



Developer's Guide to Knights Landing

Introduction to 2nd Generation
Intel[®] Xeon Phi[™] Processors

Colfax International — @colfaxintl

April 2016 - Rev. 1.2

About This Document

This document represents the materials of a Web-based training “Introduction to 2nd Generation Intel® Xeon Phi™ Processors: Developer’s Guide to Knights Landing” developed and run by Colfax International.

© Colfax International, 2013-2016

Parallel Programming Boot Camp (1-Day) / Workshop (4-Days)



Instructor-led 1-day or 4-days training, at your office or at Colfax facility in Sunnyvale, CA

[Click here to learn more](#)

1-Day Parallel Programming Boot Camp

For software engineers and architects, providing an overview of parallel programming frameworks and optimization guidelines for multi-core CPUs (Intel® Xeon®) and many-core coprocessors (Intel® Xeon Phi™):

- Discussions about three layers of parallelism: SIMD, Threads, Cluster environment
- Tips for quick porting/development of HPC software applications
- Real-life examples of code and optimization techniques
- Hardware solution and corresponding software implementations, APIs, and frameworks

4-Days Parallel Programming Workshop

For the developer who wants to hit the ground running with the modern multi-core CPUs (Intel® Xeon®), many-core coprocessors (Intel® Xeon Phi™) and leading software development tools:

- Hardware installation
- MPSS tools and the Linux environment on the Intel® Xeon Phi™ coprocessor
- Exploring differences in serial vs. parallel programming / processing / hardware usage
- Accelerated clusters
- Optimizations of vector arithmetics, memory traffic, thread parallelism and communication
- Using the Intel® Math Kernel Library

[Register Now!](#)

colfaxresearch.com/knl-webinar/

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.

Resources

COLFAX RESEARCH
Log In/Out or Register

CONTRIBUTING TO INNOVATIONS IN COMPUTING

/ READ WATCH LEARN CONNECT JOIN



To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why xx Acceleration May be Enough)




Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises.

© 2015, Colfax International. 506 pages.

Featured Video

Additional Reading



See Research material on vectorization in a streaming code

<http://colfaxresearch.com/?p=708>

Events
Documentation
Carriers

Consulting

Share

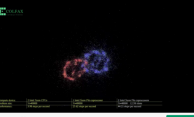



Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and clouds
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor technology
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a Hands-On Workshop: a virtual workshop to explore your computing problems, software experience in architecture, software development, and hardware design.

All Video Courses - COP 001 - Chapter 2 - Episode 2.1

Episode 2.1 — Purpose of the MIC architecture



In this video episode 2.1 we will introduce Intel Xeon Phi coprocessors based on the Intel Many Integrated Core, or MIC, architecture and will discuss some of the architecture's features.


Download Intel Architecture

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives



Colfax offers a 10-day Hands-On Workshop (HOW) series, introducing the parallel programming capabilities of the architecture. This episode is the first of the series and will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives. The series will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives.

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors




Colfax offers a 10-day Hands-On Workshop (HOW) series, introducing the parallel programming capabilities of the architecture. This episode is the first of the series and will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives.

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors




Colfax offers a 10-day Hands-On Workshop (HOW) series, introducing the parallel programming capabilities of the architecture. This episode is the first of the series and will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives.

Interview with James Reinders: future of Intel MIC architecture, parallel programming, education



Colfax offers a 10-day Hands-On Workshop (HOW) series, introducing the parallel programming capabilities of the architecture. This episode is the first of the series and will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives.

Parallel Computing in the Search for New Physics at LHC



Colfax offers a 10-day Hands-On Workshop (HOW) series, introducing the parallel programming capabilities of the architecture. This episode is the first of the series and will explore the basic architecture for the new HaaS SMR drives and the new HaaS SMR drives.

<http://colfaxresearch.com/>
 (already registered? **get \$10 off the book**)

HOW Series



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

STARTS MAY 23

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

colfaxresearch.com/how-series

Developer Access Program (DAP)

Can't wait to get your hands on Knights Landing?



Find out more at dap.xeonphi.com
or contact us at dap@colfax-intl.com.

§2. Intel Architecture: Today and Tomorrow

Intel Architecture

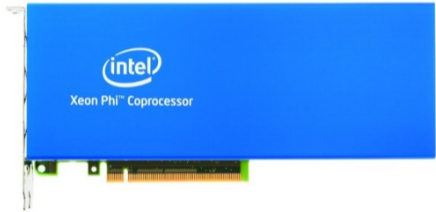
Intel Xeon Processor



Current: Broadwell
Upcoming: Skylake

Multi-Core Architecture

Intel Xeon Phi Coprocessor, 1st generation



Current: Knights Corner (KNC)

Intel Xeon Phi Processor, 2nd generation*



* socket and coprocessor versions

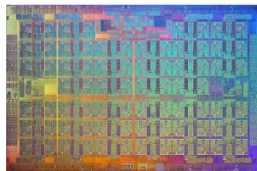
Upcoming: Knights Landing (KNL)

