In His Exalted Name

Astor: A Practical Parallel Antivirus Engine

Ahmad Siavashi
E-Mail: a.siavosh@yahoo.com
Department of Computer Science and Engineering
School of Engineering, Shiraz University, Shiraz, Iran

Submitted to—
Prof. Farshad Khunjush
Department of Computer Science and Engineering
School of Engineering, Shiraz University, Shiraz, Iran

Fall 2013

Contents

Executive Summary

I plan to modify ClamAV – the most popular open-source antivirus – and design a parallel antivirus engine. According to a known research I can get up to 100x speed-up. But unlike the previous research my purpose is to use this new engine in practice as well as targeting heterogeneous systems which may have several different processors.

While using all those processors is a great opportunity, it makes the implementation harder as the engine may run on different architectures. As we're not targeting a specific architecture, so it might seem impossible to get the best possible performance on all machines.

Thus we have to find ways and consider them in our program to get the best average performance. Of course, extracting methods to get the best average performance in heterogeneous systems will be useful in other practical projects as well.

Statement of Problem

An antivirus can operate faster using GPGPU. A work in this field[1] shows we can achieve 100X speed up. To get this performance, as it's more common, we might use CUDA for GPGPU. So far everything is okay since our purpose is research. But suppose we decide to use these results in practice. As CUDA is not available on all systems which we might install our antivirus on, we may face significant problems if the software couldn't run, since antivirus is a critical application. But what if it'd be able to run on all systems?

To solve this problem we might use OpenCL, since it'll run on all major commodity processors today. Nonetheless what should we do if there was no OpenCL capable GPU, but an OpenCL capable CPU? What if that CPU didn't have OpenCL runtime installed? Or what if we didn't have any OpenCL capable processor at all?

These are questions that I believe I have answer for them.

[1] GrAVity: A Massively Parallel Antivirus Engine
Giorgos Vasiliadis and Sotiris Ioannidis
Institute of Computer Science, Foundation for Research and Technology – Hellas,
N. Plastira 100, Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

Design Objectives

This document proposes the following objectives:

(1) Developing mechanisms for achieving the best average performance on different OpenCL capable heterogonous systems.

(2) Developing methods to run a single program on both OpenCL capable and non-OpenCL capable systems.

(3) Modifying ClamAV antivirus to design a massively parallel antivirus engine that can be used in practice.

It's obvious that when we are targeting a vast domain of processors we can't achieve the best performance on all of them. So we need to know what to consider in our program in order to get the best average performance on different heterogonous systems.

In addition, we know that our program might run on processors (Both GPUs and CPUs) produced before 2008 or 2009 which are beyond OpenCL existence date. Thus we need to apply a method to deal with these problems. As I've researched it's not impossible and it can be achieved.

The third objective, as mentioned, has been already achieved using CUDA but not published publicly. A 100X speed-up has been reported on GeForce 200 Series architecture.

Technical Approach

For the first and second objectives my plan is to target all OpenCL capable GPUs in all available platforms on the system. In this state we will get the highest performance. I can't say any number as it'll vary on different architectures; However It might be a number near the mentioned paper's number, about 100X speed up. We'll probably need the CPU off-load in this situation to do other tasks.

Suppose there were no GPUs but an OpenCL capable CPU. In this case I'll run the code on the CPU. In the worst case the OpenCL runtime can assure about 2X speed up. So far we don't need any change in the OpenCL kernel's code. We just simply change the device which runs the kernel and OpenCL runtime will handle everything for us.

Consider a situation where we have an OpenCL capable CPU but there is no OpenCL driver or OpenCL runtime on the system. Still we can run our OpenCL kernel if we embed an OpenCL dynamic loading library. We might face errors in this case but we can handle them.

Since the CPU component of OpenCL bundled with the AMD APP SDK works with any x86 CPU with SSE3 or later, as well as SSE2.x or later, it is less probable that we may not find an OpenCL capable CPU in the system but in case there was no OpenCL capable device in the system, we simply run the code which was in use in past with modifications

to optimize it for multicore systems. This can be done by using standard threading libraries like POSIX Threads or taking advantage of SSE extensions.

Conclusion

Despite this fact that designing a practical parallel antivirus engine might be an achievement in the war against malwares, gaining the experience to run a single program on a wide range of architectures with the best average performance makes this project valuable. It’s expected that this project will attain the attention of APA research lab, security labs as well as any organization, company or individual that wants to use the experience of running a single program on heterogonous systems.