高技术通讯 2012 年 第22卷 第2期:159 ~ 164

doi:10.3772/j.issn.1002-0470.2012.02.008

基于 CUDA-GPU 的宽带高速频谱分析系统的研究①

刘东亮②*** 南仁东③* 李建斌*

(*中国科学院国家天文台 北京 100012)

(**中国科学院研究生院 北京 100086)

摘 要 针对射电信号观测的需求,设计并实现了一种基于统一计算设备架构(CUDA)和图形显示卡(GPU)的宽带高速频谱分析系统。该系统通过运用库利-图基(Cooley - Tukey)快速傅立叶变换算法与谱分析算法实现实时宽带高速频谱分析。系统的关键部分在于通过 CUDA 来完成运用线程合并算法对数据在 CPU 与 GPU 之间传递时的转换,并使用并行流水算法在总线中共享多核 GPU 来降低实时运算时间。该系统主要为 500m 口径射电望远镜工程的高分辨率微波巡视项目而设计,实测中满足目标需求,并可应用在射电信号观测或类似的高速密集数据运算中。

关键词 频谱分析,并行计算,库利-图基傅立叶变换,GPU 通用计算

0 引 言

近年来,随着高速宽带信号处理技术的发展,射电信号观测技术极大地拓展开来。这类技术涉及广泛,从高分辨率微波巡视[1],到雷达监测,再到星际介质图谱绘制等,均通过使用自制定软硬件配置来实现对多样化信号的处理需求,实现对海量数据的实时、高精、高速处理。同时,随着可制定运算平台的发展,图形显示卡(graphics processing unit, GPU)开始进入科学研究,用来加速数据处理[2]。在软件实现方面,统一计算设备架构(compute unified device architecture, CUDA)提供了强大而友好的 GPU 接口与内核开发环境;在硬件执行方面,多数 GPU 已具有理论上万亿次浮点指令的高速运算能力,且在构架上更适合处理类似纹理贴图的大量并行处理。较之传统的 CPU 集群系统,GPU 还具有以下主要特点:(1) 内存存取时延(latency)[3]:CPU 通常使用缓存(cache)来减少存取主记忆体的次数,以避免内存时延影响到执行效率,GPU 则多半没有缓存(或很小),利用并行执行的方式来隐藏内存的时延。例如,当第一个线程(thread)需要等待记忆体读取结果时,则开始执行第二个 thread,依此类推。(2) 分支指令:CPU 通常利用分支预测等方式来减少分支指令造成的堵塞,GPU 则多使用类似处理内存时延的方式。不过,通常 GPU 处理分支的效率会比较差。针对这些特点,我们尝试构建 GPU 宽带频谱分析系统,通过库利-图基快速傅立叶变换算法的设计,来有效隐藏内存的时延,并有效利用 GPU 的海量执行单元进行并行计算。目前较为成熟的宽带实时频谱分析系统是搜寻地外文明项目(search for extra terrestrial intelligence, SETI)所使用的第四代近地智能信号搜寻系统(SETI V)[4]。该系统采用了 Intel CPU(Xeon CPU X5450 3.00GHz)与 Xilinx FPGA (Virtex 2 XC5VSX95T)平台的 CPU-FPGA 构架,在 Linux 环境下调用快速傅立叶变换库(fastest Fourier transform in the West, FFTW)对信号数据进行频谱分析。该系统在常规运行中,当输入端以复采样 8bit 精度对 128MHz 带宽,1Hz 谱分辨率信号进行观测时,系统将 256M 个样点的 16bit 数据(8bit 实部,8bit 虚部)由字符型(单字节)转换为单精度浮点型(四字节)数据后,调用 FFTW 对整个 1G byte 信号数据运算。该系统报告 1Gbyte 数据的运算时耗均值为 5.94s[4]。鉴于运算需求接近,我们采用同样的数据源,旨在通过 GPU 系统来实现并加速这一过程。

