International Journal of Scientific and Engineering Research Volume 3, Issue 11, November 2012
Paper ID: I019034

GP-GPU Accelerated Supercomputing

Aditya R. Jhaveri *Pratik R. Joshi*

***Abstract*--** The latest trend introduced in Supercomputing processor technology is the introduction of General Purpose Graphical processing Unit or GP-GPU. These new variety of non-standard processors ever since their introduction in the late 80's have undergone tremendous change in their internal architecture, memory interface, Hardware Abstraction(HAL) and feature set. This new technology grew rapidly with the emerging gaming market and better interconnection standards and soon surpassed the processing strength of general purpose x86 processors with the introduction of DirectX 9 Shader technology. By mid 2000s this power was realized to be harness able with the introduction DirectX 11 API and compatible shader hardware. These cheap consumer level processors were capable of generating a combined processing power in excess of 1 Teraflop in 2 007, which reached to 4.7 Teraflops by 2012. This power was achieved within the ordinary computers, with power requirements of less than 800 watts and costs under $800. Soon GPU manufacturers released their own SDKs to allow programmers to make use of this computation power within their own applications, two mainly notable being the AMD-Stream APP SDK and the NVidia CUDA SDK[3].

***Index terms*—Abstraction, Shaders, Super-computing, Parallel Computing, VLIW4, VLIW5, GCN (Graphics Core Next), CUDA**

I. Introduction

A. Makeup of a GPU

A GPU is a small processor that functionally lies between the PCI bus and the external display. It can access the system memory via the PCI-Main Bus Bridge. It also has its own memory cache and a cluster of ALU's and a DAC (Digital to Analog Converter) assigned for each display that it can connect to. Irrespective of the System memory type, it can have its own individual memory of the following types- DDR2, DDR3, GDDR4 (AMD only) and GDDR5.Most of the newer GPUs have a very wide FPU of 128bits range [4]. The GPU always runs as a slave processor, fetching its instructions from the system instruction cache under the control of the OS-Driver and the Kernel. It cannot run the binary instructions from any thread directly as it is not an x86 CPU.A fixed pipeline GPU can only work as a display controller because of the absence of a proper HAL as well as the primitiveness of the design.

B. Functioning of a nonlinear pipeline (Shader)

Post the introduction of DirectX8, game designers toyed with the idea of manipulating the pipeline at any stage to produce special effects without the additional overhead. This meant an overhaul in the design. As thepipeline was no longer fixed, it meant that the function of any ROP was to be flexible enough to perform more complex mathematical calculations at any point of time .This brought about the SHADER design technology. This design was a huge contrast from the previous architecture.

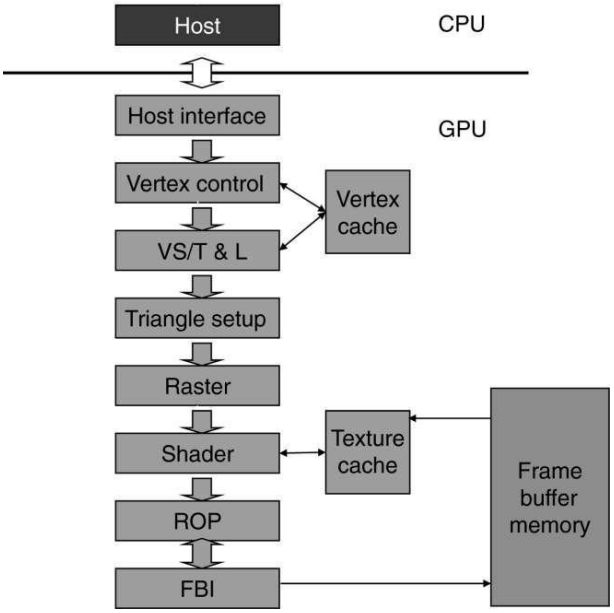

Fig 1:
The Traditional Graphics pipeline

Fixed pipeline GPU: Different parts of the chip perform specific tasks in creating the final image. These parts (ROPs) could not do anything else than what they were designed for. Its power is dependent on the speed of the ROP (Raster Operator).

