Introduction to Many Integrated Core (MIC) Coprocessors on Stampede

Aaron Birkland

Cornell CAC

With contributions from Steve Lantz from CAC, and Lars Koesterke, Bill Barth, Kent Milfield, John Cazes, and Lucas Wilson at TACC

Parallel Computing on Stampede

Oct 23, 2013

Stampede Specs

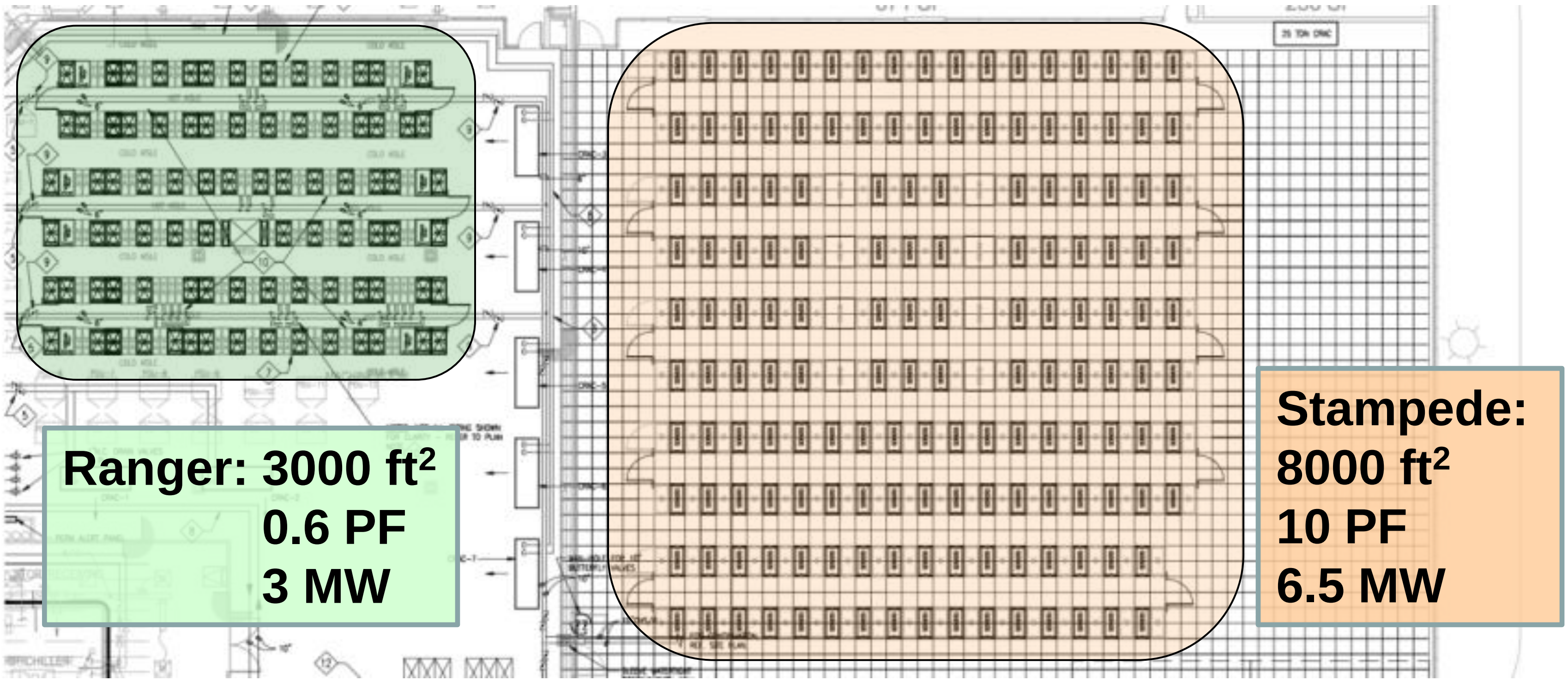
- 6400 Dell C8220X nodes in initial system
  - 16 Xeon E5 “Sandy Bridge” cores per node, 102400 total
  - 32GB memory per node, 200TB total
- At least 6400 Xeon Phi™ SE10P coprocessor cards
- 2+ petaflop/s Intel Xeon E5
- 7+ additional petaflop/s of Intel Xeon Phi™ SE10P coprocessors to change the power/performance curves of supercomputing
- Over 70% provided by Xeon Phi
- Learn to leverage the 7+


Photo by TACC, June 2012

Xeon Phi: What is it?

