Developer's Guide to Knights Landing

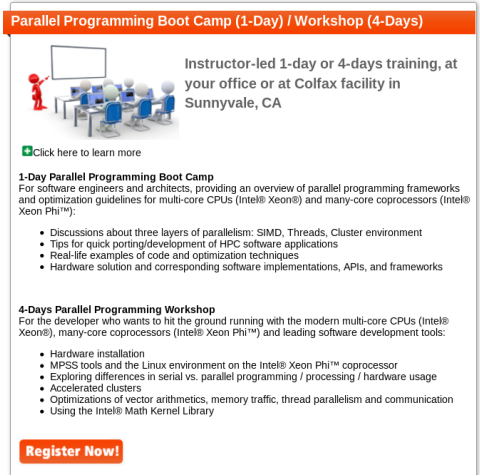
Introduction to 2nd Generation Intel® Xeon Phi™ Processors

Colfax International — @colfaxintl

May 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.



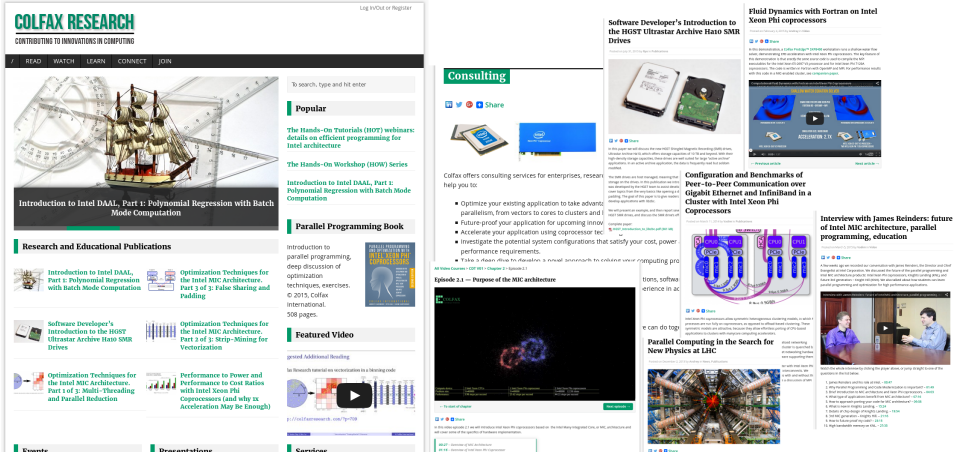
colfaxresearch.com/knl-webinar/

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



http://colfaxresearch.com/

(already registered? get $10 off the book)

HOW Series

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.

Get Ready For KNL

http://colfaxresearch.com/get-ready-for-intel-knights-landing-3-papers/

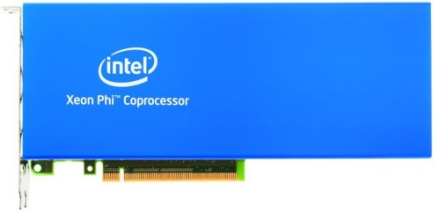
§2. Intel Architecture: Today and Tomorrow

Intel Architecture

Intel Xeon Processor

Current: Broadwell
Upcoming: Skylake

Intel Xeon Phi Coprocessor, 1st generation



Current: Knights Corner (KNC)

Intel Xeon Phi Processor, 2nd generation*

* socket and coprocessor versions

Upcoming: Knights Landing (KNL)

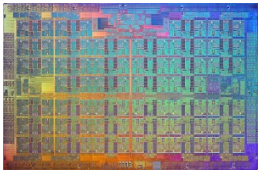
Multi-Core Architecture

Intel Many Integrated Core (MIC) Architecture

Intel Xeon Phi Processors (2nd Gen)

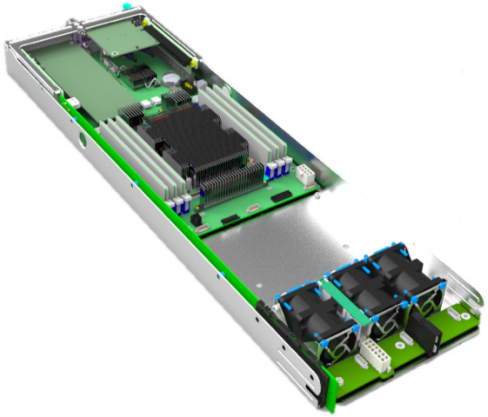
2nd Generation of Intel Many Integrated Core (MIC) Architecture.
Specialized platform for demanding computing applications.