Shader GPU: The GPU is divided into Shaders that can perform any task in the image creation stage, with the flexibility to manipulate the result. These Shaders were just a little more

than discrete ALU cores. Its power came from the number of these Shaders that work in tandem.

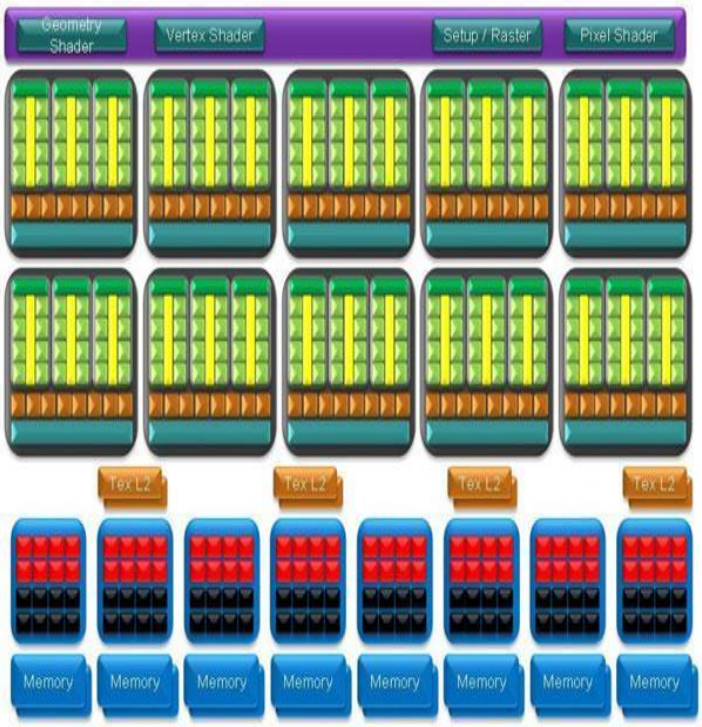

Fig 2: Shader GPU architecture

II – Benefits of GPU in supercomputing

A. Differences between a GPU and an x86 CPU

The key differences in the CPU and GPU architecture are that a CPU is a true processing package: It has its own ability to fetch the instructions from the memory, arbitrate and lock the main bus; reset the system is the heart of processing the OS and the Kernel. It has a very large instruction set. It is optimized for long instruction chains as it can execute instructions out of order. It also supports all four memory access-modes.

The GPU, on the other hand is a large cluster of ALU's which need to fetch their instructions from the main dispatcher(CPU-Memory-Main Bus).Each ALU is a very primitive CPU which can process only a small set of mathematical calculations and access the main memory only in a limited number of ways. However, it has its own memory connected via a very high-bandwidth and wide bit-size bus. It does not have an out-of-order execution optimization, but instead has the ability to break down the operation and execute multiple instructions parallel on thousands of cores together.

Comparison between CPU and a GPGPU

Ordinary x86 CPU	Shader capable GP-GPU
1) CISC processor, Bus Master and arbitrator. System essential.	1) RISC processor, Bus Peripheral, system optional.
2)Small number of very powerful and complex cores(1-16) running at 1.4-3.8 GHz (Peak 0.112 Teraflops for Core-i7)	2)Vast number of simple and weak cores(80-3600) running at 600Mhz to 1.1GHZ (Peak: 4Teraflops on AMD7990 stock[4])
3) Memory bandwidth is moderate but complex memory access is possible (Ordinarily	3) Memory bandwidth is very high but memory access is simple for system memory. (Ordinarily between 200-
4) Optimized for less number of threads, but complicated thread	4) Optimized for Large number of operations (not exactly threads) but

B. When is a GPU preferred over a GPGPU?

As the comparison chart above shows, the GPU is mostly suitable for a large number of Mathematical calculations in parallel, but less effective if parallelism is not required. This is because even though the computation capacity of the GPU is very vast; it needs all of its cores to be engaged. Each individual core of the GPU is still much weaker than those of the x86 CPU. It also is bottlenecked if it constantly needs to communicate to the main system via the same bus. The best approach is to load all the individual instructions into the GPU memory and/or the data variables. The best usage scenario is when using the GPU in particle simulation where a large number of data-elements are needed to be processed in parallel [1]. These GPU's are also used in medical research (protein folding), Ray Tracing (CGI) and space research. It is less preferred in areas where very few elements are processed in parallel or threads rely on special instructions that are difficult to simulate on the GPU [1]. This processor is also used in small-scale researches, where money, space and power are to be conserved.

