January - February 2020
ISSN: 0193 - 4120 Page No. 5657 - 5664

Tem_357 Harnessing the Power of Digital Transformation, Artificial Intelligence and Big Data Analytics with Parallel Computing

[1,2]Hoe Min, [3]Bong Chin Wei, [4]Jamal Ahmad Dargham
[1]Graduate Student, School of Science and Technology, Wawasan Open University, Malaysia.
[2]Technical Manager, Hisco (Malaysia) Sdn. Bhd., Malaysia.
[3]Dean, School of Science and Technology, Wawasan Open University, Malaysia.
[4]Associate Professor, Computer Engineering Program, Faculty of Engineering, University Malaysia Sabah, Malaysia
[1,2]hoemin@hiscomalaysia.com.my,[3]cwbong@wou.edu.my, [4]jamalad@ums.edu.my

Article Info
Volume 82
Page Number: *5657 - 5664*
Publication Issue:
January-February 2020

Article History
Article Received: *18 May 2019*
Revised: *14 July 2019*
Accepted: *22 December 2019*
Publication: *27 January 2020*

Abstract

Traditionally, 2D and especially 3D forward modeling and inversion of large geophysical datasets are performed on supercomputing clusters. This was due to the fact computing time taken by using PC was too time consuming. With the introduction of parallel computing, attempts have been made to perform computationally intensive tasks on PC or clusters of personal computers where the computing power was based on Central Processing Unit (CPU). It is further enhanced with Graphical Processing Unit (GPU) as the GPU has become affordable with the launch of GPU based computing devices. Therefore this paper presents a didactic concept in learning and applying parallel computing with the use of General Purpose Graphical Processing Unit (GPGPU) was carried out and perform preliminary testing in migrating existing sequential codes for solving initially 2D forward modeling of geophysical dataset. There are many challenges in performing these tasks mainly due to lack of some necessary development software tools, but the preliminary findings are promising.

Keywords: *Geophysical Dataset, GPGPU, Jetson TX2, Parallel Computing.*

I. INTRODUCTION

This paper reviews the didactic concept in learning and applying parallel computing in academic and industrial environments. The content covers the background, history, concepts, architectures, current and potential applications, APIs, programming languages and models of supercomputing or High Performance Computing (HPC) notably parallel computing/Embedded Parallel Computing (EPC). Serial computing performance is limited in

scaling CPU frequencies to around 10 GHz and continues to increase power consumption; the advantages of parallel computing are best befitted for solving problems with multiprocessors (multi-core and many-core) and continue to increase transistor density according to Moore's Law.

In 1955 IBM 704[1],[2]created the first

commercial mainframe computer with floating-point capability of around 5 kFLOPS, Gene Amdahl who defined the Amdahl's laws in parallelization was the principal architect. To practise digital transformation that embraces Industry 4.0, Artificial Intelligence (AI) and Big Data Analytics with parallel computing, basic knowledge of HPC/parallel computing require to learn Linux and one or more programming languages and models available depending on the types of multi-core and many-core systems use, viz. Parallel libraries, POSIX Threads or Pthreads, JAVA and wrappers, C/C++, FORTRAN, MPI, OpenMP, OpenCL, SYCL, OpenACC, Python/Numpy/Numba/Keras, Julia, CUDA and so forth as well as for GPU accelerated computing and hybrid parallel schemes, MATLAB, Mathematical; R, Scala, Hadoop and others depending on the data size, applications, complexity and features require to implement on low cost PCs and laptops, edge/IoT devices, embedded systems, clouds or to high cost data centers and supercomputer facilities for Massively Parallel Processing (MPP) capable of providing many folds of performance and speed than sequential uniprocessing.

The embedded system utilizes HPC with upcoming systems having fused sensor processing and AI programming; GPUs are coming into the mix. There is a lot of work going on autonomous and self-aware robots and other demanding and critical applications that use embedded systems thus knowledge of HPC, GPU programming and EPC become essential to render performance and scalability with its advantages to work out problems quicker or address larger problems and perform many things simultaneously by using multiple computing resources such as standard computing resources like laptop, desktop or workstation and embedded systems.

The down side of Parallel Computing are programming to target parallel architecture is difficult and require proper apprehension and practice, various codes tweaking are demanded for different target architectures to meliorate performance otherwise often execute worse than their serial counterparts due to communication and coordination overhead. With MPP where in certain occasions can affect control algorithm and does not yield good results, require more efficient cooling and higher electric power consumption.

