基数估计算法参数的分析与优化

刘绍记[1] 曹 阳[1] 崔梦天[2]
(华南师范大学计算机学院 广州 510000)[1] (西南民族大学计算机科学与技术学院 成都 610041)[2]

摘 要 基数估计算法(Cardinality Estimation Algorithm)是基于概率统计理论的估算给定数据集中不重复元素基数的算法。算法中的Hash函数和相关参数的设置是影响算法性能的两个关键因素。针对这两个问题展开研究,提出了一种基数估计的优化算法,它可以根据数据规模和数据类型动态调整Hash函数和分桶参数,以提高算法的精度和稳定性。实验结果表明,改进的基数估计算法在经过训练之后,相比传统估计算法,其估计精度和稳定性均有所提高。

关键词 基数估计,Hash函数,训练
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.02.047

Parameter Analysis and Optimization of Cardinality Estimation Algorithm

LIU Shao-ji[1] CAO Yang[1] CUI Meng-tian[2]
(School of Computer, South China Normal University, Guangzhou 510000, China)[1]
(School of Computer Science and Technology, Southwest University for Nationalities, Chengdu 610041, China)[2]

Abstract Cardinality estimation algorithm is an algorithm based on statistics, which can estimate the cardinality of the given data set. In this algorithm, hash function and some parameters are the key factors to the performance of the algorithm. Based on the related research, an algorithm was proposed which selects hash function and parameters based on the scale and type of data. The experimental results show that both accuracy and stabilization of this algorithm have significant improvement than the traditional estimation algorithm.

Keywords Cardinality estimation, Hash function, Training

1 引言

基数估计算法是基于概率统计理论的用于估算给定多重集"势"的算法[1-2]。目前,大规模数据的基数的实时计算问题在电商、搜索、P2P、数据库等方面应用广泛,相关高效算法已成为研究热点。

现阶段对基数估计算法的研究主要基于Philippe Flajolet于1985年提出的PCSA算法[3];在此基础上,他于2003年引入分桶数的概念,提出了Super-LogLog算法[4],该算法在提高算法估计精度的同时降低了时空消耗;2007年,在Super-LogLog算法的基础上,他用调和平均取代算数平均,并根据数据规模对估计值进行修正,提出了在可处理数据量、估计精度、时空复杂性等方面都有极大提升的Hyper-LogLog算法[5]。基于上述算法的理论研究,研究人员结合自身的研究方向提出了很多适用不同场景的类基数估计算法。Kyu-Young Whang[6]提出了一种针对数据库的线性时间复杂度的估计算法。Min Cai等[7]于2005年提出了性能稳定的Adaptive Counting算法。Chen Qian等[8]将基数估计算法应用于RFID(Radio Frequency Identification)领域,使货物的清点变得简单高效。Ning Zhang[9]提出的关于数据库查找算法也借鉴了基数估计算法的思想,提高了查找速度。高文文[10]研究主机分布问题时,提出采用在线+离线模式解决DDoS寻找主机基数问题。相关算法也在AddThis、Google、淘宝等公司有了成熟的商业应用。

目前,关于基数估计算法的研究焦点主要集中在估计方法的改进和算法的实际应用两个方面,而对算法参数的选择与优化以及算法中Hash函数的选取的研究较少。基数估计算法中Hash函数是获取均匀分布的前提,均匀分布又直接影响算法的精度和稳定性;同时,估计参数的微小改变将会对算法的精度产生巨大的影响。Stefan Heule曾在文献[1]中改进了Hash函数,并针对10万以下规模的数据进行了相关研究,但没有给出Hash函数选择和算法参数设定的策略。针对以上问题,本文重点分析了Hash函数和分桶参数对算法精度和时空性能的影响,提出了基于Hyper-LogLog算法的参数选择优化算法,并通过实验验证了优化算法的有效性和稳定性。

