ACCELERATED SMPP DECODER IMPLEMENTATION BASED ON GPU

Prabath Sanjeewa Abeygunawardana Weerasinghe

Registration Number : 148244C

Degree of Master of Science

Department of Computer Science and Engineering

University of Moratuwa

Sri Lanka

December 2017

Declaration

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works.

Candidate

....

Prabath Sanjeewa Abeygunawardana Weerasinghe **Date**

I certify that the declaration above by the candidate is true to the best of my knowledge and he has carried out research for the Masters Dissertation under my supervision.

....

Prof. Sanath Jayasena **Date**

Acknowledgment

This project would not have been possible without the support of many people.
A special thank goes to my supervisor, Prof. Sanath Jayasena for his valuable guidance and immense support. Many thanks to our MSc research project coordinators, Dr. Shehan Perera and Dr. Malaka Walpola, for their dedication and support. Thanks to all the lecturers at the Faculty of Computer Science and Engineering, University of Moratuwa, for their valuable advices.

Abstract

Graphic processing unit (GPU) provides a low-cost but powerful hardware platform for implementing massively parallel high performance systems. The capabilities of GPUs have been used to provide fast and low cost solutions in areas such as machine learning, complex simulations such as global warming and genetic engineering and network traffic processing.

Our research is focused on using GPUs to accelerate the decoding process of the Short Message Peer to Peer (SMPP) protocol. SMPP protocol is used to exchange *Short Messages* between Short Message Service Centers (SMSCs) and TCP/IP based applications. From the point of view of a user, a SMS taking few seconds is acceptable and therefore a SMSC is mostly focused on achieving a higher throughput than a low per packet latency.

We have developed a SMPP decoder library in C with GPU support. It supports both CPU based and GPU based decoding. The library also includes two primary APIs. The first API is for general usage by any C based application and the second API could be used by Java Native Interface (JNI) based application.

We have evaluated the performance of the library in both CPU and GPU modes and compared it with a SMPP server based on Cloudhopper, a Java implementation of SMPP protocol. The evaluation shows around five times throughput gain in GPU mode over both Java and C based CPU modes.

Contents

Declaration ---------- i

Acknowledgment ---------- ii

Abstract ---------- iii

Contents ---------- iv

List of Figures ---------- vi

List of Abbreviations ---------- viii

1 Introduction ---------- 1

1.1 Background ---------- 1

1.2 Problem Statement ---------- 1

1.3 Objectives ---------- 2

1.4 Overview of the Thesis ---------- 2

2 Literature Survey ---------- 3

2.1 Introduction to SMPP Protocol ---------- 3

2.2 Cloudhopper SMPP Library ---------- 4

2.2.1 SMPP Decoding Process ---------- 4

2.2.2 Netty Library Architecture ---------- 5

2.3 Graphic Processing Units (GPUs) ---------- 5

2.3.1 Heterogeneous Computing ---------- 6

2.3.2 Nvidia GPU Hardware Architecture ---------- 7

2.3.3 Streaming Multiprocessor ---------- 9

2.3.4 CUDA Programming Model ---------- 12

2.4 GPU Based Network Packet Processing ---------- 15

2.4.1 Packet Capturing ---------- 16

2.4.2 Packet Classification and Pre-Processing ---------- 16

2.4.3 Transferring Packets Between GPU and CPU ---------- 18

2.4.4 Processing Packets in the GPU ---------- 22

2.4.4.1 GPU Concurrent Kernel Execution Patterns ---------- 22

2.4.4.2 Thread Allocation in GPU ---------- 31

2.4.4.3 Dynamic Parallelism ---------- 32

3 Implementation ---------- 35

3.1 Introduction ---------- 35

3.2 Cloudhopper Changes ---------- 35

3.3 SMPP Decoder Library ---------- 36

3.3.1 JNI Interface -- 37
3.3.2 SMPP Decoder Library Interface -- 39
3.3.3 The Decoder Implementation-- 41
3.3.4 Utility methods for reading a Byte stream -- 44
4 Evaluation---45
4.1 Workload Generation--45
4.2 Experiment Setup--46
4.3 Analysis--46
4.3.1 Maximum Throughput Analysis for Different Batch Sizes--------------------------- 46
4.3.2 Throughput Analysis of Dynamic Parallelism GPU Mode --------------------------- 48
4.3.3 Throughput Analysis of Normal GPU Mode --- 49
4.3.4 Effect of Block and Grid Sizes on Performance -- 50
5 Conclusion and Future Work--52
5.1 Conclusion --52
5.2 Future Work ---52
6 References --54