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



Standard CPU Form Factor

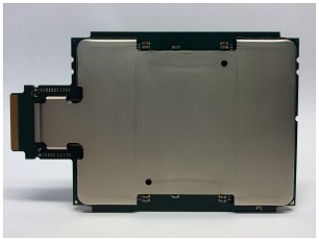
- Bootable Host Processor
  - No need for “host”. OS runs on KNL processor.
  - Supports common OS.
  - No more PCIe bottleneck!
- Direct access to ≤ 384 GiB DDR4 RAM
  - Up to ≈90 GB/s DDR4 bandwidth
- Access to PCIe Bus

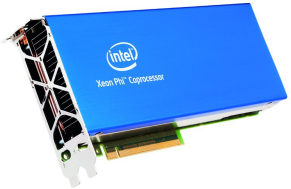


Future Releases

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.

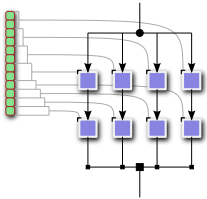
Single-threaded code on KNL
is like having 1 fireman service 72 steam engines

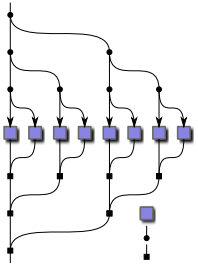
Implementing Multi-threading

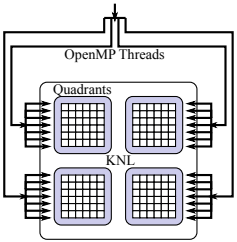
You have to use a core to benefit from it!

Threading frameworks:

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



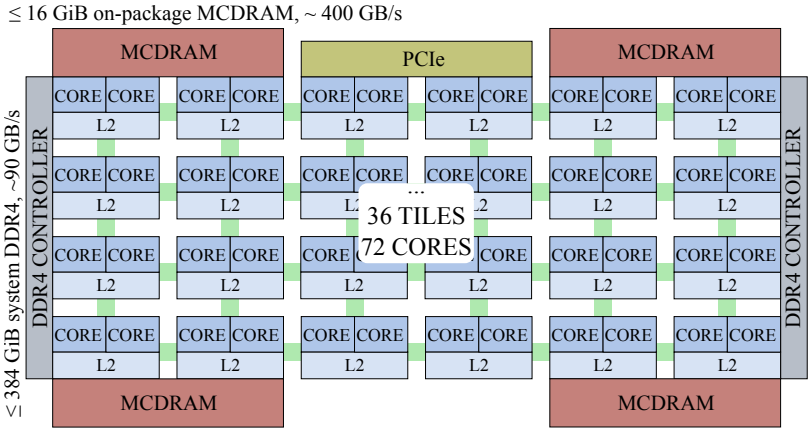




Features of KNL Cores

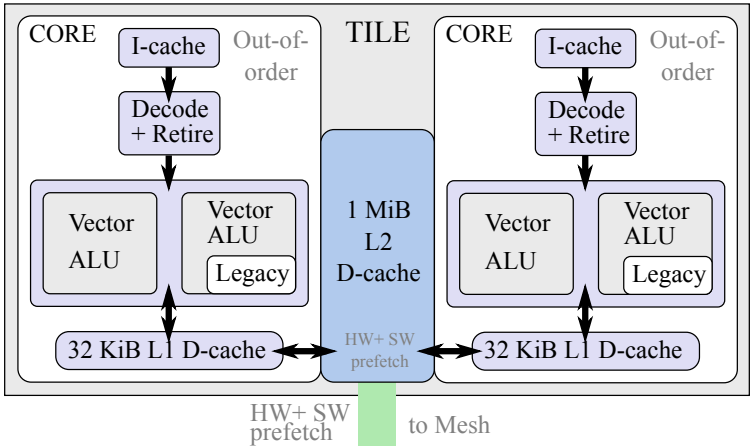
KNL Die Organization: Tiles

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



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 cpuinfo

Neighboring threads often work on nearby/same locations in memory.
→ Use thread pinning to have them share L2 cache!

