Peertechz

ENGINEERING GROUP

TRENDS IN Computer Science and Information Technology

OPEN ACCESS JOURNAL

ISSN: 2641-3086 DOI: https://dx.doi.org/10.17352/tcsit

Review Article

Current trends in heterogeneous systems: A review

Gajendra Sharma* and Prashant Poudel

School of Engineering, Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Kavre, Nepal

Received: 16 November, 2022
Accepted: 23 November, 2022
Published: 24 November, 2022

***Corresponding author:** Gajendra Sharma, School of Engineering, Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Kavre, Nepal, Tel: 9840093887; E-mail: gajendra.sharma@ku.edu.np

Keywords: Heterogeneous system; CPU-GPU Architecture; ARM big. LITTLE

Abstract

Looking at the heterogeneous system, this paper touches on the architecture of heterogeneous systems, program models, and some challenges in this field. Heterogeneous systems have become an important development trend in the current high-performance computing field. Heterogeneous systems can be seen from smartwatches, mobile phones and laptops to server systems. This paper will explore different types of heterogeneous systems such as CPU-GPU architecture, and ARM big. Little architecture. The program model for different types of heterogeneous systems where we will see how different architecture heterogeneous systems work. Then we will see some of the challenges in the heterogeneous system and finally some recommendations for future work.

Introduction

To improve the performance of existing processors, the industry has gradually shifted to the heterogeneous platform with multiple cores of CPUs and CPU-GPU combinations. There are GUP-GPU combinations in order to accelerate different demanding tasks [1]. And the future of heterogeneous systems seems to be moving towards the System on Chip (SoC) designs where all different types of processors like Graphical Processing Units (GPU), Central Processing Units (CPU), Neural Processing Units (NPU), etc. are being built in a single chip and they tend to have shared memory. This kind of heterogeneous architecture can be found in small smartwatches to laptop and desktop computers as of 2022. Big companies like Apple have started to build such heterogeneous systems. And for more power-hungry tasks there is still use of CPU-CPU, CPU-GPU hybrid heterogeneous systems are still in use. This trend shows that new heterogeneous systems are becoming more and more important from regular computing to high-performance computing. After single-core and multi-core, heterogeneous computing is considered the third age by the industry. It will be a key emerging model in the field of high-performance computing, as it will be able to successfully tackle problems such as energy consumption and scalability [2]. With different processing units there comes the problems of task scheduling and software compatibility when the heterogeneous system is being built with different architecture CPUs [3] Figure 1.

Processors in heterogeneous computing systems are becoming increasingly tightly coupled, which means they are connected via the memory bus and hence share cache-coherent memory. Although platform-wide shared memory is a clear trend, cache-coherency has been identified as a major stumbling block to scaling to higher core counts, paving the way for software-programmed coherency [4]. Applications should be able to leverage all available processor resources by smoothly operating across various processors to harness diversity and by mapping data and optimizing sharing to avoid hardware overheads to take advantage of upcoming heterogeneous platforms [5]. For a program model, starting the application on the main processor (CPU) and offloading sections of the application to a specialized accelerator (GPU) that atomically does computation and provides the result is a typical technique used in heterogeneous platforms.

In this paper, we will focus on different types of heterogeneous systems. This paper will discuss some of the problems in existing heterogeneous platforms and what could be the way forward to a heterogeneous system.

Heterogeneous system architecture

The current heterogeneous systems mainly have the following architectures: Cell/B.E., CPU+GPU, APU, CPU+MIC, and CPU+FPGA. Among them, FPGAs are mainly used in the high-performance requirements of embedded systems, while the heterogeneous system architectures used in the traditional high-performance fields mainly include Cell/B. E. CPU+GPU, APU, and CPU+MIC.

Cell/B. E. (Cell Boardband Engine)

Cell/B. E., often known as the Cell processor, is a multi-core microprocessor microarchitecture that combines a low-performance general-purpose PowerPC core with streamlined co-processing parts that substantially enhance multimedia and vector processing workloads, as well as a variety of other specific computations. The Cell processor is a totally independent heterogeneous processor with a high use cost that is currently only found in high-end servers [6]. The Cell processor's appeal in general high-performance domains is limited, its continued growth is likewise constrained, and its benefits in heterogeneous systems are increasingly diminishing Figure 2.

