INTRODUCTION TO XEON PHI

Naming Conventions

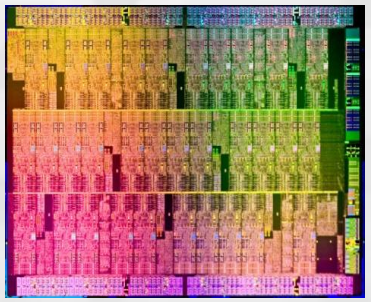
- It is a **coprocessor**, not a GPU or accelerator
- **MIC** is the name of the architecture
  - Comparable to Intel64 on CPU side
- **Xeon Phi** is the brand name of the product
- Architecture generation named as **Knight's …**
  - Comparable to ”Nehalem”, ”Sandy Bridge” etc. on CPU side
- Different models have number designations
  - i.e. 5110P, SE10

Timeline

- 2008 - **Larrabee** GPGPU announced
  - Was not productized
- 2010 - **MIC** and the **Knight's** series announced
  - Re-architected for pure computing
  - **Knight's Ferry** (**KNF**) development kit
- 2011 - **Knight's Corner** (**KNC**) development kit
  - Alpha/beta versions of the final products
- 2012 **- Intel Xeon Phi** brand introduced
  - First products based on the KNC architecture
- ???? - **Knight's Landing** (**KNL**)
  - Future architecture

Intel Many Integrated Core (MIC)

- PCI Express card, a ”PC in a PC”
- 10s of x86-based cores
  - Hardware multithreading
  - Instruction set extensions for HPC
- Very high-bandwidth local GDDR5 memory
- Runs a stripped-down version of Linux



Intel MIC Philosophy

- Design the hardware for HPC
  - Strip out ”general purpose” CPU features
    - Elaborate branch prediction, out-of-order execution etc.
- Leverage existing x86 architecture and programming models
  - Enable development using common code base
    - Same tools and libraries as on CPU
    - Same parallel paradigms (OpenMP, Cilk etc.)
  - Optimization strategies similar to CPU ones
    - Optimizations for Phi tend to improve CPU performance

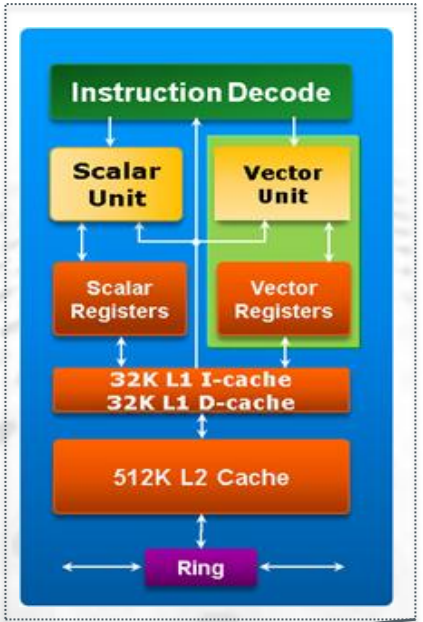
Current Xeon Phi Models

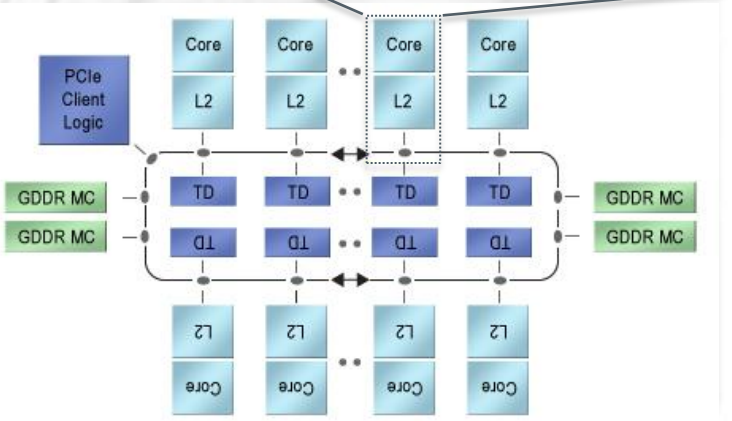
	5110P	**SE10** (special edition)	**3100**
Cores	***60***	61	TBA (1H13)
Clock Rate (GHz)	***1.053***	1.1	TBA (1H13)
Raw perf (Gflops)	***1011***	1073	>1000
Memory Size (GB)	***8***	8	6
L2 Size (MB)	***30***	30.5	28.5
Memory BW (GB/s)	***320***	352	240
TDP Power (W)	***225***	300	240
Cooling	***Passive***	Passive (SE10P) Vendor defined (SE10X)	Passive, Active