本文第2节介绍了3种基数估计的经典算法以及其各自的优缺点,最终基于Hyper-LogLog算法提出了Hash函数和

到稿日期:2015-12-11 返修日期:2016-05-08 本文受2015年度广东省高等教育教学改革项目:基于敏捷开发的软件项目实践课程迭代式教学模式设计与实践,广东省水利科技创新项目成果(2014-16),国家自然科学基金项目(61379019),四川省科技计划项目(2015JY002)资助。
刘绍记(1991-),男,硕士,主要研究方向为空间信息技术与应用,E-mail:ether1991@hotmail.com;曹 阳(1975-),女,博士,副教授,主要研究方向为空间信息理论与技术,E-mail:503581960@qq.com(通信作者);崔梦天(1972-),女,博士后,教授,主要研究方向为可信软件和优化算法,E-mail:happyzg3@163.com。

分桶参数的选择优化算法；第 3 节通过实验分析，对本文提出的优化算法和原始算法进行对比测试，验证了优化算法的有效性和稳定性；最后对优化算法和实验进行了总结。

2 基数估计算法

基数估计算法是建立在概率统计理论基础之上的算法。通过 Hash 函数预处理获得给定数据的均匀分布是该类算法的通用步骤，而对于获取统计量、使用的估计公式以及估计值的修正策略各算法则有所不同。本节介绍了 3 种基数估计的经典算法并针对其各自的特点进行了分析，最后提出了基于 Hyper-LogLog 算法的 Hash 函数和分桶参数选取的优化算法。

2.1 算法思想

基数估计算法的思想是：给定待估计的多重集 $Multi=\{x_i|x_i\in D\}$，其中 D 为待估计数据类型所属的数据域，例如所有字符串的集合或整数域 Z 等。首先，将算法预设的 Hash 函数 $H:D\rightarrow D'$ 作用于 $Multi$ 中的每个元素，从而获得 $\mathfrak{M}$ 对应的均匀分布 $S=\{h_j|h_j\in D'\}$，$h_j=H(x_i)$。设 M 为二维数组，用于统计估计公式所需的参数，通过遍历均匀分布 S，不同基数估计算法按照各自的规则完成对数组 M 的初始化。然后从 M 中获得估计公式所需的相关参数，并计算初步估计值 E。再经过算法各自的修正后，获得 E^* 作为多重集 $Multi$ 的基数的估计值。

2.2 基数估计算法

基数估计算法的目标是针对待估计的多重集 $\mathfrak{M}$，在时空消耗低、误差可控的情况下，估算给定集合中不重复元素的个数。经典的基数估计算法有 PCSA 算法、Super-LogLog 算法和 Hyper-LogLog 算法。3 种算法都是基于概率统计理论中随机数均匀分布的特点而设计的。首先将原始数据用 Hash 函数映射成服从均匀分布的(伪)随机数集合，再对这些数据集合按具体算法步骤进行处理，从而估算其基数。

PCSA 算法[3]的主要思想是：首先采用基数估计通用方法(取原始数据的 Hash 值)获得给定多重集 $Multi$ 对应的均匀分布 S。针对 S 中任一元素 h_j，利用算法中预设的 $nmap$ 值对其分组，即令 $\alpha=h_j \bmod nmap, index=\rho(h_j/nmap)$。对任意二进制数 ω，定义 $\rho(\omega)$ 为 ω 的二进制表示从左到右连续出现的 1 的个数。定义二维数组 $M[nmap, length]$，文献[3]给出了 $nmap$ 和 $length$ 的具体值以及数组 M 初始化的操作步骤。遍历完原始数据对应的均匀分布集合 S 后，即可获得数组 M，然后统计二维数组 M 的每行中连续出现 1 的个数 R_i，令 $R=\sum_{i=0}^{nmap-1}R_i$。最后得基数的估计值为：$E^*=namp/\varphi*2^{(R/namp)}$，其中 $\varphi=0.77351$ 为该算法中提出的修正值。

