THE INTEL® XEON PHI™ ARCHITECTURE

(CODENAME "KNIGHTS LANDING")

Dr. Christopher Dahnken

Senior Application Engineer
Software and Services Group

Legal Disclaimer & Optimization Notice

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

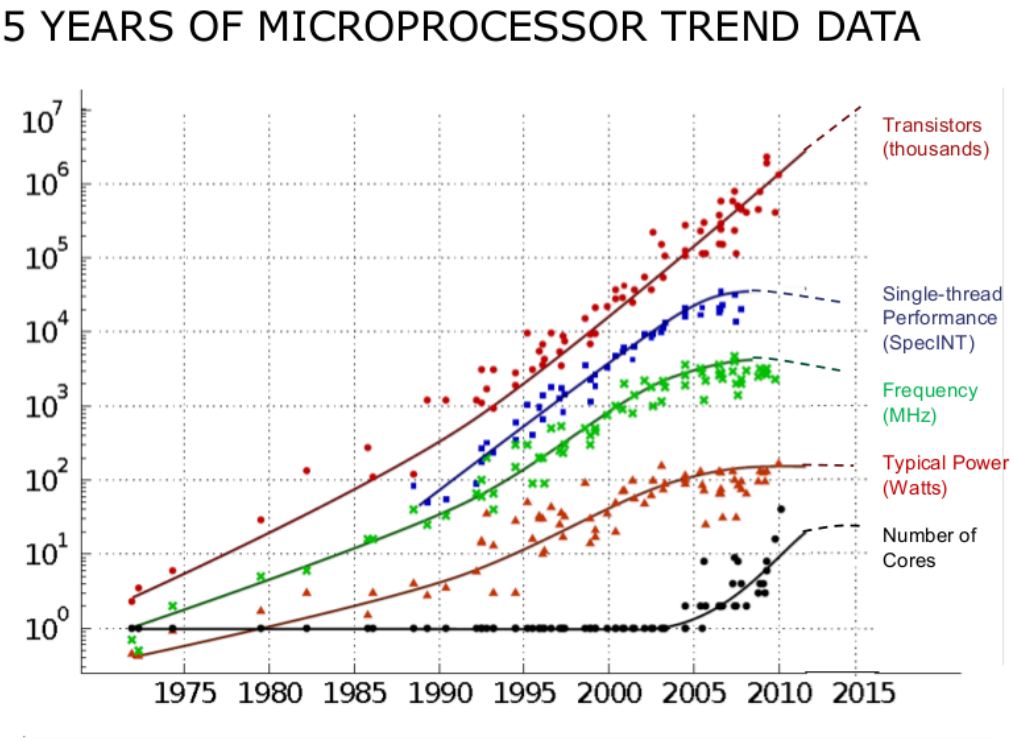
Notice revision #20110804

Agenda

1. Introduction
2. High-Level Architecture & Instruction Set
3. KNL Microarchitecture
4. Getting performance on Knights Landing

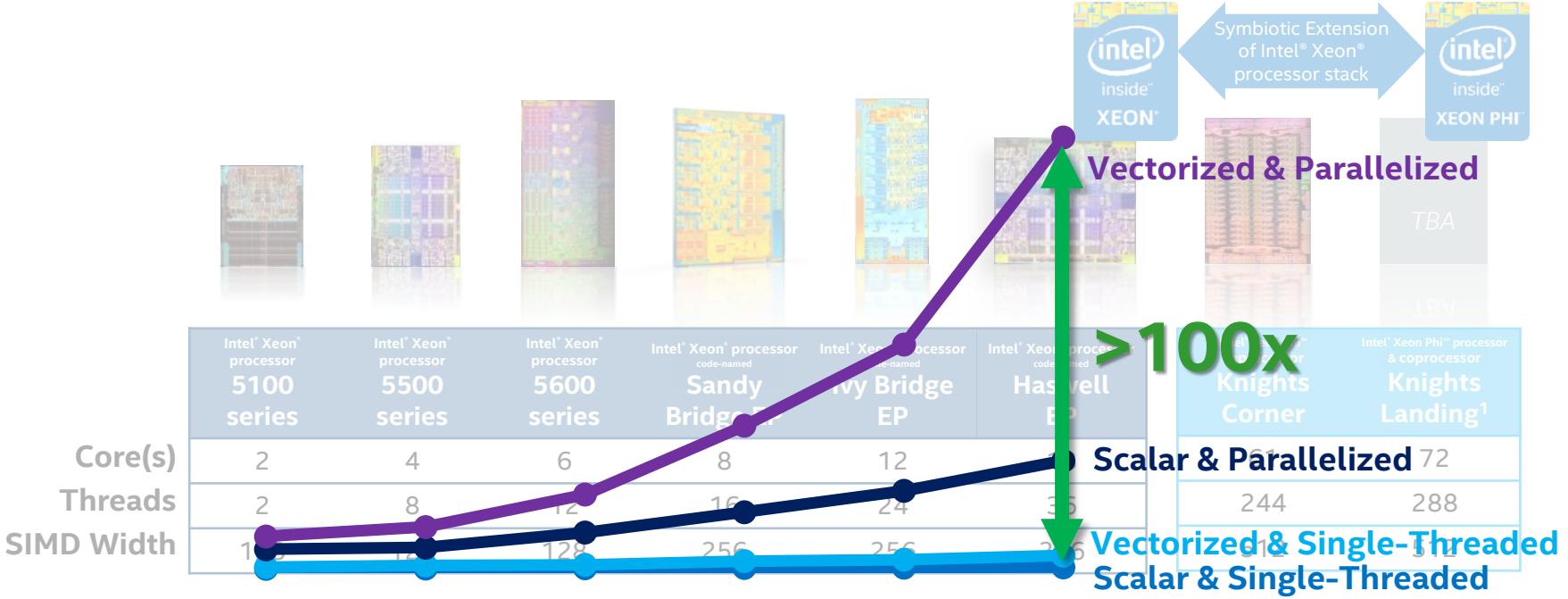
INTRODUCTION

Moore's Law and Parallelism



Original data collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond and C. Batten
Dotted line extrapolations by C. Moore

CPU Parallelism is Already a MUST



Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. Configurations: Intel Performance Projections as of Q1 2015. For more information go to http://www.intel.com/performance . Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance. Copyright © 2015, Intel Corporation

Chart illustrates relative performance of the Binomial Options DP workload running on an Intel® Xeon® processor from the adjacent generation.

