2016 IEEE International Conference on Computational Science and Engineering, IEEE International Conference on Embedded and Ubiquitous Computing, and International Symposium on Distributed Computing and Applications to Business, Engineering and Science

HPC Accelerators with 3D Memory

Manuel Ujaldón
Computer Architecture Department
University of Málaga
Málaga (Spain)
Email: ujaldon@uma.es

Abstract—**After a decade evolving in the High Performance Computing arena, GPU-equipped supercomputers have conquered the top500 and green500 lists, providing us unprecedented levels of computational power and memory bandwidth. This year, major vendors have introduced new accelerators based on 3D memory, like Xeon Phi Knights Landing by Intel and Pascal architecture by Nvidia. This paper reviews hardware features of those new HPC accelerators and unveils potential performance for scientific applications, with an emphasis on Hybrid Memory Cube (HMC) and High Bandwidth Memory (HBM) used by commercial products according to roadmaps already announced.**

I. Background: Processors

The icon for processors over the last 50 years has been Moore's law, but doubling the number of transistors every 18 months was often confused with doubling performance. Mainly because it was not hard to convert the number of functional units into GFLOPS, particularly on those early years when pipelining in the 80's and superscalar designs in the 90's were driving commercial models. Last decade, the pillar was multi-core, but once again, the idea turned out not to be scalable for a CPU design. In the meantime, we realize that scalability is possible on GPUs, and that is, in essence, its contribution along last decade, and the primary reason for the rapid transition to many-core GPUs that we are witnessing lately.

Nowadays, disruptive technologies such as heterogeneous multi-cores and GPUs offer excellent performance/cost ratios for scientific applications, and an increasing number of developers have learnt to program them to take full advantage of that emerging power. GPUs have moved closer to CPUs in terms of functionality and programmability, and CPUs have also acquired features that are GPU alike. Two good exponents of this stronger CPU-GPU coupling are the Fusion project led by AMD to integrate a CPU and GPU on a single chip, and the Larrabee project led by Intel to develop a many-core hybrid platform using x86 CPUs. The Intel movement continued with Knights Ferry, Knights Corner and Knights Landing to establish the MIC (Many Integrated Core) Architecture [4] and finally release the Xeon Phi family of accelerators [5]. In 2016, the last generation of Xeon Phi was released to include memory controllers for 3D DRAM, allowing programmers to use the x86 instruction set architecture and choose where they allocate DRAM memory, either using typical DDR modules or novel 3D cubes [6].

In parallel with the CPU evolution, the GPU started its own way towards high performance computing fifteen years ago. Graphics programming experienced a revolution with the advent of shaders, methods to program vertex and pixel processors to leverage creativity in visual effects. First, HLSL (High Level Shading Language, 2001) led by Microsoft for its Direct3D pipeline, and right after GLSL (OpenGL Shading Language, 2002), the OpenGL counterpart, became popular at that time, followed by Cg (C for Graphics, 2003), developed by Nvidia with Microsoft as partner. In 2005, Nvidia unified vertex and pixel shaders leading to a more versatile core design called streaming processor. Soon multi-core vertex and pixel processors at Nvidia (2001), which were ancestors for the subsequent multi-core CPUs at Intel (2005), turned many-core with general-purpose capabilities, and in November, 2006, CUDA (Compute Unified Device Architecture) [8] was announced as the hardware and software paradigm to design and program GPUs for HPC.

Now reaching its tenth anniversary, the evolution of CUDA has been impressive. Table I summarizes major achievements according to Nvidia and the NSF. We may fairly say that for the first time in HPC history supercomputing was democratized, combining three features never gathered before: Price, power and ubiquity. The free availability of CUDA tools to interact with other software communities (open-source compilers, wrappers, back-ends) plus the generosity of CUDA programmers, with thousands of source codes and libraries, created a friendly ecosystem for developing applications. And the last milestone attained by GPUs is energy efficiency, with all top 40 supercomputers within the green500 list [3] composed of accelerators, 31 of them being Nvidia GPUs.

In terms of raw computational power, GPUs are ahead of CPUs roughly an order of magnitude. But peak processing power is harder to reach on the GPU, so this difference shortens in practice. Typical rates are sensitive to application's nature, but in general, a GPU programmer is happy squeezing 40% of peak performance, whereas a CPU programmer is often disappointed in the 60% range.

Another remarkable difference lies in the programming model. Multi-core CPUs use vector processing when enabling multimedia extensions (MMX, SSE, AVX), instruction-level parallelism when enabling HyperThreading, and coarse-grain at thread level parallelism, either relying on the scheduler of the operating system or programming explicitly via *POSIX threads*. Many-core GPUs are programmed using the SIMT (Single Instruction Multiple Thread) model to enable data parallelism in a more scalable manner, particularly in the big data era. The idea is to run the same program in all cores, but each thread instantiates on a different data subset for each core to work effectively in parallel. That way, the more cores we have available, the smaller the working region becomes for each core, thus making the execution time to be reduced

TABLE I. THE IMPRESSIVE EVOLUTION OF CUDA OVER THE LAST DECADE.

Year	2008	2016	Multiplier
Number of GPUs accepting CUDA	$>$ 100.000.000	$>$ 600.000.000	6x
CUDA downloads monthly	$>$ 10.000	$>$ 300.000	30x
CUDA-enabled supercomputers within Top500 list	1	10 Fermi + 53 Kepler	104x
Aggregate performance for those supercomputers	77 TFLOPS	$>$ 80000 TFLOPS	1039x
University courses teaching CUDA	60	$>$ 800	13x
Scientific papers published using CUDA	4.000	$>$ 60.000	15x

proportionally without even having to recompile the code. The key for a successful massive parallelism on a SIMT deployment is data partitioning. Straightforward decompositions are usually derived from 1D, 2D and 3D matrices as input data sets, overall on regular access patterns, but the goal turns more difficult in the presence of irregular accesses or dynamic data structures.

II. BACKGROUND: MEMORY

The evolution of memory technology is a whole different story, but we can see similarities with the processor. DRAM memory has its own Moore's law: It doubles its size (Gbytes) every 18 months. That is like the transistor count for the CPU, but unlike its partner, it does not translate that easy into speed rates. Even worse, the larger a memory circuit becomes, the slower reacts. And the more density holds, the tinier its atomic cell, that is, the capacitor. It takes a while to read typical loads of 25-30 fF, so it is faster to amplify those loads before starting to read. In essence, amplifiers increase latency, but benefit bandwidth. And since latency is unavoidable, we have tried to amortize its cost with higher bandwidths. That has been the fuel propelling SDRAM memories over the past 20 years, with an overall increase of 48x, from the first SDRAM pumped at 66 MHz to provide 533 MB/s, to the last DDR4 at 2x1600 MHz to deliver 25.6 GB/s. In the meantime, CPUs contribute with larger caches and more cache levels to minimize DRAM memory accesses, and also with bigger cache lines to promote bandwidth over latency.