① 国家重大科学工程 FAST(发改高技[2007]1538 号)资助项目。

② 男,1983 年生,博士;研究方向:天文技术与方法;E-mail:dlliu@bao.ac.cn

③ 通讯作者,E-mail: nrd@bao.ac.cn

(收稿日期:2010-07-02)

1 频谱仪系统与 GPU 运算流程

频谱仪通常需要从背景噪声中检测出具有一定强度的窄带信号,通过 GPU 运算可以加速这一过程。当天线接收到目标信号,经过模数转换、采样量化后通过欠采样正交滤波器将信号转换到基带并传输至主机内存中。采样数据由主机内存传输到 GPU 的显存,经过谱分析运算后将结果输出到二进制文件中。其中包含了目标信号强度、带宽、频谱序列以及信号能量谱,实际观测中还要包含时间标记,观测站(点)经纬度等。

与传统 CPU 不同,GPU 由多个有层阶架构的多核处理器构成,其本身就是一个可执行多线程的单指令多数据(single instruction multiple data, SIMD)并行系统。每个多核处理器包含多个流处理器,流处理器可以通过共享架构的片上存储单元来共享线程数据[5]。因此设计中采用库利-图基傅立叶变换算法来优化 GPU 内存读取,通过对目标数据布局来满足线程调用的内存读取与内存交换的合并。

GPU 频谱仪运算主要包括 7 个步骤,在源信号不断输入的情况下循环进行。通常 GPU 运算每一步都需要读写全局内存数据,设计中引入了并行执行来减少全局内存的读写操作。整个运算流程如图 1所示。具体算法步骤如下:

(1) 主机内存向 GPU 内存拷贝数据:数据由主机通过 PCI-Express 总线传输至 GPU 内存中。

(2) 数据格式转换:在 GPU 中将输入的复数据由字符型(1byte)转换为浮点型(4byte),该操作是由于字符型数据会导致 GPU 运算精度不足。

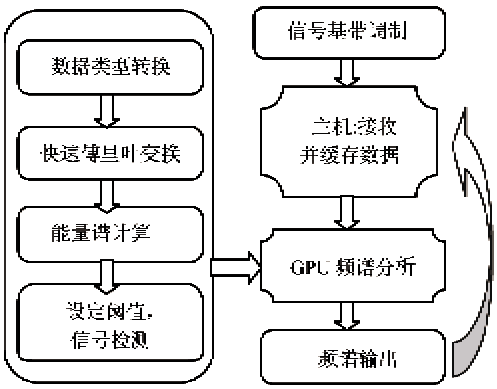

图 1 GPU 频谱仪构架与运算流程

(3) FFT 运算:在 GPU 上执行库利-图基 FFT 分解算法。

(4) 计算能量谱:从 FFT 输出结果中计算能量谱,如果样本 N 的 FFT 输出为 $X_n + iY_n$,则对应的能量谱为 $X_n^2 + iY_n^2$。

(5) 检测阈值能量谱:通过计算平均能量谱来设定阈值,从而检测并筛选出超过阈值的能量谱。

(6) GPU 内存向主机内存拷贝数据:将检测出的信号能量谱传输至主机内存。

(7) 能量谱输出:将能量谱以数值或图形的形式在主机中输出。

2 Cooley-Tukey FFT 算法与单 GPU 频谱仪系统实现

首先我们使用 Cooley-Tukey FFT 算法来实现 FFT 运算,基本思路是将长序列离散傅立叶运算转换为短序列的 FFT 多基分解与之后进行矩阵变换得出频谱,此一过程在 GPU 的并行层阶架构中得到加速。设复合数 $N = N_1 \times N_2$,N_1, $N_2 > 1$,则将序列 $X(n)$ 做 N_2 个点的 N_1 分解后进行离散傅立叶变换[6],可得 $N = N_1 \times N_2$ 点 FFT 为

$$X[k_0,k_1] = \sum_{n_1=0}^{N_2-1}\{[W_N^{n_1k_1}\sum_{n_1=0}^{N_1-1}x[n_0,n_1]]W_{N_1}^{n_0k_0}\}W_{N_2}^{n_1k_1}$$

$$= \sum_{n_1=0}^{N_2-1}W_{N_2}^{n_1k_1}\underbrace{\left(W_N^{n_1k_1}\sum_{n_0=0}^{N_1-1}x[n_0,n_1]W_{N_1}^{n_0k_0}\right)}_{N_1\text{点变换}}$$