HPC depends on using parallelism to cater performance betterments and generally involves connecting to very large computing systems or supercomputing clusters and data centres. However, with the advent of technologies HPC can also deploy in laptops, desktops, cell phones, embedded systems and edge/IoT devices with multi-core CPU or many-core CPU and GPU (vector processor), these additional cores create several challenges that virtually all applications need to be "reprogrammed" to make use of these extra cores.

Linux[3] is main operating system use in this research, it is the most popular operating system for HPC, the entireworld's Top500[4]supercomputers operate on Linux and most of them enhanced with GPGPUs. The main reasons are open source and customizable to particular needs. Linux is developed by Linus Benedict Torvalds in 1991.According to Linux Foundation, Linux is planet earth's most prominent and largest pervasive open source software project in the history of computing, it is the operating system of choice to support cutting-edge technologies such as the Internet of Things, cloud computing, and big data that helps to transform industries and disrupt the status quo. Linux is scalable and portable to any hardware platform and deeply rooted in UNIX, but Linux Kernel Organization distributes the Linux kernel and other Open Source software to the public under a free software license called the GNU (Gnu's Not UNIX)General Public License (GPL) hence one can build GNU/Linux out of its source code and GNU-tool-chain's sources from *Linux From Scratch* distribution.

Today there are more than a thousand of Linux distros of different versions, it can be very confusing in terms of selection, but that is the attractive feature of Linux. A random listing of the types of Linux distros that are available free according to user levels are Ubuntu, Linux Mint, elementary OS, MX Linux, Zorin OS and Pop!_OS for beginners; Arch Linux, Gentoo, Slackware and Knoppix for advanced users; and Fedora(Redhat), Manjaro, Debian, Open SUSE for multipurpose usage on both desktops and servers depending on the Package Management Systems likeDPKG, gnome, APT-GET, APT-CACHE, KDE, RPM, CentOS,Yum, Zypper and others to install with resources, libraries, or other packages for all its dependencies.The enterprise versions of Ubuntu, SUSE Linux Enterprise, Redhat and others are available for clouds and data centres, but usually come with fees.

The main languages utilize for this research are C/C++ and CUDA C/C++.C is a procedural language revolves around the functions and was originated from a language called B developed by Ken Thompson, followed by Martin Richards meliorated with BCPL and adopted by Dennis Ritchie in 1972 and created the C. The purpose for C was to port the UNICS operating system (renamed UNIX) from DEC PDP-7 to the PDP-11 therefore not particularized to any specific applicationmake it well befit and effectual for various undertakings than many other powerful languages.C++ is a subset of the C. When Bjarne Stroustrup exercising work with Simula software for simulations and discovered its object-oriented programming paradigm was practicable for software development, but sluggish for pragmatic use thus began to work on "C with Classes" for its portability and supports object-oriented programming features like inheritance, polymorphism, abstraction, encapsulation, etc. which released in 1985 without enduring the loss of speed or low-level functionality. C/C++ allows implementing parallelism and can combine with Threads, MPI and OpenMP.

CUDA originated from Brook project in Stanford University is the acronym for Compute Unified Device Architecture created by Nvidia is a parallel computing platform and programming model using CUDA C/C++, CUDA Fortran, pyCUDA and Open ACC which are Nvidia GPU specific. Together with their Unified Memory comprising of single memory address space capable of accessing from any processor in a system and NVLink that comes with stacked memory enabling GPU applications to gain access to larger datasetand provide a higher bandwidth and high-speed path between GPUs communicate at peak data rates of 300 gigabytes per second (GB/s) which is 10 times faster than PCIebettering efficiency and computational throughput, and abridging the frequency of off-GPU transfersin the direction of exascale computing.

II. OTHER HPC PROGRAMMING LIBRARIES AND LANGUAGES

The study and research may need to apply some of other HPC programming libraries and languages when needed especially hybrid parallel schemes, e.g.

MPI + POSIX, MPI + OpenMP + CUDA, and CUDA + OpenMX, etc.

A. Standard Libraries for Parallel Programming

The simplest form of parallel programming is to make use of the parallel libraries; many are available in open source that can be appropriated for particular problem or needs.

B. POSIX Thread (Pthreads)

POSIX is theacronym for Portable Operating System and developed in 1980 to settle the portability issuebased on System V and BSD Unixstandard.

POSIX Threads or Pthreads[5]is an execution model in UNIX and Linux independent from a languageaccords a program to control overlap in time in multiple different flows of work to execute parallelism in shared memory multiprocessor architectures. Its C language threads API is specified in IEEE POSIX 1003.1c standard embracing routines on Thread Management, Mutex Variables, and Condition Variables.