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

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

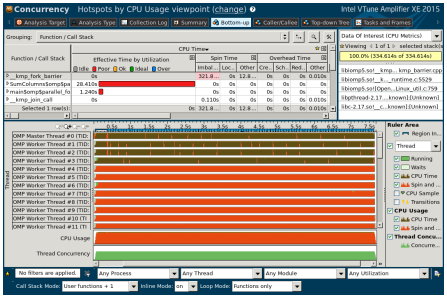
Use `KMP_AFFINITY` for Intel Compilers or `OMP_PROC_BIND` for GCC compilers.

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

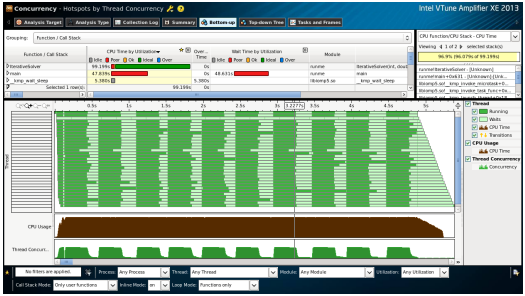
Tuning Threading

Even if you do have multi-threaded code, look out for issues such as...

- Insufficient parallelism



- Load imbalance



Learn more at: colfaxresearch.com/how-series

§4. Vectorization

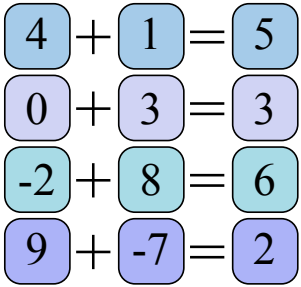
Scalar Code and Our Fireman

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

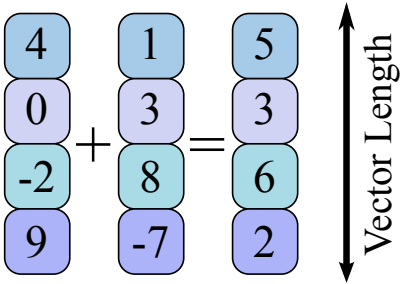
Short Vector Support

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

Scalar Instructions



Vector Instructions



Vector Instructions on KNL

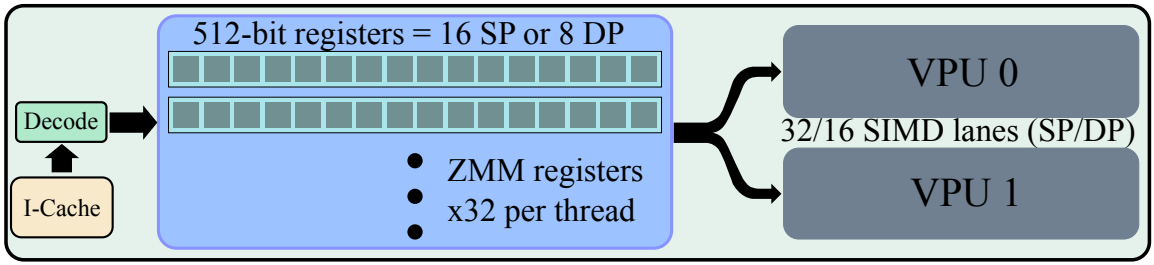
Dual VPU

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

Penalty from non-vectorized code

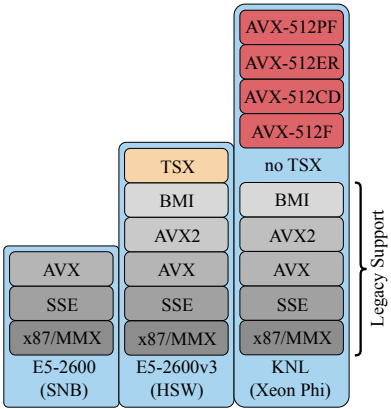
SP → 512 bit registers / 32 bits × 2 VPUs = **32** SIMD lanes

