Field Programmable Gate Arrays with Natural Language Processing

A Case Study

by Gregory F. Roberts, Jai Evans, and Keith Goddard
groberts@rosoka.com , jevans@rosoka.com, k.goddard@gemasecure.com

Introduction

In this paper we examine how using field programmable gate arrays (FPGA) with Natural Language Processing (NLP) software can increase throughput speeds by a factor of over 30 with 133x less power consumption as compared to a traditional enterprise-scale infrastructure using Cloud resources.

NLP

Natural Language Processing (NLP) is a branch of linguistics concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data. For the purposes of this experiment, we are using the industry-standard, enterprise-scale entity extraction engine by Rosoka Software. This entity extraction engine is a recursive, finite state automata. The Rosoka extraction engine is written in Java. The engine is controlled by the Rosoka LxBase. The LxBase is a data component that contains all of the entity and relationship definitions, language and domain specific dictionaries, entity extraction rules, and relationship expansion details. With a single pass on input, Rosoka provides language identification, entity extraction, relationship extraction, document level multivector sentiment analysis metrics, and entity level multivector sentiment analysis metrics. A user is guaranteed the same results if using the same engine and same LxBase on multiple passes on the same data, regardless of operating system or hardware configuration.

FPGA

Field programmable gate array (FPGA) is a hardware circuit that can be programmed to carry out one or more logical operations. FPGAs are sets of integrated circuits grouped in an array of programmable logic gates, memory, or other elements. With a standard computer processor, such as a CPU in a laptop or server, the circuits are fixed. They cannot be reprogrammed. With an FPGA, there is no predetermined processor. The user programs the hardware circuit or circuits. The programming can be a single, simple logic gate (an AND or OR function), or it can involve one or more complex functions, including functions that,

together, act as a comprehensive multi-core processor. In the past programming FPGAs could only be accomplished with a statically compiled computer language like Assembler or C. An interesting feature of the GemaSecure application of FPGA is the ability to embed non-statically compiled languages, such as Java, on the FPGA. This feature is what enables Rosoka to be able to run on the FPGA. Another interesting aspect is that multiple applications can be integrated on the FPGA to run concurrently and maximize hardware usage while still being performant.

Experiment

In our experiment, we used a corpus of 276MB of online newswire documents collected with the Rosoka News online news aggregation reference architecture. Rosoka News has a set of over 200 seed URLs that collect news from approximately 30 different languages from all over the world. The corpus consisted of 77,807 documents. The documents ranged in size from 46 bytes to 331KB. The corpus represents an average day of news aggregation for Rosoka News reference architecture.

The experiment was conducted through six runs of the Rosoka News Corpus on 3 different architectures. The Rosoka Extraction Engine is thread safe and throughput performance can be increased by expanding the number of processing threads. One set of runs were completed using a single thread for Rosoka processing. Another set of runs were completed using 4 threads for Rosoka processing.

The Laptop Runs were done using a MacBook Pro running macOS Catalina Version 10.15.7 with a 2.9 GHz 6-Core Intel Core i9. The laptop had 32GB of 2400 MHZ DDR4 RAM. The Cloud Runs were done on AWS m6i.4xlarge using AMZ Linux with a 16 core CPU that had 64GB RAM. The FPGA Runs were done on the proprietary GemaSecure FPGA hardware with Xilinx XCZU11EG MPSoC with 32GB of 2400 MHZ DDR RAM and a proprietary FPGA bitstream image to accelerate processing functions though a LLVM open architecture API.

Each run consisted of processing the Rosoka News Corpus using the Rosoka SDK Java API. The elapsed time from input ingest through final output was calculated and stored for each document. Aggregate metrics were then calculated for each run. We wanted to ensure that our comparisons were on Rosoka processing, not disk i/o, so once the Rosoka Extraction Engine was spun up, it was primed with a dummy run of several documents. The dummy documents were not factored into the final timing metrics for any of the official runs. For the multithreaded runs, all threads were processed in parallel. Since each thread completed at a different time, we took the longest processing thread as the aggregate time for these runs.

Table 1 shows the results of the three runs with the Rosoka Extraction Engine using a single processing thread.

