Programming and Optimization for Intel® Architecture

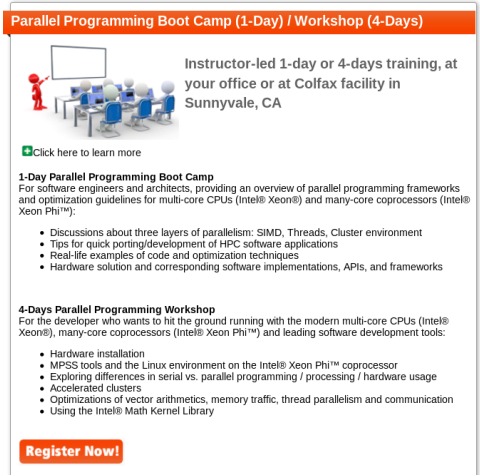
The Hands-On Workshop (HOW) Series

Andrey Vladimirov, PhD, and Ryo Asai
Colfax International — @colfaxintl

March 2016 , Rev. 02a

About This Document

This document represents the materials of a Web-based training “Programming and Optimization with Intel Architecture“ developed and run by Colfax International.



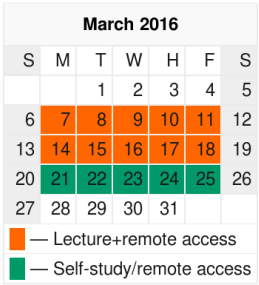
colfaxresearch.com/how-series

Disclaimer

While best efforts have been used in preparing this training, Colfax International makes no representations or warranties of any kind and assumes no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaims any implied warranties of merchantability or fitness of use for a particular purpose. The publisher shall not be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information or programs contained herein. No warranty may be created or extended by sales representatives or written sales materials.

Course Roadmap

1. Why Intel Parallel Architectures?
   - Parallelism and specialization – March 7
   - Programming model continuity – March 7
2. Programming models for Xeon Phi coprocessors
   - Native programming – March 7
   - Offload programming – March 8
3. Expressing Parallelism
   - Introduction to vectorization – March 9
   - Crash-course on OpenMP – March 10
4. Optimization – intro on March 11
   - Vectorization tuning – March 14
   - Multi-threading – March 15, 16
   - Memory traffic – March 17
5. Tools: MKL and VTune – March 18

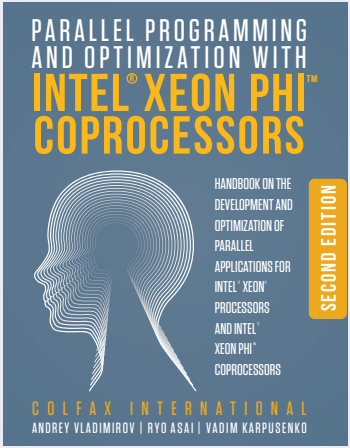


Textbook

ISBN: 978-0-9885234-0-1 (508 pages, Electronic or Print)

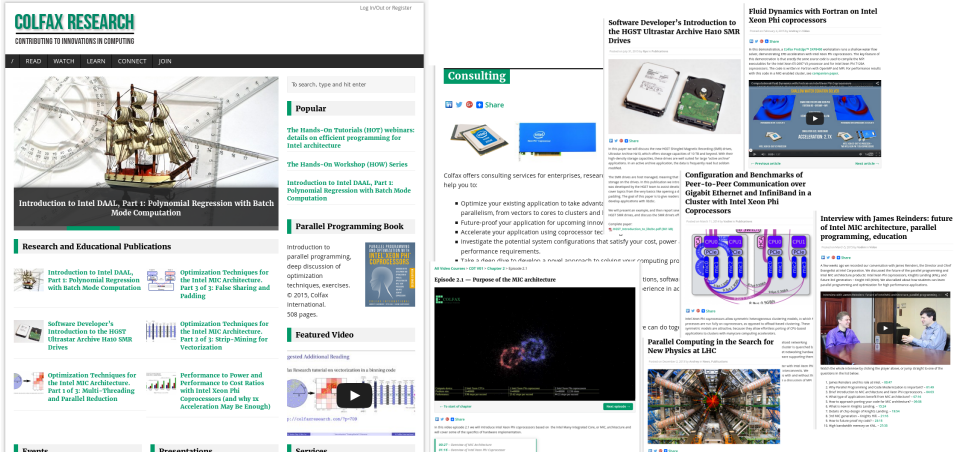
Parallel Programming
and Optimization with
Intel® Xeon Phi™
Coprocessors

Handbook on the Development and
Optimization of Parallel Applications
for Intel® Xeon® Processors
and Intel® Xeon Phi™ Coprocessors



http://xeonphi.com/book

Colfax Research



http://colfaxresearch.com/

(already registered? get $10 off the book)

HOW Online

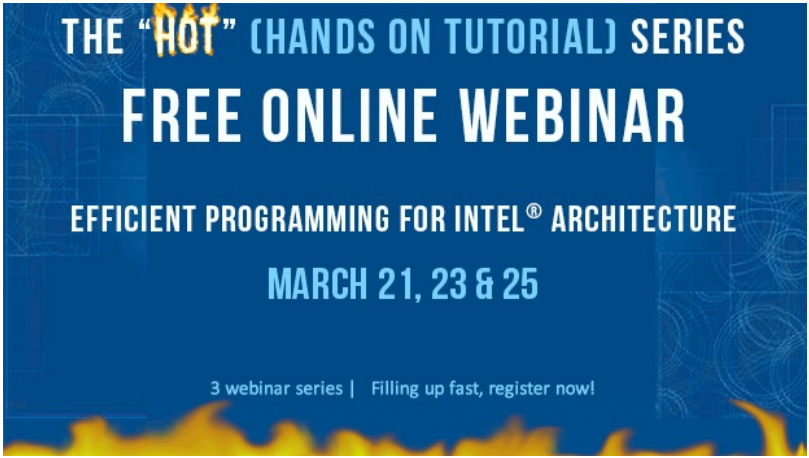
Course page: colfaxresearch.com/how-series

