23rd Summer School on PARALLEL COMPUTING

Introduction to Accelerators

Piero Lanucara – p.lanucara@cineca.it
SuperComputing Applications and Innovation Department

CINECA

Single core

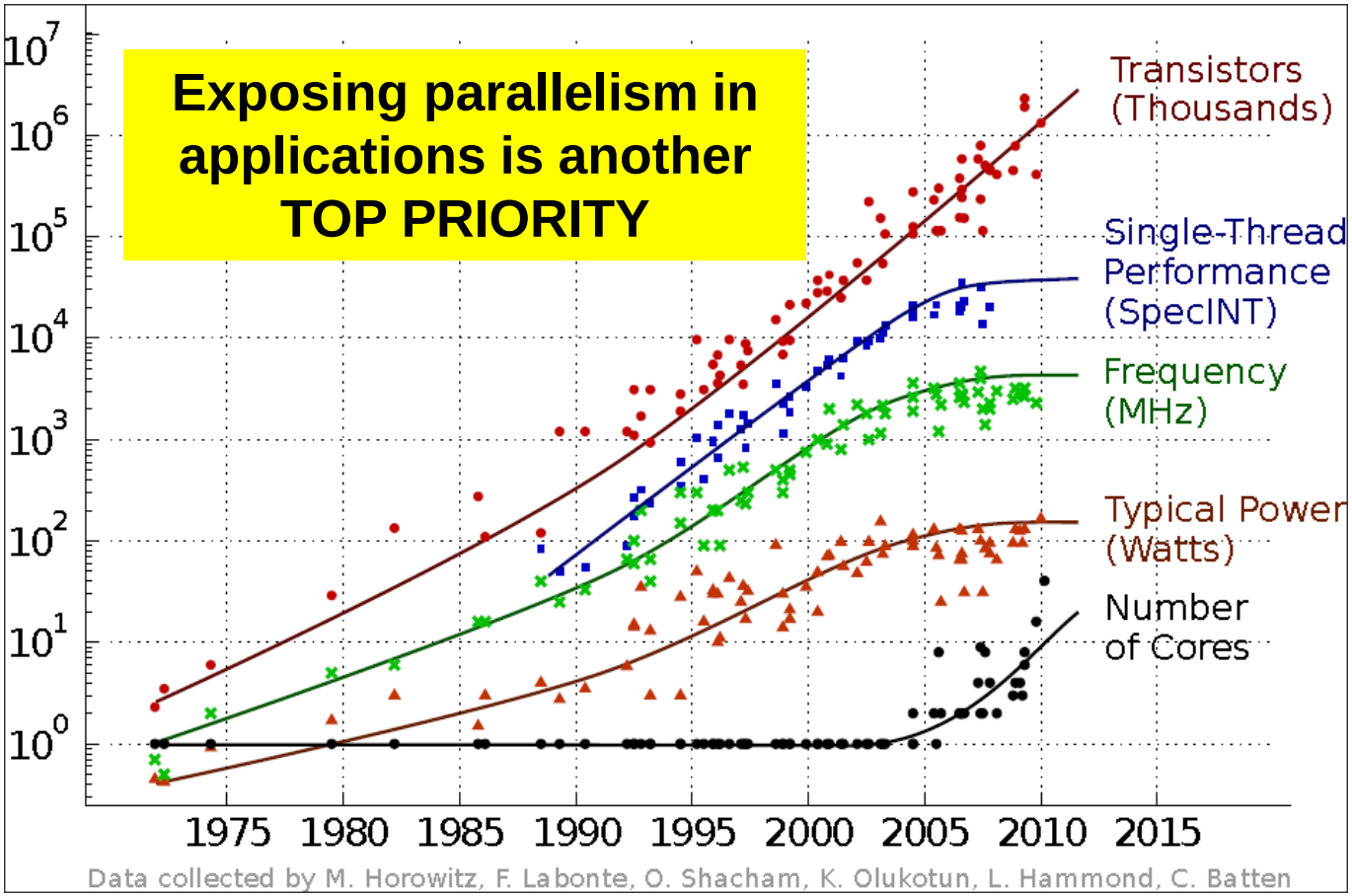
Multi core

Many core

GPU

Intel MIC

35 YEARS OF MICROPROCESSOR TREND DATA

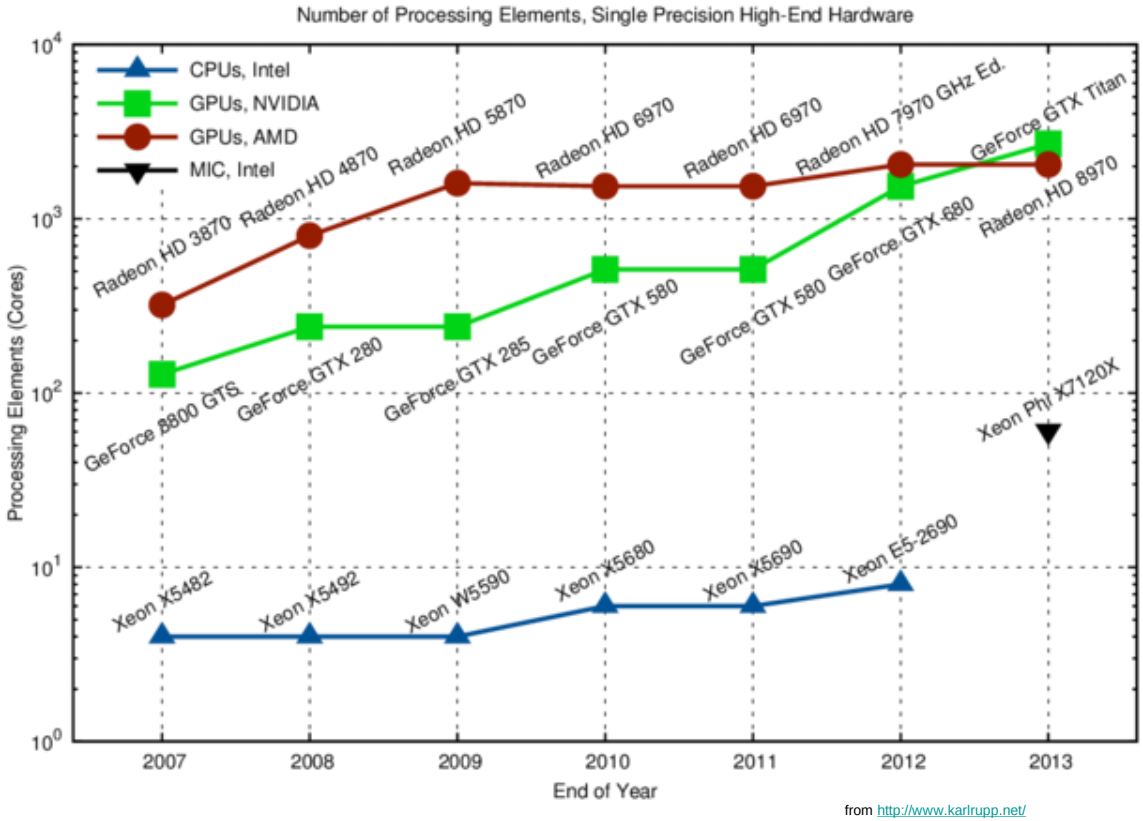


Source: C. Moore's talk at Salishan, April 2011