Super-LogLog 算法[4]是针对 PCSA 算法存储空间消耗大、不能适应更大规模数据等问题而设计的。算法的主要思想是：利用通用方法获得待估计多重集 $Multi$ 对应的均匀分布 S。对 S 中的任一元素的二进制形式 $h_j=\langle b_1,b_2,\cdots,b_k,b_{k+1}\cdots\rangle_2$，取其前 k 位作为该元素的分桶编号 $M(h_i)=\langle b_1,b_2,\cdots,b_k\rangle_2$，剩余部分记作 $\omega=\langle b_{k+1},b_{k+2},\cdots\rangle_2$。一维数组 M 为分桶数组，用于记录估计公式需要的参数，易知 $M=M[1],\cdots,M[i],\cdots,M[m]$，其中 $m=2^k$。然后计算 $\rho(\omega)$，更新分桶数组 $M[h_i]=\max(M[h_i],\rho(\omega))$，通过遍历 S 完成对 M 的初始化。最后由算法的估计公式 $E^*=\alpha_m m 2^{\frac{1}{m}\sum_j^{M[j]}}$ 获得估计结果，其中修正值 α_m 和分桶参数 k 的取值由文献[3]经理论分析后给出。

Hyper-LogLog 算法[5]针对 Super-LogLog 算法对离群值敏感、数据规模变化适应性差等问题进行改进。该算法较 Super-LogLog 算法而言主体不变，而估计公式中采用调和平均代替几何平均，算法估计公式为：$E:=\alpha_m m^2(\sum_{j=1}^{m}2^{-M[i]})^{-1}$，文献[5]给出了 k,α_m 具体的取值策略；同时提出了对估计值进行二次修正的思想和具体操作步骤，从而使 Hyper-LogLog 算法的整体性能更优。算法 1 展示了 Hyper-LogLog 算法的完整过程。

算法 1 Hyper-LogLog 算法

输入：待估计多重集 Multi$=\{x_i|x_i\in D\}$

输出：待估计多重集 $\mathfrak{M}$ 基数的估计值 E^*

步骤 1 初始化相关参数

(1)分桶数 $m=2^k, k\in[4,16]$；初始化分桶数组 $M=M[1],\cdots,M[i],\cdots,M[m]=\vec{0}$。

(2)根据文献[4]中相关的理论推导，初始化修正值 α_m 如下：$\alpha_{16}=0.673$；$\alpha_{32}=0.697$；$\alpha_{64}=0.709$；$\alpha_m=0.7213/(1+1.079)$。

(3)设定 Hash 函数 H。

步骤 2 遍历给定多重集 Multi$=\{x_i|x_i\in D\}$，通过 Hash 映射 $H:D\rightarrow\{0,1\}^\infty$ 获得其对应的均匀分布集合 $S=\{h_j|h_j\in\{0,1\}^\infty\}$，其中 $h_j=\langle b_1,b_2,\cdots\rangle_2$。取 h_j 前 k 位作为该数据所属的分桶号 $M(h_i)=\langle b_1,b_2,\cdots,b_k\rangle_2$，剩余部分记作 $\omega=\langle b_{k+1},b_{k+2},\cdots\rangle_2$。

步骤 3 计算 $\omega=\langle b_{k+1},b_{k+2},\cdots\rangle_2$ 从左到右首次出现 1 的位置，记为 $\rho(\omega)$，并将其作为 M 数组的下标，更新其对应的分桶的值为：$M[j]=\max(M[j],\rho(\omega))$。

步骤 4 根据获得的分桶数组 M 计算：$E:=\alpha_m m^2(\sum_{j=1}^{m}2^{-M[i]})^{-1}$，作为待估计多重集 $\mathfrak{M}$ 基数的初步估计值。

步骤 5 根据分桶数 m 的大小对 E 进行二次修正，获得最终估计值 E^*：

```
if(V≠0)
    E* =mlog(m/V)
else
    E* =E;
if(E≤5/2 m)
    E* =0
else if(E≤1/30 2^32)
    E* =E;
else if(E>1/30 2^32)
    E* =-2^32 log(1-E/2^32)
```