- Slides (including this one), code downloads
- Video of recorded sessions
- Chat (during webinars or offline)

Additional resources:

- More workshops like this one: colfaxresearch.com/how-series
- Video courses: colfaxresearch.com/video-courses
- Intel Many Integrated Core Architecture Forum

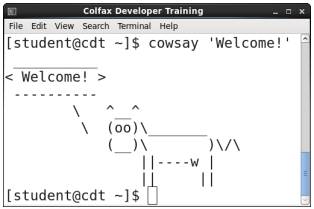
HOT Series



colfaxresearch.com/hot-16-03/

Hands-On Exercises and Remote Access

- 96 people receive a remote access token
- Can access the system the entire 3 weeks of the workshop



- Not among the 96? Stay tuned: follow along with instructor, use own system, or wait for a seat
- Use it or lose it: if you do not log in for a while, remote access token goes to next student on the list

§2. Intel Architecture

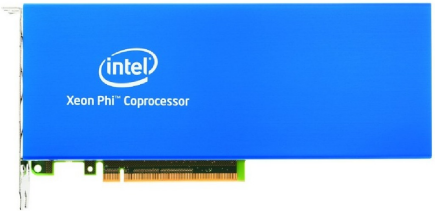
Computing Platforms

Computing Platforms

Intel Xeon Processor

Current: Haswell
Upcoming: Broadwell

Intel Xeon Phi Corocessor, 1st generation



Current: Knights Corner (KNC)

Intel Xeon Phi Processor, 2nd generation*

* socket and coprocessor versions

Upcoming: Knights Landing (KNL)

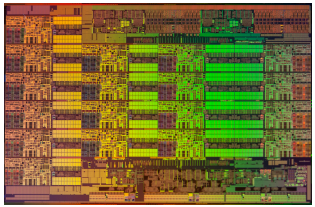
Multi-Core Architecture

Intel Many Integrated Core (MIC) Architecture

Intel Xeon CPU: Purpose and Specifications

General-purpose platform for demanding computing applications.

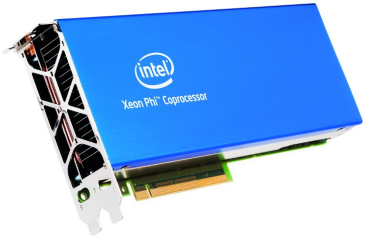
- Up to ~ 1 TFLOP/s in DP
- Up to ~ 2 TFLOP/s in SP
- Up to 768 GiB DDR4 RAM
- ~ 126 GB/s bandwidth
- Hardware-rich: forgiving of sub-optimal code

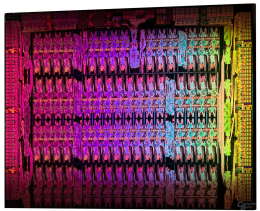


Intel Xeon Phi Processors (1st Gen)

Specialized platform for demanding computing applications.

- PCIe end-point device
- ~ 1.2 TFLOP/s in DP
- ~ 2.4 TFLOP/s in SP
- Up to 16 GiB GDDR5 RAM
- ~ 176 GB/s bandwidth
- Heterogeneous clustering
- Runs special Linux distribution

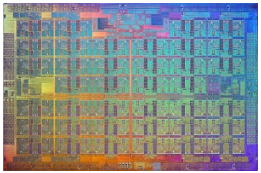




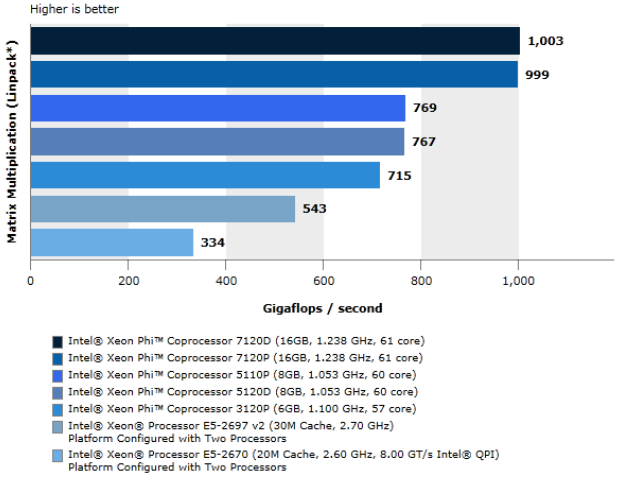
Intel Xeon Phi Processors (2nd Gen)

Specialized platform for demanding computing applications.