- System on PCIe card (Linux OS, Processor, Memory)
- x86-derived processor featuring large number of simplified cores
  - Many Integrated Core (MIC) architecture.
- Optimized for floating point throughput
- Modified 64-bit x86 instruction set
  - Code compatible (C, C++, FORTRAN) with re-compile
  - Not binary compatible with x86_64
- Supports same HPC programming paradigms with same code (MPI, OpenMP, Hybrid).
- Offers new Offload paradigm
  - C/FORTRAN markup to denote code to execute on Phi at runtime
  - Link to MKL library implementation which can offload automatically

Stampede Footprint vs. Ranger

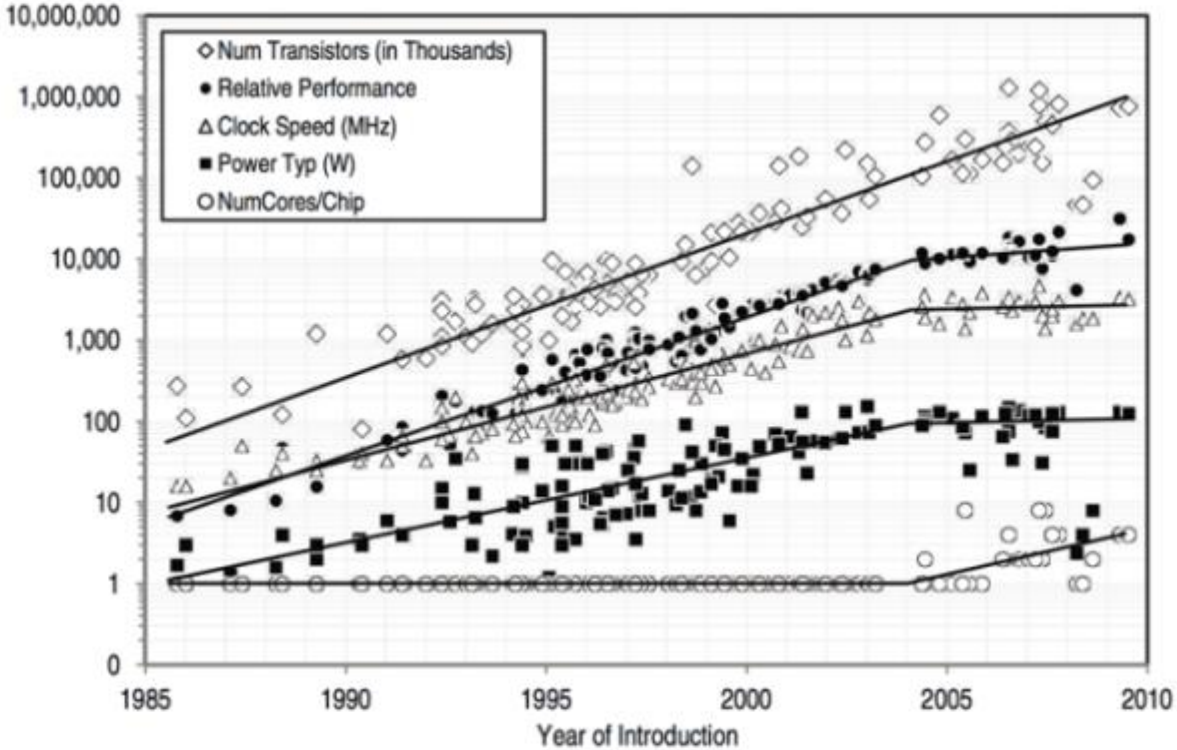


- Capabilities are 17x; footprint is 2.7x; power draw is 2.1x

How Does Stampede Reach Petaflop/s?