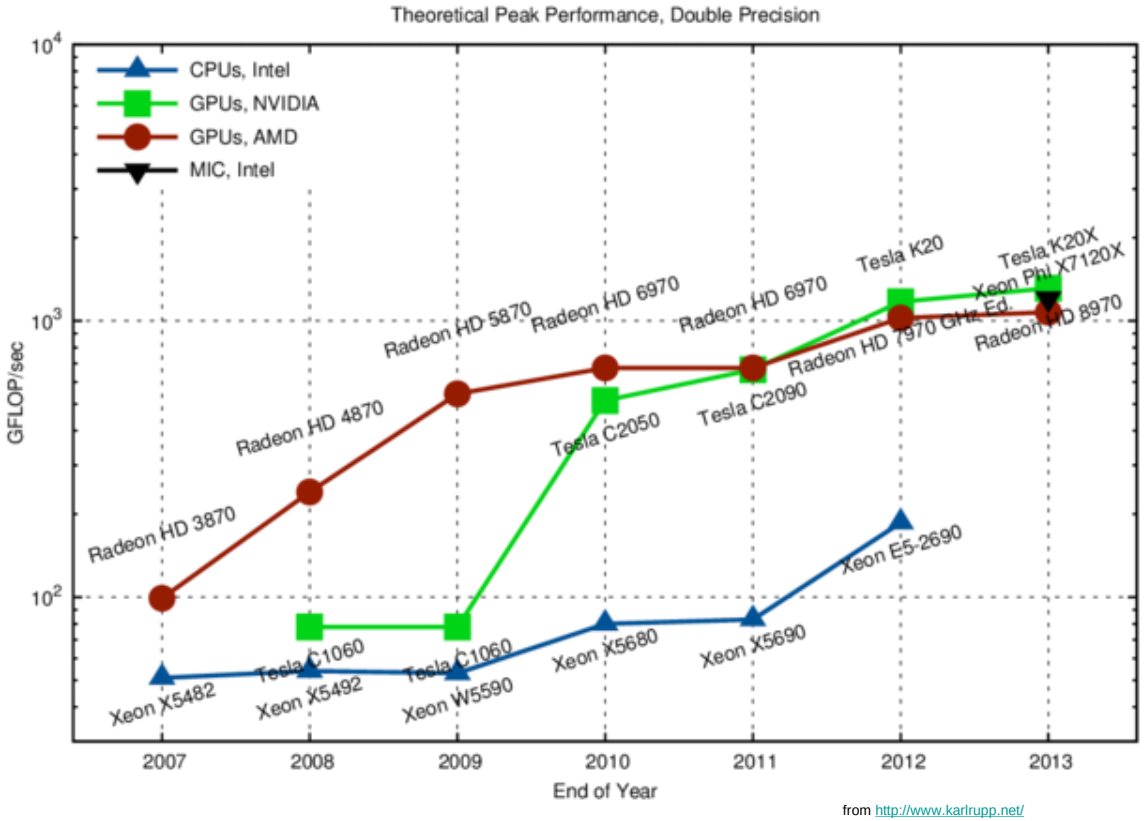
Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115984	6271.0	7788.9	2325
7	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
8	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
9	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
10	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423

