基于CUDA的并行AES算法的实现和加速效率探索

费雄伟[1] 李肯立[2] 阳王东[1,2] 杜家宜[2]
(湖南城市学院信息科学与工程学院 益阳 413000)[1] (湖南大学信息科学与工程学院 长沙 410008)[2]

摘 要 网络应用服务(尤其是电子银行和电子商务)需要数据加密提供安全通信。很多应用服务器面临着执行大量计算稠密的加密挑战。CUDA(统一计算架构)是在GPU进行并行和通用计算的平台,能够利用现有显卡资源,以低成本的方式提升加密性能。在Nvidia GeForce G210显卡上实现CUDA的AES(高级加密标准)并行算法并且在AMD Athlon 7850上实现串行AES算法。实现的AES并行算法避免了同一线程块的线程同步和通信,提升了GPU的加速性能,加速比要比Manavski的AES-128并行算法提升2.66～3.34倍。在大数据量(至32MB)加密环境下探索AES并行算法的性能模型,并首次从加速效率角度分析加速性能。该并行AES算法在16核的GPU上能最高达到15.83倍的加速比和99.898%的加速效率。

关键词 统一计算架构,高级加密标准,并行,加速比,加速效率
中图法分类号 TP309.7 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.1.013

Implementation and Exploring of Acceleration Efficiency of Parallel AES Algorithm on CUDA

FEI Xiong-wei[1] LI Ken-li[2] YANG Wang-dong[1,2] DU Jia-yi[2]
(Department of Information Science and Engineering, Hunan City University, Yiyang 413000, China)[1]
(College of Computer Science and Engineering, Hunan University, Changsha 410008, China)[2]

Abstract Many services of network applications need data encryption to provide secure communication especially for e-bank and e-business. Massive application servers are challenging for intense computation of unlimited encryption. CUDA (Compute Unified Device Architecture) is a platform for parallel and general-purpose computation on GPU(Graphics processing unit). It can improve the performance of encryption with existed graphics device resource in a low cost manner. A Parallel AES (Advanced Encryption Standard) algorithm is implemented on a Nvidia Geforce G210 graphics card and a serial AES algorithm is implemented on AMD Athlon 7850. This parallel AES algorithm avoids synchronization and communication between threads in a same block and improves performance of acceleration completely. The parallel AES algorithm's performance model is explored in a enormous data(up to 32MB) environment. The parallel AES algorithm's speedup is 2.66 to 3.34 times larger than Manavski's AES-128 parallel algorithm. Acceleration performance of parallel AES algorithm was analyzed from the aspect of acceleration efficient for the first time. Speedup of parallel AES in a 16 cores GPU is up to 15.83 times accompanying a acceleration efficiency of 99.898%.

Keywords Compute unified device architecture, Advanced encryption standard, Parallel, Speedup ratio, Acceleration efficiency

1 引言

许多网络应用尤其是电子商务和网络银行,需要提供数据加密以保证通信安全。由于加密是一个计算稠密性问题,当用户访问量很大时,给服务器带来了沉重的负担;而且随着网络应用的不断发展和普及,需要进行加密以提供安全的应用越来越多。很多应用服务器一方面承受着巨大的加/解密压力,另一方面通过升级硬件得到的性能提升也很有限。因此能够利用现有的GPU硬件,以低成本的方式提升加密性能,有很大的现实意义。

AES加密算法的性能提升可以采用FPGA[1-4]或GPU[5]等方式来实现,但采用FPGA这类专用硬件设计的AES算法实现不够灵活,难以升级维护,而且开发周期长、开发成本高,还存在扩展性差等缺点。GPU则可弥补FPGA的不足。它是最近发展起来的具有并行计算功能的单片多核处理器,采用单指令多线程(SIMT)的体系结构,可实现密集的并行计算,可以满足AES高强度的加解密需求,此外提供了通用的软件开发环境——CUDA。

到稿日期:2013-12-26 返修日期:2014-03-15 本文受国家自然科学基金重点项目(61133005,61432005),益阳市科技计划项目(2014JZ37)资助。

费雄伟(1980－),男,博士生,讲师,主要研究领域为网络与信息安全、并行计算,E-mail:25616235@qq.com;**李肯立**(1971－),男,教授,博士生导师,CCF杰出会员,主要研究领域为并行计算、生物计算;**阳王东**(1974－),男,博士生,教授,主要研究领域为并行计算;**杜家宜**(1981－),男,博士生,主要研究领域为嵌入式计算、并行计算。

