NVIDIA Pascal architecture improvements

Seminar Future Trends in Computing

Guillermo González de Garibay Barba
Fakultaet fuer Informatik
Technische Universitaet Muenchen
Email: g.garibay@tum.de

***Abstract*— Single-CPU-systems cannot achieve significant speed-ups by increasing clock speeds anymore. There is not much that can be done in that way for making sequential algorithms orders of magnitude faster. However many algorithms have at least some parallelizable sections. There are many different ideas for configuring systems that work more efficiently under heterogeneous workloads. In debate are the heterogeneous and self-hosted models, CPU and accelerator architecture, and the types of inter-/intra-node interconnects. The Heterogeneous Computing model consists on having different processors for different tasks. Concretely GPUs are in common use for throughput-intensive tasks and CPUs for latency-sensitive tasks. Latency-optimized CPUs use complex branch prediction, speculative execution, register remaining, etc. This requires a lot of silicon surface and thus also a lot of energy. Throughput-intensive systems reduce these complexities and work at minimal energy per bit. The advantage of GPUs for computation is achieved by dedicating more transistors to arithmetic logics and less to control. Nvidia is betting on Heterogeneous Computing by using their GPUs as accelerators working side by side with CPUs. Nvidia is now building their GPUs not only for graphics but also for High Performance Computing. This document focuses on Pascal — the latest architecture presented by Nvidia — with an special focus on the new interconnect, NVLink. NVLink is specially relevant because it breaks the PCIe bottlenecks appearing at GPU-to-GPU communication, allowing for faster multi-GPU systems. Then Pascal is put in contrast with the previous architectures Maxwell and Kepler. Finally there is a description of an attempt of creating a general hardware model for GPGPU.**

***Index Terms*— Nvidia, Pascal, GPGPU, NVLink, Unified Memory**

I. Nvidia Pascal architecture [?]

In April 2016, the Nvidia Pascal architecture was presented. Nvidia's most powerful chip using this architecture is called GP100, which they have built into the Tesla P100 Accelerator.

The Tesla P100 has 5.3TFLOPS of FP64 performance, double as much (10.6 TFLOPS) of FP32, and up to four times as much (21.2 TFLOPS) of FP16. The 16-bit floating point precision ability is thought for Deep Learning, which was the focus at Nvidia's presentation, althought it could also be useful for graphics. Deep Learning does not require higher precision and this brings both higher speed and higher available storage for bigger models.

In the following sections the main features of the new architecture are described. All of them are present in the Tesla P100 Accelerator.

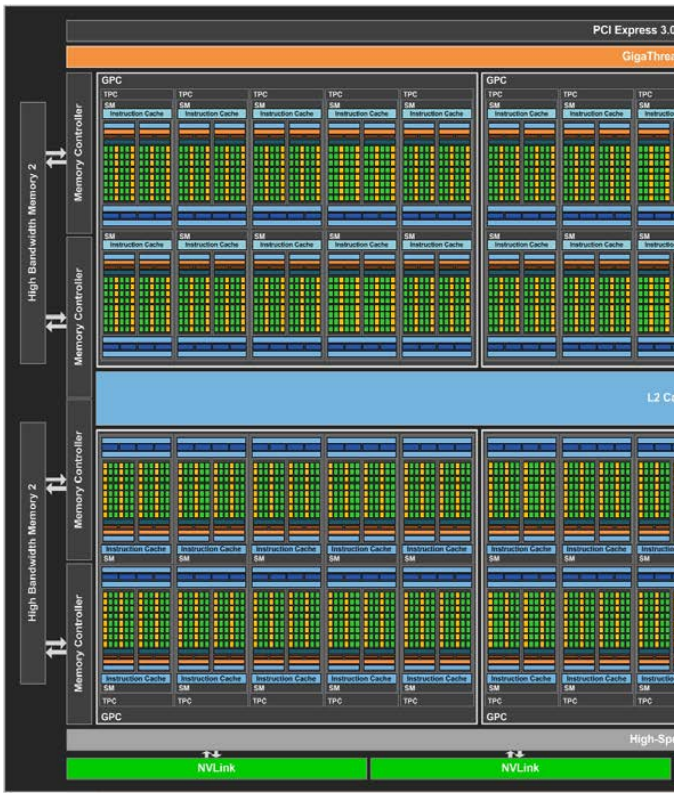

Fig. 1. Representation of half of the GP100 Pascal silicon chip. The right half is completely symmetric. [?]

