可以提高范围筛选的效率；

2) 坐标转换方法增大内存开销。我们会将坐标写入倒排索引中，之前坐标是 2 列 (经度和纬度)，现在变成 3 列 (x，y，z)，在使用中我们往往会将这数据放入到 cache 中，因此会增大内存开销；

3) 坐标转换方法增大建索引开销。此方法本质上是将计算从查询阶段放至到索引阶段，因此提高了建索引的开销。

所以，文章中所提出的基于三次多项式拟合三角函数方法计算地球空间距离，在同城范围内的优势非常明显，在效率和有效性方面想比较与以前的方法都有比较大的提升。这是今后研究的一个方向。

4 结束语

文中通过利用三次多项式拟合三角函数的方案改进了原有的地理空间距离计算的算法，从实验结果证明了，改进算法所具备了搜索目标点的快速性，同时也保证了相应的准确率。当然这个实验的结果是基于同城范围的基础上，完全符合 OTO 业务对于搜索场景的要求。同时方案中也存在有待改进的地方，在方案中是用三次多项式去拟合三角函数，根据数学知识可知，多项式的次数越高拟合的三角函数越精确，但是随之的计算的复杂度也在增加。所以如何在计算的复杂度和多项式拟合三角函数的精确度之间做出比较好的选择，可以进一步提高搜索的准确率，这个方向是今后研究的重点方向。

参考文献：

[1] Hua M C，Lou D C，Chang M C. Dual-Wrapped digital watermarking scheme for image copyright protection [J]. Computers & Security，2007，16 (1)：1－12.

[2] 司少林，关 永. 三角函数曲线拟合最佳次数的确定 [J]. 计算机工程与设计，2006. 24 (1)：30－35.

[3] Xiao Y Q，Ji Q. A robust content-based digital image watermarking scheme [J]. Signal Processing，2007，7：1264－1280.

[4] 陈 娱，徐 君. 考虑地理距离的复杂网络社区挖据算法 [J]. 地球信息科学，2013 (3)：58－65.

[5] 李耀彬，曾祥斌，沈钺武. 基于抛物线插值的正弦波拟合算法 [J]. 计算机工程与设计，2009 (11)：100－105.

[6] 薛国新，孙玉强. 正弦曲线三点拟合问题的一种新方法 [J]. 计算机仿真，2006 (2)：99－105.

[7] 罗成汉，刘小山. 曲线拟合法的 Matlab 实现 [J]. 现代电子技术，2003 (20)：54－59.

[8] 田 峥，徐 成，米 超，等. 基于消 失点和主方向估计的道路分割算法 [J]. 计算机研究与发展，2014 (4)：99－104.

[9] 谭方勇，于复生，吴建平. 基于消失点的坐标校准算法 [J]. 计算机应用，2011 (1)：101－105.

[10] 罗小松，房 斌，杨维斌. 采用韦伯局部特征的道路消失点检测 [J]. 计算机应用，2014 (S1)：219－222.