On the other hand, GPUs have introduced wider paths (384 bits), for a GPU using GDDR5 video memory to deliver 336 GB/s bandwidth (see Table VII using Titan X as example). In contrast, a typical CPU operating on a 4 channel motherboard hardly reaches 64 GB/s (see also Table VII for the column of the Intel Broadwell CPU).

Another similarity between memory and processor is the multi-core evolution. DDR (Double Data Rate) chips split memory cells into twin hemispheres sharing latency but doubling bandwidth. The DDR saga exploits this idea with DDR2, DDR3, DDR4 and GDDR5 designs the same way we have seen with dual-cores, quad-cores, octo-cores, ... until we face the truth: It is not an scalable idea. The clones of a single design require more and more infrastructure for the memory chip to be able to coordinate the efforts into a single response, and therefore, latency increases proportionally to the bandwidth increment. Figure 1 summarizes the evolution of memory latency and bandwidth for the DDR family, where we see that latency now represents more than 90% of the time required for a memory read, that 20 years ago was barely 20%.

Life for transistor within the processor was easy thanks to the silicon miracle: Every couple of years, manufacturing nodes were shrinking gates around 30%, and therefore, the atom of a processor became smaller, faster, cheaper and more power-efficient on 32, 22, 14 nm... But for the memory, capacitors did not help much, because is the delay for amplifying loads what really matters. Interfaces came to rescue with smart ideas for exploiting interleaving, essentially with (1) multi-banks enabled from multiple RAS (Row Access Strobe) signals, and (2) multi-channel deployment for memory modules along the motherboard. Unfortunately, none of these ideas turned out to be scalable either.

Ultimately, the energy cost has killed further developments for a SDRAM standard which has been pushed a long way. Additional layers of circuitry are required to compose and merge a single memory access (typically, the service of a cache line as seen in Figure 1) from all banks, chips, modules and channels involved, and that way, memory cells are placed progressively deeper with respect to the I/O lines where voltage enters the chip, leading to unatractive bandwidth/watt ratios.

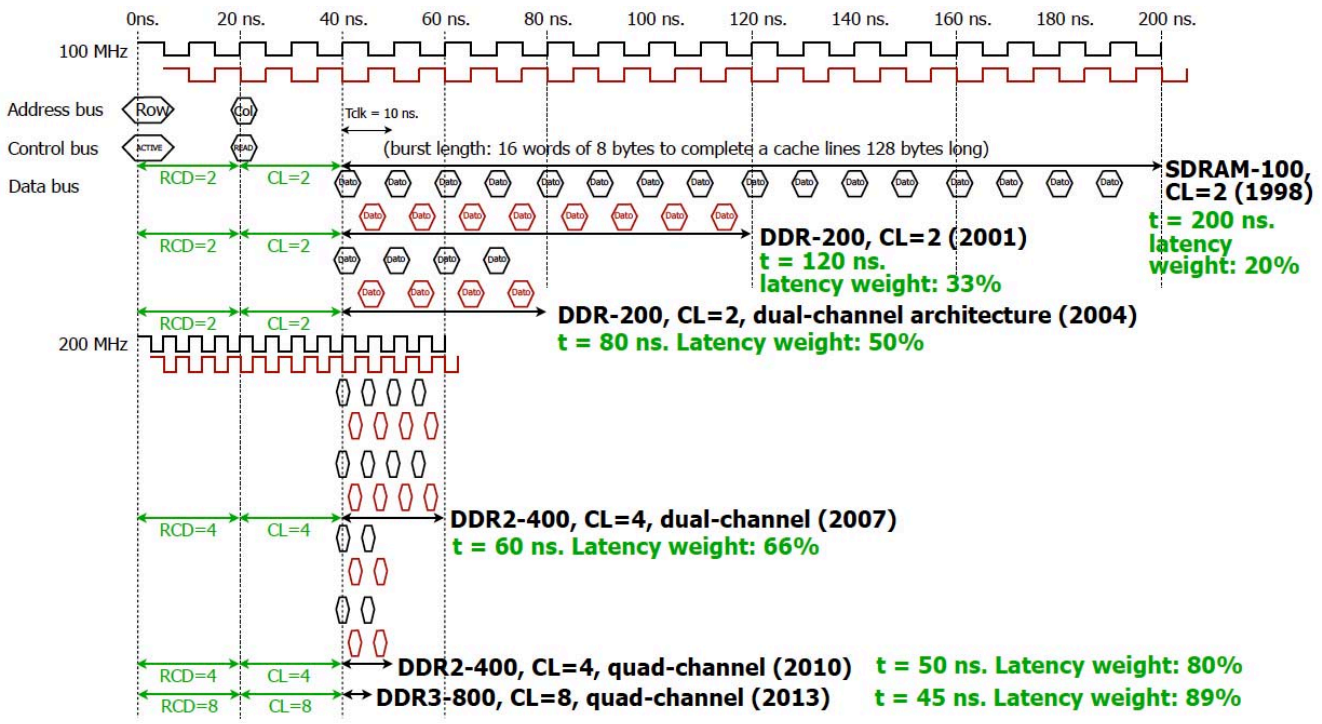
Memories, like processors, are eager to find scalable ideas. 3D-DRAM developments represent a technology able to grow in size and bandwidth at the same time, and without paying so much toll in latency and energy. That is why, like many-core GPUs, 3D-DRAM designs are here to stay and accompany them on a long journey. This paper summarizes how 3D memory works (section III), how HPC accelerators are planning to incorporate it (sections IV, V and VI), and how applications can benefit from such alliance (section VII).

III. 3D MEMORY

Emerging 3D die-stacked DRAM technology represents a chance to solve the memory wall problem on HPC systems [9]. It enables heterogeneous logic dies stacking within one DRAM package and allows the vertical communications among layers with TSVs (Through Silicon Vias) [10].

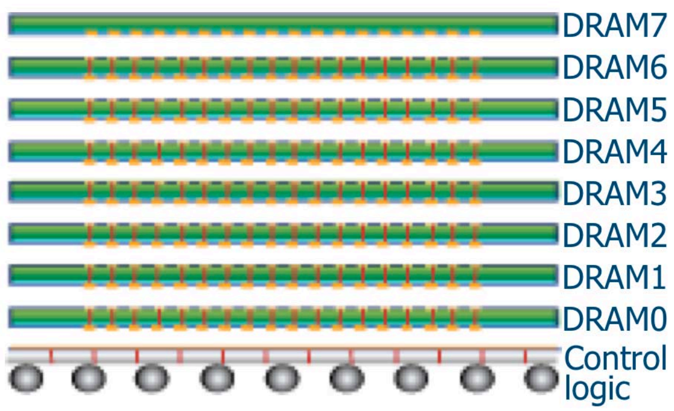
TSVs provide huge internal bandwith because they are dense and fast at the same time. Typical densities are 512, 1024, 2048 and 4096 wires, and latencies are just few picoseconds. To fully utilize this bandwidth, regular DRAM dies are re-partitioned into ranks to build individual memory banks to be stacked in a 3D fashion [11]. That way, we can increase size and speed simultaneously. Figure 2 illustrates the architecture of a 3D die-stacked DRAM for the case of 8 layers with 16 banks each. Banks are grouped into ranks, each traversed with thousands of TSVs.

