Using the Intel Xeon Phi
(with the Stampede Supercomputer)
ISC’13 Tutorial

Bill Barth, Kent Milfeld, Dan Stanzione
Tommy Minyard
Texas Advanced Computing Center
Jim Jeffers, Intel

June 2013, Leipzig, Germany
Acknowledgements

• Thanks/kudos to:
  – **Sponsor**: National Science Foundation
    • NSF Grant #OCI-1134872 Stampede Award, “Enabling, Enhancing, and Extending Petascale Computing for Science and Engineering”
    • NSF Grant #OCI-0926574 - “Topology-Aware MPI Collectives and Scheduling”
  – Many, many Dell, Intel, and Mellanox engineers
  – *Professor D.K. Panda at OSU (MVAPICH2)*
  – All my colleagues at TACC who saw way too much of each other during the last year
Thanks for Coming!

• In this tutorial, we will teach you what you need to know to start running your code on the Intel Xeon Phi Co-Processor.

• For the hands-on portion, we will use the Stampede system at TACC in Austin, Texas, USA, which has 6,880 Xeon Phi cards.
Today’s Session

• Dan: (45 minutes)
  – An overview of Intel’s Many Core Architecture
  – Overview of Stampede
  – Programming Models for the Xeon Phi
• Bill: Understanding Vectorization (45 minutes)
• Break!
• Kent: Threading models and Offload (45)
• Hands-on lab (Vectorization and OpenMP)
  – 1 hour and whatever is left
Stampede - High Level Overview

• Base Cluster (Dell/Intel/Mellanox):
  – Intel Sandy Bridge processors
  – Dell dual-socket nodes w/32GB RAM (2GB/core)
  – 6,400 nodes
  – 56 Gb/s Mellanox FDR InfiniBand interconnect
  – More than 100,000 cores, 2.2 PF peak performance

• Co-Processors:
  – Intel Xeon Phi “MIC” Many Integrated Core processors
  – Special release of “Knight’s Corner” (61 cores)
  – All MIC cards are on site at TACC
    • more than 6000 installed
    • final installation ongoing for formal summer acceptance
  – 7+ PF peak performance

• Max Total Concurrency:
  – exceeds 500,000 cores
  – 1.8M threads

• Entered production operations on January 7, 2013
Additional Integrated Subsystems

- Stampede includes 16 1TB Sandy Bridge shared memory nodes with dual GPUs
- 128 of the compute nodes are also equipped with NVIDIA Kepler K20 GPUs (and MICS for performance bake-offs)
- 16 login, data mover and management servers (batch, subnet manager, provisioning, etc)
- Software included for high throughput computing, remote visualization
- Storage subsystem driven by Dell storage nodes:
  - Aggregate Bandwidth greater than 150GB/s
  - More than 14PB of capacity
  - Similar partitioning of disk space into multiple Lustre filesystems as previous TACC systems ($HOME, $WORK and $SCRATCH)
Power/Physical

• Stampede spans 182 48U cabinets.
• Power density (after upgrade in 2015) will exceed 40kW per rack.
• Estimated 2015 peak power is 6.2MW.
Stampede Datacenter Features

- Thermal energy storage to reduce peak power charges
- Hot aisle containment to boost efficiency (and simply provide enough cooling).
- Total IT power to 9.5MW, total power ~12MW.
- Expand experiments in mineral oil cooling.
New Construction in 2012
Required for Stampede

Construction Started Nov. 2011
Stampede Footprint

Machine Room Expansion
Added 6.5MW of additional power
Stampede Datacenter – February 20th
Stampede Datacenter – March 22nd
Stampede Datacenter – May 16\textsuperscript{th}
Stampede Datacenter – June 20th
Stampede Datacenter, ~September 10th
Stampede Datacenter, ~September 10th
Some utilities are involved
Actually, way more utility space than machine space

Turns out the utilities for the datacenter costs more, takes more time and more space than the computing systems.
Innovative Component

• One of the goals of the NSF solicitation was to “introduce a major new innovative capability component to science and engineering research communities”

• We proposed the Intel Xeon Phi coprocessor (many integrated core or MIC)
  – one first generation Phi installed per host during initial deployment
  – confirmed injection of 1600 future generation MICs in 2015 (5+ PF)
An Inflection point (or two) in High Performance Computing

- Relatively “boring”, but rapidly improving architectures for the last 15 years.
- Performance rising much faster than Moore’s Law…
  - But power rising faster
  - And concurrency rising faster than that, with serial performance decreasing.
- Something had to give…
Consider this Example
Homogeneous v. Heterogeneous

Acquisition costs and power consumption

• Homogeneous system: K computer in Japan
  – 10 PF unaccelerated
  – Rumored $1.2 billion acquisition costs
  – Large power bill and foot print

• Heterogeneous system: Stampede at TACC
  – Near 10PF (2 SNB + up to 8 MIC)
  – Modest footprint: 8,000 ft², 6.5MW
  – $27.5 million
Accelerated Computing for Exascale

