



一种基于 Kepler 架构 GPU 的通信仿真加速方法

韩秉君¹, 黄诗铭², 杜 滢¹

(1. 中国信息通信研究院 北京 100191; 2. 北京邮电大学 北京 100876)

摘 要:提出了一种在 Kepler 架构 GPU (graphics processing unit, 图形处理器) 上利用 CUDA (compute unified device architecture, 统一计算设备架构) 技术加速通信仿真中 DFT (discrete Fourier transform, 离散傅里叶变换) 处理过程的方法。该方法的核心思想是利用线程级并行技术实现单条收发链路内部 DFT 运算的并行加速, 并利用动态并行和 Hyper-Q 技术实现不同收发用户对之间链路处理过程的并行加速, 从而最终达到加速仿真中 DFT 处理过程的目的。实验结果表明, 相对单核单线程 CPU 程序和上一代 Fermi 架构 GPU 程序, 该方法分别能够将 DFT 处理速度提升 300 倍和 3 倍, 具有较好的加速效果。

关键词: CUDA; 仿真平台 DFT; 加速方法; 并行处理

doi: 10.11959/j.issn.1000-0801.2015248

A Simulation Accelerating Method Based on CUDA with Kepler GPU

Han Bingjun¹, Huang Shiming², Du Ying¹

(1. China Academy of Information and Communication Technology, Beijing 100191, China;

2. Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract: An accelerating method based on CUDA (compute unified device architecture) with Kepler GPU (graphics processing unit) was proposed to speed up the DFT (discrete Fourier transform) processing in the communication simulation platform. Based on this method, the whole DFT processing was split into subtasks named molecular-subtasks corresponding to communication links and a molecular-subtask was further split into smaller parallel subtasks named atomic-subtasks which correspond to the DFT processing in a link. Then, the atomic-subtasks were processed in parallel by the threads in a GPU kernel function, as well as the molecular-subtasks were processed in parallel via several GPU kernel functions to shorten the simulation time. Simulation results show this method can speed up the DFT processing more than 300 times compared with single thread CPU program and 3 times compared with traditional GPU program.

Key words: compute unified device architecture, simulation platform DFT, accelerating method, processed in parallel

收稿日期: 2015-03-10; 修回日期: 2015-09-30

基金项目: 国家科技重大专项基金资助项目 (No. 2014ZX03003011-003)

Foundation Item: The National Science and Technology Major Project (No. 2014ZX03003011-003)

论文引用格式: 韩秉君, 黄诗铭, 杜滢. 一种基于 Kepler 架构 GPU 的通信仿真加速方法. 电信科学, 2015248

Han B J, Huang S M, Du Y. A simulation accelerating method based on CUDA with Kepler GPU. Telecommunications Science, 2015248



1 引言

随着通信领域的研究开始从 4G 技术向 5G 技术过渡,以大规模多天线阵列、超密集网络以及大带宽毫米波通信等为代表的一系列技术成为未来研究的重点。这些技术从天线数、用户数、总带宽和复杂度等维度将系统级仿真的总体运算量提升了 1~3 个数量级,进而给系统级仿真平台的搭建工作带来巨大挑战。

在计算量大增的情况下,为了使仿真总时间仍能被控制在可接受的范围内,仿真加速技术也成为未来构建系统级仿真平台的必选技术。传统仿真过程的总体运算量不大,采用加速技术的仿真平台往往仅能节省数小时至数天的运算时间,却要在硬件购置、代码开发、测试校准以及平台升级等方面花费更多时间和成本,总体来看收益有限;而在 4G 到 5G 的过渡时代,随着系统级仿真总体运算量的增大,采用加速技术的仿真平台有可能节约数周甚至数月的仿真时间,从而保证在总体运算量大增的情况下依旧能将仿真时间控制在可接受的范围内,具有较高的应用价值。