The 3D die-stacked DRAM also has a separate logic layer to implement the complicated DDR memory controller [12]. The goal is to enable bank-level parallelism to make the bus much wider than on a typical DRAM design where all banks in a rank share a common bus.

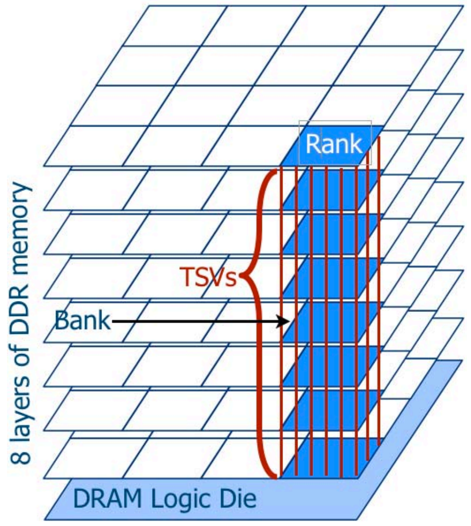


0ns. 20 ns. 40 ns. 60 ns. 80 ns. 100 ns. 120 ns. 140 ns. 160 ns. 180 ns. 200 ns.
100 MHz
Address bus
Control bus
Data bus
Row
Col
ACTIVE
READ
Tclk = 10 ns.
(burst length: 16 words of 8 bytes to complete a cache lines 128 bytes long)
RCD=2
CL=2
Dato
SDRAM-100, CL=2 (1998)
t = 200 ns. latency weight: 20%
DDR-200, CL=2 (2001)
t = 120 ns. latency weight: 33%
DDR-200, CL=2, dual-channel architecture (2004)
t = 80 ns. Latency weight: 50%
200 MHz
RCD=4
CL=4
DDR2-400, CL=4, dual-channel (2007)
t = 60 ns. Latency weight: 66%
DDR2-400, CL=4, quad-channel (2010)
t = 50 ns. Latency weight: 80%
RCD=8
CL=8
DDR3-800, CL=8, quad-channel (2013)
t = 45 ns. Latency weight: 89%

Fig. 1. Time to fill a typical cache line of 128 bytes from SDRAM, established by the JEDEC as the standard memory technology for domestic PCs. Departure point starts in the original SDRAM design, with a CAS Latency (CL) of just 2 cycles @ 100 MHz, and ends with DDR3, where CL represents 8 cycles @ 800 MHz. Latency is drawn in green (see arrows for RAS to CAS Delay, RCD, plus CL) and bandwidth acts during the next arrow in black.



(a) 3D DRAM layers.



(b) Banks, ranks and TSVs.

Fig. 2. The 3D DRAM architecture, where each DRAM die is decomposed into banks, then grouped into ranks on a 3D fashion and traversed via through-silicon vias (TSVs) in a very dense and swift manner.

In general, a die-stacked DRAM has L layers of DRAM dies stacked vertically, and each die implements B banks of DDR memory. Each bank has its own T-bit data TSV I/O. Every L stacked banks compose a 3D vertical rank. Therefore, the overall system consists of B ranks grouped in vertical. All the banks in a 3D vertical rank share a single TSV bus, which can largely relax the TSV pitch constraints [13].

The design can be enhanced by increasing any of these parameters: L (layers), B (banks) or T (bus width), with different results in terms of performance and cost. Table II shows the effect of doubling each of those parameters while leaving the other two unchanged. Numbers correspond to stacked dies of DDR3 memory published in [14].

TABLE II. TYPICAL PERFORMANCE NUMBERS (LATENCY, BANDWIDTH AND ENERGY) FOR A 3D DRAM MEMORY COMPOSED OF 8 GBITS OF DDR3 MEMORY DIES WITH A PAGE SIZE OF 8192 BITS ON A 32 NM. MANUFACTURING PROCESS. THE EFFECT OF DOUBLING LAYERS (L), BANKS (B) AND TSVS (T) IS SHOWN SEPARATELY FOR A BASELINE DESIGN COMPOSED OF 4 LAYERS, 8 BANKS PER LAYER AND 512 TSVS PER BANK. SOURCE: REFERENCE [14].

Memory architecture	Number of layers (L)	Number of banks (B)	Number of TSVs (T)	Time for a bank precharge	RAS to CAS Delay (RCD)	CAS Latency (CL)	TSV latency	Area (on-die) efficiency	Data bandwidth	Power consumption
Baseline	4	8	512	16.8 ns.	7.9 ns.	12.7 ns.	0.67 ns.	0.517	40.4 GB/s.	6.2 W.
Doubling layers	8	8	512	9.6 ns.	8.3 ns.	12.5 ns.	1.91 ns.	0.434	40.8 GB/s.	13.2 W.
Doubling banks	4	16	512	8.4 ns.	7.2 ns.	10.6 ns.	0.67 ns.	0.438	96.7 GB/s.	13.1 W.
Doubling TSVs	4	8	1024	16.8 ns.	7.9 ns.	12.1 ns.	0.67 ns.	0.445	84.4 GB/s.	10.5 W.

The baseline design reflects how fast TSVs are compared to regular DDR3 memory. We have sorted latencies from less to more influencial in the total time required to fill a typical cache line composed of 128 bytes (1024 bits). A standard DDR technology answers as shown in Figure 1. The first latency, bank precharge, only counts when a memory access changes the bank where the previous access was placed, which is never true for the set of consecutive accesses required to fill the cache line. The second latency, RAS to CAS Delay (RCD), is the time required to warm-up a row, that is, moving the page of 8192 bits from capacitors to sense amplifiers, where data can be accessed more quickly. The CAS Latency (CL), is the time spent to move data between sense amplifiers and TSV bus. And finally, the TSV latency, is the vertical data transfer from banks to the memory controller underneath. In our case, a row (also called *pag*e) of DDR3 memory contains 8 cache lines, and therefore, a single RCD is enough to obtain all consecutive accesses required to fill a cache line. Additionally, each of these accesses requires the time characterized by CL. However, once the first data has been answered with a CL, all remaining ones benefit from the pipelined DDR designs to consume just the cycle time (the inverse of the frequency for the memory chip, also including the 2x factor for DDR).

The cronograms in Figure 1 clarify this process. We can see that bandwidth has improved at the same rate that the actual frequency for the memory modules, but RCD and CL latencies remained constant in 20 ns. for almost 20 years:

- 2 cycles of a 100 MHz clock for SDRAM-100 in 1998.
- 2 cycles of a 100 MHz clock for DDR-200 in 2001.
- 4 cycles of a 200 MHz clock for DDR2-400 in 2007.
- 8 cycles of a 400 MHz clock for DDR3-800 in 2013.

On typical DDR3 memory modules, where data width is 64 bits, 16 accesses are required to fill a CPU cache line 128 bytes long. When the motherboard is endowed with a dual-channel architecture, only 8 accesses are required, and similarly, 4 memory accesses are enough for a quad-channel. Finally, on a 3D-DRAM infrastructure containing 512 TSVs, 2 accesses suffice. This way, we trade cycle times by TSV latencies. And at the same time, we are building an infrastructure which can hold much larger memories, and that means longevity.