A. Updated processor structure

The Pascal architecture is structured as an array of Graphics Processing Clusters, each GPC is formed by several Texture Processing Clusters, consisting of 2 Streaming Multiprocessors per TPC. The reason for conceptually grouping 2 Streaming Multiprocessors into one Texture Processing Cluster is not specified. The GP100 processor has 6 Graphics Processing Clusters with 10 Streaming Multiprocessors each. Each Streaming Multiprocessor contains 64 CUDA FP32 Cores and 4 texture units (see Figure **??**). This adds up to a total of 3840 CUDA Cores ($6 \times 10 \times 64$) However, 4 of the Streaming Multiprocessors on the chip are turned off, giving a total of 56 available Streaming Multiprocessor units.

Each Streaming Multiprocessor has two sets of 32 FP32 CUDA Cores, an instruction buffer, and a warp scheduler with two dispatch units.

Fig. 2. Representation of a Pascal Streaming Multiprocessor capable of simultaneously executing two sets of 32 threads

There is a new scheduler data path too. As in older architectures, each warp scheduler in Pascal Streaming Multiprocessors can dispatch two independent instructions simultaneously. There is also a 1:2 ratio of FP64 to FP32 CUDA Cores per Streaming Multiprocessor, working better with the updated data path. FP16 is handled by the same FP32 ALUs.

The minimum block size for FP32 remains of 32 threads. The minimum block size for FP16 and FP64 is not specified.

B. Compute Preemption

The GP100 chip has preemption at instruction-level. It allows interaction with long computing tasks, swapping contexts to GPU DRAM. Otherwise, long running tasks could end up being killed by the OS or the CUDA driver. Additionally, if a GPU is being used for display graphics and CUDA tasks, long running tasks could result in an unresponsive GUI. This does not happen anymore. This also makes interactive debugging work better. Interactive debugging on Kepler and Maxwell required adding instrumentation during compilation to allow completion of thread blocks after an interrupt. Thus GP100 debugging is more robust and lightweight.

C. Memory

1) On-chip memory: As in previous architectures, memory is structured hierarchically. There are two cache levels — L1/shared and L2 — which improve access to off-chip memory.

The L1/shared memory structure has not changed since Maxwell. Each Streaming Multiprocessor disposes of 64KB shared memory — up to 32KB per Thread Block. L1/texture cache is additional to that. The Register File Size per Streaming Multiprocessor remains at 256KB. It should be noted that the number of cores per Streaming Multiprocessor has been at least halved compared to previous architectures, while the number of Streaming Multiprocessors increases. A summary of this evolution can be seen in Figure **??**. All this additional fast memory improves thread concurrency capabilities.

There are eight 512-bit memory controllers to communicate with off-chip memory, as seen on the left side of Figure **??**. There is a unified 4096KB L2 cache, which gives 512KB per memory controller.

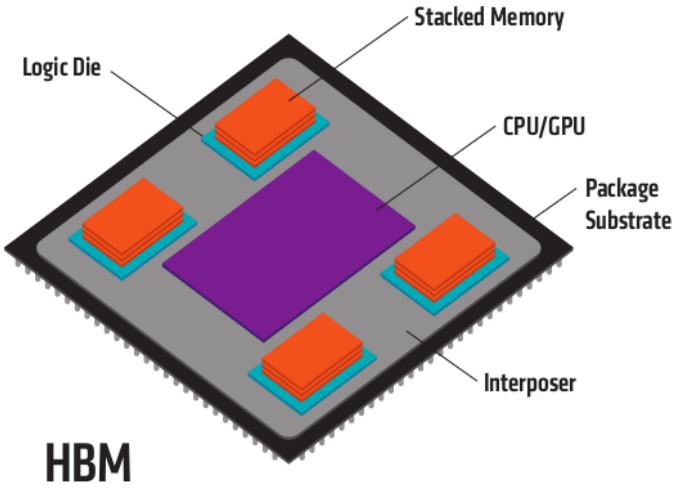

Fig. 3. Chip structure using HBM memory [**?**]. The memory and the chip share a silicon interposer connection [**?**].

2) Off-chip memory — HBM2: HBM2 is the second generation of High Bandwidth Memory. HBM is the first 3D-memory technology to be in broad use. Each stack of HBM2 is controlled by a pair of memory controllers. A HBM2 stack can be composed by 4 or 8 dies with perpendicular microscopic wires (called through-silicon vias) that go down the stack. Each HBM2 die has a 8Gb capacity, giving a total of 4 or 8GB per stack. This distribution enables a higher number of connections, increasing the communication bandwidth. Stacked memory uses less surface, which also allows for denser servers. The Tesla P100 uses 4 stacks of 4 dies, providing with 16GB of memory.