- Hardware trend since around 2004: processors gain more *cores* (execution engines) rather than greater clock speed
  - IBM POWER4 (2001) became the first chip with 2 cores, 1.1–1.9 GHz; meanwhile, Intel's single-core Pentium 4 was a bust at >3.8 GHz
  - Top server and workstation chips in 2013 (Intel Xeon, AMD Opteron) now have 4, 8, even 16 cores, running at 1.6–3.2 GHz
- Does it mean Moore's Law is dead? No!
  - Transistor densities are still doubling every 2 years
  - Clock rates have stalled at < 4 GHz due to power consumption
  - Only way to increase flop/s/watt is through greater on-die parallelism…

CPU Speed and Complexity Trends



Committee on Sustaining Growth in Computing Performance, National Research Council.
"What Is Computer Performance?"
In *The Future of Computing Performance: Game Over or Next Level?*
Washington, DC: The National Academies Press, 2011.

Trends for Petaflop/s Machines

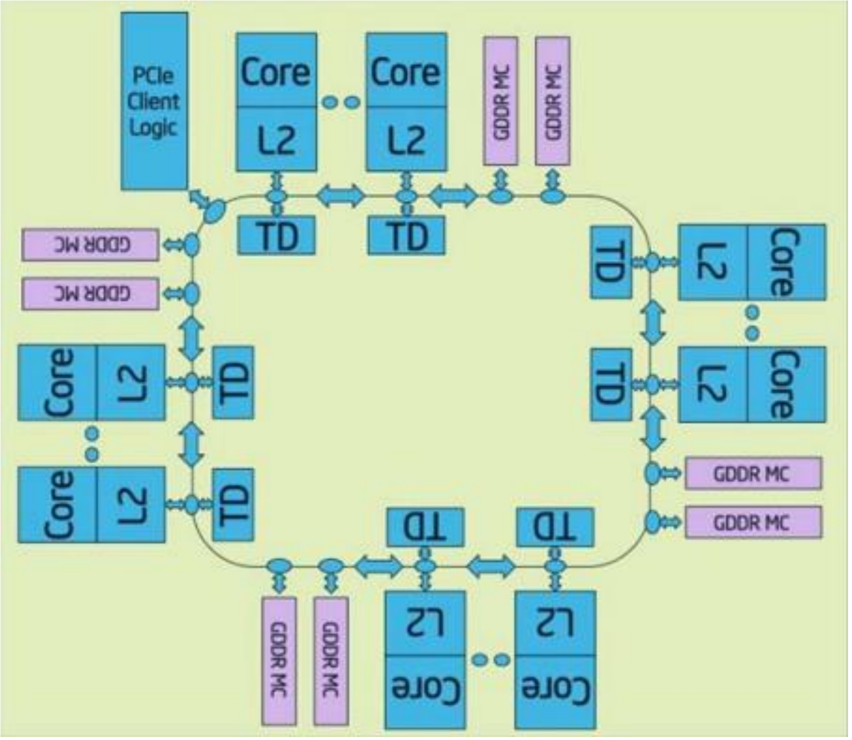
- CPUs: Wider vector units, more cores
  - General-purpose in nature
  - High single-thread performance, moderate floating point throughput
  - 2x E5-2608 on Stampede: 0.34 Tflop/s, 260W
- GPUs: Thousands of very simple stream processors
  - Specialized for floating point.
  - New programming models: CUDA, OpenCL, OpenACC
  - Tesla K20 on Stampede: 1.17 Tflop/s, 225W
- MIC: Take CPU trends to an extreme, optimize for floating point.
  - Retain general-purpose nature and programming models from CPU
  - Low single-thread performance, high aggregate FP throughput
  - SE10P on Stampede: 1.06 Tflops/s, 300W