*Product specification for launched and shipped products available on ark.intel.com.

[1]Not launched

Parallelism and Performance

Peak GFLOP/s in Single Precision

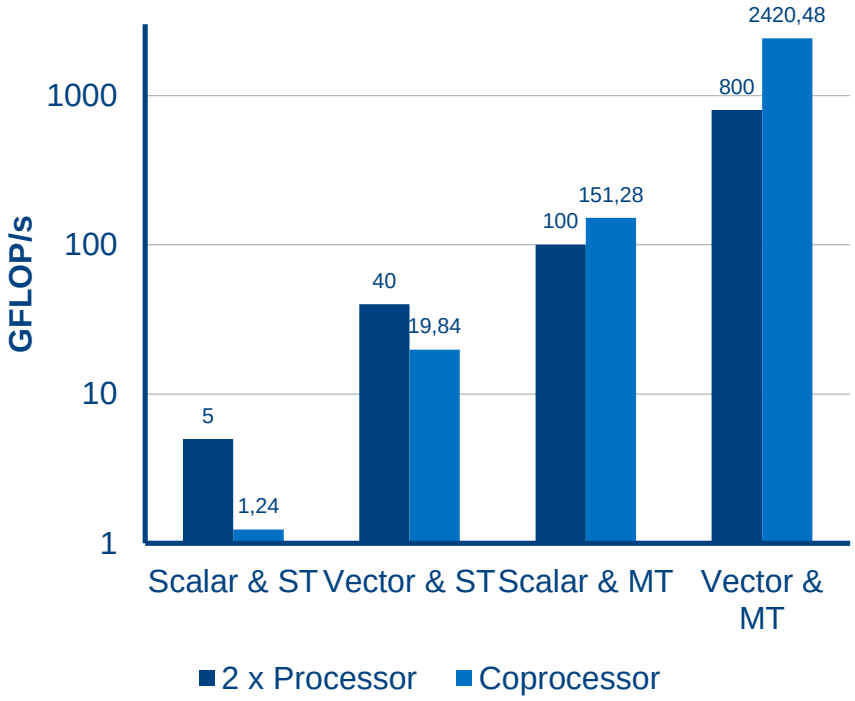
- Clock Rate x Cores x Ops/Cycle x SIMD

2 x Intel® Xeon® Processor E5-2670v2

- 2.5 GHz x 2 x 10 cores x 2 ops x 8 SIMD = 800 GFLOP/s

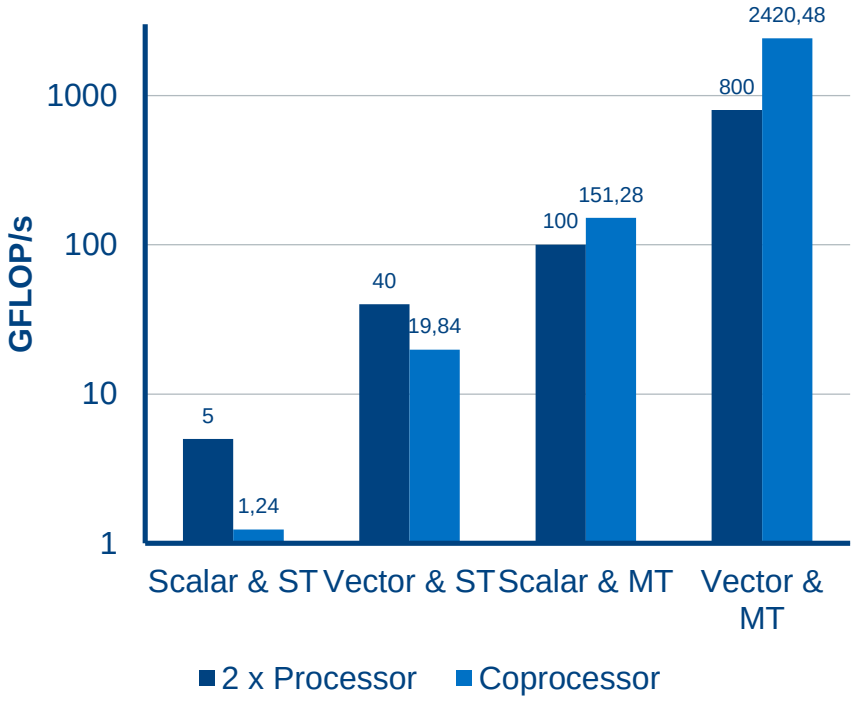
Intel® Xeon Phi™ Coprocessor 7120P

- 1.24 GHz x 61 cores x 2 ops x 16 SIMD = 2420.48 GFLOP/s



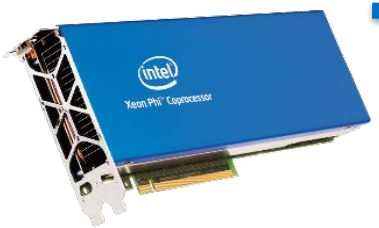
Parallelism and Performance

- On modern hardware, Performance = Parallelism
- Flat programming model on parallel hardware is not effective.
- Parallel programming is not optional.
- Codes need to be made parallel ("modernized") before they can be tuned for the hardware ("optimized").