- Socket version or coprocessor
- 3+ TFLOP/s in DP
- 6+ TFLOP/s in SP
- Up to 16 GiB MCDRAM
- ~ 400 GB/s MCDRAM bandwidth
- Up to 384 GiB DDR4 RAM
- ~ 90 GB/s DDR4 bandwidth
- Supports common OS
- Public disclosures

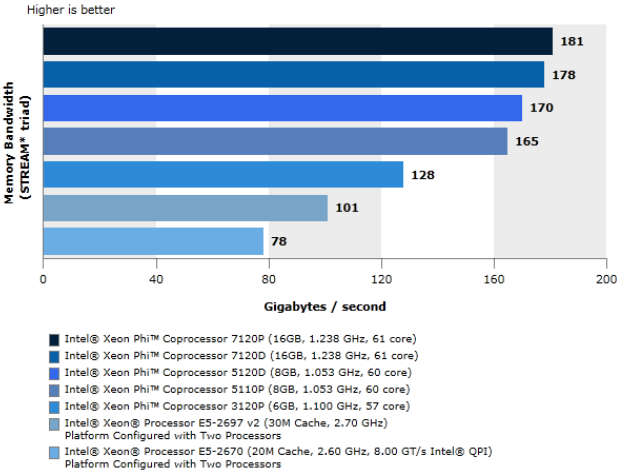


Intel Xeon Phi Coprocessors: HPC LINPACK Benchmark



Source: "Intel Xeon Phi Coprocessor LINPACK* and STREAM* Performance"

Intel Xeon Phi Coprocessors: STREAM Benchmark

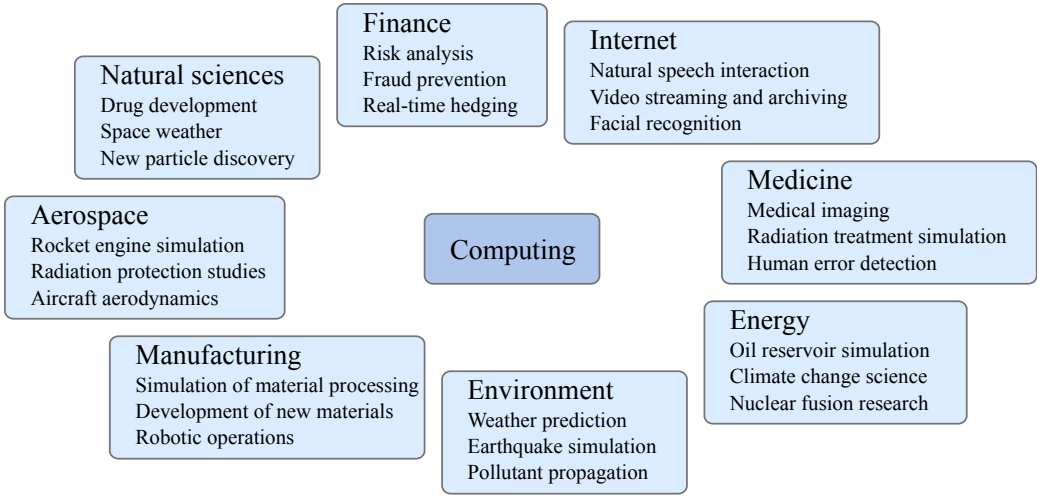


Source: "Intel Xeon Phi Coprocessor LINPACK* and STREAM* Performance"

It Is All About Performance

Computing Applications

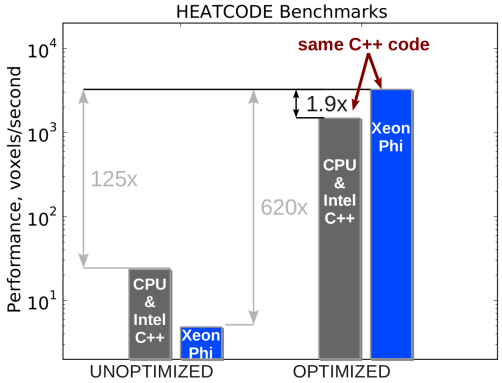
Just some examples



Will My Code Run Faster on KNL?

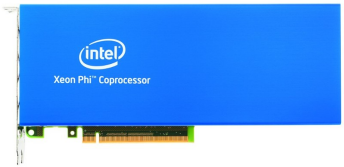
Performance on MIC architecture is a function of optimization Level

- Performance will be disappointing if code is not optimized for multi-core CPUs
- Optimized code runs better on the MIC platform *and* on the multi-core CPU
- Single code for two platforms + Ease of porting = Incremental optimization



See this case study

Coprocessor vs Processor Performance



vs.

One Intel Xeon Phi 7120P coprocessor

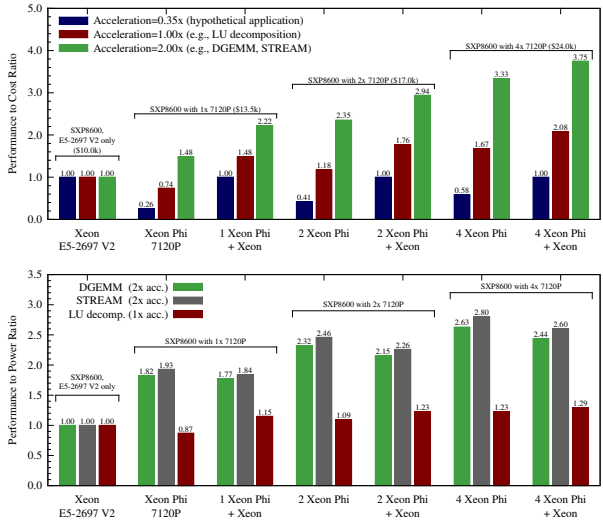
Two Intel Xeon E5-2697 v2 CPUs

- Why compare 1 coprocessor agains 2 processors?
  Same thermal design power (TDP).

See also "Intel Xeon Product Family: Performance Brief"

What is Your Performance Metric?

