International Journal of Research in Engineering, Science and Management
Volume-2, Issue-2, February-2019
www.ijresm.com | ISSN (Online): 2581-5792

Parallel Computation for Layered Model based on Heterogeneous System

Pranita A. Bhagat[1], Vaishnavi D. Bante[2], Shyam R. Ayyar[3], A. K. Gaikwad[4]
[1,2,3]*Student, Department of Computer Science and Engineering, DESSCOET, Dhamangaon Rly., India*
[4]*Professor, Department of Computer Science and Engineering, DESSCOET, Dhamangaon Rly., India*

***Abstract*: The general layered heterogeneous parallel computation model was composed of parallel algorithm design model, parallel programming model, parallel execution model, and each model correspond to the three computing phases respectively. The properties of each model were described and research spots were also given. In parallel algorithm design model, an advanced language was designed for algorithm designers, and the corresponding interpretation system which based on text scanning was proposed to map the advanced language to machine language that runs on the heterogeneous software and hardware architectures. The advanced version allows the programmer to define at runtime all the main features of the underlying parallel algorithm, which have an impact on the application execution performance, namely, the total number of participating parallel processes, the total volume of computations to be performed by each of the processes, the total volume of data to be transferred between each pair of the processes, and how exactly the processes interact during the execution of the algorithm. Such an abstraction of parallel algorithm is called a network type.**

***Keywords*: Parallel programming model, parallel execution model, Heterogeneous network of computers; Heterogeneous parallel computing.**

1. Introduction

Heterogeneous networks of computers are the most general and common parallel architecture. In the most general case, a heterogeneous network includes PCs, workstations, multiprocessor servers, clusters of workstations, and even supercomputers. Unlike traditional homogeneous parallel platforms, the heterogeneous parallel architecture uses processors running at different speeds. What is even more important, the processors demonstrate different relative speeds on different code mixtures. Speeds of data transfer between different processors in heterogeneous networks can also differ significantly. Communications between processors of the same shared-memory multiprocessor server will be much faster than communications between processors of different workstations. It makes programming heterogeneous platforms a challenging task. Data, computations, and communications should be distributed unevenly to provide the best execution performance. Parallel computation exists for many years, but it was employed mainly in high-performance computation field. While due to the reduction on hardware costs, interest in parallelism has increased. As the frequency scaling is restricted by physical constraints, and the power consumption of computers has also become a concern in recent years, multi-core processors has predominated in computer architecture. Researchers also shifted their focus from clusters to multi-core CPUs and GPUs. As is known, parallelized programs are usually realized in three steps:

- The algorithm designer describes the given problem as a numeric or nonnumeric problem.
- The program designer writes the parallel program with a certain parallel programming language.
- The program runner compiles and runs the program on a specific parallel platform to complete the computation and solves the problem.

The performance of the final parallel program is affected by many factors: designing of algorithm, program implementation, processor architecture, communication between threads, operating system, network bandwidth etc. To build the optimized parallel program, all influence factors should be parameterized and taken into consideration. Since there are too many parameters representing the different characteristics in the parallel platform, the description of the single model would be more and more complicated. Eventually the cost function would be too complicated to solve. By contrast, a layered model not only fits the heterogeneous computation platform more precisely, but also simplifies the problem. Work on parallel computation model has been done since 1970s. The first generation models focused on computation, including PRAM (Parallel Random Access Machine) model based on SIMD and APRAM(Asynchronous Parallel Random Access Machine) based on MIMD. The second generation models focused on network communication.