	Aggregate Time	MB/Hour
Laptop	2.90 hours	69.75
Cloud	2.63 hours	76.92
FPGA	0.082 hours	2467.56

Table 1: Rosoka Timing Metrics, Single Thread

Single thread results between the Laptop and Cloud processing shows the difference between using additional memory resources to increase throughput processing volumes. Doubling the RAM produces a 1.10 times increase in the throughput processing volumes. While the Rosoka architecture is memory intensive, there is an inflection point where adding additional RAM provides diminishing returns. The minimum amount of RAM necessary for the Rosoka Extraction Engine to process with reasonable results is 8GB. Prior performance testing indicates that 24GB of RAM allocated to the JVM is the optimal amount of RAM, before performance improvements start to fall off. With the Laptop runs, the entire machine has 32GB of RAM, some of which is allocated to the OS and other machine resources. The 1.10 times increase on the Cloud run is due a full 24GB RAM allocation to the JVM, with the remaining RAM used for the other Cloud resources.

Single thread results between the Cloud and FPGA run demonstrates the improved processing performance of using a customized, virtual multi-core processor tuned to Rosoka computational needs. The FPGA customization produces a 32.10 times increase in throughput performance over the Cloud run. The performance boost was due to the FPGA's ability to cut through and streamline Rosoka computationally intensive processes.

Rosoka was designed to be thread safe so a second set of runs using 4 threads to determine if additional threads would affect the FPGA performance as compared to the Cloud architecture were examined. **Table 2** shows the results of the three runs with the Rosoka Extraction Engine using 4 processing threads.

	Aggregate Time	MB/Hour
Laptop	1.21 hours	185.84
Cloud	0.88 hours	235.18
FPGA	0.038 hours	5324.74

Table 2 : Rosoka Timing Metrics, 4 Threads

Comparing the throughput increase between the single thread Cloud run and the multi-thread Cloud run shows a 3.10 times increase in throughput. This makes sense; adding 3 additional threads should increase the throughput 3 times. Adding additional threads over the 4 tested to the tested Cloud infrastructure could provide additional performance, however as the JVM RAM resources are consumed the additional performance would trail off.

Comparing the multi-thread Cloud run to the multi-thread FPGA run provides a 22.64 times increase in throughput.

Conclusion

The Rosoka extraction engine on the GemaSecure FPGA provides upwards of 30 times throughput performance enhancements with a 1u footprint and 75W of power consumption. While the exact amount of power consumption on the AWS m61.4xlarge is not directly available, Benjamin Davy provides some indications that the m61.4xlarge consumes at least 500W with CPU intensive processes, like Rosoka. (https://medium.com/teads-engineering/estimating-aws-ec2-instances-power-consumption-c9745e347959)

That means that for comparative performance, one would need to spin up and manage almost an entire rack of 20 m6i.4xlarge AWS servers that consumes 10,000W compared to similar performance of the GemaSecure 1u FPGA that consumes 75W. The Cloud infrastructure would consume 133x more power than the GemaSecure FPGA.

All of the comparisons were done using a single instance of an out-of-the box Rosoka Extraction Engine implementation with the Rosoka SDK product. To expand the Cloud infrastructure to have similar performance to the FPGA performance, additional Rosoka Extraction Engine instances are necessary. This necessitates load balancing resources to facilitate communication between the servers.

The GemaSecure FPGA with the Rosoka Extraction Engine installed will provide a customer with increased throughput speeds, with a smaller hardware footprint and lower power consumption than building out a traditional enterprise-scale hardware infrastructure.

The Rosoka/GemaSecure combination provides a more secure solution that can be installed and maintained behind a user's own firewall, all with a 1u footprint with only 75W of power consumption. This enables users to move their computational processing needs further afield without having to worry so much about where the power is coming from.

Further Research

Rosoka was built on a Java platform to enable it to run on any architecture where a JVM is available without the need to compile to a specific architecture. This allows Rosoka to run on a wide array of hardware platforms from a Raspberry Pi to a Cray Supercomputer. The results of this case study indicates that further performance improvements could be realized by rearchitecting some of the algorithms to take advantage of the GemaSecure FPGA architecture.

Historically, applications were deployed on general purpose CPU architectures. As performance increases were found by using GPUs, we believe that FPGAs have been underutilized in performance intensive applications. The revolutionary ability to run Java on FPGAs demonstrates that performance enhancements can be realized with the Rosoka/GemaSecure combination to provide near-real time processing of large volumes of narrative texts.