Attractiveness of MIC

- Programming MIC is similar to programming for CPUs
  - C/C++, Fortran
  - OpenMP, MPI
  - MPI on host and coprocessor
  - General purpose computing, not just kernels
  - In many cases, just re-compile
- Optimizing for MIC is similar to optimizing for CPUs
  - "Optimize once, run anywhere"
  - Fundamental architectural similarities
- Offers a new, flexible Offload programming paradigm
  - Resembles GPU computing patterns in some ways

MIC Architecture

- SE10P is first production version used in Stampede
  - Chip, memory on PCIe card
  - 61 cores, each containing:
    - 64 KB L1 cache
    - 512 KB L2 cache
    - 512 byte vector unit
  - 31.5 MB total coherent L2 cache, connected by ring bus
  - 8 GB GDDR5 memory
    - Very fast, 352 GB/s vs 50 GB/s/socket for E5



Courtesy Intel

Key Architectural Design Decisions

- For power saving
  - Omit power-hungry features such as branch prediction, out-of-order execution (at the cost of single-thread performance)
  - Simplify instruction decoder so that instructions are issued every other clock cycle from a given thread (a single thread can utilize at most 50% of a core)
  - Reduce clock speed (at the cost of single-thread performance, obviously)
  - Eliminate a shared L3 cache in favor of coherent L2 caches (performance impacts are subtle – can help and hurt)

Key Architectural Design Decisions

- For floating point performance
  - Use wide vector units (512-bit vs 256-bit for Xeon E5)
  - Use more cores
  - Use up to four hardware threads per core.
    - Compensates for some of the power-saving compromizes: in-order execution, simplified instruction decoder)
  - Use fast GDDR5 memory

As a result, performance characteristics are very different!

MIC vs. CPU

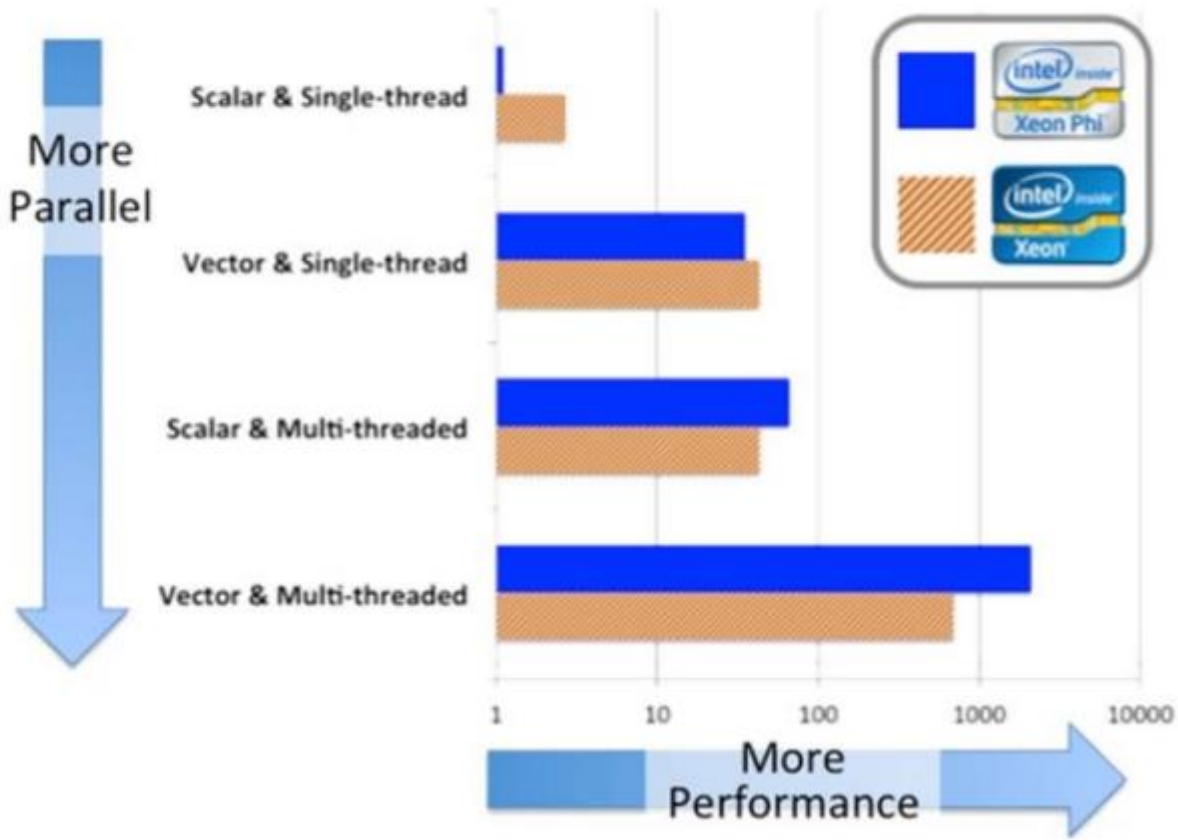
	MIC (SE10P)	CPU (E5)	MIC is…
• Number of cores	61	8	much higher
• Clock Speed (GHz)	1.01	2.7	lower
• SIMD width (bit)	512	256	higher
• DP GFLOPS/core	16+	21+	lower
• HW threads/core	4	1*	higher