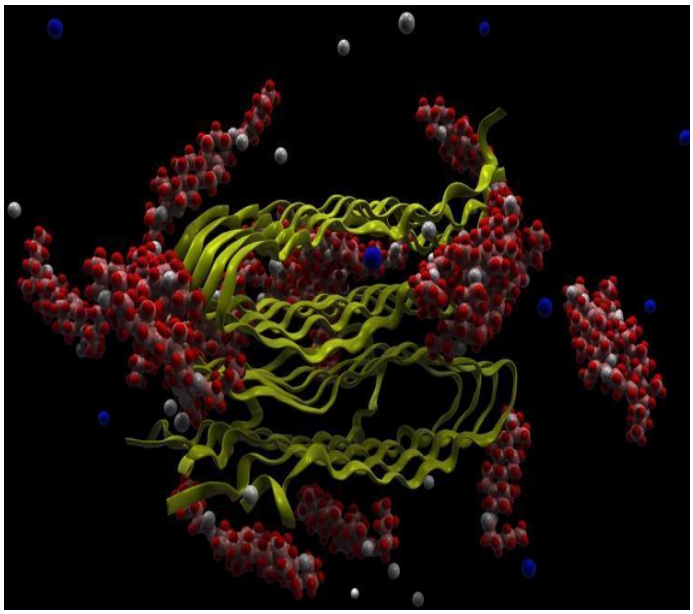

Fig 3: A protein simulation preview from the "Folding@Home" open research project [6]

III. Changes in the current architecture using GPGPU

A. When is GPU preferred over CPU?

In smaller and standard systems, integration of GPU into the system is generally simple since they are ordinary consumer peripherals and use simple PCIe bus for interconnection with the main system. The challenge is that GPGPU requires the use of graphics programming languages like OpenGL and Cg to program the GPU. Developers had to make their scientific applications look like graphics applications and map them into problems that drew triangles and polygons. This limited the accessibility to the tremendous performance of GPUs for science. This was true for fixed- pipeline GPUs that had no programmable flexibility. Now it uses a specialized software set of Drivers, HAL and kernels to be used as a GPGPU. However larger performance conscious researchers that integrate hundreds or even thousands of GPUs in clusters need to use special hardware for interconnection because the traditional PCIe has specific data channels called 'Lanes'. These lanes are limited and fixed for ordinary consumer electronic boards, and the PCIe bus arbitrator communicates by allocating a number of them to each device it services. Each lane has a transfer rate of 250mb/s excluding the overhead. This speed is very less compared to the internal GDDR5 transfer rate. This is overcome by reducing the number of devices which increases the lanes assigned to each device. However this is counter- productive since lesser GPUs mean lesser parallelism. Larger non-standard Bus boards are used to put all the GPUs, Memory DIMMs and processor(s) on the same PCB and connect them via non-standard busses of double or quad width and transfer rates. Ordinary arbitrators are simply not powerful enough to accommodate the bandwidth between the GPU-Memory-CPU. These GPUs are also connected within themselves via bridge wires to share the individual GPU memory within them. The board is divided into two parts: NORTH-BRIDGE- Connects the CPU, memory and the High speed bus arbitrator(s) SOUTH-BRIDGE. This entire section interfaces the GPU set to the main bus arbitrator in the NORTHBRIDGE.
Sometimes the instruction level optimization is introduced by use of additional chipsets such as LUCID Hydra.