C. JAVA and wrappers

James Gosling at Sun Microsystems (acquired by Oracle) developed JAVA to have the similar appearance and behavior of the C++ but easier to use and enforcean object-oriented programming model. The emerging interest in Java for HPC[6] is established on the attractivecharacteristics for programming multi-core cluster architectures and its inbuilt networking and multithreading support plusJava Virtual Machine (JVM) performance. Wrapper classes are to convert any data type into an object.

D. Fortran

Fortranstands for Formula Translator is the first high-level programming language in the world. It was developed in 1957 at IBM headedby John Backusas a programming tool for the IBM 704 mainframe. Today Fortranis still in use for many HPC applications and remains the preferred programming language for large-scale simulation of physical systems, climate models, and numerical calculations in science and engineering especially in array processing as not requiring other sophisticated data structures.

E. MPI

Message passing interface (MPI) began in 1991 during a retreat with a small group of researchers in Austria, the purpose was to create a standardized approachin running a parallel program written in C, C++ and Fortran across distributed memoryin exchanging messages between multiple computers. It is the most generic approach in using the classic MPI model with basic multicore nodes, butmemory can be a problem as CPU core counts increases.

MPI provides basic set of well-specified routines to parallel hardware vendors to expeditiously implement to produce higher-level routines not only for the distributed-memory communication environment to permit simple-to-use portable interface for the basic user, yet potent enough to provide experienced programmers to utilize the high-performance message passing operations on advanced systems.

F. OpenMP

OpenMP is abbreviation of Open Multi-Processing is the established standard in 1997 for portable API assigned for C/C++ and Fortran jointly defined and certified by major computer hardware and software marketers offers substantial benefits over both hand-threading and MPI to explicitly direct multi-threaded, shared memory parallelism with three primary API components, i.e. Runtime Library Routines, Compiler Directives and Environment Variables.

G. OpenCL and SYCL

OpenCL (Open Computing Language) was initially developed by Apple. Khronos in collaboration with CPU, GPU, embedded-processor, and software companiesAltera (acquired by Intel), AMD, Apple, ARM, IBM, Intel, Nvidia, Qualcomm, ST, TI, Xilinx, and others had its first version 1.0 public releasein 2008.

OpenCL is the open standard for parallel computing of heterogeneous system's framework for writing C and C++ programs that execute across CPUs, GPUs, DSPs (Digital Signal Processors), FPGA (Field-Programmable Gate Arrays) and other processors or hardware acceleratorsplatforms, its APIs control the platform and execute programs on various compute devices and provide standard interface on task- and data-based parallelism with headers and shared objectloaded at runtime for open source executions.

Nvidia, RapidMind and Gallium3Dimplementations of OpenCL use the front end Clang compiler are established on LLVM (Low Level Virtual Machine) compiler technology.

Khronos also developed the open standard SYCL in 2014.OpenCL and SYCL are similar to vendor-specific Nvidia CUDA, However, SYCL is the high-level single-source C++ domain-specific embedded language where as OpenCL is the low-level non-single source API.

Standard Portable Intermediate Representation or SPIR was initially created by Khronos for use by OpenCL and SPIR based on LLVM technology, but SPIR has evolved into a cross-API intermediate language with native support for shader and kernel lineaments used by APIs such as Vulkan named SPIR-V as first open standard to support OpenGL 4.6 extension. Even though SPIR–V does not use LLVM instead Khronos furnishes open source SPIR-V/LLVM conversion tools for building toolchains that apply to both intermediate languages.

H. OpenACC

OpenACC is short form for Open Accelerators developed in 2011 by CAPS, Cray, Nvidia, and PGI(Portland Group, acquired by Nvidia) for high-level heterogeneous parallel programmingthat comment directives in new and existing Fortranand C/C++ codes where the codes still remain portable and compile to run CPUs, GPUs/GPGPUsand APUs (Accelerated Processing Units) include both the CPU and GPU on a single chip.

I. Python/Numpy/Numba/Keras

Python is an interpreted, high-level and general-purpose programming language created by Guido van Rossum 1991as a successor to the ABC language which was developed atDutch Centrum Wiskunde & Informatica (CWI). Python looks and feels like MATLAB, but embarked in areas where MATLAB was ineffective. Python interpreter requires compiling the .py source file into a .pycbytecode and executing on the Python virtual machine.

Python is dynamically typed with define variable type and garbage-collected with automatic memory management to reclaim garbage or memory resided by objects that are no more needed by the program. It supports various programming paradigms with procedural, object-oriented, and functional programming available in multiple operating systems and is one of the most popular computing languages.

CPython is an open source reference implementation by community. Python Software Foundation (PSF)deals and directs resources for both Python and CPython developments.