Intel Many Integrated Core (MIC) Architecture

Intel Xeon Phi Processors (2nd Gen)

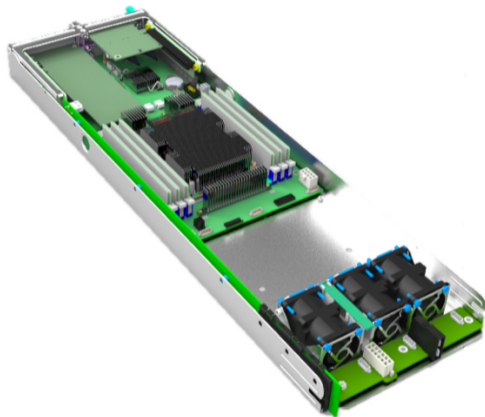
2nd Generation of Intel Many Integrated Core (MIC) Architecture.
Specialized platform for demanding computing applications.

- Bootable host processor or coprocessor
- 3+ TFLOP/s DP
- 6+ TFLOP/s SP
- Up to 16 GiB MCDRAM
- MCDRAM bandwidth $\approx 5x$ DDR4
- Binary compatible with Intel Xeon
- **Public disclosures**



Standard CPU Form Factor

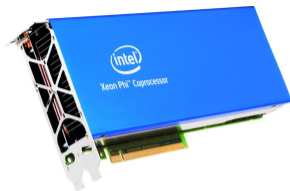
- **Bootable Host Processor**
 - ▶ No need for “host”. OS runs on KNL processor.
 - ▶ Supports common OS.
 - ▶ No more PCIe bottleneck!
- **Direct access to ≤ 384 GiB DDR4 RAM**
 - ▶ Up to ≈ 90 GB/s DDR4 bandwidth
- **Access to PCIe Bus**



Future Releases

KNLF: KNL with Fabric

- Fabric integrated on CPU
 - ▶ Intel OmniPath Architecture
- Socket mount processor



*KNC image

KNL Coprocessor

- PCIe add-in card
 - ▶ Requires host
- Multiple KNLs in a system

§3. Cores and Threading

Importance of Parallelism

If KNL was a steam engine...



It would have a lot of furnaces.

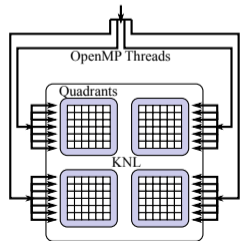
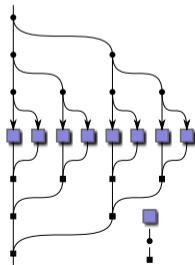
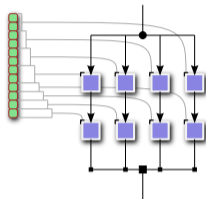
***Single-threaded code on KNL
is like having 1 fireman service 72 steam engines***

Implementing Multi-threading

You have to use a core to benefit from it!

Threading frameworks:

- OpenMP
- TBB
- Cilk Plus
- Pthreads
- Hybrid with MPI
- Any others supported by Xeon Processors

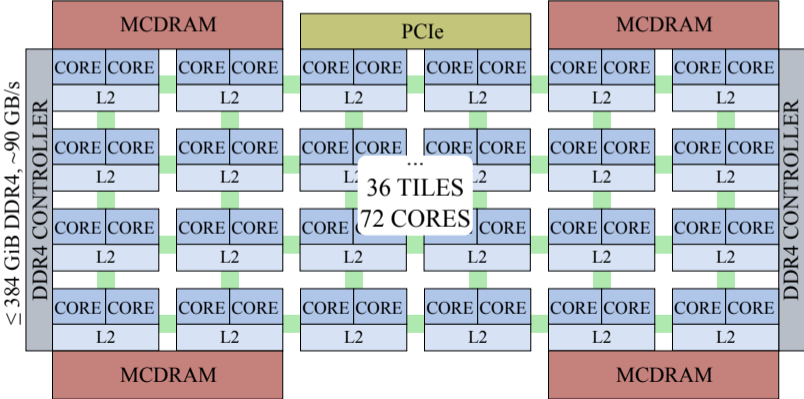


Features of KNL Cores

KNL Die Organization: Tiles

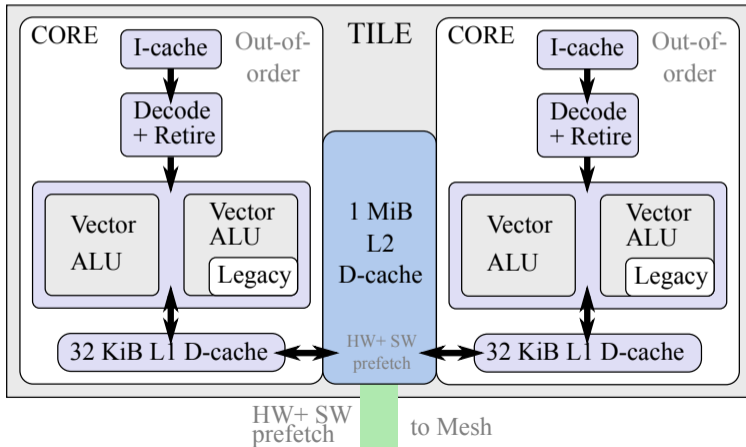
- Up to 36 tiles, each with 2 physical cores (72 total).
- Distributed L2 cache across a mesh interconnect.

