一种自适应的 Hilbert 编码算法及其并行化

王维晨[1] 贾连印[1,2] 王炳月[1] 梁彬彬[1] 卫守林[1,2*]

[1](昆明理工大学信息工程与自动化学院 云南 昆明 650500)

[2](云南省计算机技术应用重点实验室 云南 昆明 650500)

摘 要 高效的 Hilbert 曲线的编码算法作为 Hilbert 曲线应用的基础,具有重要的研究意义。现有编码算法多未考虑不同输入数据的影响,因此在编码时效率较低。为此,在融合高效位操作、快速置位检测等技术的基础上,提出一种自适应的 Hilbert 曲线编码算法 Adapt-HE。该算法根据输入数据的不同,自适应地采用不同的编码策略,能较好地适应不同的数据分布。此外,基于 OpenMP 对该算法进行并行化,进一步提高其编码效率,且可达到较高的加速比。

关键词 Hilbert 曲线 状态视图 Adapt-HE OpenMP

中图分类号 TP3 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.04.036

AN ADAPTIVE HILBERT ENCODING ALGORITHM AND ITS PARALLELIZATION

Wang Weichen[1] Jia Lianyin[1,2] Wang Bingyue[1] Liang Binbin[1] Wei Shoulin[1,2*]

[1](*Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, Yunnan, China*)

[2](*Yunnan Provincial Key Laboratory of Computer Technology Application, Kunming 650500, Yunnan, China*)

Abstract Efficient Hilbert encoding algorithms are the basis of many Hilbert curve applications, which has significant research significance. Most existing encoding algorithms are not efficiency due to not considering the influence of different input data. To address this problem, Adapt-HE, an adaptive Hilbert curve encoding algorithm, is proposed. This algorithm could deploy different encoding strategies according to different input data, thus making it adapt to different data distribution. In addition, an OpenMP based parallel algorithm was also implemented to further improve the encoding efficiency and achieve higher speedup radio.

Keywords Hilbert curve State view Adapt-HE OpenMP

0 引 言

空间填充曲线可将多维数据映射为一维数据,同时保持其在多维空间的局部性,使得复杂的多维空间问题可在一维空间下高效解决,因此其在空间数据处理[1]、图像处理[2-5]、并行计算[6]等诸多领域得到了广泛的应用。

在众多的空间填充曲线中,Z 曲线[7]和 Hilbert 曲线[8]是最为常见的两种。Z 曲线由 Morton[7]提出,其映射规则和编码过程较为简单,但是其空间跳变性较大。而 Hilbert 曲线的映射规则较为复杂,但其具有更好的空间连续性。高效的 Hilbert 曲线的编码算法作为 Hilbert 应用的基础,对其研究有重要的意义和价值。因二维 Hilbert 曲线应用范围最为广泛,本文着重于二维 Hilbert 曲线的编码。Hilbert 编码算法研究由来已久,提出了许多高效的编码算法,如 Butz[9]、Burkardt[10]、Moore[11]等。目前,经典的主流 Hilbert 二维编码算法主要以迭代类为主。

迭代算法中,Burkardt[10]通过逐阶计算进行编码,其计算开销较高。Moore[11]设计了高效非递归编码算法,其主要基于位操作实现,在二维及高维编码上均有

收稿日期:2021-01-24。国家自然科学基金项目(61562054);国家重点研发计划项目(2018YFB1003904);科技部 SKA 专项(2020SKA0110300)。**王维晨**,硕士生,主研领域:数据库,信息检索,并行计算。**贾连印**,副教授。**王炳月**,硕士生。**梁彬彬**,硕士生。**卫守林**,副教授。

较高的效率。曹雪峰等[12]设计实现了紧致 Hilbert 曲线索引算法,避免了数据维度分布差异带来的索引冗余问题,Nair 等[13]研究了在有障碍物的情况下的 Hilbert 编码问题。

近年来,基于状态视图的编码算法以其便捷高效、易实现等优点得到了广泛深入的研究。李绍俊等[14]提出了复杂度为 $O(n)$ 的基于状态矩阵的 Hilbert 快速编码算法,其构建了高效的状态视图,并通过迭代查询状态视图生成最终的编码。文献[15]通过巧妙的象限映射和逆映射,使得设计的一个统一状态视图可同时支持编码和解码操作。上述算法的主要不足在于,其对所有输入数据同等对待,未考虑数据偏斜分布的影响。为此,贾连印等[16]将状态视图和快速置位检测算法结合提出高效的免计前 0 的 Hilbert 编码算法,将其复杂度降至 $O(n-m)$,其中 m 为跳过的阶数,$m \leqslant n$,从而提高了数据向原点偏斜分布时的编码效率。

