基于CAMP稀疏重建算法的并行实现

郭 宾[1,2] 张冰尘[1]
(1. 中国科学院电子学研究所微波成像技术重点实验室 北京 100190;2. 中国科学院大学 北京 100190)

摘 要:高分辨率合成孔径雷达(SAR)数据量大,导致稀疏重建过程计算量大。复数近似信息传递(CAMP)是一种收敛速度快的稀疏重建算法,经常被用于稀疏信号重建。为了解决计算量大的问题,提出了一种基于CAMP的并行算法,在计算统一设备架构(CUDA)上对CAMP算法中Chirp Scaling算子和排序算法进行优化。在Chirp Scaling算子中,主要对矩阵转置、FFT和IFFT进行并行优化,并引入并行版本的双调排序。最后,利用串行的CAMP算法和并行的CAMP算法分别重构点目标图像。实验结果表明,在正确重建的前提下,并行的CAMP算法的比串行CAMP算法快29.55倍。

关键词:合成孔径雷达(SAR);复数近似信息传递(CAMP);稀疏重建算法;计算统一设备架构(CUDA)
中图分类号:TN957 **文献标识码**:A **国家标准学科分类代码**:510.70

Parallel implementation of sparse reconstruction algorithm based on CAMP

Guo Bin[1,2] Zhang Bingchen[1]
(1. National Key Laboratory of Microwave Imaging Technology, Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China; 2. University of Chinese Academy of Sciences, Beijing 100190, China)

Abstract: High resolution synthetic aperture radar (SAR) generates mass data, which results in huge computational load. Complex approximate message passing (CAMP) is a kind of sparse reconstruction algorithms with fast convergence speed, so it is often used for sparse signal reconstruction. To solve the problem of huge computational load, this paper presents a parallel algorithm based on CAMP and optimizes Chirp Scaling operator and sorting of CAMP on compute unified device architecture (CUDA). We mainly optimize matrix transpose, FFT and IFFT in Chirp Scaling operator, and introduce parallel version bitonic sorting. Finally, we reconstruct point targets by serial CAMP algorithm and parallel CAMP algorithm respectively. The experiment results demonstrate that, parallel CAMP algorithm is faster (up to 29.55 times) than serial CAMP algorithm on the basis of correct reconstruction.

Keywords: synthetic aperture radar (SAR); complex approximate message passing (CAMP); sparse reconstruction algorithms; compute unified device architecture (CUDA)

1 引 言

高分辨率和宽测绘带SAR数据量大,使得处理雷达数据变得十分耗时。奈圭斯特采样定理[1]指出:对于复信号,只有采样率不小于信号带宽时才能恢复出原始信号。而压缩感知理论指出:对于稀疏信号或可压缩信号,可通过远低于奈圭斯特采样率的观测数据,精确恢复出原始信号[2-4]。稀疏微波成像是近年来微波学科兴起的新理论、新体制和新方法[5],它把稀疏信号处理理论系统地引入微波成像,对SAR系统的理论发展有着重要意义。但是,观测矩阵伴随着数据量增大而显著增大,导致一般计算机的内存根本容不下观测矩阵;而且重建算法的时间复杂度为平方阶$\Theta(N^2)$,复杂度较高。为此,文献[5]提出用模拟算子对稀疏重建算法进行加速。应用模拟算子不仅减小了运行时内存占用,而且把时间复杂度从平方阶降低为线性对数阶$\Theta(N\log N)$。文献[6]提出用Chirp Scaling模拟算子对距离向和方位向解耦,使得大规模场景的稀疏重建成为可能。但是,使用模拟算子仍需要进行很多次迭代,每一次迭代都要进行一次模拟算子成像和逆成像,相比于传统匹配方法的一次成像过程依然很耗时。