- CPUs designed for all workloads, high single-thread performance
- MIC also general purpose, though optimized for number crunching
  - Focus on high aggregate throughput via lots of weaker threads
  - Regularly achieve >2x performance compared to dual E5 CPUs

Two Types of CPU/MIC Parallelism

- Threading (work-level parallelism)
  - OpenMP, Cilk Plus, TBB, Pthreads, etc
  - It's all about sharing work and scheduling
- Vectorization (data-level parallelism)
  - "Lock step" Instruction Level Parallelization (SIMD)
  - Requires management of synchronized instruction execution
  - It's all about finding simultaneous operations
- To fully utilize MIC, both types of parallelism need to be identified and exploited
  - Need at 2-4 threads to keep a MIC core busy (in-order execution stalls)
  - Vectorized loops gain 8x performance on MIC!
  - Important for CPUs as well: gain of 4x on Sandy Bridge

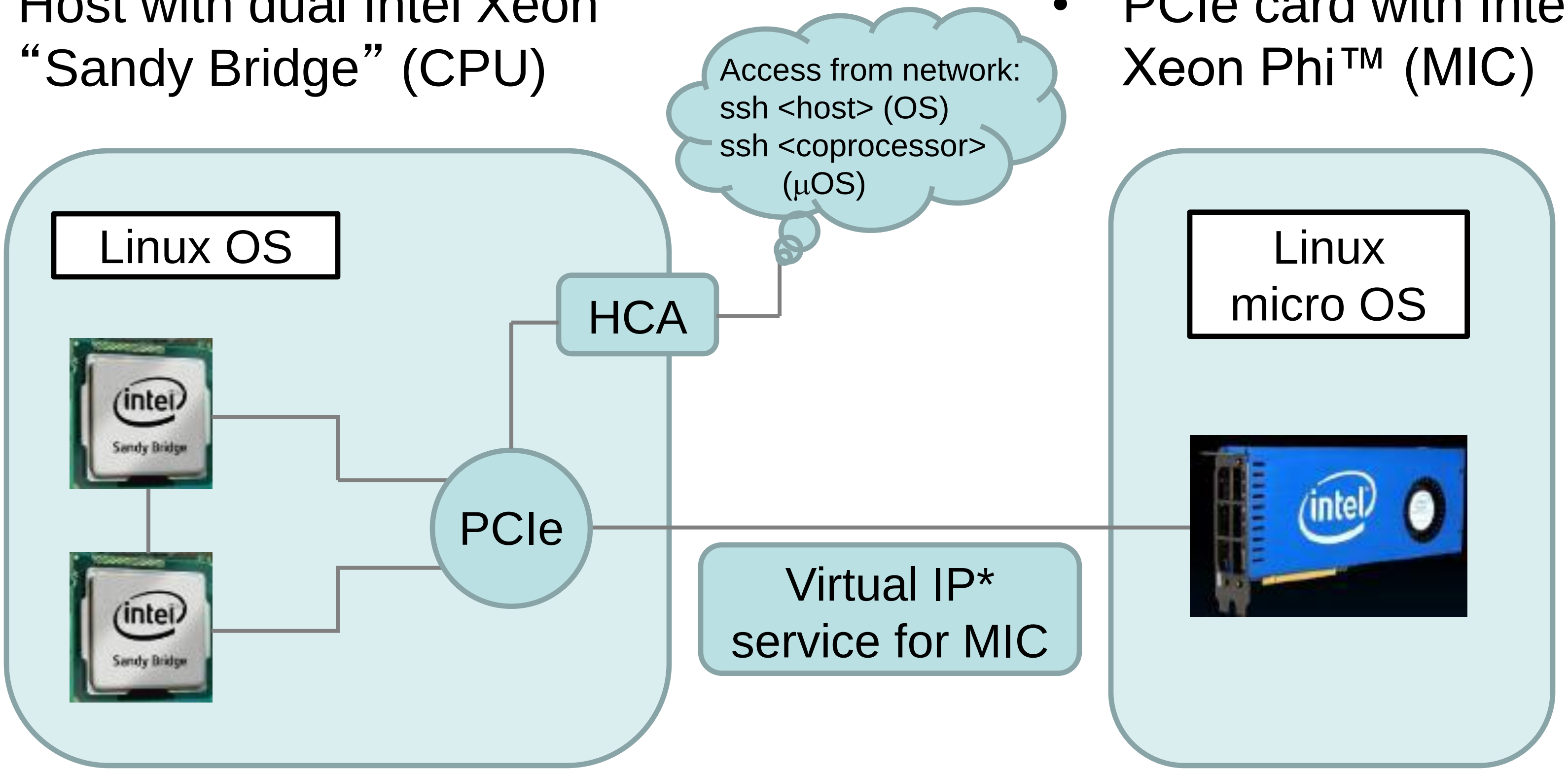
Parallelism and Performance on MIC and CPU



Courtesy Intel

Typical Configuration of a Stampede Node

- Host with dual Intel Xeon "Sandy Bridge" (CPU)
- PCIe card with Intel Xeon Phi™ (MIC)



MIC Resembles a Compute Node

- Participates in network via established APIs
  - TCP/IP, SSH, NFS; Has its own hostname.
- Runs own OS, can log into it and get shell
- $HOME, $WORK, $SCRATCH mounted on it
  - You or your programs can read/write/execute files
- MPI infrastructure can launch jobs on it

But, there are some key differences

- SLURM and batch system don't directly interact with MIC cards
- Minimal 3rd party software modules installed on it
- Forms heterogeneous cluster when using MPI on MIC and hosts.

MIC Execution Models for Stampede

Native Execution

- Compile one executable for MIC architecture

```
icc -O2 -mmic -openmp myprog.c -o myprog.mic
```

- Convenient to use `.mic` suffix for executables to serve as a reminder
- Run directly on MIC coprocessor
  - Use ssh or TACC's convenient `micrun` launcher

```
c123-456$ ssh mic0
~ $ export OMP_NUM_THREADS=180
~ $ /path/to/myprog.mic
```

MIC Execution Models for Stampede

Native Execution

- `micrun` launcher designed to make running MIC executables simple from host.
  - Set specific environment variables with `MIC_` prefix
  - Receive proper return value
  - Can be used explicitly via `micrun`, or implicitly