≤ 16 GiB MCDRAM, ~ 400 GB/s



KNL Cores

- 4-way hyper-threading (up to $4 \times 72 = 288$ logical processors)
- L2 cache shared by 2 cores on a tile.



More Forgiving Cores than First Generation (KNC)

Based on Intel[®] Atom cores (Silvermont microarchitecture)

- **Out-of-order cores:**

Better latency masking from long latency operations

- **Back-to-back instructions from single thread:**

Thread count requirement reduced to ≈ 70 (from ≈ 120 for KNC)

- **Advanced branch prediction:**

Fewer cycles wasted to branch misprediction

Generally more forgiving to non-optimized code.

Performance Considerations

Affinity and cpubinfo

Neighboring threads often work on nearby/same locations in memory.

→ Use thread pinning to have them share L2 cache!

You can use `cpubinfo` (part of Intel[®] MPI) to find which cores share cache.

```
user@knl% cpubinfo
// ... cpubinfo output ... //
L2 1  MB (0,1,64,65,128,129,192,193) (2,3,66,67,130,131,194,195) (4,5,68, ...
```

Use `KMP_AFFINITY` for Intel Compilers or `OMP_PROC_BIND` for GCC compilers.

```
user@knl% export KMP_AFFINITY=compact
user@knl% export OMP_PROC_BIND=close
```


§4. Vectorization

Scalar Code and Our Fireman



*Not using vector instructions in KNL
is like feeding a steam engine with a spoon.*

Short Vector Support

Vector instructions – one of the implementations of SIMD (Single Instruction Multiple Data) parallelism.

Scalar Instructions

$$\begin{array}{r} 4 + 1 = 5 \\ 0 + 3 = 3 \\ -2 + 8 = 6 \\ 9 + -7 = 2 \end{array}$$

Vector Instructions

$$\begin{array}{r} 4 \\ 0 \\ -2 \\ 9 \end{array} + \begin{array}{r} 1 \\ 3 \\ 8 \\ -7 \end{array} = \begin{array}{r} 5 \\ 3 \\ 6 \\ 2 \end{array}$$

↑
Vector Length
↓

Vector Instructions on KNL

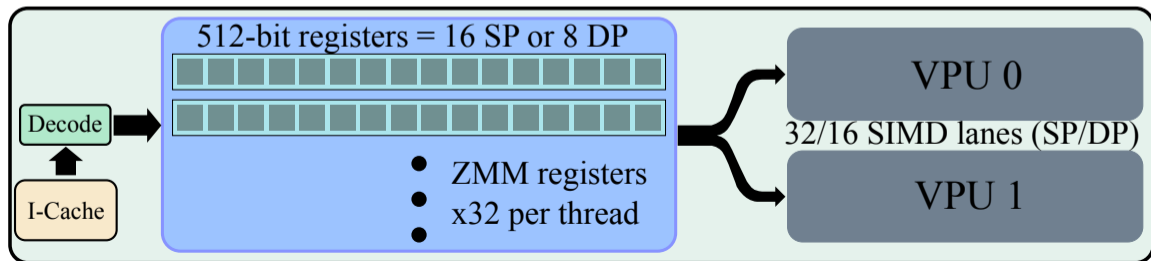
Dual VPU

Each core on KNL has two Vector Processing Units (VPUs).

Penalty from non-vectorized code

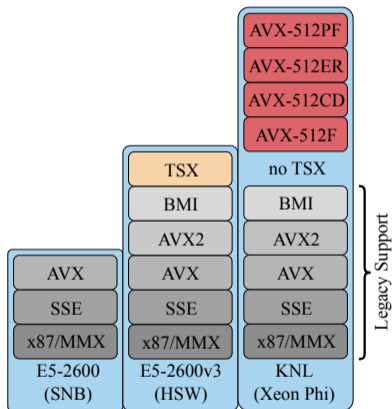
SP → 512 bit registers / 32 bits × 2 VPUs = **32** SIMD lanes

DP → 512 bit registers / 64 bits × 2 VPUs = **16** SIMD lanes



Supported Instruction Sets on KNL

- Intel[®] Advanced Vector Extensions 512 (AVX-512)
 - ▶ 512-bit vector registers.
 - ▶ Hardware gather/scatter, DP transcendental functions support and more.
 - ▶ Supported by non-Intel compilers like GCC.
- \leq Intel[®] AVX2
 - ▶ Legacy mode operation.
 - ▶ Binary compatibility with Xeon.
 - ▶ Does *not* include IMCI (from KNC).



