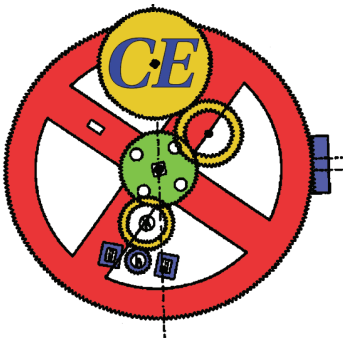

Computer Engineering
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ce.et.tudelft.nl/

2013

MSc THESIS

Implementation of Bio-Informatics Applications on Various GPU Platforms

Amora Amir

CE-MS-2013-15

Abstract

As of 2012, the world creates 2.5 quintillion (2.5×10^{18}) bytes of data every day. Much of this data generated is what we refer to as Big Data. To explore how Big Data can create potential value and show the technical challenges accompanied with Big Data applications, we choose an application from bio-informatics: the Smith Waterman genetic database alignment algorithm, which is used for finding optimal genetic sequence alignments. The continuous increase in the volume of data in genetic databases leads to the exponential increase in the time required for comparing these genetic sequences. This thesis investigates the acceleration and optimization of the Smith Waterman algorithm using GPU platforms. The thesis uses DOPA, an existing implementation, which was optimized for the GTX275 GPU platform from NVIDIA. DOPA resulted in a huge performance gain compared to other implementations running sequentially on CPU. Our thesis aims to study and improve the behavior of this implementation on different NVIDIA GPUs: the Tesla C2075 and the GeForce GT640. We improved the cores occupancy of DOPA on different GPU cards resulting in an efficient workload distribution, thereby improving the performance by about 14% to 61%. We achieved 25 GCUPS performance on the C2075 and 11 GCUPS on the GT640 compared to a straight forward DOPA port on the same cards achieving 21.9 and 6.8 GCUPS, respectievly. To achieve considerable performance for Big Data application for different platforms, two important factors have to be taken into account: increase the parallelism in the software and increase the utilization on the hardware side. We evaluated and presented other metrics such as the cost in terms of euro and watt to be considered along with GPU performance.

TU Delft
Delft University of Technology

Faculty of Electrical Engineering, Mathematics and Computer Science

Implementation of Bio-Informatics Applications on Various GPU Platforms

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

by

Amora Amir
born in Baghdad, Iraq

Computer Engineering
Department of Electrical Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Implementation of Bio-Informatics Applications on Various GPU Platforms

by Amora Amir

Abstract

As of 2012, the world creates 2.5 quintillion (2.5×10^{18}) bytes of data every day. Much of this data generated is what we refer to as Big Data. To explore how Big Data can create potential value and show the technical challenges accompanied with Big Data applications, we choose an application from bio-informatics: the Smith Waterman genetic database alignment algorithm, which is used for finding optimal genetic sequence alignments. The continuous increase in the volume of data in genetic databases leads to the exponential increase in the time required for comparing these genetic sequences. This thesis investigates the acceleration and optimization of the Smith Waterman algorithm using GPU platforms. The thesis uses DOPA, an existing implementation, which was optimized for the GTX275 GPU platform from NVIDIA. DOPA resulted in a huge performance gain compared to other implementations running sequentially on CPU. Our thesis aims to study and improve the behavior of this implementation on different NVIDIA GPUs: the Tesla C2075 and the GeForce GT640. We improved the cores occupancy of DOPA on different GPU cards resulting in an efficient workload distribution, thereby improving the performance by about 14% to 61%. We achieved 25 GCUPS performance on the C2075 and 11 GCUPS on the GT640 compared to a straight forward DOPA port on the same cards achieving 21.9 and 6.8 GCUPS, respectievly. To achieve considerable performance for Big Data application for different platforms, two important factors have to be taken into account: increase the parallelism in the software and increase the utilization on the hardware side. We evaluated and presented other metrics such as the cost in terms of euro and watt to be considered along with GPU performance.

Laboratory : Computer Engineering
Codenumber : CE-MS-2013-15

Committee Members :

Advisor: Koen Bertels, CE, TU Delft

Member: Zaid Al-Ars, CE, TU Delft

Member: Andre Bossche, EI, TU Delft

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Prof. Koen Bertels and Dr. Zaid Al-Ars for their supervision and allowing me to do work I found interesting. Without their help, it would not have been possible to complete this work. I would like to thank my husband Raad Al Moussawy and my mother and sisters for their encouragement and support in spite of the difficulty of reconciling between study and my children. I especially appreciate the support i had from my professors and classmates during my study. I would specifically like to thank my classmate Kim Wai Tang. Im really grateful for all.

Amora Amir
Delft, The Netherlands
September 4, 2013

1 Big Data

1.1 Introduction

Big data in information technology is a new technology that deals with huge quantities of information digital data. It is so big and complex in such that it cannot be processed or analyzed using the traditional database, software tools and processors. This data generates from everywhere, posts on social media sites, digital photo and videos, sensors, bank transaction, mobile phone, GPS signals and much more. Big Data is sized in peta-(10^{15}), exa- (10^{18}) and in zetta-(10^{21}) byte.

The number of Internet users have grown dramatically in the last decade, from 361 million users in December 2000 to 2406 million users in December 2012 according to Internet Worlds stats. Consequently, in 2005, man made data was estimated to be 150 exabytes (exabyte is a billion gigabytes). While in 2010 there were 1200 exabytes data information and 95% of this data was unstructured [1]. IBM expects that in 2020, 35 zettabyte of data will be produced. International Data Corporation's (IDC) forecast shows that the Big Data technology and services market is expected to increase to $16.9 billion in 2015 [13].

The Big Data era is in full force today because the world is changing. The trend of Big Data productivity is basically coming from three factors. Firstly: instrumentation [2] is an important factor for Big Data. Instrumentation such as information-sensing mobile devices, digital cameras, telescope and environmental sensory technologies, radars and wireless sensor networks. Secondly advances in communications technology is another factor for producing Big Data. People are becoming increasingly interconnected. Social media networking like Facebook, Twitter and Internet search engine are good examples for inter connectivity. Finally, the third contributing factor for increasing Big Data is that the hardware and storage -cost have decreased dramatically. Therefore intelligence system will be required almost everywhere [2].

Data-driven decision-making in business intelligence and the passion for the notion of Big Data notice widely. The hope of Big Data is actual, for example, an estimation shows that Google alone contributed 54 billion dollars to the US economy in 2009 [9], there is now a broad gap between Big Data potential and its recognition. Decisions that earlier were based on guesswork, or on agilely constructed models of reality, can now be made based on the data itself.

As stated in research at [3] Big Data is considered as the next frontier for innovation, competition and productivity. McKinsey states several important points regarding Big Data. Extract insight and mining, identify trends and use the data to improve productivity, gain competitive advantage and create substantial value for the world wide economy. Data is expected to increase in the world by 40% per year while the spending in IT sector is only 5%. Therefore, a major investment in Big Data is urgently needed.

That will facilitate the speed of discovery in science, education, engineering, security, and health and inlarge the workforce needed to develop and use Big Data technologies.

1.2 Characteristics of Big Data

Three characteristics define Big Data: Volume, Variety, and Velocity or simply known as "3Vs" model as shown in Figure 1.1. However, some people are talking about the 4th V - Value. These 3Vs characteristics together define and view the nature of the Big Data and the manner in which the value can be discovered from. Let's define each term explicitly.

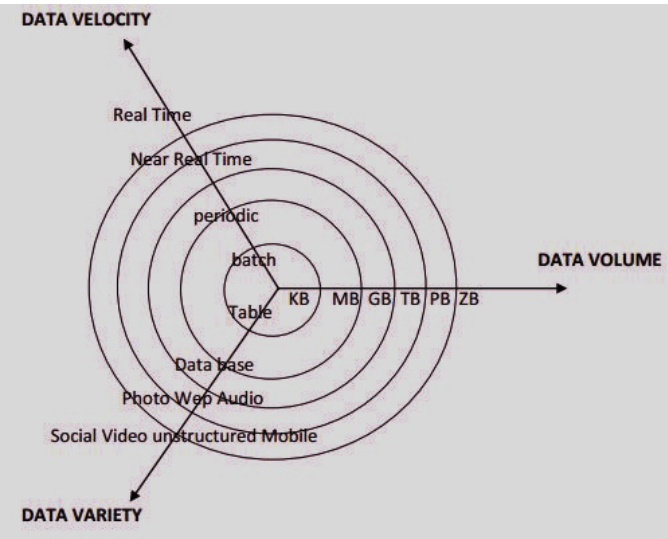

Figure 1.1: 3Vs model based on [17]

1.2.1 Volume

The sheer volume of data being produced nowadays is fairly obvious. The volume describes the amount of the generated data. It is the size of the data from kilo, mega, peta or even zettabytes. Figure 1.1 shows how the volume of the data is varying e.g. a text file is a few kilobytes, a data base is gigabytes while a social media web streaming is petabytes. In the year 2012, around 2,7 zettabytes of global data has been created [14, 16, 17]. According to IDC (Figure 1.2) the amount of produced data today has exceeded the conventional available storage. There is an evident gap between the produced data and the storage available today. As a result, of course, it is not possible to store all this data much of this data will be discarded. IBM expects this amount of data will be increased dramatically. Twitter produces every day about 7 terabytes (TB) of data, Facebook 10 TB and some enterprises produce terabytes of data each hour of every day of the year. More sources of data with larger quantities have increased the volume of data set. Therefore enterprises of most domains are urgently required to find ways to

handle the ever-increasing data volume that is being created every day. Peta-byte data sets are common these days and zettabyte is not far away.

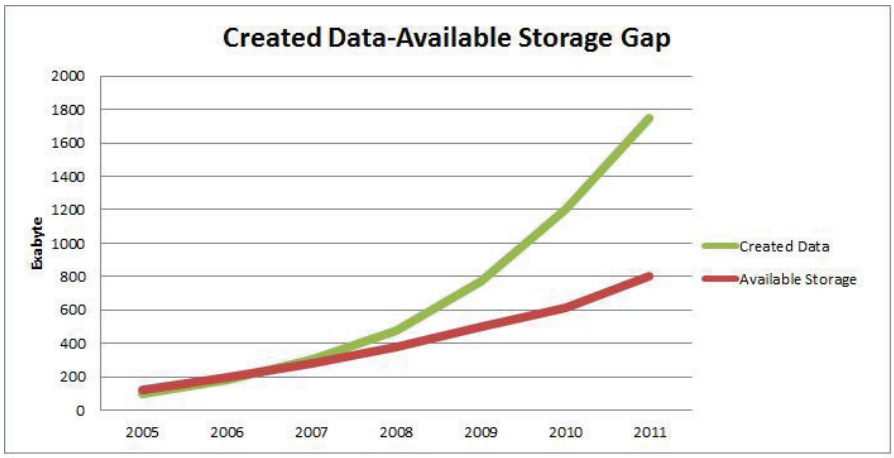

Figure 1.2: Created data vs. available storage [14]

1.2.2 Variety

The volume of Big Data phenomena introduces another challenge: the variety of the data. Data has become complex with the burst of smart phone devices, sensors, social media and other technologies. This complexity is because the data includes not only the conventional data, but also raw, semi structured and unstructured data [16]. The problem with such kind of data is the traditional database technology can struggle to store and analyze this data to get understanding and insight from the contents of these logs. Data variety will not be possible to handle with the traditional analytics platforms. However the success of an organization will depend on its ability to discover the insights from different kinds of data available today. To get benefit from the Big Data opportunity, enterprises should be able to analyze all types of data.

1.2.3 Velocity

Velocity "describes the frequency when the data is produced, captured and shared" [16]. Basically, organizations analyze data using batch processing [17] (Figure 1.1). "Batch processing [15] is the execution of a series of programs on a computer without manual intervention. This is in opposite to interactive programs which push the user for such input". The flood sources of Big Data such as social media, streaming and smart phone applications breaks down the batch process. The data is now streaming continuously into the server in real time. So the traditional existing system of data analysis should be reconstructed by adopting new infrastructure, we will see in the next sections the components of Big Data platform.

There are many examples of Big Data in our daily life, including but not limited to:

- 30 billion RFID (radio frequency ID) tags produced/year.

- 10,000 payment card transactions are made every second around the world.
- Walmart handles more than 1 million customer transactions an hour.
- 340 million tweets are sent per day. That's nearly 4,000 tweets per second. Bin Laden's death: 5106 tweets/second.
- Facebook has more than 901 million active users generating social interaction data.
- More than 5 billion people are calling, texting, tweeting and browsing web sites on mobile phones.
- Falcon Credit Card Fraud Detection System protects 2.1 billion active accounts world-wide.
- Large Hadron Collider generates 40 terabytes/second
- Airbus 380 has 1 billion lines of code each engine generate 10 TB of data every 30 minutes.
- Oil drilling platforms have 20k to 40k sensors.

1.3 Big Data challenges

Big Data has common challenges. Such challenges are, not limited to, heterogeneity, timeliness, scale, complexity, and privacy problems which impede progress at all phases of analysis. The information in this section largely comes from [9].

- **Heterogeneity** most of data today is produced in an unstructured format, such as text or video. However, machine analysis algorithms expect homogeneous data, and cannot understand unstructured data. It is a challenge to convert such data into a structured format for further analysis.
- **Timeliness** is a challenge associated with Big Data. It is a challenge to capture and link the required data as it happens and deliver that to the right people in real-time.
- **Scale and complexity** is another challenge to store and analyze data with given its size and using traditional computational capacity resources.
- **Privacy** is another challenging concern that increases the public's fear regarding their personal data. However, there are strict laws and regulatory compliance governing what can and cannot be done with personal data, there is apprehension of inappropriate use of personal data, particularly through linking and the accumulation of data from multiple sources. Managing privacy is an effective problem which must be considered seriously to return the promise of Big Data.

1.4 Big Data opportunities for emerging domains

Big Data has potential benefits [3]. The challenges with Big Data are bounded if it is compared to the potential benefits, which are bounded if we are working hardly and try to make connection between these huge available data.

Through comprehensive analysis of the wide volumes of data that are becoming available, there is a potential for rapid developing in many disciplines. But before that these technical challenges described previously and others must be fixed before this potential can be achieved fully. This section will view the potential benefits of Big Data in several domains. Much of this section based on [3, 4, 5, 6, 9]

- Scientific research, discovery and education have been altered and reformed by Big Data [4]. In the biological sciences, for example, there is now orientation to computerized biology which is known as bio-informatics domain. Biologists use the tools of data analytic such as advanced algorithms, pattern recognition, data mining, machine learning, and data visualization as well as new computation technology approaches to conclude and validate the accuracy of the tests and increase the speed of experiments of related biological systems.

- It is widely believed that the use of smart health prevents disease and minimizes the overall cost of health care while improving its quality [5]. This demand is deployed by making care more preventive and personalized and basing it on real time home-based continuous monitoring. McKinsey shows an estimation [3] that a savings of 300 billion dollars every year through applying smart health care in the US alone.

- Big Data also has the potential to improve education [6]. For example, a recent quantitative comparison study developed for 35 charter schools (charter schools are a kind of school developed to serve as an R&D engine for traditional public schools) in New York has found that one of the best five policies related with "measurable academic effectiveness" was the use of data to guide instruction [8]. The study was collecting performance of student for different education levels starting from primary school until university. This is helped to develop several models. Efforts to digitalized the education and using web deployment will produce a large quantity of detailed data about students' performance.

- In addition, there have been shown the benefit of Big Data for several domains. Such domains are smart transportation [7], urban planning, financial risk analysis system, protecting the environment and security and computer security and much more.

- Employees and engineering also benefit from Big Data. McKinsey [3] predicts a great outcome of Big Data in employment sector, where 140,000-190,000 employees with deep analytical experience will be needed in the US; additionally, 1.5 million managers will need to become data-literate.

1.5 Big Data supply chain

Big Data supply chain shows how an organization gathering, managing, processing, analyzing and acting on data within a particular time for large data sets (Figure 1.3). Our interest in this thesis is on the hardware related components like platform in order to shed light and discuss their challenges with Big Data applications. The information in this section largely comes from[11].