Each die is connected through two 128-bit channels. These channels are independent and not necessarily asynchronous. The stacks are connected to the chip via a passive silicon interposer (see Figure **??**). It is a large piece of passive silicon also using through-silicon vias (TSVs) for connections. This means a large silicon surface, which is expensive. That is why lower-range Pascal cards — such as graphics oriented GTX1080 — will use GDDR5 or GDDR5X. The height of the dies is adjusted — the top die is thicker — in order to make good contact with the heat sink.

HBM2 is a significant scale up on its previous version. HBM1 supported only 4 dies per stack and 2Gb per die. HBM1 supported 125GB/sec per stack while HBM2 supports 180GB/sec.

When working on large clusters or having long application executions, it becomes important to have a memory error correction system. HBM2 has native support for ECC. This contrasts with GDDR5 which does not have internal ECC, which only allows ECC to detect error on the bus. This approach requires allocating an 6.25% of the memory for error correction, and results in a 12-15% reduction in bandwidth.

Internal ECC detects and corrects single-bit soft errors before they affect the system, without requiring to reserve some of the memory for this function.

Another relevant incorporation is a new block on the chip called High-Speed Hub (HSHUB) — on the bottom of Figure **??** — which has access to the High-Speed Copy Engines enabling NVLink access. NVLink is an important addition that makes working with Unified Memory more efficient.

3) Unified Memory: The path to Unified Memory is being followed since several CUDA versions ago. CUDA 4 introduced Unified Virtual Addressing. Unified Virtual Addressing enabled pinning CPU memory to be accessible directly over PCIe without a memcpy. This provided convenience but no performance improvement.

The term Unified Memory was introduced in CUDA 6. In CUDA 6, not only is pinned memory accessible through PCIe, but a single pointer can refer to both CPU and GPU memory. This requires pinned CPU memory to be associated with GPU memory of the same size, and limits the unified address space to the size of GPU memory. A single memory allocation in pinned CPU memory and GPU memory is kept synchronized through software. As described in section **??**, all managed memory touched by the CPU has to be synchronized. Additionally, CPU and GPU cannot access the same memory allocation simultaneously.

Together with Pascal and NVLink, the CUDA 8 platform has been released. Now a single memory address space for CPU and GPU with 49-bit virtual addressing has been introduced. Since current CPUs have up to 48-bit virtual addressing, 49 bits is enough to cover both spaces as a single virtual address space. Now the memory limits of the GPU do not restrict the virtual space size. Additionally, the GP100 chip has hardware support for page faulting, which allows transparent access to data anywhere in the virtual address spaces. This means there is no need for synchronization of all managed memory allocations. The large address space allows direct addressing of very large datasets, much larger than the system memory. Faulted pages are automatically migrated. Pages can also be mapped on the GPU for access through PCIe or NVLink. In some cases, this can be faster than migrating. CPUs can also fault and migrate memory from GPU. CPU can even access data while a GPU kernel is running. Coherency in this case could not be guaranteed in previous architectures. Note that correct synchronization has to be taken care of as in any other parallel application.

Operating system support is being developed in collaboration with Red Hat and the Linux community to enable GPUs to access memory allocated with the default OS allocator, such as malloc.

Aside from making GPU programming easier, this also allows easier use of C++ classes on GPU. Any nested data can be automatically accessed thanks to the single virtual address.

It is important to note that Unified Memory can easily lead to communication bottlenecks. In order to alleviate this problem, Nvidia has developed NVLink.

D. NVLink

Clusters of multi-GPU systems are being interconnected with InfiniBand(R) and 100Gb Ethernet. The GPUs to CPUs ratio is increasing. Fast cluster interconnections and this increasing ratio are causing PCIe to become a significant bottleneck for data-intensive tasks. That is why NVLink has been developed.

1) Previously existing features: RDMA has already been possible for some time. GPUDirect was introduced in Kepler and allowed RDMA, lowering CPU overhead. This is thanks to the CPU not having to copy the data inside its own memory. It also enabled P2P data transfer between GPUs. GPUDirect bandwidth is doubled in GP100.

2) NVLink features: NVLink introduces GPU-to-GPU data transfers, avoiding both CPU overhead and PCIe bottlenecks at once. It uses the new Nvidia High-Speed Signaling (NVHS) interconnect. It is compatible with the GPU ISA, supporting shared memory multiprocessing. Programs can execute directly on memory of another GPU with full capabilities.

Additionally, CPU-to-GPU NVLink connections will also be possible for compatible CPUs. The connection to non-supporting CPUs will still happen through PCIe. The "Power8 with NVLink" processor is the first processor supporting the technology. It was previously known as Power8+. "Power8 with NVLink" has 6 NVLink connectors, allowing for multiple network topologies [**?**]. The next generation Power9 will support both NVLink and IBM's own CAPI [**?**]. It will be launched in 2017. An schematic of this processor series can be seen in Figure **??**.