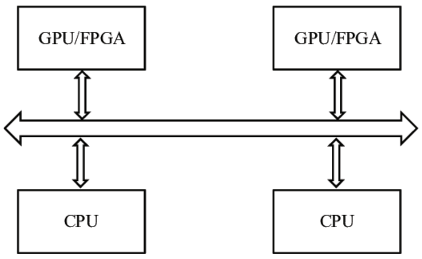

Figure 1: Architecture of CPU-GPU Heterogeneous Platform.

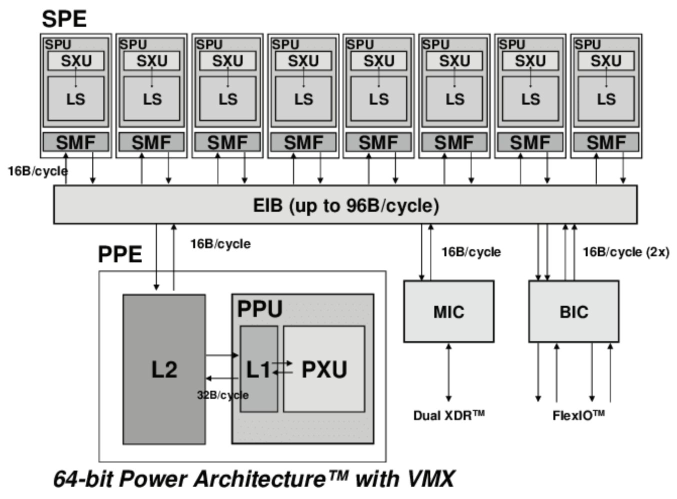

Figure 2: Cell Broadband Engine Architecture based on a heterogeneous chip multiprocessor.

GPU heterogeneous system architecture

The CPU+GPU architecture is mostly used in GPU heterogeneous systems. The primary purpose of the GPU is to provide high-throughput data-parallel computing with a large number of threads, which is ideal for large-scale data-parallel applications requiring high computational density and simple logic branching. The CPU features a complicated logic control unit and a large-capacity cache, has a short data transmission latency, is adaptable to a variety of tasks, and excels at complex logic operations. The CPU+GPU architecture is used to combine the CPU's and GPU's respective advantages, allowing the GPU to process data-intensive parallel tasks while the CPU handles complex logical transaction processing. This allows the CPU and GPU to fully exploit their respective advantages and maximize the utilization of heterogeneous systems. Processing power, lowering computational expenses, and using less energy [7] Figure 3.

APU heterogeneous system architecture

The main feature of an APU (Accelerated Processing Unit) is that it contains all the processing capabilities composed of scalar and vector hardware. Due to the integration of the two computing cores, the main frequency of the processor and the number of processing cores are limited by the chip space, manufacturing process, and heat dissipation. At the same time, the X86 CPU and GPU vector processor in the current APU has not yet achieved seamless integration [8], so its overall computing performance is slightly insufficient compared with the architecture in which the main processor and the coprocessor are separated Figure 4.

MIC heterogeneous system architecture

Intel introduced the MIC (Many Integrated Core) integrated

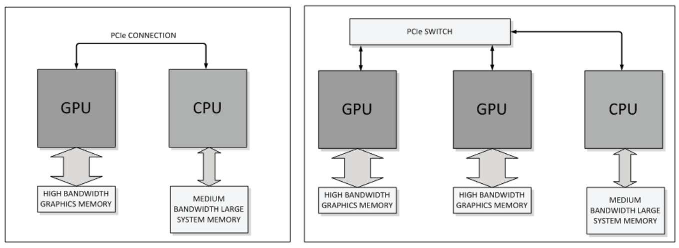
PCIe CONNECTION
GPU
CPU
HIGH BANDWIDTH GRAPHICS MEMORY
MEDIUM BANDWIDTH LARGE SYSTEM MEMORY
PCIe SWITCH
GPU
GPU
CPU
HIGH BANDWIDTH GRAPHICS MEMORY
HIGH BANDWIDTH GRAPHICS MEMORY
MEDIUM BANDWIDTH LARGE SYSTEM MEMORY

Figure 3: CPU+GPU Heterogenous Platform interconnected by PCIe.