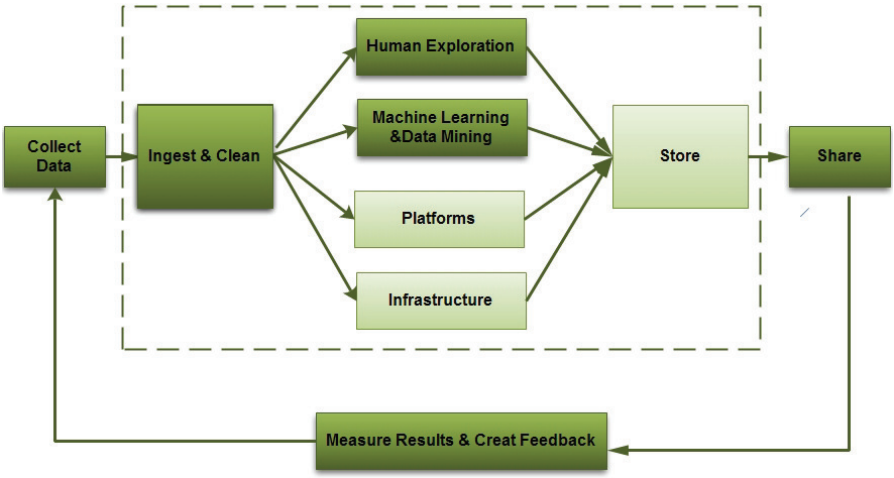

Figure 1.3: Big Data supply chain based on [11]

There are several examples of Big Data platforms, following the components of the IBM and Oracle Big Data platform that addresses the spectrum of Big Data business challenges are presented. The trend of the IBM Big Data platform provides the ability to easily adapt it's components to the existence enterprise information architectures. Figure 1.4 shows the components of IBM Big Data platform [10].

The common components of IBM Big Data platform are divided into two approaches, one is InfoSphere Streams and the other is InfoSphere BigInsight. Both approaches are designed for Big Data analytics but the first approach is for data in motion and the second is for data at rest [10, 11]:

- Visualization and Discovery: discover, explore, understand, search, gather and navigate different sources of data.
- Hadoop-based Analytics: store and analyzed any type of data (structured, semi and unstructured) in Hadoop engine which has a potential to lower the cost of processing and analyzing massive volumes of data.
- Stream Computing: analyze massive volumes of streaming data with ultra low latency to take action in real-time.
- Data Warehousing: store and analyze large amount of structured information with deep deliver insight for system operational analytics.

  The services that supporting the platform are:

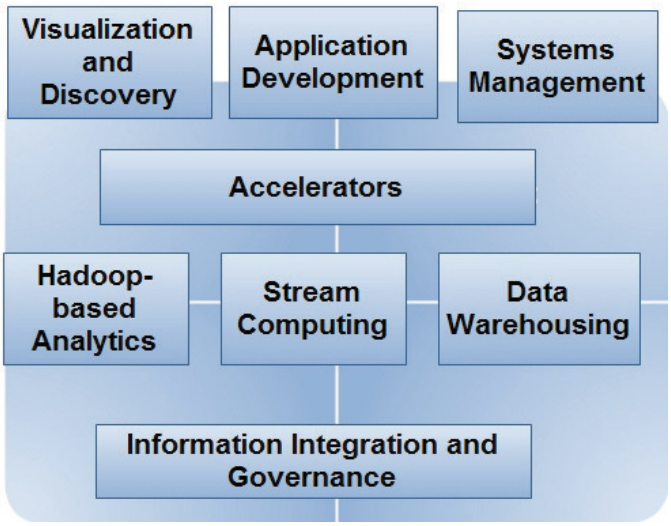

Figure 1.4: IBM Big Data platform based on [10]

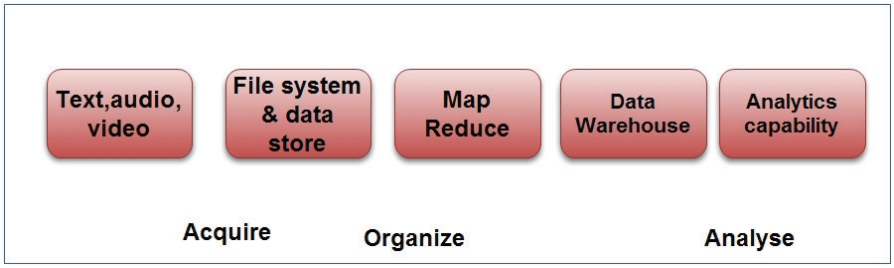

Figure 1.5: Oracle Big Data platform based on [12]

- Accelerators: performing pre-packaged analytical and application accelerators.
- Application Development: control the process of developing Big Data applications.
- Information Integration and Governance: integrate, protect, cleanse, govern data quality, deliver trusted information and manage information life cycle.
- Systems Management: monitor and manage a Big Data system for secure and optimized performance.

Oracle offers a broad and integrated Big Data platform. It has potential, like IBM, to integrate the platform with existing deployed enterprise data analysis system. Furthermore, it helps to acquire and organize the diverse data sources and analyze them to find new insights. The infrastructures of Oracle Big Data platform cover data acquisition, data organization and data analysis (Figure 1.5). During the acquisition and organization phase, distributed parallel processing architectures adopted to process data sets [12]. There are different technology for real-time distributed parallel processing. Apache Hadoop and Map Reduce are common technologies that organize and process large data

sets. Afterward, when the data is discovered and organized, it can be analyzed. Different methodology and tools are used for analyzing these data sets for example, statistical analysis using spreadsheets.

Bibliography

[1] National Institute of Standards and Technology (NIST)

[2] Book: Charis Eaton, Tom Deutsch, Dirk Deroos, George Lapis, Paul Zikopoulos. Understanding Big Data. IBM. 2012.

[3] James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers. Big data: The next frontier for innovation, competition, and productivity. McKinsey Global Institute. 2011.

[4] Advancing Discovery in Science and Engineering. Computing Community Consortium. 2011.

[5] Smart Health and Wellbeing. Computing Community Consortium. 2011.

[6] Advancing Personalized Education. Computing Community Consortium. 2011.

[7] A Sustainable Future. Computing Community Consortium. 2011.

[8] Will Dobbie, Roland G. Fryer, Jr.Harvard. Getting Beneath the Veil of Effective Schools: Evidence from New York City. University November 2011.

[9] Alexandros Labrinidis, Ann Arbor. Challenges and Opportunities with Big Data. 2012.

[10] Book: Paul Zikopoulos, Dirk Deroos, Krishnan Parasuraman, Thomas Deutsch, David Corrigan, James Giles. Harness The Power of Big Data. IBM Big Data Platform. 2013.

[11] Book: Planning for Big Data. OReilly Radar Team. ISBN: 978-1-449-32967-9 March 2012.

[12] Oracle: Big Data for Enterprise. January 2012.

[13] Sachchidanand Singh and Nirmala Singh. Big Data Analytics. International Conference on Communication. Oct 2012.

[14] http://www.idc.com/

[15] Batch processing. http://www.wikipedia.org/wiki/Batch-processing

[16] R Ray Wang. Mondays Musings: Beyond The Three Vs of Big Data Viscosity and Virality. 2012.

[17] Diya Soubra. The 3Vs that define Big Data. posted on Data Science Central. 2012. http://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data

Computer architectures of Big Data

2

Parallel computing is a sort of computation where a task is divided into sub tasks that can be solved concurrently on multiple processing elements. Each sub task is then divided into a series of instructions. Instructions from each sub task execute simultaneously on different processing elements [13].

There are several different forms of parallelism: instruction-level, data-, and task-level parallelism. Parallel computing approaches can be approximately categorized as either data-intensive or computation-intensive [14].

Data-intensive computing is a data parallel approach used to process huge amount of data basically terabyte or petabytes in size, under which Big Data can be classified. Whereas computationally-intensive computing is another approach which dedicates the majority of the execution time to computational requirements and typically requires small volumes of data [14].

The characteristics of Big Data are represented by the 3Vs (volume, velocity and variety) as mentioned earlier in Section 1.2 pose fundamental challenges for businesses and academia in many different domains. Such challenges are forming because the Big Data supply chain is not in line with the technologies that are currently available. Big Data applications require such peta, exa and zetta FLOPS-level machine. As a consequence the architectures used in the computer in terms of CPU, memory and interconnect should be taken into consideration.

Fortunately, research and the industry landscapes have started to change hardware, software and existing computational techniques to address these issues since the complexity, speed and scalability of data production reached available hardware and software limits. According to HPC Advisory Council, exaFLOPS (10^{18} operations) system will be built between 2018 and 2020 [15].

The TOP500 list certified Titan in November 2012 as the world's fastest supercomputer per the LINPACK benchmark, at 17.59 petaFLOPS. It was developed by Cray Inc. combines AMD Opteron processors with "Kepler" NVIDIA Tesla graphic processing unit (GPU) technologies [16, 17].

Nevertheless, existing data-intensive computational techniques can be applied for Big Data applications but with sophisticated algorithms that can be compatible or solvable for current Big Data problems. For example, a relational databases management system (RDMS) relies on structured data while Big Data is unstructured, therefore, developers are migrating Big Data to parallel database management systems instead. Similarly, SQL a standard popular language used to express various database queries is being replaced with NoSQL for Big Data application.

Parallelism is a key design feature for Big Data computing at various levels of abstraction. For an effective use of the available hardware parallelism on systems, several parallel programming models and languages are developing. To this end, various tech-

nologies are being introduced to exploit parallelism. To name a few: CUDA, OpenCL, OpenMP.

This chapter gives background information about todays parallel computer architectures. We explores distinct parallel processor architectures. In order to cover relevant components of the most conventional processor architectures available today, we expose core organization, memory architectures and their associated solutions. Much of the following sections present the information about computer architecture is basically derived from computer architectures books like [10].

2.1 Processor

2.1.1 Multi-core processors

The tendency to replicate multiple processor cores on a single die established when the application developers were heavily looking for advanced hardware technologies to boost the applications performance while keeping costs at a minimum. The paradigm to increase performance was by increasing the clock frequency per a single-core processor. However increasing clock frequency implied increasing in power consumption, therefore, the efficiency of the system in term of performance per watt and the problem of heat dissipation from transistors hampered the continuation of this paradigm. An estimation and projections technology shows that clock-speeds will remain near 1-2 GHz [20].

This problem led us to scale parallelism by increasing the number of processing units per system (i.e. multi-core processors). Therefore, the migration in 2001 to the first general-purpose processor that dedicate multiple processing cores on the same chip was produced the POWER4 processor of IBM.

Multi-core processor is an integrated circuit where two or multiple independent processing units (cores) plug onto a single processor (chip). The processors in multi-core architectures can be considered either as a homogeneous or a heterogeneous.

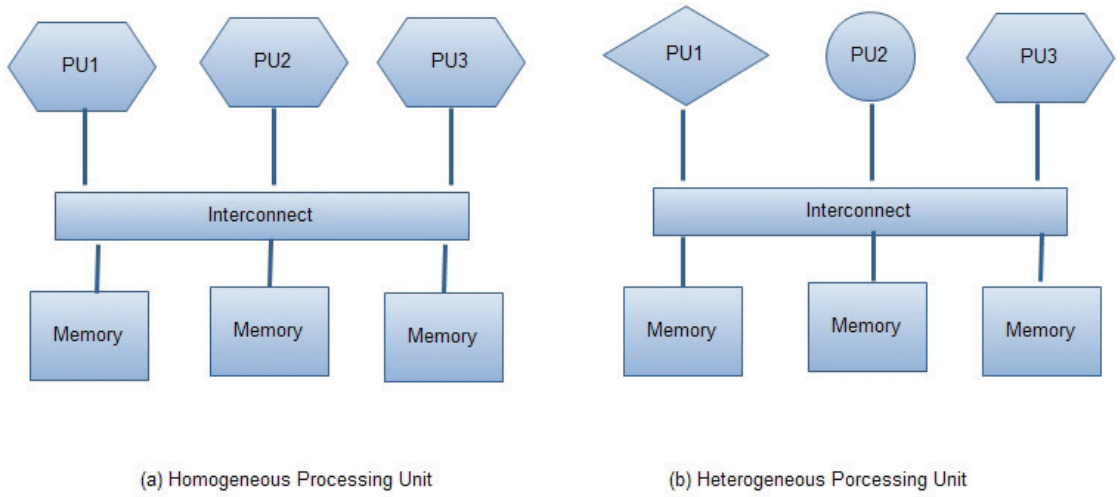

Figure 2.1: Multi-core architecture

A homogeneous architecture Figure 2.1 (a) consists of two or more identical cores. An example of a homogeneous multi-core processor is the Intel Core 2 Duo. In contrast, a heterogeneous architecture Figure 2.1 (b) consists of two or more different kinds of cores that may differ in functionality and performance. The most wide spread example of a heterogeneous multi-core processor is the Cell BE architecture, developed by IBM, Sony and Toshiba and has been used in gaming and video devices like in Sony PlayStation 3.

The tendency of multi-core processors for a better utilization of hardware resources by effective exploitation of simultaneous multithreading (SMT) (Intel brand names hyper-threading) within the applications, which is increasing applications overall system performance.

The performance of multi-core processors is bounded by the memory wall problem. The memory wall is defined as the gap between the speed of the processors and the speed of memory. However, the latency gap is minimized by increasing the number of cores and emerging the on-chip memory technology.

2.1.2 Graphics processors GPGPU

Other paradigm shifts towards increasing parallelism that is attracting hardware developers during the past few years is General Purpose Graphics Processing Unit (GPGPU).

Recently, GPU is being used for the calculation of computer graphics whereas GPU was initially designed to exploit and accelerate the memory-intensive calculation of texture mapping and geometric like rotation calculation [18, 21]. GPU is physically located in the personal computer on a video card or on the motherboard. The computation nature behind GPU have rapidly encouraged hardware developers to adapt GPUs for non-graphical calculations. Simply, they developed heterogeneous computations applications using both the CPU and GPU. Both NVIDIA [18] and ATI [19] are the main GPU vendors landscape. They develop different GPU models, platforms and programming model in order to employ parallel computing architectures to utilize the GPU's stream processors. NVIDIA developed CUDA programming models and ATI produced CTM.

Besides, GPUs are widely involved in large-scale computers. According to Top5oo 3 of the 10 most powerful supercomputers in the world use GPU technology [17].

2.2 Memory systems and cache hierarchy

Today microprocessor memory architecture consists of a number of processors connecting with registers, caches and memory. Over the years, the size of the memory has dramatically increased because of the new technology. The speed of the memory is increasing by 10% per year [10]. While the performance of a microprocessor has improved and the processor speed has increased about 55% per year. The processor-memory performance gap has increased and this problem is referred to as memory wall [10]. Therefore the computer designers conducted the memory hierarchy to hide the latency of accessing large memory. Level 1 and 2 caches (SRAM) are the fastest, smallest, and most expensive memory modules in the system. While the main memory (DRAM) is much slower and much larger than the caches, and it is located far away from the processors. External

storage is far larger than the cache, and the main memory but also operates at extremely slow rates relative to processor speeds.

Each processor in the typical architecture has an accompanying level 1 (L1) for both data and instruction cache. Level 2 (L2) cache may be implemented among the processors either as private or share. Some architecture considers an ample on-chip level 3 memory. When the processor requests data from memory, the level 1 (L1) cache is the first place to serve this request. If the requested data does not exist in the L1 cache, then the cache will request the data form the subsequent levels of the cache. If the requested data does not exist in any of the cache levels, then the main memory receives the request. After that the main memory will forward the requested data to the higher levels of the memory hierarchy and to the processor. In order to maximize the chance that the requested data is existing in the highest level of the memory hierarchy, the level 1 cache must have the appropriate data before the processor requests it. To achieve this, a replacement policy is required. Replacement policy decides which data to keep and which data to evict from the caches. The following explain the replacement policy and cache organization.

2.2.1 Replacement policy and cache organization

Initially when a processor requests data or instructions from a cache, there are two possibilities: either the requested data exists in a cache and (this attempt is referred as hit) the processor can immediately access the data, or the requested data does not exist in a cache (this attempt will fail and miss cache occur) and the processor must wait until the cache forwards the request to the lower level of the memory hierarchy. Replacement policy decides how the cache reacts to a cache miss.