From this departure point, a number of optimizations can be conducted. Table II reflects that the increase of layers, L, does not benefit bandwidth, whereas B and T do, but at the cost of area overhead. In general, the most rewarding alternative is to increase the number of memory banks, but still, CL predominates. CL can be decreased by further folding the subarrays of one bank to reduce the wire lengths between the sense amplifiers and the TSV bus [15], but this forces to break the structure within a bank, which deteriorates the DRAM density and area efficiency [11]. On the other hand, RCD and CL are for intra-layer DDR operations, which can be overlapped from the memory controller to reduce the TSV bus idle time, thus improving throughput.

IV. NVIDIA GPUS

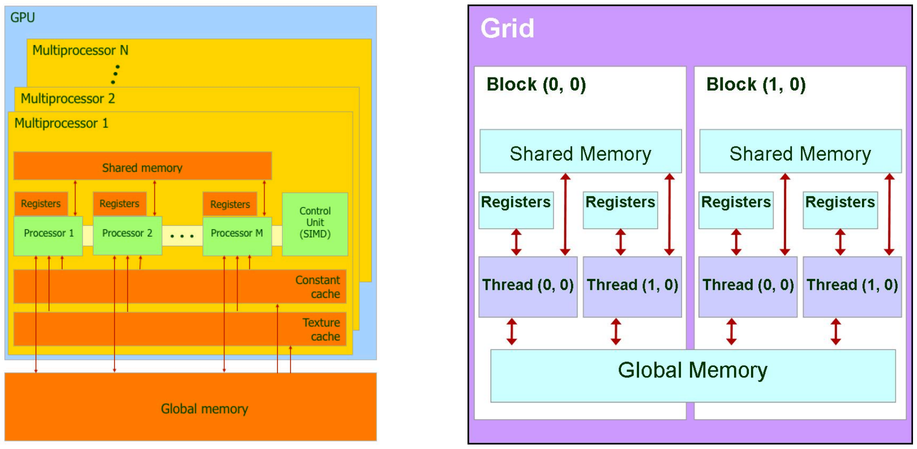
All the latest Nvidia developments on graphics hardware are CUDA-enabled processors: For low-end users and gamers, we have the GeForce series starting from its 8^{th} generation; for high-end users and professionals, the Quadro series; for general-purpose computing, the Tesla boards; finally, for low-power devices, the Tegra family. Overall, it is estimated to exist more than six 600 million CUDA-enabled GPUs in 2016.

Table III summarizes all the essential parameters for each GPU generation since CUDA was born, where we have chosen the most popular GeForce model to represent each generation. The first generation was named Tesla, and had two different architectures, the original G80 and the subsequent GT200. The second generation, Fermi, introduced caches and double precision for floating-point arithmetic. The third generation, Kepler, incorporated additional support for irregular computing, like Hyper-Q and dynamic parallelism. The fourth generation, Maxwell, reorganized cores to optimize energy and introduced unified memory. Finally, the fifth generation, Pascal, consolidates unified memory and introduces 3D DRAM.

CUDA architectures are organized into multiprocessors, each having a number of cores (see Figure 3.a). As technology evolves, future architectures will support the same CUDA executable, but they will run faster for including more multiprocessors per die, and/or more cores, registers or shared memory per multiprocessor. That is the recipe for scalability.

For example, the Pascal parallel architecture is endowed with 2560 cores in the GeForce GTX 1080 GPU. Cores are organized into 40 multiprocessors, each having a large set of 65536 registers, 64 KB shared memory (both 32 bits wide), and constants and texture caches of a few kilobytes. Each multiprocessor can run a variable number of threads, and the local resources are divided among them. In any given cycle, each core in a multiprocessor executes the same instruction on different data based on its `threadId`, and communication between multiprocessors is performed through global memory.

On the programming side, scalability is attained by declaring CUDA blocks for each kernel launched on the GPU. Blocks are mapped to multiprocessors, and threads within blocks are mapped to cores within multiprocessors. That way, when the number of blocks is high enough, the workload is balanced among multiprocessors and additional number of multiprocessors decrease proportionally the execution time. Similarly, threads within a block are mapped to cores within a multiprocessor and executed in a time shared fashion.



(a) CUDA hardware resources.

(b) CUDA programming model.

Fig. 3. The CUDA paradigm. The hardware consists of (a) a number of twin multiprocessors, which are fed from (b) CUDA blocks declared by the programmer.

TABLE III. THE EVOLUTION OF CUDA HARDWARE AND PROGRAMMING CONSTRAINTS OVER THE FIRST FIVE GENERATIONS COVERING LAST DECADE, 2007-2016. THAT IS: TESLA (WITH TWO REPRESENTATIVE MODELS, THE INAUGURAL G80 AND ITS SEQUEL, GT200), FERMI, KEPLER, MAXWELL AND PASCAL. WE TAKE AS FLAGSHIP FOR EACH GENERATION WHAT WE CONSIDER THE MOST POPULAR GPU, TOGETHER WITH A REPRESENTATIVE VIDEO MEMORY IN GRAPHICS CARDS AT THAT TIME. ALL GRAPHICS CARDS SHARE A SIMILAR COST, AROUND $400 AT LAUNCHING DATE (PEAK PRICE).

	GPU architecture GeForce reference model	**G80 (Tesla)** **8800 GTX** (2006)	**GT 200 (Tesla)** **GTX 280** (2008)	**GF 100 (Fermi)** **GTX 480** (2010)	**GK104 (Kepler)** **GTX 680** (2012)	**GM204 (Maxwell)** **GTX 980** (2014)	**GP104 (Pascal)** **GTX 1080** (2016)
Manufacturing process	Transistors	681M @ 90 nm	1.4BM @ 65 nm	3B @ 40 nm	3.54B @ 28 nm	5.2B @28 nm	7.2B @ 16nm
	Thermal D.P. (die size)	145 W	244 W	183 W	195 W (294 mm^2)	165 W (398 mm^2)	180 W (314 mm^2)
Processors	Multiprocessors (SMs)	16	30	15	8	16	40
	Cores / Multiprocesor	8	8	32	192	128	64
	Total number of cores	128	240	480	1536	2048	2560
	Cores clock (w. Boost)	1.35 GHz	1.30 GHz	1.40 GHz	1006 MHz (1058)	1126 MHz (1216)	1607 MHz (1733)
SRAM memory	32-bit registers / SM	8192	16384	32768	65536	65536	
	Shared memory / SM	16 KB		16 or 48 KB	16, 32 or 48 KB	64 KB	
	L1 cache / SM	none		48 or 16 KB	48, 32 or 16 KB	integrated with texture cache	
	L2 cache / GPU	none		768 KB	512 KB	2048 KB	
DRAM memory	Global memory	GDDR3	GDDR5	GDDR5	GDDR5	GDDR5	GDDR5X
	Memory clock	2 x 900 MHz	2 x 1107 MHz	4 x 924 MHz	2 x 3000 MHz	2 x 3500 MHz	4 x 2500 MHz
	Bus memory width	384 bits	512 bits	384 bits	256 bits	256 bits	256 bits
	Memory bandwidth	86.4 GB/s	141.7 GB/s	177.4 GB/s	192.2 GB/s	224 GB/s	320 GB/s
Programming constraints	CUDA compute capabil.	1.0	1.3	2.0	3.0	5.2	6.0
	Active blocks / SM	8	8	8	16	32	32
	Threads / block	512	512	1024	1024	1024	1024
	Threads / SM	768	1024	1536	2048	2048	2048

The CPU host and the GPU device maintain their own DRAM and address space, referred to as *host memory* and *device memory* (on-board memory). The latter can be of three different types. From inner to outer, we have *constant memory*, *texture memory* and *global memory*. They all can be read from or written to by the host and are persistent through the life of the application. Global memory is the actual on-board video memory, now moving from GDDR5 to HBM2 3D memory.

