SAR 并行成像处理的研究

鲍厚兵,黄顺吉,刘 钊
(电子科技大学电子工程学院,四川成都 610054)

摘 要: 该文对 SAR 并行成像处理进行了深入研究,在串行成像算法的基础上,提出了 SAR 并行成像处理的一种粗粒度并行算法和一种混合 FFT 算法,并在国产曙光系列并行机上进行了实现、实验结果证实了该算法是有效的.

关键词: 合成孔径雷达; 成像处理; 并行成像算法; 快速傅里叶变换
中图分类号: TN959.1 **文献标识码:** A **文章编号:** 0372-2112 (2002) 12-01745-03

The Study of Parallel Processing of SAR Imaging

BAO Hou-bing, HUANG Shun-ji, LIU Zhao
(*College of Electronic Engineering of UEST of China, , Chengdu, Sichuan* 610054, *China*)

Abstract: A large grained parallel imaging algorithm based on the existing sequential algorithms is presented in this paper after the deep research on parallel processing of SAR imaging, and a hybrid FFT algorithm is also presented. The algorithms are implemented on DAWNING parallel computers, with the validity of our work confirmed by the results of experiments.

Key words: SAR; imaging processing; parallel imaging algorithm; FFT

1 引言

随着 SAR 数字成像技术的不断发展,对 SAR 图像的成像精度和实时率的要求也愈来愈高,实现 SAR 实时数字成像处理有两条途径:一是 SAR 专用数字信号处理机;一是通用并行计算机.前者的体积小重量轻,可作为机上或星上设备,但它研制成本高,研制周期长,适应性较差.后者研制周期短,研制成本低,容易升级且适应性强.随着计算机科学的飞速发展,采用通用计算机作地面系统的实时处理已成为发展趋势.

2 粗粒度并行算法及实现

2.1 问题的划分

大量的细粒度任务不一定能产生一个好的并行算法,因为大量细粒度任务有可能增加通信代价和任务创建代价.根据表面容积效应[1],通过增加粒度的方法可以减少通信的代价.在一个二维问题中,"表面积"比例于问题的尺寸,而"容积"比例于问题尺寸的平方.因此一个计算单元的通信/计算之比随问题(任务)尺寸的增加而减少.因此,对于一个给定容积(计算)的问题,增加问题的划分粒度可以有效地提高效率,因为它减少了表面积(通信).因此,用粗粒度的划分方法来提高并行成像算法的效率.粗粒度并行成像算法按整个处理过程来划分任务,每个处理节点分得相同大小的一帧原始数据,各个节点负载均衡,所有的处理过程均可并行地完成.由于每个处理器独立地处理一帧图像,因而节省了节点间转置运算的通信时间,提高了并行效率.粗粒度的数据划分形式如图 1 所示.为了减少处理后因为卷积模糊带来的效率损失,应在内存允许的情况下尽量增大每个节点上的数据块,从而有效地利用内存.由于处理过程中会产生卷积模糊,因此在划分时应注意在数据块的交界处应重叠处理一部分数据,这一部分数据大小为:距离向上一个脉冲内的采样点数;方位向上一个合成孔径时间内的脉冲重复频率数.图 1 阴影部分为重叠处理部分.成像算法为 RD[2]算法、CS[3]算法、SRC[4]算法.

2.2 异步计算模式

设并行机共有 n 个处理节点,整个回波数据可以划分成 m 帧($m \geqslant n$)数据.在粗粒度 SAR 并行算法中有三种类型的进程,第一种为数据发送进程或主进程 MP,该进程最先产生并派生其它进程,该进程运行于节点 HN 上,HN 称为头节点;第二种为成像处理进程 CP_i,该进程运行于节点 CN_1 到 CN_n 上,CN_i 称为计算节点;第三种进程为图像接收进程 RP,该进程运行于节点 LN 上,LN 称为尾节点.