通常情况下,通信仿真中存在大量可并行处理的双精度浮点运算,非常适合采用 GPU(graphics processing unit,图形处理器)进行加速。而 CUDA(compute unified device architecture,统一计算设备架构)作为一种能够均衡硬件设备成本、开发维护成本和运行能耗成本的 GPU 并行计算架构,为通信系统级仿真加速提供了一种理想的技术手段。CUDA 程序采用类 C++ 语言开发,其最新版开发工具 CUDA Toolkit 7.5 已经针对主流开发环境如 Visual Studio 2013 实现较好整合,并针对目前主流的 Kepler 架构 GPU 提供了动态并行^[1]、Hyper-Q^[2]、统一内存寻址、CuBLAS 和 CuFFT 等一系列编程工具和函数库,显著降低了代码开发和维护的工作量。

目前,关于使用 GPU 加速通信仿真平台问题,一些文章从通信模块加速和 GPU 加速关键技术两方面提供了具有参考性的研究结果:在通信模块加速上,参考文献[3]提出了一种针对多径信道的 GPU 加速方法,参考文献[4]提出了一种针对 OFDM 系统蒙特卡罗仿真的 GPU 加速方法,参考文献[5]提出了一种针对 LDPC(low density parity check code,低密度奇偶校验码)编码处理的加速方法,参考文献[6]提出了一种利用英伟达公司 GPU 加速 FFT 处理模块的方法;在 GPU 加速关键技术上,参考文献[7]介绍了

利用 MPI(multi point interface,多点接口)解决多 GPU 系统性能的方法,参考文献[8]介绍了动态管理多核多线程 GPU 资源的方法。

然而,GPU 硬件和软件能力在近两年取得了长足进展,而目前尚未有关于如何利用这些进展加速系统级仿真平台的研究。因此,本文提出了一种在 Kepler 架构显卡下利用 CUDA 技术加速通信仿真中信道 DFT^[9](discrete Fourier transform,离散傅里叶变换)处理过程的方法。该方法的主要创新点是以接收用户为依据,将整个仿真中相互交错的通信链路拆解成一系列独立的“多发送用户到单接收用户”接收束,并基于这些接收束将 DFT 计算量分解为对同步和数据交互要求低、易于并行处理的子任务。最后,该方法一方面利用动态并行技术实现对不同接收束不同链路计算任务的并行处理,另一方面利用线程并行技术实现了对同一链路内不同采样点的 DFT 计算任务的并行处理,在控制开发复杂度的前提下达到较好的并行加速效果。

2 通信链路与并行任务模型

本节主要介绍了用于加速系统级仿真 DFT 过程的逻辑模型。该逻辑模型的核心是将收发设备之间彼此交错的信道拆分成便于函数级并行处理的接收束结构和便于线程级并行处理的天线—频点阵状结构,从而可以综合利用不同粒度并行技术达到较理想的加速效果。

2.1 接收束与分子级并行任务

一般说来,系统级仿真过程中的信道关系可以抽象为发送设备、接收设备以及发送和接收设备之间的通信链路,如图 1 所示。

图 1 中, $TE_1, TE_2, \dots, TE_n$ 表示 n 个发送设备, $RE_1, RE_2, \dots, RE_m$ 表示 m 个接收设备。

