Analysis on Heterogeneous Computing

Changxu Song
Mechanical Design manufacture and Automation Major, Beijing University of Chemical Technology, Beijing, 102202, China
*Corresponding author’s e-mail: gaoming@cas-harbour.org

Abstract. In the Internet industry, with the popularization of informatization and the rapid increase in data volume, people have new requirements for storage space. At the same time, computer applications such as artificial intelligence and big data have rapidly increased demand for computing power and diversified application scenarios. Heterogeneous computing has become the focus of research. This article introduces the choice of architecture for heterogeneous computing systems and programming languages for heterogeneous computing. Some typical technologies of heterogeneous computing are illustrated, including data communication and access, task division and mapping between processors. However, this also brings difficulties. The challenges facing hybrid parallel computing, such as programming difficulties, poor portability of the algorithm, complex data access, unbalanced resource load. Studies have shown that there are many ways to improve the status quo and solve problems, including the development of a unified programming method, a good programming model and the integration of storage and computing, intelligent task allocation, as well as the development of better packaging technologies. Finally, the application prospects and broad market prospects of heterogeneous computing systems are prospected. In the next ten years, due to the various advantages of heterogeneous computing systems, innovation in more fields will be stimulated and heterogeneous computing systems will shine in the AI artificial intelligence fields such as smart self-service equipment, smart robots, and smart driving cars. Moreover, this emerging technology will bring new industries and new jobs, thereby driving economic prosperity and social development and even benefiting the entire human society.

1. Introduction
Human beings are at the dawn of the heterogeneous computing era. With the rise of machine learning, artificial intelligence, unmanned driving, industrial simulation and other fields has caused CPUs to encounter more and more bottlenecks, such as low parallelism, insufficient bandwidth, high latency, being constrained by power consumption etc, especially when processing massive calculations and massive data/images. Additionally, as silicon chips have approached the limits of physics and economic costs have risen in recent years, Moore's Law has become ineffective. The combination of Moore's Law and the power wall limits processor design. Because today's technology cannot increase the frequency, performance is improved primarily by increasing the number of transistors. It is the balance between performance gains and development costs that has advanced processors into their current designs, and most transistors are now dedicated to massive caches and complex logic for instruction-level parallelism (ILP). Increasing the cache size or introducing more ILP will produce too little performance gain compared to the development cost. In order to meet the needs of diversified computing, more and more hardware are introduced such as GPU and FPGA for acceleration, and heterogeneous computing has emerged. The so-called heterogeneous refers to various computing units such as CPU, DSP, GPU, ASIC,
FPGA, etc., using different types of instruction sets and different architectures to form a hybrid system that performs special calculations. This method is called "heterogeneous computing". Because the instruction system of CPU and accelerator is different, it is necessary to modify the source code according to the difference of instruction system, modify the source code, and then compile it into different programs and distribute them to heterogeneous processors for execution, which increases program writing. The complexity of task allocation and data communication. In addition, the collaborative work between general-purpose processors and dedicated accelerators places higher requirements on task division and coordination, data communication and synchronization. With the diversification of computing application scenarios, various smart devices such as cloud computing are connected to the Internet, which makes the hardware and structure faced by computing increasingly diverse, and promotes the increasingly heterogeneous computer architecture. This paper analyzes the development history and research status of heterogeneous computer architectures, heterogeneous processors and programming languages, and introduces the main challenges and key technologies of heterogeneous computing, and finally gives possible future research directions.

2. Architecture selection of heterogeneous computing systems

The heterogeneous system composed of heterogeneous acceleration devices and the host has a variety of options. Roughly speaking, the low-power architecture is usually CPU combined with ASIC, however, CPU combined with GPU and CPU combined with FPGA has better general performance. The current way to increase the computer performances is focus on the parallelism instead of the clock frequency and that means to improve performance, the multi-chip, multi-core, or multi-context parallelism are significant. Additionally, according to the Flynn’s taxonomy, there are four levels of parallelism in hardware and different heterogeneous computing architectures contain different levels of parallelism.

