Augmenting Operating System with GPU

By Anusha ,Tabish ,Haseeb ,Tauqeer

WORD COUNT 1212
TIME SUBMITTED 26-NOV-2018 09:34AM
PAPER ID 42252971

Project Report

Title: Augmenting Operating System with GPU

Group Members:

1) Anusha (60295)

2) Tabish (59237)

3) M. Haseeb Hannan (59394)

4) M. Tauqeer Imam (59383)

METHODOLOGY: METHODOLOGY: (Keywords: CUDA-enabled graphics processing unit for general purpose processing)

There were basic and foremost important tasks planned for parallelism of GPUS working with Kernel/CPUS for **improvement of Operating system's Performance, Efficiency, Functionality and Security**. As increased focus on GPUS help because of their cost-effective behavior.

1)To reduce tasks which are executed repeatedly and quickly on GPUs than on CPUs (Saving space) while Speeding Executions

2)To increase output for operations like huge number of client server handlings

3)To make compatibility of advance functionalities in OS Kernel that are too slow while running on CPU.

Network Packet Processing: (Keyword: Packet Shader is high-performance PC-based software router platform that accelerates the core packet processing with Graphics Processing Units (GPUs))

These brilliant GPUs performance for packet processing and software routing are enhanced remarkably. For most quick routing table exchanges were achieved by packet shader (approx. rate for both IPV4 and IPV6 Table forwarding 40Gbps) and at most 4x speedup over the CPU-only mode using two NVIDIA GTX 480 GPUs. For IPSec, Packet Shader gets a 3.5x speedup over the CPU.
Not only that but GPU's has its involvement in SSL implementation too through SSL Shader. It runs 4x faster than equivalent CPUS. Packet Shader shows its efficiency but it costs high round trip inactivity/latency for each packet when compared with just CPUs.

This revealed the **weakness** of the GPU in a latency-oriented computing model: **the overhead caused by copying data and code into GPU memory then copying results back affects the general reaction time of a GPU computing task severely.** To deal with this offloading latency problem. KGPU prototype is proposed this prototype decreases the latency of GPU computing tasks

In-Kernel Cryptography: (Keyword: A TPM is traditionally hardware, but recent software implementations of the TPM specification, such as vTPM, are developed for hypervisors to provide trusted computing in virtualized environments where virtual machines cannot access the host TPM directly.)

GPUS are best at speeding up computations faster than CPUS to utilize them in Block cryptography AE lot of computations are required this is covered below in AES Example that proposed KGPU implements AES on the GPU for the Linux Kernel. blocks of information will represent either giant blocks of one task or variety of smaller blocks of various tasks. but Thus, the GPU can't solely speed up bulk encoding however additionally rescale the quantity of synchronic uses of the cryptography scheme, Cryptographic operations augmented by GPUs are shown to be possible and to urge vital speedup over processor versions. OS practicality creating significant use of cryptography includes IPSec. File system encryption, and content-based information redundancy reduction of filesystem blocks and memory pages. Another potential application of the GPU-accelerated cryptography is trusted computing supported the trusty Platform Module (TPM).

Pattern Matching Based Tasks:

The GPU will accelerate regular expression matching, with speedups of up to 48 times compared with CPU. A network intrusion detection system (NIDS) with GPU-accelerated regular expression matching incontestable **a fairly 60% increase in overall packet process output on an old GPU hardware.**

Different tasks like information flow, controls within the OS, virus detection (with 2 orders of magnitude speedup), rule-based firewalls, and content-based search in filesystems will doubtless profit from GPU-accelerated pattern matching.

KGPU ARCHITECTURE:

GPU is divided into three parts:

1. A module in the OS Kernel.
2. A user-space helper process.
3. NSK running on GPU.

GPU functions on the OS Kernel follows the steps:

a. The pinned-memory buffers and fills with there input that also requests a buffer for the result.
b. It builds a service request. Services are CUDA programs that have been per-load into NSK. Fe launch minimize time and that include a completion call back.
c. By the service request into request queue.
d. It will wait for the request to complete or either blocking until completion callback is called or busy-waiting on the response queue.

By the helper KGPU view the request queue in memory shared with the OS Kernel. A new service request comes the DMAs the input data buffer to the GPU using CUDA APIs. DMA completes the helper sends service request message to NSK using the message passing mechanism. The NSK sends completion message to the CPU side and resumes polling for new request message. The OS kernel through their shared response queue that avoid copy between the kernel module and the user- space helper, the pinned data buffers allocated by CUDA driver are shared between two. The data of buffers locked in physical memory for manage it carefully.

On the CPU side buffers used for different purposes:

1. Preparing for a future service call by accepting data from a caller in the OS kernel.
2. To DMA input data from main memory to the GPU for the next service call.
3. To DMA results from the last service call from GPU memory to main memory.
4. Finishing a previous service call by returning data to the caller in the OS kernel.

