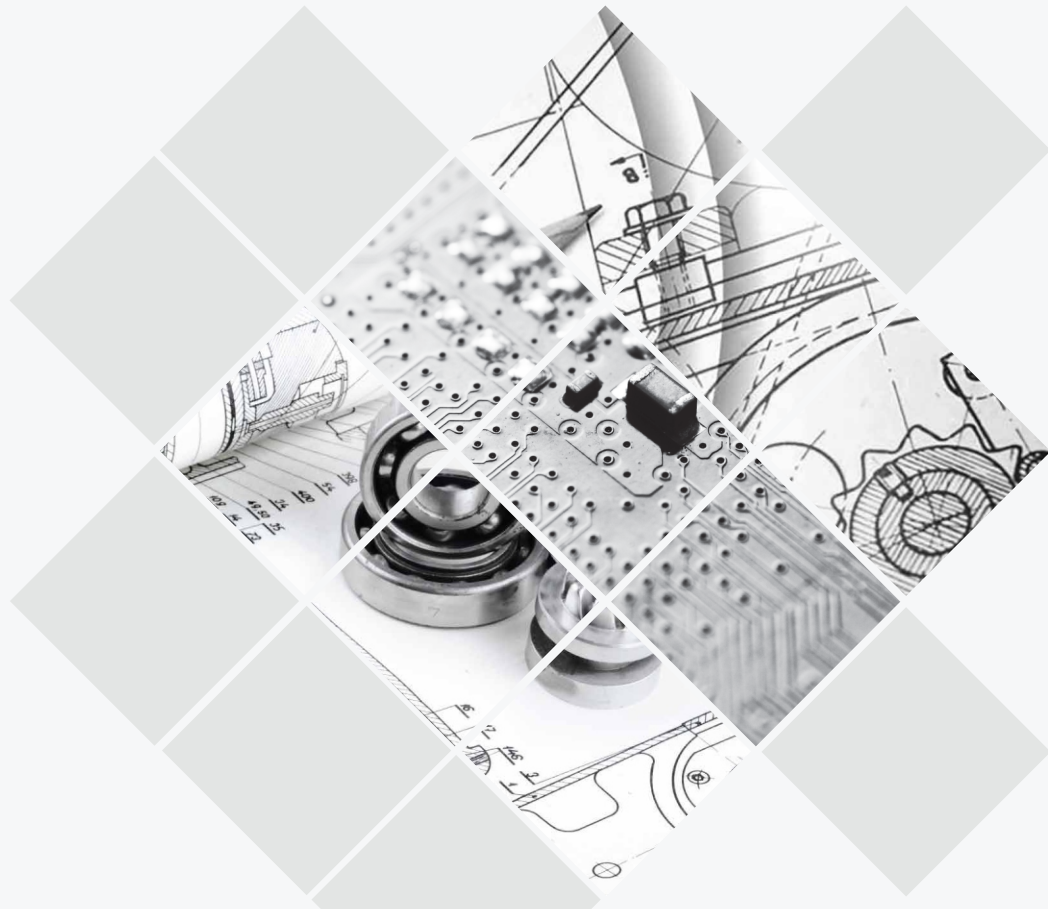# HPC platform options: Cell BE and GPU

As data processing requirements increased with new applications, new processing technologies like Stream computing and parallel execution came into being.

