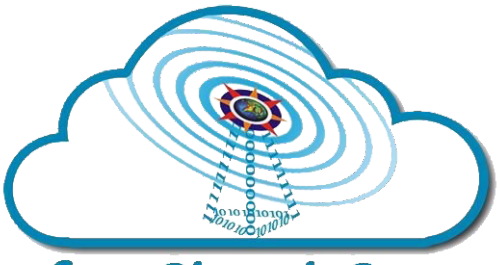

Centre for Cloud Computing and Big Data

Technical Report

Summer Internship June-July 2018

Hardware Acceleration of Big Data Analytics

under the guidance of

Dr Reetinder Sidhu

SNo	Name	Dept	USN
1.	Saahitya E	CSE	01FB16ECS322
2.	Sanjay K A	CSE	01FB16ECS341
3.	Shalini Sai Prasad	CSE	01FB16ECS350
4.	Shashank Bongale	CSE	01FB16ECS355

PES UNIVERSITY

Hardware Acceleration of Big Data Analytics

Saahitya E
Cloud Computing and Big Data Centre
PES University
Bangalore, India
saahitya.e@gmail.com

Shalini Sai Prasad
Cloud Computing and Big Data Centre
PES University
Bangalore, India
shalinisaiprasad@gmail.com

Shashank Bongale
Cloud Computing and Big Data Centre
PES University
Bangalore, India
shashankbongale@gmail.com

Sanjay K A
Cloud Computing and Big Data Centre
PES University
Bangalore, India
ska.q360@gmail.com

Dr. Reetinder Sidhu
Cloud Computing and Big Data Centre
PES University
Bangalore, India
reetindersidhu@pes.edu

***Abstract*- With the increasing popularity of specialized external hardware such as FPGAs and GPUs, there is a need to address the importance of effectively utilising hardware. We discuss techniques like queuing and threading to increase hardware utilisation time. We also discuss the benefits of zero copying in this report. We have successfully implemented these techniques in code and have seen an improvement in the time it takes to run a particular application on the external hardware. Further performance analysis can be done on the techniques we used to determine the exact speedup of the applications using our interface.**

I. Introduction

With the emergence of technology such as machine learning and big data analytics, the need for fast computational power is on the rise. We are part of a project to shift CPU-intensive matrix computations to specialised hardware, in this case, field programmable gate arrays (FPGA) to increase the performance of certain machine learning algorithms. To implement this co-design of hardware and software we recognise the need for a seamless interface between the two layers. The overall project intent is hardware acceleration of certain machine learning algorithms. In this project our aim is to improve the existing interface with an efficient process to maximise hardware utilisation.

II. Related work

In the previous semester, the seniors' team worked on developing an interface to deploy Spark Applications and perform computing intensive operations on the FPGA. They set up the Spark application using the Java Native Interface (JNI) Libraries and this enabled the Java Virtual Machine(JVM) to interface with the FPGA using C++. In their project, they evaluated the speedup obtained in performing matrix addition on the FPGA. For the past six months, another team has been working on the hardware side by designing FPGA logic to perform matrix multiplications.

III. Work environment

To test and run our code we work on a simulation of the Accelerated Function Unit(AFU)[1] provided by Intel which is the simulation of a hardware accelerator implemented in FPGA logic that accelerates or intends to accelerate an application.

IV. Proposed work

The solution to the problem of improving efficiency in the AFU is the development of an appropriate queuing mechanism between the software and the hardware calls. The queuing framework is designed keeping in mind that the AFU should remain idle for the least amount of time. We use a queue data structure as input to the the AFU and we use multithreading to switch between input operations and calls to the AFU. This makes it possible to queue up multiple inputs while the AFU is busy executing its operations on one input.

The implementation of the queue was done using ZeroMQ(ZMQ)[2], a high performance asynchronous messaging library. We use a ZMQ context and create sockets to queue the input data sent by the user, queue the output data received from the AFU as well as send messages between the layers to synchronise waits and indicate the end of certain functions. ZMQ helps us implement appropriate blocking and non blocking of the queue during insertion and deletion, and keeps the queue data type independent.

A major change that we have implemented is concurrency of the application. We use multithreading to separate the I/O functions from the AFU operations and we are able to achieve an asynchronous method of providing input to the AFU. Previously, the application had to wait for all the input data to be received before the AFU started executing its operations. We use different threads to avoid this waiting period and increase the AFU efficiency. The thread implementation and interface design is shown in Fig. 1.

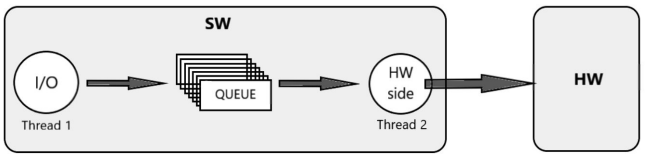

Fig 1: Diagram depicting threading and interface between hardware and software

We also implement a zero-copy model of the application as this reduces time taken during memory allocation. We allocate memory in the appropriate AFU specified method at the start of the application and ensure that data is not copied to new memory while being put on the ZMQ queue.

V. Results

The simulation shows data being queued efficiently. Time is saved by avoiding multiple memory allocations. The queue based hardware-software interface works with the Basic Linear Algebra Subprograms (BLAS) replacement matrix multiply functions to queue up 16x16 matrix chunk pairs to the hardware and send product chunks back on the reverse queue. Since the hardware and software sides of the the interface run in different threads, the FPGA and CPU can operate concurrently thereby improving hardware utilisation.

Acknowledgment

This project was supported by PES University and its Cloud Computing and Big Data Lab. We are extremely grateful to Dr. Reetinder Sidhu for his mentorship and guidance with this project. We are also thankful to our colleagues Rachana Aithal K R, Yashas N D and K Vennela who provided expertise on the hardware side that greatly assisted the research along with Vishal Rao who helped with the final implementation of the project.

References

[1] Intel BDW + FPGA Accelerator Abstraction Layer, Beta Release 5.0.3, Software Programmers Guide

[2] Pieter Hintjens, ZeroMQ - Messaging for Many Applications, O'Reilly Media, CA , 2013