AVX-512 Features

Knights Landing: first AVX-512 processor

- AVX-512F (Fundamentals)
 - Extension of most AVX2 instructions to 512-bit vector registers.
- AVX-512CD (Conflict Detection)
 - Efficient conflict detection (application: binning).
- AVX-512ER (Exponential and Reciprocal)
 - Transcendental function (exp, rcp and rsqrt) support.
- AVX-512PF (Prefetch)
 - Prefetch for scatter and gather.

Checking for AVX-512 support

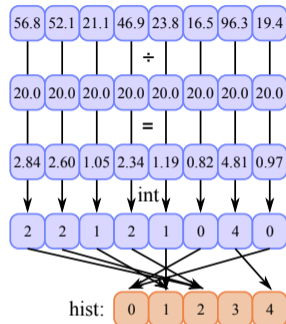
Check /proc/cpuinfo for AVX-512 flags

```
user@knl% cat /proc/cpuinfo
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
aperfmpperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 fma
cx16 xtpr pdcm sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer xsave avx
f16c rdrand lahf_lm abm 3dnowprefetch arat epb xsaveopt pln pts dtherm
tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 avx2 smep bmi2
erms avx512f rdseed adx avx512pf avx512er avx512cd
```

Also possible through C/C++ function calls: see [this blog](#).

AVX-512CD: Histograms

```
1 // worker.cc from Colfax lab 4.04
2 void Histogram(const float* age, int* const hist,
3               const int n, const float group_width,
4               const int m) {
5
6     for (int i = 0; i < n; i++) {
7         const int j = (int) ( age[i] / group_width );
8         hist[j]++;
9     }
10 }
```



```
user@knl% cat worker.optrpt
```

```
....
```

```
remark ...: vectorization support: scatter was generated for the variable hist:
remark ...: vectorization support: gather was generated for the variable hist:
remark #15300: LOOP WAS VECTORIZED
```

AVX-512ER: Transcendental Functions

Support for transcendental functions.

- Exponential.
- Reciprocal.
- Inverse Square Root.

Better precision.

- Double Precision.
- Max relative error:
 - 2^{-23} (exp)
 - 2^{-28} (rcp and rsqrt)



Source: [APOD](#)

Programming Considerations

Using AVX-512: Two Approaches

Automatic Vectorization:

- Vectorization w/ compiler.
- Portable: just recompile.
- Tuning with directives.

```
1 double A[vec_width], B[vec_width];  
2 // ...  
3  
4  
5 // This loop will be auto-vectorized  
6 for(int i = 0; i < vec_width; i++)  
7     A[i]+=B[i];
```

```
1 double A[vec_width], B[vec_width];  
2 // ...  
3 // This is explicitly vectorized  
4 __m512d A_vec = _mm512_load_pd(A);  
5 __m512d B_vec = _mm512_load_pd(B);  
6 A_vec = _mm512_add_pd(A_vec, B_vec);  
7 _mm512_store_pd(A, A_vec);
```

Explicit Vectorization:

- Vectorization w/ intrinsics
- Full control over instructions
- Limited portability

Intel Compiler support for AVX-512

Intel Compiler versions ≥ 15.0 supports AVX-512 instruction set.

```
user@knl% icc -v
icc version 16.0.1 (gcc version 4.8.5 compatibility)
user@knl% icc -help
// ... truncated output ... //
-x<code>
    ...
    MIC-AVX512
    CORE-AVX512
    COMMON-AVX512
```

- `-xMIC-AVX512` : for KNL (supports F, CD, ER, PF)
- `-xCORE-AVX512` : for future Xeon (supports F, CD, DQ, BW, VL)
- `-xCOMMON-AVX512` : common to KNL and Xeon (supports F, CD)

GCC support for AVX-512

GCC \geq 4.9.1 supports AVX-512 instruction set.

```
user@knl% g++ -v
gcc version 4.9.2 (GCC)
user@knl% g++ foo.cc -mavx512f -mavx512er -mavx512cd -mavx512pf
```

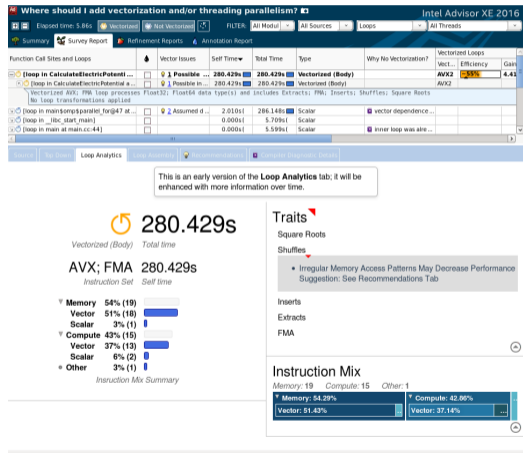
Basic automatic vectorization support: add `-O2` or `-O3`.

```
1 // ... foo.cc ... //
2 for(int i = 0; i < n; i++)
3   B[i] = A[i] + B[i];
```

```
user@knl% g++ -s foo.cc -mavx512f -O3
user@knl% cat foo.s
...
vmovapd -16432(%rbp,%rax), %zmm0
vaddpd -8240(%rbp,%rax), %zmm0, %zmm0
vmovapd %zmm0, -8240(%rbp,%rax)
```

Performance Considerations

Even if your code is vectorized, tuning may unlock more performance.



