Knowledge Discovery from Databases: Task 1

André Pereira
Bernardo Portela

University of Minho, Gualtar, Braga 4710-070, Portugal,
{AMPereira90,BLFPortela}@gmail.com

Contents

1 Introduction **2**

1.1 Motivation, Goals, and Scientific Contribution . . . 4

1.1.1 The Top Quark and Higgs Boson Decay . . . 5

1.1.2 Goals and Scientific Contribution . . . 7

2 State of The Art **8**

2.1 Hardware . . . 8

2.1.1 Homogeneous Systems . . . 8

2.1.2 Heterogeneous Systems . . . 9

2.2 Software . . . 14

2.2.1 Shared Memory Environments . . . 14

2.2.2 Distributed Memory Environments . . . 15

2.2.3 Particle Physics Frameworks . . . 15

3 New lipmini **16**

4 Thesis Planning **17**

1 Introduction

Today's computing platforms are becoming increasingly complex with multiple interconnected computing nodes, each with multiple multicore CPU chips, and sometimes coupled with hardware accelerators. While the application performance is an important issue to tackle, the efficient usage of the resources of these systems is a crucial subject that needs to be addressed. Guaranteeing that the available computational resources are being fully used by an application may require deep knowledge of the underlying architecture details of both CPUs and hardware accelerators and extensive tuning of each individual application. It is important to understand the resources on a CPU, such as the computing units organisation, benefits and limitations of using multiple cores, and cache memory hierarchy, to avoid underusing the full computational potential of the device. The architecture design of many-core hardware accelerators have significant differences from device to device with no standard yet defined, unlike current CPUs. The programmer must know the architectural details of each hardware accelerator to produce efficient code. Moreover, the communications among the CPU and hardware accelerators must be managed by the programmer and may significantly affect the efficiency of an application.

From the hardware point of view, efficiency has a different meaning: engineers consider it to be the ratio between power usage and computational throughput. This is a subject of extensive research known as "Green Computing", where the goal is to reduce power consumption of the hardware with little impact to its performance. This is not only important for mobile computing but also to reduce the cost of maintaining huge computing clusters and data centres.

Computing clusters are the most popular High Performance Computing (HPC) platform and are constituted of many computing nodes, with possibly different characteristics, interconnected by specialised communication channels in a distributed memory environment. The computing nodes may be homogeneous or heterogeneous platforms, where the first has only one or more CPUs in a shared memory environment, and the latter has hardware accelerators connected by a PCI-Express interface to the CPUs, in a distributed memory environment. This means that the data is always visible to the CPUs, but must be explicitly transferred to the accelerator devices. The management of the data may affect the performance and efficiency of an application. Code parallelism is a must to take advantage of the multiple cores in both the CPUs and the hardware accelerators, with the programming model differing from shared to distributed memory environments. Data races, resource contention and, in heterogeneous systems, explicit memory transfers are complex issues that the programmer must tackle. Also, each accelerator manufacturer uses their own frameworks and compilers for programming their devices. With the current computational systems rapidly changing, scientists restrain from investing on academic formation in computer science, opting for self-learning these complex principles, and often avoid developing code for hardware accelerators. These factors reinforced the collaboration of multidisciplinary teams of scientists from various fields with computer scientists to develop high performing, efficient and robust applications.

The European Organization for Nuclear Research [1] (CERN, acronym for *Conseil Européen pour la Recherche Nucléaire*) is a consortium of 21 European countries, and more than 30 "observer" countries, with the purpose of operating the largest particle physics laboratory in the world. Founded in 1954, CERN is located in the border between France and Switzerland, and employs thousands of scientists and engineers representing 608 universities and research groups of 113 different nationalities.

CERN research focus on the basic constituents of matter to understand the fundamental structure of the universe, which started by studying the atomic nucleus but quickly progressed into high energy physics (HEP), namely on the interactions between particles. The instrumentation used in nuclear and particle physics research is essentially formed by particle accelerators and detectors, alongside with the facilities necessary for delivering the protons to the accelerators. The Large Hadron Collider (LHC) particle accelerator (later presented) speeds up groups of particles close to the speed of light, in opposite directions, inducing a controlled collision at the detectors core (the collision of two particles is referred as an "event"). The detectors record various characteristics of the resultant particles of each collision, such as energy and momentum, which originate from complex decay processes of the original particles. The purpose of these experiments is to test models and predictions in High Energy Physics (HEP), such as the Standard Model, by confirming or discovering new particles and interactions.

CERN started with a small low energy particle accelerator, the Proton Synchrotron [2] inaugurated in 1959, but soon its equipment was iteratively upgraded and expanded. The current facilities are constituted by the older accelerators (some already decommissioned) and particle detectors, as well as the newer Large Hadron Collider (LHC) [3] high energy particle accelerator, located 100 meter underground and with a 27 km circumference length. There are currently seven experiments running on the LHC: CMS [4], ATLAS [5], LHCb [6], MoEDAL [7], TOTEM [8], LHC-forward [9] and ALICE [10]. Each of these experiments have their own detector on the LHC and conduct HEP experiments, using distinct technologies and research approaches. One of the most relevant researches being conducted at CERN is the validation of the Standard Model and discovery of the Higgs boson theory. The ATLAS experiment, a key project at CERN, aims to study the properties of the recently discovered Higgs boson [11], the search for new particles predicted by models of physics beyond the Standard Model like Susy, searches for new heavy gauge bosons and precision measurements where the top quark is of utmost importance. During the next year the LHC will be upgraded to increase its luminosity, e.g., the amount of energy of the accelerated particle beams.

