Proceedings of the 2023 International Conference on Software Engineering and Machine Learning
DOI: 10.54254/2755-2721/8/20230225

Review of architecture and model for parallel programming

Zihan Xia

Penn State University, State College, USA

zfx5078@psu.edu

Abstract. Parallel programming has become a critical aspect of modern computer systems and applications. With the increasing demand for faster and more efficient computing, it is essential to have multiple parallel architectures and parallel models to choose from. Different parallel architectures, like SISD and MIMD, perform differently and offer different benefits. It is essential to have an understanding of these architectures to select the appropriate one for the specific application being developed. Similarly, different parallel models such as OpenMP allow developers to write parallel programs using high-level constructs. By leveraging these models, developers can focus on the algorithmic aspects of the program rather than giving attention to low-level details. In conclusion, understanding the different parallel architectures and parallel models available is necessary to develop efficient parallel programs and improve the performance of modern computer systems.

Keywords: parallel programming, parallel architecture, parallel model, fortress, OpenCL.

1. Introduction

Moore's law, one of the most famous laws in computer science field, states that the number of transistors in an integrated circuit double in every two years. However, with the development of the computer, the limitation in developing of the computer architecture have appeared. Power consuming increase with the increasing of the processor's clock frequency, and the cooling problem of power consuming enlarge with the improvement of the processors' clock frequency [1]. As the result, the enormous increase of clock speed is not able to exist without the significant cost increase in cooling.

In these cases, some solutions that helping the development of computer circumvent the impediment in physical limitations are proposed. One of the most important ways in these solutions is parallel programming. Parallel programming is a technique that supporting multiple computations and processes to run in processor at the same time. In other words, it can divide a integrated process into several part and execute concurrently [2].

The usage of the parallel programming is ubiquitous in the modern world. The parallel programming technology have been deployed in a lot of daily used devices and popular computer system, such as the smart phone, personal computers, and the internet of things (IOT) [2]. The demands and requirements of parallel programming are wide and various. There are mainly three types of demand: 1. The application with intensive computation such as emulation of real-world events; 2. The application with

DOI: 10.54254/2755-2721/8/20230225

intensive data such as data mining; 3. The application with intensive network requirements such as remote control [3].

With the demand of parallel programming raise enormously in recent several decades years, the different architectures and models which can support the parallel programming are developed. And the various of them is confusing. So, this paper willing to provides a simple summary of the parallel programming techniques in the viewpoint of architecture and models.

2. Parallel architecture

The architecture of high-speed computer is classified into four different types, which is stated by Flynn in 1966 initially [4]. In Flynn's taxonomy, there are four kinds of architecture of computer [4]:

- SISD (single instruction, single data)
- SIMD (single instruction, multiple data)
- MISD (multiple instructions, single data)
- MIMD (multiple instructions, multiple data)

Following the development of computer technology, the computer's architecture is developed a lot compared with the 1966 when the Flynn first propose the four classifications. With the time flying, there are two more types of architecture appeared and join in the type of the computer architecture [5]:

- SMPD (single program, multiple data)
- MPMD (multiple programs, multiple data)

2.1. SISD

In SISD, one stream of instruction only processes one data stream. And it is the von Neumann model in computer architecture [6].

2.2. SIMD

In SIMD, the single instruction stream process multiple data stream concurrently. This architecture use data-level parallelism to implement that every part of the processor operates the same instruction but with different data. And in 1972 Flynn's taxonomy, Flynn create three subcategories for SIMD which are [7]:

- Array processor (SIMT)
- Pipelined processor (packed SIMD)
- Associate processor (predicted SIMD)

2.2.1. SIMT. SIMT is a combination of SPMD and multi thread. Compared with the SIMD, every thread in SIMT execute with lockstep. And there have already been deploy in some GPU such as the G80 architecture of Nvidia is the first one [8].

2.2.2. Pipelined processor. Every part in pipelined processor receives the same instruction and read needed data from central resource. After that, each parts process the data individually and send back to the central resource when the processing is finished. Pipelines consist with a series of segments including function, process, thread, etc. [3]. Pipelined processor can divide the whole task into several parts and running with different stages. Moreover, this technique uses in modern architecture such as Advanced Vector Extensions (AVX) as an extension of x86 instruction set [9].