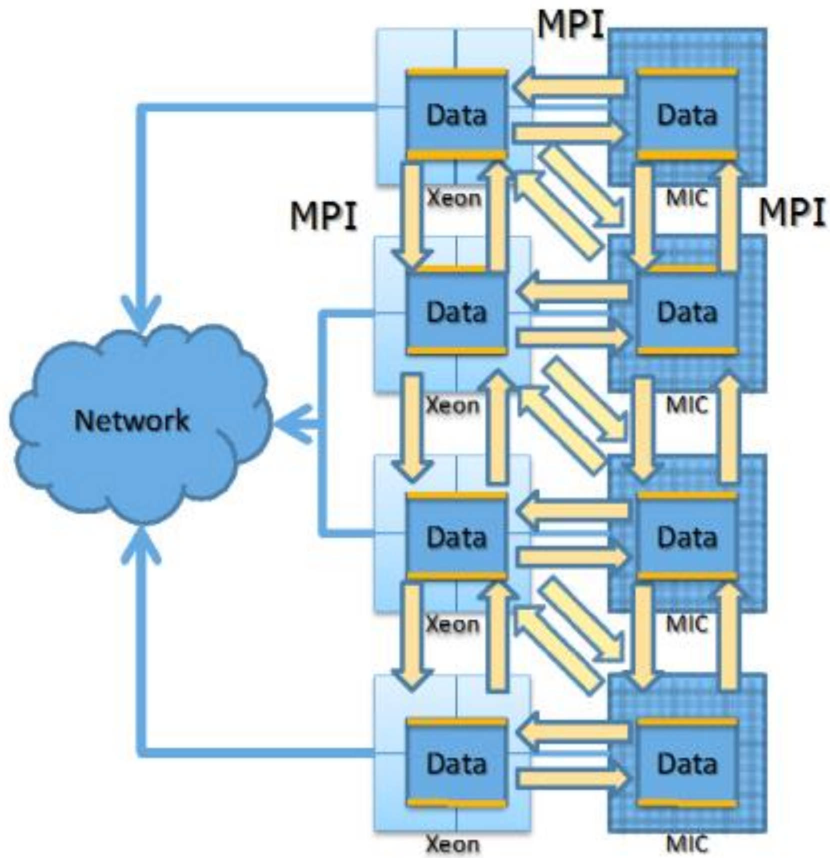
```
c123-456$ export MIC_OMP_NUM_THREADS=180
c123-456$ /path/to/myprog.mic
```

```
c123-456$ export MIC_OMP_NUM_THREADS=180
c123-456$ micrun /path/to/myprog.mic
```

MIC Execution Models for Stampede

"Symmetric" Execution

- Message passing (MPI) on CPUs and MICs alike
- Unified source code
- Code modifications optional
  - Assign different work to CPUs vs. MICs
  - Multithread with OpenMP for CPUs, MICs, or both
- Compile twice, 2 executables
  - One for MIC, one for host
- Run in parallel using MPI