步骤 6 返回 E^*。

对于基数估计算法而言，Hyper-LogLog 算法在时空复杂性、估计精度等方面都是最优的。然而，关于算法中 Hash 函数和分桶参数这两个影响 Hyper-LogLog 算法精度的重要因素的研究还很匮乏，更没有相关的选取策略。本文主要针对该问题进行了研究，并结合研究成果提出了基于 Hyper-LogLog 算法的相关的 Hash 函数和分桶参数的优化算法。

2.3 Hash 函数和分桶参数选择的优化算法

Donald Ervin Knuth[11]已经证明：理论上不存在一个将

一组非随机数映射为一组随机数的 Hash 函数，但是在实际中，构造一个近乎完美的 Hash 函数却不难做到。基数估计算法优化的思路是针对不同的数据类型，选取不同的 Hash 函数来实现优化。参照 FNVHash 和相关 Hash 函数关于碰撞率和均匀性等方面性能的比较结果[12]，本文提出的优化算法根据特别针对字符串类型数据预设了性能最优的 FNVHash 函数，如果出现非字符串数据，可将其先转化成字符串数据，然后输入算法。FNVHash 的全名为 Fowler-Noll-Vo 算法，是以 3 位发明人 Glenn Fowler，Landon Curt Noll，Phong Vo 的名字来命名的，最早于 1991 年提出。FNV 能快速 Hash 大量数据并保持较小的冲突率，它的高度分散使它适用于 Hash 一些非常相近的字符串，比如 URL，Hostname，文件名，Text，IP 地址等[13]。

另一个影响算法估计精度的参数是分桶数 k，从 2.2 节对 3 种算法的介绍中可以看出，分桶数的选取将会对算法的结果产生极大影响。本文 3.2 节针对同一数据集，对 k 的取值与估计结果进行实验分析，验证了上述结论。针对这一问题，本文弃用经验值 k，通过对训练集的学习使算法自动获得分桶数的最优值。

算法 2 基于 2.2 节 Hyper-LogLog 的算法思路，结合上述对 Hash 函数性能、分桶数 k 的选取对算法结果影响的分析，设计了针对 Hash 函数和分桶参数选取优化的 Hyper-LogLog 算法。

算法 2　Hash 函数和分桶参数选择的优化算法

输入：

(1) n 个训练集 $Multi_{T1}, Multi_{T2}, \cdots, Multi_{Tn}$，待估计多重集 $Multi=\{x_1, x_2, \cdots, x_n\}$。

(2) 给定数据类型 K=string，number，…。

(3) 数据规模 Scale(T)。

(4) 可接受的最大误差 δ，训练集 $T_1, T_2, \cdots, T_n$ 对应的基数的真实值 $actural_1, actural_2, \cdots, actural_n$。

输出：待估计多重集 $\mathfrak{M}$ 的估计值 E^*。

步骤 1　初始化相关参数：

(1) 分桶数为 $m=2^k$，预设 $k\in[12,\cdots,22]$ 为选作分桶标志的 bit 位数，k 每次调整的步长 $\tau=1$。数组 Best_b(scale) 存放的是针对给定规模最佳 k 的值。经过训练后，k 的初始值根据 Scale(T) 选择。分桶数组 M[1]，…，M[i]，…，M[m] 初始化为 0。

(2) 根据文献[4]中相关的理论推导，初始化修正值 α_m 如下：$\alpha_{16}=0.673$；$\alpha_{32}=0.697$；$\alpha_{64}=0.709$；$\alpha_m=0.7213/(1+1.079)$。

(3) Hash 函数根据待估计多重集的数据类型 K=string，number，…，针对字符串类型，选用 FNVHash。对于数据为非字符串类型，则将原始数据字符串化，再输入到算法中进行运算。

