ATBASH Encoding in Parallel

A Comparison of OpenCL, CUDA, and Python

Submitted by Alexander Stein

EECSE4750
Dr. Zoran Kostic
Assignment 1
Columbia University
October 7th, 2016

Problem Statement:

ATBASH is a simple cipher where the each letter in the chosen alphabet is encrypted as its corresponding letter in the same alphabet *backwards*. So for this assignment, using the capital English alphabet we have:

Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext: ZYXWVUTSRQPONMLKJIHGFEDCBA

The student is to implement this encoding scheme using python, OpenCL, and CUDA; then to compare the results for varying lengths of input strings. It is assumed that only English, uppercase, ASCII characters will be used.

Platform Information:

GPU: Nvidia GTX 660M
CPU: Intel Core i7 3630QM @ 2.4GHz
OS: Ubuntu 14.04 LTS
RAM: 24 GB SODIMM DDR3 Synchronous 1333 MHz (0.8 ns)

Overall Structure Pseudocode:

```
Function ENCODE(input_string):

    For (Python):
        Implement Atbash function pythonically

    For (OpenCL, CUDA):
        Obtain Platform and Device Information
        Create Context
        Build C Kernel
```

```
        Queue Commands
        Allocate device memory and transfer from host
        Specify desired hardware dimensions on device
        Run Program
        Copy outputs from device back to host

Function MAIN()
    For (i = 1 to N)
        Launch ENCODE(random input string size i)
        Store Results
    Plot Results
```