Approximately 600 millions of collisions occur every second at the LHC. Particles produced in head-on proton collisions interact with the detectors of the ATLAS experiment, generating massive amounts of raw data as electric signals. It is estimated that all the detectors combined produce 25 petabytes of data per year [12, 13]. CERN does not have the financial resources to afford the computational power necessary to process all the data, which motivated the creation of the Worldwide LHC Computing Grid [14], a distributed computing infrastructure that uses the resources of scientific community for data processing. The grid is organized in a hierarchy divided in 4 tiers. Each tier is made by one or more computing centres and has a set of specific tasks and services to perform, such as store, filter, refine and analyse all the data gathered at the LHC.

Deixo a especificação dos tiers de computação do CERN ou faço apenas uma breve síntese?

The Tier-0 is the data centre located at CERN. It provides 20% of the total grid computing capacity, and its goal is to store and reconstruct the raw data gathered at the detectors in the LHC, converting it into meaningful information, usable by the remaining tiers. The data is received on a format designed for this reconstruction, with information about the event, detector and software diagnostics. The output of the reconstruction has two formats, the Event Summary Data (ESD) and the Analysis Object Data (AOD), each with different purposes, containing information of the reconstructed objects and calibration parameters, which can be used for early analysis. This tier distributes the raw data and the reconstructed output by the 11 Tier-1 computational centres, spread among the different member countries of CERN.

Tier-1 computational centres are responsible for storing a portion of the raw and reconstructed data and provide support to the grid. In this tier, the reconstructed data suffers more reprocessing, refining and filtering the relevant information and reducing the size of the data, now in Derived Physics Data (DPD) format, then transferred to the Tier-2 computational centres. The size of the data for an event is reduced from 3 MB (raw) to 10 kB (DPD). This tier also stores the output of the simulations performed at Tier-2. The Tier-0 centre is connected to the 11 Tier-1 centres by high bandwidth optical fiber links, which form the LHC Optical Private Network.

There are roughly 140 Tier-2 computational centres spread around the world. Their main purpose is to perform both Monte-Carlo simulations and a portion of the events reconstructions, with the data received from the Tier-1 centres. The Tier-3 centres range from university clusters to small personal computers, and they perform most of the events reconstruction and final data analysis. In the CERN terminology, an analysis is the denomination of an application which is designed to process a given amount of data in order to extract relevant physics information about events that may support a specific HEP theory.

These factors enforce the need to process more data, more accurately, in less time, which often leads to investments on larger computing clusters to improve the quality of the research results. However, most scientific code was not designed and/or developed for an efficient use of the available computational resources. If these applications were adequately designed (or tuned), the event analysis throughput could be massively increased. An efficient parallel application can significantly improve its performance at a much lower cost [15].

The Laboratório de Instrumentação e Física Experimental de Partículas (LIP) [16] is a portuguese scientific and technical association for research on experimental high energy physics and associated instrumentation. LIP has a strong collaboration with CERN as it was the first scientific organization from Portugal that joined CERN, in 1986. It has laboratories in Lisbon, Coimbra and Minho and 170 people employed. LIP researchers have produced several applications for testing at ATLAS several HEP theoretical models that use Tier-3 computational resources for data analysis. Most of the analysis applications use in-house developed skeleton libraries, such as the LipCbrAnalysis and LipMiniAnalysis.

1.1 Motivation, Goals, and Scientific Contribution

With an increase in particle collisions and data being produced by the detectors at the LHC, research groups will need a larger budget to acquire and maintain the required computational resources to keep up with the analysis. Adding to this increase, research groups working on the same experiment enforce positive competition to find and publish relevant results. The amount and quality of event processing has a direct impact on the research, meaning that groups with more efficient computational resources become ahead of the competition.

Better physics are not only obtained by increasing the amount of events analysed; it is important to take into account the quality of each event analysis. Due to several intrinsic ATLAS experimental effects like energy and transverse momentum resolutions, the measured kinematic properties of particles produced in a collision, may be shifted within a range of $\pm 1\%$, implying an intrinsic uncertainty which is propagate through the event analysis. It is possible to improve the reconstruction quality by varying the values measured by the detector within the said range, but with a significant impact to the analysis execution time, requiring a trade-off between the event processing throughput and their reconstruction quality.

To aid the development of these data analysis applications, scientists at LIP created a skeleton library named LipCbrAnalysis. It contains a set of physics utilities, such as specific classes and functions, and removes the need to code the input file reading, memory allocation of each event data, and output creation. With this, the programmer only needs to code the specific bits of the analysis filtering and reconstruction of events. An iteration of this skeleton was developed, named LipMiniAnalysis, with the purpose of reading a new structure of the input data files, and stripping the former skeleton of outdated features.

An efficiency study and optimisation of one of LIP production data analysis, used also as case study for some preliminary studies of this PhD pre-thesis work, was presented in [15, 17]. It tackled the computational inefficiencies on both homogeneous and heterogeneous platforms, and identified several limitations to performance scalability, specially when using hardware accelerators. The data analysis case study and the limitations identified with the LipMiniAnalysis skeleton are presented in subsection 1.1.1.