HIGH-LEVEL ARCHITECTURE & INSTRUCTION SET

A Paradigm Shift



Coprocessor

Fabric

Memory

Server Processor

Memory Bandwidth
400+ GB/s STREAM

Memory Capacity
Over 25x KNC

Resiliency
Systems scalable to >100 PF

Power Efficiency
Over 25% better than card

I/O
200 Gb/s/dir with int fabric

Cost
Less costly than discrete parts

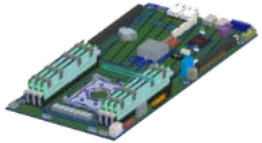
Flexibility
Limitless configurations

Density
3+ KNL with fabric in 1U

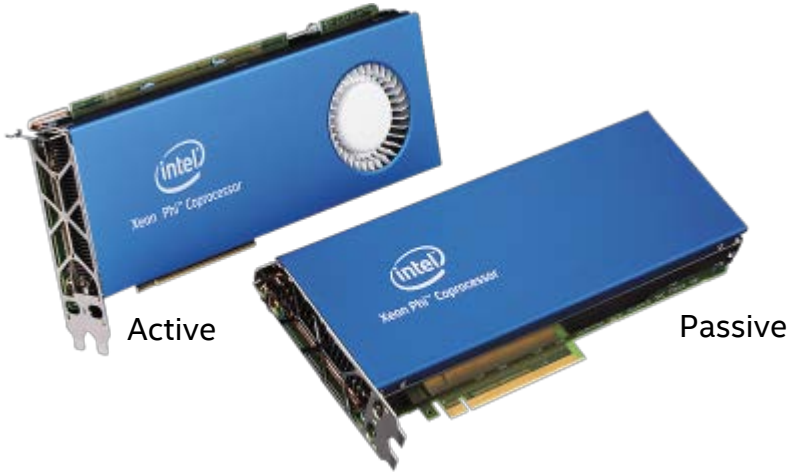
Knights Landing (Host or PCIe)

Host Processor

Host Processor
w/ integrated Fabric



Groveport Platform



Active

Passive

Knights Landing Processors

Host Processor for Groveport Platform

Solution for future clusters with both Xeon and Xeon Phi

Knights Landing PCIe Coprocessors

Ingredient of Grantley & Purley Platforms

Solution for general purpose servers and workstations

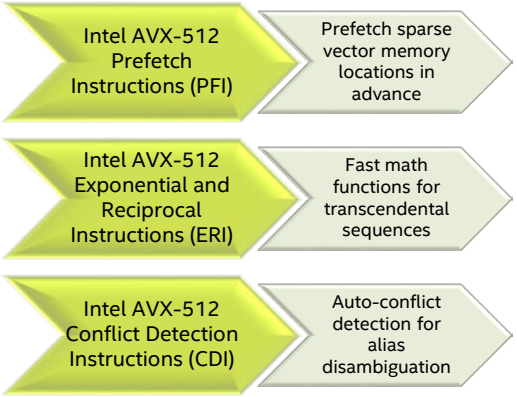
PCIe Coprocessor vs. Host Processor

Baseline	**Perf/W/$**	Up to 40% better[1]
PCIe	**I/O**	Fabric
Up to 16GB	**Mem Capacity**	Up to 384GB
Unique	**Manageability**	Standard CPU
Up to 4 in 1U	**Density**	>4 in 1U
<100 PF	**Scalability**	>100 PF
Partial	**Utilization**	Full

[1]Results based on internal Intel analysis using estimated power consumption and projected component pricing in the 2015 timeframe. This analysis is provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

KNL Instruction Set

KNL Instruction Set



CPUID	Instructions	Description
AVX512PF	PREFETCHWT1	Prefetch cache line into the L2 cache with intent to write
	VGATHERPF{D,Q}{0,1}PS	Prefetch vector of D/Qword indexes into the L1/L2 cache
	VSCATTERPF{D,Q}{0,1}PS	Prefetch vector of D/Qword indexes into the L1/L2 cache with intent to write
AVX512ER	VEXP2{PS,PD}	Computes approximation of 2^x with maximum relative error of 2^{-23}
	VRCP28{PS,PD}	Computes approximation of reciprocal with max relative error of 2^{-28} before rounding
	VRSQRT28{PS,PD}	Computes approximation of reciprocal square root with max relative error of 2^{-28} before rounding
AVX512CD	VPCONFLICT{D,Q}	Detect duplicate values within a mask and create conflict-free subsets
	VPLZCNT{D,Q}	Count the number of leading zero bits in each element
	VPBROADCASTM{B2Q, W2D}	Broadcast vector mask into vector elements

Intel® Software Development Emulator

- Freely available instruction emulator
  - http://www.intel.com/software/sde
- Emulates existing ISA as well as ISAs for upcoming processors
- Intercepts instructions with Pin; allows functional emulation of existing and upcoming ISAs (including AVX-512).
  - Execution times may be slow, but the result will be correct.
- Record dynamic instruction mix; useful for tuning/assessing vectorization content
- First step: compile for Knights Landing:
  - $ icpc –xMIC-AVX512 <compiler args>

Running SDE

- SDE invocation is very simple:
  - $ sde <sde-opts> -- <binary> <command args>
- By default, SDE will execute the code with the CPUID of the host.
  - The code may run more slowly, but will be functionally equivalent to the target architecture.
  - For Knights Landing, you can specify the -knl option.
  - For Haswell, you can specify the -hsw option.