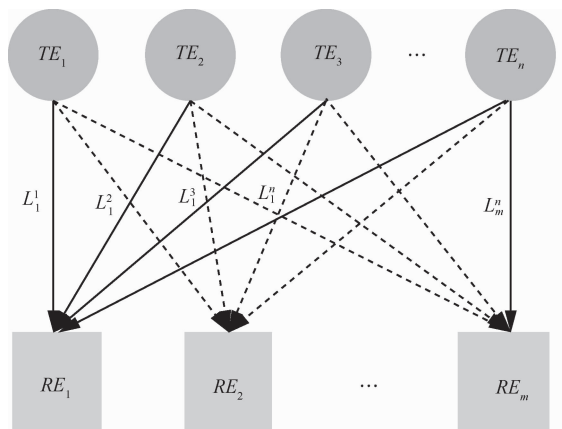


图 1 发送设备、接收设备及通信链路

$\dots, RE_m$ 表示 m 个接收设备, L_{ij}^r 表示 TE_i 到 RE_j 的通信链路。在仿真中, 不同收发设备间的通信链路处理过程通常是相互独立的, 可以并行进行; 而对于某一接收设备如 RE_1 , 则需要完整收集与其相连的所有通信链路 $L_{11}^1, L_{11}^2, L_{11}^3, \dots, L_{11}^n$ 的处理结果后, 才能继续分析该设备的接收性能。基于以上分析, 本文从逻辑上提出了一种接收束结构, 如图 2 所示。

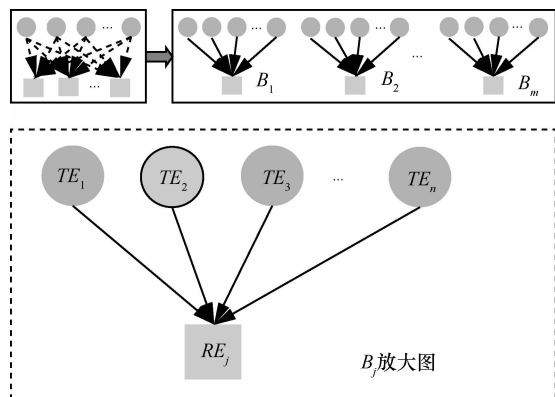


图2 接收束结构

图2中, 从接收用户角度将图1中彼此交错的通信链路分为 m 个接收束。其中, 每个接收束包含 n 个发送机、1个接收机以及它们之间的 n 条通信链路。在实际仿真过程中, 因为调度等原因, 在不同仿真时刻, 一个接收束内处于发送状态的发射机可能是不同的, 这要求处理接收束内不同链路的并行机制应具备足够弹性, 以适应子任务规模的动态变化。考虑到接收束中每一条通信链路上的处理任务还可以继续拆分为更小的可并行子任务, 因此将每条链路的处理任务称为分子级子任务。按照束状结构进行拆分后, 不同链路上的分子级子任务所需的输入数据往往是相互独立的, 不同分子级子任务的处理过程也不涉及数据共享和交互。这意味着并行处理这些分子级子任务时可以节约因数据共享和同步而造成的时间开销, 以达到更好的加速效果。

2.2 天线—频点与原子级并行任务

对于信道束 B_j , 其中某一链路 L_{ij}^r 上的处理任务又可以从逻辑上进一步拆解成图3所示的多条子径上的处理任务。图3中, $AT_i^1, AT_i^2, \dots, AT_i^p$ 为发送设备 TE_i 的 p 个发送天线, $AR_j^1, AR_j^2, \dots, AR_j^q$ 表示接收设备 RE_j 的 q 个接收天线。第 r 个发送天线 AT_i^r 到第 s 个接收天线 AR_j^s 之间的子径用 $P_{j,s}^{i,r}$ 表示。

在仿真过程中, 仿真平台会根据信道模型生成某一子

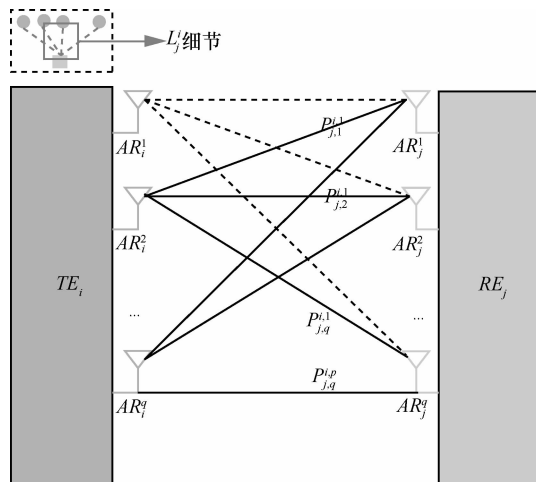


图3 通信链路中的子径

径 $P_{j,s}^{i,r}$ 的时域离散采样点。这些时域离散采样点通常需要通过离散傅里叶变换转换为频域采样值后才能继续进行下一步处理。DFT 的计算式为:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{j2\pi \frac{k}{N} n} \quad (1)$$

