International Journal of Advanced Computer Technology (IJACT)

ISSN:2319-7900

A COMPARATIVE OVERVIEW OF THE PARALLEL COMPUTATION MODELS AND ARCHITECTURE

Naiswita D. Parmar, Mohammed H. Bohara and Madhukar B. Potdar, PhD

Abstract

With ever increasing data sizes, the need for faster and efficient processing is acutely felt, especially so in case of processing of remotely sensed data. The computation involved can be SIMD or MIMD type. In case of SIMD, a repetitive computation is carried out on different data sets. To meet the requirement faster and efficient processing in SIMD environment, several computational model and technologies have been developed and they are subject of intensive research presently. This article, recapitulates various such technologies such as OpenMP and MPI using Intel Xeon Phi co-processors, OpenCl and CUDA programming using NVIDIA GPGPU processors, hybrid programming and compares them. The potentials of Intel Xeon Parallel Studio Xe for maximum code are optimization is discussed.

Introduction

The applications using satellite sensor/image applications are equipped for catching multi spectral, multi temporal and multi resolution high determination images. This high data requires time effective parallel data processing in online/offline environment. These days Multicore CPUs (co-processor accelerators) are accessible at low cost to speedup time intensive calculation. Image processing is strong case for parallel processing. To diminish the image processing time with parallel programming on multicore CPU is old methodology due to repetitive operations at local, regional and global levels. To accomplish full abilities of present day multicore frameworks a few parallel programming models and parallel augmentation to consecutive language are available.

There are two ways to parallelize an application. First, auto parallelization includes Instruction Level Parallelism (ILP) or use of parallel compiler. But the amount of parallelism is generally very low. Second, Parallel Programming in which application are developed to exploit parallelism. Parallel application development includes partitioning of the overall task into sub task and executes them on multiple processors in parallel. This helps to achieve better application performance. For this an application need to automatically scale with number of processor available. An effective parallelization programming is required. Thus, the obligation for delivering scalable parallelism falls on application developer.

The rest of this paper is organized as follows. Sections 2 and 3 define various computing models of parallel programming. Section 4 presents the latest Intel Xeon Phi processor, launched in comparison of NVidia GPU for deep learning kind of applications. It's followed by the different parallel computing techniques one can use with Intel Xeon Phi. And finally to keep track of all simultaneous work a toolbox is introduced in section 5.

Classification of Parallel Computing Model

A. OpenMP

OpenMP is directive based programming on shared memory multithreaded parallel extension to several programming language, (for example, C, FORTRAN or C++). It permits parallel execution of user-defined code regions. Most of time, OpenMP is used when compiler can't discover parallelism. It is appropriate for a processor with multicore architecture. OpenMP implements as a combination of an arrangement of compiler directives. Compiler directive is used to control the compilation of hardware description language, which used to model electronic systems. This dispenses with the prerequisite for isolated compiler support. In OpenMP, the utilization of thread is exceptionally structured as it was designed particularly for parallel applications [8]. It implements the fork-join model to present simultaneity. It is reasonable for the loop parallel programming structure design. In OpenMP, it is easy to relate the quantity of thread to the quantity of available processors and it is effectively scalable to an expansive number of processor.

In spite of the fact that it is straightforward and intense, an appropriate error handling mechanism is not available. Many profoundly parallel code blocks can't be effortlessly parallelized with OpenMP because of the possibility of dependence violation in critical code blocks. TLS (Thread Level Speculation) is the solution for this problem [4]. It allows the OpenMP programming model to be used even when dependence violation arises at runtime.

ISSN:2319-7900

B. MPI

MPI is another programming library for message passing between nothing shared processes i.e. time parallelizable code blocks. As OpenMP is only available for shared memory system, OpenMPI is the implementation of MPI specification for writing application does support for distributed memory system where in communication is achieved by message passing and not with shared variables. MPI is finest choice for data structures that are dynamic, unstructured (OpenMP 3.0 the dynamic data structures) and for the applications which require movability. Over last two decades, MPI has become the de-facto HPC (High Performance Computing) standard [8].

Though distributed memory computers are cheaper and cover more extensively the scope of solutions for problems than OpenMP, the MPI is difficult to troubleshoot and its execution is restricted by the communication network between the nodes.

C. Hybrid

The purpose of programming the hybrid model is to exploit the strength of both shared memory model and distributed memory model. The key concept is to utilize MPI crosswise over appropriated nodes and OpenMP (or any shared memory model) inside nodes. GPU additionally can be utilized as wellspring of processing force.