尽管如此,现有算法多未考虑输入数据的特点,逐阶迭代的方法因需对前部全 0 的阶进行逐阶编码,因此效率较低,而跳过前部为 0 的阶的方法在处理数据非向原点偏斜的场景时,除将退化为逐阶迭代的算法外,还需要额外的置位检测开销,故难以适应数据向非原点偏斜的场景。此外,目前基于 Hilbert 曲线编码的编码算法多为串行算法,难以适应数据规模的快速增长,目前的处理器普遍具有多个核心。因此通过并行化来提高 Hilbert 曲线编码算法的效率非常必要。

为此,本文在现有研究的基础上,提出一种自适应的编码算法 Adapt-HE,Adapt-HE 可根据输入的不同,自适应采用对应的编码策略,可有效地适应均匀分布和偏斜分布的场景。此外,本文基于 OpenMP[17-18] 实现了并行 Hilbert 编码,可进一步提高编码的效率。

1　必要前提

1.1　Hilbert 曲线

1980 年,Peano[19] 第一个提出了空间填充曲线,Hilbert 曲线是众多空间填充曲线中的一种。Hilbert 曲线通过正方形网格上的所有点一次且仅一次,保证曲线上的每一个点在曲线上的位置都是独立的、唯一的。Hilbert 曲线对空间进行网格划分并进行编码,1 阶 Hilbert 曲线将整个空间分割成 2 × 2 共 4 个子区域,然后依次将左下、左上、右上、右下的 4 个子区域依次连接起来即可构成 1 阶开口朝下的 Hilbert 曲线。通过在每个网格位置嵌入一个 1 阶曲线,并对左下网格的曲线左旋 90°并垂直翻转,对右下网格的曲线右旋 90°并垂直翻转,即可得到 2 阶 Hilbert 曲线。采用类似的变换规则可进一步得到更高阶的 Hilbert 曲线。1、2、3 阶 Hilbert 曲线的示意图如图 1 所示。

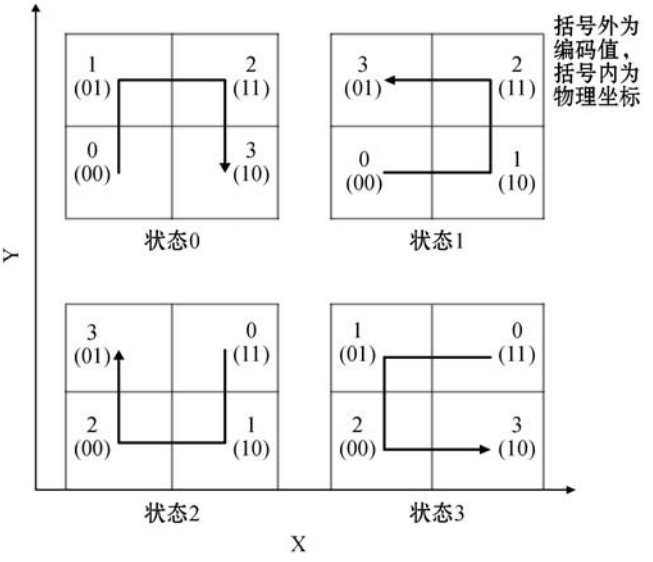

图 1　1 阶 Hilbert 曲线对应的 4 种状态

1.2　OpenMP 简介

OpenMP 是一种跨平台、基于共享内存的并行多线程编程技术,支持 C、C++、Fortran 等多种编程语言。其通过特定的预编译指令,可以非常轻松地将相应的程序段并行化。

OpenMP 采用 fork-join 的执行模型,初始时,只有主线程在执行任务,当执行到并行任务时,主线程会根据当时的任务规模 fork 出多个子线程去并行执行任务,等到并行任务结束时 join 线程,保证了执行结果的完整性。