THE GREEN 500™

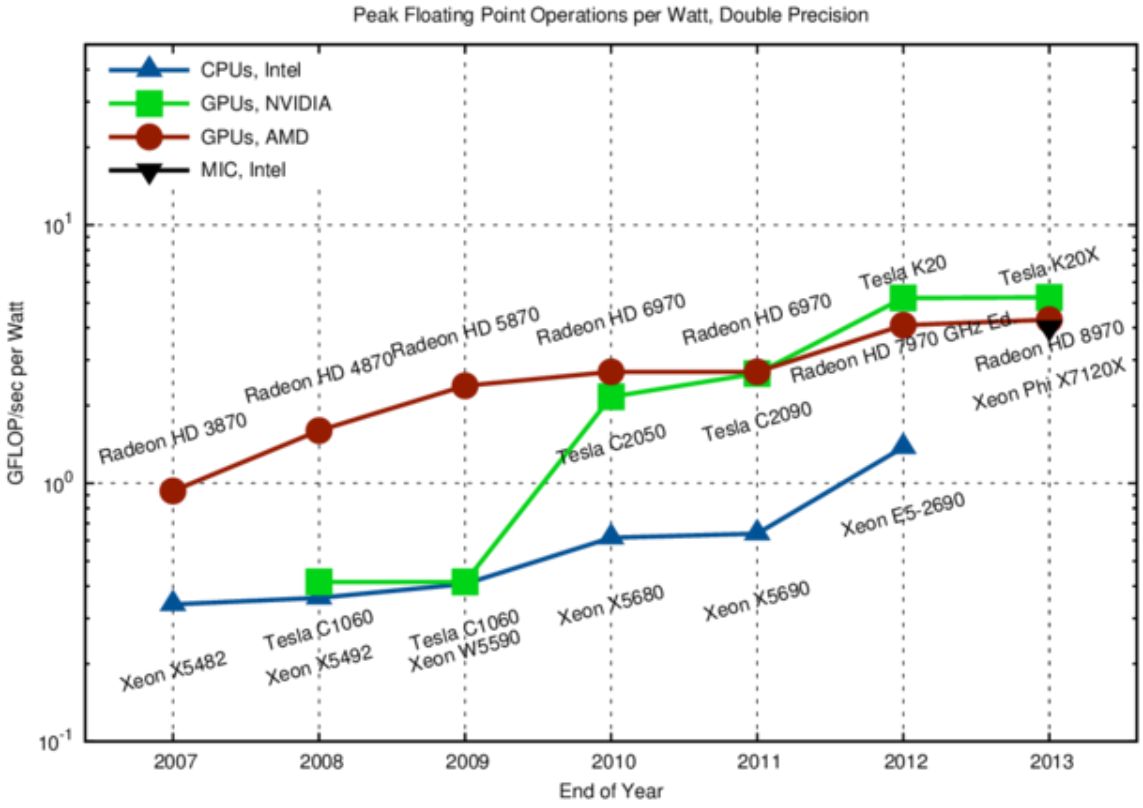
Green500 Rank	MFLOPS/W	Site*	Computer*	Total Power (kW)	
1	**4,503.17**	GSIC Center, Tokyo Institute of Technology	TSUBAME-KFC - LX 1U-4GPU/104Re-1G Cluster, Intel Xeon E5-2620v2 6C 2.100GHz, Infiniband FDR, NVIDIA K20x	27.78	
2	**3,631.86**	Cambridge University	Wilkes - Dell T620 Cluster, Intel Xeon E5-2630v2 6C 2.600GHz, Infiniband FDR, NVIDIA K20	52.62	
3	**3,517.84**	Center for Computational Sciences, University of Tsukuba	HA-PACS TCA - Cray 3623G4-SM Cluster, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband QDR, NVIDIA K20x	78.77	
4	**3,185.91**	Swiss National Supercomputing Centre (CSCS)	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Level 3 measurement data available	1,753.66	
5	**3,130.95**	ROMEO HPC Center - Champagne-Ardenne	romeo - Bull R421-E3 Cluster, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR, NVIDIA K20x	81.41	
6	**3,068.71**	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.5 - Cluster Platform SL390s G7, Xeon X5670 6C 2.930GHz, Infiniband QDR, NVIDIA K20x	922.54	