随着图形处理器的发展,GPU的浮点数计算能力远远高于同期的CPU,这为用GPU做通用计算奠定了硬件

收稿日期:2016-01

基础。2006年11月，NVIDIA公司推出了CUDA编程架构，使得在GPU上做通用计算更为容易[7]。国内外已有少量在GPU上做稀疏重建算法并行方面的成果，Andrecut等人对MP算法做并行，实现了单精度31倍的加速[8]；在国内，陈帅等人实现了对OMP算法的并行，并实现了8.8倍的加速[9]；耿旻明等人实现了对IST算法的并行，实现大约85倍的加速[10]。CAMP作为一种稀疏信号处理算法被用来进行稀疏信号重建。与文献[8-10]相比，本文的优势在于所实现的CAMP算法收敛速度快，能以指数规律收敛[11]；使用了模拟算子的CAMP算法时间复杂度更低。

本文利用GPU的CUDA编程架构对CAMP算法进行并行，分别对Chirp Scaling算子和排序算法进行并行设计。文章的第二部分简要介绍了压缩感知和CAMP算法；第三部分对CAMP算法进行了简要分析并对Chirp Scaling模拟算子和排序进行了并行设计；最后给出实验结果和总结。

2 压缩感知和CAMP算法

压缩感知理论自提出以来，被广泛用于雷达成像[12]、目标识别和目标追踪[13-14]、图像处理[15]和稀疏信号重构[16]等领域。

对一个k－稀疏信号$\boldsymbol{x}$，其中信号$\boldsymbol{x}\in C^{N\times 1}$。降采样信号$\boldsymbol{y}\in C^{n\times 1}$，$n\ll N$，$\boldsymbol{y}$是对$\boldsymbol{x}$的线性观测，$\boldsymbol{y}$可以表示为：

$$\boldsymbol{y}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{n} \tag{1}$$

式中：$\boldsymbol{A}\in C^{n\times N}$被称为观测矩阵，$\boldsymbol{n}$为观测噪声。由于$n\ll N$，可知式(1)无解或者有无穷多解。压缩感知理论指出，如果观测矩阵$\boldsymbol{A}$满足RIP条件[17]，则稀疏信号$\boldsymbol{x}$能够被正确重建。

LASSO(least absolute shrinkage and selection operator)问题如式(2)所示，其中λ表示正则化参数。

$$\tilde{\boldsymbol{x}}=\min_{x}\frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\right\|_2+\lambda\left\|\boldsymbol{x}\right\|_1 \tag{2}$$

AMP(approximate message passing)算法可解决实数LASSO问题[18]，但不能解决复数LASSO问题。CAMP算法是AMP算法应用于复数LASS0问题的延伸，CAMP算法比AMP算法拥有更高的相位转移曲线和精确度[11]，更适合应用于雷达这类复数信号的处理。给定门限τ，降采样率为δ，原始信号为$\boldsymbol{y}$，观测矩阵$\boldsymbol{A}$，观测噪声$\boldsymbol{n}$方差为σ_t^2，则CAMP算法伪代码如下所示。

初始化：$\hat{\boldsymbol{x}}=\boldsymbol{0}\quad \boldsymbol{z}^0=\boldsymbol{y}\quad t=0$

while $t<iter\max$

$\tilde{\boldsymbol{x}}^t=\boldsymbol{A}^{\mathrm{H}}\boldsymbol{z}^{t-1}+\hat{\boldsymbol{x}}^{t-1}$

$\sigma_t=\mathrm{median}(|\tilde{\boldsymbol{x}}|)\sqrt{1/\ln 2}$

$\boldsymbol{z}^t=\boldsymbol{y}-\boldsymbol{A}\hat{\boldsymbol{x}}^{t-1}+\boldsymbol{z}^{t-1}\frac{1}{2\delta}\left(<\frac{\partial\eta^{\mathrm{R}}}{\partial x^{\mathrm{R}}}(\tilde{\boldsymbol{X}}^t;\tau\sigma_t)>+<\frac{\partial\eta^{\mathrm{I}}}{\partial x^{\mathrm{I}}}(\tilde{\boldsymbol{X}}^t;\tau\sigma_t)>+2<\frac{\partial\eta^{\mathrm{R}}}{\partial x^{\mathrm{I}}}(\tilde{\boldsymbol{X}}^t;\tau\sigma_t)>\right)$