OpenMP 可通过循环并行化将循环调度到不同的 CPU 核心,实现编码的并行执行。该方式灵活方便,无须对原有串行程序进行太大的修改。OpenMP 支持四种调度策略来实现负载均衡:dynamic、static、guided 和 runtime。dynamic 调度又称动态调度,它初始时为每个线程分配一定量的迭代,可以根据线程的执行快慢,已经完成任务的线程会自动请求新的任务或任务块,而且每次领取的任务块是固定的。guided 调度称为启发式调度,每个线程分配的任务量是先大后小,随着可用迭代次数的降低,每次分配的迭代数量指数递减。static 调度静态地为每个线程分配固定次数的迭代。runtime 调度在运行时根据环境设置自动选择上述 3 种调度策略中的一种。

1.3　符号表

为便于描述和理解,首先给出本文所涉及的符号如表 1 所示。

表 1 命名法

符号	描述
$P(X,Y)$	坐标点 $X=(X_1X_2\cdots X_n)_2$ $Y=(Y_1Y_2\cdots Y_n)_2$
$Z=Z_1Z_2\cdots Z_{2n-1}Z_{2n}$	Hilbert 编码值
n	阶数
p	跳过的阶数
X_i	X 的第 i 阶
Y_i	Y 的第 i 阶
(X_i,Y_i)	第 i 阶坐标值
$Z_{2i-1}Z_{2i}$	第 i 阶的编码值 Z
S_i	第 i 阶的状态

由 $P(X,Y)$ 计算其对应的编码值 Z 的过程称之为编码。

2 Hilbert 曲线编码

2.1 状态视图的构建

状态视图是本文 Hilbert 曲线编码算法的重要基础,它反映了物理坐标、Hilbert 编码值和状态之间的联系。

本文高效的状态视图基于 1 阶 Hilbert 曲线构建。对 1 阶 Hilbert 曲线而言,不同的开口方向对应不同的 1 阶状态。根据开口方向不同,1 阶 Hilbert 曲线共包含如图 1 所示 4 种基本状态:状态 0(开口向下)、状态 1(开口向左)、状态 2(开口向上)、状态 3(开口向右)。

根据图 1,易得出各状态下 1 阶物理坐标 (X_i,Y_i) 与对应的编码值 $Z_{2i-1}Z_{2i}$ 及下一阶状态 S_{i+1} 之间的关系如下:

当 S_i 为 0 时,物理坐标(0,0)、(0,1)、(1,1)、(1,0)对应的二进制编码值分别为 00、01、10、11,S_{i+1} 分别为 1、0、0、3。

当 S_i 为 1 时,物理坐标(0,0)、(0,1)、(1,1)、(1,0)对应的二进制编码值分别为 00、11、10、01,S_{i+1} 分别为 0、2、1、1。

当 S_i 为 2 时,物理坐标(0,0)、(0,1)、(1,1)、(1,0)对应的二进制编码值分别为 10、11、00、01,S_{i+1} 分别为 2、1、3、2。

当 S_i 为 3 时,物理坐标(0,0)、(0,1)、(1,1)、(1,0)对应的二进制编码值分别为 10、01、00、11,S_{i+1} 分别为 3、3、2、0。

参照文献[16]及根据上述分析,构建的状态视图如表 2 所示。

表 2 编码状态视图

状态	坐标	编码值	下一阶状态
0	00	00	1
0	01	01	0
0	10	11	3
0	11	10	0
1	00	00	0
1	01	11	2
1	10	01	1
1	11	10	1
2	00	10	2
2	01	11	1
2	10	01	2
2	11	00	3
3	00	10	3
3	01	01	3
3	10	11	0
3	11	00	2

为降低空间开销并提高编码效率,本文将编码状态视图设计为两个 3 维的数组,分别为 PZMap 和 PSMap。其中 PZMap 存储坐标值到编码值的映射,第 i 阶编码值 $Z_{2i-1}Z_{2i}$ 可通过 PZMap$[S_i][X_i][Y_i]$ 直接获取。PSMap 则存储坐标值到下一阶状态的映射,第 $i+1$ 阶状态 S_{i+1} 可通过 PSMap$[S_i][X_i][Y_i]$ 获取。为便于分析和实现,本文规定,编码前的初始状态 $S_1=0$。