Dealing with scientific code is no trivial task due to the code structure and organization. Several studies [**SC:Nature**, **SC:Develop**, **SC:SC11**] identified the causes that lead scientists to produce poor code:

- Most scientists are self-taught programmers with no computer science background;
- Scientists disregard software engineering principals to produce long lasting, extendible, and efficient code;
- Scientists often iterate through the same application, producing legacy code (some applications currently in production are iterated on for the last 20 years), and not documenting it so that it can be used by others;
- Scientists usually are not aware of profiling and debugging tools, as well as parallelisation paradigms;
- Scientists do not understand the architectural details of computing systems, reducing the portability of the code they produce.

To improve the quality of the scientific code, scientists agree that it is important to create an interface between their field and computer science by having multidisciplinary teams. However, computer scientists often lack the scientific knowledge required to be acknowledge as an integral part of these teams.Also, scientists are often sceptical to let others restructure, and even develop from scratch, legacy code that they are using for years.

1.1.1 The Top Quark and Higgs Boson Decay

At the LHC, two proton beams are accelerated close to the speed of light in opposite directions, set to collide inside a specific particle detector. This head-on collision triggers a chain reaction of decaying particles, and most of the final particles interact with the detector, allowing to record relevant data. One of the searches being conducted at the ATLAS Experiment relates to the study of the top quark and Higgs boson couplings. Figure 1.1 represents the final state topology of the associated production of two top quarks and one Higgs boson (that decays to two b-quarks), labelled from now on as $t\bar{t}H$ production. Figure 1.2 provides a schematic representation of the system to highlight certain features, such as the bottom quarks being jets of particles, and the leptons (both l^+ and l^-) being a muon and electron in the t and $\bar{t}$ decays, respectively.

Neutrinos do not interact with the detector, so their characteristics are not recorded. Since the top quark reconstruction requires the neutrinos, their characteristics are analytically determined with the known information of the system, through a kinematical reconstruction. However, the $t\bar{t}$ system may not have a possible reconstruction: the reconstruction has an intrinsic uncertainty associated which determines its accuracy.

The amount of jets from bottom quarks and leptons present in the events may vary according to the decay channel of the W bosons produced in the top quark decays. As shown in figure 1.2, four jets and two leptons are required to be present in the events. Two of the jets, together with two leptons are required to reconstruct the $t\bar{t}$ system, and the remaining two jets are used for the Higgs boson reconstruction. For the kinematical reconstruction, every possible combination of jets and leptons must be evaluated and only the most accurate reconstruction of each event is considered. In a first step, the $t\bar{t}$ system

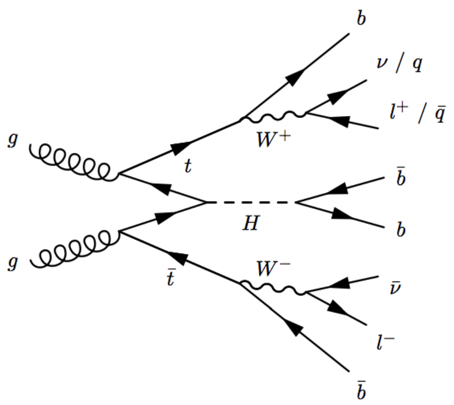

Figure 1.1: Feynman diagram of the $t\bar{t}$ and Higgs boson production.

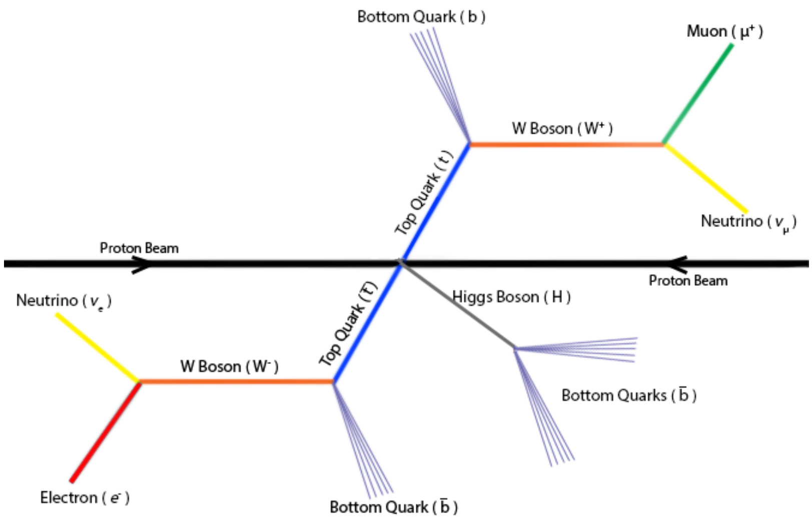

Figure 1.2: Schematic representation of the $t\bar{t}$ system and Higgs boson decay.

reconstruction is tried. If it has a possible solution, the Higgs boson is reconstructed from the jets of the two remaining bottom quarks. The Higgs reconstruction does not use the jets which were associated to the best $t\bar{t}$ system reconstruction. The overall quality of the event processing depends on the quality of both reconstructions.