根据以上推导,假设源数据存储在矩阵 A 中,数据总量为 $N = N_1 \times N_2$ 个,算法流程如图 2 所示。

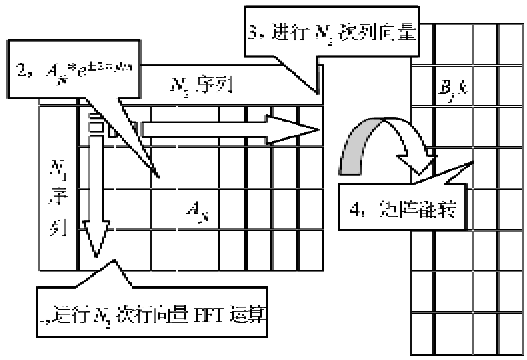

图 2 Cooley – Tukey FFT 算法与矩阵变换

具体算法步骤如下:

(1) 对矩阵 A 的行列数据,同时进行 N_1 个 N_2 点的 1-D FFT 运算。

(2) 将运算后的 A_{jk},乘以对应 $\omega_N = e^{-2\pi ij/n}$。

(3) 再次同时进行 N_1 个 N_2 点的 1-D FFT 运算

得到 $N_1 \times N_2$ 阶复矩阵。

(4) 将 $N_1 \times N_2$ 复矩阵转换为 $N_2 \times N_1$ 复矩阵。

在 CUDA 2.3 环境中,算法实现的部分代码如下:

```
Float * d_x,d_y;                          /* 在 GPU 上分配空间,并传输 GPUN1 个 N2 点数据 */
CUDA_SAFE_CALL(cudaMalloc((void **)&d_x.men_size_x));
for (j=0, j< N1-1, j++)
{   cudaMemcpy (Aj[k], 0≤ k ≤N/N1-1);          /*执行第一次 1-D 复至复 FFT 变换*/
    cufftPlan1d(&plan, N2,CUFFT_C2C, BATCH);
    cufftExecC2C(plan, d_x, d_y, CUFFT_FORWARD);}
  Bj[k] = yj[k] * e^{-2πjk/N};                          /* 乘以旋转因子 ωN = e^{-2πjk/n} */
cufftPlan1d(&plan, N2, CUFFT_C2C, BATCH);       /* 执行第二次 1-D 实至复 FFT 变换 */
cufftExecC2C(plan, d_x, d_y, CUFFT_FORWARD);
```

值得注意的是,在执行步骤 1 的过程中,由于对矩阵 A 行向量的选取的不连续会造成不连续的内存读取,大大影响 GPU 运算性能。目前大多数基于 GPU 的 FFT 算法,例如 CUFFT,都为默认数据连续选取[7]。因此在步骤 1 调用 CUFFT 前,必须在算法之前做矩阵变换,并在步骤 3 之前,再次对矩阵做变换,增加了两次变换耗时。

为了解决这个问题,我们设计了一个寄存器密集型的 FFT 内核来执行步骤 1 运算而不用进行矩阵变换。例如设计 16 点的 FFT 内核直接控制运算单元中的线程,通过 FFT 内核提前调用线程中的寄存器存取不连续数据值[8],从而实现避免矩阵变换操作。这里由于寄存器数量决定了 N_1 值,会限制 FFT 的规模,参照硬件规格本设计中使用 $N_1 = 16$[9]。CUFFT 2.3[10] 定义了 CUFFT 的最大长度即 N_2 值为 8M 点 FFT,因此基于单 GPU 可实现最大为 16 * 8M = 128M 点的 FFT 运算。步骤 4 中使用了 CUDA SDK 2.3[10] 中的转换矩阵算法,该算法优化了内存读取与栈冲突。最后在算法中添加了能量谱计算,CUDA 库优化了平方和与数据类型转换等平行运算,缩减了时耗。