KNL MICROARCHITECTURE

KNL Architecture Overview

ISA

Intel® Xeon® Processor Binary-Compatible (w/Broadwell)

On-package memory

Up to 16GB, ~500 GB/s STREAM at launch

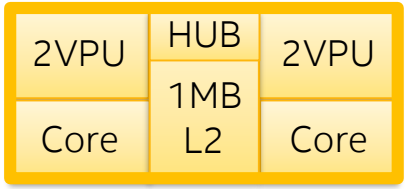
Platform Memory

Up to 384GB (6ch DDR4-2400 MHz)

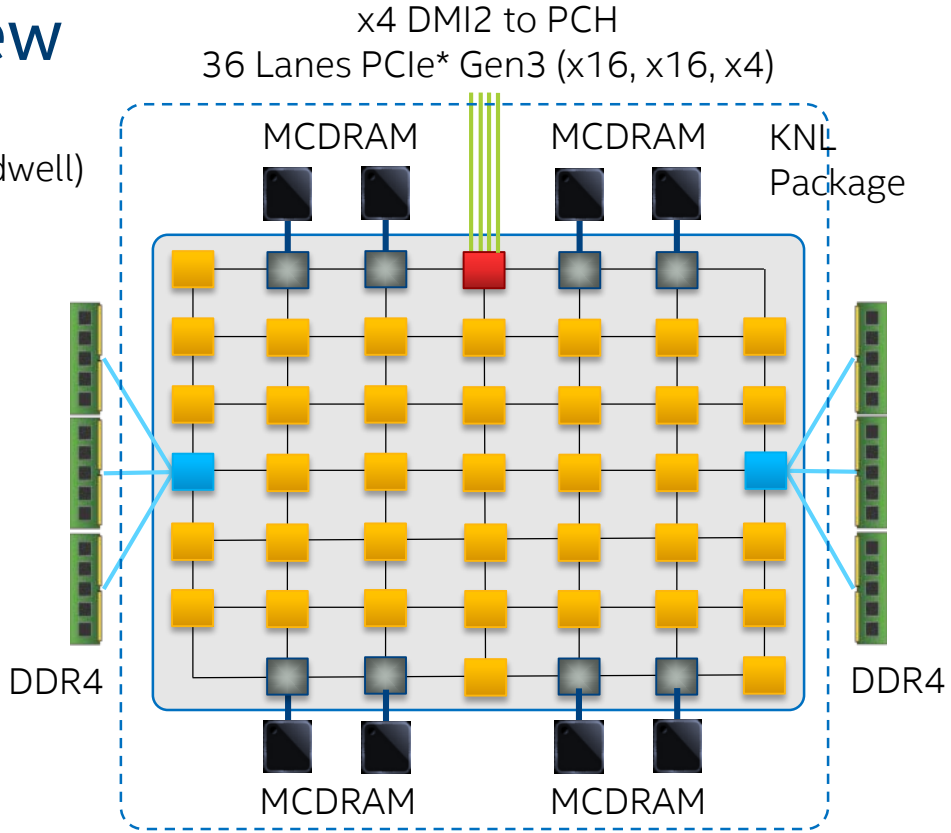
Fixed Bottlenecks

- ✓ 2D Mesh Architecture
- ✓ Out-of-Order Cores
- ✓ 3x single-thread vs. KNC

TILE: (up to 36)

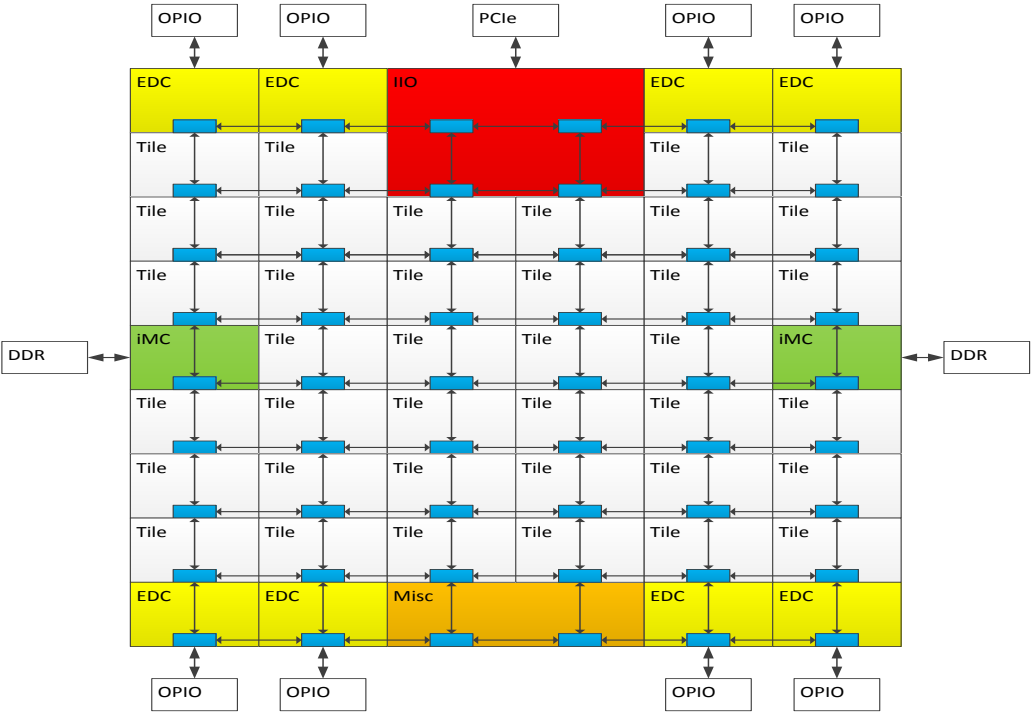


Enhanced Intel® Atom™ cores based on Silvermont™ Microarchitecture



Tile | EDC (embedded DRAM controller) | IMC (integrated memory controller) | IIO (integrated I/O controller)

KNL Mesh Interconnect



Mesh of Rings

- Every row and column is a (half) ring
- YX routing: Go in Y → Turn → Go in X
- Messages arbitrate at injection and on turn