Cook 等人[6]第一次探索了使用 GPU 进行对称加密，并使用 OpenGL 在前一代 GPU 上实现了 AES。由于受到使用的图形编程模型的能力的限制，其中包括性能的提升限制和硬件可编程能力的限制，他们被迫使用固定功能的流水线，在输出汇聚阶段采用硬件完成异或。一次完整的 AES 执行需要通过多个流水线，这对性能造成了显著的影响。他们的实验结果显示，GPU 的性能仅比 CPU 的性能提升 2.3%。

Manavski 等人[7]使用 Nvidia GeForce 8800 显卡在 CUDA 平台上第一次实现对 AES 算法的加速，这得益于 G80 支持通用计算。他们采用的是 1024 字节为一明文块，每个明文块由一个线程块加密，线程块的大小是 256，即 4 个线程负责一个明文分组（16 字节）的计算。一个线程块的线程需要同步和通信，通信方式采用的是共享存储器，在 AES 加密的每轮均需要执行线程同步。很明显，共享存储器的体冲突（bank conflicts）和线程同步对性能有着不好的影响。此外，共享存储器的容量限制了一次并行加密的明文大小。对 AES-128，实验测试的明文数据大小为 2kB 到 8MB，得到的加速比在 4.74～5.92 之间稳定增加，但没有测试更大的数据。随着数据量的增加，数据传输会形成瓶颈，加速比的运行趋势需要进一步实验测试，以得到它的性能模型。

Bos 等人[8]采用 3 种不同的体系结构对 AES-128 进行加速，其中之一是 GPU 结构。他们第一次在 GPU 上同时实现了加密和解密。GPU 的实现方法使用的是 T 表法，并行地处理多个流来获得加密或解密结果。在 GTX 295 上运行该 AES 并行算法时，每字节加密和解密分别仅需 0.17 个和 0.19个周期。Tomoiagaa 等人[9]分别采用查找表和 GPU 计算能力计算每步操作（不用查找表）这两种方法实现了 AES，并且分别测试了加密内存数据和大尺寸的加密磁盘文件这两种不同的数据，得到的吞吐量是 11.95Gbps。加密小尺寸的内存数据时，带查找表的并行 AES 要快于不带查找表的并行 AES。加密大尺寸的硬盘文件时，文件小于 3GB 则带查找的并行 AES 性能较好；当文件大小在 4G～8G 间时，不带查找的并行 AES 性能较好；当文件大小大于 9G 时，带查找表的性能又要优于不带查找表的。为了达到 AES 的安全性和性能的均衡，Duta 等人[10]采用交替密码块链模式（ICBC）实现了并行的 AES 算法，分别采用 CUDA，OpenCL 和 OpenMP 进行并行处理，实验结果显示并行对串行有着很大的性能提升，其中 CUDA 对性能提升最好。这说明基于 CUDA 的并行加密算法具有更好的前景和应用。

国内学者在 AES 方面的进展有：夏辉等人[11]采用指令集架构（ISA）扩展的方式对 AES 密码算法进行优化，提出一个高效 AES 专用指令处理器（AES-ASIP）模型，最终实现于 FPGA 中。对比 ARM 处理器指令集架构，AES-ASIP 以增加少许硬件资源为代价，提高了算法 58.4x%的执行效率并节省了 47.4x%的指令代码存储空间。向涛等人[12]基于 AES 算法的设计原理提出了一种改进的医学图像加密算法，针对 AES 算法结合斜帐篷映射对其进行改进，使其适合 DICOM 医学图像的数据特点。首先将 AES 中 4×4 的分块操作方式变成 $M\times N$ 的全图操作，其次增加了对病人基本信息的保护，最后改进了 AES 中列混合操作与密钥编排方式。改进算法具有较好的置乱效果，扩散性强，并且能够很好地保持 DICOM 文件格式的兼容性，加密同一幅 DICOM 图像，改进算法所花费的时间比 AES 算法少一半左右。

基于以上背景，在 Nvidia GeForce G210 显卡上实现 CUDA 的 AES 并行算法，避免同一线程块的线程同步和通信，充分挖掘 GPU 的加速性能（最多线程数），在大数据量到达 32MB 加密环境下测试 CUDA 并行算法的性能模型，分析加速比和加速效率。通过对 G210 上的并行 AES 算法和 AMD Athlon 上的串行 AES 算法实际运行时间的实验，得到加速比的增长模型，并第一次从加速效率角度分析加速性能。