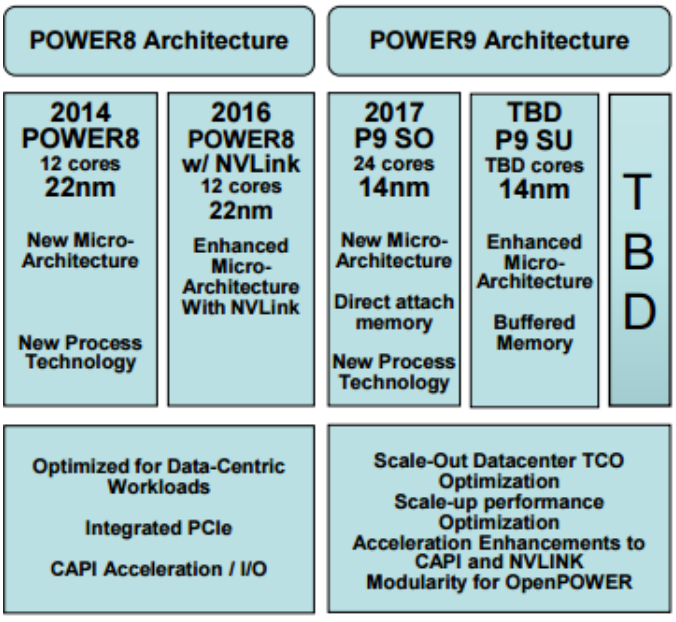

Fig. 4. IBM NVLink supporting processors timeline [OpenPower Foundation]

This processor together with Nvidia GPUs will be the building blocks of the pre-exascale Summit and Sierra supercomputers. This is described in section **??**. These supercomputers

will be the first using Volta, Nvidia's next generation GPU architecture. This will be in 2017. However a broader launch of Volta GPUs is not expected until 2018. No Intel processor support for NVLink is expected as of June 2016. This is probably due to Intel's Xeon Phi processor series, a competitor to GPU accelerators. Xeon Phi follows a different approach for hardware accelerators.

3) NVLink design: Data is transmitted over a set of differential pairs with a bandwidth of 20Gb/sec each pair. A total of 16 pairs form a Link, with 8 pairs for each direction. This results in a total bidirectional bandwidth of up to 40GB/sec per Link. It is important to remark that this is bidirectional because there has been some confusion with this. PCIe Gen 3 x16 — the current technology in use — provides a bidirectional bandwidth of up to 31.75GB/s. In this way NVLink is just about 20% faster than PCIe, and slower than PCIe 4.0 which doubles the bandwidth of PCIe 3.0. However it is NVLink's capability of joining several links to add up their bandwidths what makes it really faster. Since the Tesla P100 supports up to 4 links, this allows up to 160GB/sec bidirectional data transfer. A couple of Tesla P100 cards can thus be connected with a bandwidth of up to more than 5x that of PCIe 3 x16. While a single link of NVLink has smaller bandwidth than PCIe 4.0, a NVLink 2.0 specification is expected for 2017, improving on PCIe 4.0 [**?**]. NVLink also uses up to a third of the energy to move data compared with PCIe Gen3 x16 [**?**].

4) NVLink layer model: The NVLink controller is formed by a Physical Layer, a Data Link Layer, and a Transaction Layer. The package sizes range from 1 to 18 128-bit flits. The clock runs at 1.25GHz. It uses an embedded clock used by the receiver to capture data. The Physical Layer takes care of deskewing across lanes, framing, scrambling/descrambling, polarity inversion and lane reversal. The Data Link Layer is responsible for reliable transmission. Packets are protected with a 25-bit Cyclic Redundancy Check. It is calculated over the current header and the previous payload. It allows detection of up to 5 random bit errors or up to 25-bit bursts of errors on any lane. Packets are stored in a replay buffer until the receiver acknowledges them. If the transmitter times out waiting for acknowledgment, it starts retransmitting.

The Transaction Layer cares for synchronization, link flow control, and virtual channels. It can also aggregate multiple links together to provide higher communication bandwidth.

II. NVLink performance

In [**?**], a basic performance analysis on NVLink is presented. Since it was done in 2014, the analysis was done on a theoretical basis. It focuses on the performance benefits for GPU-to-GPU communications through NVLink. Two systems with and without NVLink are analyzed for the different cases. The systems with NVLink can be seen in Figure **??**. The base case systems are the same just removing the NVLink connections. It assumes a 80% communication efficiency for both PCIe and NVLink.