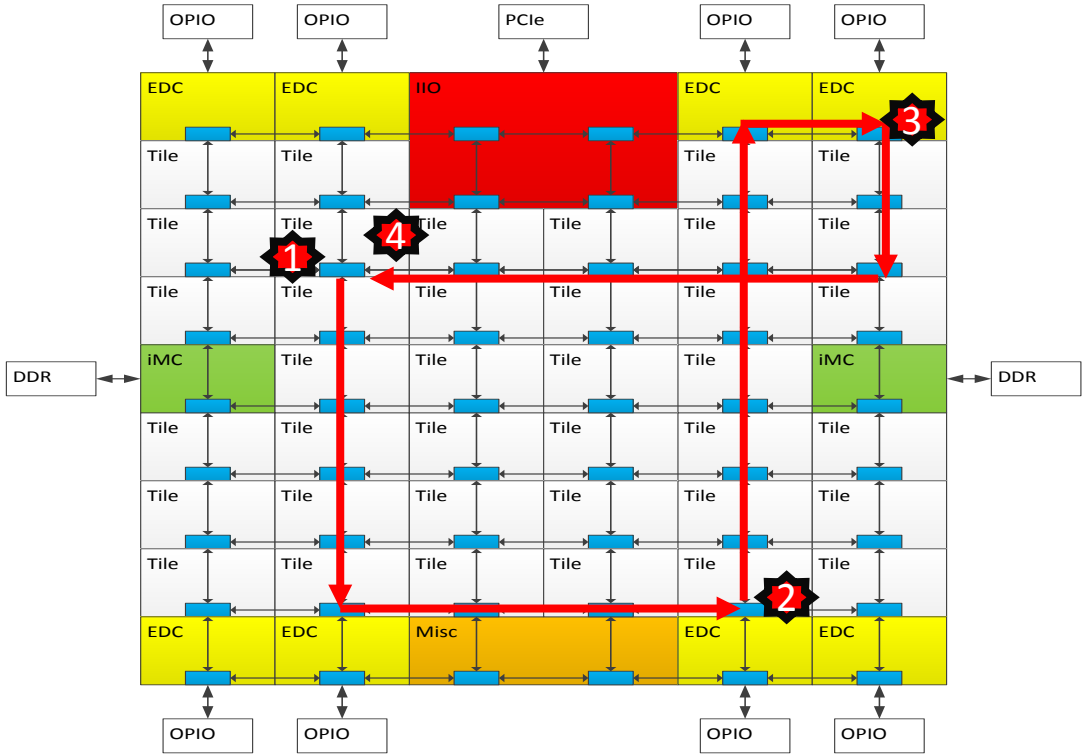
Cache Coherent Interconnect

- MESIF protocol (F = Forward)
- Distributed directory to filter snoops

Three Cluster Modes

(1) All-to-All (2) Quadrant (3) Sub-NUMA Clustering

Cluster Mode: All-to-All



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Address uniformly hashed across all distributed directories

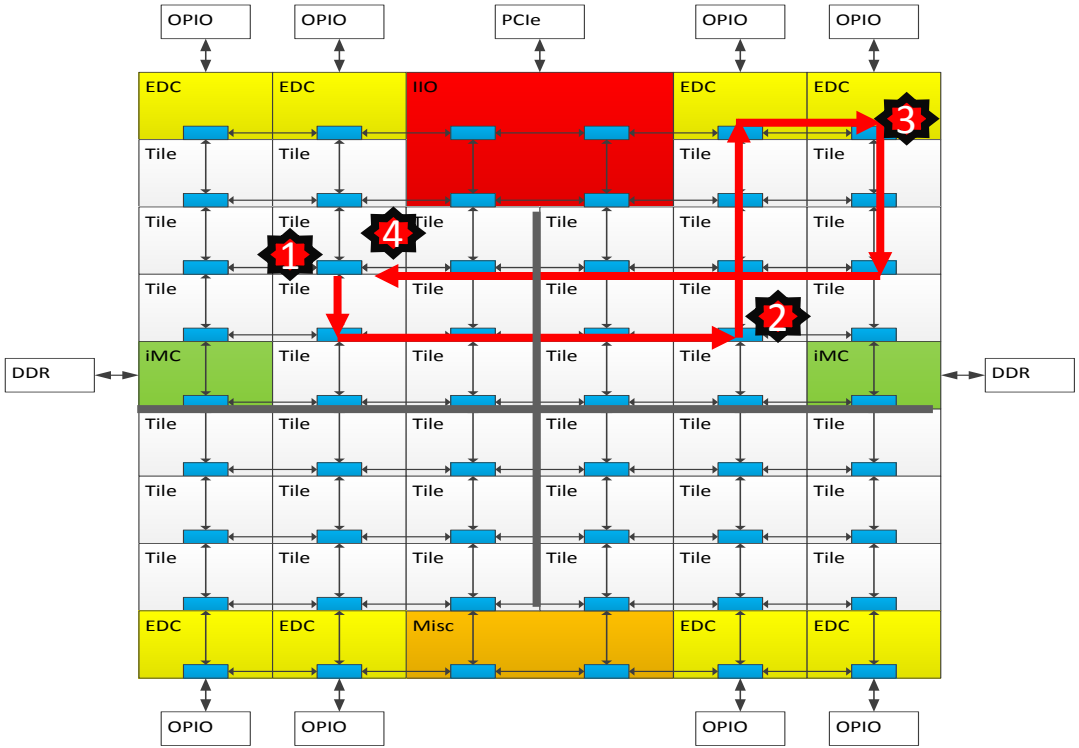
No affinity between Tile, Directory and Memory