步骤 2　遍历给定多重集 $Multi=\{x_i \mid x_i\in D\}$，通过 Hash 映射 $H: D\to\{0,1\}^\infty$ 获得对应的均匀分布的集合 $S=\{h_j \mid h_j\in\{0,1\}^\infty\}$，其中 $h_j=\langle b_1, b_2, \cdots\rangle_2$。取 h_j 前 k 位作为该数据所属的分桶号 $M(h_i)=\langle b_1, b_2, \cdots, b_k\rangle_2$，剩余部分记作 $\omega=\langle b_{k+1}, b_{k+2}, \cdots\rangle_2$。

步骤 3　计算 $\omega=\langle b_{k+1}, b_{k+2}, \cdots\rangle_2$ 从左到右首次出现 1 的位置计为 $\rho(\omega)$，将其作为 M 数组的下标，更新其对应的分桶的值为：$M[j]=\max(M[j], \rho(\omega))$。

步骤 4　根据获得的分桶数组 M 计算：$E:=\alpha_m m^2\left(\sum_{j=1}^{m} 2^{-M[j]}\right)^{-1}$，作为待估计多重集的初步估计值。

步骤 5　根据分桶数 m 的大小对 E 进行修正，获得待估计集合的最终估计值 E^*：

```
if(V≠0)
    E* =mlog(m/V)
else
    E* =E;
if(E≤(5/2)m)
    E* =0
else if(E≤(1/30)2^32)
    E* =E;
else if(E>(1/30)2^32)
    E* =-2^32 log(1-E/2^32)
```

步骤 6　根据给定误差上限 δ 和确定算法计算的该集合基数的实际值 actural 来判断估计结果的误差是否符合要求。

```
if(E*/actural<δ)
    return E*
else
    跳到步骤 7，调整参数
```

步骤 7　根据前后两次估计结果的误差 ϵ_i 和 ϵ_{i+1} 的关系，调整 b，以减小误差。

```
if(ε_i<ε_{i+1})
    b=b-τ
else
    b=b+τ
if(b 已经被用过了)
    return E*
```

3　实验结果及分析

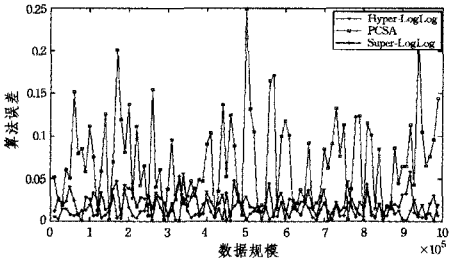
本节首先针对基数估计算法在数据规模、算法参数方面进行了实验，同时测试了采用同种算法使用不同 Hash 函数后算法的性能；然后重点针对本文提出的优化算法进行实验分析，验证了优化算法的有效性和稳定性。实验中用到的数据均为随机生成的长度为 16 个字节的字符串数据，数据规模可根据需求设定。

3.1　基数估计算法的性能分析

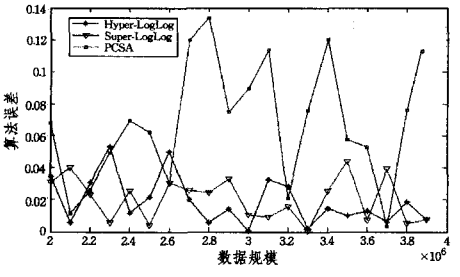
针对数据规模对算法精度的影响，本节测试了 PCSA，Super-LogLog，Hyper-LogLog 这 3 种算法关于不同数据规模的表现，并最终选择性能最优的 Hyper-LogLog 算法作为实验和改进的对象。实验结果中误差有正有负，为便于观察规律，所有结果图中的误差都取原始误差的绝对值。

图 1(a) 和图 1(b) 分别是针对 0～100 万和 200～400 万的数据规模，$k=12$，Hash 函数取 FNVHash 时 3 种算法的估计误差曲线。

相比较而言，PCSA 算法误差在 8% 左右且稳定性差；Super-LogLog 算法误差在 2% 左右且有个别情况下能达到 5%。由图 1(b) 可知，在数据规模增大的过程中，Hyper-LogLog 算法误差曲线的波动明显小于 Super-LogLog 算法的误差曲线，说明 Hyper-LogLog 算法相比 Super-LogLog 算法关于离群值的影响已经有了明显的改进。因此以下相关实验将选用综合性能最优的 Hyper-LogLog 算法进行。