2 AES 算法

高级加密标准（Advance Encryption Standard，AES）[13]是一个对称密码，它是美国标准技术局（NIST）替换 DES 的新一代对称密码标准。AES 是一个分组密码，以 128 位为一分组，可以使用 128 位、192 位或 256 位密钥进行分组加密。AES 是一个完全对称的结构，使用异或、查找表和移位操作，能够在软件上实现。128 位明文分组被设计为执行多轮的迭代转换得到同样长的密文分组。轮数由密钥位数决定：128 位密钥需 10 轮，192 位密钥需 12 轮，而 256 轮密钥需 14 轮。除了首轮有一次轮密钥相加和最后一轮少一次列混淆外，其余每轮都是一样的处理方式。每轮加密的对象为一个 16 字节（128 位）的分组，可以看做 1 个 4×4 大小的二维数组（表），称为状态（state）。每轮的加密处理可分为 4 个阶段：轮密钥加（AddRoundKey，对轮密钥和状态进行异或运算，达到盲化效果）、字节替换（SubBytes，使用 Sbox 进行非线性替换）、行移位（ShiftRows，对每行分别进行 0，1，2，3 字节的循环移位，达到字内混淆）、列混淆（MixColumns，线性变换每列以混淆数据）。整个 AES-128 的加密流程如图 1 所示。

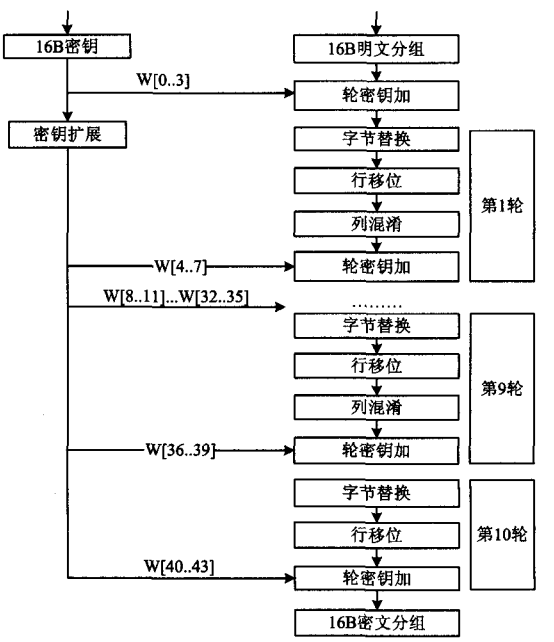

图 1 AES 的 128 位版本加密流程

AES 算法的分组加密结构具有并行性。明文分组之间不存在数据依赖，加密时按明文分组进行并行加密，能显著提高加密效率，特别是对大容量的数据。如图 1 所示，16B 的密

钥扩展为 11 个 16B 的密钥分组。第 1 轮之前的轮密钥加使用的是密钥的本身值 w[0..3],表示密钥的前 4 个字。后面的 10 轮做轮密钥加时使用的是由原密钥扩展成的 10 个 4 字组,从 w[4..7]到 w[40..43]。这些密钥分组在每次加密时都固定,故扩展密钥的部分设计为在 CPU 上串行执行 1 次。这样每轮访问密钥都没有数据依赖关系。置换表为 16×16 字节的二维表,值保持不变,每轮均按此表进行替换。行移位对第 i 行循环左移 i 个字节($0 \leqslant i < 4$)。列混淆采用的是有限域 GF(2^8)上的乘法,数据针对的是状态(state)。可见密钥扩展后,每个分组之间的加密过程都不存在数据依赖,适合并行程序设计。

3 CUDA 模型及并行算法设计

3.1 CUDA 模型

CUDA(Compute Unified Device Architecture)即统一计算架构[14],由 Nvidia 公司为它的 G80,G92 和 GT200 等图形处理器提出。CUDA 提供了一个能集成主机和 GPU 代码到同一个 C/C++源文件的编程模型,降低了开发的难度。它调用一个函数(该函数称为核函数(kernel)),来支持并行计算。核函数由 CPU 调用而由设备(GPU)执行。CUDA 使用块(block)和线程(thread)的概念来表示算法的并行性。块是一组并发的线程单位,能独立执行。块与块之间的执行顺序可以是并行的也可以是串行的。块内线程是并行执行的,有时需要进行同步(通过调用__syncthreads),有时需要通信(使用共享存储器(shared memory))。块之间的通信必须使用全局存储器(global memory)。核心的执行在调用时通过 kernal〈〈〈blocks,threads〉〉〉(parameters)修饰符提供块数量和每块的线程数,其中 blocks 表示块大小而 threads 表示每块线程数,parameters 表示函数的参数列表。对计算能力为 1.1 的 GPU 而言,每块的线程数最多为 512,对计算能力为 1.2 及以上的 GPU,每块允许的线程数可达 1024。