2.2 编 码

如前所述,现有算法多未考虑输入数据的特点,逐阶迭代的方法和跳过前部为 0 的阶的方法均有各自的适用范围和局限。逐阶迭代的方法在算法前部有较多 0 时效率较低,而跳过前部为 0 的阶的算法需要额外的置位检测算法 FFB[20],该算法基于二分检测的思路工作,需要 $O(\log n)$ 的计算开销,因此对前部的阶为 1 的情形,同样将导致编码效率的降低。

由上可见,现有算法难以适应不同分布的数据,若有一种算法能根据输入数据前部阶的特点自动选择合适的算法,将有效提升编码的效率。

在上述思想的指导下,本文设计一种自适应的 Hilbert 编码算法 Adapt-HE。该算法可以根据输入数据前部为 0 的阶的数量,自适应地选择编码的策略。从而可适应不同分布的数据。

如何快速判断输入坐标前部为 0 的阶是否大于 r 是本文算法的关键。一种直观的做法是逐阶自左向右扫描要编码的坐标 $P(X,Y)$ 的坐标分量 X 和 Y,检测 X 和 Y 前部同时为 0 的阶的数量。但如此算法复杂度较高。

为解决这一问题,本文引入一个阈值 r,通过判断 X 和 Y 的前部为 0 的阶是否大于 r 来采取不同的编码策略。为此,本文引入一个高效的位操作 $1<<n-r$ 来生成掩码变量 m_{ask},并通过比较 m_{ask} 和相应的坐标分量决定采用的编码策略。为降低比较开销,仅将 m_{ask} 与 max$V$(X 和 Y 的最大值)进行一次比较。若 m_{axV} 大于等于 m_{ask},采用逐阶迭代的策略进行编码,如此可避免相对昂贵的置位检测算法。否则,则结合置位检测算法检测识别并跳过前部为 0 的阶,仅对后部非 0 阶迭代编码。从而可使算法有较高的适应性和灵活性。

完整的 Adapt-HE 算法如算法 1 所示。

算法 1　Adapt-HE 算法

输入:X 为第一维坐标值,Y 为二维坐标值,n 为阶数,r 为输入坐标前部为 0 的阶数。

输出:Hilbert 编码值 Z。

```
1.  Z←0,S←0,m_ask←1<<n-r,maxV←MAX(X,Y)
2.  if maxV≥m_ask then
3.      for i from 1 to n
4.          Z=Z<<2 | PZMap[S][X_i][Y_i];
5.          S=PSMap[S][X_i][Y_i];
6.  else P←FFB(maxV)                          //置位检测
7.      s←(n-p-1)%2                           //计算第 n-p 阶状态
8.      for i from n-p to n
9.          Z=Z<<2 | PZMap[S][X_i][Y_i]
10.         S=PSMap[S][X_i][Y_i]
```

对于算法 1 的关键步骤作如下解释说明:

(1) 通过位操作 $1<<n-r$ 获取掩码变量 m_{ask} 的值,通过比较 X 和 Y 的最大值获取 max$V$ 的值,然后比较 m_{ask} 和 max$V$ 的大小,进而选择合适的编码策略(第 1 - 第 2 行);

(2) 如果 m_{ask} 的值不大于 max$V$ 的值,则从数据的第一阶开始逐阶迭代地查找 PZMap 和 PSMap 两个数组,获得每一阶的编码值和下一阶的状态,最后循环得到最终的编码值(第 3 - 第 5 行);

(3) 否则,首先通过 FFB 算法检测 m_{axV} 中左侧第一个为 1 的位置 p,然后计算第 $n-p$ 阶的状态(第 6 - 第 7 行);

(4) 得到前 p 阶的编码值为 0,然后再通过逐阶迭代的方式计算剩余阶的编码值,从而得到最终的编码值(第 8 - 第 10 行)。

时间复杂度:对逐阶迭代的情形,该算法的复杂度为 $O(n)$。而跳过前部 0 的情形,其主要开销在于置位检测和迭代编码两个部分,其中置位检测的 FFB 算法[20]的复杂度为 $O(\log n)$,迭代编码的复杂度为 $O(n-p)$,p 为跳过的阶的数量,故其复杂度为 $O(\log n)+O(n-p)$。随着 p 的增大,算法的时间复杂度逐步降低。

空间复杂度:Adapt-HE 算法的主要空间开销在于 PZMap 和 PSMap 的开销,二者分别需要 $4\times2\times2=16$ B 来存储一阶编码和状态,故总的空间开销为 32 B。