What we will be using

Xeon Phi Core Architecture

- 22nm process using 3D TriGate transistors
- Based on the old Pentium (P65C) core
  - Added 64-bit addressing, SIMD, distributed L2 etc.
  - In-order execution
  - Designed for 4 hardware threads per core
  - Ring bus similar to current Sandy Bridge CPUs
- Fast GDDR5 GPU memory on card
  - Discrete from host memory
  - Highly efficient HW prefetching
  - 4KB default page size
    - 2MB huge pages





Xeon Phi Instruction Set Architecture

Aka. LRBNI (Larrabee New Instructions)

- 512-bit wide SIMD vector ops
  - 8 double (or 16 float or integer) operations per cycle
- Fused Multiply-Add (FMA)
  - A <- A+(B*C) in a single cycle (with 1 rounding error)
    - 16 DP or 32 SP FLOPS/cycle
- Vector masks
  - Toggles which vector elements are affected by an instruction
- Gather / scatter operations
  - Used to access memory (in cache) in an irregular pattern
- Extended Math Unit (EMU)
  - Transcendetal operations (sqrt, log etc.)

Different from MMX, SSE, AVX, AVX2 which are used on x86 CPUs!

MIC vs. CPU

- Larger amount of simpler cores at a lower clock rate
  - Much less memory per core
  - Similar memory bandwidth per core
- In-order execution
  - Thread waits (stalls) on L1 cache miss
  - Sophisticated HW prefetching helps
- 2x wider vectors than AVX/AVX2
  - Also richer set of vector instructions (FMA, mask etc.)
  - Exploiting this fully is essential for performance on MIC
- Using 2-4 threads per core is essential
  - Using just 1 thread gives 50% performance
  - Can hide some of the latency caused by stalls

	L1 (Data + Instr)	Shared L2
Size	32 KB + 32 KB	512 KB
Miss Latency	15-30 cyc	500-1000 cycles

