文章编号:1671-4598(2014)06-1960-03　　中图分类号:TP393　　文献标识码:A

基于改进量子粒子群的分布式并行计算框架设计

王卫锋，田　亮
(新乡学院 计算机与信息工程学院，河南 新乡　453003)

摘要：为了实现用户任务在大规模计算机集群上进行高效地处理，并克服现有并行计算框架通用性不强的缺点，提出了一种基于改进量子群算法和Map-Reduce模型的通用并行计算框架；首先，对经典的Map-Reduce分布式并行计算框架以及并行计算流程进行了具体描述；然后，基于改进的量子粒子群算法设计了改进的Map-Reduce模型，在Map阶段通过多种群并行搜索并计算所有粒子适应度，在Shuffle和Sort阶段实现粒子的排序和种群的重新划分，然后在Reduce阶段更新控制系数和粒子位置，当最优解不变时，通过混沌扰动对其进行扰动；仿真实验表明同，文中设计的基于改进量子粒子群算法和Map-Reduce模型能高效地执行任务，较传统的Map-Reduce模型具有较少的执行时间，具有很强的可行性，是一种有效的通用并行计算模型。

关键词：量子粒子群；任务；并行计算；混沌

Design of Distributed Parallel Computing Framework Based on Improved Quantum Particle Swarm

Wang Weifeng, Tian Liang
(Department of Computer and Information Engineering, Xinxiang University, Xinxiang　453003, China)

Abstract: In order to realize effective management of user tasks in the large computer group, and conquer the defects of the low universality of the given parallel computing framework, a parallel computing framework is propoesd based on improved Quantum particle swarm algorithm and Map-Reduce model. Firstly, the classic Map-Reduce model and the parallel computing flow were described. Then the improved Map-Reduce model was designed based on improved Quantum particle swarm algorithm, the multi-population was parallel searched and the fitness was computed, and the particle was sorted and the particle population was divided, then the control coefficient and particle position were renewed in the Reduce stage, when the global solution was unchanged, the particle was changed by chaos interrupt. The simulation experiment shows the method in this paper can execute task effectively, and compared with the traditional Map-Reduce model it has the less execution time. Therefore, the method in this paper has strong feasibility and universal parallel computing model.

Key words: quantum particle swarm; task; parallel computing; chaos

0　引言

随着通信技术和多核处理器的快速发展，计算模式经历了集中式的大型处理机模式、基于网络的P2P分布式处理模式和基于云计算的按需分布式并行处理（Parallel Computing）模式[1-2]，并行计算在时间上通过流水线技术，在空间上通过多处理器的并发执行，能最大程度地实现大规模运算和减少任务执行的实际时间[3]。

为了获得用户任务高效的处理性能，并行计算已经成为当前计算机领域的研究热点。文献[4]建立了一种基于GPU通用计算平台的中心差分格式的并行计算方法。文献[5]设计了基于Hadroop平台的K-Medoids并行聚类算法。文献[6]在对无网格局部MLPG方法进行改进的基础上，设计了基于MLPG的弹性动力学问题解决方法。文献[7]提出了一种基于带权图并行分解的层次化社区并行算法。文献[8]为了克服Petri网模型中变迁仅能加减运算的问题的不足，将计算任务分解成若干个子任务并行进行处理。

上述工作均研究了并行算法，但没有提出一个统一的并行框架。本文在上述工作的基础上设计了一种基于Map-Reduce模型和改进粒子群算法的并行计算框架，并通过实验证明了其有效性。

1　Map-Reduce模型

Map-Reduce模型是Google提出的分布式并行编程/任务调度模型[9-12]，用于处理大规模数据集，当在集群上运行Map-Reduce任务时，用户和程序员不再关注输入数据的分块、任务的分配和调度、集群资源节点失效和资源节点之间的通信问题，Map-Reduce模型通过采用任务内部松耦合，高并行的方式，在云计算、多核和多处理器以及异构机群上都表现出了优良的性能，其模型如图1所示。