Two principles are behind the replacement policy: temporal locality and special locality. Temporal locality states that the data that is currently being accessed has a high probability that it will be accessed again during the lifetime of the program. The temporal locality is related to a time. While the special locality on the other hand depends on the location of a data. Special locality states that when a processor accesses a particular data there is a high probability that a processor will access neighboring locations during the lifetime of the program. There are several replacement policies. Least-recently-used (LRU) is one example. LRU keeps track of a data in a cache. The data that has not been used for long time is most likely not needed in the near future. Therefore when a new data enters a cache, the LRU evicts the old data that is least recently used by a processor.

Cache organizations basically vary between direct mapped and N-way set associative. Direct mapping strategy maps each location in main memory to exactly one entry in the cache. This strategy has fast access and it does not have a replacement policy. The other type of a cache organization is N-way set associative. This strategy maps each location in main memory to one of N places in the cache. There might be 2, 4 or 8 ... -way set associative cache design. Direct map is 1-way set associative.

2.2.2 Processing In Memory (PIM) technology

PIM, sometimes called a processor in memory, is the integration of a core processor, with usually, a DRAM (dynamic random access memory) on a single die. Sometimes

referred to as a PIM chip. Processing in memory is one approach to improve memory wall problem by combining logic and memory on a single chip. The data-intensive architecture (DIVA) project exploits PIM chips as smart memory coprocessors [12]. This technology exploits considerable memory speedup. Jeffrey et. al stated that [12] PIM architecture get its benefits from the implicit memory bandwidth for target several bandwidth-limited applications, such applications are multimedia applications, pointer-based and sparse-matrix computations. To demonstrate the benefit of PIM approach DIVA project used PIM chips in their prototype system instead of standard DRAMs. Jeffrey et. al stated that their work reported a speedup of 35x on a matrix transpose operation.

2.2.3 Reconfigurable cache technology

Current processor designs assign(50%- 80%) of the on-chip transistors to caches [1]. Big Data applications like media processing and video encoding do not use these large caches efficiently because of the streaming nature of data accesses and the large working sets in these applications. Moreover, the real-time applications require guaranteed cache access times, in a multithreading technology might not these application get these guarantees because the memory references pattern have less temporal locality [2, 3, 6].

The cache configuration basically depends on the application requirements constrain, like performance, power, area, or price which has led to the different cache design architecture can be seen in different processors [9]. There is no cache organizations achieves all the requirements of all applications, therefore the approach of reconfigurable cache memory is raised.

Researchers at MIT University, NC State, the University of Rochester and others like at [1, 2, 3, 4, 5, 6, 7, 8, 9] have studied and proposed new cache organizations which are referred to as reconfigurable caches. The concept of conventional reconfigurable architectures is not new [1]. A reconfigurable architecture basically requires large changes in both hardware and software. The configurable cache design of the aforementioned researchers were extended a conventional cache design with minor changes in a hardware and a software.

The research conducted in the reconfigurable caches can be categorized according to the researcher's major concerns. [1, 2, 3, 7, 9] are focused on cache utilization that is leading to overall better system performance while [4, 5, 6] are focused on low power consumption. For example, the finding of Ranganathan et. al [1] by using reconfigurable caches for instruction reuse of media processing reported IPC (inter-process communication) improvements ranging from 1.04x to 1.20x in simulation across eight media processing benchmarks. Likewise, Albonesi's approach [4] demonstrated a 40% reduction in overall cache energy dissipation for 4-way set associative caches with less than 2% overall performance degradation. While Chen et. al [8] proposed a model that improved both the overall performance and energy consumption in embedded system.

Initially, the overall cache size is determined by multiplying the line size, the number of sets, and the associativity [10]. The line size, the number of sets, and the associativity are the main design parameters of traditional caching techniques [7, 9, 10]. In current cache design these parameters are static during the lifetime of the processor. In order to reconfigure a cache, techniques must be developed to change one or all of these

parameters during program execution [7].

Changing the line size and the number of sets poses a problem [7]. Changing the cache line size affects the number of tag bits. The cache line size and the number of tag bits are inversely related [7, 10]. The cache looks in the tag fields to compare one tag with another to find if the wanted data is located in the cache. Therefore the complexity of the tag comparison model will increase dramatically if the number of tag bits is dynamically changed. Nevertheless the same problem will accrue if the number of sets is varying too. When the number of sets will change, the number of tag bits that is needed for each cache line will also change. Therefore most researchers leave these two parameters (line size and number of sets) static during a program life cycle while they have utilized a variable associativity parameter to reconfigure a cache [1, 7].

2.2.4 Techniques for reconfigurable caches

Several studies and approaches like at [1, 4, 5, 7] have been conducted to reconfigure a cache. In this section, we present these approaches and techniques in more detail and we report their performance.

Ranganathan et. al [1]. developed an associative-based partitioning technique. Their design dynamically divides the cache (SRAM) arrays to multiple partitions. These partitions can be used for different processor activities that is requiring storage. Two key design challenges are determined: First, how to partition the cache SRAM array and secondly, how to address the different partitions as to not affect the access time.

They exploit the conventional set-associative cache organization and they reconfigure it with minor hardware and software changes. Figure 2.2 shows the block diagram of conventional 2-way set-associative cache organization. A 2-way set-associative cache is divided into 2 (data and tag) arrays and it has 2 ways either way 1 or way 2. The "Tag" part of the input address is sent to the comparators of all the 2 ways to check if there is a match with any tags that might come from the tag array [1]. If a match occurs then a hit is signaled on the output line and the data will move from the corresponding data array and is sent onto the output data line. The "Index" part of the input address is used to index the 2 ways of the data and tag array [1].

The associative-based partitioning technique for reconfigurable cache as shown in Figure 2.2 is constructed on the conventional cache organization with a number of changes. The modifications on the original design are colored in orange as shown in the Figure 2.3. A reconfigurable cache with n partitions must duplicate the address input "Address in" n times and duplicate data out and hit/miss signals n times. A conventional cache with n-ways can be dynamically reconfigured into n-partitions.

A special hardware register is required for the associative-based partitioning technique. This register is called the cache status register. It is responsible to track the number and the sizes of the partitions and control the routing of the hit/miss signals [1].

Ranganathan et. al [1] used cache scrubbing technique for data consistency. This approach ensures that after reconfiguration, the correct data resides in a correct partition or the data is moved from the cache to lower levels of memory. Scrubbing techniques use the partitioning information in the cache status register. Another issue with reconfigurable cache organization is the detection mechanisms for deciding when to reconfigure.

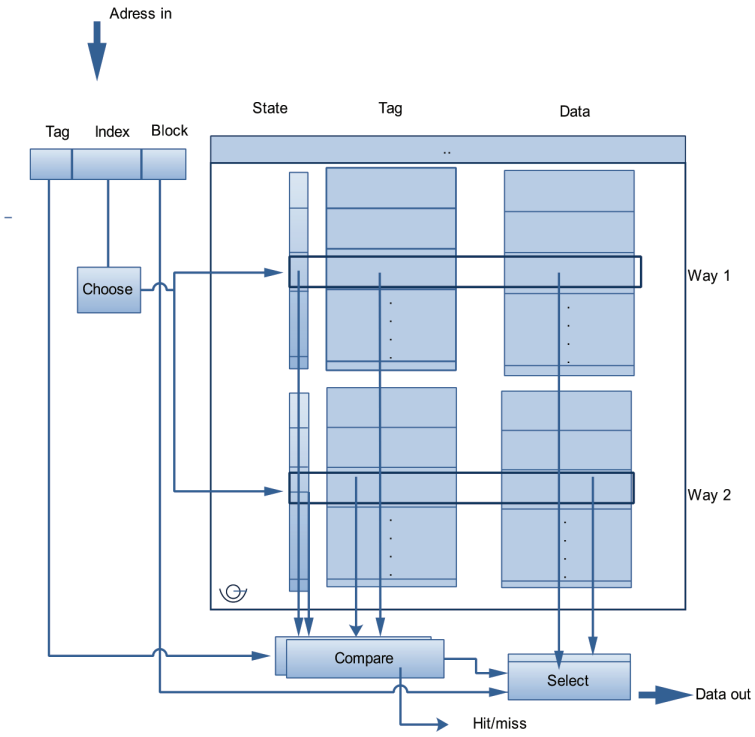

Figure 2.2: Conventional cache organization based on [1]

Detection mechanism can be either software or hardware controlled.

A software controlled approach can be controlled through the compiler. Alternately, a hardware controlled approach could use hardware support to automatically determine when and how to change the partitions. Ranganathan et. al [1] used a software controlled approach.

The results (using CACTI version 2.0 an analytical model of the cache access and cycle time model) show that a reconfigurable cache organization can increase the cache access time by between 1% and 15 %. They have concluded that the small number of partitions (2-way), reconfigurable caches usually increase the access time over traditional cache (non-configurable) by less than 5% (4% for a 128KB cache and 1% for a 1MB cache). While larger number of partitions (4-way and 8-way) have the ability to increase the cache access time. For smaller cache sizes (128KB cache) the increased access time is varying between 7-15% and 2-6% for the 1MB cache. They also evaluated media processing applications and obtained an IPC (inter-process communication) improvement ranging from 1.04X to 1.20X across eight different benchmarks [1].

Albonesi [4] developed a selective cache ways technique. Albonesi approach has the

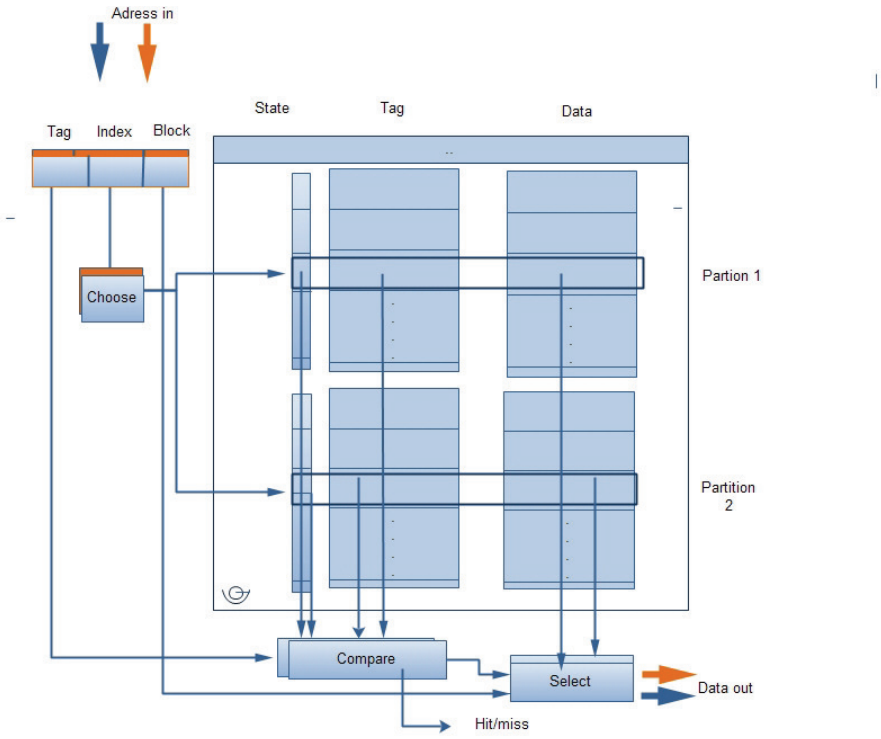

Figure 2.3: Reconfigurable cache organization based on [1]

ability to disable a subset of the ways in a set associative cache when this cache does it's work, while the full cache stay working for more cache-intensive periods. Albonesi attempted to reduce the cache energy wasting by utilizing a conventional set associative cache design and exploited the principle of cache partitioning. Figure 2.4 shows the conventional 4-way set associative cache. The data array is partitioned into four subarrays. Each subarray is associated with its own decoder and the sense amplifier.

Moreover, there are two tag subarrays and they are also associated with its own decoder and the sense amplifier.

A selective cache ways technique also partitions the data and the tag arrays into one or more subarrays for each cache way. This technique requires minor software modifications while several hardware modification is required [4]. The modifications on the original design are colored in orange as shown in Figure 2.5. This figure shows a 4-way set associative cache using selective cache ways. The ways 1-3 are similar to way 0 therefore they are not depicted in this figure.

According to Albonesi [4] the following are the configurations that are required for such design:

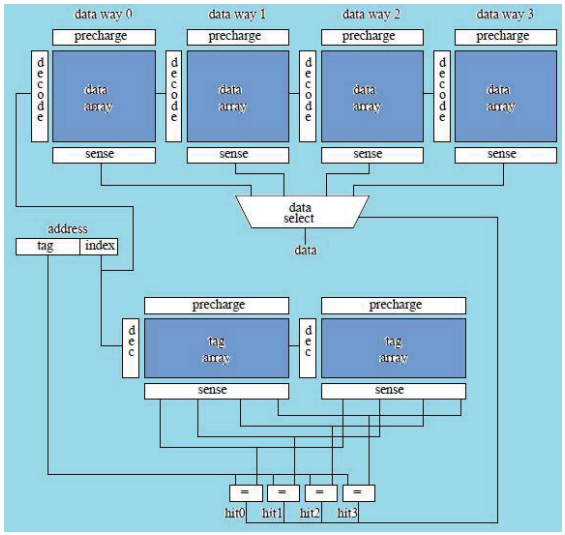

Figure 2.4: Conventional 4-way set associative cache based on [4]

- Decision logic and gating hardware for disabling the operation of particular ways.
- A software register, know as the Cache Way Select Register (CWSR), that inform the hardware to enable/disable specific ways with specific instructions, WRCWSR and RDCWSR, to write and read this register, respectively.

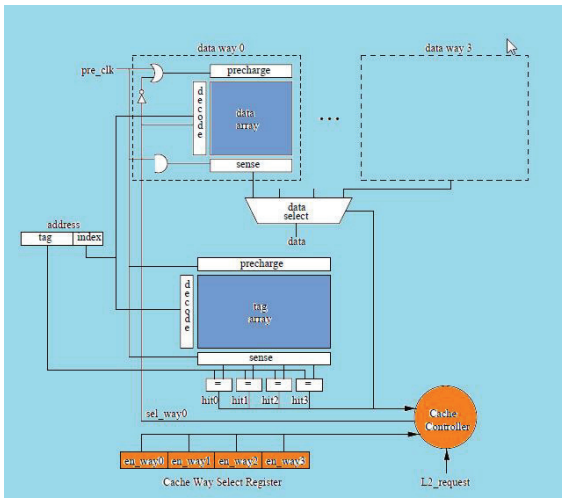

Figure 2.5: 4-way set associative cache using selective cache ways based on [4]

Albonesi stated that the selective cache partitioning technique can decrease the overall power consumption while produces high performance [7]. The results showed that a 40% reduction in overall cache energy wasting can be reported for 4-way set associative caches and less than a 2% overall performance degradation.

Adam Spanberger [7] developed the tournament caching technique. The tournament caching technique (TCA) has three modes of operation: normal mode, small tournament mode and large tournament mode.

The simple idea behind TCA is to compare the current cache configuration to either of these three modes configuration. These three different modes allow the cache to dynamically change its size to save power while maintaining performance.

Tournament caching reduces power consumption by shutting down parts of the cache without degrading performance. This technique utilizes a variable associativity parameter. Figure 2.6 depicts the working of the tournament cache technique. Each circle performs a mode of operation, and each arrow performs a transition between modes [7].

To grasp the idea of tournament cache technique we explain in detail the work of each mode. In normal mode, the tournament cache operates as a traditional cache would operate and it is the initial state in the state diagram (Figure 2.6). In this mode the number of consecutive misses is measured (miss saturation counter increases by 1 on a cache miss, decreases by 1 on a cache hit and it never goes below 0). This number (consecutive misses) suggests whether the cache enters the large tournament mode or the small tournament mode.

If the miss saturation counter (transition "E" in Figure 2.6) overcomes the max miss saturation, the cache begins a large tournament mode. Alternately, the cache begins small tournament mode if the tournament access counter (transition "A" in Figure 2.6) overcomes the accesses between tournaments.