MIC in CUDA (GPGPU) terms

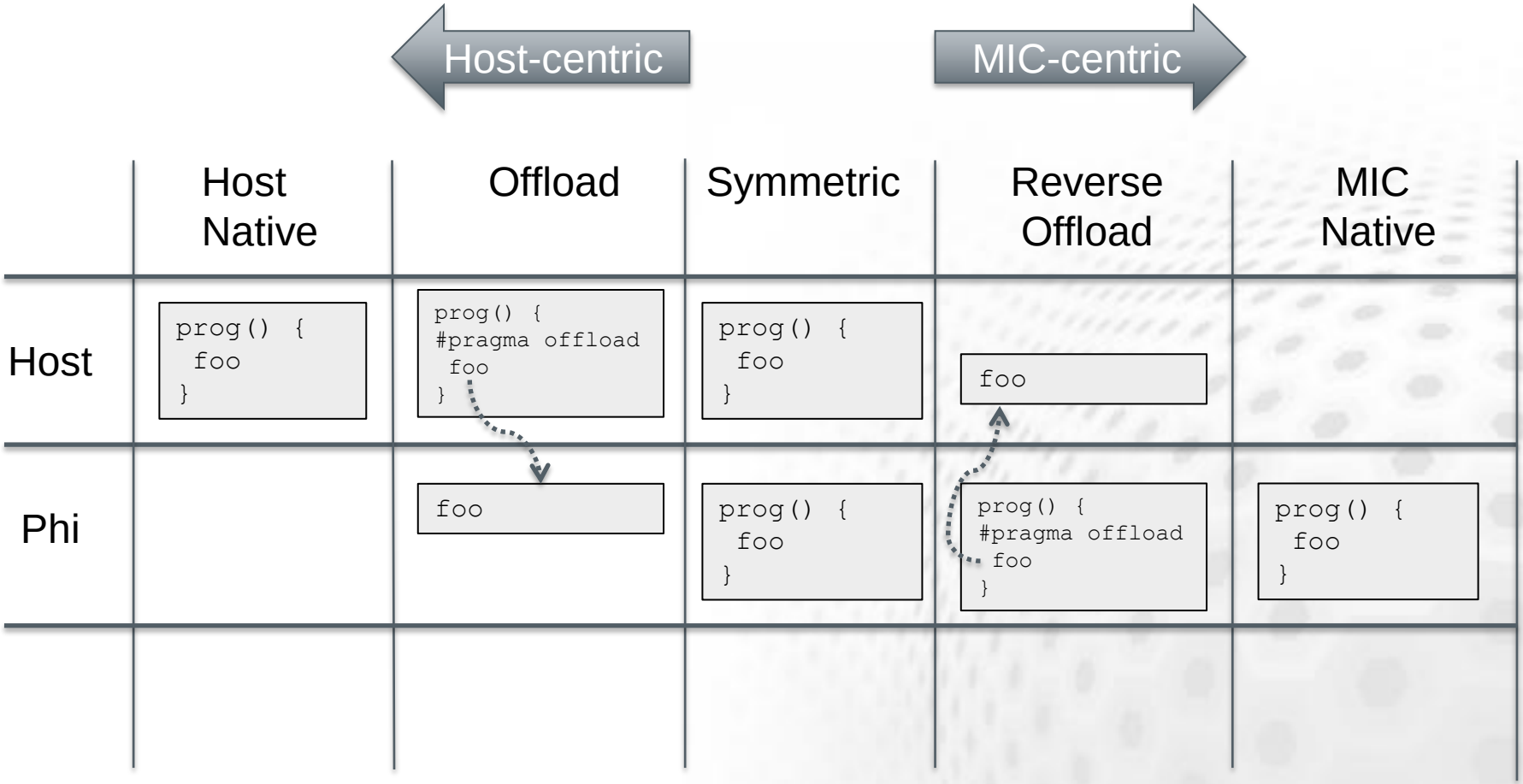
CUDA	MIC
CUDA core (1 FP op / cycle)	~= MIC SIMD lane
CUDA symmetric multiprocessor	~= MIC core
CUDA thread block	~= MIC threads in a core
One operation on a CUDA Warp	~= One MIC SIMD operation
Large oversubscription of work (>4 x resident warps per SM optimal)	Moderate oversubscription of work (2-4x threads per core optimal)
Automatic and manual local caching	Coherent, automatic L2 cache and hardware prefetching
CUDA, OpenCL, OpenACC offloads, Libraries (CUBLAS, CUFFT etc.)	Legacy programming models (OpenMP etc.), LEO offloads, OpenCL , Libraries (MKL etc.)
Host CPU needed for execution	Independent native execution of code

Operating System & Environment

- Standard Linux kernel with patches
- Lightweight ”Busybox” shell environment
  - A subset of the standard shell commands & tools
    - Usage differs from their full counterparts
- Only core set of libraries (glibc etc.) by default
- Binary architecture: **k1om** (not x86_64)
- Standard set of ”de facto” HPC languages
  - C/C++/Fortran
- MIC’s local disk is mapped into it’s memory

Copying files onto the Phi’s local ”disk” takes up precious GDDR5 memory!

Programming Models



Useful Links

- PRACE Xeon Phi Best Practices Guide
  - http://www.prace-ri.eu/Best-Practice-Guides
- Xeon Phi Developer's Quick Start Guide
  - http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide
- Dr. Dobb's Xeon Phi Guide
  - http://www.drdobbs.com/parallel/programming-intels-xeon-phi-a-jumpstart/240144160
- Phi programming for CUDA developers
  - http://www.drdobbs.com/parallel/cuda-vs-phi-phi-programming-for-cuda-dev/240144545