其中, $x[n]$ 是时域上的 N 个采样点, $X[k]$ 为转化后的频域上的采样值。根据欧拉公式, 可以得到:

$$e^{j2\pi \frac{k}{N} n} = \cos\left(2\pi \frac{k}{N} n\right) + j\sin\left(2\pi \frac{k}{N} n\right) \quad (2)$$

结合式(2)和式(3)可得:

$$X[k] = \sum_{n=0}^{N-1} x[n] \left(\cos\left(2\pi \frac{k}{N} n\right) + j\sin\left(2\pi \frac{k}{N} n\right) \right) \quad (3)$$

从式(3)可以看出, 无论是不同子径上的 $X[k]$, 还是同一子径上不同采样点的 $X[k]$, 其计算过程都是彼此无关的, 这意味着并行进行不同 $X[k]$ 计算时不会涉及数据共享和同步问题, 能够简化代码开发难度并节省较多的时间开销。每一个 $X[k]$ 的计算过程很难进行更细的拆分, 因此将其称为原子级子任务。

3 不同子任务的 CUDA 处理

本节提出了一种基于 Kepler 架构 GPU 的 DFT 并行加速方法。具体而言, 该方法首先利用线程级并行来加速某一通信链路内不同子径及不同频域采样点的 DFT 计算, 又利用动态并行技术将以上不同通信链路的 DFT 处理过程进一步并行, 以提高加速效果。

3.1 天线—子载波的线程级并行处理

线程是 CUDA 程序处理一个任务的最基本单元,



CUDA 程序通过启用多个线程并行处理一个大任务中的不同部分,以实现加速效果。CUDA 的线程通过 grid、block 和 thread 3 个级别进行组织^[10],每个线程都可以通过其所在的 3 级组织编号得到自己唯一的线程编号,该编号可以是一维、二维或三维的。CUDA 加速的核心原理是将一个计算任务分解成若干个子任务,并根据线程编号设计每个线程和子任务的对应关系。程序执行过程中会根据任务需要启动合适数量的线程,这些线程根据其线程编号和预设对应关系自动并行处理其所对应的子任务,以达到加速效果。

因此,将任务分解并建立线程和分解后子任务的关系,是使用 CUDA 加速的最核心问题。对于第 2.2 节描述的某一通信链路的 DFT 处理过程,由于其不同子径和不同采样点的 DFT 形式均相同,每一采样点的求取过程运算量不大,且很难再进一步分解成可以并行执行的子任务,因此非常适合采用线程级并行加速。

以图 3 中通信链路为例,图 3 中发射天线有 p 个,接收天线有 q 个,假设每个子径有 v 个频域采样点需要求取,则该通信链路共需要进行 $p \times q \times v$ 次 DFT 运算,这些 DFT 运算之间没有相关性,可以并行处理。因此,启动 $p \times q \times v$ 个线程来一次性并行处理这些 DFT 运算,其线程号与任务映射关系如图 4 所示。

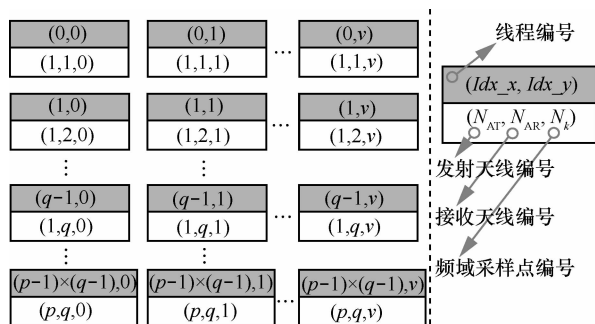


图 4 线程号与 DFT 运算的映射关系

图 4 中每一矩形均表示某一个线程与其所承担 DFT 子任务之间的对应关系,矩形中深色部分为该线程的线程编号,采用二维编号体系,形式为 (Idx_x, Idx_y) ;浅色部分表示该线程所承担 DFT 子任务的三维索引,形式为 (N_{AT}, N_{AR}, N_k) ,其 3 个维度分别代表发送天线号、接收天线号以及频域采样点号,表示线程号为 (Idx_x, Idx_y) 的线程负责处理第 AT 个发送天线与第 AR 个接收天线之间第 k 个频域采样点上的 DFT 运算。其中线程编号和子任务之间的关系如式(4)所示。