7	**2,702.16**	University of Arizona	iDataPlex DX360M4, Intel Xeon E5-2650v2 8C 2.600GHz, Infiniband FDR14, NVIDIA K20x	53.62	
8	**2,629.10**	Max-Planck-Gesellschaft MPI/IPP	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	269.94	
9	**2,629.10**	Financial Institution	iDataPlex DX360M4, Intel Xeon E5-2680v2 10C 2.800GHz, Infiniband, NVIDIA K20x	55.62	
10	**2,358.69**	CSIRO	CSIRO GPU Cluster - Nitro G16 3GPU, Xeon E5-2650 8C 2.000GHz, Infiniband FDR, Nvidia K20m	71.01	



from http://www.karlrupp.net/



from http://www.karlrupp.net/

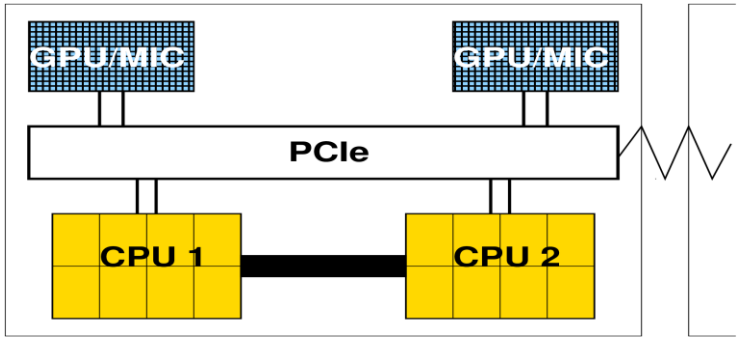


from http://www.karlrupp.net/

GPU and many core computing: a view from the top

Basic principle (today's GPUs, many-core coprocessors):

- accelerator “cards” for standard cluster nodes (PCIe)
- many (~50...500) “lightweight” cores (~ 1 GHz)
- high thread concurrency, fast (local) memories



System architecture:

- currently: x86 “Linux-clusters” with nodes comprising
- 2 CPUs (2x 8 cores)
- max. 2...3 accelerator cards (GPU, MIC) per node
- future: smaller CPU component (extreme: “host-less”,many-core chips)