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.
- ≤ 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

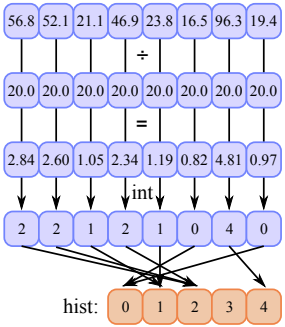
Check `/proc/cpuinfo` for AVX-512 flags

```
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
aperfmperf 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 ≥ 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
```

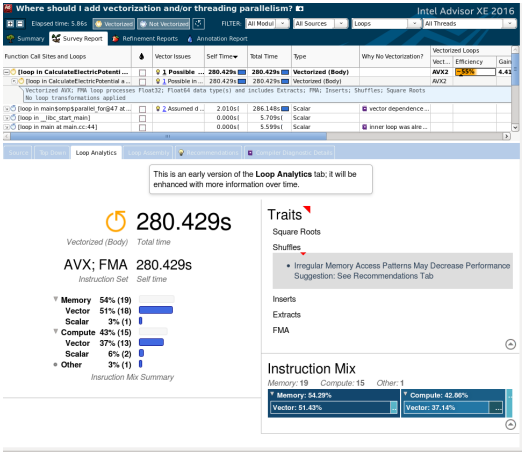
Basic automatic vectorization support: add -O2 or -O3.

```
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.



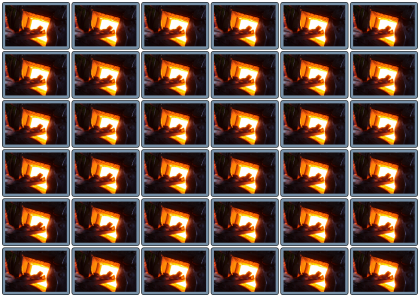
- Providing enough parallelism.
  - More consecutive vector operations required to overcome vectorization latency.
- Loop pipelining and unrolling.
  - Double the pipeline stages to populate.
- Better vectorization patterns.
  - Avoid long latency operations with unit-stride and unmasked operations.

§5. Memory Architecture

Where's the Fuel?

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

Source: wikipedia

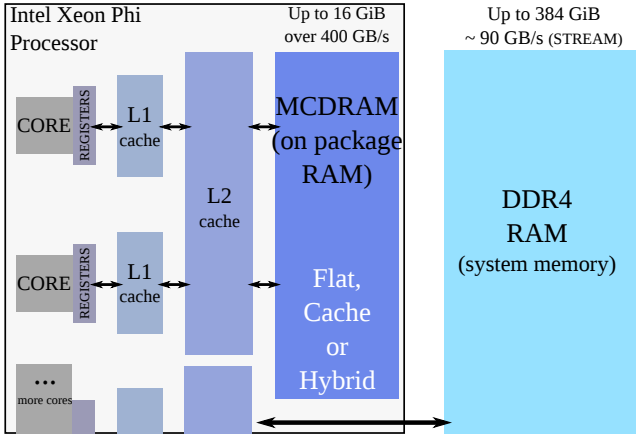


KNL memory must be used efficiently to feed the cores.

MCDRAM on KNL

KNL Memory Organization