对于输入信号最大长度选择,取决于 GPU 视频存储器(video random access memory, VRAM)的大小。对于当前市场的 GPU,例如 Tesla 系列 GPU 均具有 4GB VRAM[10], 因此在单 GPU 下可以满足 128M 点 FFT 不间断运算的需求。

3 GPU 集群频谱仪系统的实现

GPU 集群频谱仪系统的目标为实时处理更宽带的信号频谱。如前所述,GPU 内存大小限制了频谱带宽,为了实时处理信号,需要将数据分布到多个 GPU 之中。这里给出基于多 GPU 频谱仪的设计,该设计可以在双 GPU 系统中处理 256M 点的信号频谱,并可拓展至 4 到 8 个 GPU 协同工作。

设计的难点在于分布式 FFT 的实现,以及减少 GPU 与 CPU 之间大量的传输操作引起的性能下降。这里使用如下 4 点来改进性能:

(1) 通过设置数据缓冲区来减少传输数据总量。

(2) 通过交叠冗余的计算和数据传输来减少执行时间。

(3) 使用并行流水构架来减少运算时间。

(4) 省略单 GPU 算法步骤(4)中的矩阵转换操作,减少传输数据。

3.1 FFT 算法实现

参照单 GPU 频谱仪实现的思路,设有 M 个 GPU,$N = N_1 \times N_2$ 点表示输入信号谱长度,数据存储为 $N_1 \times N_2$ 的复矩阵中。多 GPU 频谱仪则有如下操作:

(1) $N_1 \times N_2/M$ 点数据从主机传输到每个 GPU 中。每个 GPU 传输的数据为 $2 \times 8 \times N/M$ bit $= 2N/M$ byte,总数据为 $2N$ byte。

(2) 将输入字符型数据转换为浮点型数据(1byte ×4)。

(3) 同时执行 N_2/M 个 N_1 点 FFT。

(4) 将运算值乘以旋转因子。

(5) 将 GPU 内存中 $N_1 \times N_2/M$ 个结果数据传输至主机。每个 GPU 传输数据量为 $2 \times 4 \times N/M$ byte,总量为 $2 \times 4 \times N$ byte。

(6) 再将 $N_1 \times N_2/M$ 个数据从 CPU 传回 GPU 中,传输数据的大小为每个 GPU 为 $2 \times 4 \times N_1 \times N_2/M$byte,总数据量为 $2 \times 4 \times N$ byte。

(7) 同时执行 N_1/M 个 N_2 点 FFT,完成算法。

该算法在 GPU 与 CPU 中传输的总数据量为 $2N+8N$ byte。图 3 中显示了当 $M=2$ 时整个数据流程。该算法的优点在于可以灵活按照 GPU 的数量来定制，并且当 GPU 数量增加时，由于每个 GPU 所需要的处理量减小，传输数据总量依然维持恒定（最多为 $10N$byte）。另一方面，如果 GPU 与 CPU 之间的带宽减小时，$10N$byte 数据传输就成为了 GPU 频谱仪的瓶颈。

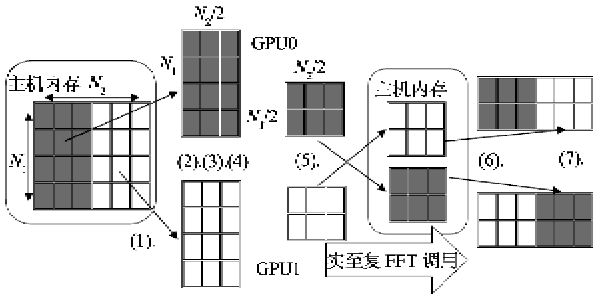

图 3 双 GPU 系统的 FFT 实现数据流

3.2 并行流水线结构

GPU 通过 PCI-Express 接口与主机相连。例如在 NVIDIA Tesla S1060 系统中，具有 4 个 GPU、2 条 PCIe 总线。当 GPU 共享总线时，需要 2 倍的传输时间来完成数据传输，降低了系统性能。针对这种情况，我们设计了协同的时序来满足共享总线 GPU 之间的交替工作，从而避免同时传输而降低性能。按照单 GPU 算法实现的前 3 个步骤，将整个数据传输也分为 3 部分：