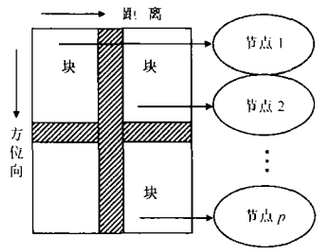

图 1 粗粒度方案的数据划分

当程序开始运行时,MP 先派生成像处理进程 CP_i 和图像接收进程 RP 并将总帧数 m 广播给各节点,然后等待从计算节点 CN_i 发来的数据请求,从 CN_i 发来的数据请求到来后会存储在头节点 HN 的接收缓冲区中,HN 不断探测自己的接收缓冲区,当探测到有请求到达时,HN 判断是否所有的数据帧已发送完毕,如果未完,HN 向该节点发送

收稿日期:2001-07-10;修回日期:2002-03-26

一帧数据和其对应帧号,然后重复上述过程,如果已完,则向计算节点广播结束帧号 $f(f>m)$,最后 HN 进入等待状态,等待尾节点 LN 发来的程序终止标志,一旦接收到终止标志 HN 便退出 MP,整个成像处理过程结束.

CP_i 由 MP 派生而成,分布于各 CN_i(计算节点)上,CN_i 先接收 HN 广播的总帧数 m,然后向 HN 发送数据请求,当接收到 HN 发来的帧号后,先判断帧号是否大于总帧数,若是则将结束帧号 f 发送给尾节点,然后结束,若不是则接收原始数据并进行成像处理,处理完后,将帧号和图像数据发送给 LN,然后重复上述过程.图 2 显示了 CN_i 上成像处理进程的流程.

RP 也是由 MP 派生而成,它运行于 LN 上,开始时 LN 先接收 HN 广播的总帧数 m,然后探测接收缓冲区,一旦有帧号数据到达,就接收并判断帧号是否大于总帧数,若是则向 HN 发结束标志并退出,若不是则接收该帧图像数据,然后重复上述操作.在成像开始时由 HN 读出原始数据并将其划分成一帧一帧的数据发送到各个 CN_i 上,数据是串行加载的,加载时间开销是较大的,采用异步模式可以使数据的传输和计算重叠起来.提高了处理效率.

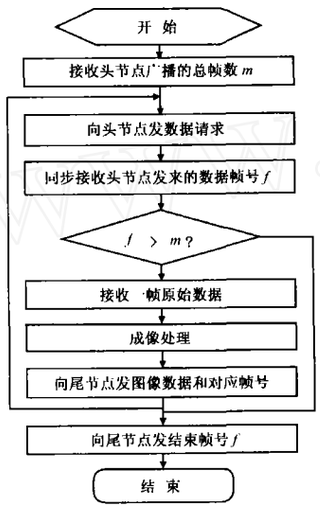

图 2 计算节点处理流程

2.3 多线程机制

多线程机制可在节点内部实现算法细粒度化,以充分利用平台的并行机制.曙光 3000 的计算节点有 4 个 375MHz Power3-II.可在进程 CPi 内生成 4 个计算线程,此 4 线程可同时运行于 4 个不同的 CPU 上.在 CPi 内的循环几乎都可无关地分成 4 个部分,每个线程运行一部分.为了保证运行安全,线程间的同步是极为重要的.将公共存储私有化为线程局部存储和采用关键段同步机制可解决之;每线程处理了 1/4 部分后,需要一大同步,采用信号机制来解决之.即主线程处理循环后,等待其他线程发信号以示处理完毕,设一信号计数器,如计数器到 3,主线程不再等待,执行后续代码.

3 混合 FFT

成像处理中,FFT 运算占用了很大比例的处理时间.在浮点乘法周期长于浮点加法周期的处理平台上实时成像或近似实时成像处理时,Cooley-Tukey FFT 算法不仅速度不够快,而且其单次处理长度局限于 2 的幂,不适合成像信号处理中非 2 的幂处理长度的要求.因此,本文提出了一种混合 FFT 算法,该算法综合了 Winograd FFT 和 Cooley-Tukey FFT 的特点.速度测试和 SAR 成像并行处理的结果,证实了该算法的有效性.