First one is SISD (Single Instruction Single Data). As shown in figure 1, SISD refers to one instruction processing one data and all single processor computers of von Neumann architecture are of this kind. Second one is SIMD (Single Instruction Multiple Data). As shown in figure 2, it means one instruction broadcast to multiple processors, but each processor has its own data. This system is used in digital signal processing, vector operation processing, and its parallel function is usually completed by the compiler. For example, the single-chip CBEA consists of a traditional CPU core and 8 SIMD accelerator cores. It is flexible and each core can run a separate program in MPMD mode and communicate via a fast on-chip bus. Its main design criterion is to maximize performance while consuming minimum power.
Third one is MISD (Multiple Instruction Single Data). Multiple instructions correspond to single data. Unfortunately, no system is designed in accordance with this structure. This kind of classification is only put forward for the sake of complete classification.

Last one is MIMD (Multiple Instruction Multiple Data). Multiple instructions correspond to multiple data, and each processing unit has independent instructions and data. The two subdivisions of MIMD are single-program multi-data (SPMD) and multi-program multi-data (MPMD). For example, figure 3 shows a GPU with a 30-degree multi-threaded SIMD accelerator core and a standard multi-core CPU. GPU has very superior bandwidth and computing performance, and optimized for little or no SPMD program running in synchronism. It is designed for high performance graphic, wherein the throughput of data is essential.

3. Heterogeneous programming languages extended from existing languages

3.1. Cuda
CUDA is a parallel computing platform and programming model invented by NVIDIA. It can greatly improve computing performance by using the processing power of a graphics processing unit (GPU). CUDA is an extension of the popular C programming language, and programmers can use heterogeneous computing systems including CPUs and massively parallel GPUs. For CUDA programmers, the computing system consists of a host and one or more devices. The host is a traditional CPU and the device is a processor with a large number of arithmetic units. In GPU-accelerated applications, the logical execution part of the workload runs on the CPU, while the intensive computing part of the application runs in parallel on thousands of GPU cores. When using CUDA, developers can use popular languages, such as C/C++/Python/MAT-LAB for programming, and express parallelism through the expansion of some basic keywords. Although CUDA is an extension of C, CUDA needs to explicitly manage the data transmission between the host memory and various device memories, and it needs to tune the GPU memory utilization according to the underlying GPU architecture.

So far, the sales of CUDA-based GPUs have reached millions. Software developers, scientists, and researchers are using CUDA in various fields, including image and video processing, computational
biology and chemistry, fluid dynamics simulation, and CT image reconstruction, Seismic analysis and ray tracing, etc. [Nvidia official website].

3.2. **Opencl**
OpenCl is an open standard for maintaining the Khronos family. OpenCL is based on the C language standard and can provide cross-platform parallel computation APIs. OpenCL aims to develop portable parallel applications for systems with different computing devices, all OpenCL code can be moved between devices. The development of OpenCL is to meet the rapidly growing demand for high-performance standard application development platforms of various parallel computing platforms and bypasses the critical limitations of previous programming models of parallel computing systems that differed in terms of application portability.

3.3. **Comparison**
CUDA is the most comprehensive extensibility method for GPU programming, but it lacks portability and requires a lot of coding and optimization to achieve the best performance of the architecture. OpenCL standard represents the research direction of general programming interface for heterogeneous devices, and many manufacturers have joined the OpenCL standard development team. However, the programming model is not simple and the usability is limited. Using the graphics processor as Clusters with heterogeneous nodes for accelerators requires very complex programming. Because in addition to the underlying technologies of MPI and SHMEM, this type of cluster also requires the use of CUDA or OpenCL base technologies. What is developing and coming is new processors with a large number of cores and effective implementation of them requires a new programming model. This is because developers of new parallel programming languages do not have enough time to keep track of the architectural diversity of multi-core processors. The possibility of designing and promoting a new high-level language within 5-10 years is minimal, so programmers will have to use mixed languages to combine various parallel programming models.