Programming paradigms:

- use CPU for program control, communication and maximum single-thread performance
- “offload” data-parallel parts of computation to accelerator for maximum throughput performance
- requires heterogeneous programming & load-balancing, careful assessment of “speedups”

Motivation

Compute performance

- GPU/many-core computing is promising huge application-performance gains
- caveat: sustained performance on “real-world”, scientific applications
- observations:
- apparent GPU success stories: PetaFlops performance (Gordon-Bell Price nominations)
- from aggressive marketing for Intel MIC, NVIDIA GPUs…

… towards more realistic attitudes: factor 2x..3x speedups (GPU vs. multi-core CPU)

Energy efficiency

- GPU/many-core computing is promising substantial energy-efficiency gains (a must for exascale)
- caveat: sustained efficiency on “real-world” CPU-GPU clusters

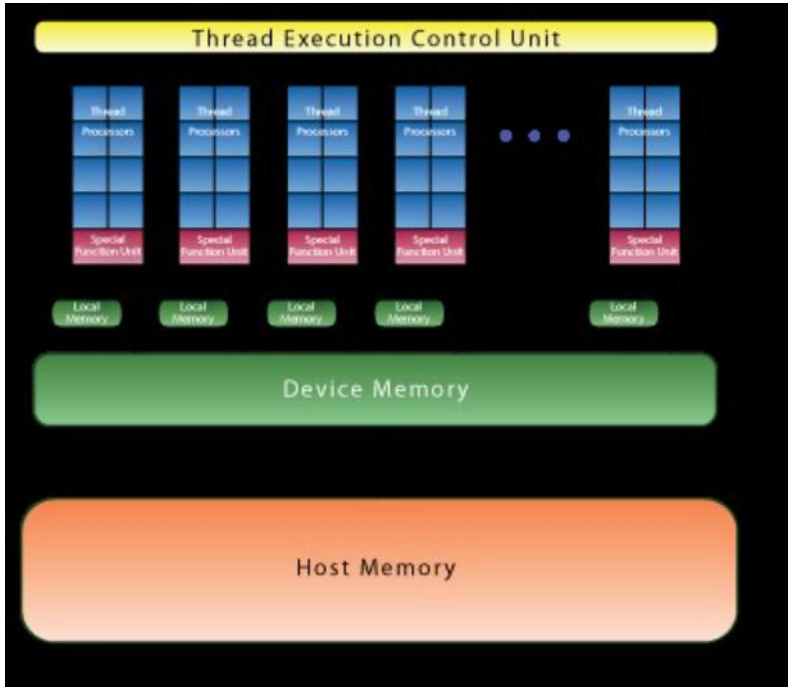
Existing resources

- there is significant GPU/many-core-based compute-power around in the world
- by many, the technology is considered inevitable for the future
- **caveat: the price to pay for application development** ?

NVIDIA GPU TECHNOLOGY

Hardware overview (NVIDIA Tesla series)

- since 2011: “Fermi”: first product with HW support for double-precision and ECC memory
- up to 512 cores, 6 GB RAM
- high internal memory bandwidth ~180 GB/s
- 0.5 TFlops (DP, floating point)
- data exchange with host via PCIe (~8 GB/s)
- enhancements: MPI optimization, intra-node comm. (“GPU direct”, “HyperQ”, ...)



Q1/2013: “Kepler K20”:

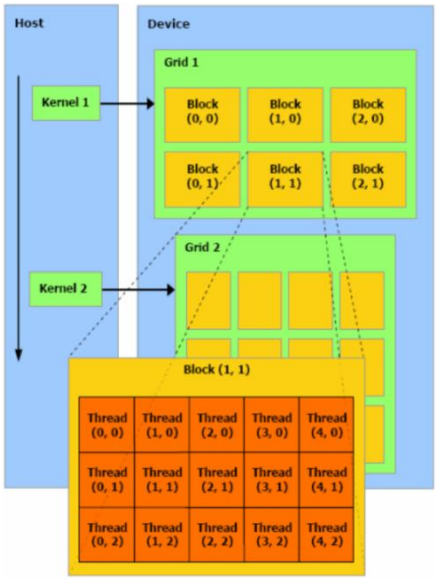
- GK110 GPU: up to 2880 cores, 6...12 GB RAM
- internal memory bandwidth: ~200 GB/s
- nominal peak performance: ~ 1.3 TFlops (DP)

plans for a “hostless” chip (for Exascale)