Anoop Thomas
QuEST Global

contents
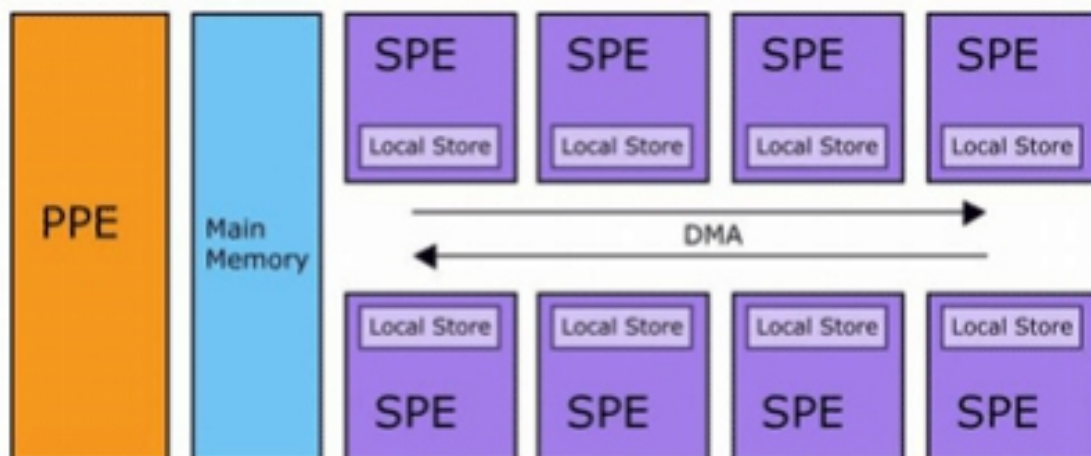
HPC Platform options: Cell BE and GPU

Overview

As data processing requirements increased with new applications, new processing technologies like Stream computing and parallel execution came into being. This write-up briefly compares two competing performance architectures for data parallelism – Cell Broadband Engine (Cell BE™ in short) and the GPU (Graphics Processing Unit).

Cell Broadband Engine (Cell BE)

Evolution and Roadmap

The Cell BE™ Processor architecture was developed in collaboration between IBM, Sony and Toshiba. Development started in 2001 and first set of products based on this architecture started appearing in 2005. The Cell processor architecture primarily consisted of one PowerPC based core called "PowerPC Processing Element" (PPE in short) and 8 "Synergistic Processing Element" (SPE in short) cores. The PowerPC core is the core on which the OS is run, and the SPEs were dedicated for data processing, with special instructions for numerical/mathematical operations. SPEs lack General purpose instructions or direct access to main memory, because of this, OS cannot be run on this core.

The Cell processor connects to fast XDR RAM memory to which only the PPE has direct access. There would be special programs for both PPE and SPE cores, since the instructions for these are different. The scheduling of programs on the various SPE in the CPU is managed by the OS or the PPE program running currently. SPEs can load/store data from main memory using fast DMA. Each SPE has 256Kbytes of static local RAM associated with it. Any data needs to be fetched into this memory for processing. After processing, the resultant data may be transferred back to main memory using DMA.

HPC Platform options: Cell BE and GPU

The Cell CPU is designed to operate at up to 4GHz clock speeds. It is manufactured using the 90nm SOI process; more recent variants use the 65nm fabrication technology. The raw compute power of Cell CPU is rated at 260 Gigaflops (Single Precision). The architecture is scalable to increase/reduce the count of the PPEs or SPEs in the processor die to meet specific computing needs as required.

The first variant of the Cell processor was used in the Sony PlayStation™ 3 Gaming console, as the main CPU. This variant consists of one PPE and 6 SPEs with a clock speed of 3.2GHz. One SPE was disabled and another reserved for system tasks by the embedded OS. A more powerful variant of the CPU was used widely in the IBM QS blade server series. Toshiba uses a custom variant of the Cell CPU with only 4 SPEs as a coprocessor in their range of laptop computers, for enhanced graphics processing.There are many more applications using the Cell CPU, as add-in

boards for data processing etc.
However, the main use of the architecture was in the Sony PS3 game station and IBM blade servers. IBM introduced a more powerful version of the Cell CPU in 2008, with 10 times double precision compute power (102 Gigaflops) than before, targeted at scientific, data intensive applications. This variant was
called the PowerXCell 8i™.

The Cell CPUs are also used in some powerful supercomputers of the time, including IBM Road Runner and in many new mainframe computers from IBM. Certain consumer electronic products are also in the market,using Cell processor. Most notable among them is the Cell TV from Toshiba, which uses Cell processor to accelerate the HD decoding and preview feature in its HDTV models.

There are many Linux distributions that support the Cell CPU, with full range of applications that take advantage of the Cell processor's computing power. This includes Red Hat Fedora, Yellow Dog Linux etc. Since the Cell

is based on PowerPC core, most PowerPC based Linux distributions run on Cell systems.