Python being interpreter is sluggish in processing and not suited for numerical computations where as NumPy is the underlying package for scientific computing with Python contains powerful N-dimensional array object, sophisticated broadcasting functions with static array data structure, quick mathematical operations for multidimensional arrays and tools for random numbers and linear algebra.

Numba is a just-in-time compiler for Python that perform well on code that uses NumPy arrays, functions, and loops.

Keras is an open-source neural-network Python library for fast investigation with deep neural networks running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidMemphasizes on user-friendly, modularity, and scalability.

J. Julia

Julia began in 2009 by Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman of MIT and released Julia 1.0 in 2012is a fairy new high-level, high-performance, dynamic and general purpose computing language inherently parallelize for implementation and worthy to embed easily to any application for numerical analysis and computational science with its distinctive features that include a type system with parametric polymorphism and multiple dispatch as its core computing paradigm on concurrent, parallelism with or without the MPI and/or OpenMP type of threads. Its other characteristics included is tributed computing and direct calling of C and Fortran libraries.

Julia is garbage-collected engages eager evaluation, and effective libraries for floating-point computations, linear algebra, random number generation and regular expression matching. Julia’s high speed, capacity, functionality, flexibility and scalability make it the premier selection for GPUs and other supercomputers that employ accelerators.

K. MATLAB

MATLAB (Matrix Laboratory) was invented by Cleve Moler in 1970 to make less difficult of coding endeavours needed to develop amulti-paradigm numerical programming workplace and computing language that allows matrix manipulations, functions and data plotting, algorithms execution, user interfaces, and interfacing with other languages including C, C++, C#, Fortran, Java, Python and with various toolboxes like MuPAD engine to access symbolic computing powers, Simulink for multi-domain graphical simulation and model-based design for embedded and dynamic systems.

The freeware version similar to MATLAB is called Octave.

L. Mathematica

In 1968 Stephen Wolfram created Mathematical is a general multi-paradigm computational language called Wolfram Language utilizes computer algebra system to manipulate symbolic relationships in technical programming fields be that in neural networks, machine learning, image processing, data science, geometry and visualizations, etc.

M. R

R is a programming language primarily applied for statistical analysis that facilitates the analysis of Big Data using R code distributes across multiple systems and run on almost all operating systems with its excellent graphical capabilities for visualization patterns and associations within Big Data systems.

N. Scala

Scala is a general-purpose and popular programming language in data science, particularly in Big Data Analytics. Scala is to work with Spark, in fact the Apache Spark cluster computing solution is written in Scala. The opening of API endpoints can access to other languages with Scala superior concurrency support in parallelizing lot of the processing needed for large data sets. Scala operates on Java virtual machine (JVM) making best use with a framework like Apache Hadoop.

O. Hadoop

Hadoop was built by Doug Cutting who also created Apache Lucene open sourced web search engine from Apache Nutch project in 2002. Hadoop is a distributed file system and Map Reduce framework for big data where the primary hardware employs clusters of hundreds or thousands of commodity servers in peta byte range which is different from HPC that uses supercomputing clusters.

III. PARALLEL PROCESSING ON GEOPHYSICAL DATA

Parallel Computing for geophysical data processing and Big Data Analytics are widely use with traditional programing models like MPI, Fortran, JAVA and MATLAB, etc. including more recent Deep Learning (DL) neural network and cloud for large geophysical datasets such as pyGIMLi which is an open source library for modeling and inversion in geophysics[7], parallelizing large-scale geophysical applications in Java[8], parallel computing in seismic data processing[9], Apache Spark big data analytics scalable to seismic data analytics and computation[10], large-scale parallel geophysical algorithms in Java[11], Electromagnetic (EM) for 3D parallel inversion of time-domain airborne EM data[12], deep learning in electromagnetic inversion with convolutional neural networks[13], Magnetotelluric (MT) engaging a MPI + OpenMP + CUDA Hybrid Parallel Scheme for MT Occam Inversion[14], and joint inversion expending Parallel, large and dense matrix problems for application to 3D sequential integrated inversion of seismological and gravity data[15]plus many others. However, only rarely see articles employing C++ and in particular EPC with GPGPU as the technology and embedded systems only available in recent years.