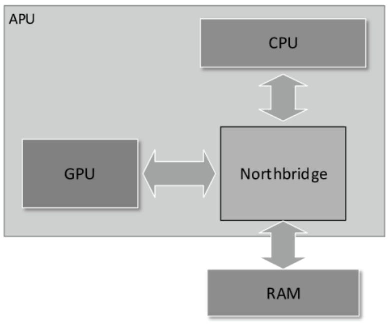

Figure 4:AMD APU Architecture.

many-core coprocessor in 2011. Its vector expands the typical microprocessor and then merges several enlarged cores to increase computational capacity even further. Unlike typical accelerators, the MIC coprocessor has its own independent micro-operating system, making it more akin to a high-performance computing node that can be accessed, programmed, and fully functioning [9] Figure 5.

Arm big. LITTLE

ARM big. LITTLE technology is a two-type processor heterogeneous processing architecture. While ARM CPUs with "conventional" core setups exist, big. LITTLE-based CPU architectures include two "clusters" of cores, each with a distinct design for various workloads. We'll see "high performance" cores built to tackle demanding activities, as well as "power-efficient" cores that do more mundane jobs, in these types of CPUs. High-performance cores are often high-specced and power-hungry, with significantly higher clock rates, whereas power-efficient cores are weaker, lower-clocked, and use significantly less power [10] Figure 6.

Program model

The common practice in heterogeneous platforms (CPU-GPU) is to start the application on the main processor (CPU) and during execution, offload parts of the application to a specific accelerator (e.g., GPU) that atomically executes a computation and returns the result. Offloading requires providing the remote processor with the code to be executed and the data on which it has to operate; in most architectures, this requires a data copy. This computational approach is suitable for machines with little control, such as modern GPUs. For growing heterogeneous platforms, however, forcing the execution flow to return to the source processor at the end of each offloaded function is too rigid [11].

When we have to deal with multiple CPUs with different Instruction Set Architectures (ISA) there is a problem with software compatibility. Processors implementing different ISAs not only have different instructions and register sets but different native data storage formats, defined as the ISA's application binary interface (ABI). For different-ISA processors with shared memory, they must agree on a common data storage convention or data must be converted between formats upon execution migration [3].

In the case of ARM big. In LITTLE architecture there are two clusters of CPU cores with different clock speeds. A high-performance core for raw performance and efficiency-core to save power. A LITTLE core is coupled with each BIG core. At any one moment, only one core in each pair is active, with the inactive core turned off. The pair's active core is chosen based on current load levels. The scheduler is aware of the big and little cores' different performance and energy characteristics. The scheduler monitors the performance requirements of each individual thread and utilizes that data to choose which type of processor to use [12].

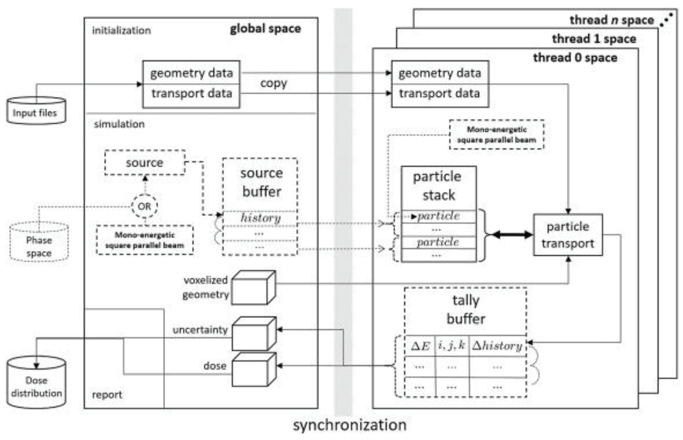

Figure 5: Intel MIC Architecture.