$$\begin{cases} N_{AT} = \lfloor Idx_x / q \rfloor + 1 \\ N_{AR} = (Idx_x \bmod q) + 1 \\ N_k = Idx_y \end{cases} \quad (4)$$

其中:

$$\begin{cases} 0 \leq Idx_x \leq p \times q - 1 \\ Idx_y \leq v \end{cases} \quad (5)$$

按照式(4)所述的逻辑,一共启用 $p \times q \times v$ 个线程来并行处理对应的 $p \times q \times v$ 个 DFT 运算。图 4 中根据线程编号将不同线程排列成一个矩形阵列。处于同一行的线程表明其处理的是同一子径上不同频域采样点的 DFT,前 q 行的线程表明其处理的是第 1 个发射天线到所有 q 个接收天线之间子径的 DFT,即图 3 中 $P_{j,1}^{i,1}, P_{j,2}^{i,1}, \dots, P_{j,q}^{i,1}$ 子径的 DFT。以此类推,第 $p \times q$ 行表示处理的是第 p 个发送天线到第 q 个接收天线的 DFT 任务,即图 3 中 $P_{j,q}^{i,p}$ 子径。

在进行 DFT 任务并行处理过程中,有几个原则需要进一步解释:首先,尽管 CUDA 支持对线程的三维编号,本部分依然将收发天线压缩成一维,主要是考虑经过如此方式压缩后,输入输出数据都可以比较方便地采用二维矩阵存储,后继处理也可以利用 MKL、CuBLAS 等矩阵运算工具,可以降低开发难度;第二,没有使用快速傅里叶变换(FFT)算法,主要是考虑在系统级仿真过程中,每一个子径的时域采样点数目通常不大,一般为 20~30 个,而仿真最终需要用的频域采样点数目也不确定,典型的 20 Mbit/s 带宽的 LTE 系统可能有 50~1 200 个采样点,因此采用 DFT 实现会更灵活一些。

3.2 不同收发链路的函数级并行

如图 5 所示,在 Kepler 架构下,CUDA 提供了全新的动态并行和 Hyper-Q 技术,从而使 CUDA 程序能够更加便利地实现多个 GPU 函数之间的并行。关于动态并行,传统的 CUDA 程序只支持由 CPU 函数启动 GPU 函数,而在 Kepler 架构显卡下,动态并行技术允许一个 GPU 函数

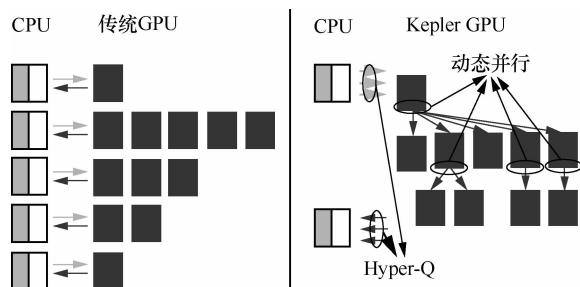


图 5 动态并行和 Hyper-Q 技术

启动另一个 GPU 函数, 该技术减少了 CPU 和 GPU 之间不必要的信息交互, 在简化编程代码复杂度的同时提升了程序的效率。关于 Hyper-Q, 当使用流的形式在 CPU 和 GPU 之间复制数据时, 传统架构显卡中不同流之间的数据要在硬件队列中排队, 而 Hyper-Q 允许 GPU 能够同时并发执行最多 32 个活动的任务, 并且增加了硬件队列的数量^[11], 从而极大提升了数据复制效率和程序的运行速度。