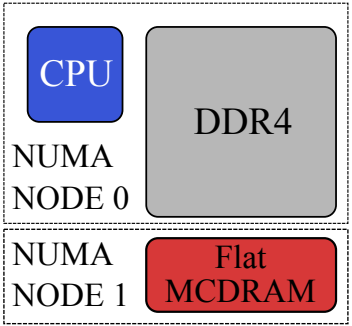
- Direct access to on package MCDRAM *and* system DDR4 (socket)
- Use MCDRAM as cache, in flat mode, or as hybrid



MCDRAM Memory Modes

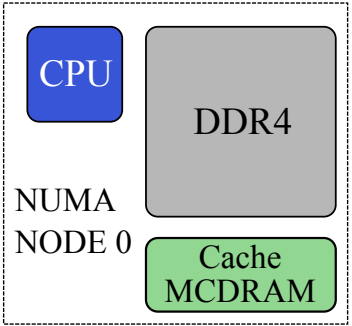
Flat Mode

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



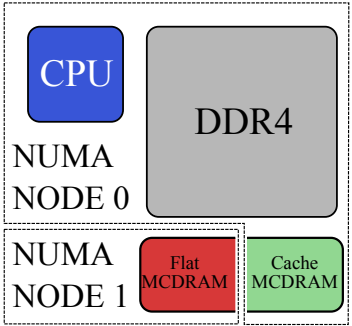
Cache Mode

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

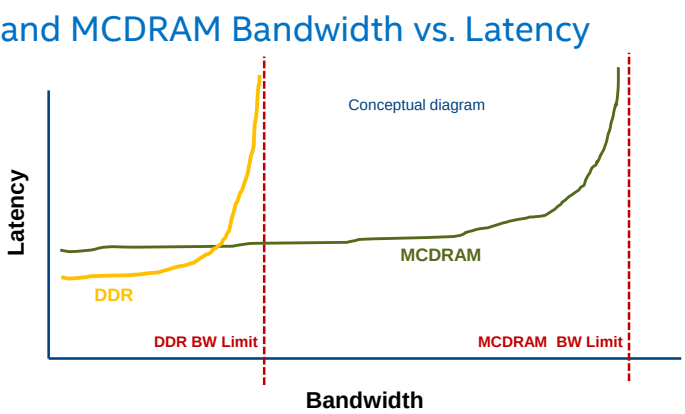


Hybrid Mode

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



MCDRAM Performance



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

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.

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

Fortran Allocations.

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

Compilation with Memkind Library and hbwmalloc

To compile C/C++ applications:

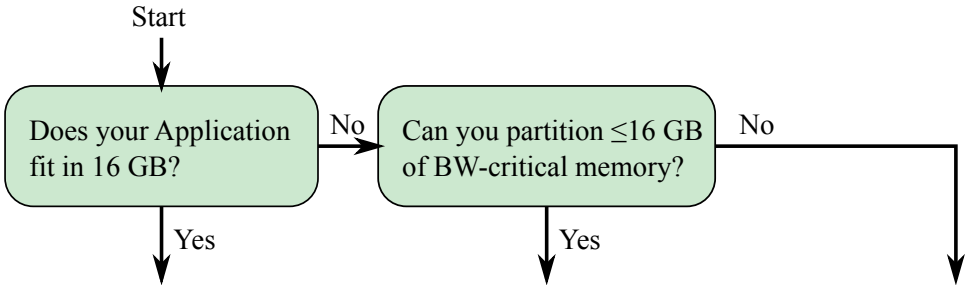
```
user@knl%  icpc -lmemkind foo.cc -o runme
user@knl%  g++ -lmemkind foo.cc -o runme
```

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

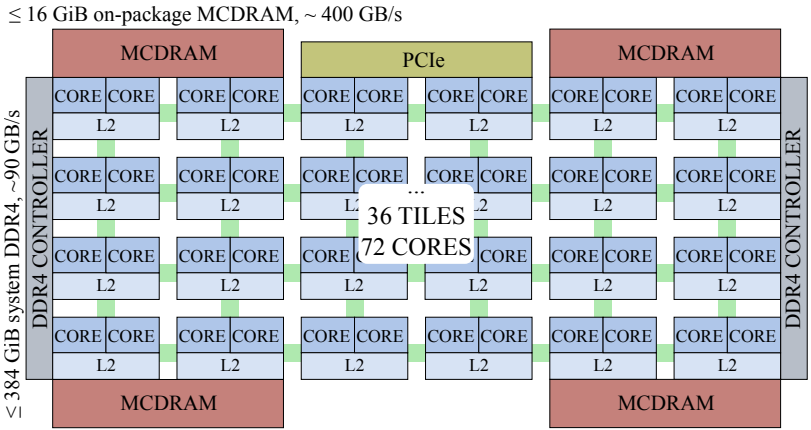


numactl	**Memkind**	**Cache mode**