3.4. **Independent programming language for specific heterogeneous mixed structure**
Chestnut proposed by Stromme et al [1], a domain-specific GPU parallel programming language for parallel multidimensional grid applications. Chestnut aims to greatly simplify the programming process on the GPU and provide a powerful language for people with or without programming experience. In addition, Chestnut has an optional GUI programming interface, even novice programmers can use GPU computing. The language provides a simple parallel structure that allows Chestnut programmers to "think in order" when expressing his Chestnut program; programmers do not have to think about parallelization, data layout, GPU to CPU memory transfer and synchronization. The performance results of the Chestnut prototype implementation by Stromme et al. show that for a set of multiple parallel applications, the performance of the Chestnut application is almost as good as the handwritten CUDA code. In addition, Chestnut code is much simpler and smaller than handwritten CUDA code. The program in Chestnut consists of serial code and parallel loops on parallel data. Chestnut's compiler converts the program written in Chestnut into C++ source code by calling Walnut library functions. Auerbach et al. [2] designed the language LIME, which can be adapted to heterogeneous computing architectures. LIME provides a stream computing model that can support hybrid computing of multiple heterogeneous processors such as GPUs and FPGAs. Linderman et al. [3] proposed a universal programming model-merge framework. Merge offers a parallel computing mode similar to Map-Reduce mode. This can provide high-level parallel programming descriptions for heterogeneous computing platforms.

4. **Advantages of heterogeneous computing**

4.1. **Parallel task division technology**
According to different computing modes, there are two main modes for dividing parallel tasks between
heterogeneous processors. One is the MP mode, that is, different task programs are executed on different processors. Researchers [4-6] analyzed the processor characteristics of heterogeneous architectures and divided the computing tasks to reduce communication overhead and improve bandwidth and peak performance. The other is the MD mode, in which the same task program is executed on different processors, but the data processed is different. For example, a mainstream cooperation of the CPU + accelerator mode is: the CPU performs logic control tasks, and the accelerator performs a large number of calculation tasks; another less common cooperation is: the CPU and the accelerator perform the same calculation task, only the calculated value different. The division of parallel tasks should pay attention to the problem of load balancing on heterogeneous processors; additionally, the assigned computing tasks must be consistent with the computing characteristics of the processors, so that the computing efficiency of heterogeneous processors can be fully utilized. The division of tasks can be determined according to the characteristics of the processor and the nature of the task. When dividing according to the characteristics of processors, improving the utilization rate of each processor is the ultimate goal.

4.2. Data communication technology between heterogeneous processors
Some dissimilar processors communicate through the bus; some communicate directly through the network. There are two modes of communication between heterogeneous processors: synchronous and asynchronous. Asynchronous communication can realize the overlap of communication and calculation and hide the communication time. For different bus communication methods, Li et al. [7] proposed that for the CPU-GPU heterogeneous architecture, simply unloading the kernel library to the GPU will cause a large amount of data to pass through the low-speed PCIe bus, and the network communication overhead also affects its scalability. Without solving the above problems, Shui et al. [8] proposed a design example of a fine-grained algorithm centered on GPU and proved its effectiveness. Zhang et al. [9] established a real CPU + MIC heterogeneous cluster and analyzed its performance and behavior by examining different communication methods, such as message delivery methods and remote direct memory access.

5. Problems and the corresponding solutions in the field of heterogeneous computing

5.1. Programming difficulties
Because multiple computing devices in heterogeneous systems have different system architectures, instruction sets, and programming models, heterogeneous systems often have different programming models from CPUs. Generally. Two sets of different program codes are required for heterogeneous hybrid computer systems, which increases the difficulty of application development and reduces the portability of codes. Secondly, the expansion of the scale of heterogeneous systems has led to a sharp increase in the number of computing cores in the system, which puts forward higher requirements on the algorithm itself, and higher programming requirements. In response to this problem, a common approach is to use the CPU/GPU coordinated calculation, one of which is that the CPU is only responsible for managing the work of the GPU, such as distributing data and coordinating management, while the GPU is responsible for all calculations. The other is to let the CPU also take on part of the calculation task, and complete the calculation together with the GPU. In addition, the development of corresponding supporting tools, libraries, frameworks and programming methods also greatly improves programming efficiency.