由以上描述可以看出, 动态并行和 Hyper-Q 技术能够根据任务量在显卡中启动相应数量的函数, 并能够较为便捷地进行 CPU/GPU 之间数据交互, 非常适合第 2.1 节提出的数目不定的不同通信链路的处理过程。因此, 提出了如图 6 所示的分子级任务并行方案。图 6 中, 所有接收束被分成了多个批次。首先, 第一批接收束的输入参数借助多个流传入 GPU 中; 第二, GPU 通过动态并行技术, 并行执行第一批接收束中的所有通信链路的 DFT 运算, 其中每个通信链路的 DFT 处理过程如第 3.1 节所述; 第三, 通过多流技术, 将处理结果传回 CPU 中; 最后, 重复以上步骤, 直到所有批次的接收束都处理完成为止。

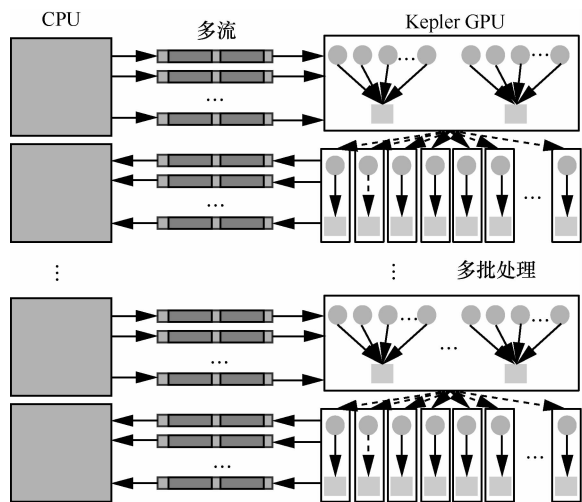


图 6 不同通信链路的并行方法

关于接收束的并行加速方案, 有如下几个细节需要特别解释: 首先, 并行方案中之所以对接收束分批次处理, 主要是因为实际测试表明, 当一次并行的任务量过大时, GPU 函数的处理效率会变得极低, 从而拉长仿真时间。而采用分批处理, 能够规避这一问题。此外, 进行多批次处理, 也有利于将处理后的结果及时传回, 从而使 CPU 在 GPU 处理下一批接收束时能够同步处理本次返

回的接收束结果。第二, 仿真中采用多流的原因是: 实际测试表明, 通过多个流的复用可以显著节省程序中的数据复制时间。

4 加速结果与分析

本节参照典型的 19 小区/57 扇区 20 Mbit/s 带宽 LTE 下行链路提出了如下仿真参数^[12], 见表 1。

表 1 系统仿真参数

条目	取值
系统拓扑	57 扇区/570 用户
发送天线数/根	4
接收天线数/根	2
最小频域采样点/个	50(50 RB, 1 个/RB)
最大频域采样点/个	1 200(100 RB, 12 个/RB)
时域采样点	20

需要强调的是, 传统的仿真平台为了节省仿真时间, 对通信链路的处理过程进行了简化, 而本实验过程假设所有通信链路都进行 DFT 处理, 并以此为基础研究运算总量、每批次处理数量以及所用流数目对总仿真时间的影响。

首先, 对比了采用 CPU、Fermi 架构 GPU 以及 Kepler 架构 GPU 处理系统全部 $57 \times 570 \times 4 \times 2 \times 50 \approx 1\,300$ 万次 DFT 运算的总时间, 见表 2。表 2 中的 CPU 运算时间是在单核单线程条件下得到的, 多核 CPU 运算时间近似为该时间/核数。从表 2 中数据可以看出, 本文所提出的加速方法相对于 CPU 大约加速 300 倍, 相对于上一代 Fermi 架构 GPU 大约加速 3 倍。此外, 考虑到本文所提方法代码实现更简单, 因此在实际应用过程中还能进一步节省开发、测试和维护时间。