$\hat{\boldsymbol{x}}^t=\eta(\tilde{\boldsymbol{x}}^t;\tau\sigma_t)$

end

输出：$\hat{\boldsymbol{x}}^t$

根据上述算法伪代码CAMP算法可以描述为以下7步：

1)初始化重建后的信号$\hat{\boldsymbol{x}}=\boldsymbol{0}$，残差矩阵$\boldsymbol{z}^0=\boldsymbol{y}$，迭代次数$t=0$；

2)检查终止条件$t<iter\max$是否满足；

3)计算更新信号的噪声估计信号$\tilde{\boldsymbol{x}}^t$；

4)利用中位数估计噪声标准差σ_t；

5)更新和计算残差矩阵$\boldsymbol{z}^t$；

6)用阈值函数更新重建后的信号$\hat{\boldsymbol{x}}^t$；

7)迭代终止，输出重建信号$\hat{\boldsymbol{x}}^t$。

式中：$\boldsymbol{z}^t$表示第t次迭代之后的残差，median表示求解噪声估计信号的中值操作，$<\cdot>$表示对一个向量求平均值，$\boldsymbol{A}^{\mathrm{H}}$表示对观测矩阵$\boldsymbol{A}$的共轭转置。而$\eta(u+vi;\lambda)$可以表示式(3)：

$$\eta(u+vi;\lambda)=\begin{cases}u+vi-\dfrac{\lambda(u+vi)}{\sqrt{u^2+v^2}}, & u^2+v^2>\lambda^2\\ 0, & u^2+v^2\leqslant\lambda^2\end{cases} \tag{3}$$

式中：$\eta(u+vi;\lambda)$也称为CAMP算法的复数软阈值函数。η^{R}和η^{I}分别表示复数软阈值函数的实部和虚部，$\frac{\partial\eta^{\mathrm{R}}}{\partial x^{\mathrm{R}}}$表示$\eta^{\mathrm{R}}$对输入实部的偏导数，$\frac{\partial\eta^{\mathrm{I}}}{\partial x^{\mathrm{I}}}$表示$\eta^{\mathrm{I}}$对输入虚部的偏导数，$\frac{\partial\eta^{\mathrm{R}}}{\partial x^{\mathrm{I}}}$表示$\eta^{\mathrm{R}}$对输入虚部的偏导数。上述算法是用观测矩阵进行计算的，如果利用模拟算子进行计算，需要将算法中第3步中转置后的观测矩阵与向量相乘改为Chirp Scaling成像算子，把第5步中观测矩阵与向量相乘改为Chirp Scaling成像逆算子即可。

3 CAMP算法并行实现

在CAMP算法中，每一次迭代的输入都和上一次迭代的输出有关，也就是上下两步操作有循环依赖，两次迭代间不能并行。在一次迭代过程中，下一步操作也依赖于上一步的操作，因而它们也不能并行。根据阿姆达尔定律，只有对最耗时的部分进行有效并行，才能实现整个算法的加速比最大化。在CAMP算法伪代码中，Chirp Scaling算子操作和中值求解中位数时的排序最为耗时，每次迭代都会出现，而且能够进行并行实现。下面分别对Chirp Scaling算法和排序进行分析和实现。

3.1 Chirp Scaling算子的并行实现

Chirp Scaling算子，就是Chirp Scaling算法写成算子的形式，对编程来说就是一个独立的函数。Chirp Scaling算法是常用的SAR处理算法，通过Chirp Scaling操作将RCMC通过相位相乘实现，从而避免了时域插值操作。Chirp Scaling的算法流程如图1所示。