Lower performance mode, compared to other modes. Mainly for fall-back

<u>Typical Read L2 miss</u>

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

Cluster Mode: Quadrant



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

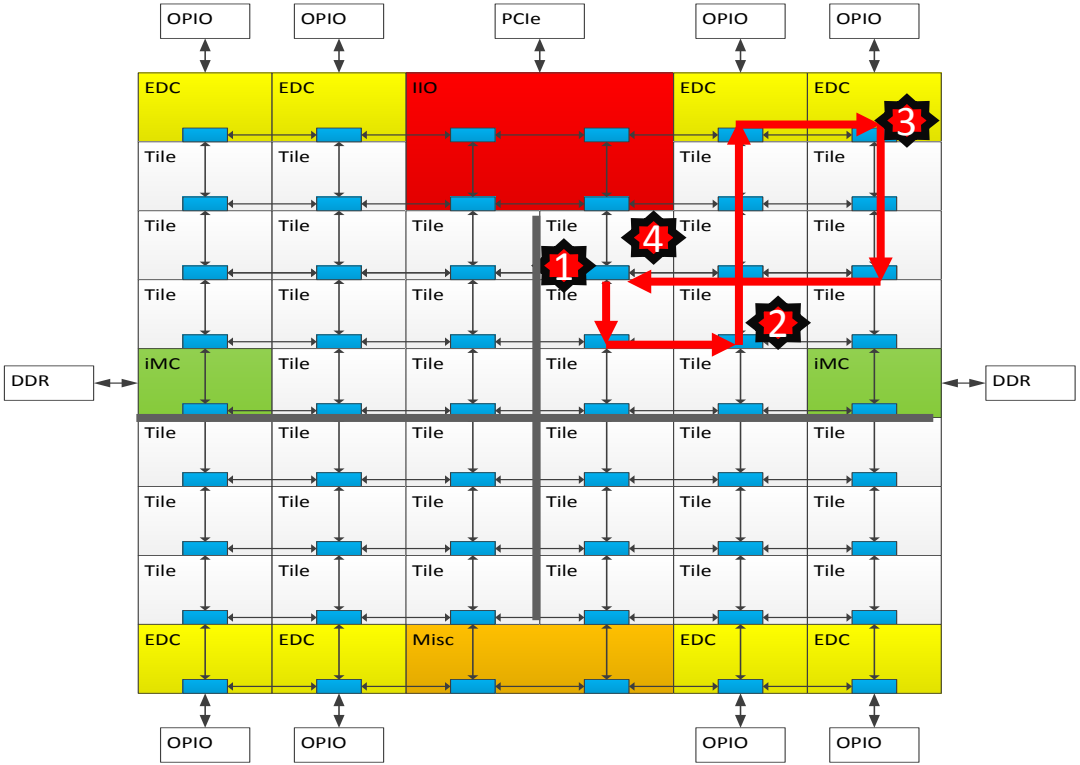
Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

Lower latency and higher BW than all-to-all. SW Transparent.

Cluster Mode: Sub-NUMA Clustering (SNC)