首先，需要传输 N/M 次 M 点 FFT 的数据，其次进行旋转因子的相乘与 N/M 点的 FFT 运算，最后计算信号能量谱，检测阈值，并输出频谱文件。图 4 中显示了共享 PCIe 总线系统中的 4GPU 的流水线时序图，其中 GPU0 与 GPU2 共享 PCIe1，GPU1 与 GPU3 共享 PCIe2。当 GPU2 进行第一步传输 FFT 数据时，GPU0 进行第二步 FFT 运算，图中 4-GPU 流水线结构之后 2 个 GPU 同时进行第 3 步，如此往

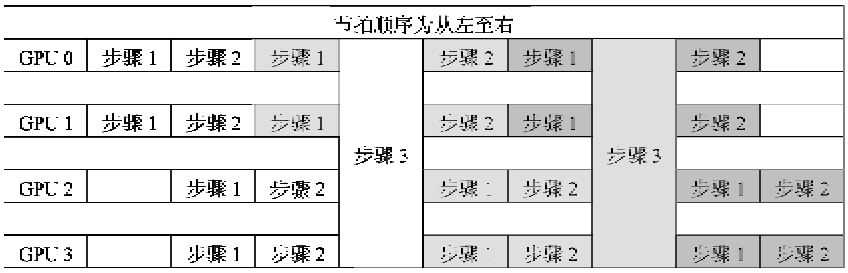

图 4 4GPU 共享总线流水图示

复。此时，总线不会产生冲突，满足每个 GPU 的传输要求。

4 GPU 频谱仪系统测试

我们在 NVIDIA Tesla S1060 计算平台上进行了 GPU 频谱仪的测试。系统主要参数如表 1 所示。

表 1 测试系统参数

CPU：Core i7 920
主板：MSI X58M
内存：4GB DDR3-1333
软件环境：CUDA Toolkit 2.3
总线：PCI-Express 2.0x16 × 2

实测中配置了 2 块 NVIDIA Tesla C1060 GPU，每个 GPU 具备 4GB 显存，构建 2GPU 的频谱仪系统。测试环境中的输入信号为夹杂周期脉冲的白噪声——模拟射电观测的通常信号源。白噪声可由伪随机数来生成，这里使用 MTMG（Mersenne twister Mseudo-random generator[8]）程序生成白噪声。基于时序考虑，生成了不同长度的源信号，并对每组信号通过 GPU 频谱仪运算 20 次，取平均时耗值。整个测试系统如图 5 所示。

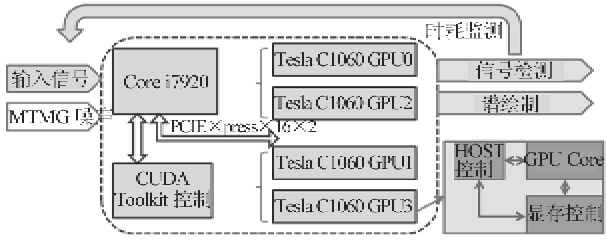

图 5 GPU 频谱分析系统图

脉冲信号长、目标信号谱宽与输出数据长如

表2所示。这里要注意的是,在对实际宽带信号观测中,还需要时间标度(time stamp),观测站(点)经纬度等信息。因此输出数据量较理论值大。

表 2　测试数据参数

输入信号谱宽	输出数据大小	目标信号谱宽
16M	24kB	2047
32M	48kB	4096
64M	96kB	8192
128M	177kB	15082
256M	192kB	16383

4.1　单 GPU 频谱仪测试结果

对单 GPU 频谱仪,对不同长度的输入谱宽进行测试,从 16M 点到 128M 点。图 6 和图 7 给出单 GPU 的执行时间、每个步骤占用百分比以及每步操作在总时间中的比例,表 3 给出了具体数值。