从图1可以看出，Map-Reduce模型的执行过程可以描述为：用户作业的提交、Map任务分配和执行、Reduce任务的分配和执行及作业的完成。具体可以描述为：

(1) Input阶段：用户向Master控制器提交任务，Master根据用户指定的参数来分配Map任务，每个Map任务都必须指明输入和输出地址以及一些辅助参数，输入目录下的大数据块会被分割成小数据库快（splits），Map任务通过键值来读取所需的数据块。

(2) Map阶段：Map程序将数据分割为不相关的区块，根据用户定义的操作，生成新的中间的键值对<key, value>，

收稿日期:2013-12-27；　修回日期:2014-02-17。

作者简介:王卫锋(1978-),男,河南襄城人,硕士研究生,讲师,主要从事软件工程方向的研究。

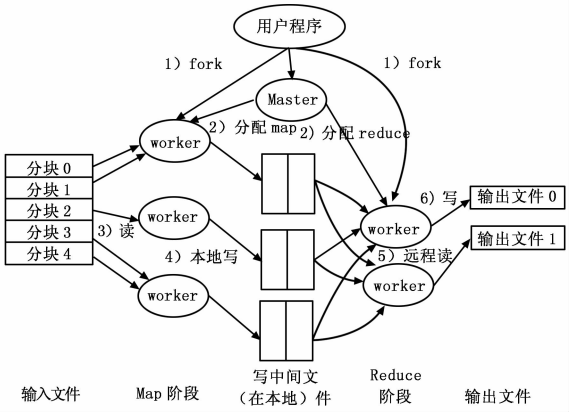

图 1　Map－Reduce 模型示意图

这组新的键值与用户输入的将任务分配给大量计算机处理达到分布式并行运算的效果。

（3）Shuffle 和 Sort 阶段：为了将具有相同键 Key 的中间结果存储在数据节点的同一个数据分区（Partition），Shuffle 阶段和 Sort 阶段为了对 Map 阶段的任务输出中间结果进行排序，需要通过 Shuffle 阶段完成混排交换数据，将具有相同键 Key 的中间结果尽量汇聚同一个节点上，同时在 Sort 阶段根据键 Key 对 Reduce 的输入数据进行分组排序，因此，Shuffle 阶段和 Sort 阶段可以同时进行。

（4）Reduce 阶段：Reduce 函数是根据一定的映射规则将输入的键值对＜key，value＞转换为另一个或一组键值对＜key，value＞，执行每个唯一的键 key 执行用户定义的 Reduce 函数，输入参数可以定义为 ＜key，（list of values）＞，输出结果为新的键值对＜key，value＞。

（5）Output 阶段：根据 Reduce 阶段的输出，将结果写入到输出目录中。

2　基于改进粒子群算法的并行计算框架

2.1　量子粒子群算法

粒子群优化算法（Particle swarm optimization algorism，PSO）[12] 是由 Kennedy 和 Eberhart 于 1995 年首次提出，目前已经在很多领域获得成功应用。在经典粒子群算法中，由于粒子速度有限，因此在搜索过程中只能覆盖一个有限区域而不能覆盖整个可行空间，不能保证以概率 1 收敛到全局最优。

量子粒子群算法（Quantum－behaved Particle Swarm optimization，QPSO）是从量子力学的角度提出的改进 PSO 算法，采用粒子表示问题的候选解，在量子空间中，粒子仅有位置向量没有速度向量，由 M 个粒子组成的粒子种群 $X=\{x_1,x_2,\ldots,x_M\}$，第 t 轮迭代时粒子 i 的当前位置可以表示为 $x_i(t)=\{x_{i1}(t),x_{i2}(t),\ldots,x_{in}(t)\}$，粒子 i 个体最优位置 $p_i(t)=\{p_{i1}(t),p_{i2}(t),\ldots,p_{in}(t)\}$，所有粒子的全局最优位置为 $p_g(t)=\{p_{g1}(t),p_{g2}(t),\ldots,p_{gn}(t)\}$，采用波函数 $\psi(x)$ 描述粒子位置，并通过求解一维 δ 势阱 Schrodinger 方程得到粒子概率分布函数可以表示为：

$$D(x)=e^{-2|p_r(t)-x|/L} \tag{1}$$

其中：$p_r(t)$ 为个体最优 $p_i(t)$ 和全局最优位置 $p_g(t)$ 之间的一个随机位置，是每个粒子收敛的局部吸引点，可以通过下式计算：

$$\begin{cases} p_{ri}(t)=\alpha p_i(t)+(1-\alpha)p_g \\ \alpha=c_1r_1/c_1r_1+c_2r_2 \end{cases}(t) \tag{2}$$