- Providing enough parallelism.
 - ▶ More consecutive vector operations required to overcome vectorization latency.
- Loop pipelining and unrolling.
 - ▶ Double the pipeline stages to populate.
- Better vectorization patterns.
 - ▶ Avoid long latency operations with unit-stride and unmasked operations.

§5. Memory Architecture

Where's the Fuel?

Cores can't do work if the data is not there.



Source: [wikipedia](https://en.wikipedia.org/wiki/Coal_train)

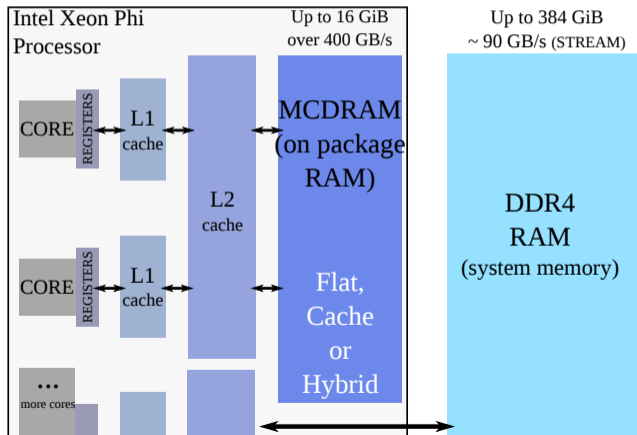


KNL memory must be used efficiently to feed the cores.

MCDRAM on KNL

KNL Memory Organization

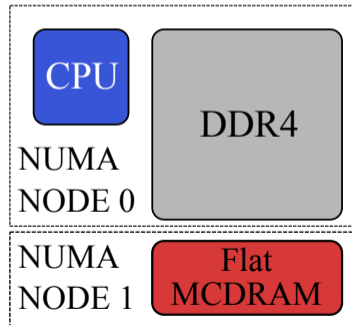
- Direct access to on package MCDRAM *and* system DDR4 (socket)
- Use MCDRAM as cache, in flat mode, or as hybrid



MCDRAM Memory Modes

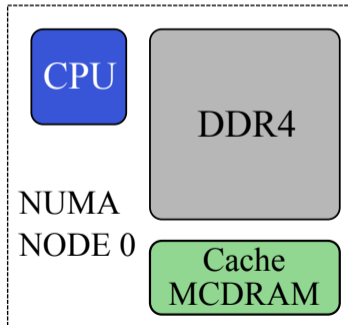
Flat Mode

- MCDRAM treated as a NUMA node
- Users control what goes to MCDRAM



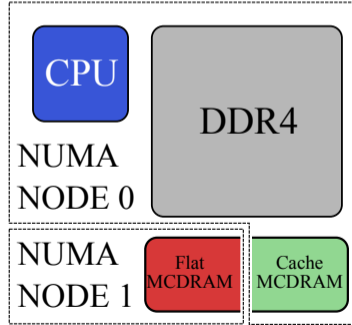
Cache Mode

- MCDRAM treated as a Last Level Cache (LLC)
- MCDRAM is used automatically



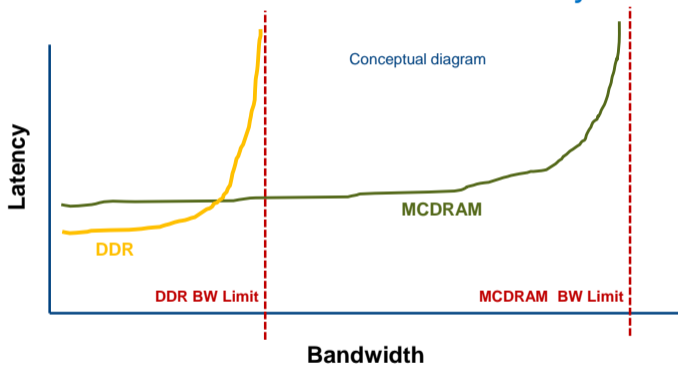
Hybrid Mode

- Combination of Flat and Cache
- Ratio can be chosen in the BIOS



MCDRAM Performance

DDR and MCDRAM Bandwidth vs. Latency



MCDRAM latency more than DDR at low loads but much less at high loads

Copyright © 2015, Intel Corporation. All rights reserved Avinash Sodani ISC 2015 Intel® Xeon Phi™ Workshop.