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

Looks analogous to 4-Socket Xeon

Affinity between Tile, Directory and Memory

Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

KNL Core and VPU

Out-of-order core w/ 4 SMT threads

VPU tightly integrated with core pipeline

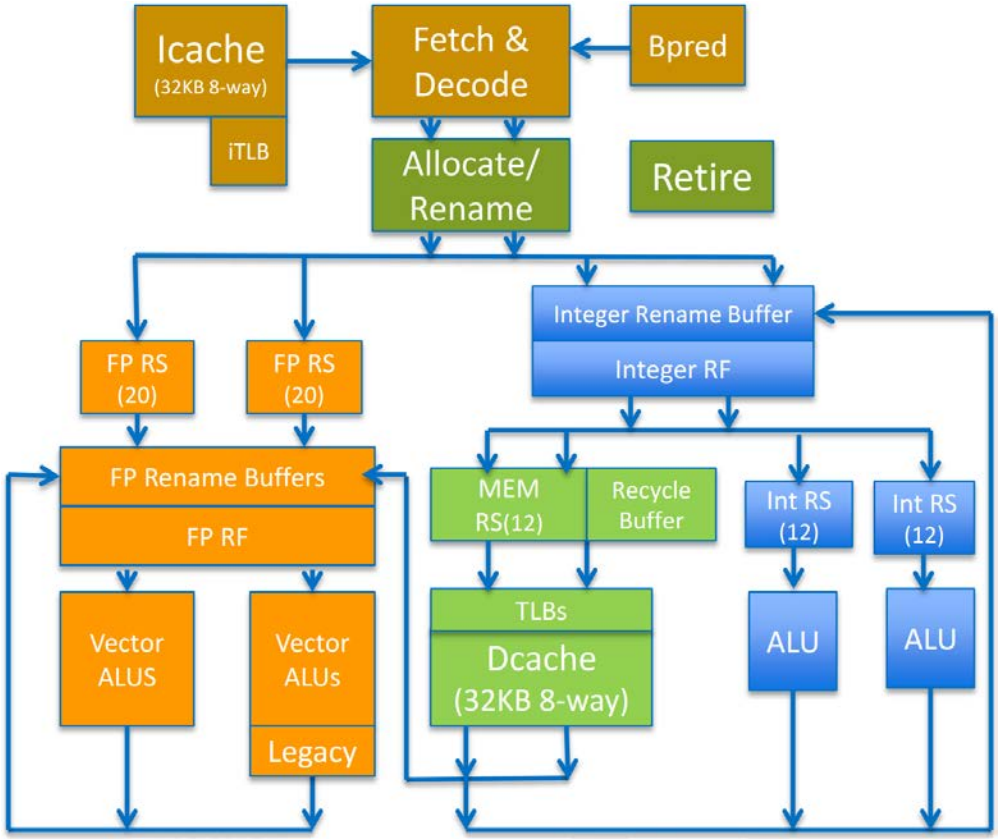
2-wide decode/rename/retire

2x 64B load & 1 64B store port for D$

L1 prefetcher and L2 prefetcher

Fast unaligned and cache-line split support

Fast gather/scatter support



KNL Hardware Threading

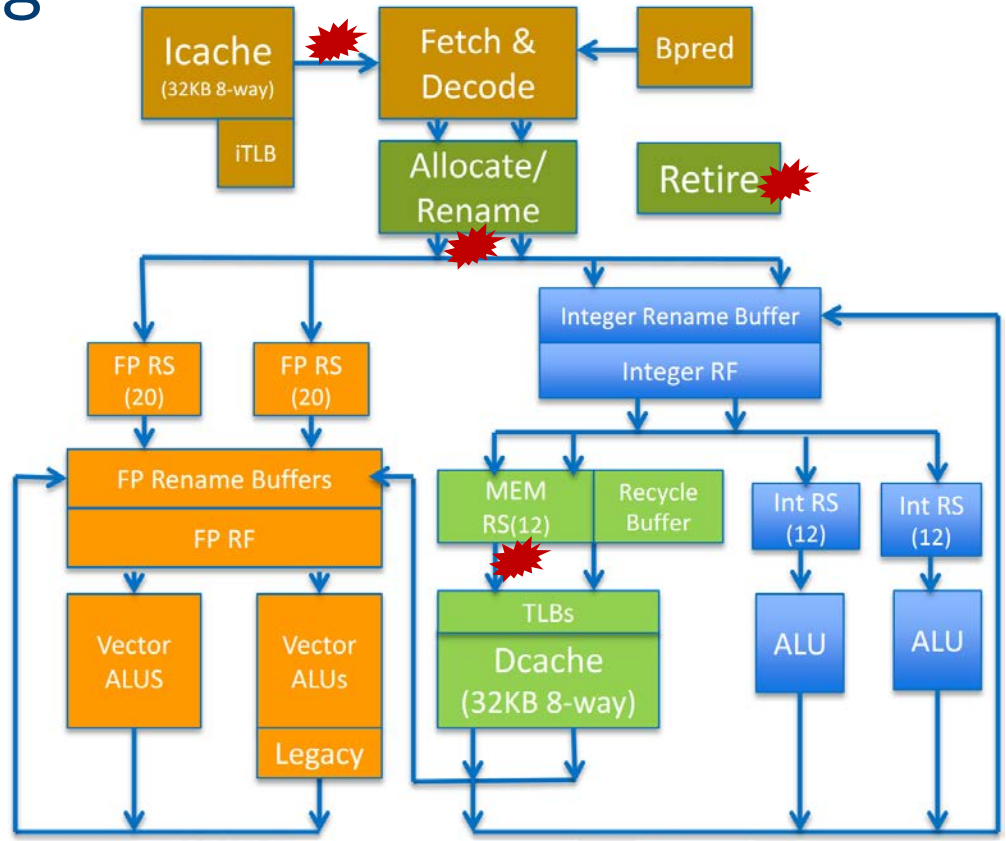
4 threads per core SMT

Resources dynamically partitioned

- Re-order Buffer
- Rename buffers
- Reservation station

Resources shared

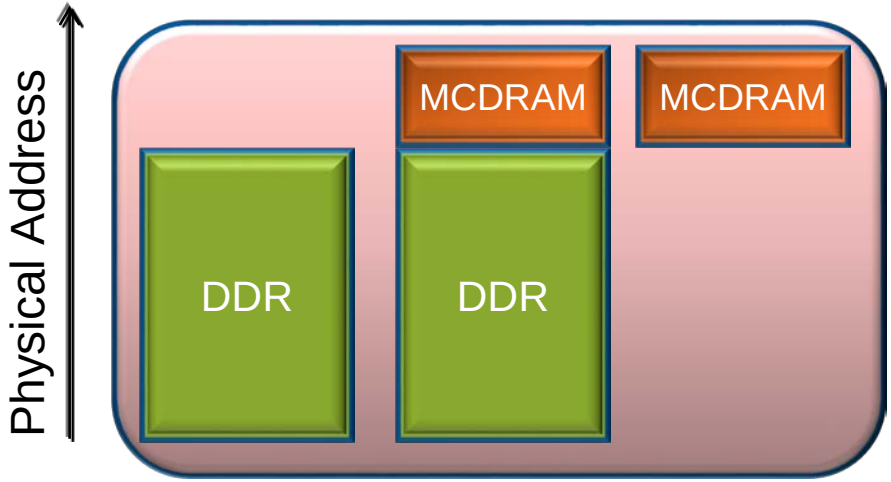
- Caches
- TLB



Thread selection point

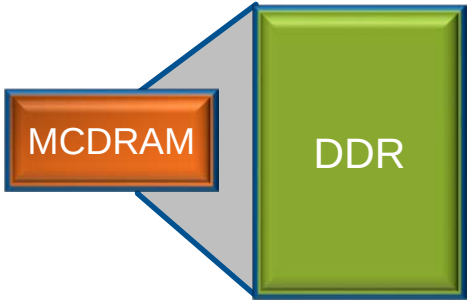
KNL Memory Modes

- Mode selected at boot
- MCDRAM-Cache covers all DDR

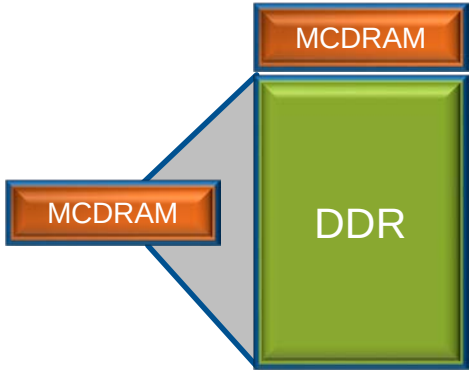


Flat Models

Cache Model



Hybrid Model



MCDRAM: Cache vs Flat Mode

Recommended

	DDR Only	MCDRAM as Cache	MCDRAM Only	Flat DDR + MCDRAM	Hybrid
Software Effort	No software changes required			Change allocations for bandwidth-critical data.	
Performance	Not peak performance.		Best performance.		

Limited memory capacity

Optimal HW utilization + opportunity for new algorithms

GETTING PERFORMANCE ON KNIGHTS LANDING

Efficiency on Knights Landing

- 1st Knights Landing systems appearing by end of year
- How do we prepare for this new processor without it at hand?
- Let's review the main performance-enabling features:
  - Up to 72 cores
  - 2x VPU / core, AVX-512
  - High-bandwidth MCDRAM
- Plenty of parallelism needed for best performance.