By consolidating the both one can get higher efficiency, memory saving and simplicity of programming from shared memory model and versatility preferred standpoint of distributed memory model.

Table 1. Compares the OpenMP and MPI programming models based on MC processors

	OpenMP	MPI
Memory Model	Shared memory model	Distributed memory model
Language Support	C,C++ and Fortran	C,C++, Fortran and provisional support for JAVA
Overhead	Thread can be created and joined for particular task depending on implementation	Transferring message from one process to another
Variable	Private and shared	Only private

Heterogeneous Parallel Computing Model

Distinctive sorts of co-processors are more reasonable for any particular task. For proficient usage of co-processor heterogeneous programming comes in picture. CPUs and GPUs both consolidated on single integrated circuit to increase better data exchange rate and low power utilization. The challenging task is to implement application operations and acquire the best performance. Furthermore, performance is varies widely from one co-processor to another.

A. CUDA Programming

NVidia's CUDA (Compute Unified Device Architecture) is an API for parallel computing. The CUDA model is designed to create applications scaling transparently with the expanding number of co-processor cores gave by the GPUs [3]. Host and device memories are considered as alternate entities in CUDA. The CUDA parallel computing platform gives a couple of basic C and C++ augmentations that empower communicating fine-grained and coarse-grained data and task parallelism. In fine grained multithreading switching between threads done on each instruction cycle, which possibly slows down execution of individual thread. On the other hand coarse grained multithreading switches between thread only done when current execution thread causes some long latency event. The programmer can express the parallelism in a high level language, for example, C, C++, FORTRAN or open source as OpenACC directives.

Thread (worker) administration in CUDA is done implicitly i.e. Developers are not in charge of thread creation and demolition. Workload partitioning and mapping is done explicitly.

B. OpenCL Programming

Open Computing Language (OpenCL) is a framework for parallel programming executed on heterogeneous platforms. The OpenCL distinguishes between the device and host and defines a multilevel memory model; consequently portability is the key component of OpenCL. It designed principally for GPUs. However, it can be used with other platforms, for example, multicore CPUs. That is the reason it underpins both data parallelism for GPUs and task parallelism for CPUs. The motivation behind this hybrid model is to exploit the qualities of both shared memory model and distributed memory model.

These days CUDA and OpenCL are the dominant GPGPU frameworks. CUDA is NVidia framework, therefore it does

not cover wide range of applications support like OpenCL, but where it is interspersed the top quality NVidia support ensures unparalleled performance.

Table 2. Comparison of heterogeneous computing models based on co-processors

	CUDA	OpenCL
Proprietor	NVidia	Open Source
Compilation	No runtime compilation	Runtime compilation
Reusability	No source code reuse across platforms	Source code reusable across platform
Hardware accessibility	It gives direct access to the hardware specific technologies	Programmer is not able to use hardware specific technologies
Range of applications	Limited to NVidia	Wider than CUDA
Performance	Superior to OpenCL	Preferable

Intel Xeon Phi Processor

Intel Xeon Phi is latest main cum co-processor designed, manufactured, marketed and sold by Intel [10]. It boosts parallel application performance and energy efficiency on HPC. Intel Xeon Phi processor, a foundational component of Intel Scalable System structure bring dramatic performance for some of today's most demanding application up to 1.2 tera flops for every coprocessor [10].

Intel Xeon Phi has 61 core SMP (Symmetric Multiprocessor) chip (out of these 60are available for computation). Every Phi contains 8 GB of RAM that gives all the memory and file system storage that each user process, the Linux OS, and auxillary daemon processes will need [10]. Intel Xeon Phi offers programmers scope to use their favored programming language C and C++ and parallelism model. However, the challenge lies in accomplishing best execution. For that, the designers need to use both parallelism and vector processing. With Intel Xeon Phi one can use both host and coprocessor to enable hybrid execution. Best performance is achieved when all available host and coprocessor are used [5].

Intel Xeon Phi is processor launched in competition of NVidia's GPU. A closest contender to Intel Xeon Phi in HPC market is NVidia Tesla and AMD file stream.

Both CUDA and Intel Xeon Phi coprocessors are intended to give high degrees of parallelism that can convey fabulous application performance. Yet, the challenge lies in accomplishing the most ideal performance with less programming efforts.