5.2. Poor portability of the algorithm
The accelerators provided by different companies all adopt their own unique processor architecture and have their own execution instructions and compilers. Thus they do not have a unified architecture. Moreover, porting parallel programs to heterogeneous processors requires not only recompilation, but also code rewriting in more cases. The difficulty of writing heterogeneous parallel programs is obviously greater than that of traditional parallel programs, and it lacks the efficient execution means of a hybrid platform with multiple acceleration devices. Therefore, a good software framework for heterogeneous
parallel programming needs to enable programmers to make full use of rich heterogeneous resources while programming without having to consider complex hardware details.

5.3. Complex data access
There are multi-level storage structures such as host memory, host cache, device main memory, device cache, and registers in a heterogeneous hybrid computer system, resulting in a variety of data access methods. Data often needs to be moved on multiple types of storage, and the program may also access data in multiple types of storage at the same time during execution. In a parallel program, multiple threads access the same memory with data sharing and access conflicts, access to different memories has data transmission and communication problems, and the access methods and delays of different memories are not the same. In order to solve this problem, researchers [17-18] completely encapsulated the hybrid heterogeneous system into one chip. For example, APU combines CPU and GPU on the same chip to reduce the overhead of data transmission.

5.4. Unbalanced resource load
Due to the gap in the computing power of heterogeneous processors and their respective computing characteristics, it is difficult to accurately divide the tasks and easily lead to unbalanced loads among heterogeneous processors. In addition, the architecture and parallel mode of heterogeneous processors are also different, which makes it difficult for a task division strategy to adapt to the requirements of heterogeneous processors. In order to solve this problem, researchers [10-12] proposed a model that can evaluate the performance of heterogeneous multi-core systems so that the computing tasks can be reasonably divided.

6. The future of heterogeneous computing
With the Internet of Everything, the amount of data has exploded. The emergence of cameras has made data acquisition sources diversified, coupled with the various sensors in the daily use of mobile phones, making the amount of data even larger. Deep learning in the current popular AI requires all kinds of data for machine learning to make AI smarter. Similarly, the data in the artificial intelligence model is called source data, and the type and quantity it needs basically come from the first edge devices close to the data product, such as driverless data. This type of data needs to be processed quickly on the edge.

One of the benefits of heterogeneous computing is that it can quickly provide cost-effective solutions for a variety of different applications in the future. For example, the high-speed rail is a good example, because the high-speed rail actually needs to do a lot of fast calculations, and at the same time, many devices along the line also need calculations. It is a huge system in itself with many different calculation loads.

In the era of the Internet of Everything, IoT devices have sprung up like mushrooms after a rain, and multiple devices have different versatility requirements and unique requirements for performance and data. In this case, heterogeneous computing is required, which can be achieved by using different chip combinations or even different processes. However, there are many applications in daily life that require scientists to quickly give some solutions instead of redefining some chip architectures. If you want to solve this problem quickly and easily, heterogeneous computing is necessary. It can miniaturize the solution and achieve a good balance between cost and efficiency.

But the biggest problem of heterogeneous computing at present is whether it can meet the inclusive programming or development model, otherwise it cannot be called heterogeneous, but can only be an embedded system or a dedicated SoC, so a unified development environment and programming platform will determine Whether heterogeneous computing technology can break through and become a more universal application technology.

If the versatility of heterogeneous computing is improved, it will further promote applications in the field of high-performance computing represented by general computing and data centers. This is a more feasible technical route to increase computing power before major breakthroughs in the basic theories of computer systems or semiconductor physical devices cannot be realized.
In the future, in the field of heterogeneous computing, more talents are needed to invest in innovation, so that as the system becomes more and more complex, there are ways to simplify the work of developers. Moreover, the future heterogeneous computing will be a major driving force for diversified innovation, and it will also be the primary framework for future innovation, which will help open the door to a new intelligent world. Future devices will have intelligent autonomous capabilities, such as smart robots, smart cars, etc., which are characterized by perception capabilities, in addition to decision-making and actions. In the future, heterogeneous computing will surely be able to quickly solve practical application problems. Through some technological processes such as 3D packaging, such devices with different computing capabilities will be made smaller in size and lower in power consumption, and will eventually become an important carrier for future smart devices. But in the future, it is difficult for heterogeneous computing to explode suddenly, but it will slowly penetrate into various applications, and then change the form of some applications, and even the underlying logic. Solving the problem of heterogeneous acceleration will inevitably involve the use of the most suitable heterogeneous computing architecture for a specific type of operation. In this case, some new technical routes or solutions will be produced. For example, similar to the requirements of short videos and micro-servers required for the Internet of Everything, the requirements for low latency, low power consumption, and high computing power will produce a heterogeneous computing adaptation process. Therefore, in terms of technological penetration, heterogeneous computing will change the shape of the industry, create some new job opportunities and fields, and will also promote the growth of some entrepreneurial companies and underlying technology solution providers.