Courtesy Scott McMillan, Intel

MIC Execution Models for Stampede

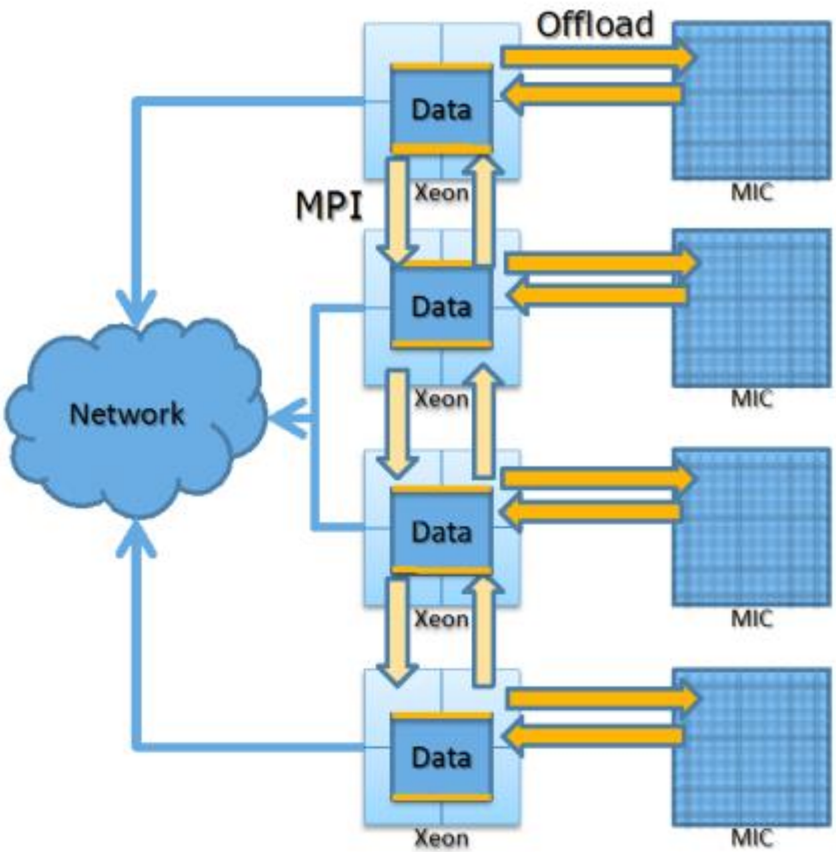
Symmetric Execution

- Use `ibrun.symm` MPI launcher.
  - Like `ibrun`, but adds capability of launching processes on MIC coprocessors
  - Use `-c` argument to specify host CPU executable, `-m` to specify MIC executable
  - Standard SLURM params (`-N`, `-n`) determine *total* number of compute nodes, and host processes
  - `MIC_PPN` environment variable to control number of MIC processes per Phi card
  - Only `MIC_` prefixed environment variables are sent to MIC processes
- Right now, only Intel MPI implementation (`impi`) supported.

MIC Execution Models for Stampede

Offload Execution

- Directives indicate data and functions to send from CPU to MIC for execution
- Unified source code
- Code modifications required
- Compile once
- Run in parallel using MPI and/or scripting, if desired



Courtesy Scott McMillan, Intel

MIC Execution Models for Stampede

Offload Execution

- Option 1: With compiler-assisted offload, you *write* code and offload annotations
  - No specific compiler flags needed, offload is implicit where markup is encountered
  - Offload code will automatically run on MIC at runtime if MIC is present, otherwise host version is run
- Option 2: With automatic offload, you *link* to a library that can perform offload operations (e.g. MKL)
  - Stampede MKL is offload-capable, all you do is link to it (`-lmkl`)!
  - Need to explicitly tell MKL to use offload at runtime via environment variable `MKL_MIC_ENABLE=1`

Which Execution Model?

- Native is very useful for performance testing, empirical analysis
  - Works well for interactive jobs
  - Re-compile and run!
- Use Symmetric to run existing MPI code on MIC only, or Host +MIC
  - MIC coprocessor is just another node
  - Using both Host and MIC creates a heterogeneous cluster
    - Potential balancing issues, but these may possibly be addressed by runtime parameters, not necessarily code changes
- Use automatic offload for code that uses an API found in MKL (BLAS, LAPACK)
- Compiler-assisted offload can give fine-grained control: Keep slow, serial parts on CPU, run tight parallel loops on MIC or both.

Labs

- Interactive Launching
  - Run native code on host, MIC interactively
- Simple Symmetric MPI
  - Use `ibrun.symm` to control number of jobs running on host and MIC, verify that they're running where you think they are

Next week in advanced MIC:

- Non-trivial Symmetric example
  - Use hybrid code (MPI+OpenMP) to calculate PI
  - Investigate issues related to performance disparity between host and coprocessor

Reference

- Some information in this talk was gathered from presentations at the TACC–Intel Highly Parallel Computing Symposium, Austin, Texas, April 10–11, 2012: http://www.tacc.utexas.edu/ti-hpcs12.
- Stampede User Guide https://portal.tacc.utexas.edu/user-guides/stampede
- Intel press materials http://newsroom.intel.com/docs/DOC-3126
- Intel MIC developer information http://software.intel.com/mic-developer