(a)3 种算法针对不同数据规模的误差



(b)3 种算法针对不同数据规模的误差

图 1 3 种估计算法的误差

3.2 Hash 函数和分桶数的选取

为探究 Hash 函数的选取对基数估计算法估计精度的影响,实验中针对不同数据规模、Hyper-LogLog 算法分别采用 FNVHash 和 MurMurHash32[14]两种 Hash 函数的情况进行了测试,实验结果如图 2 所示。

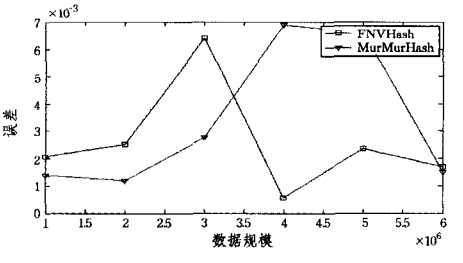


图 2 不同 Hash 函数 Hyper-LogLog 算法误差的对比

通过观察上述实验结果,可以很清楚地看到Hyper-LogLog算法在选用 FNVHash 作为 Hash 函数时,算法的误差大小和误差的波动情况都要明显优于选用 MurMurHash 作算法的 Hash 函数。

为探究分桶数 $m=2^k$ 时对基数估计算法估计精度的影响,实验分别选取 100 万、250 万、500 万的数据规模,通过改变 k 值来测试观察两者的关系,实验结果如图 3 所示。

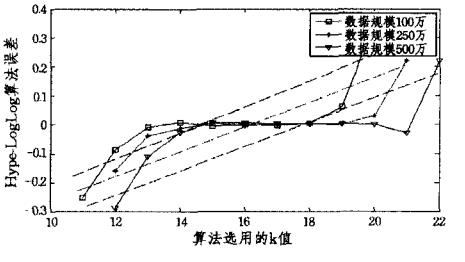


图 3 不同数据规模下 k 值和误差的关系

实验发现,百万级数据规模下,最优的 k 值集中在 15～18 之间。针对十万级(40 万以上)的数据规模,推荐使用12～16 之间的整数。

3.3 算法优化后的性能对比

实验采用随机生成指定规模、长度为 16 的字符串数据作为训练集和测试集。对于字符串数据,用 10 组不同规模的数据进行训练,使优化后的算法自主学习 k 值的选取策略。针对相同测试数据,挑选原始 Hyper-LogLog 算法中 k 取值过大、过小两种情况进行分析。表 1 和图 4 是改进算法和原始 Hyper-LogLog 算法误差的实验结果。

表 1 改进算法和原始算法误差的对比

	原始_k 小	改进算法	原始_k 大
方差	0.000262	3.21E-05	0.964624
误差总和	0.173938	0.038766	4.474377

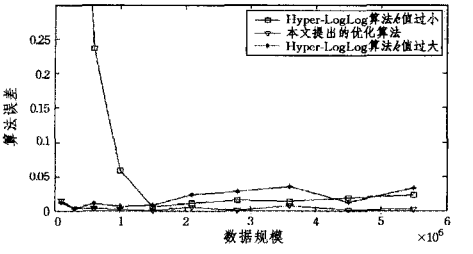


图 4 改进算法和原始算法误差的对比

所提优化算法相比于原始算法,两者耗时一致,但是所提算法经过训练后在精度上平均提升了 50%以上,算法的稳定性也更强,误差波动更小。

结束语 基于概率的基数估计算法为大规模数据实时求算基数问题提供了新的解决思路。相比于确定算法,基数估计算法可以减小 90%左右的内存消耗,时间节约 70%以上;相比于未优化的 Hyper-LogLog 算法,本文提出的优化算法经过训练后可以将其误差进一步减小 50%以上,而时空消耗不会明显增加。

参 考 文 献

[1] HEULE S, NUNKESSER M, HALL A. HyperLogLog in Practice: Algorithmic Engineering of a State of The Art Cardinality Estimation Algorithm [C]// Proceedings of the 16th International Conference on Extending Database Technology (EDBT'13). 2013: 683-692.