- Performance per System
- Performance per Watt
- Performance/Cost Ratio

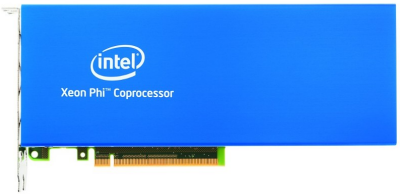


See this paper for details

Common Architecture, Programming Models

Bird's Eye View

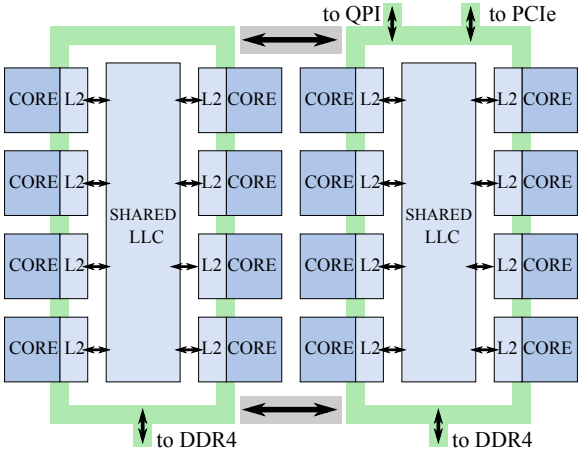
- C/C++/Fortran; OpenMP/MPI
- Standard Linux OS
- Up to 768 GiB of DDR4 RAM
- ≤18 cores/chip ≈3 GHz
- 2 hyper-threads per core
- 256-bit AVX vectors



- C/C++/Fortran; OpenMP/MPI
- Special Linux distribution
- 3–16 GiB cached GDDR5 RAM
- Up to 61 cores at ≈1.2 GHz
- 4 hardware threads per core
- 512-bit IMCI vectors

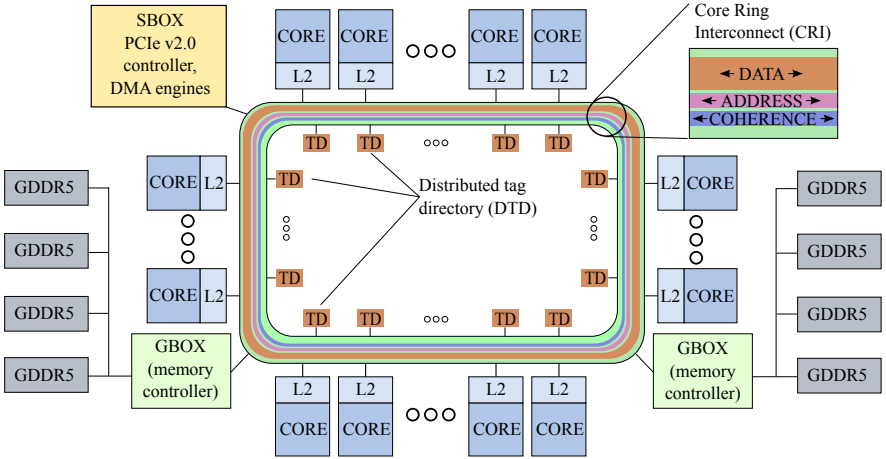
Intel Xeon CPU: Die Organization

Likes data locality, but large LLC is forgiving.



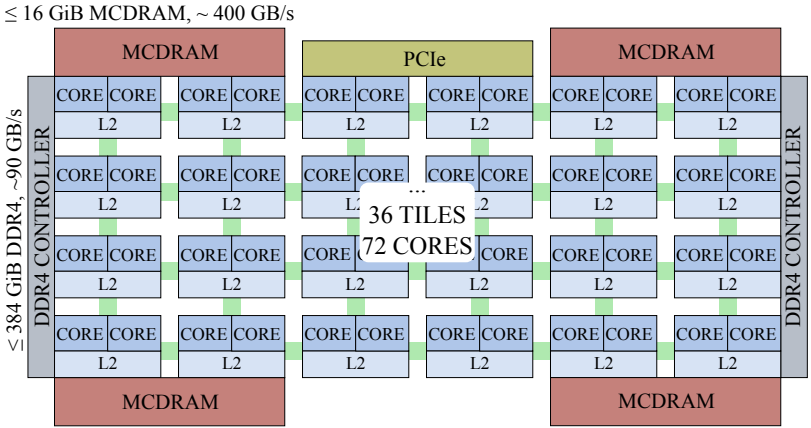
KNC Die Organization

In a ring bus with distributed cache, data access locality is key.



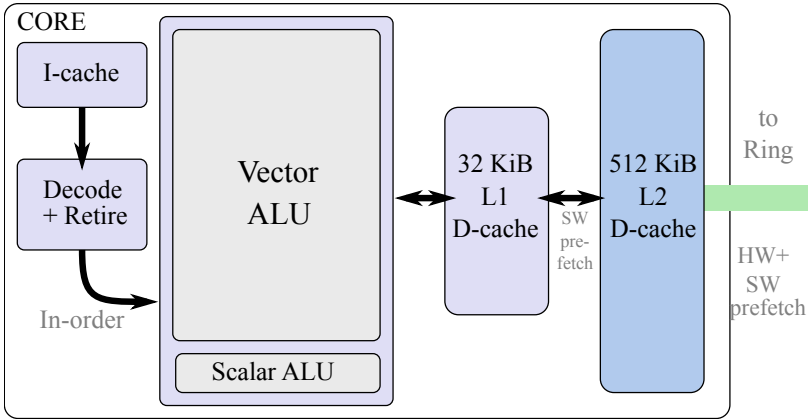
KNL Die Organization

- Mesh interconnect relaxes data locality requirement [somewhat]
- All-to-all, quadrant or sub-numa domain communication in mesh