Figure 1 : Python vs. PyOpenCL

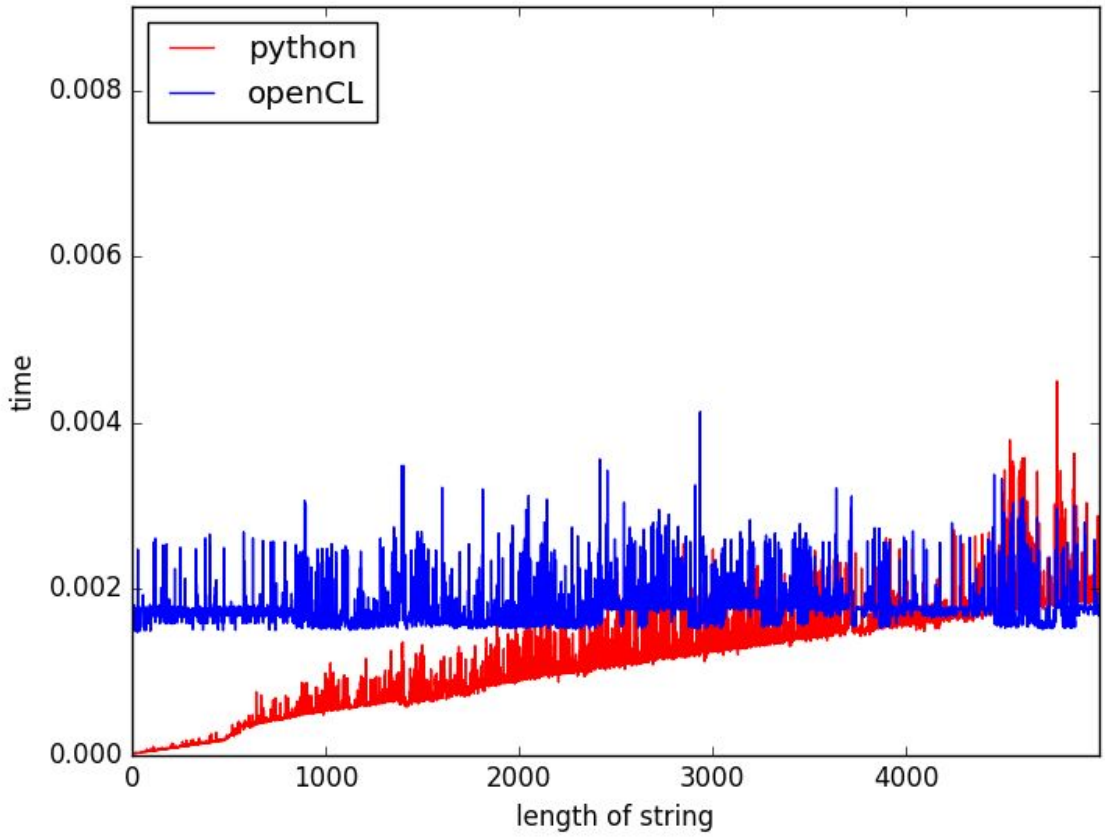


Figure 2 : Python vs. PyCUDA

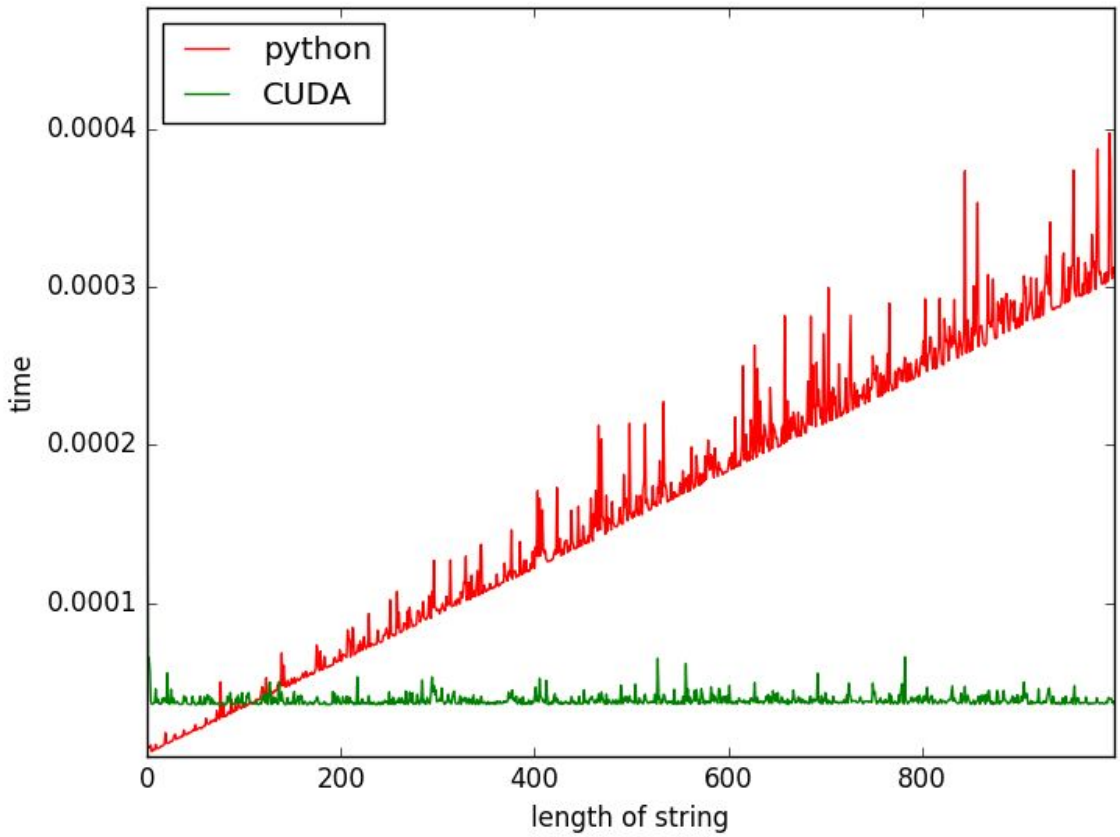
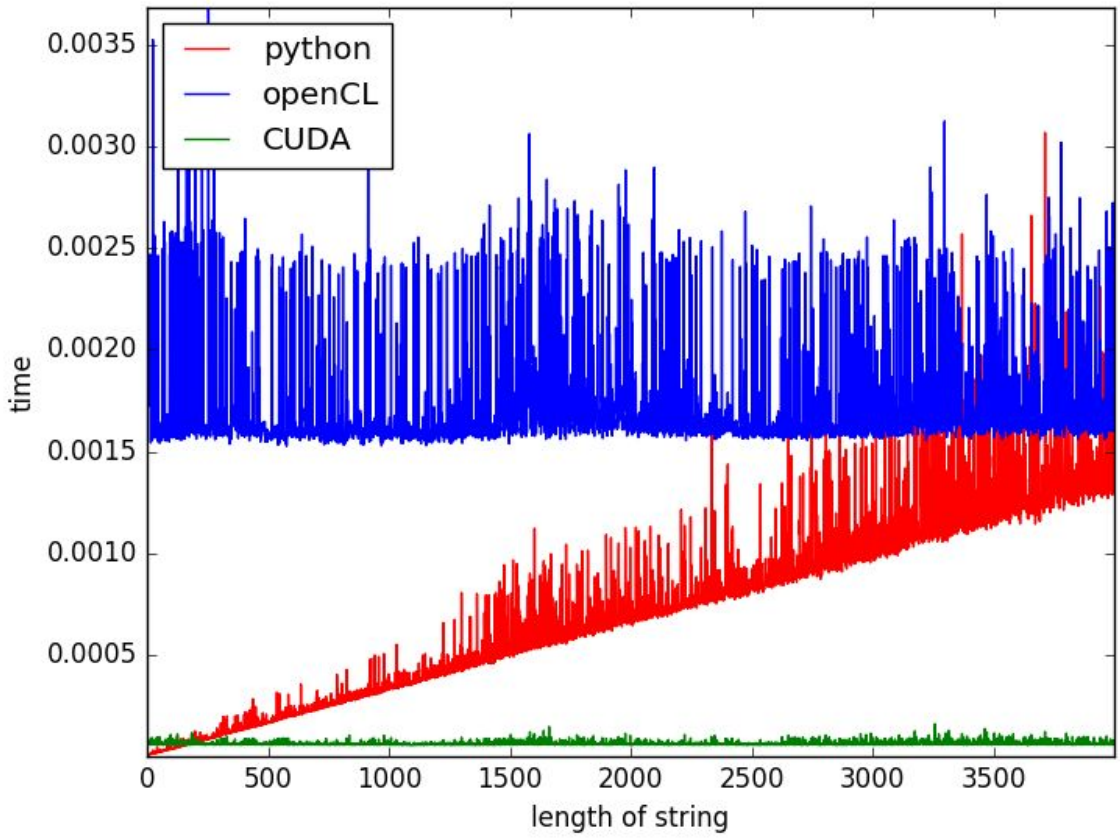