In November 2009, IBM stated that it has discontinued development of the Cell processor model with 32 SPE cores. But IBM also stated that the cell BE architecture will be used in the next generations of PowerPC based CPUs from IBM

Cell Development

Development for the Cell platform could be done using multitude of tools and SDKs released by different vendors, including IBM, Sony, Mercury and others. Primary OS running on Cell systems would be Linux, so most of the tools and compilers available for Cell development are based on Linux/gcc tool chain.

IBM has released the Cell SDK for developing applications that take advantage of the Cell Be™ architecture. The IBM xlc compilers and associated tools help in profiling and debugging Cell based applications under Linux. The SDK supports Eclipse as an IDE for Cell development.

Sony has its own proprietary SDK for developing games targeting the PlayStation™ 3. The Ps3 also supports loading Linux onto it, after which the tools from IBM can be used for developing Linux applications targeting the cell processor. Mercury is the key development kit supplier for Cell based systems which provides their own SDKs, tools and analyzers for developers.systems which provides their own SDKs, tools and analyzers for developers. Add-in

boards with Cell CPU based development kits can be procured from Mercury.

Most Linux distributions running on the Cell CPU do not drive the SPE units. The program that wants to use the SPE will have to schedule the SPE kernels/programs for execution on the available SPE units in the system, and move the data efficiently to achieve maximum processing throughput. In effect a scheduler has to be implemented by the application developer in an

HPC Platform options: Cell BE and GPU

efficient manner (in the PPE program) to achieve the maximum output. This means optimum loading of the SPE cores, and associated memory and local data registers. Since this is an additional overhead and more difficult to tune for performance compared to more conventional type of development on other type of systems (like x86), Cell based application development is considered to be difficult and cumbersome.

Graphics Processing Unit (GPU)

Roadmap and Future

In the early 80s and 90s, the PC subsystem used display coprocessors for the sole purpose of displaying graphics/text on the video unit. The functionality of these units were limited to the acceleration of 2D graphics including raster operations such as drawing shapes, rendering text etc. on the screen. With the popularity of PC gaming increasing in 2000, more and more PC based 3D Games started appearing in the market, which was demanding in 3D graphics processing. Display board/chip vendors started adding 3D processing capability to the graphics chips to accelerate 3D games and some professional 3D applications such as AutoCAD™ and 3D studio MAX™. OpenGL was primarily used for accelerating 3D graphics; later on DirectX™ was introduced by Microsoft.

Prior to 2001, with OpenGL 1.5 and DirectX™ 7, display chips had only fixed graphic functions implemented inside it. With DirectX™ 8 and OpenGL 2.0 specifications, vertex and pixel shaders were introduced, which made 3D graphics more flexible by making the 3D rendering process programmable.Developers could visualize effects such as shadows and custom lighting etc. The first 3D graphics accelerators supporting shaders were the real GPUs. These chips could execute the shaders in parallel on multiple pixels/vertices to be rendered on screen. As the games and 3D applications got more demanding, the GPU vendors (nVIDIA and ATI were the most prominent at the time) increased the processing

capability of their GPUs accordingly.