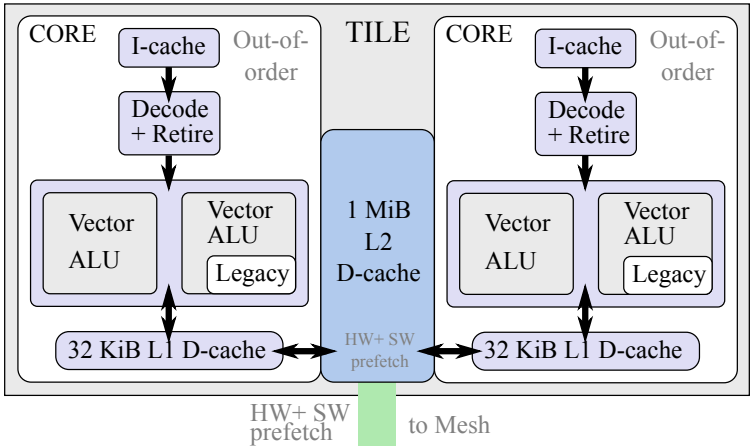
KNC Cores

Computing power is in vector units. Scalar support only for legacy usage.

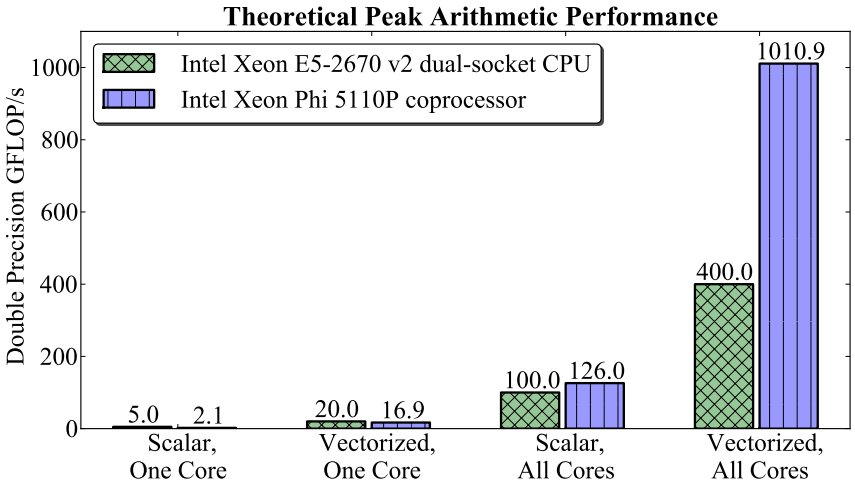


KNL Cores

- Even more power in vector units
- Binary compatible with Xeon, but in legacy mode

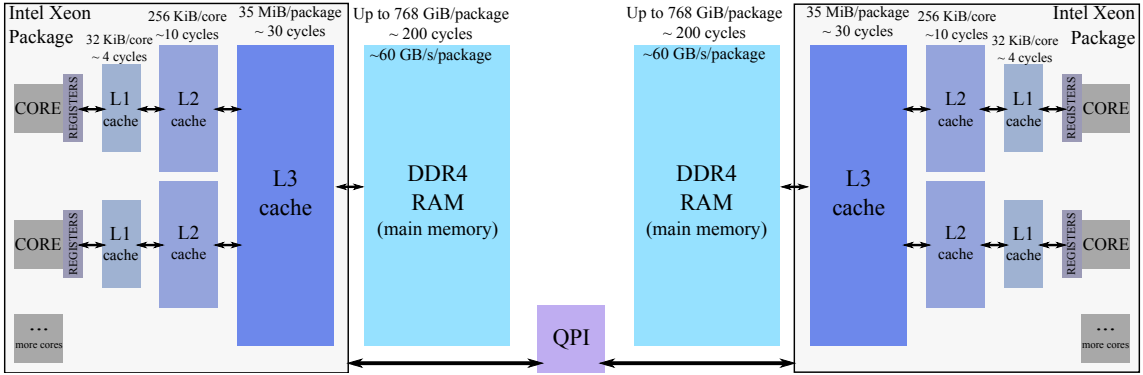


Task and Data Parallelism



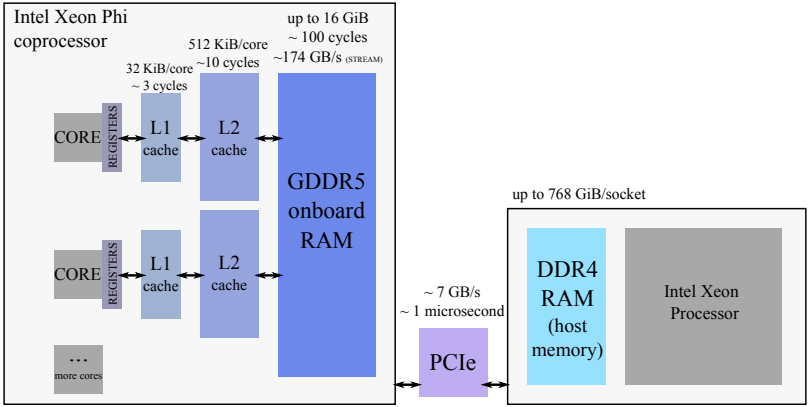
Intel Xeon CPU: Memory Organization

- Hierarchical cache structure
- Two-way processors have NUMA architecture



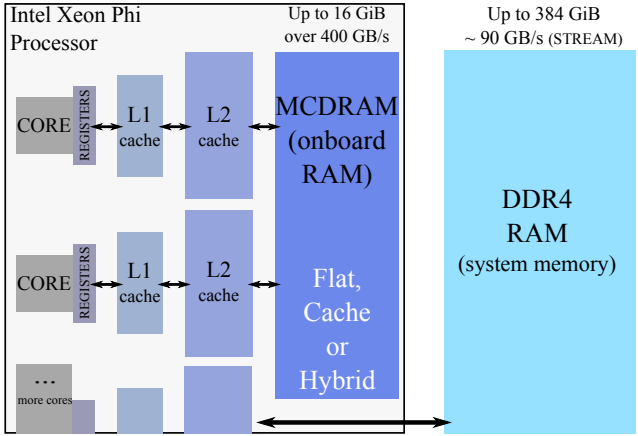
KNC Memory Organization

- Direct access to ≤ 16 GiB of cached GDDR5 memory on board
- No access to system DDR4, connected to host via PCIe