Source: Intel, [ISC 2015 KNL keynote \(pdf\)](#)

Programming with MCDRAM

Numactl

- Finding information about the NUMA nodes in the system.

```
user@knl% # In Flat mode with All-to-All
user@knl% numactl -H
available: 2 nodes (0-1)
node 0 cpus:  ... all cpus ...
node 0 size: 98207 MB
node 0 free: 94798 MB
node 1 cpus:
node 1 size: 16384 MB
node 1 free: 15991 MB
```

- Binding the application to MCDRAM (Flat/Hybrid)

```
user@knl% gcc myapp.c -o runme -mavx512f -O2
user@knl% numactl --membind 1 ./runme
// ... Application running on MCDRAM ... //
```

Memkind Library and hbwmalloc

Manual allocation to MCDRAM possible with hbwmalloc and Memkind Library.

```
1 #include <hbwmalloc.h>
2 const int n = 1<<10;
3 // Allocation to MCDRAM
4 double* A = (double*) hbw_malloc(sizeof(double)*n);
5 // No replacement for _mm_malloc. Use posix_memalign
6 double* B;
7 int ret = hbw_posix_memalign((void*) B, 64, sizeof(double)*n);
8 .....
9 // Free with hbw_free
10 hbw_free(A); hbw_free(b);
```

Fortran Allocations.

```
1 REAL, ALLOCATABLE :: A(:)
2 !DEC$ ATTRIBUTES FASTMEM :: A
3 ALLOCATE (A(1:1024))
```

Compilation with Memkind Library and hbwmalloc

To compile C/C++ applications:

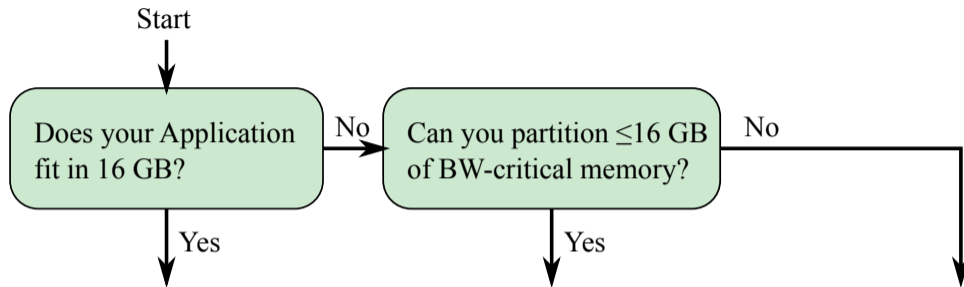
```
user@knl% icpc -lmemkind foo.cc -o runme
user@knl% g++ -lmemkind foo.cc -o runme
```

To compile Fortran applications:

```
user@knl% ifort -lmemkind foo.f90 -o runme
user@knl% gfortran -lmemkind foo.f90 -o runme
```

Open source distribution of Memkind library can be found at:
<http://memkind.github.io/memkind/>

Flow Chart for Bandwidth-Bound Applications



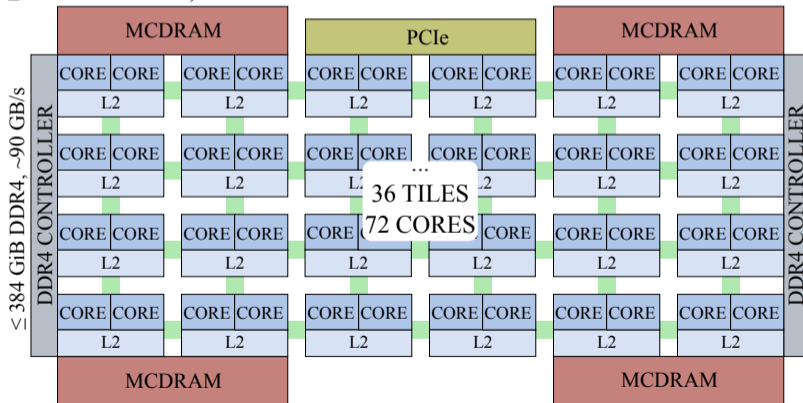
numactl	Memkind	Cache mode
<ul style="list-style-type: none">• Simply run the whole program in MCDRAM• No code modification required	<ul style="list-style-type: none">• Manually allocate BW-critical memory to MCDRAM• Memkind calls need to be added.	<ul style="list-style-type: none">• Allow the OS to figure out how to use MCDRAM• No code modification required