The author initial attempt is to modify the MT2D MT Forward Modeling program written in C++with the permission from the developer Prof. Ren Zhengyong to CUDA platform and programming models with the Nvidia Jetson TX2 Developer Kitthat comes withPascal Architecture GPU, 2 Denver 64-bit CPUs + Quad-Core ARMA57 Complex, 8 GB L128 bit DDR4 Memory, 32 GB eMMC 5.1 Flash Storage and carrier board comprising various types of interfaces, interconnects and a camera. TX2 embedded system is to harness AI at the Edge. However, Nvidia PGI compilers and Tools for Open ACC and CUDA Fortran only available for Linux x86-64, IBM Open POWER, and Windows x64 CPU with Nvidia specific GPGPU, but not on TX2 as it uses ARM CPU. According to recent Nvidia announcement, PGI shall support ARM processors at the end of 2019.

For MT2D code, it make use of numerous third party software packages that include open source Eigen libraries for linear algebra, matrices, vectors, numerical solvers, and related algorithms; Intel MKL PARDISO Solver;Carnegie Mellon University (CMU)2D Mesh Generator and Delaunay Triangulator, and output to MATLAB for visualization.

In the meantime the author is working on alternate method with laptop PC consists of x86 CPU and Nvidia GPU to install with Ubuntu Linux OS, PGI and Python compilers and tools until such time PGI is able to support on TX2. With the release of CUDA Toolkit 10.1 in August 2019, the author’s aim is to use CUDA GPU-based Method for generating Delaunay Triangulations[16], CUDA cuBLAS library with standard basic linear algebra subroutines (BLAS), Eigensolver, CuSPARSE Sparse Matrix library and cuSOLVER library of dense and sparse direct solvers for GPU-accelerated implementation.

IV. DISCUSSION AND CONCLUSION

In this paper HPC/Parallel Computing/EPC encompasses its background, history, concepts, architectures, applications, Linux OS and various libraries and programmable languages and in particular for geophysical data processing, 2D/3D Forward Modeling and Inversion and Big Data Analytics are presented. Although presently lack of suitable and essential software tools for use with TX2, but foresee the growth of edge computing especially for high-end embedded systems is about to take a huge leap in generating their data outside the traditional data center or cloud.

REFERENCES

1. Wilson, Gregory V., "The History of the Development of Parallel Computing".
2. Wikipedia, "IBM 704".
3. M. K. Dalheimer, M. Welsh, Running Linux, 5th Edition, O’Reilly Media, Inc., 2006.
4. Top 500 Organization, "Top 500".
5. B. Barney, "POSIX Threads Programming".
6. Guillermo L. Taboada, et al., "Java in the High Performance Computing Arena: Research, Practice and Experience," Science of Computer Programming, May 2013.
7. Carsten Rücker, Thomas Günther, Florian M. Wagner, "pyGIMLi: An open-source library for modelling and inversion in geophysics," Computers and Geosciences, vol. 109, pp. 106-123, 2017.
8. M. Karrenbach, M. Jacob, M. Philippsen, "Parallelizing Large-Scale Geophysical Applications in Java".
9. D. Bhardwaj, S.Yerneni, S. Phadke, "Parallel Computing in Seismic Data Processing," in Proceedings Third Int. Pet. Conf. & Exbn.,, New Delhi, 1999.
10. Yuzhong Yan, Lei Huang, Liqi Yi, "Is Apache Spark Scalable to Seismic Data Analytics and Computations?," 2015.
11. Matthias Jacob, Michael Philippsen, Martin Karrenbach, "Large-Scale Parallel Geophysical Algorithms in Java: A Feasibility Study," Citeseerxf.
12. Liu Yun-He Liu, Chang-Chun Yin, Ren Xiu-Yan Ren, Chang-Kai Qiu, "3D parallel inversion of time-domain airborne EM data," APPLIED GEOPHYSICS, vol. 13, no. 4, pp. P. 701-711, 2016.
13. Vladimir Puzyrev, "Deep learning electromagnetic inversion with convolutional neural networks," arXiv:1812.10247v1 [physics.geo-ph] 26 Dec 2018.
14. Yu Liu, Renhao Xiong, "A MPI + OpenMP + CUDA Hybrid Parallel Scheme for MT Occam Inversion," International Journal of Grid and Distributed Computing, vol. Vol. 9, no. No. 9, pp. pp.67-82, 2016.
15. R. Tondi, C. Cavazzoni, P. Danecek, A. Morelli, "Parallel, ‘large’, dense matrix problems: Application to 3D sequentialintegrated inversion of seismological and gravity data," Computers & Geosciences, vol. 48, pp. 143-156, 2012.
16. Cristobal A. Navarro, Nancy Hitschfeld-Kahler, Eliana Scheihing, "A GPU-based Method for Generating quasi-Delaunay Triangulations based on Edge-flips," in GRAPP2013-International Conference on Computer Graphics Theory and Applications, 2013.