List of Figures

Figure 2-1 : SMPP frequently used message flow 3
Figure 2-2 : SMPP PDU structure 4
Figure 2-3 : Netty Inbound and Outbound Message Handling Architecture [6] 5
Figure 2-4 : GPU streaming Processor Core Block Diagram [11]... 7
Figure 2-5 : AMD CPU Processor Core Block Diagram 8
Figure 2-6 : Nvidia Fermi Architecture - streaming Multiprocessor Block Diagram [11] 10
Figure 2-7 : Dual Warp Scheduler in Fermi Architecture [11] 11
Figure 2-8 : CUDA Grid and Block Structure 14
Figure 2-9 : Basic CUDA data flow 15
Figure 2-10 : Three options for pre-processing and classifying packets. Option A shows classification in the CPU and option B shows classification in the GPU and option C shows both classification and pre-processing in the GPU. 17
Figure 2-11 : Normal data flow from CPU to GPU 20
Figure 2-12 : Data flow from the CPU to the GPU when data stored in a pinned memory .. 20
Figure 2-13 : Timeline diagram of synchronized decoding cycles 21
Figure 2-14 : Timeline diagram for using non-blocking execution cycles with multiple streams 22
Figure 2-15 : Sequential execution pattern with one copy engine and one execution engine 24
Figure 2-16 : Execution pattern B with one copy engine and one execution engine. The number of streams is two. 24
Figure 2-17 : Execution pattern C with one copy engine and one execution engine. The number of streams is two. 24
Figure 2-18 : Sequential execution pattern with two copy engines and one execution engine. The number of streams is two. 25
Figure 2-19 : Execution pattern B with two copy engines and one execution engine. The number of streams is two. 25
Figure 2-20 : Execution pattern C with two copy engines and one execution engine. The number of streams is two. 26
Figure 2-21 : Timeline diagram for the execution order for H2Da-1, H2Db-1, K-1, D2H-1, D2H-2 for two streams... 27
Figure 2-22 : Timeline diagram for the execution order for D2H-2, H2Da-1, H2Db-1, K-1, D2H-1 for two streams... 27
Figure 2-23 : Timeline diagram for the kernel execution order Ka-1, Kb-1, Ka-2, Kb-2 with two streams. 28
Figure 2-24 : Timeline diagram for the kernel execution order Ka-1, Ka-2, Kb-1, Kb-2 with two streams. 28
Figure 2-25 : Timeline diagram of the kernel execution order Kb-1, Ka-1, Kc-2, Kd-2. Kernel Kb and Kd have twice the execution time of Ka or Kc. 29
Figure 2-26 : Timeline diagram of the kernel execution order Kb-1, Kc-2, Ka-1, Kd-2. Kernel Kb and Kd have twice the execution time of Ka or Kc. 29
Figure 2-27 : Timeline diagram of the kernel execution order Kb-1, Kc-2, Kd-2, Ka-1. Kernel Kb and Kd have twice the execution time of Ka or Kc. 30

Figure 2-28 : Timeline diagram of the kernel execution order Kb-1, Kc-2, Kd-2, Ka-1. Kernel Kb and Kd consumes 2/3 of the GPU resources while kernel Ka and Kc consumes 1/3 of the GPU resources. 30
Figure 2-29 : Timeline diagram of the kernel execution order Ka-1, Kb-2, Ka-2, Kb-1. Kernel Kb and Kd consumes 2/3 of the GPU resources while kernel Ka and Kc consumes 1/3 of the GPU resources. 30
Figure 2-30 : The SMPP PDU decoding process flow diagram 32
Figure 2-31 : Dynamic Parallelism parent kernel and child kernel execution model 34
Figure 3-1 : Primary components in the SMPP decoder library 35
Figure 3-2 : Structure of the decodable element accepted by the API 39
Figure 3-3 : Dynamic Parallelism based SMPP decoding flow 42
Figure 3-4 : SMPP decoding flow with CPU based pre-processing and GPU based decoding 43
Figure 3-5 : Cloudhopper SMPP PDU original decoding flow 35
Figure 3-6 : The altered design with SMPP decoder library integration 36
Figure 4-1: Max throughput analysis for four decoding modes and four batch sizes 46
Figure 4-2 : Throughput analysis for different thread configurations - Dynamic parallelism mode 48
Figure 4-3:Throughput analysis for different thread configurations - Normal GPU mode ... 49
Figure 4-4 : Performance for different Block and Grid sizes. The batch is 10,000 elements. X axis format is (Block Size * Grid Size) 50
Figure 5-1 : Updating decoded fields with delimiters 53

List of Abbreviations

GPU	Graphics Processing Unit
SMPP	Short Message Peer to Peer
CPU	Central Processing Unit
SMS	Short Message System
SMSC	Short Message Service Center
ESME	External Short Message Entity
PDU	Protocol Data Unit
TLV	Tag Length Value
CUDA	Compute Unified Device Architecture
SM	Streaming Multiprocessor
SP	Streaming Processor
UMA	Unified Memory Access