3.2 并行算法设计

AES 的并行算法设计为对每个分组的加密部分在核函数上并行处理。密钥扩展在 CPU 上执行一次,生成 11 组 4×4字节的密钥分组,存储于全局存储区。具体代码如图 2 所示。

```
AES aes(key);//构造函数完成初始化及扩展密钥
unsigned char * dkey;//定义设备密钥指针
cudaMalloc(&dkey,16 * 11);//分配设备密钥空间
cudaMemcpy(dkey,aes. w,16 * 11,cudaMemcpyHostToDevice);//复制密钥
```

图 2 密钥扩展及传输代码图

同样,对于 S 盒(Sbox)的处理也是存储于全局存储区。对于明文和密文的处理是:由于 1 个明文分组对应 1 个密文分组,随机生成的明文数据加密后覆盖存储于同一存储区。具体代码如图 3 所示。

```
unsigned char * input, * dinput;//分别定义明文指针
input=(unsigned char * )malloc(size);//在主机上为明文数据分配 size 字节的空间
cudaMalloc(&dinput,size);//在设备上为明文数据分配 size 字节的空间
for(int i=0;i<size;i++)
input[i]=rand()%128;//随机生成明文数据
cudaMemcpy(dinput,input,size,cudaMemcpyHostToDevice);//传输明文至设备
```

图 3 明文生成、存储于传输代码图

核函数的调用需要提供块数和线程数,这与设备的计算能力有关。G210 的计算能力为 1.1,块的最大维度 x/y 均为 512,z 为 64,块内最大线程数为 512。网格(Grid)最大数目为 65535。G210 中每个流多处理器(SM)最多可以同时管理 768 个线程或 8 个块。每一个线程束(warp)的大小是 32 个线程,也就是一个 SM 最多可以有 768/32=24 个线程束。因此,块内的线程数目可以选择 96(一个 SM 中有 8 个 block)、128(一个 SM 中有 6 个 block)、192(一个 SM 中有 4 个 block)或 256(一个 SM 中有 3 个 block)。512 个线程数由于 8192 个寄存器数目的限制,不能用在这个并行 AES 中。为了达到最好的加速性能,采用 256 个线程每块。支持块的大小(256)和网格的大小定义如下:int BK=256;Dim3 dmblock(BK);Dim3 dmgrid(size/16/BK)。核函数的调用方式为:kernel〈〈〈dmgrid,dmblock〉〉〉(dinput,dkey,dbox),dmgrid 表示网格的三维数据,dmblock 表示块的三维结构,dinput 表示明文,dkey 表示密钥,dbox 表示 S 盒(Sbox)。核函数提供对分组的加密处理,由多个块并行执行。具体代码如图 4 所示。

```
__global__ void kernal(unsigned char * input,unsigned char * key,
unsigned char * Sbox)
{unsigned char state[4][4];
int tid=threadIdx. x+blockDim. x * blockIdx. x;//计算线程 id
getState(state,&input[tid * 16]);//读取明文分组至状态,16 字节 1 分组
AddRoundKey(state,&key[0]);//单独的轮密钥相加
for(int i=1; i<=10; i++)
{SubBytes(state);//字节替换
ShiftRows(state);//行移位
if(i!=10)
MixColumns(state);//列混淆
AddRoundKey(state,key[i * 16]);//轮密钥相加
setState(state,&input[tid * 16]);}//存储密文分组
```