Cluster Modes on KNL

KNL Die Organization

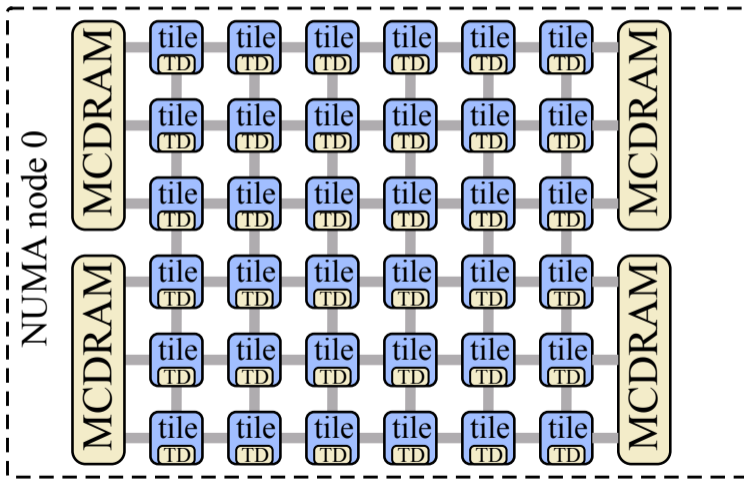
- Mesh interconnect relaxes data locality requirement [somewhat]
- All-to-all, quadrant or sub-numa domain communication in mesh

≤ 16 GiB MCDRAM, ~ 400 GB/s



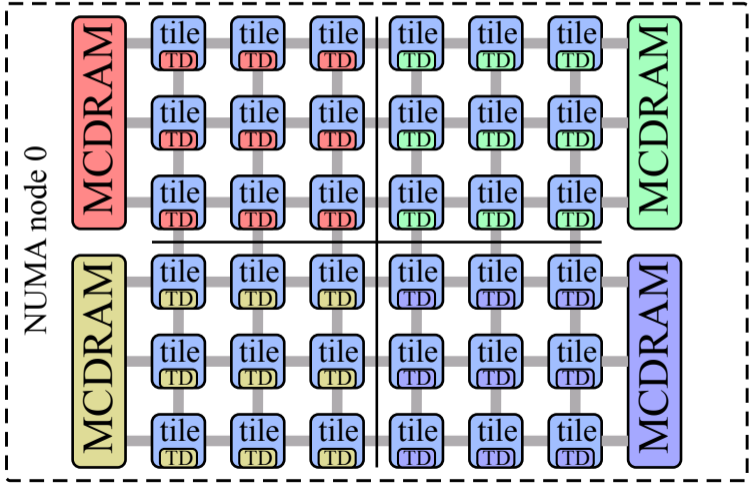
Cluster Modes: All-to-All

No affinity between the distributed Tag Directory (TD) and memory.



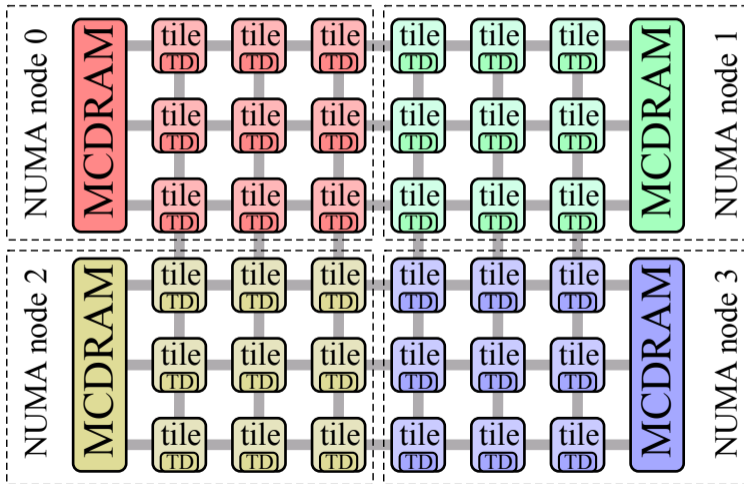
Cluster Modes: Quadrant/Hemisphere

Tag Directory (TD) and memory reside in the same quadrant.



Cluster Modes: SNC-4/SNC-2

Will appear as 4 NUMA nodes (similar to quad-socket system).



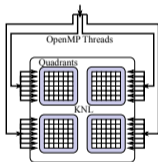
Programming with Cluster Modes

How to Use Cluster modes

Thread affinity to match the memory/directory affinity.

Nested OpenMP

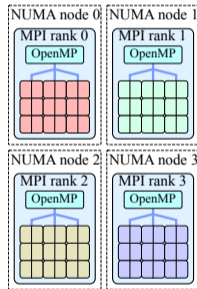
```
1 #pragma omp parallel
2 {
3     // ...
4     #pragma omp parallel
5     {
6         // ...
7     }
8 }
```



```
user@knl% OMP_NUM_THREADS=4,72
user@knl% OMP_NESTED=1
```

MPI+OpenMP

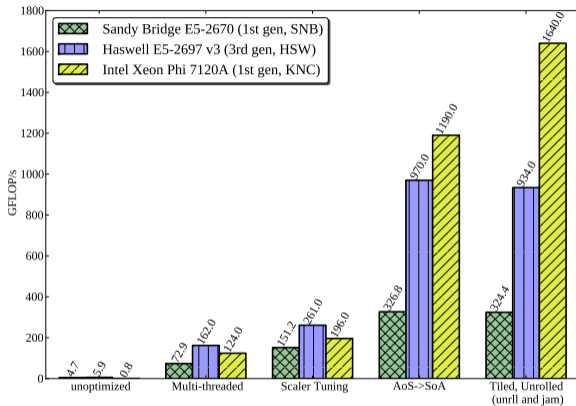
```
1 stat = MPI_Init();
2 // ...
3 #pragma omp parallel
4 {
5     // ...
6 }
7 // ...
8 MPI_Finalize();
```



```
user@knl% mpirun -host knl \
> -np 4 ./myparallel_app
```