Multiprocessors have on-chip memory that can be of two types: *registers* and *shared memory* (see Figure 3.b). Each processor has its own set of local 32-bit read-write *registers*, whereas a parallel data cache of *shared memory* is shared by all the processors within the same multiprocessor.

Since the introduction of unified memory in Maxwell, programmer may choose to allocate memory together for the CPU and GPU spaces. The driver is responsible for migrating pages back and forth between main and video memory according to access patterns to maximize likelihood for a given processor to find its data inside its closer DRAM memory. Then Pascal provides hardware support for a joint access to unified memory, either from CPU or GPU. Obviously, that goal is easier when counting on 3D memory, and, in addition, a page migration engine has been included to complete the transition phase.

V. INTEL XEON PHI

The Xeon Phi story is quite hard to summarize, starting with Larrabee back in 2008 and changing names into MIC architecture to release Knights Ferry (KNF), Knights Corner (KNC) and finally Knights Landing (KNL).

KNF was prototyped under 45 nm. lithography using a PCI-e 2.0 card endowed with 32 cores, 8 MB of L2 cache and 2 GB of GDDR5 memory, but the product was cancelled without being commercialized. KNC entered the market in 2013, and was manufactured on 22 nm. using tri-gate 3D transistors (as the Ivy Bridge saga) to reach 1 TFLOPS in double precision. KNL represents the third generation of Xeon Phi Processors, launched in June, 2016. It has been manufactured on 14 nm. lithography, like the Broadwell Xeon E5 and E7 processors, to include more than 8 billion transistors, the largest chip that Intel has made so far.

TABLE IV. TESLA ACCELERATORS FOR THE LAST THREE GENERATIONS AT NVIDIA CHARACTERIZED BY THE MOST REPRESENTATIVE GPU MODEL.

		Models with 2D Memory		**Models with 3D Memory**	
	Codename (generation)	**GK110 (Kepler)**	**GM200 (Maxwell)**	**GP100 (Pascal)**	
	Commercial name	**Tesla K40**	**Tesla M40**	**Tesla P100 (with NV-Link)**	**Tesla P100 (with PCI-e)**
Manufacturing	Year released	2012	2014	2016	
	Lithography	28 nm.	28 nm.	16 nm. FinFET	
	GPU die size	551 mm^2	601 mm^2	610 mm^2	
	Number of transistors	7.1 billion	8 billion	15.3 billion	
	Thermal Design Power (TDP)	235 W.	250 W.	300 W.	250 W.
Processor	Number of Multiprocessors	15	24	56	
	FP32 CUDA Cores / Multip.	192	128	64	
	FP32 CUDA Cores / GPU	2880	3072	3584	
	FP64 CUDA Cores / Multip.	64	4	32	
	FP64 CUDA Cores / GPU	960 (1/3 FP32)	96 (1/32 FP32)	1792 (1/2 FP32)	
	Base clock	745 MHz	948 MHz	1328 MHz	1126 MHz
	GPU Boost clock	810 / 875 MHz	1114 MHz	1480 MHz	1303 MHz
	Peak performance (FP64)	1680 GFLOPS	213 GFLOPS	5304 GFLOPS	4670 GFLOPS
Memory	Register File Size / Multip.	64 Kregs.	64 Kregs.	64 Kregs.	
	Register File Size / GPU	960 Kregs.	1536 Kregs.	3584 Kregs.	
	Shared Memory / Multip.	48 KB	96 KB	64 KB	
	L2 Cache Size	1536 KB	3072 KB	4096 KB	
	Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2	3072-bit HBM2 (12 GB) 4096-bit HBM2 (16 GB)
	Memory size	Up to 12 GB	Up to 24 GB	16 GB	12 or 16 GB
	Memory bandwidth	288 GB/s	288 GB/s	720 GB/s	540 GB/s (12 GB) 720 GB/s (16 GB)

TABLE V. XEON PHI PRODUCTS COMMERCIALLY AVAILABLE.

	Knights Corner (KNC)			**Knights Landing (KNL)**			
Model	**3120P / 3120A**	**5120D / 5110P**	**7120X / 7120P / 7120D / 7120A**	**7210**	**7230**	**7250**	**7290**
Release date	Q2'13 / Q2'13	Q2'13 / Q4'12	Q2'13 / Q2'13 / Q1'14 / Q2'14	Q2'16	Q2'16	Q2'16	Q3'16
Lithography	22 nm.	22 nm.	22 nm.	14 nm.	14 nm.	14 nm.	14 nm.
Cost	$1695	$2759 / $2649	$4129 / $4129 / $4235 / $4235	$2438	$3710	$4876	$6254
Cost per TFLOPS	$1695	$2732 / $2623	$3412	$916	$1393	$1601	$1810
Processor							
Clock speed	1.10 GHz	1.05 GHz	1.24 GHz	1.3 GHz	1.3 GHz	1.4 GHz	1.5 GHz
Cores & threads	57	60	61	64 & 256	64 & 256	68 & 272	72 & 288
Peak perf. (FP64)	1.0 TFLOPS	1.01 TFLOPS	1.21 TFLOPS	2.66 TFLOPS	2.66 TFLOPS	3.05 TFLOPS	3.46 TFLOPS
Thermal D. P.	300 W.	245 / 225	300 / 300 / 270 / 300	215	215	215	245
L2 cache	28.5 MB.	30 MB.	30.5 MB.	32 MB.	32 MB.	34 MB.	36 MB
DRAM							
Size and type	6 GB. GDDR5	8 GB. GDDR5	16 GB. GDDR5	Up to 384 GB DDR4 (to be purchased separately)			
Max. clock speed	2x 2.5 GHz	2x2.75 GHz / 2x2.5 GHz	2x2.75 GHz	2133 MHz	2400 MHz	2400 MHz	2400 MHz
Bus width	384 bits	512 bits	512 bits	384 bits	384 bits	384 bits	384 bits
Bandwidth	240 GB/s	352 GB/s / 320 GB/s	352 GB/s	102 GB/s	115.2 GB/s	115.2 GB/s	115.2 GB/s
3D Memory							
Size and type	do not include 3D Memory			16 GB MCDRAM (included in all models)			
Bandwidth / pin				6.4 GT/s.	7.2 GT/s.	7.2 GT/s.	7.2 GT/s.
Bandwidth				355 GB/s.	400 GB/s.	400 GB/s.	400 GB/s.