图 4 核函数代码图

4 实验及讨论

4.1 实验环境

(1)硬件设备

GPU 的参数为:Nvidia Geforce G210,512MB DDR2 存储器,8.1GB/s 带宽,时钟频率为 550MHz。

CPU 的参数为:AMD Athlon 7850 Black Edition,2.81GHz主频,2G DDR2 存储器。

(2)软件平台:WINDOWS XP SP3,Visual Studio 2008,CUDA 4.2。

4.2 实验结果

实验结果取同样明文规模的 AES 并行加密算法和串行加密算法的实际运行时间。每种明文大小的算法均运行了 3

次，取中值，以减少数据的误差。具体实验数据如表 1 所列。

表 1　并行 AES 加密算法和串行 AES 加密算法性能比较表
（单位：ms）

明文大小	串行执行时间	并行加密执行时间	并行 AES 执行总时间	加速比	加速效率	并行密钥扩展及传输时间	并行 AES 执行总时间
16B	0.061	0.005	0.805	0.076	0.005	0.032	0.772
32B	0.115	0.007	0.772	0.149	0.009	0.024	0.748
64B	0.222	0.006	0.778	0.286	0.018	0.023	0.755
128B	0.431	0.006	0.779	0.553	0.035	0.023	0.757
256B	0.874	0.006	0.799	1.094	0.068	0.022	0.777
512B	1.745	0.006	0.836	2.087	0.130	0.022	0.814
1kB	3.479	0.006	0.846	4.112	0.257	0.022	0.824
2kB	6.855	0.012	1.105	6.206	0.388	0.225	0.879
4kB	14.052	0.012	1.174	11.97	0.748	0.030	1.144
8kB	28.440	0.011	1.821	15.614	0.976	0.029	1.792
16kB	56.952	0.018	3.802	14.981	0.936	0.232	3.569
32kB	113.2	0.026	7.478	15.137	0.946	0.239	7.239
64kB	226.97	0.022	14.65	15.489	0.968	0.238	14.415
128kB	458.15	0.025	29.050	15.771	0.986	0.237	28.813
256kB	913.06	0.025	57.822	15.791	0.987	0.238	57.584
512kB	1827.21	0.027	115.368	15.838	0.990	0.236	115.132
1MB	3629.85	0.027	231.448	15.683	0.980	0.239	231.210
2MB	7254.41	0.030	462.064	15.700	0.981	0.239	461.83
4MB	14487.87	0.032	925.29	15.66	0.979	0.237	925.06
8MB	28981.89	0.032	1843.4	15.72	0.983	0.236	1843.17
16MB	58038.09	0.033	3688.09	15.74	0.984	0.238	3687.85
32MB	115990.45	0.034	7377.37	15.72	0.983	0.269	7377.1

4.3　实验分析与讨论

在表 1 中，实验数据完整地给出了明文大小从 64B 到 32MB 的并行 AES 和串行 AES 的时间和性能比较情况。串行 AES 执行总时间 *Ts-all* 表示串行 AES 算法在 AMD Athlon 7850 上的加密时间。并行 AES 执行总时间 *Tp-all* 表示在 Nvidia Geforce G210 上执行并行 AES 算法的总时间，包括密钥扩展及传输时间 *Tkpt* 和并行数据加密及传输明密文时间 *Tloadstore*，计算公式如式(1)所示：

$$Tp\text{-}all = Tkpt + Tloadstore \tag{1}$$

加速比表示并行 AES 相比串行 AES 的性能提升，计算公式如式(2)所示：

$$Sp = Ts\text{-}all / Tp\text{-}all \tag{2}$$

加速效率 *Ae* 表示并行程序提升加速比的性能，由加速比和 GPU 的核心数 *C*(G210 为 $C=16$)之比表示，计算公式如式(3)所示。

$$Ae = Sp / C \tag{3}$$

随着明文大小成倍增加，串行 AES 的计算时间也成倍增加，而并行 AES 的执行总时间以缓慢的速度增长，如图 5 所示。明文大小在 16B～128B 之间，并行 AES 算法所化时间要比串行 AES 时间多，因为线程数目在 1～16 之间，无法隐藏开销的延时，也说明 1 个线程的 GPU 并行 AES 要远远慢于单线程的 CPU 串行 AES。无论并行 AES 还是串行 AES，峰值加密时间均出现在明文大小为 32MB 时。并行 AES 算法中有密钥扩展和密钥及一些表的传输，只需在 CPU 上执行一次。从表 1 可看出，这部分时间每次几乎相等。并行 AES 算法中的负载总时间 *Tloadstore*(包括明文加密、明文传进 CPU 和密文传出时间)，在明文≤8kB 时增长缓慢，之后成倍地增长。这说明短明文传输开销较小，而明文增大后(>8kB)，传输开销越来越大，这与主存缓存的命中率下降有关。