For the global event reconstruction, several solutions can be tested if we assume that the ATLAS detector has an experimental energy-momentum resolution of $\pm 1\%$, by varying these quantities within their uncertainty. This uncertainty is propagated into the $t\bar{t}$ system and Higgs reconstructions, affecting their accuracy. To improve the quality of the reconstructions several random variations are applied to the measured values, within a maximum range of $|1\%|$ next to the measured values. The quality of the reconstructions and the application execution time is directly proportional to the amount of variations performed per combination. The goal is to do as many variations as possible within a reasonable time frame.

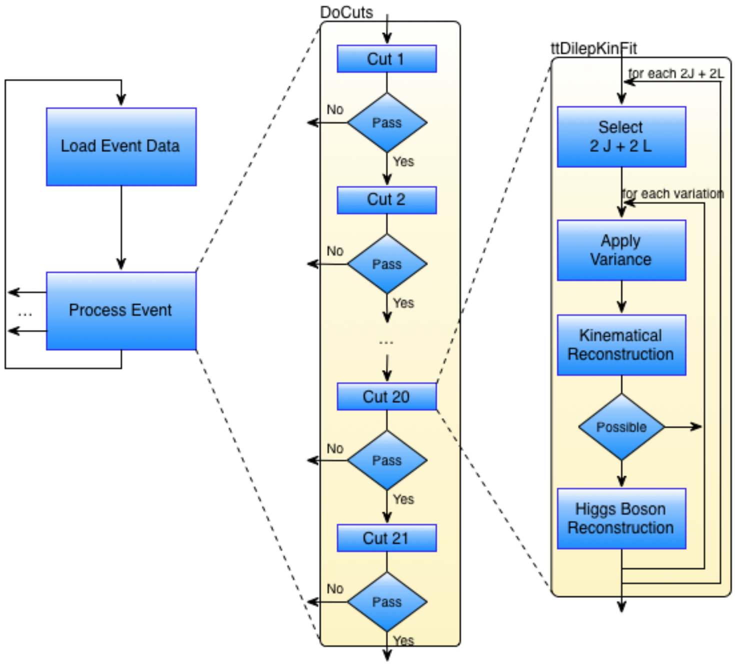

Figure 1.3: Schematic representation for the `ttH_dilep` application flow.

To reconstruct the $t\bar{t}H$ system a data analysis application was developed, the `ttH_dilep`. The application flow is presented in figure 1.3. Each event data on an input file is individually loaded into a single global state, shared between the data analysis code and the LipMiniAnalysis, and it is overwritten every time a new event is loaded. The event is then submitted to a series of cuts, which filters events that are not suited for reconstruction. When an event reaches the cut 20, the $t\bar{t}$ system and Higgs boson are reconstructed in the function `ttDilepKinFit`, which is expected to be the most computing demanding. If the $t\bar{t}$ system reconstruction fails, the current combination is discarded and the next is processed. If an event has a possible reconstruction it passes the final cut and its final information is stored.

1.1.2 Goals and Scientific Contribution

frameworks criadas

legacy code

2 State of The Art

2.1 Hardware

Most scientific research groups have access to computing clusters. These highly parallel systems usually are constituted by racks of computing nodes interconnect by a specialised network, but each with an individual instance of the operating system. The cluster has a distributed memory configuration, where the data must be explicitly transferred among nodes. The nodes may have different characteristics and configurations as long as they use the same interface to communicate with the others.

Cluster often nodes dedicated to centralise the data storage, with an abstraction layer to the user. However, when running an application, the user file system is mounted on the nodes that will perform the computation, copying all the data needed to avoid unnecessary communication. These computing nodes may be homogeneous or heterogeneous systems.

2.1.1 Homogeneous Systems

Homogeneous systems are the most common computing platforms, constituted by one or more CPU devices with their own memory bank (RAM memory) and interconnected by a specific interface. Although these systems use a shared memory model, where all the data is shared among CPUs, when considering a multiple CPU system, each has its own memory bank, which causes the system to have a Non Unified Memory Access (NUMA) pattern, as presented in figure 2.1. This means that the access time of a CPU to a piece of memory in its memory bank will be faster than accesses to the other CPU bank. The threads of an application must have the data that they will use on the memory bank of the CPU device that they are running to avoid the increased costs of NUMA.

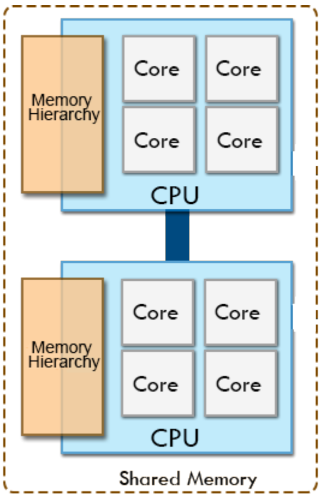

Figure 2.1: Schematic representation of a homogeneous system.

CPU devices

Gordon Moore predicted, in 1965, that for the following ten years the number of transistors on the CPU chips would double every 1.5 years [18]. This was later known as the Moore's Law and it is expected to remain valid at least up to 2015. Initially, this allowed the increase in CPU chips clock frequency by the same factor as the transistors. Software developers did not spend much effort optimising their applications and only relied on the hardware improvements to make them faster.