2.3　OpenMP 的融合

随着空间数据的规模的增大和多核处理器的普及,对大量数据编码的串行编码将导致效率的降低。故本文结合 OpenMP 实现并行化来进一步提高 Adapt-HE 算法的效率。

OpenMP 主要通过预编译指令对 for 循环并行化的方式实现并行。对本文而言,直观来看,有两种方法可以实现并行。第 1 种是把对算法 1 的第 8 行的 for 循环并行化。第 2 种是对需要进行编码的每个坐标点进行并行化。对前者而言,for 中各次循环之间存在依赖关系,后一阶需要依赖前一阶的计算结果才能继续计算。故本文采用后者对编码进行并行化。

本文实现的并行自适应 Hilbert 曲线编码算法(P-Adapt-HE)如算法 2 所示。

算法 2　P-Adapt-HE 算法

输入:M 为需要编码的坐标集,N 为坐标集的数量。

输出:编码结果 r_{es}。

```
1.  #pragma omp parallel for schedule(static)
2.  for i from 1 to N
3.      r_es = Adapt-HE(M)
```

3　实　验

3.1　实验环境和数据集

实验主要硬件平台:Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz,6 核 12 线程,内存 8 GB。软件环境:Windows 10 64 位,Microsoft Visual Studio C++ 2010。

本文考虑均匀分布和偏斜分布两种情况来生成不同的数据集。对于均匀分布,分别随机生成 4、8、16、24、32 阶的一百万个坐标点。对于偏斜分布,考虑数据偏向原点的情形,并通过阈值参数 r 来控制数据的偏斜程度,r 表示生成的数据前部连续模式至少为 r 阶。为每个特定的 r,各生成 1 百万条 32 阶的随机坐标数据作为实验数据。

3.2 串行算法

为考察本文算法的效率,将其分别与本领域内代表性算法 Moore[11]、李绍俊等[14]提出的算法进行对比。为便于描述,对编码而言,本文将上述算法对应的编码算法增加后缀“-HE”。

3.2.1 均匀分布

为考察均匀分布时阶数对不同算法的影响,给定阶数分别为 4、8、16、24、32,其实验结果如图 2 所示。

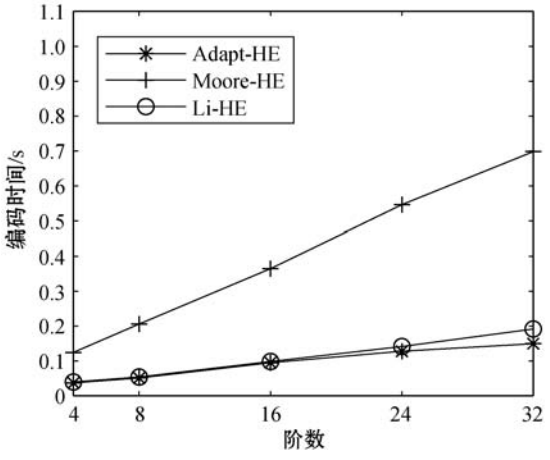

图 2 均匀分布下编码时间对比

由图 2 可知,所有算法的编码时间均随着阶数的增加而增加。Adapt-HE 的效率稍高于 Li-HE,但远高于 Moore-HE。这些结果表明 Adapt-HE 算法避免对输入数据中以连续 0 开始的阶进行迭代编码,可以有效提高编码的效率。

3.2.2 偏斜分布

为考察数据偏斜分布对不同算法的影响,固定编码数据的阶数为 32,设置阈值参数 r 分别为 4、8、16、24,图 3 给出了数据偏向原点时的编码时间对比。

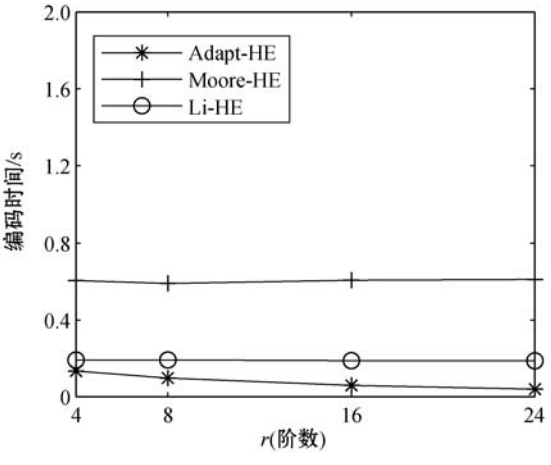

图 3 偏向原点分布编码时间对比

由图 3 可知,Adapt-HE 的编码时间随着偏斜阶数 r 的增加而降低,而其他算法则不随 r 的增加明显变化。这表明 Adapt-HE 算法具有更好的对偏斜数据分布的适应性。