• Exascale systems, predicted for 2018, would have required 500MW on the “old” curves.
• Something new was clearly needed.
• The “accelerated computing” movement was reborn (this happens periodically, starting with 387 math coprocessors).
Key aspects of acceleration

- We have lots of transistors… Moore’s law is holding; this isn’t necessarily the problem.
- We don’t really need lower power per transistor, we need lower power per *operation*.
- How to do this?
  - nVidia GPU
  - AMD Fusion
  - FPGA
  - ARM Cores
Intel’s MIC approach

• Since the days of RISC vs. CISC, Intel has mastered the art of figuring out what is important about a new processing technology, and saying “why can’t we do this in x86?”

• The Intel Many Integrated Core (MIC) architecture is about large die, simpler circuit, much more parallelism, in the x86 line.
What is MIC?

Basic Design Ideas:

- Leverage x86 architecture (a CPU with many cores)
- Use x86 cores that are simpler, but allow for more compute throughput
- Leverage existing x86 programming models
- Dedicate much of the silicon to floating point ops., keep some cache(s)
- Keep cache-coherency protocol
- Increase floating-point throughput per core
- Implement as a separate device
- Strip expensive features (out-of-order execution, branch prediction, etc.)
- Widened SIMD registers for more throughput (512 bit)
- Fast (GDDR5) memory on card
Xeon Phi — MIC

• Xeon Phi = first product of Intel’s Many Integrated Core (MIC) architecture
• Co-processor
  – PCI Express card
  – Stripped down Linux operating system
  – But still a full host… you can log in to the Phi directly and run stuff!
• Lots of names
  – Many Integrated Core architecture, aka MIC
  – Knights Corner (code name)
  – Intel Xeon Phi Co-processor SE10P (product name of the version for Stampede)
  – I should have many more trademark symbols in my slides if you ask Intel!
Intel Xeon Phi Chip

- 22 nm process
- Based on what Intel learned from
  - Larrabee
  - SCC
  - TeraFlops Research Chip
MIC Architecture

- Many cores on the die
- L1 and L2 cache
- Bidirectional ring network for L2
- Memory and PCIe connection
Knights Corner Core

George Chrysos, Intel, Hot Chips 24 (2012):
George Chrysos, Intel, Hot Chips 24 (2012):
Speeds and Feeds
Stampede SE10P version
(Your mileage may vary)

• Processor
  – ~1.1 GHz
  – 61 cores
  – 512-bit wide vector unit
  – 1.074 TF peak DP

• Data Cache
  – L1 32KB/core
  – L2 512KB/core, 30.5 MB/chip

• Memory
  – 8GB GDDR5 DRAM
  – 5.5 GT/s, 512-bit*

• PCIe
  – 5.0 GT/s, 16-bit
What we at TACC like about MIC

• Intel’s MIC is based on x86 technology
  – x86 cores w/ caches and cache coherency
  – SIMD instruction set

• Programming for MIC is similar to programming for CPUs
  – Familiar languages: C/C++ and Fortran
  – Familiar parallel programming models: OpenMP & MPI
  – MPI on host and on the coprocessor
  – Any code can run on MIC, not just kernels

• Optimizing for MIC is similar to optimizing for CPUs
  – “Optimize once, run anywhere”
  – Optimizing can be hard; but everything you do to your code should
    *also* improve performance on current and future “regular” Intel chips, AMD CPUs, etc.
The MIC Promise

• Competitive Performance: Peak and Sustained

• Familiar Programming Model
  – HPC: C/C++, Fortran, and Intel’s TBB
  – Parallel Programming: OpenMP, MPI, Pthreads, Cilk Plus, OpenCL
  – Intel’s MKL library (later: third party libraries)
  – Serial and Scripting, etc. (anything a CPU core can do)

• “Easy” transition for OpenMP code
  – Pragmas/directives added to “offload” OMP parallel regions onto MIC
  – See examples later

• Support for MPI
  ✓ MPI tasks on host, communication through offloading
  ✓ MPI tasks on MIC, communication through MPI
Programming the Xeon Phi

• Some universal truths of Xeon Phi coding:
    • At least 2 per core to have a chance
    • 3 or 4 per core are often optimal.
    • This is not like old school hyperthreading, if you don’t have 2 threads per core, half your cycles will be wasted.
  – You need to vectorize.
    • This helps on regular processors too, but the effect is much more pronounced on the MIC chips.
    • You can no longer ignore the vectorization report from the compiler

• If you can do these 2 things, good things will happen, in most any language.
MIC Programming Experiences at TACC

• Codes port easily
  – Minutes to days depending mostly on library dependencies

• Performance requires real work
  – While the silicon continues to evolve
  – Getting codes to run *at all* is almost too easy; really need to put in the effort to get what you expect

• Scalability is pretty good
  – Multiple threads per core *really important*
  – Getting your code to vectorize *really important*
Stampede Programming Models