In the large mode, the cache compares the current cache configuration to a larger cache configuration by enabling extra way and the number of hits in this extra way is measured. If the tournament hit counter (transition "F" in Figure 2.6) overcomes the hits to win, the cache reconfigures to the larger cache by enabling extra way. If the tournament access counter (transition "D" in Figure 2.6) overcomes the accesses between tournaments, disables this extra way and keeps the existing configuration and return to normal mode.

In the small mode, the number of cache hits in the least recently used (LRU) cache block is measured. If the tournament hit counter (transition "B" in Figure 2.6) overcomes the hits to win, the existing configuration will be hold and return to normal mode. Whilst, if the tournament access counter (transition "C" in Figure 2.6) overcomes the accesses between tournaments, it reconfigures to the smaller cache size by one less associative than the existing associative and return to normal mode.

The benchmark results showed that tournament caching in the L1 instruction cache decreased overall energy consumption by an average of 8.2% while the delay was increased by 0.25%.

Zhou et.al [5] developed an adaptive mode control technique. This design allows the processor to shut down specific lines of the cache in order to save power instead of shutting down large partitions like what Albonesi did [4]. Their model requires a line idle counter (LIC) and adaptive mode control to monitor accesses to each line of the cache. By monitoring this, they can dynamically shut down the lines that have not been used for a specified period [5]. Their approach shows large savings in power consumption, however, the LIC and the adaptive mode control to check the counters produced considerable overhead [7]. Table 2.1 shows the summary of the cache reconfigurable techniques of the

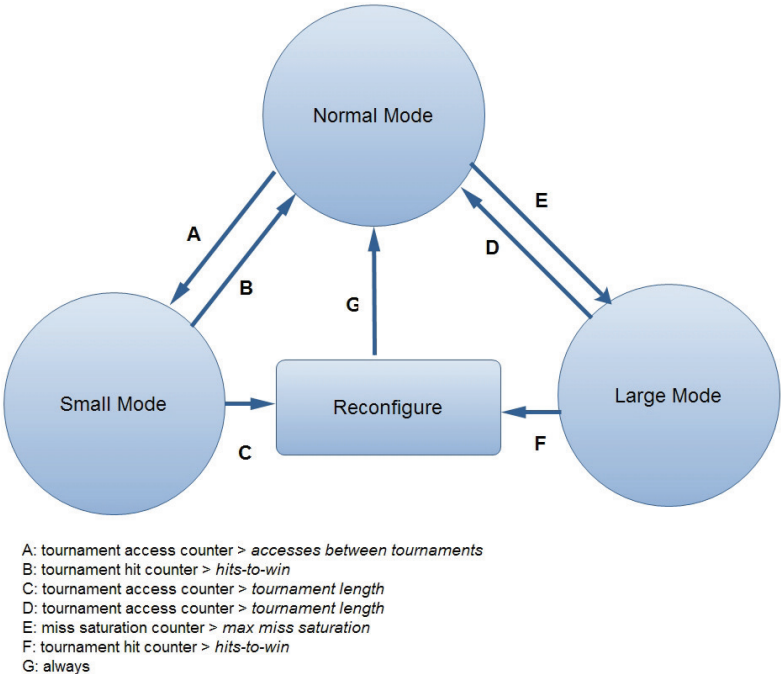

Figure 2.6: Tournament cache technique state digram based on[7]

researchers[1, 4, 7].

Table 2.1: Summary of different reconfigurable cache techniques

Researcher	Technique	Reconfigurable cache level	Addressing scheme	Data consistency technique	Performance	Power consumption	Detection mechanism when to reconfigure
Ranganathan et.al [1]	Associative-based partitioning	Any level of caches Trade off in terms of the size, granularity, access time, and usage of the partitions will determine the level to partition. [1] reconfigured L1 cache in their design	based on cache set-associative	cache scrubbing	Reco. cache can increase the cache access time by between 1% and 15 %	-	Software control
Almonesi [4]	Selective cache ways	-	based on cache set-associative	cache scrubbing	-	40% reduction in overall cache energy dissipation	Have special instructions inserted in the code that explicitly changes the configuration
Adam Spanberger [7]	Tournament caching technique	[7] reconfigured L1 cache in their design	based on cache set-associative	-	achieve optimal high performance by minimizing the energy cache consumption	decreased overall energy consumption by an average of 8.2%	tournament three modes scheme to detect when reconfigure

Bibliography

[1] Ranganathan, Parthasarathy, Sarita Adve, and Norman Jouppi. Reconfigurable Caches and their Application to Media Processing. Proceedings of the 27th International Symposium on Computer Architecture 2000.

[2] Chiou, Derek, Larry Rudoplh, Srinivas Devadas and Boon Ang. Dynamic Cache Partitioning via Columnization. MIT Laboratory for Computer Science Computation Structures Group. 1999.

[3] G. Edward Suh, Larry Rudolph and Srinivas Devadas Laboratory for Computer Science. MIT. Dynamic Cache Partitioning for Simultaneous Multithreading Systems. 2001.

[4] Albonesi, David. Selective Cache Ways: On-Demand Cache Resource Allocation. Journal of Instruction-Level Parallelism. 2000.

[5] Zhou, Huiyang, Mark Toburen, Eric Rotenburg, and Thomas Conte. Adaptive Mode Control: A Static-Power-Efficient Cache Design. Proceedings of the International Conference on Parallel Architectures and Compilation Techniques 2001.

[6] Matthew Ziegler, Adam Spanberger, Ganesh Pai, Mircea Stan, Kevin Skadron. Dynamic Way Allocation for High Performance, Low Power Caches. Departments of ECE and Computer Science, University of Virginia.

[7] Thesis: Adam Spanberger. Designing a Dynamically Reconfigurable Cache for High Performance and Low Power. 2002.

[8] Chen, L., Zou, X., Lei, J., Liu, Z. Dynamically Reconfigurable Cache for Low-Power Embedded System. Third International Conference on Natural Computation. 2007.

[9] Santana Gil, A. D, Benavides Benitez, J.I., Hernandez C., M. Herruzo G., E. Reconfigurable Cache Implemented on an FPGA. International Conference on Reconfigurable Computing. 2010.

[10] Book: Heuring, Vincent and Harry Jordan. Computer Systems Design and Architecture. Massachusetts: Addison-Wesley, 1997.

[11] Richard C. Murphy, Peter M. Kogge, and Arun Rodrigues. The Characterization of Data Intensive Memory Workloads on Distributed PIM Systems.

[12] Jeffrey Draper and J. Tim Barrett and Jeff Sondeen and Sumit Mediratta and Chang Woo Kang and Ihn Kim and Gokhan Daglikoca. A Prototype Processing-In-Memory (PIM) Chip for the Data-Intensive Architecture (DIVA) System, 2002.

[13] Blaise Barney, Lawrence Livermore. Introduction to Parallel Computing. National Laboratory

[14] Anthony M. Middleton, Ph.D. LexisNexis Risk Solutions. HPCC Systems: Data Intensive Supercomputing Solutions. April 2011.

[15] Gilad Shainer, Brian Sparks, Richard Graham. Towards Exasclae computing. HPC Advisory Council.

[16] Tibken, Shara. Titan supercomputer debuts for open scientific research Cutting Edge - CNET News. News.cnet.com. Retrieved 2013-02-28.

[17] TOP500 Supercomputer Sites. http://top500.org/blog/lists/2012/11/press-release/. Retrieved 2013-03-22.

[18] NVIDIA. http://www.nvidia.co.uk/page/home.html

[19] AMD. http://www.amd.com/uk/Pages/AMDHomePage.aspx

[20] BOOK: Palma, J.M.L.M.; Dayd, M.; Marques, O.; Lopes, J.C.. High Performance Computing for Computational Science – VECPAR 2010: 9th

[21] GPU. http://www.wikipedia.org/wiki/GPU

3 Big Data in bioinformatics

We have seen in Section 1.4 the potential benefits of Big Data in several domains. To explore how Big Data can create potential value and show the technical challenges accompanied with Big Data applications, we have decided to choose an application from the bioinformatics domain: the Smith Waterman genetic database search tool. The continuous increase in the volume of data in genetic databases leads to the exponential increase in the time required for comparing these genetic sequences. The chapter starts with an overview of the basics of molecular biology. Basic knowledge of the subjects such as DNA and protein construction are important to understand. Such knowledge is important to understand the relevance of sequence alignment: the algorithm of the Smith Waterman database search tool for finding optimal sequence alignments. Much of this thesis is based on this algorithm.

Bioinformatics is one of the relevant Big Data application domains. Bioinformatics is a field that develops computational techniques to analyze the information of biological data (molecular data). There are three major aims of bioinformatics [19]: First, bioinformatics arranges data that helps the researchers to find the required data and deliver the new produced data. For example, 3D model simulation is used to visualize biological structures. The second aim is to produce software tools that help the researchers in the analysis of the relevant data. For example, having a DNA query sequence, it will be interested to find and compare with existing reported and analyzed sequences. FASTA and SSearch are tools developed for the alignment of genetic sequences. The third aim is to use these tools to analyze the results.

In molecular biology, bioinformatics defines itself as a discipline [19] and it includes several subject domains such as structural biology, gene expression and genetics studies [20]. In the sector of the structural biology, bioinformatics aids in the simulation and modeling of protein structures, molecular interactions and DNA and RNA structures. While in the sector of genetics it helps by understanding and analyzing the gene and protein sequencing by using the software tools and reporting any relevant observed information.

"Biological data are being produced at a phenomenal rate" [1, 19] which results in enormous quantity and variety of information that is being produced. For example as of April 2013, the GenBank repository of nucleic acid sequences contains about 164 millions entries [3] and the Swiss-Prot database of protein sequences contains about (release May 2013) 540 thousand sequence entries [4]. According to [5] these databases, on average, are doubling in size every 15 months. As a result of this burst in data, computers have become substantial to biological research as they are ideal to process huge amount of data.

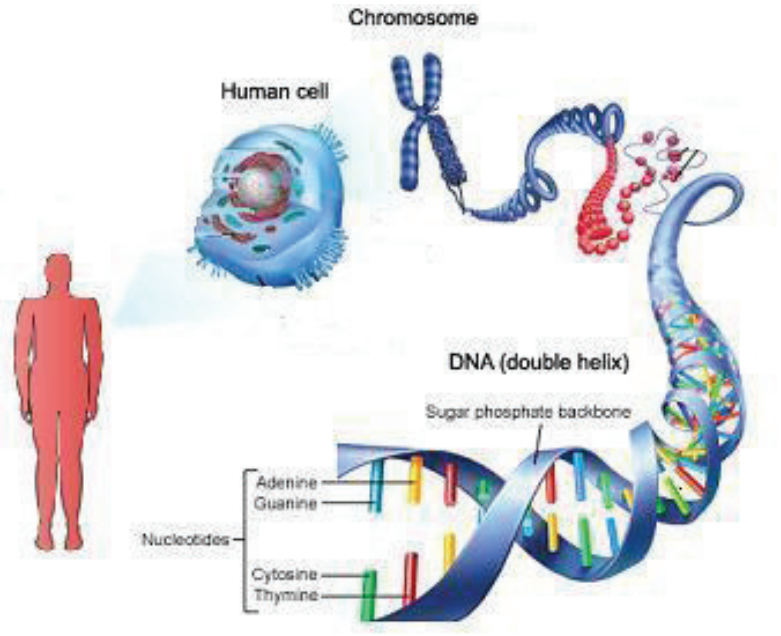

Figure 3.1: Representation of cell, chromosome and DNA based on [12]

3.1 Cells, DNA and proteins

Cells are "the structural and functional units of all life forms" [17]. Organisms such as bacteria consist of a single cell while humans have around 75 trillion cells. Cells vary in appearance, size, complexity and functionality Figure 3.1. Virtually all cells of organisms share a common component: genetic information DNA (Deoxyribo Nucleic Acid). DNA is a double helix shaped molecule. On each backbone there are four types of nucleotides, namely Adenine (A), Thymine (T), Cytosine (C) and Guanine (G). Adenine is always paired with thymine; while cytosine couples with guanine. DNA is represented as a sequence of the alphabet (A, C, G, T) [17].

The presence of proteins is important for cells to survive and reproduce. Proteins are created within the cell itself after a complex biological operation. This operation is directed by the genes producing those proteins. Humans have approximately between 20,000 and 25,000 different genes. The produced protein can consist of 20 types of amino acids and can be represented as strings of an alphabet {A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y}. The table below shows the standard amino acid abbreviations which corresponds to these letters.

Table 3.1: Standard 20 amino acid abbreviations

Amino Acid	3-letters	1-letter	Amino Acid	3-letters	1-letter
Alanine	Ala	A	Arginine	Arg	R
Asparagine	Asn	N	Aspartic acid	Asp	D
Cysteine	Cys	C	Glutamic acid	Glu	E
Glutamine	Gln	Q	Glycine	Gly	G
Histidine	His	H	Isoleucine	Ile	I
Leucine	Leu	L	Lysine	Lys	K
Methionine	Met	M	Phenylalanine	Phe	F
Proline	Pro	P	Serine	Ser	S
Threonine	Thr	T	Tryptophan	Trp	W
Tyrosine	Tyr	Y	Valine	Val	V

3.2 Genetic sequences alignment

Genetic sequence alignment is defined as an arrangement of 2 or more sequences (DNA, proteins or others) to identify the similar regions between these sequences and to indicate the genetic relatedness between the organisms [17]. Typically, the similarity or differences between these sequences may be a consequence of functional, structural or evolutionary relationship between the sequences [14]. The similar sequences, in turn, share a common ancestral sequence and their relative differences are the result of mutations. To maximize the degree of similarity and minimize the mismatch between two sequences, inserts, deletions or substitutions (known by the term indel) can be used. The following is an example of two possible alignments. The base sequences are ACACACTA and AGCACACA:

Alignment 1 (without gap)
A C A C A C T A
A G C A C A C A

Alignment 2 (with gap)
T A C C A G T - -
- A G C A C A C A

We can see that the alignment with gaps is more relevant to the similarities among the sequences. In the first alignment (without gaps), there are two similar items in the sequences which are A in the beginning and A in the end of the sequences. However, in the second alignment (with gaps) three similar items A, C and A are aligned. Note that, many other alignments are possible. In the following alignment (alignment 3) 7 similar

items are aligned.

Alignment 3
A - C A C A C T A
A G C A C A C - A

Obviously it makes sense to devise a way to rate and then select the best alignment(s). An easy way to achieve this is using a scoring scheme. The best match is defined using the formula bellow:

$\sum_{i=1}^{L} S(X(i), Y(i))$

Where L is the length of the alignment, S is a scoring function, and X(i) and Y(i) are the aligned sequences

The parameters can be set as follows:

- if two items are identical a match occurs and the scoring function S(X(i),Y(i)) gives a value of, for example, +2.
- if two items are different a mismatch occurs and the scoring function S(X(i),Y(i)) gives a value of, for example, -1.
- for gap opening the scoring function S (X(i), -) or S (-, Y(i)) gives a value of, for example, -3.

Using the aforementioned scoring system, the first none gapped alignment scores +2-1-1-1-1-1-1+2=-2, the second gapped alignment scores -3+2-1+2+2-1-1-3-3=-6 and the last alignment scores +2-3+2+2+2+2+2-3+2=8. We can see that the score for alignment 3 is 8 which is larger than the score of alignment 2 which is -6. Alignment 3 achieves a better matching.

Scoring scheme for DNA can be performed out of 4 letters and 20 letters for proteins (amino acids). For amino acids a standard 20 * 20 triangular substitution scoring matrix such as BLOSUM62 or PAM is used as shown in Figure 3.2.