3.3 并行算法

3.3.1 均匀分布下 Adapt-HE 的调度策略

为考察不同调度策略对 Adapt-HE 算法的影响,给出了 dynamic、static 和 guided 3 种不同调度策略下,线程数依次为 2、4、8、16、32、64 时,Adapt-HE 的执行时间。在均匀分布下,固定编码数据的阶数为 32 时,3 种调度策略随着线程数量变化对算法性能的影响如图 4 所示。

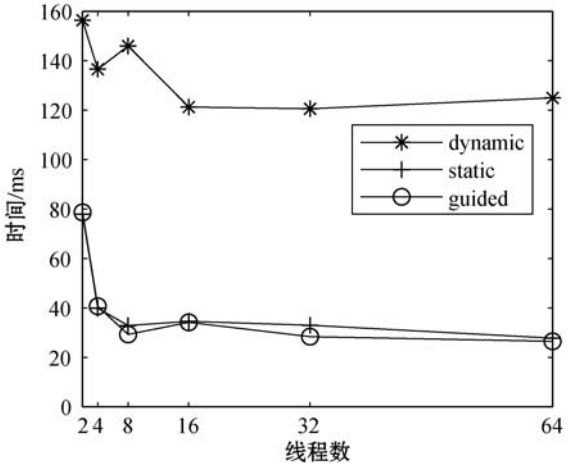

图 4 APA-HE 并行编码时间对比

由图 4 可见,static 和 guided 调度策略优于 dynamic,且 guided 综合性能最优,此实验结果表明 guided 的指数调度算法能较好地实现动态负载均衡,使算法的性能进一步提高。

3.3.2 均匀分布下 Adapt-HE 的加速比

为研究本文实现的并行 Adapt-HE 算法的效率,研究并行算法的加速比,加速比 S 的定义如式(1)所示。

$$S = T_1 / T_Q \tag{1}$$

式中:T_1 表示串行下算法执行的时间;T_Q 表示 Q 个线程时执行的时间。本文以 8 阶的输入坐标为例,计算并行 Adapt-HE 算法在线程数分别为 2、4、8、16、32、64 时的加速比如图 5 所示。

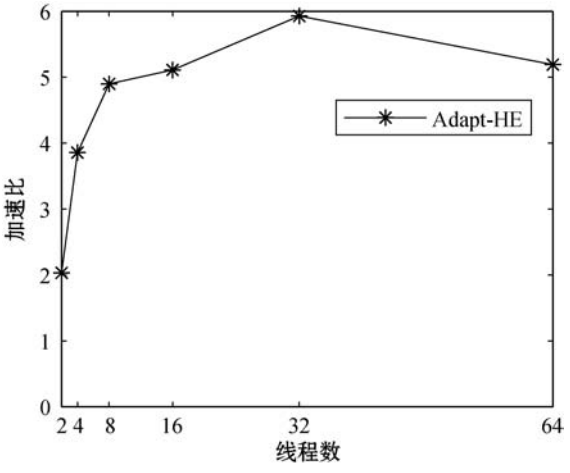

图 5 Adapt-HE 并行加速比

由图 5 可见,Adapt-HE 的加速比随着线程数的增加快速增大,在线程数为 32 时达到或接近峰值,而后随着线程数增加略有下降,通过研究发现,略有下降的原因在于,增加过多的线程会带来额外的开销。对于 Adapt-HE 而言,线程数为 32 时,加速比高达 5.93,取得了较好的加速性能,很大程度上提高了串行编码的效率。

4 结　语

为提高 Hilbert 曲线的编码效率,本文设计一种高效的编码状态视图,结合此状态视图提出一种自适应的编码算法 Adapt-HE。借助高效的掩码操作、置位检测算法等技术,可以根据输入数据的不同,自适应地选择不同的编码策略,从而提高编码效率。并利用 OpenMP 实现其并行编码,使多个输入数据同时编码,将编码效率进一步提高。实验结果表明,本文编码算法可有效提高数据均匀分布和偏斜分布下的编码效率,且具有较好的数据偏斜适应性。下一步拟对高维偏斜分布下的 Hilbert 编码算法进行研究。

参 考 文 献

[1] Kim H, Hong S, Chang J. Hilbert curve-based cryptographic transformation scheme for spatial query processing on outsourced private data[J]. Data & Knowledge Engineering, 2016,104:32 - 44.