• Traditional Cluster, or “Host Only”
  – Pure MPI and MPI+X (OpenMP, TBB, Cilk+, OpenCL…)
  – Ignore the MIC
• “Offload” MPI on Host, Offload to Xeon Phi
  – Targeted offload through OpenMP extensions
  – Automatically offload some library routines with MKL
• “Native” Phi
  – Use one Phi and run OpenMP or MPI programs directly
• “Symmetric” - MPI tasks on Host and Phi
  – Treat the Phi (mostly) like another host
    • Pure MPI and MPI+X (limited memory: using ‘X’ is almost mandatory)
Programming Intel® MIC-based Systems

**MPI+Offload**

- MPI ranks on Intel® Xeon® processors (only)
- All messages into/out of processors
- Offload models used to accelerate MPI ranks
- Intel® Cilk™ Plus, OpenMP*, Intel® Threading Building Blocks, Pthreads* within Intel® MIC
- Homogenous network of hybrid nodes:
Programming Intel® MIC-based Systems

Many-core Hosted

- MPI ranks on Intel® MIC (only)
- All messages into/out of Intel® MIC
- Intel® Cilk™ Plus, OpenMP*, Intel® Threading Building Blocks, Pthreads used directly within MPI processes
- Programmed as homogenous network of many-core CPUs:
Programming Intel® MIC-based Systems
Symmetric

- MPI ranks on Intel® MIC and Intel® Xeon® processors
- Messages to/from any core
- Intel® Cilk™ Plus, OpenMP®, Intel® Threading Building Blocks, Pthreads* used directly within MPI processes

- Programmed as heterogeneous network of homogeneous nodes:
Native Execution

• Build for Phi with –mmic
• Execute on host (runtime will automatically detect an executable built for Phi)
• ... or ssh to mic0 and run on the Phi
• Can safely use all 61 cores
  – Use at least 2 per core, really
  – Offload programs should certainly stay away from the 61st core since the offload daemon runs here
Symmetric MPI

- Host and Phi can operate symmetrically as MPI targets
  - High code reuse
  - MPI and hybrid MPI+X (X = OpenMP, Cilk+, TBB, pthreads)
- Careful to balance workload between big cores and little cores
- Careful to create locality between local host, local Phi, remote hosts, and remote Phis
Symmetric MPI

- Typical 1-2 GB per task on the host
- Targeting 1-10 MPI tasks per Phi on Stampede
  - With 6+ threads per MPI task
MPI with Offload to Phi

- Existing codes using accelerators have already identified regions where offload works well
- Porting these to OpenMP offload should be straightforward
- Automatic offload where MKL kernel routines can be used
  – xGEMM, etc.
LBM Example

- Lattice Boltzmann Method CFD code
  - Carlos Rosales, TACC
  - MPI code with OpenMP
- Finding all the right routines to parallelize is critical

![Execution times KNC(B0,1.0GHz) vs SB(3.1GHz)]
PETSc/MUMPS with AO

- Hydrostatic ice sheet modeling
- MUMPS solver (called through PETSc)
- BLAS calls automatically offloaded behind the scenes
Will My Code Run on Xeon Phi?

- Yes
- ... but that’s the wrong question
  - Will your code run *best* on Phi?, or
  - Will you get great Phi performance without additional work?
Other options

• In this tutorial, we focus mainly on C/C++ and Fortran, with threading through OpenMP.

• Other options are out there, notably:
  – Intel’s cilk extensions for C/C++
  – TBB (Threading Building Blocks) for C++
  – OpenCL support is available, but we have little experience so far.
  – It’s Linux, you can run whatever you want (ie., I have run a pthreads code on Phi).
Stampede: How Do Users Use It?

- **2+ PF Xeon-only system (MPI, OpenMP)**
  - Many users will use it as an extremely powerful Sandy Bridge cluster—and that’s OK!
    - They may also use the shared memory nodes, remote vis

- **7+ PF MIC-only system (MPI, OpenMP)**
  - Homogeneous codes can be run entirely on the MICs!

- **~10PF heterogeneous system (MPI, OpenMP)**
  - Run separate MPI tasks on Xeon vs. MIC; use OpenMP extensions for offload for hybrid programs
Summary

• MIC experiences
  – Early scaling looks good; application porting is fairly straightforward since it can run native C/C++, and Fortran code
  – Some optimization work is still required to get at all the available raw performance for a wide variety of applications; but working well for some apps
  – **vectorization** on these large many-core devices is key
  – **affinitization** can have a strong impact (positive/negative) on performance
  – **Algorithmic threading performance** is also key; if the kernel of interest does not have high scaling efficiency on a standard x86_64 processor (8-16 cores), it will not scale on many-core
  – **MIC optimization efforts also yield fruit on normal Xeon** (in fact, you may want to optimize there first.)

• Questions?

• Then let’s dive in deeper