The theoretical test-cases are a multi-GPU exchange and sort, a Fast Fourier Transform, AMBER-Molecular Dynamics,

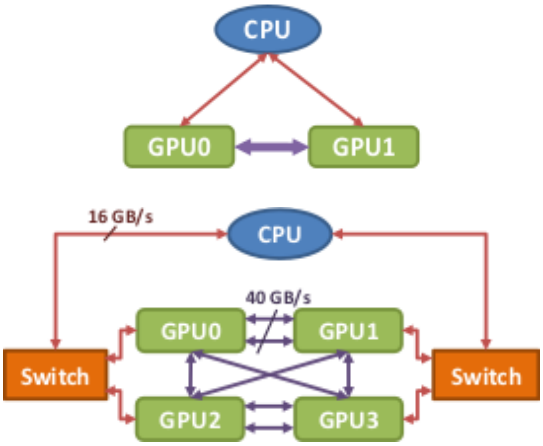

Fig. 5. Two possible NVLink + PCIe configurations

ANSYS Fluent, and a Lattice Quantum Chromodynamics library. The exchange and sort case achieves an improvement directly proportional to the parallel communication speed-up. The FFT case shows that NVLink also makes multi-GPU scaling worth for smaller problems by reducing the communication/computation time ratio. AMBER's PMEMD algorithm can be decomposed into a N-body simulation and a FFT which can run asynchronously. This is already a complex case to analyze, but it is estimated that a 30-50% improvement for 4 GPUs can be achieved just by using NVLink. In the ANSYS Fluent case an improvement of 25% for 2 GPUs is expected to be achieved through a synchronous communication speed-up. In the LQCD case, the speed-up depends on the memory bandwidth, the interconnect bandwidth and the surface-to-volume ratio of the computation. It is estimated that a 25% improvement for 4 GPUs is also to be expected.

III. NVLink systems

NVLink allows for many new possible configurations. With 4 NVLink ports per Tesla P100 card and 6 per Power8 CPU, and future CPUs with both PCIe and NVLink available, the number of possible configurations is very large. This design flexibility enables system optimization for different parallel/serial code ratios. This helps minimizing the impact of Amdahl limits. In this section come under discussion some of the systems which have already been announced.

A. NVIDIA DGX-1

In April 2016 the DGX-1 was announced to be produced by Nvidia. It is recommended by Nvidia for Deep Learning, achieving 170 FP16 TFLOPS in a single node. It consists of 2 Dual 20-core Intel Xeon E5-2698 v4 2.2 GHz and 8 Tesla P100 GPUs. 512GB of DDR4 memory and 128GB of HBM2 memory are available for use as Unified Memory. Each CPU is connected to 4 GPUs through 2 PCIe switches. These 2 subsets of 4 GPUs are fully connected through NVLink. Each subset is also connected through NVLink to the respective GPU of the other subset, as can be seen in Figure **??**.

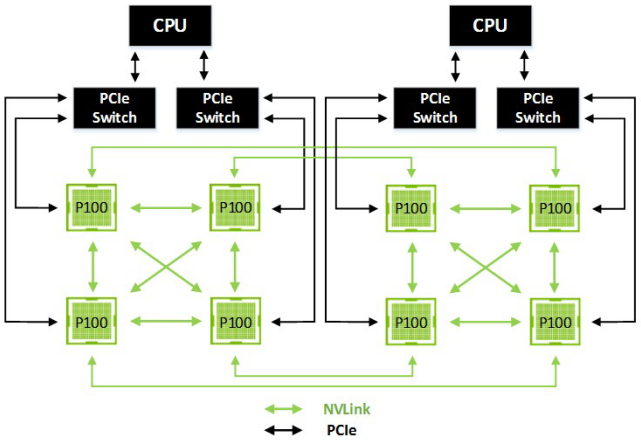

Fig. 6. DGX-1 is connected as a Hybrid Cube Mesh [**?**]

It is expected to provide up to a 12x speed up with respect to the fastest Maxwell node. The FP16 capability alone accounts for a 2x factor. Most of the remaining 6x factor is due to increased memory and interconnect bandwidths.

B. NVLink in supercomputing [?]

Pushed by the U.S. Department of Energy, the Collaboration of Oak Ridge, Argonne, and Livermore, are planning pre-exascale systems. Two of them — Summit and Sierra — will be based on the OpenPOWER platform, but will have unique system configurations Both will consist of nodes formed by several Power9 CPUs and several Volta GPUs. However, only the approximate details of Summit are available and discussed in this section.

Sierra — managed by the Lawrence Livermore National Laboratory — will be used for the U.S. nuclear program and counterterrorism. Sierra will work at at least 100 petaFLOPS. It will replace Sequoia — a IBM Blue Gene/Q system — which was the fastest supercomputer in 2012.