在式（2）中，r_1 和 r_2 为区间 0 到 1 之间的随机数，c_1 和 c_2 为学习因子，c_1 和 c_2 分别用于调节粒子向全局最优值飞行和粒子向个体最优值飞行，c_1 和 c_2 通取值为 2，L 可以通过迭代获得：

$$\begin{cases} L(t+1)=2\beta\mid p_a(t)-x(t)\mid \\ p_a(t)=\dfrac{\sum_{i=1}^{M}p_{li}(t)}{M} \end{cases} \tag{3}$$

在式（3）中，$p_a(t)$ 为第 t 次迭代粒子种群的位置均值，β 为常量惯性权重，采用 Monte Carlo 方法，获得第 t 次迭代时粒子位置为：

$$\begin{cases} x_i(t)=p_{ri}(t)\pm\dfrac{L}{2}\ln[\dfrac{1}{u}] \\ p_{ri}(t)=\alpha p_i(t)+(1-\alpha)p_g(t) \end{cases} \tag{4}$$

在式（4）中，u 为服从［0，1］上均匀分布的随机数，粒子在第 $t+1$ 次迭代时的位置可以表示为：

$$x_i(t+1)=p_{ri}(t)\pm\beta\mid p_a(t)-x_i(t)\mid\ln\frac{1}{u} \tag{5}$$

在式（5）中，β 是量子粒子群算法中的控制系数，也可以称为收缩—扩张系数，β 的选择和控制决定了算法的收敛程度。

2.2　改进的自适应控制系数

为了提高算法的全局收敛能力，使得算法能尽快获取全局最优解，使控制系数 β 的值随着迭代次数的变化进行自适应的改变，使其在算法初期具有较大的值，随着搜索的不断迭代 β 值的变化趋于平缓，以加快算法收敛速度和尽快获取全局最优解。

控制系数 β 随迭代次数变化的情况可以表示为下式所示：

$$\beta=\pm(0.5+\frac{0.5(t_{\max}-t)}{t_{\max}}) \tag{6}$$

在式（6）中，当 $u\leqslant 0.5$ 时，β 取负值，否则取正值，$t_{\max}$ 为迭代次数最大值。

2.3　Map 阶段模型

种群中的所有粒子在初始化后，将种群分为若干子种群，每个子种群分别通过一个 Map 函数进行并行搜索，每次迭代过程均可以通过 Map－Reduce 模型进行并行操作完成。

在 Map 阶段，将键值对＜$key1$，$value1$＞作为输入，输出为＜$key\ 2$，$value\ 2$＞，Map 中必须包含下列两个函数：

$$key2=RepresentIndividual(key1) \tag{7}$$

$$value2=(indi(key1),pbst(key1),gbst(key1)) \tag{8}$$

其中：$RepresentIndividual$ 对应了问题的编码方式，$indi(key1)$ 为粒子的适应度值，$pbst(key1)$ 为粒子的个体最优位置，$gbst(key1)$ 为粒子的全局最优位置。

因此，在 Map 阶段主要负责粒子个体的适应度评估，并获得适应度最高的最优个体，并将其存储到分布式文件系统（DFS）中，其操作流程如下所示：

（1）map（$key1$，$value1$）；

（2）通过公式（7）和公式（8）获取粒子位置表示 $key2$ 和计算适应度值 $indi(key1)$；

（3）判断 $indi(key1)$ 是否大于粒子的个体最优适应度值 $keypbest$：如果大于，则采用 $indi(key1)$ 更新个体最优适应度

值 $indi(key1)$ 值，并对 $value2$ 中的 $pbst(key1)$ 进行更新；

(4) 判断 $indi(key1)$ 是否大于所有粒子的最优适应度值 $keybest$；如果大于，则采用 $indi(key1)$ 更新所有个体的全局最优适应度值 $keybest$ 值，并采用 $indi(key1)$ 对当前种群中所粒子 $value2$ 中的 $gbst(key1)$ 进行更新；