Each performance will be done concurrently so along with the service currently running on the GPU the total depth of the service call pipeline is five stages. In the current KGPU prototype, we statically allocate four buffers, and each changes its purpose over time.

Example: A GPU AES Implementation

The AES algorithm is currently the standard block-cipher algorithm that has replaced the Data Encryption Standard (DES). Back in 1997 the National Institute of Standards and Technology (NIST) made a public call for new cipher algorithms that could replace the DES. A rough summary of the requirements made by NIST for the new AES were the following:

- Symmetric-key cipher
- Block cipher
- Support for 128-bit block sizes
- Support for 128-, 192-, and 256-bit key lengths

The process is relatively simple, but some brief cryptographic explanations are necessary to understand what is going on. In cryptography, algorithms such as AES are called *product ciphers*. For this class of ciphers, encryption is done in *rounds*, where each round's processing is accomplished using the same logic.

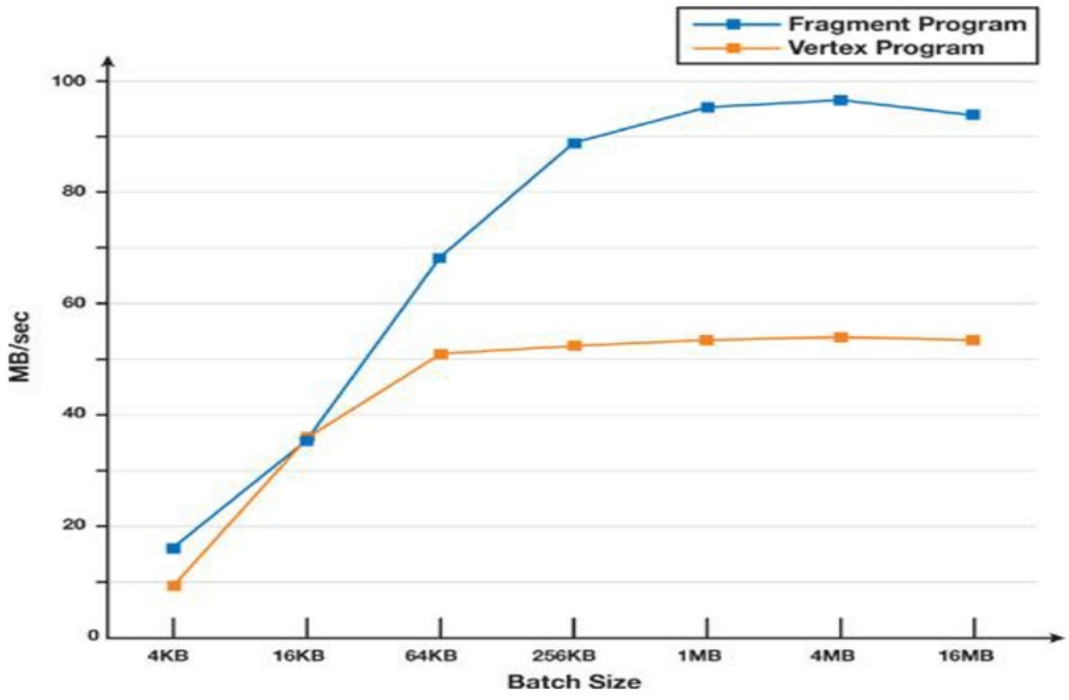
Now that we have a working AES implementation, let us measure the performance of GPU-based encryption. The decryption is omitted because it performs the same as the encryption in the AES algorithm. Our tests were performed on a test machine with the following specifications:

- CPU: Pentium 4, 3 GHz, 2 MB Level 2 cache
- Memory: 1 GB
- Video: GeForce 8800 GTS 640 MB
- System: Linux 2.6, Driver 97.46

Vertex Program vs. Fragment Program:

We have compared the performance of the vertex program in the transform feedback mode pipeline with that of the fragment program in the traditional rendering pipeline. The fragment program is the same code as the vertex program except that input/output registers were redefined appropriately, exploiting the GPU's unified architecture.

Our results were obtained by processing a *plaintext* of 128 MB filled with random numbers and averaging measurements from ten runs. As illustrated in below shown chart, the throughput for the vertex program is 53 MB/sec, whereas for the fragment program, the throughput is 95 MB/sec with a batch size of 1 MB. Our implementation spends most of its processing time in referencing tables. In other words, fetching textures.



Augmenting Operating System with GPU

ORIGINALITY REPORT

66%

SIMILARITY INDEX

PRIMARY SOURCES

1	www2.cs.utah.edu Internet	455 words — 36%
2	http.developer.nvidia.com Internet	319 words — 26%
3	www.cs.utah.edu Internet	36 words — 3%
4	shader.kaist.edu Internet	18 words — 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		