Today's GPUs exceed the typical CPU in raw processing power. They also provide better performance/watt and performance/cost. These achievements were possible by new fabrication technology, enabling more and more transistors to be packed into smaller chip dies. Although targeted at 3D gaming in general, the processing power of GPUs today attracted many into running general purpose computations on GPUs to achieve better throughput. This paradigm is now known as 'GPGPU'.Earlier the GPGPU developer could use only 3D graphics APIs such as OpenGL and DirectX™ to program them. Today GPGPU frameworks such as nVIDIA CUDA™ and OpenCL are available to make GPU computing/programming easy.
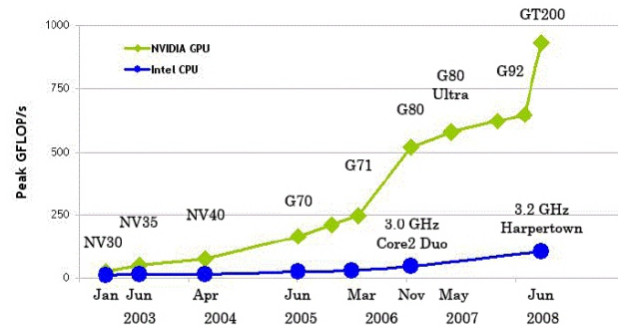
The major GPU vendors in the PC industry are nVIDIA and ATI. Both have their own GPU offerings for consumer, professional and Industrial graphics markets. They also provide developer frameworks for GPGPU computing that leverage the computing power of their GPUs. nVIDIA has released the CUDA™ SDK and toolkit which supports their GeForce™/Quadro™/Tesla™ Range of GPUs and ATI has their Stream SDK targeting ATI GPUs. Common standards for Stream computing such as OpenCL have evolved during the meantime, which is supported by both these GPU vendors. Microsoft has introduced DirectCompute API, which takes advantage of the massively parallel GPUs for compute intensive applications. The current GPU models offer up to 2+ Teraflops of raw computing power.

HPC Platform options: Cell BE and GPU

Current GPU models from ATI include their RV870 based Radeon™/FireGL™/FireStream™ GPUs that support OpenCL and DirectCompute APIs. Programming can be done by using the SDK released by ATI. The current series of GPUs from nVIDIA are the GT200 architecture based derivatives, in GeForce, Quadro and Tesla product ranges targeted at the consumer, professional and Industrial market segments respectively. These GPUs are built on CUDA™ technology for GPGPU computing. nVIDIA offers the C for CUDA™ language for taking advantage of the computing power of these GPUs. CUDA™ SDK and toolkit are most popular among GPGPU programmers worldwide.

nVIDIA recently unveiled the latest CUDA™ architecture codenamed 'Fermi'. This new architecture based products are targeted at GPU computing. Fermi based GPUs will start shipping by late Q1 2010, according to nVIDIA. Fermi based GPUs are targeted at HPC applications, with features like
ECC memory support and on-device debugging.

To complement the new Fermi GPUs, nVIDIA is developing Nexus – a debugging tool that can debug code inside the actual device, which was not possible earlier. New CUDA™ SDK with updated features and support for Fermi based GPUs is under
development and will be released alongside the new GPU variants.

GPU based Development

There are many options for developing GPU based applications. The oldest method is to use Graphics APIs such as DirectX™ and OpenGL. Graphics programmers will find this easy, as they are more adept to this. This causes the need for learning 3D APIs to implement GPGPU applications. Then there are GPGPU frameworks such as ATI Stream, nVIDIA CUDA™. ATI stream was discontinued and ATI later adopted OpenCL for programming their GPUs.

nVIDIA released CUDA™ SDK/Toolkit along with their GeForce™ 8800 (G80) architecture in November 2006. The C for CUDA™ language is a high level language similar to C, and is easily learned by most developers. nVIDIA has been updating their CUDA™ SDK and tools to support recent GPU models, with enhanced features and compute power (G92, GT200 based GPUs). CUDA™ SDK and toolkit supports both Windows™ and Linux (Major desktop distributions such as Red Hat, SUSE, Ubuntu, FreeBSD etc).

Comparison

• Max throughput: The GPU wins here since it offers at least 2-3 times performance than a Cell CPU. This is reflected in real world tests/applications too.

• Performance per watt: The GPU scores here over the Cell CPU. Even though the total power consumption of a GPU is much more than that of the cell, the FLOPS offered is much more times than the cell.

• Performance-cost ratio: The GPU outscores the Cell in most scenarios. The cheapest Cell system available is the Sony PlayStation™ 3 (Cell CPU with 6 SPEs, clocked at 3.2GHz) which will cost $400, and a typical GPU like the nVIDIA GeForce™ GTX260 will cost $200 (~900 Gigaflops).