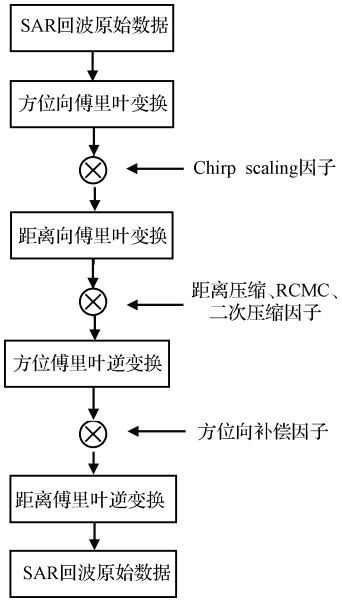


图1 Chirp Scaling算法流程

在Chirp Scaling算法中最为耗时的部分为两次FFT和两次IFFT，而Chirp Scaling操作、距离向补偿因子相乘和方位向补偿因子相乘均是与顺序无关点的数值操作，虽然没有FFT和IFFT耗时长，却非常适合在GPU上做数据并行。但CUDA不区分行和列FFT及IFFT操作，在进行方位向FFT之前需要先对数据进行转置操作，之后再处理距离向数据时，还需要将矩阵转置回来。然而，在GPU全局内存和CPU内存之间来回转移数据耗费时间巨大，会影响整个程序的执行时间。为了充分挖掘GPU的并行计算能力，这里将所有的Chirp Scaling算法执行流程都放在GPU端处理，CPU只负责对Chirp Scaling算法的控制。

GPU端矩阵转置同CPU端一样，从被转置的矩阵中按行读取元素，然后写入新矩阵的各个列中。按行读取CUDA全局内存中的元素是连续访问的，但是写入的时候，由于地址是不连续的，访问CUDA全局的代价较大，效率不高。CUDA共享内存是流多处理器芯片上内存，访问速度比CUDA全局内存快很多，对共享内存的非连续访问不会导致性能上的缺失。因此，可先按行读取被转置的元素到共享内存中来，然后非连续访问共享内存把数据按行写入转置的矩阵中。这样对两块CUDA全局内存都做到连续访问，大大加快了算法执行速度。GPU矩阵转置示意如图2所示。

首先将CUDA二维线程索引映射到要转置矩阵的元素上，将连续读取的元素存入共享内存。然后变换线程索引，将共享内存中元素连续写入到转置后的矩阵中去。

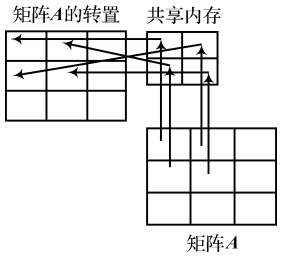


图2 CUDA矩阵转置示意

进行FFT时，直接在GPU端调用CUDA提供CUFFT库函数。CUFFT库是CUDA的FFT库，不仅可以进行正向和逆向FFT操作，而且可以对FFT进行批量计算。这样就使得可以一次进行距离向或方位向的FFT、IFFT操作，不需要使用循环，间接加快算法的执行速度。Chirp Scaling操作、距离补偿因子和方位补偿因子都是与顺序无关的点乘运算，将全局线程索引对应于数据元素下标，就能把CUDA线程映射到对应的数据元素上，理论上所有元素都可以并行执行。

3.2 排序算法的并行

双调排序是一种排序网络，排序网络是排序算法的一种。排序网络是指各个数据元素相互独立，两两元素之间的操作与顺序无关，非常有利于并行实现。双调序列是指一个前一段递增、后一段递减的序列，或者相反的序列。有序序列被认为是一种特殊的双调序列。对于一个长度为2的幂次的双调序列$\{a_n\}$，假设前半段序列满足$a_0 \leqslant a_1 \leqslant \cdots \leqslant a_{n/2-1}$，后半段序列满足$a_{n/2} \geqslant a_{n/2+1} \geqslant \cdots \geqslant a_{n-1}$，则对序列$\{a_n\}$的前后半段每个相隔距离为$n/2$的对应元素进行两两比较和交换，把较小的元素放在前半段，把较大的元素放在后半段。这样，$[0,n/2)$和$[n/2,n)$内的元素序列均为双调序列，而且前半段序列的元素值都不大于后半段序列的元素值。按照以上方式递归进行再划分和比较，当子序列长度为2时，自然就成了单调序列，而且前面序列的元素值小于后面序列的元素值，整个序列是单调递增的，即排序完成。

但是，递归版本的双调排序不利于CUDA并行，文献[19]实现了串行双调排序的循环版本。利用三层循环实现双调排序。第一层循环变量用于bit位的选择，最终决定是升序排序还是降序排序；第二层循环变量是两个元素的距离，用于选取等待排序的两个元素；第三层循环对满足条件的每个元素进行比较操作。其伪代码如下所示。

```
int i,j,k;
for(k=2;k<=N;k<<1)
    for(j=k>>1;j>0;j>>1)
        for(i=0;i<N;i++)
            int   i×j=i^j;
```

```
if(i×j>i)
    if(x)swap(i,i×j);
    if(y)swap(i,i×j);
```