NVIDIA GPU TECHNOLOGY

Software & programming models

- paradigm: split program into host code (CPU) and device code (GPU)
- GPU hardware architecture requires highly homogeneous program flow (SIMT, no if-branches!)
- PCIe bottleneck for communication of data between CPU and GPU:
- O(n2)...O(n3) computations for communication of n data
- overlapping of communication and computation phases



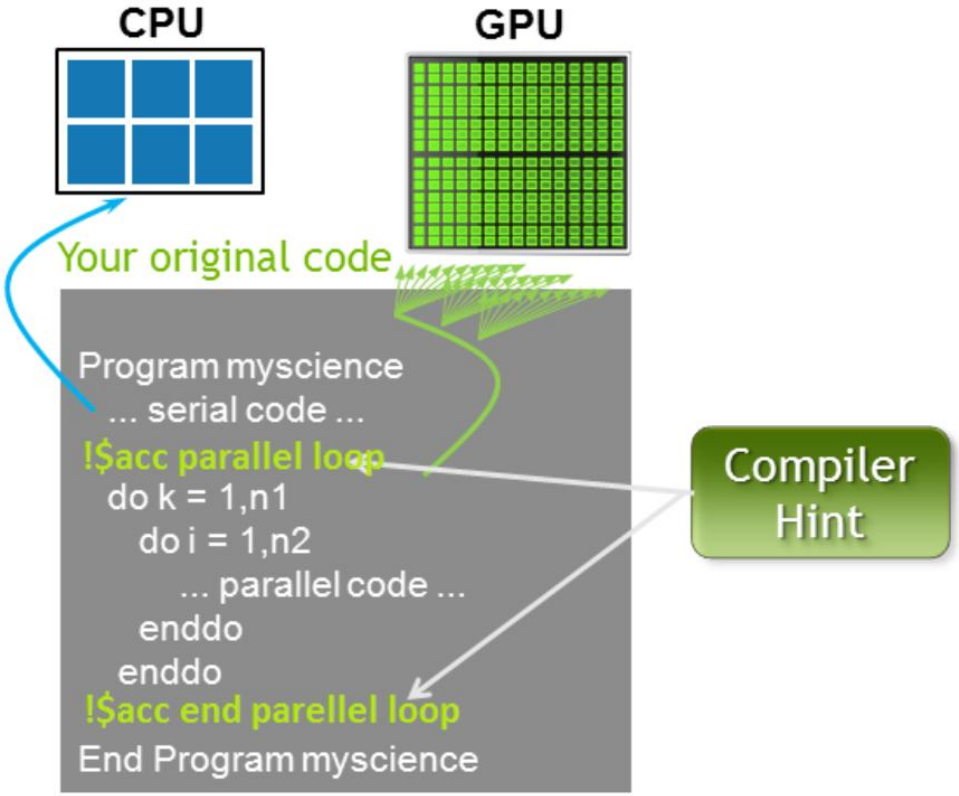
Programming languages

- CUDA (NVIDIA**), OpenCL** (open standard)
- host program (C, executes on CPU) and device kernels (C, launch on GPU)
- numerical libraries: CUBLAS, CUFFT, higher LA: CULA, MAGMA
- tools: debuggers, profiling, system monitoring,…
- CUDA-FORTRAN (PGI)
- directive-based approaches (PGI, CRAY, CAPS, OpenACC, OpenMP-4)
- high-level, comparable to OpenMP
- proprietary (CRAY, PGI, HMPP, ...) → OpenACC → OpenMP