3.3 DNA and protein databases

Many databases and search engines for biological data, such as protein and DNA sequences, are available. The major DNA database sequences are the International Nucleotide Sequence Databases (INSD). INSD have been developed and maintained collaboratively between DNA Data Bank of Japan (Japan), GenBank (USA) and the European Nucleotide Archive (UK) for over 18 years

UniProt (Universal Protein Resource) is a protein sequence database; it contains no DNA data. Uniprot contains two sub-databases: Swiss-Prot and TrEMBL. The search engine in the UniProt database is SSEARCH. Our focus in this thesis is on the Smith Waterman search tool for protein sequences. During the benchmarking in this thesis we compare the query sequences with Uniprot database sequences and use SSEARCH as reference for correctness of the maximum score of Smith Waterman for relevant sequences. UniProt/Swiss-Prot release June 2013 contains 540261 sequence

	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val
Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 3.2: Blosum62 scoring matrix

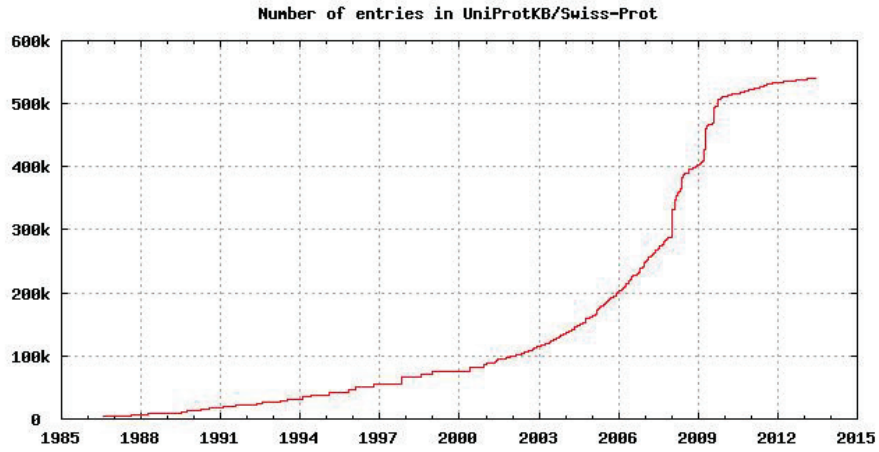

Figure 3.3: UniProt entries based on [15]

entries (Figure 3.3). The average sequence length in UniProt/Swiss-Prot is 355 amino acids. The shortest sequence is 2 amino acids and the longest sequence is 35213 amino acids (see Figure 3.4).

Smith Waterman sequence alignment algorithm

The Smith Waterman algorithm was first proposed by Temple Smith and Michael Waterman in 1981 [8]. The Smith Waterman algorithm compares and aligns common regions of similarity between two strings of nucleotide (DNA) or protein sequences (amino acid) [8]. Instead of looking at the complete sequence, the Smith Waterman method performs optimal local alignment between two sequences. This algorithm is used by biologists to search meaningful sequences in biological databases.

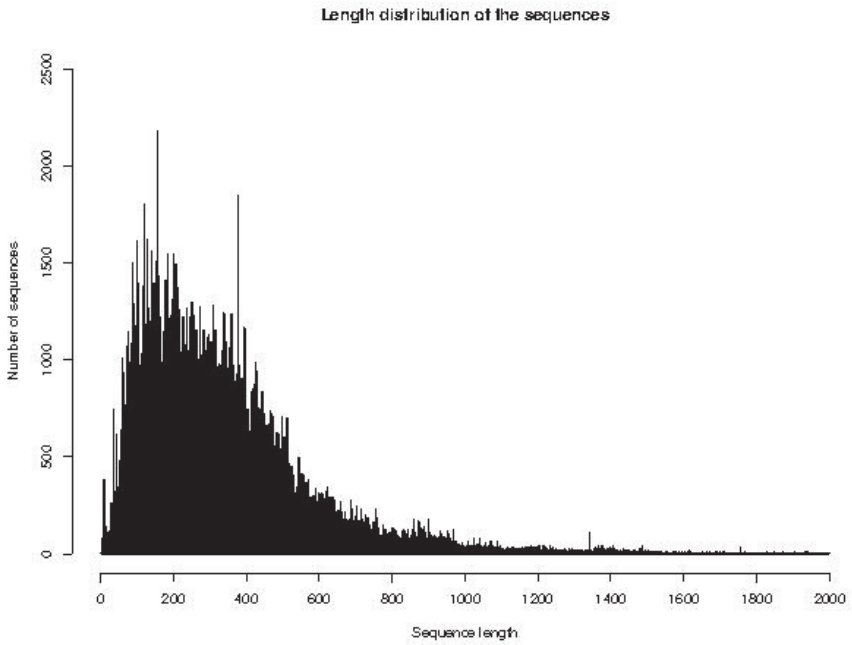

Figure 3.4: Sequence length based on [15]

The Smith Waterman algorithm basically proposed a linear gap penalty function which assigns fixed gap penalty parameters for aligning different residue positions. One year later (1982), Gotoh [10, 11] had proposed an affine gap penalty function for the Smith Waterman algorithm. The affine gap penalty function assigns an initial penalty for a gap opening, and an additional penalty for gap extensions.

The optimized Smith Waterman algorithm [8] can be gained computationally using Dynamic Programming (DP) [9]. Smith Waterman algorithm has three phases: initialization, matrix fill and trace back. Consider a matrix H constructed with a *query* sequence of length m lined up against the columns of a matrix, and a *database* sequence of length n lined up against the rows. With the equations bellow, the H matrix will be built.

Initialization

The top row and leftmost column of matrix H are initialized to 0. Then a matrix H has (m + 1) * (n + 1) dimensions. Figure 3.5 depicts the initialization step.

$H_{i,0} = 0, i \leq n$
$H_{0,j} = 0, j \leq m$

Matrix Fill

The H matrix is filled by calculating score for each cell with the recurrent Equations 3.1, 3.2 and 3.3. Three separate scores are calculated: the F (Equation 3.2) matrix is the score resulting from vertical (upper) element with gap penalty and E matrix (Equation 3.3) is the score resulting from horizontal (left) element with gap penalty and W is a match/mismatch (diagonal) score (similarity score) between the intersecting elements of the query and database sequences (W is the substitution matrix such as Blosum62 for protein alignment). The G_{init} and G_{ext} are the gap opening and extension penalties, respectively. The maximum of these three scores (or a 0) is assigned to the cell. The matrix fill step is the most computationally intensive task, accelerating Smith Waterman is mainly proposed by speeding up this step.

$$H_{i,j} = max \begin{cases} 0 \\ E_{i,j}, \\ F_{i,j}, \\ H_{i-1,j-1} + W(q_i, d_j) \end{cases} \tag{3.1}$$

$$F_{i,j} = max \begin{cases} F_{i-1,j} - G_{ext}, \\ H_{i-1,j} - G_{init} \end{cases} \tag{3.2}$$

$$E_{i,j} = max \begin{cases} E_{i,j-1} - G_{ext}, \\ H_{i,j-1} - G_{init}, \end{cases} \tag{3.3}$$

Trace Back

After the H matrix is completely filled, the trace back step starts at the maximum value in the matrix. The optimal local alignment can be found by tracing back the path resulting in the maximum value in the matrix. This step will be terminated if 0 is reached in the matrix. The resulting alignment depends on the path. A diagonal trace back results in a match, a vertical trace back results in an indel insert in the database sequence, a horizontal trace back results in an indel insert in the query sequence.

The example bellow shows the three steps of Smith Waterman algorithm for two sequences: the query sequence is AGCACACA and the database sequence is ACACACTA. The match value is 2, the mismatch is -1, the G_{init} is -1 and the G_{ext} is -1. The best alignment is
A-CACACTA
AGCACAC-A

All three matrices E, F and H, when they are calculated, are dependent on each other (Figure 3.6, 3.7 and 3.8). Each cell in the matrix F is required to compare the maximum values between the previous calculated cell in F, and the previous calculated cell in H. The same scenario is for the E matrix, every cell in the E matrix is required to compare the maximum values between the previous by calculated cell in E and the previous calculated cell in the H matrix. In the Figure 3.8 we can see the trace back starts at the cell (8,8) which contains the maximum number in the matrix which is 12

		A	C	A	C	A	C	T	A
	0	0	0	0	0	0	0	0	0
A	0								
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

Figure 3.5: Matrix initialization

		A	C	A	C	A	C	T	A
	0	0	0	0	0	0	0	0	0
A	0	-1	1	0	1	0	1	0	-1
G	0	-1	0	0	0	0	0	0	-1
C	0	-1	-1	2	1	2	1	2	1
A	0	-1	1	1	4	3	4	3	2
C	0	-1	0	3	3	6	5	6	5
A	0	-1	1	2	5	5	8	7	6
C	0	-1	0	3	4	7	7	10	9
A	0	-1	1	2	5	6	9	9	9

Figure 3.6: The F matrix

and trace it back to the cell (7,7) because it comes from the diagonal (for convenient trace back the letters H, F, E are inserted beside each number in the cells to indicate from where the maximum filled number came).

		A	C	A	C	A	C	T	A
	0	0	0	0	0	0	0	0	0
A	0	-1	-1	-1	-1	-1	-1	-1	-1
G	0	1	0	1	0	1	0	-1	1
C	0	0	0	0	0	0	0	-1	0
A	0	-1	2	1	2	1	2	1	0
C	0	1	1	4	3	4	3	2	3
A	0	0	3	3	6	5	6	5	4
C	0	1	2	5	5	8	7	6	7
A	0	0	3	4	7	7	10	9	8

Figure 3.7: The E matrix

		A	C	A	C	A	C	T	A
	H ↘ 0	E↓ 0	0	0	0	0	0	0	0
A	F→ 0 0	H 2	1	2	1	2	1	0	2
G	0	E 1	1	1	1	1	1	0	1
C	0	0	H 3	2	3	2	3	2	1
A	0	2	2	H 5	4	5	4	3	4
C	0	1	4	4	H 7	6	7	6	5
A	0	2	3	6	6	H 9	8	7	8
C	0	1	4	5	8	8	H 11	F 10	9
A	0	2	3	6	7	10	10	10	H 12

Figure 3.8: The H matrix

Bibliography

[1] Reichhardt T. It's sink or swim as a tidal wave of data approaches. Nature 1999.

[2] Nabeel Ahmad1, Akhilesh Bind, and Sanjiv Kumar Maheshwari. Bioinformatics New Era: Introduction and Overview.Plant Molecular Biology Lab, Department of Biotechnology, College of Engineering and Technology, IFTM Campus.India .

[3] http://www.ncbi.nlm.nih.gov/genbank/statistics.

[4] http://web.expasy.org/docs/relnotes/relstat.html.

[5] Benson DA, et al. GenBank. Nucleic Acids Res 2000.

[6] Bernstein FC, et al. The Protein Data Bank. A computer-based archival file for macromolecular structures. Eur J Biochem 1977; 80 (2): 319-24.

[7] Berman HM, et al. The Protein Data Bank. Nucleic Acids Res 2000.

[8] T. F. SMITE AND M. S. WATERM. Identification of Common Molecular Subsequences. 1981.

[9] R. Giegerich. A systematic approach to dynamic programming in bioinformatics. Bioinformatics, vol. 16, pp: 665-677.

[10] Gotoh, O. An improved algorithm for matching biological sequences. Journal of Molecular Biology, 162 (3), 705-708.

[11] Rolf Backofen. Sequence Alignment Gap Penalties. Gotoh's Algorithm and Smith Waterman's Local Alignment.

[12] http://www.virtualmedicalcentre.com/anatomy/dna-deoxyribonucleic-acid/37C41.

[13] Rognes T. Smith waterman database searchs with inter-sequnces SIMD parallelisation. 2011.

[14] D. M. Mount, Bioinformatics: Sequence and Genome Analysis, Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 2nd ed., 2004.

[15] UniProtKB/Swiss-Prot protein knowledgebase release 2013-06 statistics. http://web.expasy.org/docs/relnotes/relstat.html.

[16] Ali Khajeh-Saeed, Stephen Poole, J. Blair Perot. Acceleration of the Smith-Waterman algorithm using single and multiple graphics processors.2010

[17] Book: William J., Thieman and Micheal A.Introduction to Biotechnology. Second edition.

[18] http://en.wikipedia.org/wiki/SmithWaterman.

[19] N. M. Luscombe, D. Greenbaum, M. Gerstein. What is Bioinformatics? A Proposed Definition and Overview of the Field. Department of Molecular Biophysics and Biochemistry Yale University, New Haven, USA. 2001.

[20] Bioinformatics. http://en.wikipedia.org/wiki/Bioinformatics

Implementation of Smith Waterman algorithm

4

This chapter explains different implementations of the Smith Waterman algorithm. First, we take a look at our sequential implementation of the Smith Waterman algorithm and how the Smith Waterman algorithm can be parallelized. Then, the accelerated implementations of Smith Waterman using Streaming SIMD Extensions (SSE) and GPU are presented. We also present SSEARCH and DOPA which are accelerated and optimized implementations of the Smith Waterman algorithm using SSE and GPU, respectively.

4.1 Sequential implementation

This Section discusses the steps taken to produce the straightforward sequential implementation of the Smith Waterman algorithm. We used a sequential way of calculating the cell matrix values. In order to compare protein sequences, the BlOSUM62 substitution matrix is implemented into our code.

The substitution matrix BlOSUM62 is implemented as a 26 * 26 matrix which includes all the alphabet characters. By using the ASCII code of each character in sequence we can easily find the required indexes (Figure 3.2).

We have implemented [13] the algorithm for protein sequences so that the maximum score of optimal alignments can be found. An affine gap penalty version of Smith Waterman has been implemented. The implementation returns maximum score of the similarity matrix of the Smith Waterman algorithm not the actual alignments. Optionally our implementation returns the results matrix, as well as the performance measured in GCUPs (Giga Cell Updates Per Second) and length of a sequence.

The program starts by reading the query sequence and database sequence files. The query sequence includes only one sequence while the database sequence file includes multiple sequences. The format of these files is in FASTA format. The FASTA format is a text-based format for representing nucleotide sequences in which nucleotides are represented using single-letter codes. Each sequence has a single description line which is denoted by the first character of the greater than > sign. The end of the description line is denoted by a return-line.

As we see in Figure 4.1 and the Equations 3.1, 3.2, 3.3 we need at least three matrices: E, F, and H to calculate the algorithm. If the length of database and query sequences is large, this will consume a lot of memory space. As we have mentioned before our goal is simply to find the maximum score not the actual alignments, so the values of these matrices are temporary. Therefore, to reduce memory usage, only the current value of the E matrix is saved in a variable, and the previous values are overwritten.

After reading the query and database sequences the computation starts with the initialization of the E, F and H (E i,j = F i,j = H i,j = 0 for all i = 0 or j = 0) to 0.

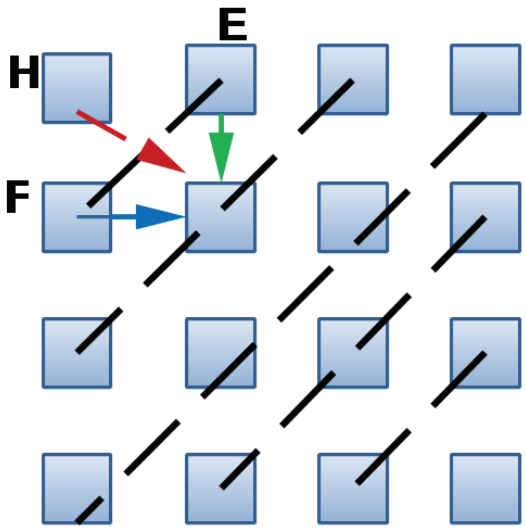

Figure 4.1: Data dependency of the Smith Waterman algorithm

To compute matrix H (Figure 4.1) we start at i=1 and j=1 then scan through the end of the row, then repeat this procedure for the next row until we reach to the last element in the matrix. In every iteration the elements in the F matrix always use the element from its left (previous) side element, while the elements in the E matrix always use the element from its upper position element. Obviously, after the left and upper old values are used, these will never be used again. Therefore, we make a variable to store these temporary values. The H matrix will require the value from vertical (E), horizontal (F) and diagonal directions. And each computed score will be used at most 3 times. So we can also make them temporary. This specific implementation will align a query sequence to every database sequences, effectively offering no parallelism at all.

In the sequential Smith Waterman algorithm the (intermediate) data are stored sequentially in the memory. In Figure 4.2 we can see how the cells for the H matrix are stored in the memory. The arrows indicate the locations in the memory for two rows. This sequential order is a consequence of the sequential nature of the Smith Waterman algorithm, because in the H matrix the cells are computed and stored one by one.