3.1 素因子分解

将输入数据长度分解成素因子积,可将一维 DFT 映射成多维 DFT.一维 DFT

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\ k=0,1,\cdots,N-1 \tag{1}$$

其中 $W_N=e^{-j2\pi/N}$.不妨先假设 N 可分解为

$$N=\prod_{i=1}^{M}N_i,(N_i,N_j)=1;i\neq j \tag{2}$$

采用下列指数变换和中国余数定理,将一维 DFT 映射成多维 DFT.

$$\left.\begin{aligned}n&=\sum_{i=1}^{M}Nn_i/N_i \pmod N\\ k&=\sum_{i=1}^{M}Nm_ik_i/N_i \pmod N\end{aligned}\right\},\ n_i,k_i=0,1,\cdots,N_i-1,\ \langle Nm_i/N_i\rangle_{N_i}=1 \tag{3}$$

由式(3)可得多维 DFT

$$X(k_1,k_2,\cdots,k_M)=\sum_{n_M=0}^{N_M-1}\cdots\sum_{n_2=0}^{N_2-1}\sum_{n_1=0}^{N_1-1}x(n_1,n_2,\cdots,n_M)\times W_{N_2}^{n_2k_2}\cdots W_{N_M}^{n_Mk_M}$$

3.2 多维 DFT 后输出整序

由广义中国余数定理,取输入、输出分别映射为

$$\left.\begin{aligned}n&=\Big\langle\sum_{i=1}^{M}(N/N_i)n_i\Big\rangle_N\\ k&=\Big\langle\sum_{i=1}^{M}N/N_i\times\langle N/N_i\rangle_{N_i}^{-1}k_i\Big\rangle_N\end{aligned}\right\} \tag{4}$$

取常数 $O=\langle\sum_{i=1}^{M}(N/N_i)\rangle_N$,因而可得 n 和 k 的关系式:

$$n=\langle O\times k\rangle_N \tag{5}$$

最后式的输出可按式(5)来得到正序输出.事实上,可在每小 N_i 点 DFT 后整序,从而避免最后整序,提高了运行效率.作下列映射:

$$\left.\begin{aligned}n&=\langle N_2n_1+N_1n_2\rangle_N,n_1,k_1=0,1,\cdots,N-1\\ k&=\langle N_2k_1+N_1k_2\rangle_N,n_2,k_2=0,1,\cdots,N-1\end{aligned}\right\} \tag{6}$$

其中 $N=N_1N_2,(N_1,N_2)=1$,将式(6)带入式(1)得:

$$\hat{X}(k_1,k_2)=\sum_{n_2=0}^{N_2-1}\Big[\sum_{n_1=0}^{N_1-1}\hat{x}(n_1,n_2)W_{N_1}^{n_1k_1}\Big]W_{N_2}^{n_2k_2} \tag{7}$$

其中 $\hat{x}(n_1,n_2)=x(n);\hat{X}(k_1,k_2)=X(k)$

在式(6)中,输入,输出采用了相同的映射,因而式(7)将产生正确的输出顺序,不需最后的整序操作.为了说明如何实现由式(7)所代表的变换,考虑括号中的项,即

$$\bar{X}(k_1)=\sum_{n_1=0}^{N_1-1}\bar{x}(n_1)W_{N_1}^{n_1k_1} \tag{8}$$

其中 $\bar{X}(k_1)=\hat{X}(k_1,k_2);\bar{x}(n_1)=\hat{x}(n_1,n_2)$.

作变换 $k'_1=\langle N_2k_1\rangle_{N_1}$,则式(1)变为一正序的普通 DFT,即

$$\bar{X}(k_1)=\sum_{n_1=0}^{N_1-1}\bar{x}(n_1)W_{N_1}^{n_1k'_1} \tag{9}$$

因此,通常的小因子 DFT 将对产生正常顺序的输出.

3.3 Winograd 大 N FFT

在伽罗华域 $GF(N)$内将小 N FFT 转化为 Radar 短循环卷积[5]的形式,再结合 Winograd 短循环卷积快速算法[5],可得到长度分别为 2,3,4,5,7,8,9,16 的 Winograd 小 N FFT 算法,同 Cooley-Tukey FFT 相比,具有更少的乘法次数和增加不多的加法次数.将大于 N DFT 按素因子分解成多维小 N DFT 的形式,每一小 N DFT 可有 Winograd 小 N FFT 来计算,可得 Winograd 大 N FFT.因而,可得最大 Winograd 大 N FFT 长度为 $16\times9\times7\times5=5040$.利用矩阵克罗内克积的性质,将 Winograd 大 N FFT 化为小克罗内克积级联的形式,亦可得大 Winograd 大 N FFT.