§6. Importance of Code Optimization

N-body Simulation on Intel Architecture



Get Ready

The best way to prepare your code for KNL is to optimize for KNC

[Learn More](#)

HOW Series



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

STARTS MAY 23

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

colfaxresearch.com/how-series

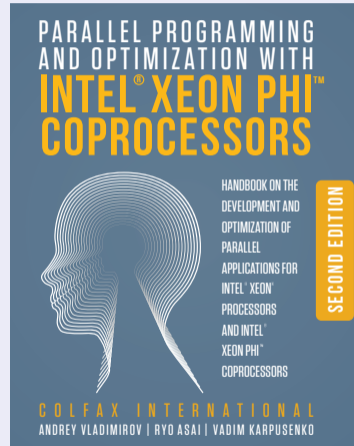
Textbook

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

Parallel Programming and Optimization with Intel® Xeon Phi™ Coproprocessors

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

© Colfax International, 2015



<http://xeonphi.com/book>

§7. Closing Words

COLFAX RESEARCH
Log In/Out or Register

CONTRIBUTING TO INNOVATIONS IN COMPUTING

/ READ WATCH LEARN CONNECT JOIN



To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation



Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why xx Acceleration May be Enough)


Featured Video

Additional Reading

Research material on vectorization in a streaming code



<http://colfaxresearch.com/?p=708>

Events
Documentation
Carriers

Consulting

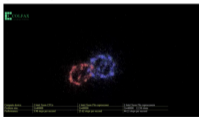


Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and clouds
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor technology
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a Hands-On Workshop: a virtual workshop to explore your computing problems, software experience in architecture, software

All Video Courses - COP 001 - Chapter 2 - Episode 2.1

Episode 2.1 — Purpose of the MIC architecture



In this video episode 2.1 we will introduce Intel Xeon Phi coprocessors based on the Intel Many Integrated Core, or MIC, architecture and will take a look at the architecture of the Intel Xeon Phi coprocessors.

Download Download the Intel Xeon Phi coprocessors

Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives



In this paper we will discuss the new HGST Ultrastar Archive HaaS SMR drives, a different feature set which offers storage capacities of 10 TB and beyond. These drives have high density and large capacities. These drives are well suited for large "data center" applications, such as data archive applications. The drives benefit from the latest features.

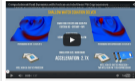
For more information on the new HGST Ultrastar Archive HaaS SMR drives, visit the website www.hgst.com and download the HGST white paper. You can also visit the HGST website for more information on the new HGST Ultrastar Archive HaaS SMR drives. The rest of this paper is for your review.

Download the Intel Xeon Phi coprocessors

Download Download the Intel Xeon Phi coprocessors

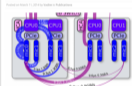
Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

In this presentation we will discuss the Intel Xeon Phi coprocessors and the application of the Fortran language on the Intel Xeon Phi coprocessors. The Intel Xeon Phi coprocessors are well suited for the performance needs of the code in the Intel Xeon Phi coprocessors.



Download the Intel Xeon Phi coprocessors

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors



In this paper we will discuss the configuration and benchmarks of peer-to-peer communication over Gigabit Ethernet and InfiniBand in a cluster with Intel Xeon Phi coprocessors.

Parallel Computing in the Search for New Physics at LHC



Download the Intel Xeon Phi coprocessors

Interview with James Reinders: future of Intel MIC architecture, parallel programming, education

In this interview we will discuss the future of Intel MIC architecture, parallel programming, and education. James Reinders is a senior advisor at Intel and a leading expert in the field.



1. Intel Xeon Phi coprocessors: what are they?
2. Intel Xeon Phi coprocessors: how do they work?
3. Intel Xeon Phi coprocessors: what are the benefits?
4. Intel Xeon Phi coprocessors: what are the challenges?
5. Intel Xeon Phi coprocessors: what are the future prospects?
6. Intel Xeon Phi coprocessors: what are the key takeaways?
7. Intel Xeon Phi coprocessors: what are the next steps?
8. Intel Xeon Phi coprocessors: what are the conclusions?

<http://colfaxresearch.com/>
 (already registered? get \$10 off the book)

Developer Access Program (DAP)

Can't wait to get your hands on Knights Landing?



Find out more at dap.xeonphi.com
or contact us at dap@colfax-intl.com.

Bottom Line

KNL is a **highly-parallel** successor of KNC, with improvements in both **performance** and **ease-of-use**.



KEEP CALM
AND
GO PARALLEL

Thank you for Tuning In!