从代码中可以很容易地计算出算法的时间复杂度为 $\Theta(N\log^2 N)$。从伪代码中可以看出，最后一层循环中各个元素的比较是相互独立的，文献[20]对其做了并行。因此，可以将全局线程索引映射到对应元素上去，这样所有满足条件的元素可以并行执行。当然也可以利用共享内存加快执行速度，但是共享内存有限，只能处理数目不多的元素。

经过加速的双调排序算法每个处理器的工作为：$\Theta(\log^2 N)$，相比于串行双调排序的时间复杂度，并行后理论上可加速 n 倍。

4 实验结果

实验的 CPU 平台为 Intel Xeon E5-2650 主频为 2.0 GHz的工作站，实验的 GPU 平台为 Quadro K4000 图形处理器。仿真参数如表 1 所示，场景大小为：距离向 512 个采样点，方位向 512 个采样点。

表 1 稀疏场景仿真参数

参数	值
雷达中心频率/GHz	5.3
发射脉冲宽度/μs	2.5
发射信号带宽/MHz	50
距离向采样率/MHz	60
脉冲重复频率/Hz	100
多普勒带宽/Hz	80

场景为点目标构成的飞机模型，图 3 所示为匹配滤波算法、串行的 CAMP 算法和并行后 CAMP 稀疏重建算法恢复的场景目标。

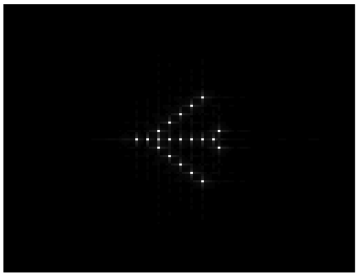
从图 3 可以看出，串行 CAMP 算法和并行 CAMP 算法以及匹配滤波算法恢复的场景位置信息相同，CAMP 算法恢复的图像旁瓣很低，比匹配滤波方法成像质量高。并且，并行后的 CAMP 算法和串行 CAMP 算法重构的场景目标一样，都能正确重建场景目标。

对串行和并行 CAMP 算法的执行时间进行 30 次蒙特卡洛模拟，记录 CPU 上串行 CAMP 算法和 GPU 上并行 CAMP 算法执行时间的平均值，如表 2 所示。

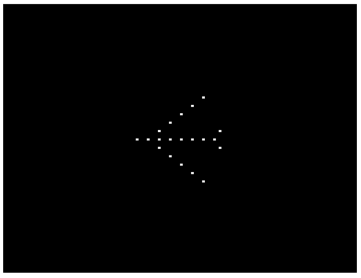
表 2 CAMP 算法在 CPU 和 GPU 上的执行时间

CPU 上的执行时间/s	101.653 124
GPU 上的执行时间/s	3.439 790
加速比	29.55

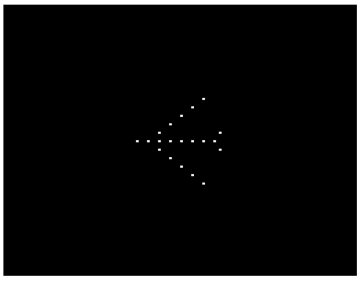
加速比定义为 CAMP 算法串行运行时间与并行运行时间的比值，从表 2 的结果可以计算出并行后的 CAMP 算法实现了大约 29.55 倍的加速。



(a) 匹配滤波算法重建的场景目标



(b) 串行CAMP算法重构的场景目标



(c) 并行CAMP算法恢复的场景目标

图 3 匹配滤波算法、串行 CAMP 算法和并行 CAMP 稀疏重建算法恢复的场景目标

5 结 论

本文提出了基于 CAMP 稀疏重建算法的一种并行实现。通过对 CAMP 算法中最耗时的 Chirp Scaling 算子操作和排序进行分别并行，最终实现了大约 29.55 倍的加速，为使用稀疏重建算法进行大规模的数据重建提供了新的途径。但是，不同的并行方式产生的优化结果也不相同，更好的并行方法有待进一步研究。

参 考 文 献

[1] CUMMING I G, WONG F H. 合成孔径雷达成像-算法与实现[M]. 洪文，胡东辉等译. 北京：电子工业出版社，2007：30-31.