• Developer friendliness: It's a tie. Earlier (before CUDA™), when GPU frameworks were not available, OpenGL or DirectX™ were the only options for GPU computing, and programmers had to learn Graphics to do GPGPU. In the case of Cell, the hardware architecture makes the parallelization of existing serial code difficult. Add to that the developer has to take care of the SPE scheduling and data transfer activities, in an optimized fashion to achieve best performance of the cell, the FLOPS offered is much more times than the cell.

• Cost factor: GPU wins over the cell here. As mentioned above, the cheapest cell system is

HPC Platform options: Cell BE and GPU

the PS3 ($400), then the Mercury development board ($8000) and then an IBM blade PC ($10,000). A GeForce™ GTX 260, providing 900 Gigaflops of computing power is available for $200.

• Data transfer latency: Since the GPU is available as an Add-on card which attaches to

the PC using the PCI-Express interface; the data

transfer speeds are not as fast like a CPU accessing the RAM. The speed is limited to that of the PCI-Express interface. In the case of Cell,

the PPE has direct access to the XDR RAM, and the SPEs use fast DMA for data transfer, which is similar to the PPE memory access speeds. Hence the data transfer latency is near-zero.

The Cell outscores the GPU here.
• Developer tools: It is a tie in this aspect. Apart from CUDA™ and OpenCL, there are no major development options for GPUs. For the Cell platform, IBM provides the Cell SDK and Eclipse can be used as the development IDE.
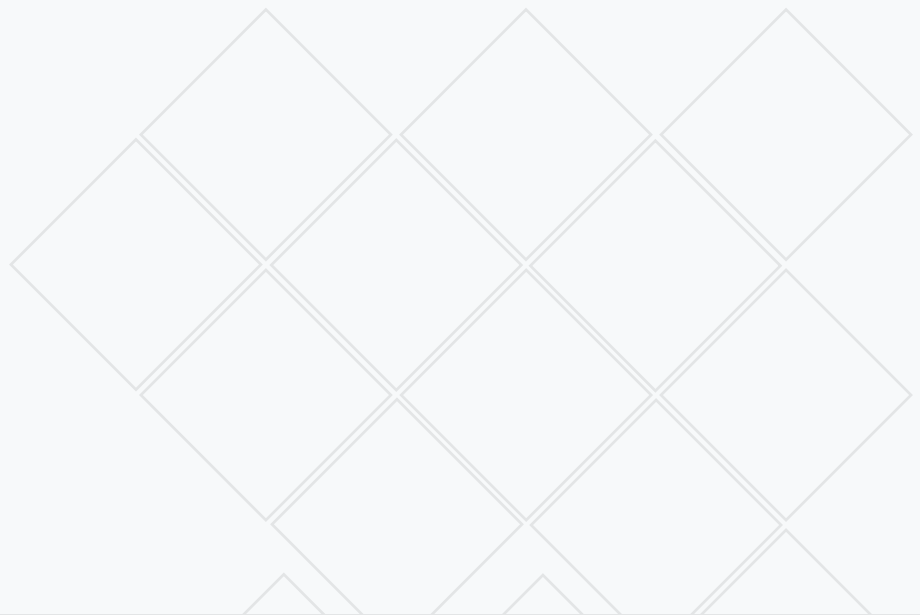
Conclusion

Considering all of the above aspects, the GPU has a clear advantage over the Cell platform. Most importantly, looking at the Cell roadmap indicates that the Cell family of processors won't be developed further, although current designs will be used in future IBM products. The GPU has got a clear roadmap laid out, with strong emphasis on HPC and compute oriented workloads, apart from the traditional 3D= gaming/professional applications market.

In light of the current circumstances, the GPU seems to be the platform of choice for HPC/compute intensive applications today. But this does not mean that the Cell BE is a non-performer –

It still is a good option for certain type of applications/Workloads which are better suited to itsarchitecture and/or not portable to GPU.

References

1. IBM Cell developer documentation [http://www.ibm.com/developerworks/power/cell/downloads.html]

2. nVIDIA CUDA developer page [http://www.nvidia.com/object/cuda_get.html]

3. ATI GPU developer central [http://developer.amd.com/GPU/Pages/default.aspx]

BORN TO ENGINEER