Whether in terms of economic benefits or time costs, heterogeneous computing will certainly be able to play a huge role in the future.

7. Conclusion
The development of heterogeneous computing is of great significance. Heterogeneous processors have the advantages of fast processing speed, low power consumption, and small size. By trying to match applications with various architecture systems, an optimized balance of performance, power consumption, and energy efficiency can be achieved. For example, the current smart robots and smart driving cars all use CPU + AI processors. Heterogeneous computing technology will change the shape of the industry, bring new fields and multiple types of jobs, and will also give birth to the growth of some entrepreneurial companies and underlying technology solution providers. The various variables and possibilities in the process of industrial change will be far more exciting than the technology itself.

References
[1] Stromme, A., Carlson, R., Newhall, T. (2012). Chestnut: A GPU programming language for non-experts. In: Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores. PP: 156-167.
[2] Auerbough, J., Bacon, D. F., Cheng, P., et al. (2010). Lime: a Java-compatible and synthesizable language for heterogeneous architectures. In: Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications. PP: 89-108.
[3] Linderman, M. D., Collins, J. D., Wang, H, et al. (2008). Merge: A programming model for heterogeneous multi-core systems. ACM SIGOPS Operating Systems Review, 42(2): 287-296.
[4] Yang, W., Li, K., & Li, K. (2017). A hybrid computing method of SpMV on CPU-GPU heterogeneous computing systems. Journal of Parallel and Distributed Computing, 104(6): 49-60.
[5] Hosseinabady, M., Ninez, Y. J. (2020). Sparse Matrix-Dense Matrix Multiplication on Heterogeneous CPU+ FPGA Embedded System. In: Proceedings of the 11th Workshop on Parallel Programming and Run-Time Management Techniques for Manycore Architectures. PP: 1-6.
[6] Kobayashi, R., Fujita, N., Yamaguchi, Y., et al. (2019). GPU-FP-GA Heterogeneous Computing
with OpenCL-Enabled DirectMemory Access. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 489-498.

[7] Li, A., Song, S., Chen, J., et al. (2019). Evaluating Modern GPU Inter-connect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect. IEEE Transactions on Parallel and Distributed Systems, 31(1): 94-110.

[8] Shui, C., Yu, X., Yan, Y., et al. (2020). Revisiting linpack algorithm on large-scale CPU-GPU heterogeneous systems. In: Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. PP: 411-412.

[9] Zhang, J., Jung, M. (2017). An in-depth performance analysis of many-integrated core for communication efficient heterogeneous computing. In: IFIP International Conference on Network and Parallel Computing. Cham: Springer, PP: 155-159.

[10] Goddeke, D., Wobker, H., Strzodka, R, et al. (2009). Co-processor acceleration of an unmodified parallel solid mechanics code with FEASTGPU. International Journal of Computational Science and Engineering, 4(4): 254-269.

[11] Kalidas, R., Daga, M., Keommydas, K., et al. (2015). On the Performance, Energy, and Power of Data-Access Methods in Heterogeneous Computing Systems. In: IEEE International Parallel & Distributed Processing Symposium Workshop. IEEE.

[12] Zhong, Z., Rychkov, V., & Lastovetsky, A. (2012). Data partitioning on heterogeneous multicore and multi-GPU systems using functional performance models of data-parallel applications. In: 2012 IEEE International Conference on Cluster Computing. PP: 191-199.

[13] Yang, W., Li, K., & Li, K. (2017). A hybrid computing method of SpMV on CPU-GPU heterogeneous computing systems. Journal of Parallel and Distributed Computing, 104(6): 49-60.