GPU requires rewriting of application kernel in the programming paradigm CUDA or OpenCL. The Intel Xeon Phi can achieve high performance and low power consumption without modify of application kernel. Compute kernel can be proficiently ported to Intel Xeon Phi with no or minor code changes. The exertion of porting logical applications to CUDA or OpenCL can be much higher contrasted with directive based programming model like OpenMP [7].

Table 3. Comparison of Intel Xeon Phi co-processor with NVidia co-processor

	Intel Xeon Phi	GPU
Parallelism	Task parallelism	Data parallelism
Tools	Intel native compiler	OpenCL
Superiority	For logical/arithmetic c	For FP calculation, as it is stack based
Libraries	Intel MKL libraries + others	CUDA libraries + others
Directives	OpenMP and PHI directive	OpenACC
Native programming model	Vector intrinsic	CUDA

A. Intel Xeon Phi with CUDA

CUDA developers can run their product with existing application code on Intel Xeon Phi processor. To run CUDA programs on Intel Xeon Phi processor, the CUDA code should be changed manually. While it is technically conceivable to run CUDA on phi coprocessor, products, for example, CUDA-86 don't as of now produce code for these devices [11]. OpenCL compiler can do this for Intel Xeon Phi processor. That infers CUDA developers can consider CUDA-to-OpenCL (CU2CL) source transfer to port their code.

B. Intel Xeon Phi with OpenMP

Designer of Intel Xeon Phi processors are pooling that there is sufficient vector as well as thread parallelism in application code. These will serves to profitably exploit the hardware and to draw out the superior performance. It doesn't make a difference in SIMD or MIMD computations because Intel Xeon Phi can run both SIMD and MIMD efficiently without putting any restrictions on how threads run or communicate. OpenMP is pragma based approach that streamlines the execution of the generic threads, despite the fact that the developer's need to guarantee that thread don't deadlock, enter race condition or limit scalability.

In Contrast with CUDA, the Intel Xeon Phi is composed as a coprocessor. On the other hand GPU only acts as accel-

ISSN:2319-7900

erator. Intel Xeon Phi coprocessor can be used as support processor just by aggregating existing application to run locally on it. While it is generally simple and straightforward to port CUDA application to Intel Xeon Phi coprocessor, surely this does not imply that interpreted CUDA code can accomplish the best possible performance. On the other hand with OpenMP, Intel TBB, the coprocessors appear like SMP on single chip [7]. So with OpenMP construct which use synchronization the overhead is lesser than on big SMP machine. It easier to port OpenMP on Intel Xeon Phi and it does not require laborious rewrite of kernel. Therefore compared to accelerators it reduces programming effort a lot [7].

Intel Parallel Studio XE

Intel Parallel Studio XE is first of its type vector programming designer toolbox for HPC and technical computing applications. It helps to enhance processing cycle through simultaneous operations on Intel Xeon Processors, Intel Xeon Phi coprocessor and other compatible processors. The fundamental objective of the Parallel Studio XE is to boost application performance by exploiting the increasing processor count and vector register width accessible in most recent processor.

First, the increase in cores leads to increase in the number of available threads. And it is not handy to keep track of all the things that may happen parallel. Furthermore, it becomes more difficult proportionally when the core count is increased. Second, vector register width helps vector operations to build. This in turn helps developers to design, build, verify and tune code in FORTRAN, C++, and C. Intel Parallel Studio XE supports OpenMP 4.0 [12].

Summary

The need for faster and efficient processing is acutely felt with ever increasing data sizes, especially in case of processing of remotely sensed data. The computation involved can be SIMD or MIMD type. In case of SIMD, a repetitive computation is carried out on different data sets. To meet the requirement faster and efficient processing in SIMD environment, several computational model and technologies have been developed and they are subject of intensive research presently. This article, recapitulated various such technologies such as OpenMP and MPI, using Intel Xeon Phi co-processors, OpenCl and CUDA programming using NVIDIA GPGPU processors, hybrid programming and compares them. The potentials of Intel Xeon Parallel Studio Xe for maximum code are optimization is discussed.

Craving for more and more compute power, especially for SIMD computations, leads to increasing in core counts in modern CPUs to speed up an application; such as Intel Xeon Phi coprocessor and accelerators such as GPU. First, GPU is more capable for atomic operations on irregular data. Second, Intel Xeon Phi coprocessor in light of many integrated Intel core design makes utilization of both host and coprocessor. As GPU can improve the computational power up to a great extent, it has quite complicated implementation and restricted utilization. On the other hand, the Intel Xeon Phi is beneficial for large, regular dataset and best performance is achieved when hybrid (of OpenMP and MPI) execution is used. But it is very challenging task to get full benefit of all available coprocessor available in system.