4.2 Streaming SIMD Extensions (SSE)

In this section we will take a look at how to parallelize the Smith Waterman algorithm since parallelization is an important aspect to accelerate the performance of database search alignment. After that we present our straightforward SSE implementation [13]. Then we provide an overview of SSearch an optimized CPU implementation using SSE. To parallelize the Smith Waterman algorithm two different forms of granularities can be performed: fine-grained and coarse-grained parallelism.

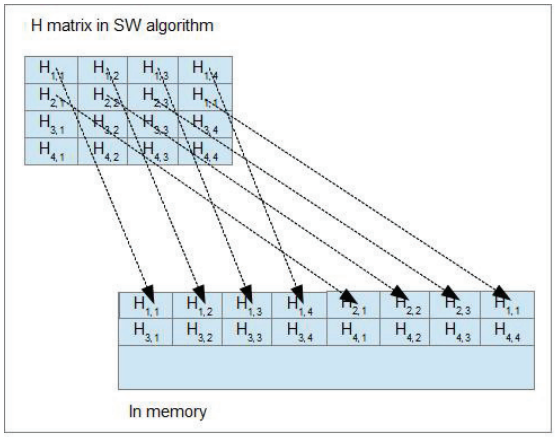

Figure 4.2: Sequential order memory

Fine-grain parallelism

In pair-wise sequence alignment, the query sequence is compared against one database sequence and it is also called intra-task parallelization. A regular data dependency in Smith Waterman algorithm is implied due to the recurrence of the Equations 3.1, 3.2 and 3.3. Compute $H_{i,j}$ (Equation 3.1) depends on its left previous cell neighbors F $(_{i,j-1})$, H $(_{i,j-1})$, upper neighbor E$(_{i-1,j})$, H$(_{i-1,j})$, and upper left neighbor H$(_{i-1,j-1})$. Fine-grain means that processors will work together in computing the H matrix, cell by cell [3]. In Figure 4.1 the arrows represent the data dependency in the Smith Waterman algorithm and the black dashed lines represent the elements that can be computed in parallel.

Matrix fill step of the Smith Waterman algorithm can be performed along rows or along columns of the H matrix. If the matrix is filled along rows or along columns then the computation will be performed serially. However, we can see in Figure 4.1 the elements that are located in anti-diagonals have no dependency among each other and therefore, all cells along the anti-diagonal can be computed in parallel using the previously computed anti-diagonal cells.

Coarse-grain parallelism

Aligning a single query against several database sequences in parallel is known as inter-task parallelization or coarse-grained parallelization. The pair-wise alignment can be performed independently [2, 3] and the query sequence and the substitution matrix are shared data. Intra and inter -task parallelization give rise to many hardware platforms implementations, like SIMD and GPU. The following section presents our implementation of streaming SIMD extensions for the Smith Waterman algorithm.

4.2.1 Straightforward SSE2 implementation

Streaming SIMD Extensions (SSE) is a method provided by modern processors to accelerate computation intensive programs. SSE supports several types of instructions in-

cluding instructions for adding, subtracting and computing the maximum. Those three instruction types are required in the Smith Waterman algorithm, therefore in principle the Smith Waterman algorithm can be implemented with SSE. In Streaming SIMD Extensions 2 (SSE2) the data type "short" is supported. In the Smith Waterman algorithm this data type "short" is preferred because it allows eights processing elements simultaneously (Figure 4.3 (c)) being processed. This is a consequence of the fact that there is a 128-bit register file available for SSE instructions. Using "short" as the data type in the Smith Waterman algorithm limits the maximum score not to be larger than 65536 however, the longest sequence in Swiss-Prot 35213 proteins and most sequences being much smaller, making this acceptable. Figure 4.3 shows the different organization models for the register file in SSE2.

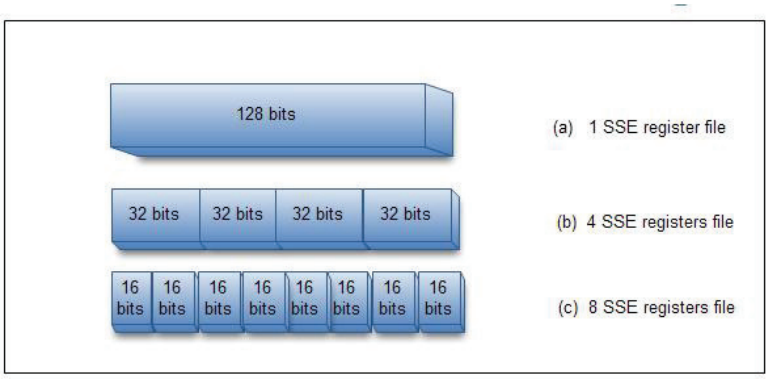

Figure 4.3: SSE registers file

We propose in our implementation [13] inter-task parallelism for the Smith-Waterman algorithm. Instead of aligning query sequence against one database sequence at a time, items from eight multiple database sequences are retrieved and processed in parallel.

In order to achieve higher performance using SIMD instructions the data are stored in memory in an interleaved way. When using parallel hardware the sequential order is not convenient to use anymore because of the parallel hardware architecture. Figure 4.4 shows the interleaved order with two processing elements. In Figure 4.4 we can see the H matrices for two different alignments where the first index in the subscript represents the alignment number. The first cells from both alignments are sequentially stored in the memory and then the second cells from both alignments are stored and so forth. The arrows indicate the locations in the memory for the first rows. This interleaved order can be extended for SSE2 with 8 processing elements.

4.2.2 Optimized SSE2 implementation

In this section we present the optimized method using SIMD instructions for Smith Waterman CPU implementation which is known as the SSearch implementation [8]. The SSearch implementation is an intra-task optimization approach which enables SIMD registers to access values parallel to query sequence in a striped pattern. See Figure 4.5. Three major techniques are proposed by SSearch optimization: striped query profile,

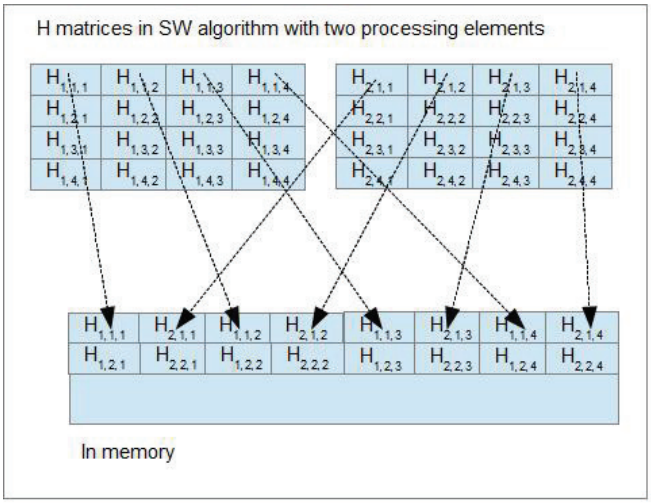

Figure 4.4: Interleaved order in the memory

SWAT optimizations [9] and lazy F evaluation. Following we briefly explain these techniques.

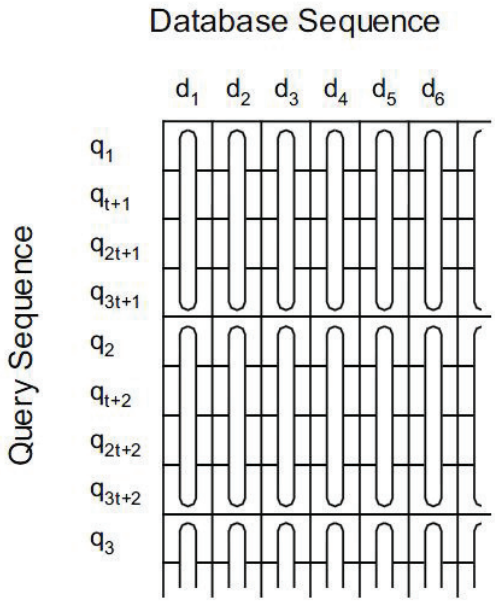

Figure 4.5: SSearch stripped pattern based on [8]

Striped query profile

Striped optimization applied the similar approach of [9] by pre calculating the query profile but with different layout. The query profile layout is a striped access parallel to the query sequence (see Figure 4.5). The query is divided into equal length segments,

S (Figure 4.6). The number of segments is equal to the number of elements allowed in a SIMD register. The major improvement when using striped query profile that data dependencies are moved out of the inner loop and done just once in the outer loop [8].

$$
\begin{array}{lccccc}
S_1 = & q_1 & q_2 & q_3 & \cdots & q_t \\
S_2 = & q_{t+1} & q_{t+2} & q_{t+3} & \cdots & q_{2t} \\
S_3 = & q_{2t+1} & q_{2t+2} & q_{2t+3} & \cdots & q_{3t} \\
S_4 = & q_{3t+1} & q_{3t+2} & q_{3t+3} & \cdots & q_{4t} \\
S_5 = & q_{4t+1} & q_{4t+2} & q_{4t+3} & \cdots & q_{5t} \\
S_6 = & q_{5t+1} & q_{5t+2} & q_{5t+3} & \cdots & q_{6t} \\
S_7 = & q_{6t+1} & q_{6t+2} & q_{6t+3} & \cdots & q_{7t} \\
S_8 = & q_{7t+1} & q_{7t+2} & q_{7t+3} & \cdots & q_{8t}
\end{array}
$$

Figure 4.6: Equal length segments based on [8]

SWAT optimizations

SSearch optimization considered SWAT optimization. SWAT optimization is an improved approach of the Smith Waterman algorithm [10]. Most cells values in the matrix F and E (Equations 3.2 and 3.3 respectively) are zero; consequently do not contribute to H (Equation 3.1). According to [10] as long as $H_{i,j}$ is not larger than the threshold G_{init} + G_{ext}, E and F will stay at zero along a column or row in the matrix. Mostly the cells values are not above G_{init} + G_{ext}. If all values in the SIMD registers are not above the threshold then F can basically be ignored in the computation of H. Therefore, removing data dependency and significantly simplifying the computations. The cells in the SIMD registers can be checked simultaneously whether any of them are above the threshold. If one or more of the register cell exceeds this threshold, then F values must be recalculated and taking H values into consideration. The SSearch optimization moves this correction step outside the inner loop. SSearchs method for F correction is presented in the following section.

Lazy F evaluation

If $H_{i,j}$ is larger than G_{init} + G_{ext}, then F should be considered in the calculations of H. The computation of the lazy F evaluation starts after the inner loop to compute H is completed. F is checked against the value of H by access the loop of Lazy F. The value of F are shifted to the left by one (see Figure 4.7) and if any elements are greater than the values of H - G_{init}, then H is recalculated because F can change the value of H [8].

Advantage of this technique is that all conditional branches are moved out of the inner loop to the outer loop therefore it reduces the impact of branching on the runtime.

4.3 GPU implementation

The GPU is a platform that is providing a high speed optimized implementation for Smith Waterman algorithm. The following section focuses mainly on our straightforward GPU implementation and DOPA optimizations.

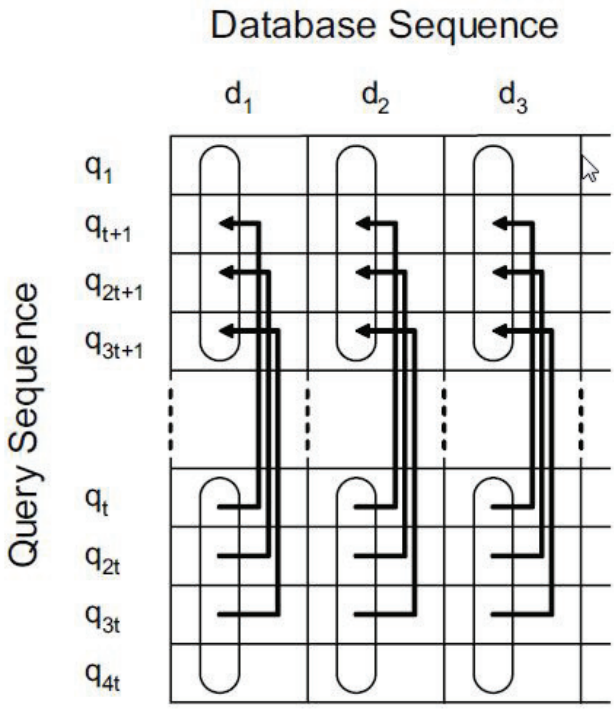

Figure 4.7: Lazy F based on [8]

4.3.1 Straightforward GPU implementation

Our GPU implementation [13] is also proposed inter-task parallelism for the Smith Waterman algorithm. We assigned each thread only one sequence from the database. Therefore, there is no dependency among all the threads. The task of each thread is exactly as in the sequential as well as in the SSE2 implementation, which is to calculate the maximal score. Before we process the calculation, we transform the database sequences from characters into integers hence we can only copy the integer array of database to device. Gap penalty and gap extension values are temporary variables stored in shared memory instead of registers to decrease register pressure. Usually, the maximal score will not be larger than 65536 (see Section 4.2.1). This makes it possible to use data type "short" for temporary variables. The "short" data type only has 2 bytes while "int" data type has 4 bytes. So the access time of "short" data type will be shorter. This is also an optimization to memory access. Although the database sequence of each thread is different and independent, the query profiling is the same for all database sequences. Thus we implemented query profiles and allocated them to texture memory. It is a read-only memory and it is faster than global memory. And it can be accessed by all the threads within the grid. Its advantage is that it is able to cache data from "near" position. So the read can be fast. We calculate the query profile data in CPU then copy them to GPU's global memory. And finally, we bind the query profile data on global memory to texture memory. After the calculation of Smith Waterman kernel is done on the device, the result maximum score is copied back to the CPU. We flexibly specify

number of blocks and fixed the number of threads. Since the total number of database sequences is constant, therefore, the required number of blocks is finding by dividing the number of database sequences over number of threads per block.

4.3.2 Optimized GPU implementation

DOPA is an optimized implementation for the Smith Waterman algorithm implemented for GeForce GTX 275. The optimizations of DOPA mainly deal with smart memory utilization by reducing the number of memory accesses. DOPA improves performance by optimizing the database organization as well as efficient way for work load distribution. DOPA returns few of the highest scoring sequences of the database without performing the trace back step on GPU. The following presents the implementation optimization approaches which is considered by DOPA.

Database conversion

The FASTA format database is converted to a format that better suits the GPU capabilities. The converting process needs to be done once. This process involves separate the sequence descriptions and store in other file, this file is not uploaded to the GPU and therefore saving memory. Moreover, the database sequences are sorted by their sequence length to minimize length differences between neighboring threads. Finally, sequence items (characters) are replaced with numeric indexes to make lookup in substitution matrix easier [11].

Memory coalescing

DOPA[11] stated "Memory bandwidth represented a serious bottleneck while developing the GPU implementation". DOPA greatly utilized memory bandwidth by coalescing memory accesses. Furthermore, the trace back step of Smith Waterman algorithm is skipped on the GPU and is processed on CPU to save GPU memory utilization. In a naive GPU implementation each thread uses a 32 byte memory access with 28 bytes of bandwidth is wasted. With coalescing memory accesses the required data is stored (aligned) in neighboring addresses and a single 64 byte load instruction is used to load a data.

Efficient work load distribution The sorted sequences are grouped into sequence sets. These sets are consisting of 16 sequences and half-warp of threads working on them. Sorted sequences by length have almost equalized workload for threads inside each set. To equalized work load for a half-warp working on a sequence set is achieved by concatenated the sequences in the sequence sets with leftover sequences and form sequence groups. The length of each sequence group within a set is roughly equal to the length of the longest sequence in that set. Two types terminator are inserted. Sequence terminators are inserted between the concatenated sequences to inform GPU kernel to start new alignment. While, sequence group terminators are inserted at the end of each group to indicate the end of a group of concatenated sequences and therefore, the execution of the half-warp threads will be terminated.