• Simply run the whole program in MCDRAM • No code modification required	• Manually allocate BW-critical memory to MCDRAM • Memkind calls need to be added.	• Allow the OS to figure out how to use MCDRAM • No code modification required

Cluster Modes on KNL

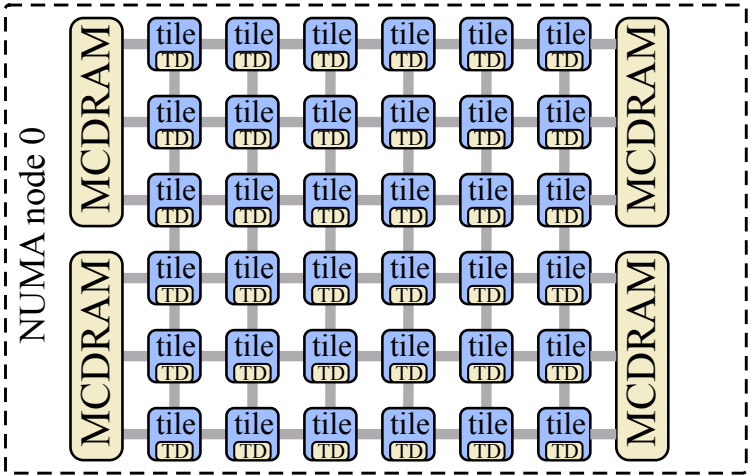
KNL Die Organization

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



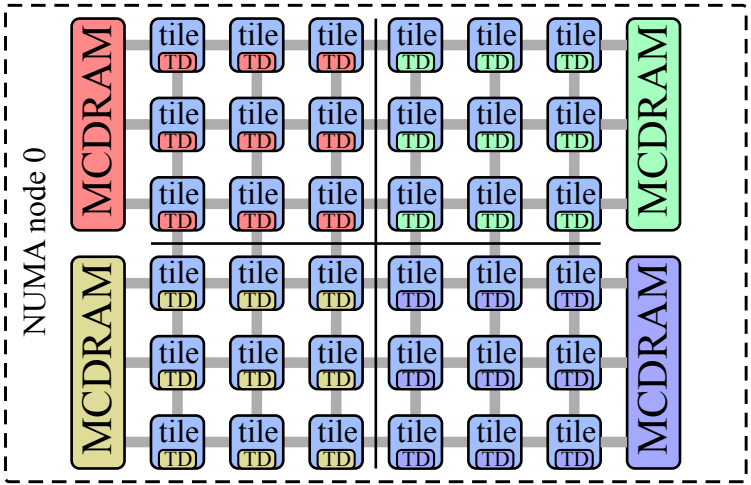
Cluster Modes: All-to-All

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



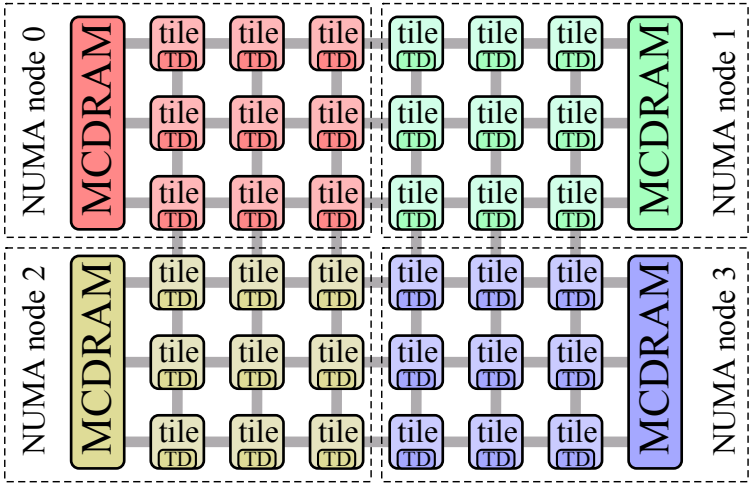
Cluster Modes: Quadrant/Hemisphere

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).



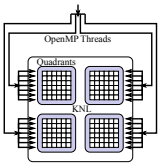
Programming with Cluster Modes

How to Use Cluster modes

Thread affinity to match the memory/directory affinity.

Nested OpenMP

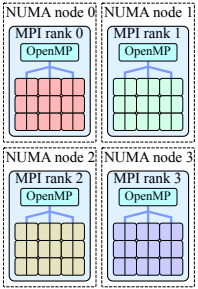
```
1 #pragma omp parallel
2 {
3   // ...
4 #pragma omp parallel
5   {
6     // ...
7   }
8 }
```




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

MPI+OpenMP

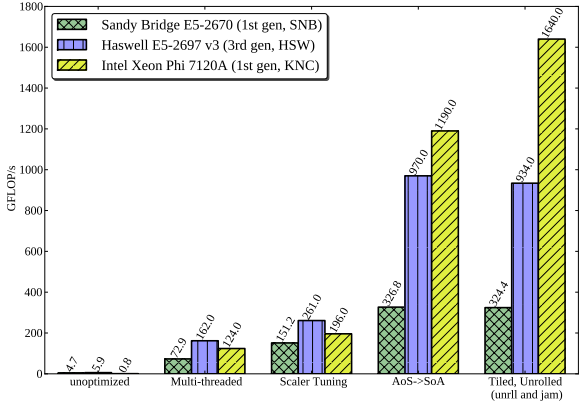
```
1 stat = MPI_Init();
2 // ...
3 #pragma omp parallel
4 {
5     // ...
6 }
7 // ...
8 MPI_Finalize();
```