Table V summarizes all specifications for the KNC and KNL products available. The first Xeon Phi products based on KNL are the 7200 Series, introducing three major differences with respect to its predecessor KNC. First, instead of a co-processor, it is a stand-alone directly bootable infrastructure. Second, rather than the Pentium 54C cores used in Knights corner, cores in KNL are based on a heavily modified *Silvermont* version of the Atom processor that can execute four threads per core and is around 3x the single-threaded performance of P54C cores. Third, cores are organized into 8 octants of high bandwidth stacked MCDRAM (Multi Channel DRAM), which scales up to 16 GB of capacity (2 GB per octant).

The performance jump compared to KNC coprocessors ranges between 2.6x and 2.9x, with the price rising by 1.4x-1.5x. Cores are tiled in pairs, with each core having two AVX512 vector processing units (the multimedia extensions) and 1 MB of L2 cache shared across the tile. Tiles are linked to each other using a 2D mesh interconnect, which also hooks into the six DDR4 memory controllers that feed into what is called *far memory*. Far memory scales up to 384 GB capacity and delivers around 90 GB/s on the STREAM Triad memory benchmark test. On the other hand, *near memory* is used for implementing MCDRAM to reach 400 GB/s. The concepts of near and far memory are explained in the HMC specification (see section VI), but anyway, keep in mind that near memory is integrated within the processor, whereas far memory has to be purchased separately in DIMM modules.

KNL offers 3 modes for memory addressing, all selected at booting time:

1) **Flat**, where DDR4 and MCDRAM are combined into a NUMA single address space. An API is provided to allow programmers to explicitly select MCDRAM using ”Fast Malloc” functions (hbw_malloc/hbw_free).
2) **Cache**, where MCDRAM acts as a L3 cache for the DRAM, with the hierarchy managed by the system.
3) **Hybrid**, where MCDRAM is partitioned into two chunks, each of them used as Flat and Cache modes.

Subsequent versions of Xeon Phi are coming out in October, 2016, with integrated dual-port, 100 Gb/s Omni-Path.

Adding those links to the chip package boost the price by $278 and raises energy by 15 watts. Later, Intel plans to release the co-processor version based on PCI-express 3.0 cards.

VI. 3D Memory Consortiums: HMC versus HBM

Pioneer research projects about Stacked DRAM were developed during the 2003-2006 period. The first commercial announcement of the technology was performed by Tezzaron Semiconductors back in January, 2005. Now the company has entered into its 4^{th} generation of products, the DiRAM4 [16], reaching a bandwidth of 1 TB/s. with a latency of 9 ns.

From that on, several manufacturing lines have been set up, often oriented to other segments like PDAs or smartphones. A good example is Samsung's Wide I/O, designed to provide SoCs with maximum bandwidth at minimum power. This section focuses on the two consortiums having as target the HPC arena: HMC and HBM.

A. Hybrid Memory Cube (HMC)

In October, 2011, the HMC Consortium was founded by Micron Technologies and Samsung Electronics, and soon a long list of companies signed as adopted members. Among them, we may cite Microsoft, Altera, ARM, Cray, HP, IBM, GlobalFoundries, Xilinx and SK Hynix. The specification for HMC 1.0 was available in April, 2013, with production samples based on the standard finished during the second half of 2014. The consortium claims that 400 GB/s of bandwidth are possible via HMC, with production expected in late 2016.

HMC is designed for high-end servers to respond to multi-core scenarios and deliver data with much higher bandwidth and lower latency. Primary goals are to strip out the duplicative control logic of modern DIMMs, simplify the design, connect the entire stack in a 3D configuration, then use a single control logic layer to handle all read/write traffic. Major drawbacks are cost and power consumption. HMC is also dependant on a number of profound improvements to semiconductor manufacturing, and it is not a JEDEC standard.

Intel's MCDRAM memory is a variant of HMC that has a proprietary interconnect between the processor interconnect and that memory. That is why you do not see the company officially into the consortium. But the concepts of near memory and far memory that we find within the HMC specification are used by Intel to explain its implementations of MCDRAM and DDR4 memory controllers, respectively.

B. High Bandwidth Memory (HBM)

The development of HBM started at AMD in 2008 to reduce the increasing power consumption and form factor of the DDR saga. Soon partners from the memory industry (SK Hynix), interposer industry (UMC) and packaging industry (Amkor, ASE) joined the specification. HBM was adopted as industry standard by JEDEC in October, 2012 following a proposal by AMD and SK Hynix in 2010. Nvidia joined later in 2014. AMD used HBM 1.0 chips for its Fiji GPU, released during the summer of 2015 as a Radeon R9 Fury X model. Nvidia uses HBM2 for Pascal, available in early 2017, with the GPU integrated by TSMC using FinFET 16 nm. transistors and the memory cubes manufactured by Samsung.

HBM is explicitly designed with a very wide bus for graphics and HPC GPU environments. It may not reach the bandwidth of HMC, but should be cheaper and more power efficient. Nvidia claims bandwidth over 1 TB/s. at lower latencies, 2.5x higher sizes and four times more energy efficient.

HBM uses 128-bit wide layers and stacks up to eight of them for a 1024-bit interface. Each memory controller is independently timed and controlled. Table VI summarizes all major features of HBM and compares it with the HMC consortium and also with the existing DDR3/4 saga.

Figure 4 shows the GP100 GPU on the printed circuit board. Memory is structured into four cubes, each composed of four layers. The GPU is separated from its cubes in what it is called a 2.5D memory. The challenge here is to move massive amounts of data coming down from the HBM cubes into the horizontal plane all the way to the GPU. The silicon interposer, which is composed exclusively of wires, integrates the required interconnection density. That way, the package substrate only manages externally the traffic that goes along NV-Link or PCI-express to meet the host processor, and a interconnection hierarchy is implemented. The heatsink extends along the four cubes and the GPU, all placed at the same height, but a challenge remains with heat on intermediate layers when its number increases. Table VII summarizes the HBM features and compares them against typical main memory for CPUs (DDR3) and existing video memory for GPUs (GDDR5).

VII. Performance analysis

We use the roofline model [17] for a comprehensive performance analysis of accelerators endowed with 3D memory. The roofline sets an upper bound on performance (GFLOPS, drawn in the vertical axis) for an application depending on its operational intensity (FLOP/byte, drawn in the horizontal axis). Think of the operational intensity for an application as a column hitting the roof to score a GFLOPS mark as the expected performance attained on that platform. Then, when it hits the roof on the flat part, the application is compute-bound; otherwise, it is memory-bound.