Summit — managed by the Oak Ridge National Laboratory — will be used for fundamental scientific research. Titan — with 1 GPU per CPU — is the current top system of ORNL and the 2nd in the TOP500 list. It is currently oversubscribed. Summit will perform at 150-300 petaFLOPS, which is about 5-10x of what Titan performs, using only about 10% additional power. It will consist of more than 3400 compute nodes, each performing at more than 40 TFLOPS, with 512GB DDR4+HBM and 800GB NVRAM configured as burst buffer or extended memory. The nodes will be connected as a full non-blocking fat-tree through dual-rail Mellanox EDR Infiniband. The whole system will use a General Parallel File System with 1TB/s of I/O bandwidth and 120PB of disk capacity.

The key ideas mentioned for the design of both systems are OpenPOWER, NVIDIA Tesla, Heterogeneous Computing and NVLink.

IV. Unified Memory Analysis [**?**]

An evaluation of Unified Memory is provided in [**?**]. It describes the Unified Memory programming model and presents an evaluation methodology to try to understand how the automatic system works. It is known that the system handles data migration transparently, but the workflow is unknown. The tests are carried on the K40 Accelerator and the Jetson TK1 (embedded). The latter is interesting because it has physically unified memory.

To evaluate the performance, three different benchmarks are used. One is the Matrix Multiplication benchmark. It consists of tests with both cuBLAS and a block matrix multiplication that uses shared memory. The second is the Diffusion 3D Benchmark. The last one is the Parboil Benchmark Suite, created for studying the performance of computer architectures and compilers. Only the benchmarks using a constant memory size are used. The comparison is made by modifying them to use Unified Memory. No optimization changes are made. The tool used to analyze the different runs is the NVIDIA Profiler. The Profiler provides a graphical interface breaking down the time consumption among sub-tasks, streams, etc. It is also shown that the structure and content of PTX codes (pseudo-assembly codes for CUDA) does not change.

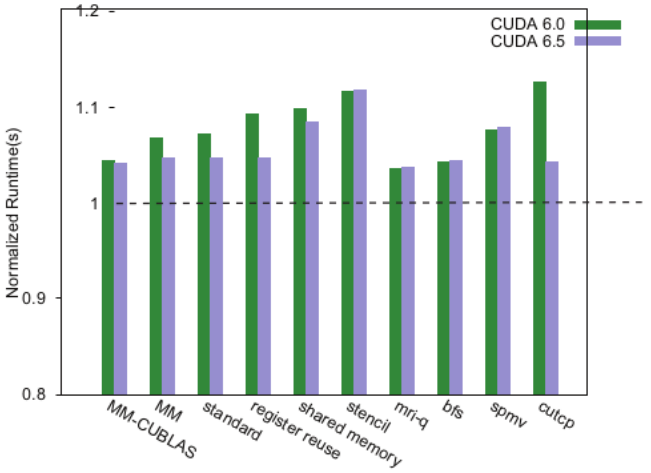

Fig. 7. Benchmark comparison results, where 1 is the runtime of the version without Unified Memory [**?**]

As seen in Figure **??**, it is found that there are 10% performance losses in average. This is shown to be due to redundant memory transfers and page caching faults. These losses are in some benchmarks considerably smaller under CUDA 6.5 than under CUDA 6.0. The kernel running time in CUDA 6.5 does not improve. It is the performance of launching and synchronizing what improves between both CUDA versions. Interestingly, the Jetson TK1 uses the memory of CPU and GPU separately, even when it is physically one, resulting in unnecessary copies. It is also shown that pinned memory is used for Unified Memory. This is revealed by the fact that copying times from host to device are much lower under Unified Memory.

Finally, five micro-benchmarks are created to look for the conditions causing the performance loss. It is seen that memory is copied from host to device even if no kernel uses

it. Memory also gets copied back and forth even if it is only read by the GPU. Only in the case that the CPU never reads or writes the data, are there no redundant memory transfers.

This same research group intends to analyze the performance under of Unified Memory in multi-GPU systems in the future. These results should suffer very significant changes due to hardware support for page faulting and the higher speeds of NVLink.