2. Overview of the system

A. Hardware trends in heterogeneous system designs

Some future trends in the heterogeneous system designs are evident from USA's DOE plans for the next generation of supercomputers. The aim is to deploy three different platforms by 2018, each with over 150 peta flops of peak performance [18]. The Aurora system, on the other hand, will offer a more homogeneous model by utilizing the Knights Hill Xeon Phi architecture, which, unlike the current Knights Corner, will be

a stand-alone processor and not a slot-in coprocessor, and will also include integrated Omni-Path communication fabric. All platforms will benefit from recent advances in 3D-stacked memory technology, and promise major performance improvements:

- CPU memory bandwidth is expected to be between 200 GB/s and 300 GB/s using HMC.
- GPU memory bandwidth is expected to approach 1 TB/s using HBM.
- GPU memory capacity is expected to reach 60 GB (NVIDIA Volta).
- NV Link is expected to deliver from 80 up to 200 GB/s of CPU-to-GPU bandwidth.
- In terms of computing power, the Knights Hill is expected to be between 3.6 and 9 teraflops, while the NVIDIA Volta is expected to be around 10 teraflops.

B. Basic model of heterogeneous parallel algorithm

The language is an ANSI C superset designed specially for programming parallel computations on common networks of diverse computers. The main goal of parallel computing is to speed up solving problems on available computer resources. Just this differs parallel computing from distributed computing, the main goal of which is to make different software components, inherently located on different computers, work together. In the case of parallel computing, partition of the whole program into a number of distributed components located on different computers is just a way to speed up execution of the program not its inherent property. Therefore, when designing the language, the primary attention was paid to the means that facilitate development of high efficient and portable programs solving single problems on common networks of computers. A parallel program running on the network of computers is a set of processes interacting (that is, synchronizing their work and transferring data) by means of message passing. Source code does not specify how many processes constitute the parallel program as well as which computer runs one or another process. This is done by some means external to the language when the program is started up. Source code only describes which computations are performed by each of the processes constituting the program.

3. System analysis

A. Architecture of the parallel computation model

Parallel computers can be roughly classified according to the level at which the hardware supports parallelism, with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, massively parallel processing (MPPs) and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks. Parallel computer programs are more difficult to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically one of the greatest obstacles to getting good parallel program performance. Parallel computing is usually accomplished by breaking the problem into independent parts so that each processing element can execute its part of the algorithm simultaneously with the others. The processing elements can be heterogeneous and include resources such as a single computer with multiple processors, several networked computers, specialized hardware, or any combination of the above.

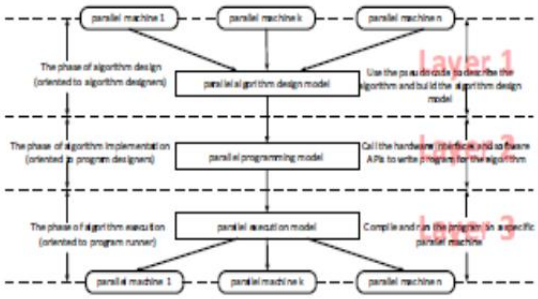

Fig. 1. Architecture of general layered heterogeneous model of parallel computation

Fig. 1 shows the architecture of the general layered heterogeneous model of parallel computation. In the general layered heterogeneous model of parallel computation, the whole model of a parallel computation would be divided into three child models: parallel algorithm design model, parallel programming model and parallel execution model. In the parallel algorithm design model, algorithm designers focus on the abstract parameters of the parallel machine. In this layer, algorithm designers do not need to concern about the concrete realization of algorithms, which is different from the traditional parallel development. In the parallel programming model, based on the different interfaces of hardware and application programming interface (API) of software, program designers design the corresponding parameter library and method library for parallel program. In the parallel execution model, through the appropriate compiler, parallel program would be compiled into machine language that runs on the corresponding hardware and software architecture. The parallel computing model processes as shown in Fig. 2.

In the layer of algorithm design, developers first design the parallel algorithm to describe the target problem; after being processed through the interpretation system, the parallel algorithm is interpreted into parallel program, which is based on different hardware and software architectures; furthermore, with reference to the current system software and hardware parameters, cost function and calculating behavior, system will optimize the parallel functions and improve the efficiency; after being processed though the build system the parallel parts would be compiled into the corresponding machine language that runs on different hardware. At last, multiple hardware

architectures (e.g. cluster, multi-core CPU, multi-core GPU) could work together to complete the computing tasks.