Due to thermal dissipation issues, the clock frequencies of CPU chips started to stall in 2005. Manufacturers shifted from making CPUs faster to increasing their throughput by adding more cores to a single chip, reducing their energy consumption and operating temperature. This marked the beginning of the multicore and parallel computing era, where every new generation of CPUs get wider, while their clock frequencies remain steady.

CPU devices are designed as general purpose computing units, and may contain multiple cores, each based on a simple structure of small processing units with a very fast hierarchical memory attached (cache, whose purpose is to hide the high latency access to global memory), and all the necessary data load/store and control units. They are capable of delivering a good performance in a wide range of operations, from executing simple integer arithmetic to complex branching and SIMD (single instruction multiple data, later explained) instructions. A single CPU core implements various mechanisms for improving the performance of applications, at the hardware level, with the most important explained next:

ILP instruction level parallelism (ILP) is the overlapping of instructions, performed at both the hardware and software level, which otherwise would run sequentially. At the software level it is denominated as static parallelism, where compilers try to identify which instructions are independent, meaning that the outcome of one does not affect the execution of the other, and schedules them to be executed at the same time, if the hardware has resources to do so. At the hardware level, ILP can be referred as dynamic parallelism as the hardware dynamically identifies which instructions execution can be overlapped while the application is running.

Vector instructions are a special set of instructions based on the SIMD model, where a single instruction is simultaneously applied to a set of data. CPU instruction sets offer special registers and instructions that allow to execute a operation on a chunk of data in a special arithmetic unit. One of the most common examples is addition of two vectors. The hardware is capable of adding a given number of elements of the vectors . This optimization is often done at compile time.

Multithreading is the hardware support for the execution of multiple threads in a CPU core. This is possible by replicating part of the CPU resources, such as registers, and can lead to a more efficient utilisation of the CPU core hardware. If one thread is waiting for data, other thread can resume execution while the former is stalled. It also allows a better usage of resources that would otherwise be idle during the execution of a single thread. If multiple threads are working on the same data, multithreading can reduce the synchronisation between them and lead to a better cache usage.

2.1.2 Heterogeneous Systems

With the emerging use of hardware specifically designed for some computing domains, denominated hardware accelerators, whose purpose is to efficiently solve a given problem, a new type of computing platform is becoming increasingly popular. This marked the beginning of heterogeneous systems, where one or more CPU devices, operating in a shared memory environment similar to homogeneous systems presented in subsection 2.1.1, are coupled with one or more hardware accelerators. In current heterogeneous systems, CPUs and accelerators operate in a distributed memory model, meaning that data must be explicitly passed from the CPU to the accelerator, and vice-versa.

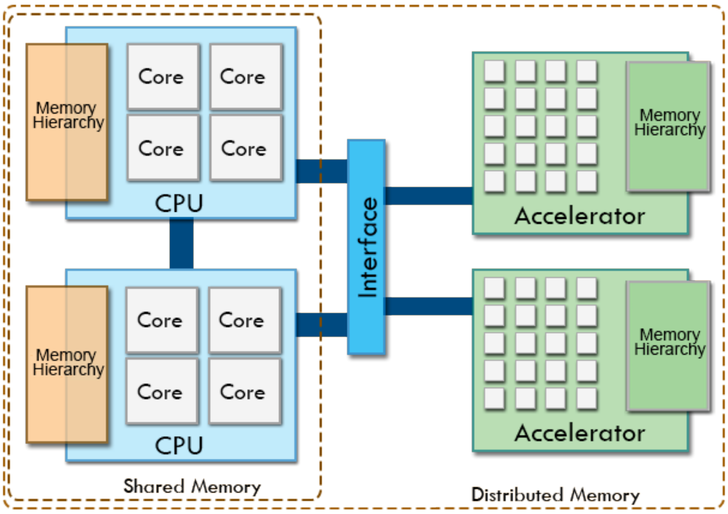

Figure 2.2: Schematic representation of a heterogeneous system.

Figure 2.2 presents a schematic representation of a heterogeneous system. Note that both CPUs must use the same interface to communicate with the hardware accelerators, which may cause contention. This high latency interface, PCI-Express being the most common, usually is a potential bottleneck for applications that use accelerators.

Computing accelerators are usually built with a large number of small and simple processing units, aimed to achieve the most performance possible on specific problem domains, as opposed to general purpose CPUs. They are oriented for massive data parallelism processing (SIMD architectures), where a single operation is performed on huge quantities of independent data, with the purpose of offloading the CPU from such data intensive operations. Several manycore accelerator devices are available, with me most popular being the general purpose GPUs to the Intel Many Integrated Core line, currently known as Intel Xeon Phi [19]. An heterogeneous platform may have one or more accelerator devices of the same or different architectures.

As of November 2013, over 50 of the TOP500's list [20] are computing systems with hardware accelerators, which indicates an exponential growth in usage compared to previous years. The Intel Xeon Phi is becoming increasingly popular, being the accelerator device of choice in 13 clusters of the TOP500, with one of them being the Tianhe-2 which is the faster system on the list. NVidia GPUs remain as the most used accelerator, with the AMD steadily losing their share.

Graphics Processing Unit

The Graphics Processing Units (GPU) were one of the first hardware accelerators on the market. Their purpose is to accelerate computer graphics applications, which started of as simple pixel drawing and evolved to support complex 3D scene rendering, such as transforms, lighting, rasterisation, texturing, depth testing, and display. Due to the industry demand for customisable shaders, this hardware later allowed some flexibility for the programmers to modify the image synthesising process. This also allowed using this GPUs as a hardware accelerator for wider purposes beyond computer graphics, such as scientific computing, as some researchers saw the potential to use these devices to boost the performance of numerical computation.