NVIDIA GPU TECHNOLOGY

OpenACC

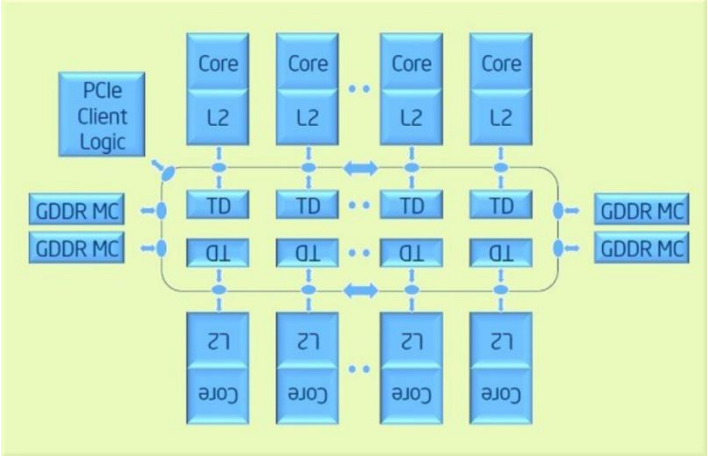
- joint effort of vendors to shortcut/guide OpenMP 4.0 standardization effort
- functional (not performance) portability
- minimally invasive to existing code
- facilitates incremental porting
- compilers: PGI, CRAY, CAPS
- no free lunch!



INTEL MIC TECHNOLOGY

Hardware overview

- since 2011: “Knights Ferry”: software development platform
- Q4/2012: “Knights Corner”: first product of the new Intel Xeon Phi processor line (MIC arch)
- approx 60 x86 cores (~ 1GHz), 8 GB RAM
- internal memory bandwidth: 175 GB/s
- nominal peak performance: 1 TFlops (DP)
- more than a device: runs Linux OS, IP addressable
- data exchange with host via PCIe (~8 GB/s)
- towards a true many-core chip (“Knights Landing”, 2014)



Software & programming models

- paradigms:

1) offload model (like GPU: split program into host code (CPU) and device code (MIC))

2) cluster models (MPI ranks distributed across CPUs and/or MICs)

- tools & libraries: the familiar Intel tool chain: compilers, MPI/OpenMP, MKL, ...
- syntax: “data offload” directives + OpenMP (and/or MPI)
- **OpenCL**

Similarities between GPU and MIC

GPU	MIC
CUDA OpenACC OpenCL	MPI+OpenMP offload directives OpenCL

Both are devices connected through a PCIe to a CPU (=bottleneck in bandwith).
Both have a large number of computing units.

Similarities between GPU and MIC

1. get the data on the GPU/MIC and keep it there

2. give the GPU/MIC enough work to do

3. reuse and locate data to avoid global memory bandwith bottlenecks

caveat: not always true... Not always possible..
What is the trasde-off between performance and effort?

Differences between GPU and MIC

Both GPU and MIC utilize a **large number of concurrent *threads* of execution** to achieve high performances….but….

…programming multiple GPU and MIC coprocessor threads in parallel can be very different.

In fact GPU threads must be grouped together into **blocks of threads (called "thread blocks" in CUDA and "work-groups" in OpenCL**) that execute concurrently on the **GPU Streaming Multiprocessors (SMs)** according to a SIMD model…

…while MIC coprocessors run generic MIMD (**Multiple Instruction Multiple Data**) threads individually on the x86 cores.

Differences between GPU and MIC

GPU devices also support ***MIMD*** execution by allowing any block of threads to be scheduled to run on any SM on the GPU.

This MIMD capability is most efficient when the problem matches the **granularity of the SM thread block size (32 threads for current GPU devices**).

GPUs that support this modified form of MIMD capability are referred to as

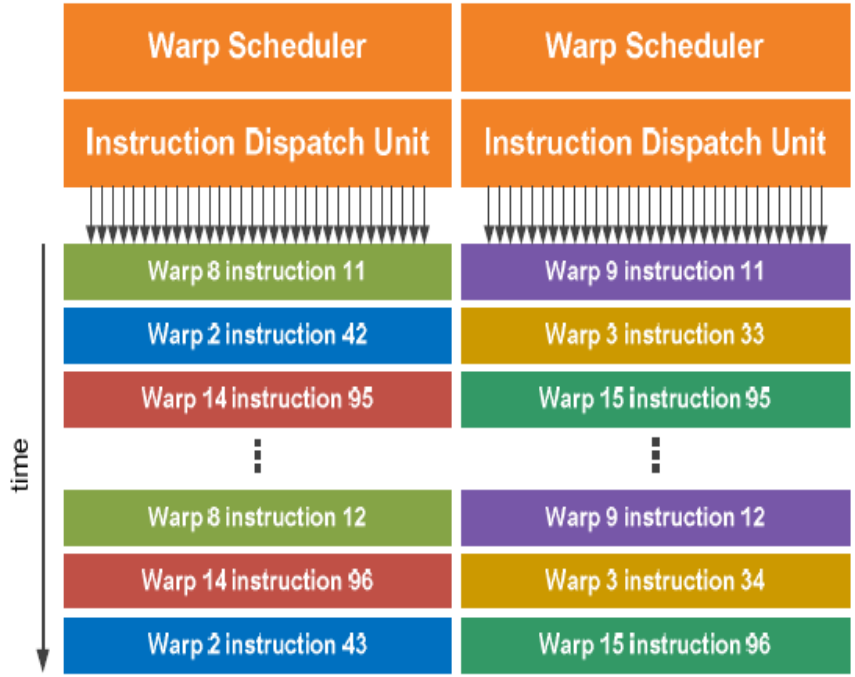
SIMT (**Single Instruction Multiple Thread**) devices.