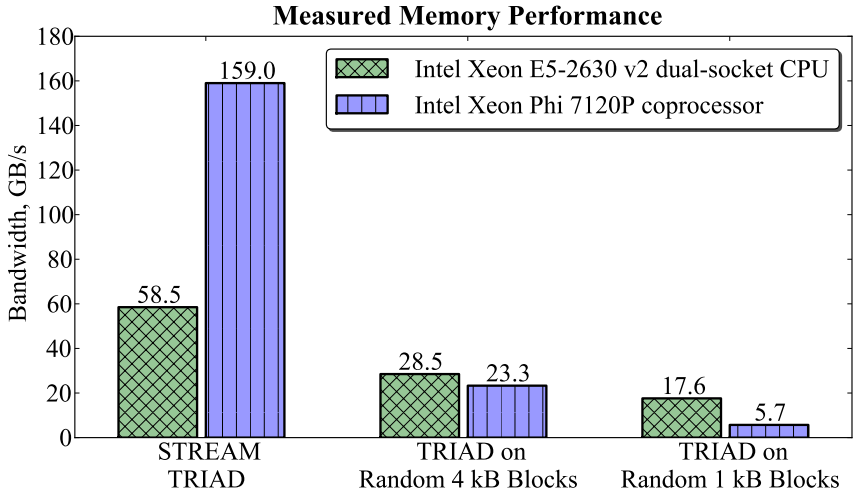


KNL Memory Organization

- Direct access to onboard MCDRAM *and* system DDR4
- Use MCDRAM as cache, in flat mode, or as hybrid



Memory Access Pattern



What's New in Knights Landing

KNL: What's New

	Haswell	**KNC**	**KNL**
Form factor	Socket	PCIe	PCIe, Socket
Core	Out-of-order	In-order	Out of order
Vectors	≤ AVX2	IMCI	≤ AVX2 + AVX-512
Memory	DDR4	GDDR5	MCDRAM+DDR4
RDMA	Over PCIe	Over PCIe	PCIe or Integrated
Programming	Traditional	Native, Offload	Traditional, Offload
Performance, Optimization	High and forgiving	Higher; needs good code	3x KNC 1-threaded and parallel

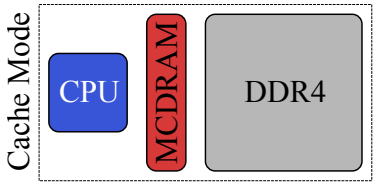
Using High-Bandwidth Memory (MCDRAM)

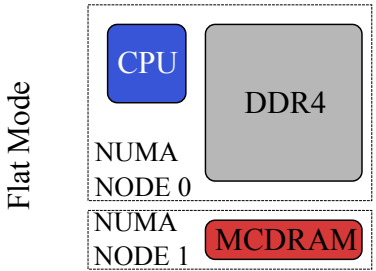
Option 1 : cache/hybrid mode

- Treat it as LLC
- Data locality techniques
- Miss latency 2x the direct DDR4 access

Option 2 : flat mode

- Application fits in 16 GiB? `numactl`
- More than 16 GiB data? Use special allocators (e.g., `memkind`)





Using New Vector Features

Efficient gather/scatter, conflict detection, high-precision exponential and reciprocal, two vector units per core...

Option 1 : automatic vectorization

- No changes to code, possibly adjust to 512-bit vector width
- Recompile with `-xMIC-AVX512`

Option 2 : explicit vectorization

- Bite the bullet

See also this post

Bottom Line

The best way to prepare...

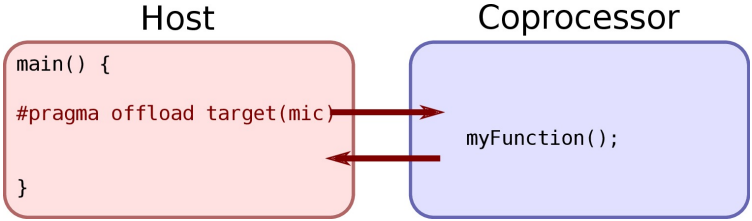
KEEP
CALM
AND
GO
PARALLEL

§3. Programming Coprocessors

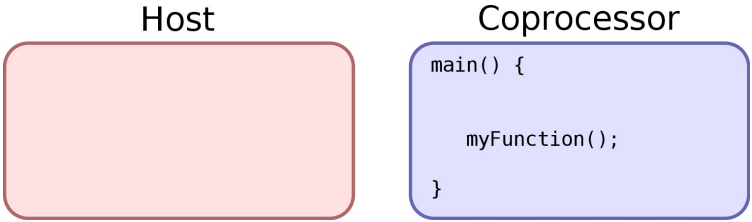
Offload and Native Models

Offload and Native Models

- Offload model (explicit/virtual-shared memory/OpenMP 4.0):