The GPU architecture is based on the SIMD model. Its original purpose was to process and synthesise images, which are, from the computation point of view, a large set of numbers representing pixels. The processing of each pixel usually does not depend on the processing of its neighbours, or any other pixel on the image, so the computation has no data dependencies in the best case scenario. This allows to process all pixels simultaneously. The massive data parallelism is the most important characteristic that was considered when designing the GPU architecture.

As the GPU manufacturers allowed more flexibility to program their devices, the High Performance Computing (HPC) community started to use them to solve specific massively data parallel problems, such as numerical computation problems. However, the highly specialised architecture of GPUs affects the performance of many other problem domains. Due to the increased demand for these devices by the HPC community, manufacturers began to generalise more of the GPUs features, such as adding support for double precision floating point arithmetic, and later began producing accelerators specifically oriented for scientific computing. NVidia is the number one GPU manufacturer for scientific computing GPUs, with a wide range of available hardware known as Tesla. These devices characteristics differ from the general purpose GPUs, as they have more GDDR RAM, a different structural design to fit in cluster nodes, and different cooling options. The chip itself is different, offering more processing units and larger caches. The latest architecture released by NVidia is named Kepler [21], and its relevant architecture details are explained next.

Figure 2.3: Schematic representation of the NVidia Kepler architecture.

As figure 2.3 shows, the Kepler architecture is organised into two main components: the Streaming Multiprocessor (SMX) and the memory module. The focus of this architecture was not only the performance but the energy efficiency, offering up to to 3x more performance per watt than Fermi (the previous architecture). In addition, the Kepler has implemented several features to improve the usage of resources:

Dynamic Parallelism: a kernel (algorithm coded to run on the GPU) on the GPU is capable of being called recursively, which allows to dynamically generate new load to process, without the CPU interfering. This allows for less regular algorithms to run on the GPU and reduces the communication between CPU and GPU as it is capable of managing the workload.

Hyper-Q: this system increases the amount of work queues to 32 simultaneously hardware managed connections. With this, multiple CPU cores can launch new different kernels on the CPU at the same time, increasing the resource usage. Now, multiple threads of the same application are not required to have exclusive usage of the GPU, reducing the amount of synchronisations.

Grid Management Unit: to allow for dynamic parallelism a new grid (a collection of threads of a kernel, later explained in more detail) management system is required. The new system also allows to schedule multiple grids, which allows for different kernels, from possibly different threads, to run simultaneously (Hyper-Q).

NVidia GPUDirect: this feature allows GPUs in a single system, or in a interconnected network, to share data without the interference of the CPU and system memory, creating a direct connection to Solid State Drives and other similar devices, and reducing the communication latency.

The SMX are the units responsible for performing all computations on the GPU, and a chip may have up to 15. Each SMX has 192 single precision and 64 double precision CUDA cores, small processing units capable of performing basic arithmetic, 32 special function units, to perform complex computations such as trigonometric operations, and 32 load and store units. These computing units operate at the GPU main clock rate. The SMX features 4 warp schedulers (warps are explained next) and 8 instruction dispatchers.

Memory wise, each SMX has 65536 32-bit registers, with a maximum of 255 registers per CUDA thread, a 64 KByte very fast memory for L1 cache and shared memory, and a similar fast 48 KByte memory cache for read-only data. Finally, the Kepler architecture provides 1536 KB of L2 cache shared among all SMX units. The high end available Tesla K40 has a memory bandwidth of 280 GB/s to its main memory. Since the GPU is connected by PCI-Express interface, the bandwidth for communications between CPU and GPU is restricted to only 12 GB/s (6 GB/s in each direction of the channel). Memory transfers between the CPU and GPU must be minimal as it greatly restricts the performance.

A kernel is executed by a given amount of parallel workers named CUDA threads. They are grouped into blocks, to be scheduled among SMX and the threads inside a block can only run in a given SMX, and these are grouped into a grid, which contains all CUDA threads (up to $2^{31} - 1$) for a given kernel. The CUDA threads are grouped in batches of 32, called warps, to be dispatched by a warp scheduler. The scheduler has a scoreboard with up to 48 entries to manage which warps are stalled waiting for resources or data and which are ready to be executed.

Intel Many Integrated Core architecture

The Intel Many Integrated Core (MIC) architecture, with the current production device known as Intel Xeon Phi, is an emerging technology becoming adopted by various clusters in the TOP500 list. It has a design different from the NVidia GPUs presented previously, opting to have fewer computing units but capable of performing more complex operations. Figure 2.4 presents a schematic representation of the architecture. The current high end model, the Intel Xeon Phi 7120p, has 61 cores and 16 GB GDDR5 RAM. The device has three operating modes:

Native: the device acts as an independent system itself, with one core reserved for the operating system execution. The application and all libraries must be compiled specifically to run on the device, and later copied to the its memory along with the necessary input data, prior to its execution. No further interaction with the CPU is required until the application has executed.