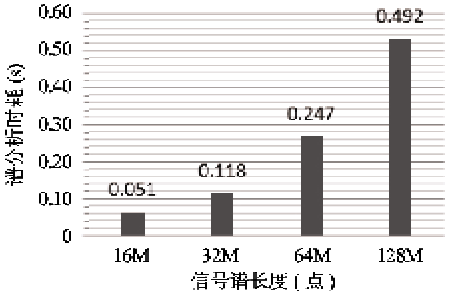

图 6　单 GPU 频谱分析时耗统计

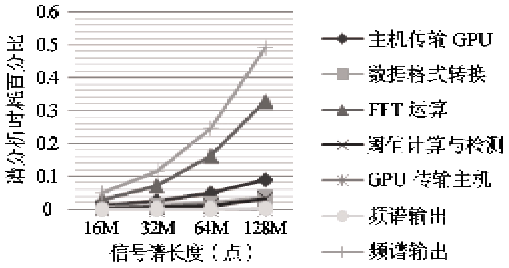

图 7　单 GPU 频谱分析时耗百分比统计

表 3　单 GPU 频谱分析时耗数值

	主机传输至 GPU		格式转换		FFT 运算		阈值计算与检测		GPU 传输至主机		频谱输出		总耗时
	耗时(s)	%	耗时(s)	%	耗时(s)	%	耗时(s)	%	耗时(s)	%	耗时(s)	%	(s)
16M	0.0122	23.7	0.0062	12.6	0.0283	55.1	0.0039	7.6	0.0003	0.6	0.0005	1.0	0.051
32M	0.0268	22.8	0.0102	8.6	0.0718	61.0	0.0077	6.5	0.0005	0.4	0.0007	0.6	0.118
64M	0.0506	20.5	0.0202	8.2	0.1423	65.7	0.0323	4.9	0.0008	0.3	0.0012	0.5	0.247
128M	0.0912	18.5	0.0374	7.6	0.2735	66.5	0.0825	6.6	0.0011	0.2	0.0023	0.5	0.492

图 7 中显示所有运算均在 1s 以内完成。在处理 128M 点时,16 点 FFT 内核的表现为 67 GFlops(Flops 值由公式 $M\times5\times N\log_2 N/$(FFT 执行时间)估算,其中,$M$ 表示同时进行的 N 点 FFT 个数,由表 3 可知 128M 点时时耗为 0.27s,则 $1\times5\times27\times2^{27}/0.27\approx67.1$G),而其理论峰值为 920 GFlops[5]。

这表明该设计可以在大至 128M 点时满足实时处理频谱的需求。图中还表明除传输带宽瓶颈外,运算中 FFT 占有最大比例,因此要进一步提升性能,需要对 FFT 算法再优化。

4.2　双 GPU 测试结果

对双 GPU 系统,我们测试了 64M,128M,256M 点。其中测试时间为从前一能量谱输出到后一能量谱输出的时耗,双 GPU 系统在 128M 点时,较单 GPU 提升了 30% 的性能。在表 4 中总结了不同 GPU 个数,不同长度信号与执行时间。从表中可以看出,对同样长度的信号频谱分析,双 GPU 运算 FFT 时,时间减半,这主要在于将 FFT 分散到 2 个 GPU 中,每个 GPU 执行的 FFT 长度减半。具体而言,在执行 128M 点 FFT 时,单 GPU 耗时 0.27s,而双 GPU 耗时为 0.16s。此时数据传输并没有减少耗时,传输总量恒定,具体而言,128M 点信号输入时,单 GPU 传输总时耗为 0.092s,双 GPU 传输时耗为 0.13s。从测试结果看来,均满足设计目标,即实时处理 128M 点的频谱信号。