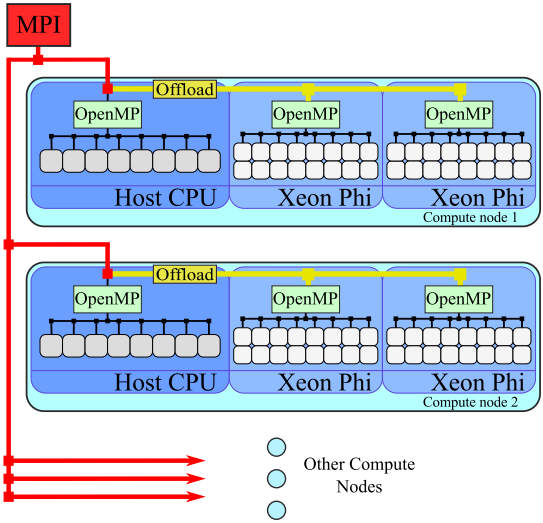
- Native model (standalone application/MPI process):



Heterogeneous Distributed Computing with Xeon Phi

Option 1: MPI+OpenMP with Offload.

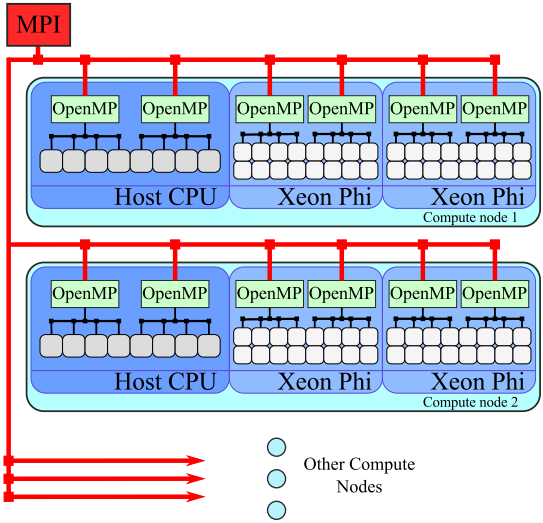
- MPI processes are multi-threaded with OpenMP.
- MPI runs only on CPUs.
- MPI processes offload to coprocessor(s).
- OpenMP in offload regions.



Heterogeneous Distributed Computing with Xeon Phi

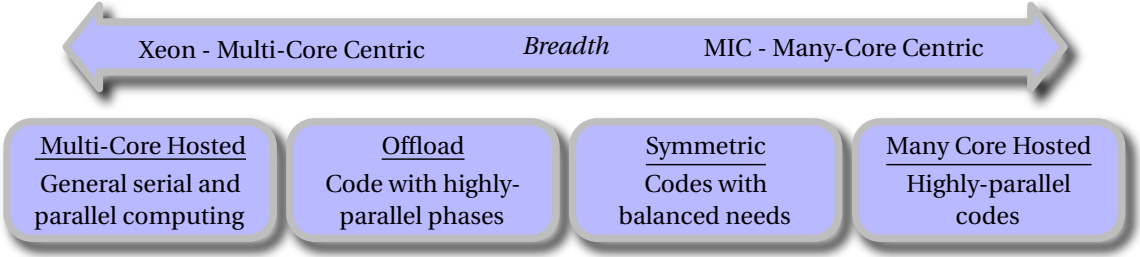
Option 2: Symmetric hybrid MPI+OpenMP.

- MPI processes on hosts
- Native MPI processes on the coprocessor.
- Multi-threading with OpenMP.



Teaming Xeon Processors with Xeon Phi Coprocessors

Programming models allow a range of CPU+MIC coupling modes



Software Configuration

Intel Manycore Platform Software Stack

- `micinfo` – system information
- `micsmc` – monitor and modify the physical paramaters: temperature, power modes, core utilization, etc.
- `micctrl` – configure the Intel Xeon Phi coprocessor operating system
- `miccheck` – verify the Intel Xeon Phi coprocessor configuration
- `micrasd` – log of hardware errors reported by Intel Xeon Phi coprocessors
- `micflash` – flash memory agent

Monitoring MIC activity with `micsmc` (an MPSS tool)

Linux Environment on Intel Xeon Phi Coprocessors

```
vega@lyra% lspci | grep -i "co-processor"
06:00.0 Co-processor: Intel Corporation Xeon Phi coprocessor 7120 series (rev 20)
82:00.0 Co-processor: Intel Corporation Xeon Phi coprocessor 7120 series (rev 20)
vega@lyra% sudo service mpss status
mpss is running
vega@lyra% cat /etc/hosts | grep mic
172.31.1.1  lyra-mic0 mic0
172.31.2.1  lyra-mic1 mic1
vega@lyra% ssh mic0

vega@mic0% cat /proc/cpuinfo | grep proc | tail -n 3
processor: 241
processor: 242
processor: 243
vega@mic0% ls /
amplxe   dev    home   lib64    oldroot  proc   sbin     sys    usr
bin      etc    lib    linuxrc  opt      root   sep3.10  tmp    var
```

Software Necessary to Build Xeon Phi Applications:

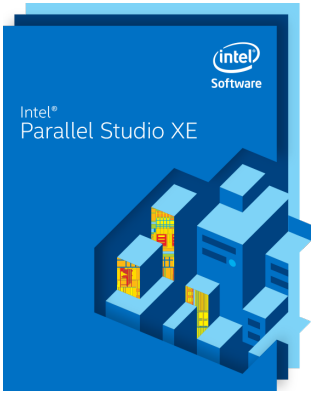
Compilers : Intel C Compiler, Intel C++ Compiler, and Intel Fortran Compiler — mandatory