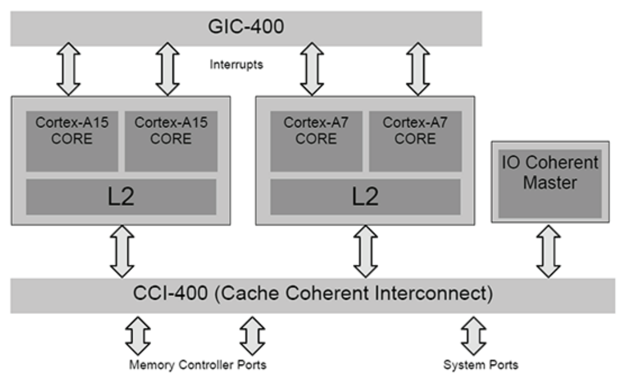

Figure 6: Arm big. LITTLE Architecture.

Challenges

Heterogeneous computing systems provide unique issues that aren't present in traditional homogeneous computing systems. The presence of many processing units increases all of the problems that homogeneous parallel processing systems have, but the amount of heterogeneity in the system might add non-uniformity to system development, programming methods, and overall system capabilities. Here are some challenges of a heterogeneous system.

Instruction-Set Architecture (ISA)/Application Programming Interface (API)

Heterogeneous systems contain multiple computing devices with different system architectures, instruction sets, and programming models thus, heterogeneous systems frequently have different programming models from CPUs. The instruction set architectures of compute elements (Programs) may differ, resulting in binary incompatibility. Compute elements, similarly, may interpret memory in many ways. This includes endianness, calling convention, and memory layout, and is determined by the architecture and compiler used. The solution to this problem is to let the CPU control the computation and finish the calculation with the help of the GPU.

Memory Interface and Hierarchy

Compute elements can have varied cache architectures and cache coherency protocols, and memory access might be uniform or non-uniform (NUMA). Differences in the capacity to read arbitrary data lengths can also be discovered, as some processors/units can only conduct byte-, word-, or burst reads. Different threads in a parallel program access the same memory with data sharing and access conflicts, different memories with data transfer and communication concerns, and distinct memories with different access methods and delays [13].

Interconnect

Aside from fundamental memory/bus interfaces, compute elements may have many types of connection. Dedicated network interfaces, Direct Memory Access (DMA) devices, mailboxes, FIFOs, and scratchpad memories are examples of such devices. Furthermore, while certain parts of a heterogeneous system may be cache-coherent, others may require active software participation to guarantee consistency and coherency. A robust software framework for heterogeneous parallel programming should allow programmers to utilize rich heterogeneous resources to the fullest extent possible while still avoiding having to pay attention to intricate hardware details.

Performance

In a heterogeneous system, CPUs with identical architectures may have underlying micro-architectural variances that result in varying degrees of performance and power consumption. Capability asymmetries combined with opaque programming models and operating system abstractions can sometimes lead to performance predictability issues, particularly with mixed workloads [14]. In order to overcome this issue, researchers have proposed a model to analyze the performance of heterogeneous multi-core systems to fairly divide computing jobs [15].

Data partitioning

While splitting data on homogeneous platforms is frequently straightforward, it has been demonstrated that the problem is NP-Complete in the general heterogeneous case. It has been demonstrated that optimal partitionings that fully balance the load and minimize communication volume exist for modest numbers of partitions. Further research is necessary to obtain optimal partitioning in a heterogeneous platform.

Conclusion and future work

In the field of high-performance computing, heterogeneous systems have emerged as a significant development trend. It is critical to combine the newest multi-core technology and ARM big. LITTLE architecture to make an efficient, and energy-saving new heterogeneous system. The primary issues encountered in the process of employing the coprocessor to speed the application program in the new heterogeneous system are: how to make the main processor and the coprocessor communicate properly? And how to get the most out of the heterogeneous systems by using simplifying programming approaches to boost application computing performance. To address these issues, we must develop simple and effective high-level abstract programming approaches that can adapt to new heterogeneous systems, and support fine-grained and coarse-grained parallelism, portability and scalability, and energy-efficient programming. Investigate a unified programming model capable of enabling simple and efficient application development on heterogeneous systems, so that developed applications can adapt to different hardware architectures and corresponding underlying support software, and, when combined with corresponding compilation, runtime systems, and performance optimization mechanisms, fully exploit the efficient computing power of heterogeneous systems to deal with complex scientific computing problems. A significant study area in structural computing.