Offload: the device acts an accelerator, such as a GPU. Only part of the application is set to run on the Xeon Phi, and data must be explicitly passed between CPU and device as required by the code that it will compute. All library functions called inside the device must be specifically compiled for it. Note that it is not possible to have an entire library compiled for the Xeon Phi and CPU simultaneously.

Message passing: the device acts as an individual computing system in the network. Memory transfers are explicitly and the device can be programmed using the Message Passing Interface (MPI) [22]. The restrictions mentioned in the previous point are also applicable.

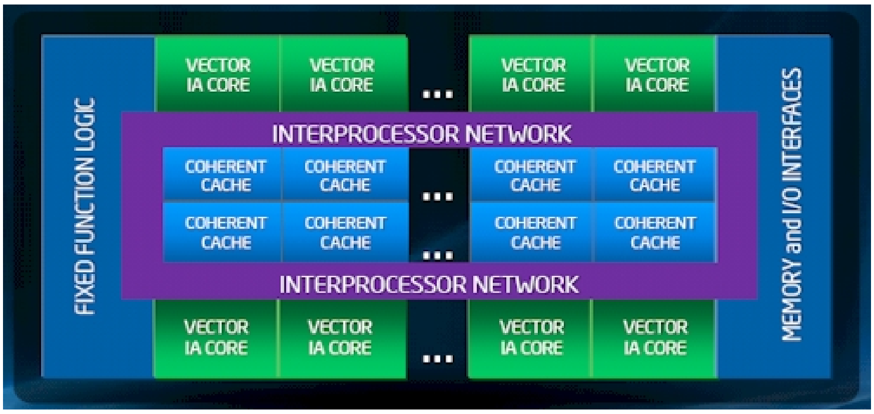

Figure 2.4: Schematic representation of the Intel Many Integrated Core architecture.

Each core is able to run 4 threads simultaneously, and most of the massive parallelism is obtained by he 32 512 bit wide vector registers available. The core as 64 KB for data and 64 KB for instruction L1 cache, and 512 KB L2 cache. There is no shared cache among the 61 cores of the chip, and no cache consistency and coherence is automatically guaranteed among them. The cores are interconnected by a bidirectional ring network. MIC does not support out of order execution, which greatly compromises the use of ILP. Also, the clock frequency is limited to 1.33 GHz, which is less than half of the modern CPUs.

Since it uses the same instruction set as conventional x86 CPUs, Intel claims that current applications can be easily ported to run on the device. This may be true for common matrix arithmetic and similar applications, efficient ports of complex applications that require the use of many external libraries is very difficult, or even infeasible [15].

Other hardware accelerators

More hardware accelerators are coming to the market due to the increasingly popularity of GPUs and Intel MIC among the HPC community. Texas Instruments developed their new line of Digital Signal Processors, best suited for general purpose computing while very power efficient. Their capable of delivering 500 GFlop/s (giga floating point operations per second), consuming only 50 Watts [23].

ARM processors are now leading the mobile industry and, alongside the new NVidia Tegra processors [24] that are steadily increasing the market share, are likely to be adopted by the HPC community[1] due

[1] e.g. the ARM based Montblanc project will replace the MareNostrum in the Barcelona Supercomputing Center (BSC)

to their low power consumption while delivering a significant performance [25]. Due to the increased complexity of mobile applications, the shift from 32 bit to 64 bit mobile processors has already happened, which will greatly benefit computing clusters using this type of hardware.

2.2 Software

Most programmers (both computer scientists and self-taught programmers) are only used to code and design sequential applications, showing a lack of know-how to develop algorithms for parallel environments. This lack of expertise is even greater when programming for heterogeneous systems, where programming paradigms shift when considering different hardware accelerators. The mainstream industry is still adopting the use of multicore architectures with the purpose of increasing their processing power, reflecting in a lack in the academic training of computer scientists on code optimisation and parallel programming. Self taught programmers have an increased obstacle due to the lack of theoretical basis when using these new parallel programming paradigms.

Programming for multicore environments requires some knowledge of the underlying architectural concepts of CPU devices and how they are interconnected. Shared memory, cache coherence and consistency and data races are architecture-specific aspects that the programmer does not face in sequential execution environments. However, these concepts are fundamental not only to ensure efficient use of the computational resources, but most importantly the correctness of the application.

Heterogeneous systems combine the flexibility of multicore CPUs with the specific capabilities of manycore accelerator devices. However, most computational algorithms and applications are designed with the specific characteristics of CPUs in mind. Even multithreaded applications usually cannot be easily ported to these devices expecting high performance. To optimise the code for these devices it is necessary a deep understanding of the architectural principles behind their design.

The workload balance between the cores of a single CPU chip is an important aspect to extract performance and get the most efficient usage of the available resources. A inadequate workload distribution may cause some cores of the CPU to be starved, unnecessarily increasing the application execution time. A good load balancing strategy ensures that all the cores are used as most as possible. Considering a multi-CPU system, it is important to manage the data in such a way that it is available in the memory bank of the CPU that will need it to avoid the NUMA latency. The same concepts apply when balancing the load between CPU and hardware accelerators, with the increased complexity of the distributed memory environment and high latency data transfers.

Some computer science groups developed libraries that attempt to abstract the programmer from specific architectural and implementation details of these systems, providing an easy API as similar as possible to current sequential programming paradigms. The next subsections will present frameworks to aid the development of parallel applications for homogeneous and heterogeneous systems.