表 2 DFT 运算时间

设备型号	Intel Q9500	Tesla C2050	Tesla K20c
硬件架构	Core2	Fermi	Kepler
运行时间/s	32	0.31	0.10

第二, 研究了本文所提算法的加速效果和总体运算量的关系, 如图 7 所示。可以看出, 随着运算总量的增加, GPU 运算总时间近似呈线性增长; 在运算量较大的情况下, 进行单位次数 DFT 运算的时间更小, 其原因主要是加

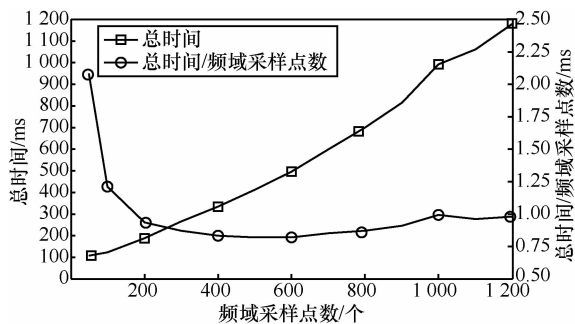


图7 加速效果与总体运算量关系

速过程中 CPU-GPU 间的数据复制、GPU 线程管理等相对固定的时间开销在大计算量情况下得到了“分摊”,这也说明本文所提出的加速方法在运算量较大的情况下加速效果会更明显。

第三,研究了 1 200 个采样点条件下加速效果和处理批次的关系,如图 8 所示。可以看出,如果采用单批次处理,GPU 处理完全部 DFT 运算大约耗时 7.8 s,而改为 8 批次处理后,耗时降为大约 1.2 s,两者相差 6 倍以上。该研究结果说明在使用 CUDA 技术加速通信仿真平台时,应将每次交给 GPU 的计算任务控制在合适范围内,否则可能会影响最终的加速效果。

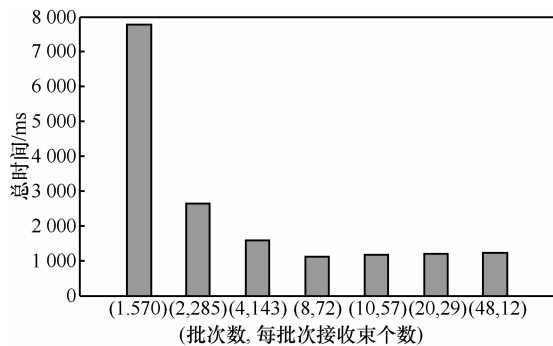


图8 加速效果和处理批次的关系

最后,研究了加速效果和输入输出流数目的关系,如图 9 所示。可以看出,随着输入输出流数目的增加,运算时间呈现出明显的递减趋势。当流数超过 16 时,仿真时间基本稳定。该研究结果说明输入输出流的数目会对加速效果构成较大影响,应尽量选择尽可能多的流进行输入输出,以避免数据传输成为加速瓶颈。

5 结束语

本文提出了一种基于 Kepler 架构 GPU 的系统级仿真 DFT 加速方法。该方法利用动态并行和 Hyper-Q 等技术,

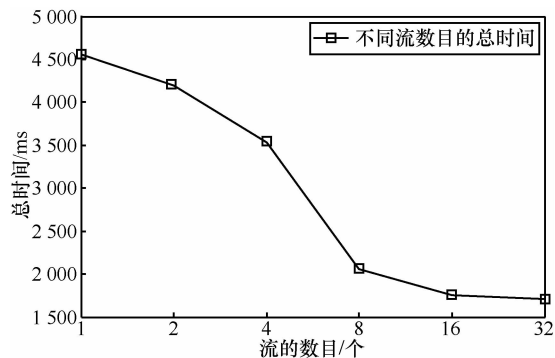


图9 加速效果与输入输出流个数的关系

使 DFT 处理过程相对于单核单线程 CPU 程序加速 300 倍,具有较好的应用价值。此外,该方法中提出的从接收用户角度将加速任务分为多个接收束的思想,对于处理 DFT 之外的加速任务同样有着重要的参考意义。