2.2.3. Predicted SIMD. All units get the same instruction and make the independent decision based on the local data that if running this instruction or not.

2.3. MISD

MISD architecture means that the computer run multiple instructions in same data stream. This architecture is not a widely use one. The example of MISD is hardly ever [11]. However, there is still a

DOI: 10.54254/2755-2721/8/20230225

few examples of it. Some space shuttle control computers are used this technique to enforce the fault tolerance [12].

2.4. MIMD

In MIMD, each processing unit process their own data with their own instructions. This architecture is the widest used architecture in modern and huge number of modern parallel system is fitted with MIMD [6]. MIMD have some two subcategories [9]:

- Multi-core superscalar processor
- Distributed system

And MIMD can be applied with either single share memory space or a distributed memory space.

2.5. SMPD

SPMD is the mainstream of parallel programming, and every processor process their own different data with the same program [5]. SPMD, initially, use to support the development of IBM Research Parallel Processor Prototype (RP3) [13]. Back to 1984, the SPMD is different with the popular model, master-slave, in that time. SPMD is a general model, and it can support different instruction operate concurrently, and permitted the application-level parallelism and process self-controlling [14]. Message passing (MPI) is the first and popular deployment of the SPMD. More about MPI will be discussed in section 3.

2.6. MPMD

MPMD means that multiple programs operate with multiple data stream simultaneously. Normally, multiple processors run at least two independent programs separately. The MPMD have a node as “host” or “manager”. This manager node distributes missions to other node and the result return to manager after the processing [9].

3. Parallel programming model

The programming model is execution model of a set of API and particular code pattern. In other words, the parallel programming model is the abstraction of the computer architecture. There are two different kind of parallel programming model, the pure parallel programming model and the heterogeneous programming model [15]. The heterogenous programming model consists a structure that a host CPU control and distribute tasks to other processing unit [16]. Moreover, the host CPU usually use the different instruction set with the other processing unit and this is why it called the heterogenous [16]. On the other hand, the pure parallel programming model is more classical, and two famous examples are OpenMP and MPI (message passing). Moreover, there are multiple parallel programming model existed to support the parallel program’s development. And they have different features and different way to support such as library and complier. In this section, it mainly focusses on describe and introduce the different parallel programming models.

3.1. Pthread

POSIX threads (Pthreads) is a set of header file or functions support thread programming. This model and library support programmer obtain an access to OS to run thread operations such as create and join. Pthreads use shared memory and shared address in architecture and communication model. The pthread provide mainly four ways in thread programming, which is:

- Thread management such as create and join.
- Mutexes
- Conditional variable
- Synchronization

Because Pthread is model of thread programming, they are lighter in parallelization of program. One process can contain multiple thread and the multiple threads in one process share heap can let the communication between thread is easier than the communication between processes. Pthreads is a low-

DOI: 10.54254/2755-2721/8/20230225

level collection and library, which bring the flexibility for Pthread. On the other hand, the low-level provide some disadvantages with their advantages. The low-level structure forces programmer to manually do some tasks include synchronization and mutexes [17]. And this feature may let the pthread's using is not as simple as some other model and library in parallel programming.

3.2. OpenMP

OpenMP (Open Multi-Processing) is an API which define a portable and extendible parallel programming model [18]. OpenMP provides support for parallel programming in languages that C, C++ and Fortran. And it has been used in parallel programming construction in the platform that from the personal computer to supercomputer [18]. OpenMP utilize the shared memory programming model, shared memory system architecture and the shared address communication model [15]. And the working unit is thread. OpenMP is a easy to use and having an implicit work management. In other words, when using the OpenMP, the programmer only needs to put the key words like "#pragma omp parallel" and the OpenMP will automatically parallelize the code by reading the key words. These kinds of design, let programmer to escape the complicated manual thread management. All in all, the OpenMP have already became one of the most important and popular parallel programming model due to its performance, availability and simplicity in coding.