[2] 张洋. 解读 Cardinality Estimation 算法[EB/OL]. [2014-12]. http://blog.codinglabs.org/articles/algorithms-for-cardinality-estimation-parti.html.

[3] FLAJOLET P, MARTIN G N. Probabilistic Counting Algorithms for Data Base[J]. Journal of Computer and System Sciences, 1985, 31(2): 182-209.

[4] DURAND M, FLAJOLET P. Loglog Counting of Large Cardinalities [J]. Lecture Notes in Computer Science, 2003, 2832: 605-617.

[5] FLAJOLETL P, FUSYL E, GANDOUET O, et al. HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm [C]// Conference on Analysis of Algorithms, AofA 07, DMTCS proc. AH, 2007: 127-146.

[6] WHANG K Y, VANDER-ZANDEN B T, TAYLOR H M. A Linear-Time Probabilistic Counting Algorithm for Database Applications [J]. ACM Transactions on Database Systems, 1990, 15(2): 208-229.

[7] CAI M, PA J P, YU K, et al. Fast and Accurate Traffic Matrix Measurement Using Adaptive Cardinality Counting [C]// Proce-

(下转第 301 页)

基于 MGPS 和基于分组的 PMVS 算法的重建精度略差，而基于特征提取优化的 PMVS 算法的重建精度误差较小。

表 6　多种并行 PMVS 算法的重建精度误差

	基于 MGPS 的 PMVS	基于特征提取优化的 PMVS	基于多线程的 PMVS	基于分组的 PMVS
片面数	4288491	3961573	4210386	4175329
误差率/%	10	2	8	7

结束语　本文提出了一种基于 CPU 多线程和 GPU 的两级粒度并行策略 MGPS，并将基应用于 PMVS 算法的并行优化，实现了 16 幅分辨率为 4081×2993 的无人机航拍遥感影像序列的快速三维重建。实验结果表明，基于 MGPS 的任务分配策略是有效的，且优于现有的并行 PMVS 算法。基于 MGPS 的 PMVS 三维重建算法能够充分利用多核 CPU 和 GPU 的并行计算能力，在可接受的误差范围内取得了明显的加速效果。

从实验结果中可以看出，基于 MGPS 的 PMVS 算法目前还有以下 3 方面值得改进：1）片面扩散的 GPU 并行加速效果并不理想，因此下一步需要对 PMVS 算法的稀疏片面集生成和片面扩散的 CUDA 加速做深入的改进优化；2）基于 CPU 多线程和 GPU 并行的 MGPS 两级粒度并行策略采用的是静态任务划分的方式，需要预先测量单位任务的 CPU 和 GPU 执行的时间比，同时没有充分考虑 CPU 多线程开销和存储器性能，今后还需要深入研究线程开销、存储器以及其他硬件特性对 CPU 多线程程序的执行效率的影响模型；3）若将 MGPS 策略与图像分组的思想相结合，则能够进一步优化无人机影像 PMVS 三维重建的计算速度。

参 考 文 献

[1] SHU B, QIU X J, WANG Z Q. Survey of shape from image[J]. Journal of Computer Research and Development, 2010, 47(3): 549-560. (in Chinese)
束搏，邱显杰，王兆其. 基于图像的几何建模技术综述[J]. 计算机研究与发展，2010，47(3)：549-560.

[2] XIE Z. UAV aerial image-based three-dimensional reconstruction of outdoor scenes[D]. Hangzhou: Zhejiang University of Technology, 2014. (in Chinese)
谢榛. 基于无人机航拍图像的室外场景三维重建技术研究[D]. 杭州：浙江工业大学，2014.

[3] FURUKAWA Y, ACCURATE P J. Dense, and Robust Multi-view Stereopsis[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2009, 32(8): 1362-1376.