Figure 3 : Python vs. PyCUDA vs. PyOpenCL



Results: Analysis

Each of figures 1-3 above show clearly that the time taken to execute the Atbash cipher in Python (on the host CPU) increases linearly in time with the size of the input string, while the execution of the same function takes no additional time with that increase. Comparing Python with CUDA, we can see that for strings of length > 100, CUDA yields the best performance. Comparing Python with OpenCL, we see that for strings of length > 4000, OpenCL yields the best performance. It is likely that CUDA is performing better than OpenCL because the platform used in this instance is an Nvidia GPU, for which CUDA is optimized. Also Nvidia does not support OpenCL past version 1.1, which may exclude many optimizations. Regardless, the benefits of parallelism are clear in both cases.

Theory Questions

1. What is the difference between a thread, a task, and a process?

    a. A single process can launch multiple threads, each of which can in turn launch multiple tasks. Each process is assigned its own virtual memory space by the operating system, and -- though possible -- it is very difficult to share memory between two processes. Multiple parallel threads can exist within the same memory context of a single process, and sharing memory between them is simple and expected behavior. A task is simply an instruction or set of instructions which have been queued for execution in a device's memory, launched by a thread from within a process.

2. What are the differences between concurrency and parallelism?
    a. Experts generally agree that it is safe to use the terms "concurrency" and "parallelism" interchangeably w.r.t. computing. Still, 'concurrency' could imply that two potentially independent and different tasks are executing during an overlapping time period, as in multi-core CPU processing. Parallelism, on the other hand, could imply that two very similar tasks are executing and finishing at (almost) the exact same time, as in GPU vector addition for example.

3. Why not replace GPU with CPU?

    a. GPUs have a few disadvantages: they operate at slow frequencies; they have slow latency for each instruction; they do not easily support some commonly known programming techniques like branching and branch prediction; they have very small on-chip caches; programming GPUs requires device specific knowledge and therefore can make portability difficult. CPUs have many good qualities: they support complex branch prediction schemes; out of order execution and dependencies; operate at high frequencies; have large on-chip caches; have extremely fast latency per each instruction. However, Moore's Law is slowing down - limiting increases in chip speed. This has driven CPU producers to introduce multi-core (-8) parallelism to continue increasing performance. Power consumption is also an increasingly important consideration. GPUs hide their low-latency execution with massive parallelism (potentially millions of parallel tasks), coming hand-in-hand with their lower-power ALUs and smaller cache sizes. Furthermore, Big Data and scientific type processing are executed much faster on GPUs because they are conducive to SIMD type programs. CPUs will long have their place in the computing

landscape to handle serial non-parallelizable code, but I still think the pertinent question to ask would be “Why not replace CPUs with GPUs?”

4. What are the advantages and disadvantages of using Python over C/C++?

    a. From a very high level, Python is much more “user-friendly” than C/C++, because it does not require strict syntax and cumbersome headers. Things like explicit memory allocation, dealing with pointers, cleaning up memory leaks, are taken care of for you by Python. Furthermore, there are myriad python libraries available from the open-source community which can accomplish almost every low-level task thinkable. C/C++ though, have the advantage of performing much faster than Python, and give explicit control over memory allocation, and are scalable to a larger array of platforms (including mobile and IoT devices). For this project specifically it is notable that GPU cores simply cannot consume Python kernels. However, the heavy tasks of obtaining device information, transferring data between host and device, building the kernel, etc. required by CUDA and OpenCL to be implemented in C/C++ are reduced to an afterthought with the libraries PyCUDA and PyOpenCL (particularly PyCUDA with its “autoinit” feature).

References

http://stackoverflow.com/questions/3042717/what-is-the-diff-between-a-thread-process-task
http://stackoverflow.com/questions/1050222/concurrency-vs-parallelism-what-is-the-difference
https://www.quora.com/What-is-the-difference-between-concurrency-and-parallelism