Figure 5.a represents this model for three Nvidia GPUs and Intel Xeon Phi models (see specific names and features in Tables IV and V, respectively). Colored thick lines characterize each accelerator, with the leaning line showing the performance of the memory system and the horizontal line showing the peak performance for computational units. Vertical thin lines in black represent four well-known scientific kernels, three of them are memory-bound and one is compute-bound (typically, around 70% of scientific codes are memory-bound). All of them are characterized by its operational intensity, and they all score different performance in GFLOPS on each accelerator. Vertical dotted and colored lines represent the borderline between the memory-bound and the compute-bound regions for each platform, that is, the point when extra bandwidth does not translate into additional performance because all ALUs and FPUs are fully utilized.

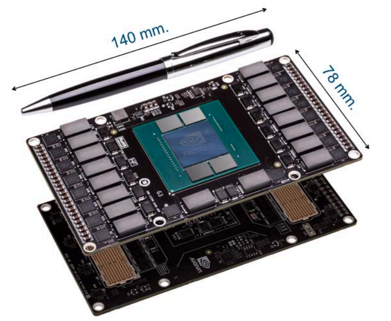
We can see that memory enhancements in Pascal benefit 3 of the 4 applications, and for the fourth one, GFLOPS are what really matters. Over the years, scientific applications have struggled against hardware memory constraints by shifting their implementations to the right side of the charts, where the

TABLE VI. THE DDR STANDARD COMPARED TO HMC AND HBM CONSORTIUMS.

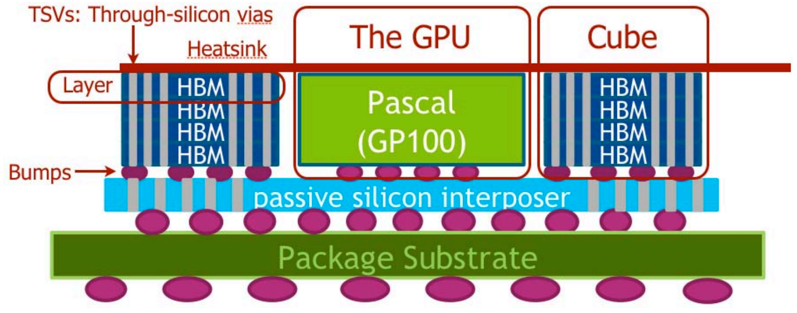
Standard/Consortium	**DDR3 & DDR4**	**HMC**	**HBM**
Target platforms	PCs, laptops, servers	High-end servers and enterprises	GPUs, HPC
Cost	Low	High	Medium
JEDEC standard	Yes	No	Yes
Energy consumption	Medium	High	Low
DRAM interface	Traditional parallel interface, single-ended, bidirectional strobes, separated clocks	Chip to chip SerDes interface	Wide parallel, multi-channel interface, DDR signaling
Voltage	DDR3: 1.5, 1.35, 1.25 v. DDR4: 1.2 v.	1.2 v.	1.2 v.
Width	From 4 bits / chip to 64 bits / module (72 bits when ECC enabled)	16 bidirectional lanes / link, 4 links / cube (in Gen2)	128 bits / channel, 2 channels / layer, 4 layers / cube
Data Rate per pin	DDR3: Up to 2133 Mbps DDR4: Up to 3200 Mbps	10, 12.5 or 15 Gbps /lane 80, 100, 120 GB/s /cube (each way)	Up to 2 Gbps (2x 1GHz DDR clock)
System configuration	PCB based connections, DIMM modules	PCB based, point to point, short reach SerDes interface	2.5D TSV based silicon interposer
Available in market	2008 (DDR3), 2014 (DDR4)	2016 (Gen2)	2015 (HBM1), 2016 (HBM2)
Benefits	- Mature infrastructure and low cost - Low risk - Familiar interface	- High and scalable bandwidth - Power efficiency - PCB connectivity host-DRAM	- High and scalable bandwidth - Power efficiency
Challenges	- Speed no longer scalable - Signal integrity - Customers unprepared for integration.	- Relies on TSVs - Not a JEDEC standard - Cost - PHY IP infrastructure	- Relies on TSVs - Relies on interposer - Cost - PHY IP infrastructure

TABLE VII. THE INCOMING HBM FOR GPUS COMPARED TO EXISTING VIDEO MEMORY (GDDR5) AND TYPICAL MAIN MEMORY ON CPUS (DDR3).

Memory technology	**DDR3**	**GDDR5**	**HBM1**	**HBM2**
Adopted by	Intel CPU motherboards	Existing GPU boards	AMD GPUs in 2015/16	Nvidia GPUs in 2017
Energy consumed	18-22 pJ / bit		6-7 pJ / bit	
Cubes per GPU	does not apply		4	
Prefetching	8 / pin		2 / pin	
Pins for data	8 / chip	32 / chip	2x128	
Access granularity	8 bytes / chip	32 bytes / chip	64 bytes / layer	
Bandwidth	2 GB/s / chip (2 Gbps / pin)	28 GB/s / chip (7 Gbps / pin)	32 GB/s / layer (1 Gbps / pin)	64 GB/s / layer (2 Gbps / pin)
Chips or layers	2-16 chips / module	12 chips / card	4 layers / cube	4, 8 layers / cube
Bandwidth example	Broadwell CPU 2 GB/s x 8 chips x 4 channels = 64 GB/s.	Maxwell Titan X 28 GB/s x 12 chips = 336 GB/s (saturated here)	AMD Fiji GPU 32 GB/s x 4 layers x 4 cubes = 512 GB/s	Pascal models 64 GB/s x 4, 8 layers x 4 cubes = 1, 2 TB/s



(a) A Tesla P100 prototype.

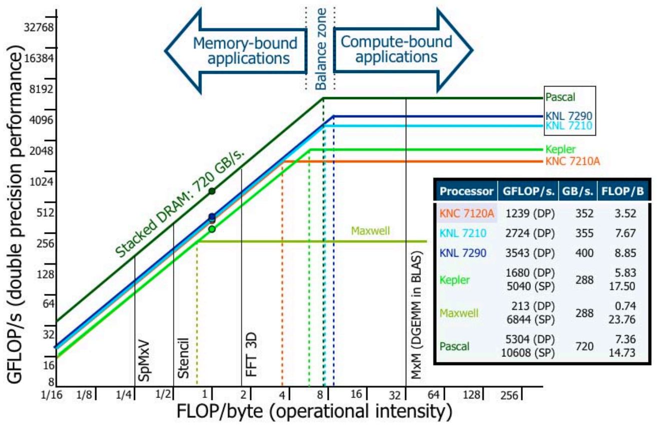


(b) The accelerator outlined sectionally.

Fig. 4. An example of the 3D integration of Pascal architecture: The Tesla P100 based on the GP100 GPU.