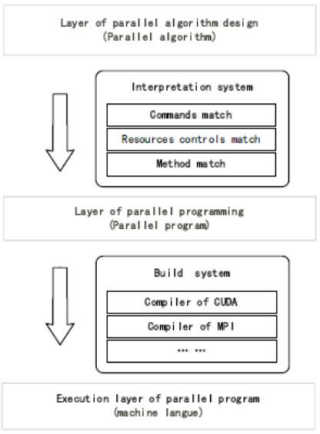

Fig. 2. Process of the general layered heterogeneous model of parallel computation

B. OpenCL design

Open CL follows a "close-to-the-silicon" approach, meaning that it is as close to the hardware implementation as possible, with just enough abstraction to make the API vendor and hardware neutral. In a OpenCL system, all computation resources in a host system are seen as peers. These resources can include CPUs, GPUs, mobile processors, microcontrollers and DSPs. OpenCL support both data and task parallel compute models, and have clearly defined floating point representation (IEEE 754 with specified rounding and error). The resultant executable is capable of executing on a number of devices, dynamically allocating computation resources available. OpenCL defines a set of models and a software stack. The models specify how OpenCL is constructed, and how data and tasks are handled. The software stack on the other hand, shows a general development work flow, indicating how a developer should utilize the OpenCL libraries. Other than the standard OpenCL, there is also a Embedded version of the specification, named OpenCL Embedded Profile. Compared to the full specification, there are a number of notable differences

- Embedded profile devices do not support any 64 bit data. This means that there are no double, along, and some vector data types
- Embedded profile devices do not have to support 3D operations.
- EEE 754 rounding requirement is not implemented as strictly, full IEEE 754 require the rounding to be to the nearest even number, however in Embedded profile, only the basic mode of IEEE 754 is supported, meaning all rounding can be done to the nearest zero.

4. Conclusion

The development and evolution of the parallel computing model, based on the model's functions and object-oriented feature, it is sufficient and necessary to divide the parallel computing model into three layers which are parallel algorithm design model, parallel programming model and parallel execution model. Parallel method library and parameter library were also provided to achieve the comprehensive utilization of different and heterogeneous computing resources and assign parallel tasks reasonably to reduce the complexity of parallel development.

References

[1] C.Y. Lin, J.S. Liu, and Y.C. Chung, S. Q. Chen, and L. T. Yang, "HPM: a hierarchical model for parallel computations," International Journal of High Performance Computing and Networking, 2004, vol. 1, pp. 117–127.

[2] J.S. Liu, and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," Proceedings of the 6th Symposium on Operating System Design and Implementation. SanFrancisco, USA: USENIX Association, 2004, pp. 137-150.

[3] A.J.C. Bik, B. A. Sanders, B. L. Massingill, Patterns for Parallel Programming, USA: Addison-Wesley Professional, 2004.

[4] X. H. Sun, "Scalable computing in the multicore era," Proceedings of the Inaugural Symposium on Parallel Algorithms, Architectures and Programming, Hefei: University of Science and Technology of China Press, Sep. 2008, pp. 118.

[5] Yunquan Zhang, Guoliang Chen, Guangzhong Sun, and Qiankun Miao, "Models of parallel computation: a survey and classification," Frontiers of Computer Science in China, 2007, vol. 1, no. 2, pp. 156–165.

[6] Guoliang Chen, Guangzhong Sun, Yunquan Zhang, and Zeyao Mo, "Study on Parallel Computing," Journal of Computer Science and Technology, Special Issue Dedicated to the 20th anniversary of NSFC, 2006, vol. 21, (5), pp. 665–673.

[7] M. Isard, M. Budiu, and Y. Yu, "Dryad distributed data-parallel programs from sequential building blocks," ACM SIGOPS Operating Systems Review, 2007, vol. 41, no. 3, pp. 59–72.

[8] Guoliang Chen, Qiankun Mao, Guangzhong Sun, and Yun Xu, "Layered models of parallel computation," Journal of University of Science and Technology of China, vol. 38, June 2008.