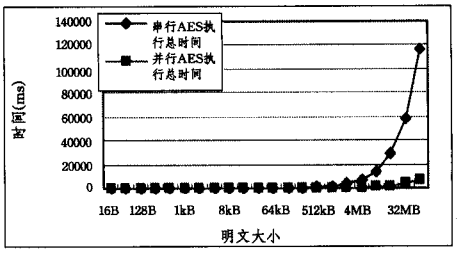

图 5　并行 AES 和串行 AES 的执行时间比较

如表 1 所列，并行 AES 的加速比明文＜1kB 时按倍数增加，明文在 1kB 到 4kB 之间呈现线性增长，而明文从 8kB 到 32MB 的加速比保持稳定，不再增长。峰值加速比出现在明文为 512kB 时，达到了 15.83816458 倍。加速比和加速效率的增长趋势，如图 6 所示。从明文大小为 64B～32MB 综合来看，整个并行 AES 算法性能模型包含 3 个阶段，分别是指数增长、线性增长和稳定阶段。

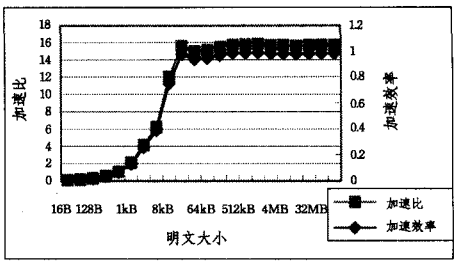

图 6　并行 AES 算法的加速比和加速效率

不同 GPU 的核心数目不同，带来的加速性能的评价不能单纯地从加速比判断。为了消除不同的 GPU 给并行算法带来的加速比差异性，引入加速效率的评价标准。加速效率的计算如式(3)所示。实验得出的加速效率峰值为 0.989885286，非常接近 1；8kB 以上的明文加速效率达到了 94%以上且较为稳定。说明该并行算法充分挖掘了 G210 多核心计算能力，非常适宜处理 8kB 及以上的 AES 加密。而典型的一次 SSL 会话的传输对象的大小为 35kB(只含 HTML 内容)到 150kB(一次 HTTP1.1 会话发送的文本和图像内容)之间[15]。这说明使用当前普通的显卡带来的加速效率能够对 SSL 提供很好的加密支持。

相比 Manavski[7] 的 AES-128 并行算法，所提算法性能有 2.66～3.34 倍的提升，具体比较如表 2 所列。

表 2　与 Manavski 并行 AES 的加速比比较

明文大小	Manavski[7] 的并行 AES 加速比	本文并行 AES 加速比	性能提升倍数
512B	4.74	15.84	3.34
1M	5.43	15.68	2.89
4M	5.69	15.66	2.75
8M	5.92	15.72	2.66

结束语　对实现的并行 AES 算法按明文大小从 64B 到 32MB 充分探索了并行的性能模型。性能模型分为 3 个阶段：指数增长、线性增长和稳定阶段。由于采用 256 线程 1 块，1 线程加密 1 明文分组，避免了线程通信和同步开销，提升了加密的性能；也不受共享存储器容量限制，支持大容量达到 32MB 的明文加密。得到的性能加速比要比 Manavski[7] 的 AES-128 并行算法性能提升 2.66～3.34 倍。第一次从加

(下转第 74 页)

测可以预知下一个阶段的资源瓶颈，能更有针对性地提出资源重构的建议。程序阶段性分析在调度算法中的资源分配应用也是一个值得研究的问题，试想在有多个不同处理器核的动态异构处理器上调度进程，如何根据阶段性技术和资源瓶颈分析进行资源分配优化也需要更多的实验工作。

参 考 文 献

[1] Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manualys[OL]. http://www.intel.com/content/www.us/en/processors/architectures-software-developer-manuals.html

[2] Kalla R, Sinharoy B, Starke W J, et al. Power7: IBM's Next-Generation Server Processor[J]. IEEE Micro, 2010, 30(2): 7-15

[3] Ipek E, Kirman M, Kirman N, et al. Core fusion: accommodating software diversity in chip multiprocessors[C]// Proceedings of the 34th annual international symposium onComputer architecture. Dean Tullsen ed. San Diego, California, USA: ACM, 2007: 186-197

[4] Kim C, Sethumadhavan S, Govindan M S, et al. Composable Lightweight Processors[C]// Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, 2007: 381-394