```
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

Interested? Sign-up at:

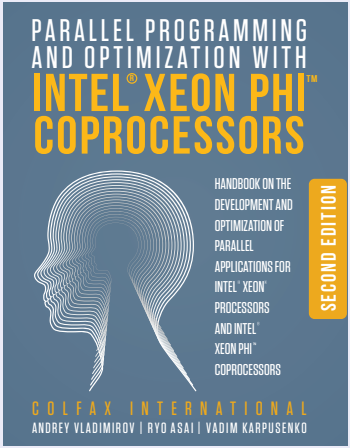
colfaxresearch.com/how-series

Textbook

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

Parallel Programming
and Optimization with
Intel® Xeon Phi™
Coprocessors

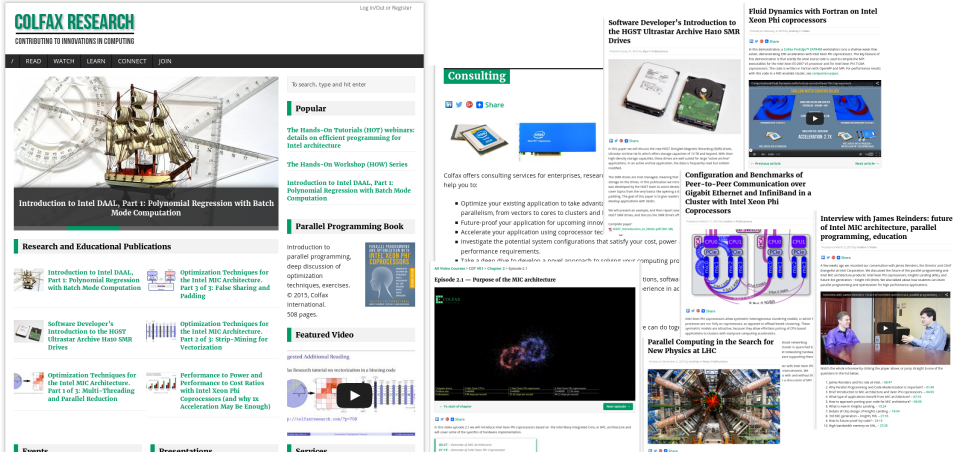
Handbook on the Development and
Optimization of Parallel Applications
for Intel® Xeon® Processors
and Intel® Xeon Phi™ Coprocessors



http://xeonphi.com/book

§7. Closing Words

Colfax Research



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**.

Thank you for Tuning In!