3.4 混合 FFT 算法结构

将 Winograd 大 N FFT 和 Cooley-Tukey FFT 级联起来,可得混合 FFT,其实现框图如下:

其中 $x(n)$为输入.由 3.4 的讨论,可得到多种不同长度的 Winograd大 NFFT,再与 Cooley-TukeyFFT级联,可得更多不同的非 2 的幂的处理长度.

4 实验结果

表 1 中的时间单位为秒.图 4 为两种不同成像算法下的实时率和并行效率,横轴为节点数.随着处理器(节点数×4)数的增加,相应线性地增加问题规模,并行效率保持在 75%以上,且实时率还在逐渐上升.图 5 为一帧 XSAR 图像的 256 级灰度图.

5 结论

粗粒度并行成像算法的每个节点完成一帧图像的处理,对并行平台的通信要求不高,但对内存要求较大,同时,FFT 算法的效率对整个处理性能的影响也是很大的.但 FFT 算法的效率愈高,实现的复杂性也愈高.如何设计更优的算法结构以充分地综合利用给定并行平台的资源,是进一步提高系统处理性能的关键.

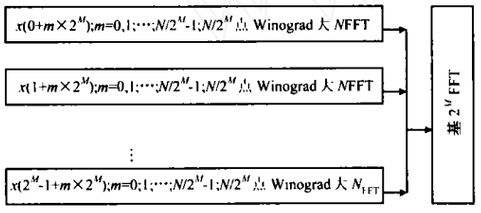

图 3　混合 FFT 结构图

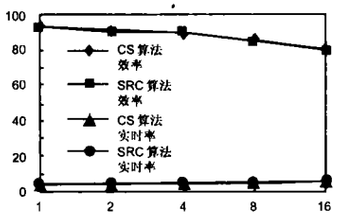

图 4　效率、实时率

图 5　XSAR 图像

表 1　[数据块大小:10080×10080 平台:曙光 3000-Ⅱ + AIX4.33 + PVM;实现语言:C]

节点数	1		2		4		8		16	
CPU 数	4		8		16		32		64	
数据块数	1		2		4		8		16	
算法	SRC	CS	SRC	CS	SRC	CS	SRC	CS	SRC	CS
派生子任务	3.55	2.95	2.63	3.1	3.54	2.98	2.84	3.46	3.25	3.05
加载原始数据	4.66	4.69	8.03	8.46	15.59	16.32	33.52	33.42	69.35	70.53
总计算时间	28.86	78.27	26.78	78.6	26.92	78.1	26.82	78.4	26.73	78.6
输出及存储	4.38	3.79	3.78	4.23	5.31	3.82	4.92	4.05	4.58	3.94
总时间	34.28	85.32	37.19	89.9	46.76	97.9	65.09	118.9	98.49	150.7

参考文献:

[1] Kai Hwang, Zhiwei Xu. Scalable Parallel Computing Technology, Architecture, Programming[M]. New York: Mc Graw Hill, 1998: 75 - 100.

[2] R Balmer. A comparison of range-Doppler and wave-number domain SAR focusing algorithms[J]. IEEE GRS, 1992, 30: 706 - 713.

[3] M Y Jin, F Cheng, M Cheng. Chirp scaling algorithms for SAR processing[J]. Proc IGARSS, 1993: 1169 - 1172.

[4] M Y Jin, C Wu. A SAR correlation algorithm which accommodates large range migration[J]. IEEE GRS, 1984, GE-22: 592 - 597.

[5] E Oran Brigham. The Fast Fourier Transform[M]. New Jersey: Prentice-Hall Inc, 1979. 65 - 100.

作者简介:

鲍厚兵　男,博士生,1970 年生于湖北武汉市.现在电子科技大学电子工程学院攻读通信与信息系统专业博士学位.主要从事 SAR 信号处理、目标识别的研究.

黄顺吉　男,博士生导师,1932 年生于浙江金华市.电子科技大学教授.主要从事 SAR 信号处理、ISAR 信号处理、目标识别的研究以及雷达系统的设计.