[2] DONOHO D L. Compressed sensing [J]. IEEE Transactions on Information Theory, 2006, 52(4): 1289-1306.

[3] CANDES E J, TAO T. Decoding by linear programming [J]. IEEE Transactions on Information Theory, 2005, 51(12): 4203-4215.

[4] CANDES E J, ROMBERG J, TAO T. Robust uncer-

tainty principles: Exact signal reconstruction from highly incomplete frequency information [J]. IEEE Transactions on Information Theory, 2006, 52(2): 489-509.

[5] ZHANG B C, HONG W, WU Y R. Sparse microwave imaging: Principles and applications [J]. Science in China Series F: Information Sciences, 2012, 55(8): 1722-1754.

[6] JIANG C L, ZHANG B C, FANG J, et al. Efficient l_q regularization algorithm with range-azimuth decoupled for SAR imaging [J]. IET Electronic Letters, 2014, 50(3): 204-205.

[7] NVIDIA. NVIDIA CUDA C programming guide [EB/OL]. 2015-03-05. http://developer.download.nvidia.com/compute/cuda/7_0/toolkit/docs/CUDA_C_Programming_Guide.pdf.

[8] ANDRECUT M. Fast GPU implementation of sparse signal recovery from random projections [J]. Engineering Letters, 2009, 17(3): 151-158.

[9] 陈帅, 李刚, 张颢, 等. SAR图像压缩采样恢复的GPU并行实现[J]. 电子与信息学报, 2011, 33(3): 610-615.

[10] 耿旻明, 蒋成龙, 张冰尘. 基于CUDA的阈值迭代算法并行实现[J]. 中国科学院大学学报, 2013, 30(5): 676-681.

[11] MALEKI A, ANITORI L, ZAI Y, et al. Asymptotic analysis of complex lasso via complex approximate message passing [J]. IEEE Transactions on Information Theory, 2013, 59(7): 4290-4308.

[12] 王康, 叶伟, 劳国超, 等. 一种基于压缩感知的宽带SAR信号侦察方法[J]. 国外电子测量技术, 2014, 33(4): 40-43.

[13] 谈宇奇, 王雪, 林奎成. 基于视觉压缩感知的传感器网络行人目标辨识方法[J]. 仪器仪表学报, 2014, 35(11): 2433-2439.

[14] 孙斌, 金心宇. 压缩感知在无线传感器网络目标跟踪中的应用[J]. 电子测量与仪器学报, 2014, 28(5): 463-468.

[15] 牛涛, 沈为, 张之江, 等. 基于图像间相关性的光场压缩感知[J]. 电子测量技术, 2014, 37(3): 58-65.

[16] 庄晓燕, 赵贻玖. 谱稀疏信号随机等效采样重构方法研究[J]. 电子测量与仪器学报, 2015, 29(10): 1507-1512.

[17] CANDES E J, TAO T. Near-optimal signal recovery from random projections: Universal encoding strategies? [J]. IEEE Transactions on Information Theory, 2006, 52(12): 5406-5425.

[18] DONOHO D L, ARIAN M, ANDREA M. Message passing algorithm for compressed sensing [J]. Proceedings of national academy of sciences of the united states of America, 2009, 106(45): 18914-18919.

[19] THOMAS W. Christopher. Bitonic Sort [EB/OL]. 2015-04-13. http://www.tools-of-computing.com/tc/CS/Sorts/bitonic_sort.htm.

[20] 张舒, 褚艳丽. GPU高性能运算之CUDA[M]. 北京: 中国水利水电出版社, 2009: 190-197.

作者简介

郭宾, 1993年出生, 硕士研究生。主要研究方向为雷达信号处理。

E-mail: yanchixiaguobin@163.com

张冰尘, 研究员。主要研究方向为雷达系统、雷达信号处理、稀疏信号处理。