Intel Xeon Phi co-processor can be programmed with available programming language such as CUDA and OpenMp. CUDA has preferable execution times, but the intricacy to construct code is bigger than OpenACC and OpenMP. On the other hand it is quite simple and straightforward to port any OpenMP application to Intel Xeon Phi. OpenMP code delivers the best performance when running natively on Intel Xeon Phi (Intel.com). To build an application that has multithreading, vectors which take an advantage of all available resources in modern CPU is a complex and error prone task. But an application using OpenMP or CUDA that fulfills the entire above requirement achieves best performance. Though there are many advantages, the memory limit will probably be the fundamental constraint for the present era of Intel Xeon Phi coprocessor.

Acknowledgments

We are thankful to Director, BISAG for providing infrastructure and encouragements. Special thanks Mr. Amit Chauhan, Parul University, for initial guidance.

References

[1] Jaroš, Milan, et al. "Implementation of K-means segmentation algorithm on Intel Xeon Phi and GPU: Application in medical imaging." Advances in Engineering Software (2016).

[2] Dokulil, Jiri, et al. "High-level Support for Hybrid Parallel Execution of C++ Applications Targeting Intel® Xeon Phi™ Coprocessors." Procedia Computer Science 18 (2013): 2508-2511.

[3] Kamalakannan, Anandhanarayanan, and Govindaraj Rajamanickam. "High Performance Color Image Processing in Multicore CPU using MFC Multithreading." International Journal of Advanced Computer Science and Applications 4.12 (2013).

[4] Aldea, Sergio, et al. "An OpenMP extension that supports thread-level speculation." IEEE Transactions on Parallel and Distributed Systems 27.1 (2016): 78-91.
[5] Teodoro, George, et al. "Comparative performance analysis of Intel Xeon Phi, GPU, and CPU." arXiv preprint arXiv:1311.0378 (2013).
[6] Patel, Sumit, Bhadreshsinh Gohil, and M. B. Potdar. "Optimization of Support Vector Machine on Multi-core processor with OpenMP.".
[7] Cramer, Tim, et al. "Openmp programming on intel xeon phi tm coprocessors: An early performance comparison." Proceedings of the Many-core Applications Research Community (MARC) Symp. at RWTH Aachen University. RWTH Achen University, 2012.
[8] Diaz, Javier, Camelia Munoz-Caro, and Alfonso Nino. "A survey of parallel programming models and tools in the multi and many-core era." IEEE Transactions on parallel and distributed systems 23.8 (2012): 1369-1386.
[9] Ledur, Cleverson Lopes, Carlos MD Zeve, and Julio CS dos Anjos. "Comparative analysis of OpenACC, OpenMP and CUDA using sequential and parallel algorithms." 11th Workshop on parallel and distributed processing (WSPPD). 2013.
[10] Intel Xeon Phi Product Family, http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html
[11] CUDA vs. Phi: Phi Programming for CUDA Developers, http://www.drdobbs.com/parallel/cuda-vs-phi-phi-programming-for-cuda-dev/240144545
[12] OpenMP Specification, http://www.openmp.org/

Biographies

NAISWITA D. PARMAR received the diploma degree in Information Technology from M.S. University, Vadodara, Gujarat in 2010, the B.Tech. degree in Information Technology from DDU, Nadiad, Gujarat in 2014.currently, she is pursuing her master's degree from Parul University, Vadodara, Gujarat and doing her internship for final year dissertation at BISAG, Gandhinagar, Gujarat.

MOHAMMED H. BOHARA received the B.Tech. degree in Computer Science & Engineering from R.G.P.V., Bhopal, Madhya Pradesh in 2011, the M.Tech(ICT) degree in Computer Network from the DAIICT, Gandhinagar, Gujarat. He is an assistant professor of Computer Science department at Parul University, Vadodara, Gujarat, His teaching and research area includes RESTfull Web Services, Cloud Computing, Computer Network, Algorithm Design, Machine Learning, and Agile Methodology.

DR. M.B. POTDAR is a 1982 Ph. D. in Physics from Physical Research Laboratory of Dept. of Space, Govt. of India. Later for 28 years, he was associated with the Indian Space Research Organization (ISRO of Dept. of Space, Govt. of India) in various capacities. He worked extensively in development of land and atmospheric applications of Remote Sensing data. Since March 2011, he is holding position as Project Director at BISAG and organizing and steering research in various area of software development and applications of geo-spatial data.