[5] Watanabe Y, Davis J D, Wood D A. WiDGET: Wisconsin decoupled grid execution tiles[C]// Proceedings of the 37th Annual International Symposium on Computer Architecture. André SeznecedSaint-Malo, France, ACM, 2010: 2-13

[6] Sherwood T, Sair S, Calder B. Phase tracking and prediction[C]// Proceedings of the 30th Annual International Symposium on Computer Architecture. New York, NY, USA, ACM, 2003: 336-349

[7] Sherwood T, Perelman E, Calder B. Basic block distribution analysis to find periodic behavior and simulation[C]// Proceedings 2001 International Conference on Parallel Architectures and Compilation Techniques. Barcelona, Catalunya, Spain, IEEE Computer Society, 2001: 3-14

[8] Sherwood T, Perelman E, Hamerly G, et al. Automatically characterizing large scale program behavior[C]// Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems. New York, NY, USA, ACM, 2002: 45-57

[9] Dhodapkar A S, Smith J E. Comparing Program Phase Detection Techniques[C]// Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture. IEEE Computer Society, Washington DC, USA, 2003: 217-228

[10] Lau J, Schoenmackers S, Calder B. Transition Phase Classification and Prediction[C]// Proceedings of the 11th International Symposium on High-Performance Computer Architecture. IEEE Computer Society, Washington DC, USA, 2005: 278-289

(上接第 62 页)

速效率角度分析加速性能，实现的并行 AES 算法在 16 核的 Nvidia Geforce G210 上能最高达到 15.83 倍的加速比和 99.898%的加速效率。使用当前普通的显卡(如 Nvidia Geforce G210)加密 SSL 会话数据(大小 35kB～150kB)能提供非常高(>94%)的加速效率。

参 考 文 献

[1] Wang Y, Ha Y. FPGA-Based 40.9-Gbits/s Masked AES With Area Optimization for Storage Area Network[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2013, 60(1): 36-40

[2] Mathai A, Sathyanarayana M. Design of Area Optimized AES 128 Algorithm Using Mixcolumn Transformation[J]. International Journal of Innovative Research and Development, 2013, 2(7): 173-176

[3] Prathyusha C, Rani P S. Implementation of Fast Pipelined AES Algorithm on Xilinx FPGA[J]. International Journal of Science and Research, 2013, 2(8): 377-381

[4] Granado-Criado J M, Vega-Rodríguez M A, Sánchez-Pérez J M, et al. A new methodology to implement the AES algorithm using partial and dynamic reconfiguration[J]. INTEGRATION, the VLSI journal, 2010, 43(1): 72-80

[5] Nhat-Phuong T, Myungho L E E, Sugwon H, et al. High Throughput Parallelization of AES-CTR Algorithm[J]. IEICE TRANSACTIONS on Information and Systems, 2013, 96(8): 1685-1695

[6] Cook D, Keromytis A D. Cryptographics: Exploiting Graphics Cards for Security (Advances in Information Security)[M]. Springer, 2006: 78-96

[7] Manavski S A. CUDA compatible GPU as an efficient hardware accelerator for AES cryptography[C]// IEEE International Conference on Signal Processing and Communications. IEEE, 2007: 65-68

[8] Bos J W, Osvik D A, Stefan D. Fast Implementations of AES on Various Platforms[C]// IACR Cryptology ePrint Archive. 2009, 2009: 501

[9] Tomoiagaa R D, Stratulat M. Accelerating Solution Proposal of AES Using a Graphic Processor[J]. Advances in Electrical and Computer Engineering, 2011, 11(4): 99-104

[10] Duta C L, Michiu G, Stoica S, et al. Accelerating Encryption Algorithms Using Parallelism[C]// IEEE International Conference on Control Systems and Computer Science (CSCS). 2013: 549-554

[11] 夏辉，贾智平，张峰，等. AES 专用指令处理器的研究与实现[J]. 计算机研究与发展，2011，48(8)：1554-1562

[12] 向涛，余晨韵，屈晋宇. 基于改进 AES 加密算法的 DICOM 医学图像安全性研究[J]. 电子学报，2012，40(2)：406-411

[13] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard[OL]. http://www.itl.nist.gov/fipspubs/, Nov. 2001

[14] NVIDIA Corporation. CUDA Technology[OL]. http://www.nvidia.com/CUDA, Sep. 2008

[15] Levering R, Cutler M. The portrait of a common HTML web page[C]// Proceedings of the 2006 ACM symposium on Document engineering. 2006: 198-204