Substitution matrix access

Substitution matrix BLOSUM62 is implemented in DOPA. Substitution matrix is accessed frequently and randomly, every time two sequence items are aligned, hence its access is significant to the performance of DOPA . Therefore, a query profile matrix is

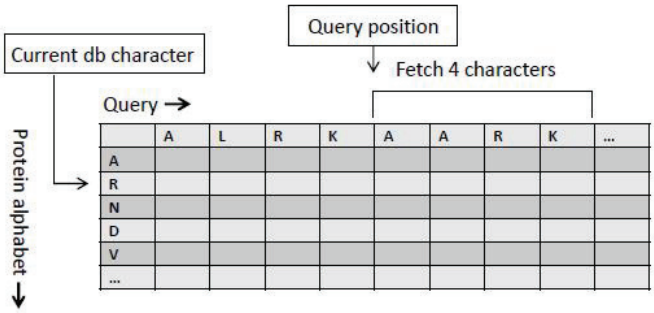

Figure 4.8: Query profile based on [11]

implemented in DOPA and it is generated once for every query sequence. Figure 4.8 depicts the query profile where the query sequence is used along the top row of the matrix instead of the protein alphabet. The access for specific database items in a query profile now become more deterministic and query sequence lookup is avoided.

4.4 Summary

This chapter described the sequential implementation of the Smith Waterman algorithm. In addition, it highlighted how the Smith Waterman algorithm can be parallelized and it presented our SSE2 and SSearch implementation and our GPU and DOPA implementation. Next chapter focuses on DOPA performance on different GPU cards NVIDIA Tesla C2075, GTX 275 and GT640 and describes the performance of all discussed implementations and shows the variation in the performance and how other factors like power and price contribute to the design decision.

Bibliography

[1] http://fasta.bioch.virginia.edu

[2] Qianghua Zhu, Fei Xia, and Guoqing Jin. Accelerating the Smith-Waterman Algorithm for Bio-sequence Matching on GPU Electronic Engineering College, Naval University of Engineering, Wuhan, P. R. China. 2012

[3] Vipin Chaudhary, Feng Liu, Vijay Matta, Laurence T. Yang. Parallel Implementations of Local Sequence Alignment: Hardware and Software.

[4] Wozniak, A.: Using video-oriented instructions to speed up sequence comparison. Computer Applications in the Biosciences 13(2), 145-150 (1997)

[5] Rognes, T., Seeberg, E.: Six-fold speed-up of Smith Waterman sequence database searches using parallel processing on common microprocessors.2000

[6] Arpith Jacob, Marcin Paprzycki, Maria Ganzha and Sugata Sanyal. Applying SIMD Approach to Whole Genome Comparison on Commodity Hardware.2008.

[7] Thesis: M. Kentie. Biological Sequence Alignment Using Graphics Processing Units.2010

[8] Michael Farrar. Striped Smith-Waterman speeds database searches six times over other SIMD implementations.2006.

[9] Rognes T, Seeberg E: Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. Bioinformatics 2000.

[10] Phil Green. SWAT Program http://www.phrap.org/phredphrap/swat.html

[11] Laiq Hasan, Marijn Kentie and Zaid Al-Ars. DOPA: GPU-based protein alignment using database and memory access optimizations.2011

[12] Ligowski, L. Rudnicki, W. An efficient implementation of Smith Waterman algorithm on GPU using CUDA, for massively parallel scanning of sequence databases. 2009.

[13] Amora Amir, Kim Wai Tang, Dongni Fan. Report Open Assignment. Advanced Multicore Systems (ET4381) course.

Performance analysis and benchmarking

5

Throughout this thesis different implementations of one Big Data application; Smith Waterman algorithm; on different GPU platforms for several problem cases have been experimented. This chapter focuses on analyses and evaluates the experimental results of SSE2, GPU and DOPA implementations. A detailed analysis is presented involving measuring the execution time for Smith Waterman algorithm on different platforms, determine memory transfer bottlenecks on the GPU. Section 5.1 shows the experimental setups and performance of our sequential, SSE, GPU and DOPA implementation. Furthermore Section 5.2 presents and evaluates DOPA in term of performance capability and relative costs and power on three different graphics cards: NVIDIA Geforce GT640, GeForce GTX 275 and Tesla C2075 GPU.

5.1 Experimental setup

In this section we explain the computer configuration that we have used during our benchmarking for SSE, GPU and DOPA implementations.

System
OpenSUSE 11.3 Linux with a 2 cores processor Intel Core 2 Duo CPU with speed of 2GHz and total memory 3.9 GB. NVIDIA Geforce GT640 and Tesla C2075 graphics cards.

Query sequences
During the benchmark we test protein sequences. The protein query sequences are used in our test are the same as the other researcher have used in their benchmarking. We tested the same benchmarking with SSearch [7] to be able to compare and also make sure of the validity of the results. These query sequences are selected from UniProt/Swiss-Prot database. The query sequences are P02232, P05013, P14942, P07327, P01008, P03435, P27895, P07756, P04775, P19096, P28167, P0C6B8, P20930 and Q9UKN1. The length of these queries varies from 144 to 5478 items.

Database
For convenient benchmarking we have created a synthetic test database selected from UniProt/Swiss-Prot database with a total length of 13432526 items (23299 sequences). We used the synthetic test database we used it during testing our implementations while we used Swiss-Prot release October 2010 and release July 2013 for DOPA test experiments.

DOPA program settings
Substitution matrix is BLOSUM62. Gap penalty: -10 and gap extend penalty: -2. NVIDIA's Compute Visual Profiler profiling application. DOPA source code is derived from [1].

Table 5.1: Performance of our C implementation.

Query Sequences	Length	Execution Time (sec.)	Performance (GCUPS)	Our Score	FASTA Score
P02232	144	14.2	0.136217165	719	719
P05013	189	18.62	0.136345189	977	977
P14942	222	22.02	0.135423287	1135	1135
P07327	375	17.37	0.135517817	1957	1957
P01008	464	46.17	0.134994413	2392	2392
P03435	567	55.91	0.136223256	3048	3048
P27895	1000	99.18	0.135435834	5064	5064
P07756	1500	148.36	0.135810117	7713	7713
P04775	2005	198.23	0.135863465	10379	10379
P19096	2504	247.5	0.135899172	12924	12924
P28167	3005	297.24	0.135798481	15646	15646
P0C6B8	3564	352.68	0.135742097	19896	19896
P20930	4061	402.03	0.135685118	21359	21359
Q9UKN1	5478	541.34	0.13592821	27936	27936

Measuring method
The execution time of the application was timed using the C clock() instruction. The execution time considered for the comparisons was the time spent on running the kernel program thus loading the database sequences and copy to GPU are excluded from the measured time. This way indicates the time required by the GPU compared to the time required by the CPU to run the same part of Smith Waterman algorithm.

5.2 Sequential vs. parallel implementations

Table 5.1 shows the performance of our sequential implementation. The performance is in giga cell updates per second (GCUPS). GCUPS is the total number of Smith-Waterman score matrix cells that are calculated per second. The formula below is used to calculate the performance.

GCUPS = query length * total database length /execution time/10^9

To validate the results of the maximum scores, we have tested the same benchmarking (same query sequences and database) on SSearch/FASTA. Our naive Smith Waterman implementation running on the multi-core CPU platform is achieved 0.13 GCUPS.

Table 5.1 presents the performance in term of GCUPS and the execution time of the sequential implementation. The shortest sequence in our benchmarking is the sequence P02232 which consists of 144 items and it needs 14.02 seconds to be calculated while the largest sequence Q9UKN1 with length of 5478 needs 541.34 seconds. The execution time of the sequential implementation takes about 1 second for every 10 items. The sequential implementation is very slow and therefore is hardly used especially with larger databases like UniProt/Swiss-Prot. Our implementation using SSE2 technology and tests were carried out on Intel Core 2 Duo CPU with speed of 2 GHz and total memory 3.9 GB running on linux openSUSE 11.3. The implementation code is written in C and compiled using the gcc compiler. In the tests, the local alignment score between two protein sequences was calculated without reconstructing the alignment. The BlOSUM62 substitution matrix was used. We used a value of +7 for a gap open and

+1 for gap extension penalties. The performance results of our SSE2 implementation

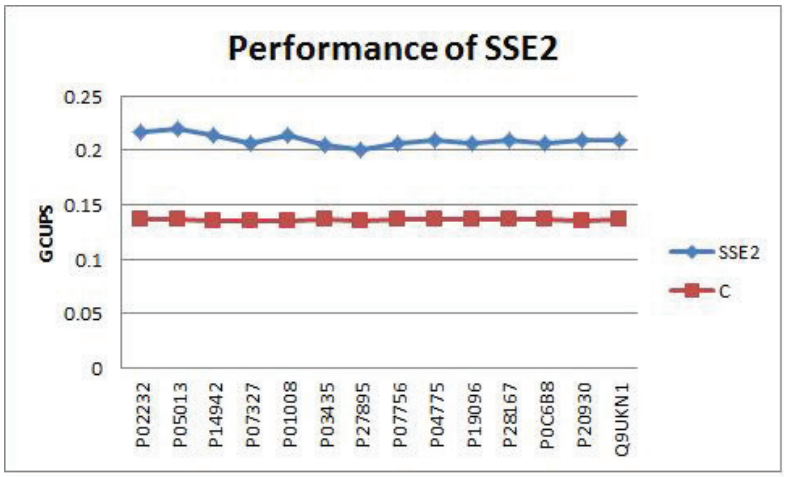

Figure 5.1: SSE2 performance

compared with the sequential implementation are shown in Figure 5.1. We can see that the performance has improved and reached 0.22 GCUPS. However our implementation is only straightforward there is no optimization considered in our design just exploited the benefit of the instruction set architecture of SSE2. The implementation recorded 1.6x speedup over the sequential implementation. The reason behind this improvement is the 8 processing elements performing 8 multiple alignments simultaneously.

The performance of the SSearch implementation is taken from [8]. Tests were done on a 2.0 GHz Xeon Core 2 Duo processor with 2 GB of RAM running Windows XP. The number of CPU cores had no effect on the execution time because the program was run as a single threaded application. The database sequence was Swiss-Prot contains 75,841,138 amino acids (208,000 sequences) and 11 query sequences with length ranging from 143 to 567 amino acids. While we used in our benchmarking synthetic test database and the total length is 13432526 amino acids (23299 sequences) and we used 14 query sequences with length ranging from 144 to 5478. SSearch implementations tested the program using different scoring matrices which are BLOSUM62 and BLOSUM50 and different gap penalties. The values of scoring matrix and gap penalties affect the performance of SSearch implementations; this is due to calculation of lazy F. The higher the value of gap open and gap extension penalty the less iterations are required to be calculated. Three tests were performed. First test used scoring matrix BLOSUM62 gap open 10 and gap extension is 1 and second test was used BLOSUM50 scoring matrix and gap open 10 and gap extension is 1 while the last test used the same 11 query sequences with the BLOSUM50 and BLOSUM62 scoring matrices, but now four different gap open and gap extension values 10-1, 10-2, 14-2 and 40-2 were used respectively. For the first test SSearch implementation completed the search in 113 seconds with an average of 2.5 GCUPS and a peak of around 3 GCUPS. In the second test was affected by the higher values of H and therefore more time was needed to calculate F. This requires 159 seconds to complete the search with an averaging of 1.8 GCUPS and a peak of 2.25 GCUPS.

Table 5.2: Properties of three different graphics cards

Device Properties	Geforce GTX 275	Geforce GT640	Tesla C2075
CUDA Capability	1.3	3	2
CUDA Cores	240	384	448
MEMORY Size	896MB	2GB	6GB
Memory Bandwidth	127GB/s	28GB/s	148GB/s
Memory Frequency	1.13 GHz	891 MHz	1.6 GHz
GPU Frequency	633MHz	928 MHz	1147 MHz

Finally the last test was used to test the efficiency of the inner loop when using the large gap opening of 40 and gap extension of 2. Mainly most of the time required to calculate F was skipped and SSearch implementation took 90 seconds to complete the search this is as 20% to 40% improvement over the gap opening of 10 and gap extension of 1.

Table 5.2 depicts device properties of the three GPU cards GT640, Tesla C2075 and GTX275. Our GPU straightforward implementation reported the performance of 0.47 GCUPS on GT640 and on Tesla C2075 achieved 0.80 GCUPS. Because we do not have the graphics card GTX 275 in our lab we extract the required data from [3] and the straightforward Smith Waterman implementation on GTX 275 achieved 0.54 GCUPS.

Table 5.3: Sequential vs. straightforward GPU implementation

Sequential performance on Intel Core 2 Duo	Performance on GTX275	Performance on GT640	Performance on Tesla C2075
0.13 (GCUPS)	0.54 (GCUPS)	0.47 (GCUPS)	0.80 (GCUPS)
speedup	4.2x	3.6x	6.1x

Basically, the implementation utilized inter-task parallelism without balancing the workload. Because each thread will do the computation of one query sequence and database sequence pair, the calculation time depends heavily on the length of both query sequence and database sequence. This explains why when we use longer query sequences for test, the time is increases. And also, we bind the database sequences on texture memory. So when the size of the database is very large, we cannot execute our program with exceeding the size of texture memory. Compared to our sequential implementation, the performance has been improved achieving 0.80 GCUPS on the Tesla C2075 (see Table 5.3).

5.3 Performance analysis of DOPA on different GPUs

DOPA [3] implementation was optimized and ported specifically for NVIDIA GTX275, which resulted in a huge performance gain which is 21.4 GCUPS comparing to its counterpart implementation that running sequentially on CPU. It could massively utilize the

Table 5.4: An overview of the performance of DOPA on different platforms, where number of blocks is 4 multiples of the number of multiprocessors

Platforms	Num. blocks (threads)	Throughput GCUPS	Performance gain vs. sequential (0.13 GCUPS)
GTX275	120(64)	21.4	164.6x
GT640	8(960)	6.8	52.3x
Tesla C2075	56(137)	21.9	168.5x

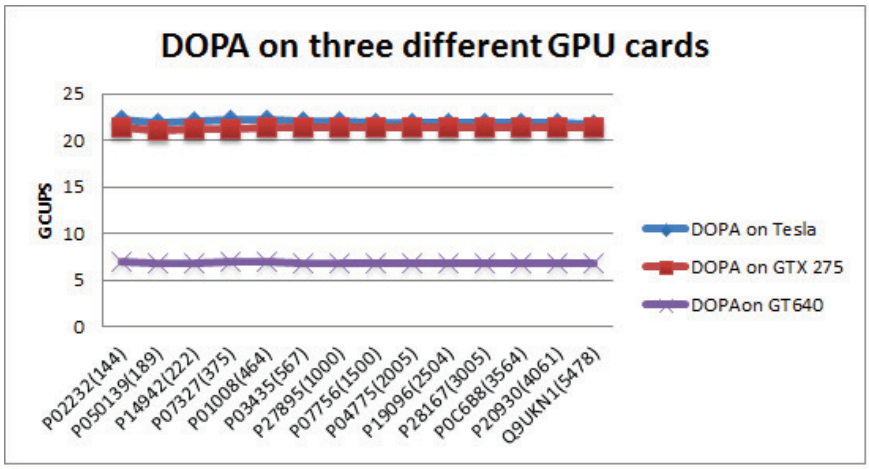

Figure 5.2: Performance of DOPA on different platforms

GPU computing power and benefit from its underlying features. So, this section studies the behavior of this implementation on different graphics cards that have higher power of computations. Initial expectations were to capture a better performance when the number of the processing units (cores) increases.

DOPA is implemented to launch the kernel with 7680 threads in total, and recommended to distribute those threads in (4 multiples of the number of multiprocessors) blocks. Thereby, it used 120 Blocks (4 x 30 multiprocessors), with 64 threads per each. Table 5.4 shows an overview of the performance with these settings.

From Teble 5.4 we can see (Figure 5.2) the performance of DOPA fluctuates and according to this, two observations emerge:

1. the performance of DOPA on Tesla C2075 GPU is almost the same as on GTX 275.

2. the performance of DOPA is dramatically decreased on NVIDIA Geforce GT640.