3.3. MPI

MPI (massage passing interface) is one of the most important parallel programming models. MPI use the massage passing model, and it uses both distributed and shared memory in system architecture [15]. Moreover, the communication model of MPI is message passing or shared address [15]. The MPI standard includes [19]:

- Point-to-point message passing
- Collective communications
- Group and communicator concepts
- Process topologies
- Environmental management
- Process creation and management
- One-sided communication
- Extended collective operations
- External interfaces
- I/O
- Some miscellaneous topics
- A profiling interfaces.

MPI support the language that C, C++ and Fortran. The working unit of MPI is processes. In MPI, worker management do not need to be done by programmer manually, but the workload partitioning, and task mapping must do by programmer manually [20]. MPI is also easy to use by adding the mpirun which is a command line tool, and it can manifest that the number of processes can be used during runtime.

3.4. UPC

UPC (Unified Parallel C) is a C language's extension, and it is face to the high-performance computing on large-scale parallel machines [21]. This language and model support the same programming model for shared and distributed memory. UPC apply the SPMD parallel architecture. For parallelism, UPC extends the ISO C99, which is a standard of C programming language, to that [21]:

- An explicitly parallel execution model
- A shared address spaces.
- Synchronization primitives
- A memory consistency model

DOI: 10.54254/2755-2721/8/20230225

- Parallel utility libraries.

One of the features of the UPC is that UPC make the address space to be shared and logical partitioned between the different process in system, but the local variable still exists in their own processes' address space physically [20]. In other words, programmer can code the program with a view of global address space. The workload distribution of UPC is implicit which means that the programmer does not need to manipulate the distribution manually and they only need to add some key words in code to indicate the parallelism. However, UPC can choose either implicit or explicit in the workload partitioning and worker mapping which bring some flexibility in using of UPC [20].

3.5. CUDA

CUDA (Compute Unified Device Architecture) is an platform and programming model for parallel computing for computation in GPU (graphical processing unit) which is developed by NVIDIA [22]. The CUDA is initially published by NVIDIA in 2006, and until now, this programming model is became widely used. For example, Adobe, MATLAB MathWorks and Wolfram Mathematical use CUDA to develop[22]. In CUDA, a host (i.e. CPU) and a computation resource (i.e. GPU) construct the parallel system [15]. GPU run the tasks in several parallel threads. The GPU with CUDA have two-level hierarchy design which is block and grid. Blocks consist of a set of threads which distinguished by thread id. Grids consist of a set of Blocks and different grid have close size and dimension in one system. Compared with classic general computing with graphical API, the CUDA have some advantages that [23]:

- Scattered reads
- Unified virtual memory and memory
- Shared memory
- Faster download and readbacks to and from GPU
- Full support for integer and bitwise operations

3.6. OpenCL

OpenCL (Open computing Language) is an open and royalty-free standard for parallel programming [24]. It uses to code the heterogenous parallel program in different platform, such as CPU and GPU. All in all, the OpenCL provides an API for parallel computing based on task-level and data-level parallelism. OpenCL is initially developed by Khronos group and Apple in 2009. And from 2009 to now, the OpenCL have already been used or deployed in multiple companies' products and multiple applications such as arm and AMD use OpenCL.

The OpenCL have an impressive advantage which is portability. It allows that OpenCL running in a lot of application and architectures as soon as they supported it. However, the enforcement of portability brings some bad manifestation of OpenCL. Because the developer wants the OpenCL run different platform, the OpenCL hardly have support in some special design in some hardware platform. In addition, compared with the opponent with similar function, CUDA, the performance of the OpenCL does not have large advantages. There is research finds that performance of CUDA is thirty percent better than the OpenCL at most [25].

4. Conclusion

In conclusion, with the physical limitations of traditional computing systems, parallel programming has become an essential aspect of modern computer systems and applications. Parallel programming allows multiple computations and processes to run simultaneously, thus improving the performance and efficiency of the system. It is necessary to have an understanding of different parallel architectures, such as SISD and MIMD, and parallel models, like OpenMP and MPI, to develop efficient parallel programs. With the increasing demand for faster and more efficient computing, parallel programming will continue to play a critical role in the development of future computer systems.