[2] Schrack G, Stocco L. Generation of spatial orders and space-filling curves[J]. IEEE Transactions on Image Processing, 2015,24(6):1791 - 1800.

[3] Murali P, Sankaradass V. An efficient space filling curve based image encryption[J]. Multimedia Tools and Applications,2019,78:2135 - 2156.

[4] Weissenböck J, Fröhler B, Gröller E, et al. Dynamic volume lines: Visual comparison of 3D volumes through space-filling curves[J]. IEEE Transactions on Visualization and Computer Graphics,2019,25(1):1040 - 1049.

[5] El-Sharkawy Y, Elbasuney S. Laser induced fluorescence with 2-D Hilbert transform edge detection algorithm and 3D fluorescence images for white spot early recognition[J]. Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy,2020,240:118616.

[6] Liu H, Wang K, Yang B, et al. Dynamic load balancing using Hilbert space-filling curves for parallel reservoir simulations[C]//SPE Reservoir Simulation Conference,2017.

[7] Morton G M. A computer oriented geodetic data base and a new technique in file sequencing[EB/OL]. [2021 - 01 - 02]. https://dominoweb.draco.res.ibm.com/0dabf9473b9c86d48525779800566a39.html.

[8] Hilbert D. Ueber die stetige abbildung einer line auf ein Flächenstück[J]. Mathematische Annalen, 1891, 38(3): 459 - 460.

[9] Butz A. Alternative algorithm for Hilbert's space-filling curve[J]. IEEE Transactions on Computers,1971,20(4):424 - 426.

[10] Burkardt J. Hilbert curve convert between 1D and 2D coordinates of hilbert curve[EB/OL]. [2021 - 01 - 02]. http://people.math.sc.edu/Burkardt/c_src/hilbert_curve/hilbert_curve.html.

[11] Moore D. Fast Hilbert curve generation, sorting, and range queries[EB/OL]. [2021 - 01 - 02]. https://github.com/Cheedoong/Hilbert.

[12] 曹雪峰,万刚,张宗佩. 紧致的 Hilbert 曲线 Gray 码索引算法[J]. 测绘学报,2016,45(S1):90 - 98.

[13] Nair S, Sinha A, Vachhani L. Hilbert's space-filling curve for regions with holes[C]//2017 IEEE 56th Annual Conference on Decision and Control,2017:313 - 319.

[14] 李绍俊,钟耳顺,王少华,等. 基于状态转移矩阵的 Hilbert 码快速生成算法[J]. 地球信息科学学报,2014,16(6):846 - 851.

[15] Xctorres. Hilbert spatial index[EB/OL]. [2021 - 01 - 02] https://github.com/xcTorres/hilbert_spatial_index.

[16] 贾连印,陈明鲜,李孟娟,等. 基于状态视图的高效 Hilbert 编码和解码算法[J]. 电子与信息学报,2020,42(6):1494 - 1501.

[17] Jia L, Zhang C, Li M, et al. An efficient two-level-partitioning-based double array and its parallelization[J]. Applied Sciences,2020,10(15):5266.

[18] 李孟娟,贾连印,陈文焰,等. 基于 OpenMp 的并行集合包含查询算法[J]. 云南大学学报(自然科学版),2016,38(3):376 - 382.

[19] Peano G. Sur une courbe, qui remplit toute une aire plane[J]. Mathematische Annalen,1970,36(1):157 - 160.

[20] Rosetta. Find first and last set bit of a long integer[EB/OL]. [2021 - 01 - 02]. http://rosettacode.org/wiki/Find_first_and_last_set_bit_of_a_long_integer.

(上接第 164 页)

[16] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation[EB]. arXiv:1406.1078, 2014.

[17] Gehrmann S, Deng Y, Rush A M. Bottom-up abstractive summarization[EB]. arXiv:1808.10792,2018.