2.4 Shuffle 和 Sort 阶段模型

在 Map 阶段得到所有子种群中的粒子位置、粒子个体最优位置和全局最优位置后，在 Shuffle 和 Sort 阶段将 Map 阶段输出的所有中间结果，即粒子进行重新组合，然后通过某种方式进行排序输出将结果提供给 Reduce 函数作为输入数据：

(1) 将粒子适应度为所有粒子中前 1/4 的个体替代位于后 1/4 的粒子个体；

(2) 通过哈希函数对粒子的键值对进行散列：

$$Hash(key)\%number\_reduces \tag{9}$$

在式 (9) 中，$number\_reduces$ 为 Reduce 的个数；

(3) 根据哈希函数的散列值，将每个个体划分到不同的类中，相同的类对应了同一个 Reducer。

2.5 Reduce 阶段模型

Reduce 阶段首先获取由 Shuffle 和 Sort 阶段的输出作为其输入，然后对控制系数进行调整，并粒子位置进行更新，同时加入混沌扰动，以加速粒子获取全局最优解，可以描述为：

(1) Reducer 接受 Shuffle 和 Sort 阶段的输出，每个 Reduce 函数接收一个新的子种群作为输入；

(2) 根据当前迭代次数对控制系数进行调整：

$$\beta = adjust(t) \tag{10}$$

(3) 通过式 (3) 计算当前种群的各粒子的平均适应度；

(4) 对当前种群中的所有粒子的位置根据式 (5) 进行更新；

(5) 当连续几次迭代过程粒子的最优位置 $x_{ik}(e_k \leqslant x_{ik} \leqslant f_k)$ 不发生变化，对粒子进行混沌扰动，根据公式 (11) 将 x_i 映射到区间 (0，1) 上得到 y_{ik}，如下所示：

$$y_{ik} = \frac{x_{ik} - e_k}{f_k - e_k} \tag{11}$$

(6) 采用公式 (12) 对映射值 y_{ik} 进行混沌扰动，得到 z_{ik}：

$$z_{ik} = \begin{cases} 2y_{ik} & 0 \leqslant y_{ik} \leqslant 1/2 \\ 2(1-y_{ik}) & 1/2 \leqslant y_{ik} \leqslant 1 \end{cases} \tag{12}$$

(7) 根据公式 (13) 将 z_{ik} 重新映射到原空间，如下所示：

$$x_{ik} = a_k + z_{ik}(b_k - a_k) \tag{13}$$

(8) 选取所有种群中粒子的最优适应度的粒子作为问题的最优解。

3 仿真实验

实验硬件服务器平台和软件平台分别为 Dell PowerEdge R910 和 Java 和 Hadoop，服务器中包含 24 核 Intel Xeon 处理器，内存 80 G，随机输入一组任务，任务个数从 20 增长到 1 000 个，粒子个体从 1 000 增加到 20 000，服务器的数量随着粒子个数的从 2 增加到 10，为了对比文中并行框架的优点，将文中方法与串行程序的运行时间和 Map－Reduce 时间进行比较，如表 1 所示：

从表 1 中可以看出，文中方法得到的任务并行运行时间，多源输出从最初的 12 个发展到执行结束时的 92 个，具有很好的并行能力。虽然在仿真初期粒子数较少时，文中并行方法任务执行时间大于串行执行所需时间，但当粒子数大于 9 000 后，任务串行执行时间增加迅速，当任务数为 1 600 个而粒子数为 18 000 时，基于经典 Map－Reduce 的串行执行方法由于内存无法满足任务执行的要求而发生溢出，而文中并行算法增幅较缓慢，在所有任务执行完毕时花费的总时间也仅为 585 s，同时文中的方法并行集群数目一直较传统的 Map－Reduce 模型少而执行效果却更优，这是因为文中方法对传统的 Map－Reduce 方法进行了优化，将粒子群的并行化求解问题加入到了 Map－Reduce 模型中，因此，文中方法在执行任务具有很好的效果。