Table 5.4 shows that the performance on Tesla C2075 is almost the same as on GTX275 and both are about 3x better than on GT640 although the number of cores on

the Tesla C2075 is about 2x more than GTX275 and 1.5x more than GT640. That shows that the computing capability of the hardware does not guarantee higher performance when the code is optimized for a specific platform. That is due to the differences in the hardware architecture (See Table 5.5), the memory bandwidth as well as the workload distribution and number of the active threads that share the same set of resources, i.e. registers and shared memory. Thus, to achieve a better performance, many other aspects have to be taken into account like knowing the underlying features of a certain platform to tune the CUDA configuration accordingly to increase the number of active threads that can concurrently occupy the available cores.

Table 5.5: An overview of the architectures of the experimenting platforms

Platform	Num. Multiprocessors	Num. cores per multiprocessor	Total num. cores
GTX275	30	8	240
GT640	2	192	384
Tesla C2075	14	32	448

Generally, for an optimal GPU utilization, the number of blocks is recommended to be a multiple of number of multiprocessors and number of threads per block is recommended to be a multiple of number of warp size. However, in Table 5.4, the workload distribution was totally different from one platform to another. For instance, on GTX275 the workload was chosen to be 120 blocks, 64 threads per each, or in other words 240 warps in total, 2 warps per each block. Whereas on GT640, it was chosen to be 8 blocks, 960 threads per each. And on Tesla C2075, it was chosen to be 56 blocks, with around 137 per each. During the benchmarking we saw that DOPA required 63 registers and when we do the theoretical calculation we determine the number of registers to be 63 and according to these settings theoretically, this workload is expected to be distributed as in Table 5.6.

Table 5.6: Theoretical calculations on how the scheduler could distribute the workload over the multiprocessors (considering the configurations in Table 5.4)

Platform	Num. MP	Num. active blocks/MP	Num. active warps/MP	Total active blocks on GPU	Num. active threads/MP	Occup.
GTX275	30	4	8	120	256	25%
GT640	2	1	30	2	960	47%
Tesla C2075	14	3	15	42	480	31%

This explains why the performance on Tesla C2075 was slightly better than GTX275 and considerably better than on GT640. Although GT640 has 192 cores per multiprocessor, and sustains about 30 active warps (as we can see in Table 5.6), however, all these threads share the same resources (shared memory and registers). In addition, it has only 2 multiprocessors that run in parallel. It is also possible to explain from Table 5.6 why the final performance of Tesla C2075 was not much higher than GTX275. Tesla C2075 has 42 active blocks compared to 120 active blocks on GTX275, nevertheless the

number of active warps is higher on Tesla C2075 and run on 32 cores compared to 8 cores however they share the same resources. Moreover, the number of threads per block is not a multiple of warp size on Tesla C2075. That means benefiting from the additional number of cores in a multiprocessor can still be constrained by the available resources that can be shared between the active warps. For efficient utilization it is better to keep the cores and/or multiprocessors busy as much as possible, to avoid the time wasted for idle cores.

Therefore, we tried to launch the kernel with tuned configurations. The second experiment has fixed number of blocks to 120 and 64 threads and set the number of registers to 63. Table 5.7 shows the performance with the tuned configurations.

Table 5.7: An overview of the performance on different platforms, where number of blocks is 120 with 64 threads per each

Platform	Throughput GCUPS
GTX275	21.4
GT640	10.20
Tesla C2075	25.03

Table 5.8 demonstrates how the workload was distributed on the GPUs with this configuration. It shows that the number of active blocks that resides on the GPU was increased and since the number of threads per block was a multiple of warp size that improved the occupancy of the cores per multiprocessor. Thus, these results can still be acceptable if we consider the other important factors like for instance memory bandwidth and resources shared per multiprocessor.

Table 5.8: Theoretical calculations on how the distributor could distribute the workload over the multiprocessors (considering the configurations in Table 5.7)

Platform	Num. MP	Num. active blocks/MP	Num. active warps/MP	Total active blocks on GPU	Num. active threads/MP	Occup.
GTX275	30	4	8	120	256	25%
GT640	2	16	32	32	1024	50%
Tesla C2075	14	8	16	112	512	33%

Another important performance factor is memory bandwidth. DOPA [2] stated "Memory bandwidth represented a serious bottleneck while developing the GPU implementation". The maximum theoretical memory bandwidth for GTX 275 is 127 GB/s. And during the benchmarking, [2] stated that with the test database about 50 GB/s of bandwidth was used in practice. It can be concluded that memory accesses are not a limiting factor for the GTX 275 platform. The maximum theoretical memory bandwidth for Tesla C2075 is 144 GB/s and during the benchmarking we found about 50 GB/s of bandwidth was used in practice. Again the memory accesses are not a limiting factor on Tesla. While the maximum theoretical memory bandwidth for GT640 is only 28 GB/s

and 50 GB/s of bandwidth was required in practice therefore here on GT640 the memory access is really a limiting factor on this platform.

Practically the kernel requires around 63 registers and as the author of DOPA mentioned that most of optimization that had been done increased the register pressure, because too many query and database symbols need to be stored in registers to avoid spilling these data in the slower memory. So there is a trade-off for DOPA implementation, increasing the performance depends on increasing number of registers which results in decreasing the occupancy, while improving the cores occupancy to improve the performance requires to decrease the number of registers.

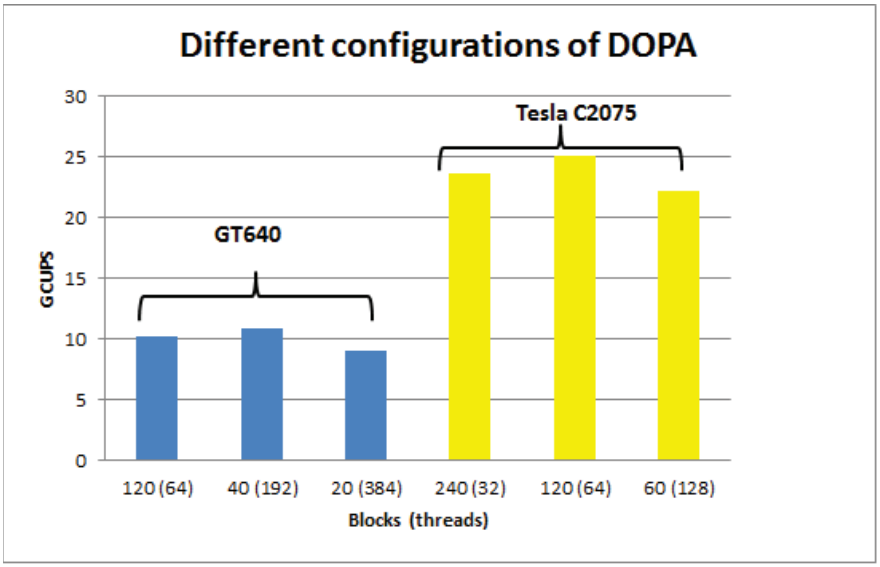

Figure 5.3: Performance of DOPA with different configurations for both Tesla C2075 and GT640

We also experimented with another set of threads configurations on those platforms (Figure 5.3) to find out the most optimum configuration, where the number of blocks is a multiple of the number of multiprocessors and keeping the number of threads per block a multiple of warp size. See Table 5.9 and 5.10.

Table 5.9: Performance on Tesla C2075 with different configurations

Num. blocks(thre.)	Throughput GCUPS
240(32)	23.7
120(64)	25.03
60(128)	22.15

From Table 5.9 and 5.10, we can see that the performance noticeably changes when the configuration is tuned based on the used architecture as this improves the threads distribution and maximizes the GPU utilization. As a conclusion, these experiments show that this application is portable and scalable. Finally, we experimented with the

Table 5.10: Performance on GT640 with different configurations

Num. blocks(thre.)	Throughput GCUPS
120(64)	10.20
40(192)	10.87
20(384)	9.03
10(768)	9.06

optimized implementation on these different platforms with the same configuration mentioned above but for different problem sizes, these experiments show that this application is scalable.

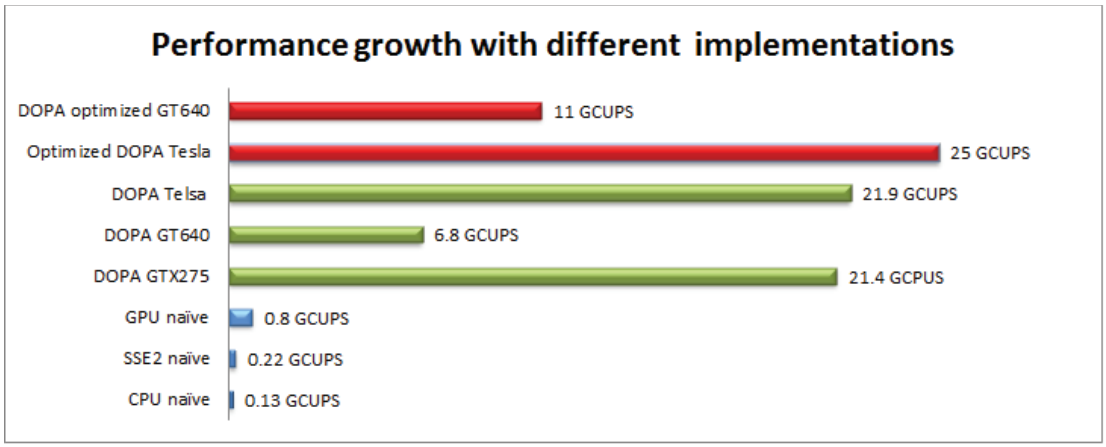

Figure 5.4: Performance growth of Smith Waterman algorithm

Figure 5.4 depicts the performance of Smith Waterman implementations on several platforms. Our naive Smith Waterman version running on CPU platform only achieved 0.13 GCUPS. The performance is improved gradually by using different optimization strategies, such strategies are general compiler optimization O3 (option O3 is enabling optimization to the speed of the code and it is recommended for code with loops and perform much calculations or processing large data sets) and Streaming SIMD Extension (SSE2), the performance reported 0.22 GCUPS. Our GPU straightforward implementation achieved 0.80 GCUPS on Tesla C2075 which is 6.1x speedup over the sequential implementation. While DOPA implementation achieved 21.9 GCUPS on Tesla C2075 which is 168.5x speedup over the naive CPU based implementation. The performance of our optimized DOPA is increased by nearly 14% to 61% reaching 25 and 11 GCUPS on Tesla C2075 and GT640, respectively. Comparing our optimized DOPA running on Tesla C2075 to the naive sequential Smith Waterman CPU based implementation reported 192x speedup.

Table 5.11: Properties of three different graphics cards

Metrics	Geforce GTX 275	Geforce GT640	Tesla C2075
Performance	21.4 GCUPS	10.87 GCUPS	25.03 GCUPS
Watt	219	65	215
Price(euro)	250	100	2000

5.4 Performance versus cost, power and flexibility

In this thesis we discussed two platforms, the CPU and the GPU, and their potential to accelerate Smith Waterman algorithm. For DOPA running on fast PC with one of three different NVIDIA GPUs cards (Geforce GT640 and Tesla C2075 and GTX 275) different performance numbers have been reported but this is not only the competitive metric in practice. Other metrics like cost, power consumption and flexibility play an important role to purchase the product.

We can see from Table 5.8 that the hardware cost in terms of purchase price and watt for these three cards varies considerably. DOPA can deliver different amount of GCUPS per card. Different efficiency scenarios can be established depending on the bioinformatics domain requirement. Note these cards should be plugged on a motherboard of a CPU thus the price of a system (a CPU) must be taken in account along with a GPU card price.

However we focus in our analysis on the price of the graphics cards and excluding the CPU price. The information in Table 5.11 is derived from [4, 5, 6]. The prices are checked and compared with many online computer web-shops.

DOPA produces performance in term of GCUPS on Tesla C2075 which is 17% higher than on GTX 275 and 230% higher than on GT640 but on the other hand the cost of the Tesla C2075 in terms of watt and euro is very high. Choosing now for GTX 275 with 17% less performance and tolerable payment in terms of euro than on Tesla but GTX 275 is consuming much watt than on Tesla. So choosing between these two cards is really a trade-off. It depends on the requirement of the consumer. In the research field for example of bioinformatics where the budget is low, the price will be an important issue. GT640 is a candidate for consumers that want to pay less in terms of watt and euro but the performance gain in term of GCUPS is far less than the other cards.

Plugging three GT640 cards on a commodity motherboard will cost around 300 euro (see Table 5.11). With this amount of money we still can get performance as much as on Tesla C2075 but the cost would be cheaper than on Tesla. However, the problem that should be considered is the flexibility of debugging GPU code which is not as common as debugging CPU code. For this, we probably need a skilled engineer to manage optimizing the workload of the kernel among these three cards to get the expected desired performance and to hide the interconnect latency. However, one card of GT640 consumes at maximum 65 watt and with the aforementioned scenario three cards then consume at maximum 195 watt which is still less than the power of Tesla C2075 and GTX275.

Bibliography

[1] http://kentie.net/article/thesis/index.htm

[2] Laiq Hasan, Marijn Kentie and Zaid Al-Ars. DOPA: GPU-based protein alignment using database and memory access optimizations.2011

[3] Thesis: M. Kentie. Biological Sequence Alignment Using Graphics Processing Units.2010

[4] Tweakers.net Pricewatch, August 2013, http://tweakers.net/pricewatch

[5] Geforce GT640, August 2013, http://www.nvidia.com

[6] Tesla C2075, August 2013, http://www.nvidia.com

[7] http://fasta.bioch.virginia.edu

6 Summary and future work

To achieve considerable performance for Big Data applications for different platforms, two important factors have to be taken into account: increase the parallelism in the software and increase the utilization on the hardware side. This thesis showed an example of implementing a Big Data bioinformatics applications on different GPU platforms. By increasing the parallelism of these applications a huge performance gain has been achieved.

In this thesis, we load the same set of instructions with the same data structure and same CUDA configurations on different GPU architectures.

The results were not as expected due to an inefficient distribution of the workloads. Whereas, we improved the cores occupancy, the performance was improved by about 14% to 61%. With improving the cores occupancy on used GPU cards, we achieved 25 GCUPS performance on Tesla C2075 and 11 GCUPS on GT640 compared to a straightforward port on these cards achieving 21.9 and 6.8 GCUPS, respectively. Further, the optimized CPU based Smith Waterman implementation has significantly efficient performance by re-ordering the sequence of instructions and removing data dependency.

In practice, the higher performance of the Smith Waterman algorithm in bioinformatics is not only the demand. Other metrics are also important. The price of the hardwares and the power consumption are important issues. So the requirements of the applications determine the decision making for which card can be selected.

6.1 Thesis contribution

Here we present briefly what we have achieved in this thesis.

1. Studying the the state of the art of Big Data and specifying its challenges and determine its potential in several domains.

2. Analyzing the sequential implementation of Smith Waterman from scratch.

3. Analyzing the accelerated Smith Waterman using SSE2.

4. Analyzing the accelerated Smith Waterman using GPU.

5. Ported DOPA to be run on Linux platform, Tesla C2075 and GT640 GPU graphics cards.

6. Determining and analyzing the decrease of the performance of DOPA on GT640 GPU, and analyzing the reasons the performance on Tesla C2075 is the same as on GTX 275.

7. Improving the cores occupancy configuration for DOPA on two GPU cards and achieving higher performance.

8. Including several metrics during the analysis like power, price and flexibility.

6.2 Future work

Due to time and scope restrictions of this thesis, improving the occupancy was not the main focus. So, for future work it is recommended to increase the parallelism of DOPA if possible by finding a way to minimize the register pressure and as the Genebank database is increasing in size the performance will increase for free by increasing the number of blocks on CUDA configuration when tuning the code for the dedicated architectures. Memory access in the GPU platform significantly affect the performance of Smith Waterman algorithm. NVIDIA announced new technology (Volta 3D stacked memory) around 2015-2016 and Maxwell virtual memory will see the light around 2014. These promise to improve the performance of Big Data applications as well as reduction in term of watt.

Our SSE2 implementation has improved the performance over the sequential implementation using SIMD instructions. Efforts should be investigated to exploit the benefit of new technologies such as SSE3, SSE4 or even by using Advanced Vector Extensions (AVX) which is an advanced version of SSE.