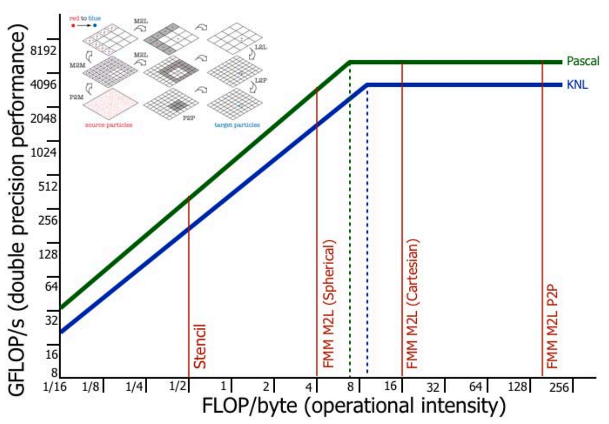
raw computational power (GFLOPS) use to be more generous. That is the case for the Fast Multipole Method (FMM), a numerical technique developed to speed up the calculation of long-ranged forces in the gravitational n-body problem [18]. Figure 5.a shows four implementations chronologically developed from left (older) to right (newer), with successive code optimizations trying to escape from the memory-bound region. But in 2016, thanks to the enhancements produced by accelerators with the advent of 3D memory, a performance gap can be enjoyed without programming effort.

The roofline model also allows you to compare the accelerators performance regardless of the application executed. For example, KNC works well for memory-bound kernels, but trails on compute-bound kernels. On the other side, Maxwell is the worst in compute-bound kernels when double-precision is required. The two KNLs behave better on compute-bound kernels than on memory-bound ones. And Pascal is the leader in both sides, offering a performance gap which is slightly better on the memory-bound region (note that logarithmic scaling makes the gap to appear shorter visually).



Processor	GFLOP/s.	GB/s.	FLOP/B
KNC 7120A	1239 (DP)	352	3.52
KNL 7210	2724 (DP)	355	7.67
KNL 7290	3543 (DP)	400	8.85
Kepler	1680 (DP) 5040 (SP)	288	5.83 17.50
Maxwell	213 (DP) 6844 (SP)	288	0.74 23.76
Pascal	5304 (DP) 10608 (SP)	720	7.36 14.73

(a) Accelerators.



(b) Case study for the Fast Multipole Method (FMM).

Fig. 5. The roofline model for three representative accelerators of Intel and Nvidia and four popular scientific codes: the sparse matrix-vector multiply (SpMxV), a stencil for a 8-points template, the 3D Fast Fourier Transform (FFT), and the Matrix Product on double precision numbers (DGEMM).

VIII. Concluding remarks

When I started programming GPUs 15 years ago, I was fascinated by extraordinary bandwidths, but soon I realized that it was not enough. GPUs have always been two generations ahead of CPUs in terms of DRAM (GDDR3 vs. DDR at that time). And they can afford to dedicate more perimeter to deploy wires (256 vs. 64 bits then). Speed and width merge to combine an advantageous bandwidth (6-10x versus CPUs). GPU boards have managed to keep the leadership in technology and logistics, showing the right way to trailing CPUs. The DDR saga and the multi-channel motherboards are two good lessons that CPUs learnt well.

Now we have entered the heterogeneous computing era, and CPUs and GPUs converge on the same chip. Every year we get closer to the SoC (System on a Chip) approach. Stacked DRAM has taught us that we can grow as modern cities do: Through the third dimension. We have seen a thick GP100 GPU with four layers of DRAM, and CPUs can do the same with caches, that now occupy more than 50% of the silicon die. The ultimate goal would be a 3D chip composed of CPU+GPU in the basement, controllers at the floor level, and multiple memory layers on top. That is all: Computation, control and storage. TSVs are fast, dense, reliable and, soon, cheap, so we have the technology to live on a dreamt skycrapper with plentiful and fast elevators.

The pessimistic side is heat. We are not improving dissipation as much. That reminds me the story about batteries in laptops: Our needs are way beyond what technology can provide us. Memory consortiums have published specifications for 8 layers of Stacked DRAM, few of them even more, but still, we do not see those in commercial products. Because of the heat. We are masters on how to spread heat on a surface and remove it, but it has to be a visible layer. With four layers, you still can stick heatsinks to two of them, but the story turns challenging on 8, 12, 16 layers. TSVs are very dense and can be built on good materials as far as dissipation is concerned. That is something, but not enough to make our dream come true. And at the end, what happened to the CPU once we realized that it was too hot? We got tired of fighting, we learnt to relax frequency, and gave up. More perseverance will be required this time.

Acknowledgment

We thank Nvidia for hardware donation and travelling support under GPU Education Center 2011-16, GPU Research Center 2012-16 and CUDA Fellow 2012-16 Awards. Special thanks to Lorena Barba for her contribution to the roofline model applied to FMM. This work was partially funded by Junta de Andalucia, Project of Excellence, PR12-TIC-1741.

References

[1] *General-Purpose Computation on GPUs*. http://www.gpgpu.org.

[2] *The Top 500 Supercomputers List*. http://www.top500.org.

[3] *The Green 500 Supercomputers List*. http://www.green500.org.

[4] *Intel. Intel Delivers New Architecture for Discovery with Intel Xeon Phi Coprocessors*. https://newsroom.intel.com/news-releases/intel-delivers\-new-architecture-for-discovery-with-intel-xeon-phi-coprocessors

[5] J. Jeffers and J. Reinders. *Intel Xeon Phi Coprocessor High-Performance Programming*. Morgan-Kaufmann, 2013.

[6] *The Hybrid Memory Cube Consortium*. www.hybridmemorycube.org.

[7] R. Fernando and M.J. Kilgard. *The Cg Tutorial. The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley, 2005.

[8] *CUDA Zone*. https://developer.nvidia.com/cuda-zone.

[9] J. Jeddeloh and B. Keeth. *HMC New DRAM Architecture Increases Density and Performance*. VLSI Technology 2012, pages 87-88.

[10] D. Woo, N. Seong, D. Lewis and H. Lee. *An Optimized 3D Stacked Memory Architecture by Exploiting Excessive, High Density TSV Bandwidth*. The 2012 HPCA Conference, pages 1-12.

[11] k. Chen and S. Li. *CACTI-3DD: Architecture-level modeling for 3D die stacked DRAM main memory*. DATE'12 conference, pages 33-38.

[12] G. Loh. *3D-stacked memory architectures for multi-core processors.* Proceedings ISCA'08, pages 453-464.

[13] R. Anigundi, H. Sun, J. Lu, K. Rose and T. Zhang. *Architecture design exploration of 3D integrated DRAM*. Procs. ISQED'09, pages 86-90.

[14] Q. Zhu, B. Akin, H.E. Sumbul, F. Sadi, J.C. Hoe, L. Pileggi, F. Franchetti. *A 3D-Stacked Logic-in-Memory Accelerator for Application-Specific Data Intensive Computing*. Procs. IEEE 3DIC'13, pages 1-7.

[15] D. Woo, N. Seong and H. Lee. *Heterogeneous die stacking of SRAM row cache and 3D DRAM: An empirical design evaluation*. Proceedings MWSCAS'11, pages 1-4.

[16] *DiRAM 3D Memory.*. http://www.tezzaron.com/products/diram4-3d-memory/.

[17] S. Williams, A. Waterman and D. Patterson. *Roofline: an insightful visual performance model for multicore architectures.* Communications of the ACM, vol. 52, no. 4, April, 2009.

[18] L.A. Barba and R. Yokota. *How Will the Fast Multipole Method Fare in the Exascale Era?* SIAM News, Vol. 46, No. 6, 2013.