表 4　单、双 GPU 频谱分析耗时对比

	每 GPU 谱长度	每 GPU 传输文件	主机传输 GPU(s)	FFT 运算(s)	阈值检测(s)	传输主机(s)	总耗时(s)
单 GPU	128M	256MB	0.0912	0.2735	0.0825	0.0011	0.4483
双 GPU	64M	256MB	0.1324	0.1581	0.0172	0.0005	0.3176

5 结 论

本文给出了基于 GPU 集群的宽带实时信号频谱分析系统的构架、算法与测试。在使用库利 - 图基快速傅立叶变换算法实现了单 GPU 的测试系统后，进一步移植至多 GPU 系统中，对算法进行优化，并加入了流水构架设计。系统测试结果表明，在对不大于 128M 点信号进行频谱运算时，该系统均可以在 1s 内完成，达到了 SEIT V 项目计中的运算需求，并将 5.94s[4] 这一目标缩短了 5/6，证明了该系统可以高性能地进行实时宽带谱的观测。

后续的工作为再深入对算法，尤其是 FFT 实现上的优化，并且考虑 4GPU 等更多 GPU 集群系统进行更宽带频谱的实时运算。我们相信，随着对 GPU 构架本身的熟悉和新的 CUDA 库的完善，GPU 在一定领域里取代 CPU 集群将成为未来科研应用的趋势。这些新技术和产品的研发，将为科学研究的不断深入与拓展提供有力的支持。

参考文献

[1] 南仁东. 500 m 球反射面射电望远镜 FAST. 中国科学 G 辑-物理学/力学/天文学，2005，35(5)：449-466

[2] 针对研究领域的 CUDA，CUDA ZONE. http://www.nvidia.cn/object/cuda_research_cn.html：NVIDIA，2010

[3] Blake G，Dreslinski R G，Mudge T. A survey of multicore processors. *Signal processing Magazine*，*IEEE*，2009，26(6)：26-37

[4] Werthimer D，Ng D，Bowyer S，et al. The Berkeley SETI program：SERENDIP III and IV instrumentation. *Progress in the Search for Extraterrestrial Life*，1995，74：293-301

[5] Harris C，Haines K，Simith L S. GPU accelerated radio astronomy signal convolution. *Experimental Astronomy*，2008，22(1-2)：129-141

[6] Moreland K，Angel E. The FFT on a GPU. In：Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware，Eurographics Association，Aire-la-Ville，Switzerland，2003. 112-119

[7] Bailey D H，FFTs in external of hierarchical memory. In：Proceedings of the 1989 ACM/IEEE conference on Supercomputing，Reno，USA，1989. 234-242

[8] Owens J D，Luebke D，Govindaraju N，et al. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*，2007，26(1)：80-113

[9] Nukada A，Ogata Y，Endo T，et al. Bandwidth intensive 3-d FFT kernel for GPUs using CUDA. In：Proceedings of the 2008 ACM/IEEE Conference on Supercomputing，Piscataway，USA，2008. 1-11

[10] CUDA CUFFT library version 2.3，CUDA ZONE. http://developer.nvidia.com/object/gpucomputing.html，NVIDIA，2009

A high-speed wideband spectrometer system based on CUDA-GPU

Liu Dongliang[*,**]，Nan Rendong[*]，Li Jianbin[*]

([*] National Astronomical Observatories，Chinese Academy of Sciences，Beijing 100012)

([**] Graduate University of the Chinese Academy of Sciences，Beijing 100086)

Abstract

This paper presents the design and implementation of a high-speed wideband Spectrometer system based on the compute unified device architecture (CUDA) and graphics processing unit (GPU) for radio signal observation. This system executes the Cooley-Tukey FFT algorithm and the spectrum analysis algorithm to achieve high-speed analysis of a wideband spectrum. This was accomplished by the algorithm design in CUDA，with an optimized combination of Multi-Memory access，parallel and pipeline processing to reduce the redundant calculation. In particular，this system was designed for the HRMSFF project (high resolution microwave survey for the radio telescope FAST)，and it also can be utilized on other radio signal processing and related computationally intensive calculation.

Key words：spectrum analysis，parallel computation，Cooley-Tukey FFT，CUDA-GPU