B. Nonstandard Peripheral boards and SOCs

On the hardware side of integration of GPUs into the HPC, use of standard PCIe bus is often avoided in favor of a system of externally connected bus interface that forms a high speed channel between the processor and the Southbridge with its own power supply and cooling solution that is specific to each module. These modules have also their own memory. The only thing that they miss is storage devices and a program instruction dispatcher. The main CPU loads the programs as well performs task relating to human interface. Once the programs are taken from the storage or the interface, it gets converted into a problem solvable by the GPU by the use of OpenCL programming language and the OpenCL kernel [2]. This language is executable on both the CPU as well as GPU once their respective kernels are loaded. The software selects the target processor and the drivers perform re-compilation of the program and then transferred to the GPU via the PCIe bus substrate. With the correct hardware in place, the operation of GPUs was largely limited by the lack of proper HAL. Initially Graphics card manufacturers concentrated on creating more and more powerful cards without realizing its potential as a General processor. They did not create means – drivers, compilers, assemblers – to effectively use this processor outside of the DirectX and OpenGL framework. These HAL tools were needed for executing non graphical instructions on graphic processors. These tools once in place run just like any other driver in the background, enabling selected instructions to run on the GPU. Once these are run, the results have to be fetched back to the main program memory in the NORTHBRIDGE from the GPU memory. The implementation of correct HAL drivers and standardized programming languages has made this quite easier and sensible for the programmer.
A SOC is a simplified, albeit weaker solution to the integration problem. Here the GPU and CPU are built on the same die, while sharing a lot of components in common. While they do share the same memory - in terms of hardware – they do not share the memory as far as the programs are concerned. The kernel segregates the memory domains to avoid conflicts, and this often leads to bottlenecks [3]. Also sharing the same die area renders them weaker and reduces the TDP as the die area for heat dissipation is reduced and cooling becomes a problem. The SOC in supercomputing is still in its infancy and not powerful enough for most practical supercomputing needs. The only example of SOC are the processor line of AMD-Bulldozer and Piledriver series [5] under the label of APU(Accelerated Processing Units) which are a set of 4 x86 cores, 400 GPU cores, DDR3 memory interface controller on the same chip. These have shown promising performance for future mobile-level supercomputing for power saving devices like Laptops and Tablets.

C. BIOS optimizations and custom Linux OS kernels

The Basic Input Output System (BIOS) of the graphic cards here are usually molded to add additional mathematical and memory access functions useful only for HPC fields and remove unnecessary constraints related to graphic card standards. These are usually done by the manufacturer, that too only when it intends to sell a model as GPGPU for HPC and not intended for market commodity graphics card. This also means that the memory interface is widened: - in an instance, NVidia widened its memory interface to 512 bits, the widest known so far. Also the display ports are removed as the card is no longer capable of servicing those ports.

On the other hand, the operating systems could be suitably optimized to be able to take advantage of additional computational (and memory) capabilities. This is possible on Linux operating system, since it uses monolithic kernels with individually loadable modules that are open source and hence modifiable [3]. They compile the programs on the go for the processor, giving the user to select the processor (x86 or GPU) on the fly. The hardware present is automatically considered while compiling the final program and loaded into the nearest memory of the selected processor.

It must be noted that even though OpenCL [2] provides a common language to program both GPUs and x86 CPUs, the actual implementation of these instructions is independent of the platform and unique to the hardware. Hence, having a unified platform at the OS or kernel level is necessary. This makes sense in case of Linux too, since nearly no two versions of Linux are the same and yet they are inter-compatible in program framework, this is also another reason why most of the supercomputers.

IV. References

[1] Zhe Fan, Arie Kaufman, Suzanne Yoakum-Stover. *GPU Cluster for HPC, 2004.*

[2] P.B. Sunil Kumar. *Introduction to Parallel Computing, Department of Physics IIT Madras, Chennai, 2010*

[3] Felipe.A.Cruz. *Tutorial on GPU Computing, University of Bristol, Bristol, United Kingdom.*

[4] www.anandtech.com

[5] www.techpowerup.com

[6] folding.stanford.edu