2.2.1 Shared Memory Environments

Homogeneous systems often operate in a shared memory environment. Using multiple CPU devices may cause the memory banks to be physically divided but hardware mechanisms, such as specialised CPU interconnections, allow for a common addressing space. Libraries and frameworks for parallelising for this environment are presented next.

2.2.2 Distributed Memory Environments

Heterogeneous systems use distributed memory address space for handling the data between CPU and accelerator devices. Even though the CPU devices work on a shared memory space, data must be explicitly passed to the accelerators. General purpose frameworks for parallelising on the devices and on the heterogeneous platforms as a whole are presented next.

2.2.3 Particle Physics Frameworks

3 New lipmini

frameworks general purpose nao agradam cientistas e problemas com elas

cientistas preferem frameworks com coisas deles e rotinas paralelizadas

especificaçao do que a nova framework tem de ter

o que ja foi feito até agora

4 Thesis Planning

References

[1] European Organization for Nuclear Research. *CERN European Organization for Nuclear Research*. Nov. 2012. URL: http://public.web.cern.ch/public/ (cit. on p. 2).

[2] European Organization for Nuclear Research. *The Proton Synchrotron*. July 2013. URL: http://home.web.cern.ch/about/accelerators/proton-synchrotron (cit. on p. 3).

[3] European Organization for Nuclear Research. *The Large Hadron Collider*. Nov. 2012. URL: http://public.web.cern.ch/public/en/lhc/lhc-en.html (cit. on p. 3).

[4] European Organization for Nuclear Research. *Compact Muon Solenoid experiment*. Nov. 2012. URL: http://cms.web.cern.ch/ (cit. on p. 3).

[5] European Organization for Nuclear Research. *ATLAS experiment*. Nov. 2012. URL: http://atlas.ch/ (cit. on p. 3).

[6] European Organization for Nuclear Research. *The Large Hadron Collider beauty experiment*. Nov. 2012. URL: http://lhcb-public.web.cern.ch/lhcb-public/ (cit. on p. 3).

[7] European Organization for Nuclear Research. *The Monopole and Exotics Detector at the LHC*. Nov. 2012. URL: http://moedal.web.cern.ch/ (cit. on p. 3).

[8] European Organization for Nuclear Research. *Total Cross Section, Elastic Scattering and Diffraction Dissociation at the LHC*. Nov. 2012. URL: http://totem.web.cern.ch/Totem/ (cit. on p. 3).

[9] European Organization for Nuclear Research. *The Large Hadron Collider forward experiment*. Nov. 2012. URL: http://home.web.cern.ch/about/experiments/lhcf (cit. on p. 3).

[10] European Organization for Nuclear Research. *A Large Ion Collider Experiment*. Nov. 2012. URL: http://aliceinfo.cern.ch/ (cit. on p. 3).

[11] Georges Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Phys.Lett.* B716 (2012), pp. 1–29 (cit. on p. 3).

[12] European Organization for Nuclear Research. *Computing*. July 2013. URL: http://home.web.cern.ch/about/computing (cit. on p. 3).

[13] European Organization for Nuclear Research. *Animation shows LHC data processing*. July 2013. URL: http://home.web.cern.ch/about/updates/2013/04/animation-shows-lhc-data-processing (cit. on p. 3).

[14] European Organization for Nuclear Research. *The Worldwide LHC Computing Grid*. July 2013. URL: http://wlcg.web.cern.ch/ (cit. on p. 3).

[15] André Pereira. "Efficient Processing of ATLAS Events Analysis in Homogeneous and Heterogeneous Platforms". MA thesis. University of Minho, Sept. 2013 (cit. on pp. 4, 13).

[16] Laboratório de Experimentação e Física Experimental de Partículas. *Laboratório de Experimentação e Física Experimental de Partículas*. Nov. 2012. URL: http://www.lip.pt/ (cit. on p. 4).

[17] A. Onofre A. Pereira and A. Proença. "Removing Inefficiencies from Scientific Code: the Study of the Higgs Boson Couplings to Top Quarks". In: *The International Conference on Computational Science and its Applications* (July 2014) (cit. on p. 4).

[18] Gordon E. Moore. "Cramming more components onto integrated circuits." In: *Electronics, 38(8)* (Apr. 1965) (cit. on p. 9).

[19] Intel. *The Intel® Xeon Phi Datasheet*. Tech. rep. Apr. 2014 (cit. on p. 10).

[20] TOP 500. *November 2013*. May 2014. URL: http://www.top500.org/lists/2013/11/ (cit. on p. 10).

[21] NVIDIA. *NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110*. Tech. rep. 2012 (cit. on p. 11).

[22] Edgar Gabriel et al. "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation". In: (Sept. 2004), pp. 97–104 (cit. on p. 13).

[23] Texas Instruments. *Digital Signal Processors*. May 2014. URL: http://www.ti.com/lsds/ti/dsp/overview.page (cit. on p. 13).

[24] NVIDIA Corporation. *Tegra*. May 2014. URL: http://www.nvidia.com/object/tegra.html (cit. on p. 13).

[25] Sixto Ortiz Jr. "Chipmakers ARM for Battle in Traditional Computing Market." In: *Computer, 44(4):14-17* (Apr. 2011) (cit. on p. 14).