表 1 串行和并行运行时间比较

粒子个体数	串行时间 (s)	并行时间 (s)	Map 时间 (s)	Reduce 时间 (s)	Map－Reduce 集群并行 (个)	文中新集群并行 (个)	并行多源输出
1 000	0.02	13	5	7	16	14	12
2 000	0.45	24	10	12	23	21	12
3 000	0.78	30	11	14	24	22	13
4 000	1.14	31	10	14	23	21	14
5 000	4.54	34	13	15	28	25	14
6 000	10.36	35	14	13	30	28	15
7 000	18.57	36	14	15	34	29	15
8 000	27.85	37	15	17	38	35	18
9 000	38.54	38	17	20	47	21	19
10 000	54.67	39	18	19	63	22	26
11 000	73.78	47	20	21	70	21	26
12 000	95.57	50	21	22	102	25	30
13 000	267.78	58	23	24	104	28	31
14 000	489.97	102	25	28	114	29	36
15 000	868.46	150	30	36	127	68	37
16 000	1 075.67	210	34	38	150	83	42
17 000	超出内存	305	50	55		118	57
18 000		407	49	58		152	69
19 000		504	85	95		163	79
20 000		585	89	92		210	92

4 结论

为了实现一种高性能的通用并行计算，在经典 Map－Reduce 模型的基础上，设计了一种基于改进量子群算法的并行计算框架。首先描述了经典的 Map－Reduce 模型，然后基于改进的量子粒子群算法，对 Map－Reduce 模型的各阶段即 Map 阶段、Shuffle 和 Sort 阶段和 Reduce 阶段都重新进行了设计，使得现有粒子群算法可以充分利用 Map－Reduce 模型进行问题求解。仿真实验证明文中设计的并行框架在执行任务时更为高效，是一种可行通用的模型，具有很强的实用性。

参考文献：

[1] 陈国良，孙广中，徐　云，等. 并行计算的一体化研究现状与发展趋势 [J]. 中国科学，2009，54 (8)：1043－1049.

(下转第 1966 页)

为考核移动节点间的通信情况，节点发送的数据帧中设置一个16位字段用于对数据包计数，每产生一个数据包计数加1并写入该字段。汇聚节点收到数据包后提取该值，用于统计是否有数据包的丢失。当机器人在区域中移动时，汇聚节点能实时接收到移动节点发送的位置，其中当机器人移动到图中各参考点时，定位结果和实际位置坐标如表1所示。

使机器人按照一定的规则移动，搜集汇聚节点接收到的坐标信息进行作图，得到的机器人实际运动轨迹如图6所示。

表1 定位结果对比

实际坐标	定位结果	误差(m)
(10,10)	(10.17,10.08)	0.19
(10,30)	(9.91.30.22)	0.23
(10,50)	(9.84,49.22)	0.18
(30,10)	(30.22,10.08)	0.23
(30,30)	(29.86,30.10)	0.17
(30,50)	(29.79,50.12)	0.24
(50,10)	(50.09,9.80)	0.21
(50,30)	(49.79,29.86)	0.25
(50,50)	(49.89,49.80)	0.23

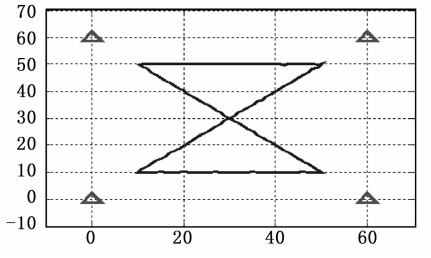

图6 机器人实际运动轨迹

实验表明使用NanoLOC节点模拟基站及移动节点，可以对60 m×60 m的区域内处于静止和移动状态的移动节点进行良好地通信以及较高精度地定位，且对机器人的相对定位误差均小于0.25 m；通信方面移动节点周期性快速发送，汇聚节点接收到的数据包计数连续无中断，即说明无丢包现象。