V. Nvidia and GPGPU

A. Background

GPUs with programmable shaders supporting floating point operations were the beginning of GPGPU. It started by using graphics libraries to implement tensor operations. One of the first implementations was a LU factorization in 2005 [**?**]. Nvidia's CUDA platform was launched in 2008 to allow programming while ignoring the underlying graphical concepts. Not alike alternatives — such as OpenCL — CUDA only works on Nvidia GPUs. Its development has been tied to the development of the Tesla family of GPUs. Just as GeForce is Nvidia's Desktop family, and Jetson their embedded family, Tesla is their HPC family. Tesla was also the name of the microarchitecture in 2008, when they introduced CUDA. Since then there has been a mostly continuous progress towards their latest microarchitecture, Pascal, presented in April 2016. The biggest changes were introduced in the Fermi microarchitecture.

B. Historical progress

In 2010, Nvidia presented the Fermi microarchitecture, easing intensive use of GPUs in HPC. Fermi introduced 40-bit memory addressing, and ISA support for up to 64-bit addressing [**?**]. It adopted the IEE 754-2008 floating point standard and enabled 64-bit floating point operations at only half of the throughput of FP32. Fermi also introduced the current hierarchical cache with L1 cache connected to L2 cache and L2 to DRAM, with unified access to the three levels. L2 and DRAM were enabled to access host CPU memory through PCIe. The amount of L1 and L2 were configurable for the programmer, allowing a variable fixed fraction of shared memory.

The next generation, Kepler, was introduced in 2012 and introduced Dynamic Parallelism. Dynamic Parallelism refers to enabling GPU threads to launch other threads (kernels launching kernels). This provides the programming model with more flexibility. This architecture is the one currently in use for GPGPU HPC tasks requiring 64-bit floating point precision.

The Maxwell microarchitecture, presented in 2014 introduced Unified Virtual Memory. This was an important step in simplifying the programming model. In this architecture, FP64 efficiency dropped by one order of magnitude. Maxwell focused on graphics and per Watt performance.

The Pascal microarchitecture, presented in 2016, brings the features of the Kepler and Maxwell microarchitectures together. It improves the Unified Virtual Memory model, thanks to page faulting and NVLink. NVLink comes as an alternative to PCIe, but it does not replace it. Pascal also brings back FP64 at high rates. Memory bandwidth is improved thanks to the use of HBM2 memory. FP16 operations are supported at double the rate of FP32 operations. Pascal also introduces preemption at the instruction level. All this comes with a reduction of the transistor size by using a 16nm FinFET technology.

The Volta microarchitecture was supposed to follow Maxwell. However, lack of maturity of Hybrid Memory Cube versus High Bandwidth Memory, forced Nvidia to switch to the Pascal microarchitecture [**?**]. Volta systems will be installed in supercomputers in 2017, coming to the market in 2018.

C. Recent evolution — Pascal vs Kepler and Maxwell

Tesla Products	Tesla K40	Tesla M40	Tesla P100
GPU	GK110 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)
SMs	15	24	56
TPCs	15	24	28
FP32 CUDA Cores / SM	192	128	64
FP32 CUDA Cores / GPU	2880	3072	3584

Fig. 8. Extract of table comparing Kepler, Maxwell and Pascal [**?**]

The intent of this section is to try to make the recent architecture changes clear. The relevant architectures for this comparison are Pascal, Maxwell and Kepler. The chip representing Pascal is the GP100, installed in the Tesla P100. Kepler has 2 representing chips, the GK110, part of the Tesla K40, and the GK210, part of the Tesla K80. The K40 was announced in November 2013 and the K80 a year later, extending the life of Kepler. This extended life was probably needed due to the lack of FP64 performance on the newer 2014 Maxwell architecture. The top Maxwell chip GM200 was released in 2015 with the best performance if not considering FP64 performance. Note that the letter preceding the chip model number, stands for the first letter of the architecture name.

In the more recent architectures a clearer hierarchical organization can be seen. This is related to the GPGPU model presented in Section **??**. GM200 and GP100 have both 6 Graphics Processing Clusters, formed by Texture Processing Clusters, formed by smaller Streaming Multiprocessors. In contrast, the Kepler chips are structured as an array of 15 Streaming Multiprocessors with lots of processing cores each. The number of CUDA Cores per Streaming Multiprocessor keeps decreasing. The GK110, GM200, and GP100 have respectively 192, 128 and 64 FP32 CUDA Cores per Streaming Multiprocessor. Kepler has a ratio of 1:3 FP64 to FP32 CUDA Cores, which was decreased to 1:32 in Maxwell. Pascal has increased this ratio to 1:2 by having FP16 as part of the FP32 ALUs. The number of Streaming Multiprocessors per GPU increases continuously, giving an increasing number of CUDA Cores per GPU.

The scheduler data path in Pascal is changed with respect to Kepler and Maxwell. A big improvement in the scheduler

in comparison to Maxwell is that each warp scheduler can dispatch two warp instructions per clock. This allows for launching simultaneously two independent instructions (e.g. to ALU and Load/Store Unit).