Optimization tools : Intel VTune Amplifier XE and Intel Trace Analyzer and Collector (ITAC) — highly recommended

Mathematics support : Intel Math Kernel Library (MKL) — highly recommended

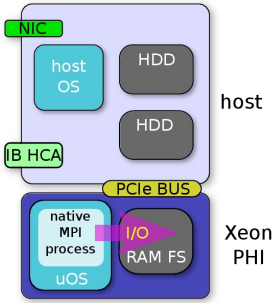
Cluster Development : Intel MPI — industry standard parallel framework

Development : Intel Inspector XE, Intel Advisor XE — optional

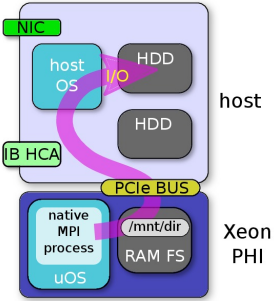


All-in-One bundle,
Intel Parallel Studio XE

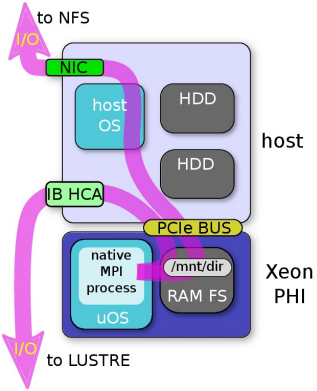
Working with Files on Coprocessors



RAM Filesystem



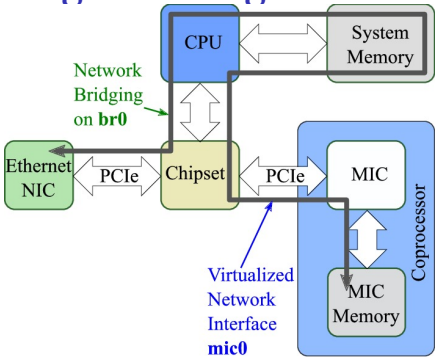
VirtIO Transfer

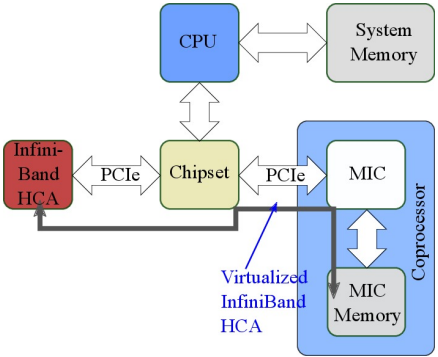


Network Storage

Details: http://xeonphi.com/papers/io

Bridged Configuration for Peer-to-Peer Communication





- Left: Gigabit Ethernet bridging and TCP/IP virtualization
- Right: InfiniBand on Xeon Phi with the Coprocessor Communication Link (CCL) technology
- Details: `http://xeonphi.com/papers/p2p`

§4. Hands-On Part: Native Programming

Native Execution

"Hello World" application:

```
1 #include <cstdio>
2 #include <unistd.h>
3 int main(){
4     printf("Hello world! I have %ld logical cores.\n",
5     sysconf(_SC_NPROCESSORS_ONLN ));
6 }
```

Compile and run on host:

```
vega@lyra% icpc hello.cc
vega@lyra% ./a.out
Hello world! I have 48 logical cores.
vega@lyra%
```


Native Execution

Compile and run the same code on the coprocessor in the native mode:

```
vega@lyra% icpc hello.cc -mmic
vega@lyra% scp a.out mic0:~/
a.out 100% 10KB 10.4KB/s 00:00
vega@lyra% ssh mic0
vega@mic0% pwd
/home/lyra
vega@mic0% ls
a.out
vega@mic0% ./a.out
Hello world! I have 244 logical cores.
vega@mic0%
```

- Use `-mmic` to produce executable for MIC architecture
- Must transfer executable to coprocessor (or NFS-share) and run from shell
- Native MPI applications work the same way (need Intel MPI library)

Alternative Native Application Launcher: micnativeloadex

The tool micnativeloadex automatically transfers code and dependent libraries and runs the application.

```
vega@lyra% export SINK_LD_LIBRARY_PATH=/opt/intel/composerxe/compiler/lib/mic
vega@lyra% icpc hello.cc -mmic
vega@lyra% micnativeloadex a.out
Hello world! I have 244 logical cores.
vega@lyra%
```

- Set SINK_LD_LIBRARY_PATH to help the tool find libraries
- Do not have to SSH into the coprocessor
- Runs under micuser on coprocessor

Native Applications with Autotools

- Use the Intel compiler with flag `-mmic`
- Eliminate assembly and unncecessary dependencies
- Use `--host=x86_64` to avoid "program does not run" errors

Example, the GNU Multiple Precision Arithmetic Library (GMP):

```
vega@lyra% wget https://ftp.gnu.org/gnu/gmp/gmp-5.1.3.tar.bz2
vega@lyra% tar -xf gmp-5.1.3.tar.bz2
vega@lyra% cd gmp-5.1.3
vega@lyra% ./configure CC=icc CFLAGS="-mmic" --host=x86_64 --disable-assembly
...
vega@lyra% make
...
```


Review and What's Next

- Intel Xeon and Intel Xeon Phi – parallel processors
- Xeon Phi (MIC architecture) – specialized for highly parallel workloads without complex memory access
- Coprocessor – either offload device or an additional compute node
- Native+offload programming allow for a range of design options

Next session: details of the offload model.