[4] SEITZ S M, CURLESS B, DIEBEL J, et al. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms[C]// Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Washington: IEEE Computer Society, 2006: 519-528.

[5] YU M, QI F F, YU Y, et al. 3D reconstruction algorithm based on multi-view stereo[J]. Computer Engineering and Design, 2013, 34(2): 730-733. (in Chinese)
于明，齐菲菲，于洋，等. 基于立体视觉的三维重建算法[J]. 计算机工程与设计，2013，34(2)：730-733.

[6] SHI L M, GUO F S, HU Z Y. An Improved PMVS through Scene Geometric Information[J]. ACTA Automatica Sinica, 2011, 37(5): 560-568. (in Chinese)
史利民，郭复胜，胡占义. 利用空间几何信息的改进 PMVS 算法[J]. 自动化学报，2011，37(5)：560-568.

[7] ZHU Q. The study and realization of multi-view image-based 3D reconstruction algorithm [D]. Lanzhou: Lanzhou University, 2009. (in Chinese)
朱芹. 基于多视点图像的三维重构算法的研究及实现[D]. 兰州：兰州大学，2009.

[8] FURUKAWA Y, CURLESS B, SEITZ S M, et al. Towards Internet-scale multi-view stereo[C]// IEEE Conference on Computer Vision & Pattern Recognition. 2010: 1434-1441.

[9] XIAO X W, GUO B X, LI D R, et al. Multi-View Stereo Matching Based on Self-Adaptive Patch and Image Grouping for Multiple Unmanned Aerial Vehicle Imagery[J]. Remote Sensing, 2016, 8(2): 89-119.

[10] HUI Z, XIE Y, HENG P A. Accelerating feature extraction for patch-based Multi-View Stereo algorithm[C]// 2010 International Conference on Computer Design and Applications (ICCDA). IEEE, 2010: V5-511-V5-515.

[11] XIAO H. Research on high efficiency heterogeneous parallel computing based on CPU+GPU in image matching [D]. Wuhan: Wuhan University, 2011. (in Chinese)
肖汉. 基于 CPU+GPU 的影像匹配高效能异构并行计算研究[D]. 武汉：武汉大学，2011.

[12] ZHANG L, ZHAO Y, HOU K. The Research of Levenberg-Marquardt Algorithm in Curve Fittings on Multiple GPUs[C]// 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). IEEE, 2011: 1355-1360.

[13] XIANG Z, DIANWEN Z. Efficient parallel Levenberg-Marquardt model fitting towards real-time automated parametric imaging microscopy[J]. Plos One, 2013, 8(10): e76665-e76665.

(上接第 282 页)

dings of the 2005 ACM SIGCOMM Workshop on Mining Network Data(MineNet'05). 2005: 205-206.

[8] CHEN Q, LIU Y H, NI, et al. Parallel and Distributed Cardinality Estimation for Large-scale RFID Systems [J]. Parallel and Distributed Systems, 2011, 22(9): 1441-1454.

[9] ZHANG N, ABOULNAGE, et al. Accurate and Fast Cardinality Estimation for XPath Queries [C]//Proceedings of the 22nd International Conference on Data Engineering, 2006(ICDE '06). 2006.

[10] GAO W W. An Algorithm Based on Virtual Vector for Measuring Host Cardinality Distribution[D]. Dalian: Dalian Maritime University, 2013. (in Chinese)
高文文. 基于虚拟向量的主机基数分布测量算法[D]. 大连：大连海事大学，2013.

[11] KNUTH D E. The Art of Computer Programming vol. 3-Sorting and Searching (Second Edition) [M]. Addison-Wesley Professional, 1998.

[12] 字符串 Hash 函数对比[EB/OL]. http://blog.csdn.net/icefireelf/article/details/5796529.

[13] FNV 哈希算法[EB/OL]. http://blog.csdn.net/hustfoxy/article/details/23687239.

[14] 更快更好的哈希函数[EB/OL]. http://blog.csdn.net/wisage/article/details/7104866.