Instruction-level preemption is a very relevant Pascal feature vs block-level preemption on the Maxwell and Kepler architectures.

Maxwell improved on the 64KB configurable shared memory/L1 cache of Kepler by increasing the available memory and creating separate spaces for shared memory and L1 cache. The L2 cache in Pascal is not much larger with 4096KB, but the total L1 cache available per CUDA Core gets doubled. The Register File Size per Streaming Multiprocessor has also been kept constant while decreasing the number of cores. All this effectively increases the amount of fast cache and registers available per core, improving execution speed and concurrency. GK210 already included double as much L1/shared memory per core vs GK110.

GK110 and GM200 had a 384-bit GDDR5 memory interface. GP100 has a 4096-bit memory interface when using High Bandwidth Memory v2. This makes GP100 have 3x the bandwidth of the Maxwell GM200 GPU even with HBM2 having a 3-4x slower clock. Higher memory bandwidth improves work on larger data sets.

Kepler implemented atomic memory operations in software just like Fermi in a lock/update/unlock pattern. Maxwell enabled native hardware support for atomic operations with 32-bit integers and 32-bit and 64-bit compare-and-swap operations. Pascal allows atomics in Unified Memory, including 32 and 64-bit support for both integers and floats.

A constant in all this evolution is the TFLOPS/W improvement, effectively serving as a modification of Moore's law.

VI. GPGPU

The paper [**?**] introduces a formal model trying to generalize the computational model of GPUs. It refers to [**?**] as introducing a new computing era, called GPGPU. There are many attempts of describing the internal mechanisms of GPU operation. Some are focused on a graphics perspective, some others on general purpose. New generation devices are surpassing these models and [**?**] tries to develop a broader model. The task is difficult since most reports are either too simple or black-box descriptions [**?**]. White papers focus on capabilities, and programming guides focus on code syntax. Some micro-architecture descriptions are too focused on a detail, ignoring the rest.

A hierarchical descriptive model for GPGPU is proposed in [**?**]. It combines the ideas of OpenCL and the Multi-BSP [**?**] bridging model. A Processing Element of a higher level in the hierarchy sees lower-level Processing Elements as atomic. The CUDA architecture is then described within this model as a two-level structure with nested components. The architecture details are not public and won't probably ever be [**?**]. Thus the description has to be based on official patents and research papers.

1) CUDA under the descriptive model: The GPU is the atomic element forming the second level of the hierarchy. The first level is composed by the GigaThread Engine, CUDA Work Distributor, Data Transfer Engine, Streaming Multiprocessor array, and local/global memory. Each Streaming Multiprocessor is broken into analogous components of the 0th level. These are the Thread controller, Warp scheduler, Load/Store Units, local/shared registers and ALU arrays.

Analogous components of different levels have different behaviors. For example, the Processing Elements which are the ALU arrays (0th level) and Streaming Multiprocessor array (first level). Each Streaming Multiprocessor works asynchronously, whereas groups of 32 ALUs have to work synchronously under a common program counter. Other analogous components at different levels are Load/Store Units vs Data Transfer Engine arrays, local/shared registers vs local/global memory, task Buffers (program/block counters vs reference counter), Warp scheduler vs CUDA Work Distributor, and Thread Controller vs GigaThread Engine. The GigaThread work scheduler balances work among Streaming Multiprocessors. The Streaming Multiprocessors scheduler executes concurrent threads to help reduce the effect of long latency loads from DRAM memory.

*2) Future Trends proposed in [**?**]:* The current hierarchy consists of 2 levels, but this is not intrinsic to the model. We could expect to see hierarchies with more levels or tasks of different levels being launched on the same hardware level. Recursive kernel invocation and maybe the DGX-1 — with several GPUs forming a higher level — point in this direction.

The hierarchical cache system reflects the convergence trend between CPU and GPU. GPGPU is fundamentally a hierarchical system of moving data to computing. It could make sense to develop a system where computing is moved to data. We could thus also expect GPUs to part away from this Von Neumann hierarchical model and go in the way of neuromorphic chips. These neuromorphic chips have ALUs and memory mixed, which allows simultaneous and asynchronous execution of different dataflows.

VII. Conclusion

There is a large ongoing competition on how to architecture massively parallel systems. Nvidia's Pascal architecture represents a big step forward in the CPU-Accelerator approach. The most relevant new features are the new GPU-to-GPU interconnect NVLink and how this enables a functional Unified Memory model. Additionally, Nvidia's ability to influence CPU design in through the OpenPower Foundation, brings a new competition approach. It will be interesting to see how the competition with Intel's Many Integrated Core Architecture develops.