DOI: 10.54254/2755-2721/8/20230225

References

[1] Manferdelli JL, Govindaraju NK, Crall C. Challenges and opportunities in many-core computing. Proceedings of the IEEE, 2008, 96(5):808-815.

[2] Gerencer T. Parallel computing and its modern uses: [J]. HP Tech Takes, 2019, 30(October).

[3] Li Y, Zhang Z. Parallel Computing: Review and Perspective[C]. In: 2018 5th International Conference on Information Science and Control Engineering (ICISCE). IEEE, Zhengzhou, China, 2018:365-369.

[4] Flynn MJ. Very high-speed computing systems. Proceedings of the IEEE, 1966, 54(12):1901-1909.

[5] Algorithms and Theory of Computation Handbook. CRC Press LLC, 1999. "single program multiple data", in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed. 17 December 2004. Available from: https://www.nist.gov/dads/HTML/singleprogrm.html. Retrieved March 16, 2023.

[6] Mattson TG, Sanders BA, Massingill BL. Patterns for parallel programming. Elsevier, 2013.

[7] Flynn MJ. Some Computer Organizations and Their Effectiveness. IEEE Transactions on Computers, 1972, 21(9):948–960.

[8] NVIDIA Corporation. Nvidia Fermi Compute Architecture Whitepaper [EB/OL]. (2009)[2014-0717].https://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf.

[9] Wikimedia Foundation. Flynn's taxonomy. Wikipedia(Retrieved February 24, 2023). https://en.wikipedia.org/wiki/Flynn's_taxonomy#Single_instruction_stream,_multiple_data_streams_(SIMD).

[10] Spector A, Gifford D. The space shuttle primary computer system. Communications of the ACM, 1984, 27(9):872-900.

[11] Tanenbaum AS, Austin T. Structured computer organization. 6th ed. Prentice Hall, 2013:591.

[12] Spector A, Gifford D. The space shuttle primary computer system. Commun ACM, 1984, 27(9):872-900.

[13] Pfister G et al. The research parallel processor prototype (RP3). Proceedings of the IBM Kingston Parallel Processing Symposium, 1984, Nov 27-29 (IBM Confidential); and Proceedings of the ICPP, August 1985.

[14] Darema F. SPMD Computational Model. In: Padua D(ed). Encyclopedia of Parallel Computing. Springer, Boston, MA, 2011.

[15] Diaz J, Munoz-Caro C, Nino A. A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era. IEEE Transactions on Parallel and Distributed Systems, 2012, 23(8):1369-1386.

[16] Shan A. Heterogeneous Processing: a Strategy for Augmenting Moore's Law. Linux Journal, 2006.

[17] Pervan B, Knezović J. A Survey on Parallel Architectures and Programming Models. In: 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO). Opatija Croatia, 2020:999-1005.

[18] OpenMP. http://www.openmp.org. Retrieved March 18, 2023.

[19] MPI Forum. MPI: A Message-Passing Interface Standard Version 3.1. http://mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf. Retrieved: 03/18/2023.

[20] Kasim H, March V, Zhang R, See S. Survey on Parallel Programming Model. In: Network and Parallel Computing. NPC 2008. Lecture Notes in Computer Science, vol 5245. Springer, Berlin, Heidelberg, 2008.

[21] UPC Team. Berkeley UPC - Unified Parallel C. https://upc.lbl.gov/. Retrieved March 18, 2023.

[22] NVIDIA. CUDA Zone. https://developer.nvidia.com/cuda-zone. Retrieved March 18, 2023.

[23] Wikimedia Foundation. CUDA. Wikipedia(Retrieved February 15, 2023). https://en.wikipedia.org/wiki/CUDA.

[24] The Khronos Group Inc. OpenCLTM. https://www.khronos.org/opencl/. Retrieved March 18, 2023.

DOI: 10.54254/2755-2721/8/20230225

[25] Fang J, Varbanescu AL, Sips H. A Comprehensive Performance Comparison of CUDA and OpenCL. In: 2011 International Conference on Parallel Processing. Taipei Taiwan, 2011:216-225.