6 结束语

针对危险的特殊作业环境，为了能够准确获取机器人在空间的位置信息，并顺利执行任务，设计了一套基于无线传感器网络的高精度定位系统，借助具有宽带线性调频扩频技术的NanoPAN5375芯片，并加入了极大似然估计法定位算法，大大提高了定位精度。通过在60 m×60 m区域内的实验结果表明，系统工作稳定可靠，在可将基站架设在一定高度的视距条件下，该系统在通信和定位方面均能有良好的表现，测量的相对定位误差均小于0.25 m，为在矿井搜救、核泄漏检测和火山探索等特殊环境下工作的机器人提供了实时高精度的定位数据。

参考文献：

[1] 牛国臣，徐 萍，冯 琦. 基于里程计和PTZ视觉的移动机器人自定位 [J]. 计算机应用，2011，31 (10)：2821－2824.
[2] 罗小敏，蔡昌盛. GPS/GALILEO组合单点定位精度分析 [J]. 大地测量与地球动力学，2013，33 (3)：136－140.
[3] 陈三风，陈万明. 基于RSSI误差分析的无线传感器网络定位研究 [J]. 计算机工程与应用，2011，47 (14)：10－14，75.
[4] 孙红新，叶小岭，胡 凯. 基于超声波的移动机器人的同时定位和地图构建 [J]. 计算机测量与控制，2011，19 (11)：2769－2771.
[5] 段翠翠，王瑞荣，王建中，等. 基于无线传感器网络的高危生产区人员定位系统 [J]. 传感技术学报，2012，25 (11)：1599－1602.
[6] 余义斌，曹长修，李昌兵. 基于权重重心法的传感器网络节点定位 [J]. 计算机仿真，2007，24 (6)：296－330.
[7] 葛日波，沈海龙. 基于UWB编码的井下无线传感器网络定位方法 [J]. 计算机应用与软件，2013，30 (12)：284－287.
[8] 张广峰，段其昌，刘 政. 基于加强学习与联想记忆粒子群优化算法的节点定位 [J]. 传感器与微系统，2013，32 (3)：72－73.
[9] 石琴琴，霍宏，方 涛，等. 使用最速下降法提高极大似然估计算法的节点定位精度 [J]. 计算机应用研究，2008，25 (7)：2038－2040.
[10] 刘瀚文，姜春兰，李 明，等. 网络化弹药的临时集中控制轮询时分多址协议 [J]. 探测与控制学报，2013，35 (2)：50－54.

(上接第1962页)

[2] Kc K，Anyanwu K. Scheduling Hadoop Jobs to meet deadlines [A]. Cloudcon' 10 Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science [C]. 2010：388－392.
[3] 李 莉，廖剑伟，欧 灵. 云计算初探 [J]. 计算机应用研究，2010，27 (12)：4419－4422.
[4] Joldes G R，Wittek A，Miller K. Real－time nonlinear finite element computations on GPU－application to neurosurgical simulation [J]. Computer Methods in Applied Mechanics and Engineering，2010，199：3305－3314.
[5] 蔡 勇，李光耀，王 琥. GPU通用计算平台上中心差分格式显式有限元并行计算 [J]. 计算机研究与发展，2013，50 (2)：412－419.
[6] 张雪萍，龚康莉，赵广才. 基于MapReduce的K－Medoids并行算法 [J]. 计算机应用，2013，33 (4)：1023－1025.
[7] 曾清红. 无网格局部Petrov－Galerkin方法的并行计算研究 [J]. 计算力学学报，2012，29 (2)：205－209.
[8] 林旺群，卢风顺，丁兆云，等. 基于带权图的层次化社区并行计算方法 [J]. 软件学报，2012，23 (6)：1517－1530.
[9] 万 军，赵不贿. 并行计算的Petri网建模和FPGA实现 [J]. 计算机应用研究，2013，9 (30)：2660－2663.
[10] Bryant R E. Data intensive super computing：The case for DISC，CMU－CS－07－128 [R]. Pittsburgh，PA，USA：Carnegie Mellon University，Department of Computer Science，2007.
[11] Polo J，Carrera D，Becerra Y，et al. Performance－driven task co－scheduling for map－reduce environment [A]. 12 th IEEE/IFIP Network Operation and Management Symposium [C]. 2010：373－380.
[12] 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.