The future of heterogeneous systems lies in SoC technology with big. LITTLE architecture. Apple has already started to implement this technology of shared memory architecture in their laptops and desktop computers. Shared memory architecture makes much more sense as it will allow processors to communicate better but for that, there needs to be a better program model in terms of task scheduling, ABI translation (for heterogenous CPU-CPU system), and a better cache coherency model.

References

1. Yang XJ, Liao XK, Lu K. The TianHe-1A supercomputer: its hardware and software. Journal of Computer Science and Technology. 2011; 26(3): 344-351.
2. Brodtkorb AR, Dyken C, Hagen TR, Hjelmervik JM, Storaasli OO. State-of-the-art in Heterogeneous Computing. Scientific Programming. 2010; 1-33.
3. Lyerly R, Antonio B, Christopher J, Vincent L, Anthony C, Binoy R. Operating System Process and Thread Migration in Heterogeneous Platforms. 2016.
4. Baumann A, Barham P, Dagand PE, Harris T, Isaacs R, Peter S, Roscoe T, Schupbach A, Singhania A. The Multikernel: A New OS Architecture for Scalable Multicore Systems. In Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles. SOSP. 2009; 9.
5. Beckmann N, Sanchez D. Jigsaw: Scalable Software-defined Caches. In Proceedings of the 22Nd International Conference on Parallel Architectures and Compilation Techniques, PACT. IEEE. 2013; 213–224.
6. Brodtkorb A, Hagen T, Sætra M. Graphics processing unit (GPU) programming strategies and trends in GPU computing. Journal of Parallel and Distributed Computing. 2013; 73: 4–13.
7. Nvidia Corporation. Compute unified device architecture programming guide [OL]. https://developer.nvidia.com/cuda-zone, 2022
8. Daga M, Aji AM, Feng WC. On the Efficacy of a Fused CPU+GPU Processor (or APU) for Parallel Computing. 2011 Symposium on Application Accelerators in High-Performance Computing. 2011: 141-149.
9. Liu X, Smelyanskiy M, Chow E, Dubey P. Efficient sparse matrix-vector multiplication on x86-based many-core processors. In Proceedings of the 27th international ACM conference on International conference on supercomputing (ICS '13). Association for Computing Machinery, New York, NY, USA, 2013; 273–282.
10. ARM Technologies. https://www.arm.com/technologies/big-little, 2022
11. Barbalace A, Sadini M, Ansary S, Jelesnianski C, Ravichandran A, Kendir C, Murray A, Ravindran B. Popcorn: Bridging the Programmability Gap in heterogeneous-ISA Platforms. In Proceedings of the Tenth European Conference on Computer Systems, EuroSys '15. 2015; 29:1–29.
12. Gottipati S. 2021. Exploring ARM and heterogeneous compute architecture. https://www.druva.com/blog/exploring-arm-and-heterogeneous-compute-architecture/
13. Zhong Z, Rychkov V, Lastovetsky A. Data partitio-ning on heterogeneous multicore and multi-GPU systems using functional performance models of data-parallel applications. In: 2012 IEEE International Conference on Cluster Computing. 2012; 191-199.

14. Kalidas R, Daga M, Keommydas K. On the Per-formance,Energy,and Power of Data-Access Methods in Heter-ogeneous Computing Systems. In: IEEE International Parallel &Distributed Processing Symposium Workshop. IEEE. 2015.

15. Goddeke D, Wobker H, Strzodka R. Co-processor acceleration of an unmodified parallel solid mechanics codewith FEASTGPU. International Journal of Computational Science and Engineering. 2009; 4(4): 254-269