总之,该方法为解决仿真时间随计算量增加而恶性膨胀的“仿真时间瓶颈”问题提供了切实可行的应对方向及技术手段,是保证 4G/5G 大规模仿真预研工作顺利开展的基础性技术。未来将基于接收束的思想,研究如何利用 CUDA 技术加速通信仿真中接收机信噪比计算的方法以及基于迭代容器的更加灵活的 CPU、GPU 接口,以构造更易用的通信仿真基础模块。

参考文献

- 1 NVIDIA Corporation. CUDA toolkit documentation v7.5. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, 2015
- 2 NVIDIA Corporation. Nvidia kepler GK110 next-generation CUDA compute architecture. http://www.nvidia.com/content/PDF/kepler/NV_DS_Tesla_KCompute_Arch_May_2012_LR.pdf, 2012
- 3 Abdelrazek A F, Kaschub M, Blankenhorn C, *et al.* A novel architecture using NVIDIA CUDA to speed up simulation of multi-path fast fading channels. Proceedings of the 69th IEEE Vehicular Technology Conference, Barcelona, Spain, 2009
- 4 Laguna-Sanchez G A, Prieto-Guerrero A, Rodriguez-Colina E. Speedup simulation for OFDM over PLC channel using a multithreading GPU. Proceedings of IEEE Latin-American Conference on Communications (LATINCOM), Belem, Brazil, 2011
- 5 Potluri S, Wang H, Bureddy D, *et al.* Optimizing MPI communication on multi-GPU systems using CUDA inter-process communication. Proceedings of the 26th IEEE International on Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), Shanghai, China, 2012: 1848~1857

- 6 Wu J, JaJa J, Balaras E. An optimized FFT-based direct Poisson solver on CUDA GPUs. *IEEE Transactions on Parallel and Distributed Systems*, 2014(1): 550~559
- 7 Beermann M, Monro E, Schmalen H, *et al.* High speed decoding of non-binary irregular LDPC codes using GPUs. *Proceedings of IEEE Workshop on Signal Processing System (SiPS)*, Taipei, China, 2013
- 8 Rodriguez A, Valverde J, Torre E, *et al.* Dynamic management of multikernel multithread accelerators using dynamic partial reconfiguration. *Proceedings of the 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, Montpellier, France, 2014
- 9 Proakis J G. *Digital Signal Processing*, 4th Revised Edition. London: Pearson Prentice Hall, 2009: 105~129
- 10 Noga A, Topa T. Kernel execution strategies for GPU-accelerated version of method of moments. *Proceedings of the 20th International Conference on Microwaves, Radar, and Wireless Communication (MIKON)*, Gdansk, Poland, 2014
- 11 Wilt N, *The CUDA Handbook*. Upper Saddle River: Addison-Wesley, 2013
- 12 Bilel B R, Navid N. Cunetsim: a GPU based simulation testbed for large scale mobile networks. *Proceedings of International Conference on Communications and Information Technology (ICCIT)*, Hammamet, Tunisia, 2012

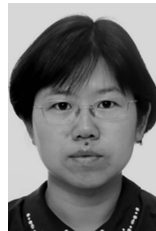
[作者简介]



韩秉君,男,博士,中国信息通信研究院标准工程师,主要研究方向为移动通信领域系统级仿真平台构建,在干扰共存、系统级仿真平台加速技术等方面有较深积累。



黄诗铭,男,北京邮电大学硕士生,主要研究方向为通信系统级仿真平台构建,对GPU加速、通信模型并行化处理方面有较深积累。



杜滢,女,中国信息通信研究院高级工程师,主要从事无线通信技术研究、标准化和评估工作。作为主要成员参与 3GPP LTE、LTE-Advanced 技术研究和国际标准化工作,目前负责 LTE R13 国际标准化制定和 5G 国际标准化预研工作。