SIMT:

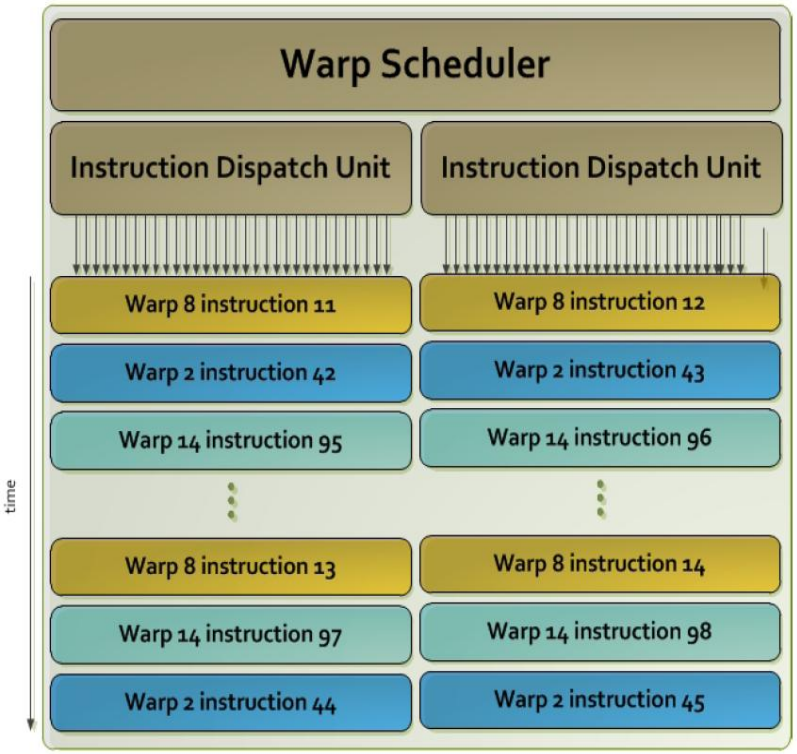
- Threads in groups (or 16, 32) that are scheduled together call **Warp.**
- All threads in a **Warp** start at the same PC, but free to branch and execute independently.
- A warp executes one common instruction at a time
  - To execute different instructions at different threads, the instructions are executed serially
    - To get efficiency, we want all instructions in a **Warp** to be the same.

GPU Warp scheduler evolution

- FERMI: 2 per SM: representing a compromise between cost and complexity



- Kepler: 4 per SM with 2 instruction dispatcher units each.



Differences between GPU and MIC: degree of parallelism

K40 GPU claims to have 2,880 "CUDA cores"…but…
…a K40 SM can concurrently execute up to **64 Warps** ,

meaning that **2048 SIMD threads** can be actively running (and not queued) at any moment in time.

A K40 GPU contains 15 SMs, which means **up to 2K*15 = 30K threads** can be queued to run at any given moment on the device.

The MIC coprocessor **combines many-core parallelism with a wide per-core wide vector unit** providing additional operations per clock.

As a result, the per core vector unit **basically multiplies the number of cores by the number of concurrent vector operations** to increase the MIC coprocessors degree of parallelism.

Open question: what can MICs learn from GPUs and viceversa?

It seems that GPUs and MICs are like two different cars that should run on the same path…

BUT….

… we still miss a **common** (consolidated) approach to drive…

Need for a **common** programming model???

?