MPI needs help

- Many codes are already parallel (MPI)
  - May scale well, but…
  - What is single-node efficiency?
  - MPI isn't vectorising your code…
  - It has trouble scaling on large shared-memory chips.
    - Process overheads
    - Handling of IPC
    - Lack of aggregation off-die
- Threads are most effective for many cores on a chip
- Adopt a hybrid thread-MPI model for clusters of many-core

OpenMP 4.x

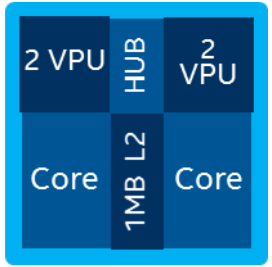
- OpenMP helps express thread- and vector-level parallelism via directives
  - (like #pragma omp parallel, #pragma omp simd)
- Portable, and powerful
- Don’t let simplicity fool you!
  - It doesn’t make parallel programming easy
  - There is no silver bullet
- Developer still must expose parallelism & test performance

Lessons from Previous Architectures

- Vectorization:
  - Avoid cache-line splits; align data structures to 64 bytes.
  - Avoid gathers/scatters; replace with shuffles/permutes for known sequences.
  - Avoid mixing SSE, AVX and AVX512 instructions.
- Threading:
  - Ensure that thread affinities are set.
  - Understand affinity and how it affects your application (i.e. which threads share data?).
  - Understand how threads share core resources.

Data Locality: Nested Parallelism

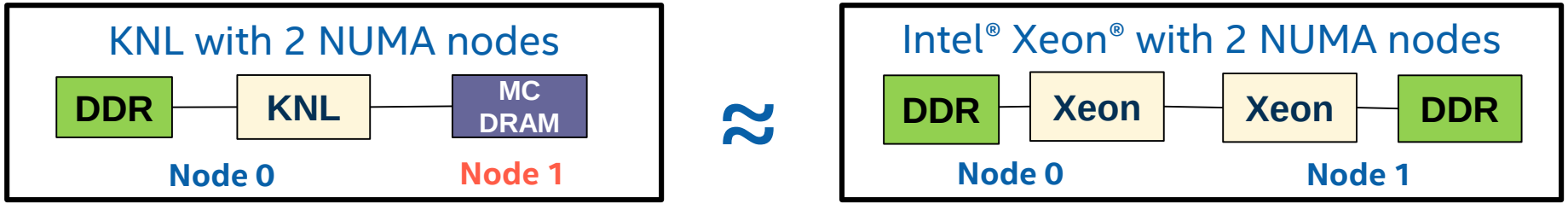
- Recall that KNL cores are grouped into tiles, with two cores sharing an L2.
- Effective capacity depends on locality:
  - 2 cores sharing no data => 2 x 512 KB
  - 2 cores sharing all data => 1 x 1 MB
- Ensuring good locality (e.g. through blocking or nested parallelism) is likely to improve performance.




```
#pragma omp parallel for num_threads(ntiles)
for (int i = 0; i < N; ++i)
{
      #pragma omp parallel for num_threads(8)
      for (int j = 0; j < M; ++j)
      {
            …
      }
}
```

Flat MCDRAM: SW Architecture

MCDRAM exposed as a separate NUMA node



- Memory allocated in DDR by default
  - Keeps low bandwidth data out of MCDRAM.
- Apps explicitly allocate important data in MCDRAM
  - "Fast Malloc" functions: Built using NUMA allocations functions
  - "Fast Memory" Compiler Annotation: For use in Fortran.

Flat MCDRAM using existing NUMA support in Legacy OS

Memory Allocation Code Snippets

Allocate 1000 floats from DDR

```
float   *fv;

fv = (float *)malloc(sizeof(float) * 1000);
```

Allocate 1000 floats from MCDRAM

```
float   *fv;

fv = (float *)hbw_malloc(sizeof(float) * 1000);
```

Allocate arrays from MCDRAM & DDR in Intel FORTRAN

```
c     Declare arrays to be dynamic
      REAL, ALLOCATABLE :: A(:), B(:), C(:)

!DIR$ ATTRIBUTES FASTMEM :: A

      NSIZE=1024
c
c     allocate array 'A' from MCDRAM
c
      ALLOCATE (A(1:NSIZE))
c
c     Allocate arrays that will come from DDR
c
      ALLOCATE  (B(NSIZE), C(NSIZE))
```

hbwmalloc – “Hello World!” Example

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <hbwmalloc.h>

int main(int argc, char **argv)
{
        const size_t size = 512;
        char *default_str = NULL;
        char *hbw_str = NULL;

        default_str = (char *)malloc(size);
        if (default_str == NULL) {
                perror("malloc()");
                fprintf(stderr, "Unable to allocate default string\n");
                return errno ? -errno : 1;
        }

        hbw_str = (char *)hbw_malloc(size);
        if (hbw_str == NULL) {
                perror("hbw_malloc()");
                fprintf(stderr, "Unable to allocate hbw string\n");
                return errno ? -errno : 1;
        }

        sprintf(default_str, "Hello world from standard memory\n");
        sprintf(hbw_str, "Hello world from high bandwidth memory\n");
        fprintf(stdout, "%s", default_str);
        fprintf(stdout, "%s", hbw_str);

        hbw_free(hbw_str);
        free(default_str);

        return 0;
}
```

Fallback policy is controlled with **`hbw_set_policy`**:

- HBW_POLICY_BIND
- HBW_POLICY_PREFERRED
- HBW_POLICY_INTERLEAVE

Page sizes can be passed to **`hbw_posix_memalign_psize`**:

- HBW_PAGESIZE_4KB
- HBW_PAGESIZE_2MB
- HBW_PAGESIZE_1GB

Memory Modes

MCDRAM as Cache

- Upside:
  - No software modifications required.
  - Bandwidth benefit.
- Downside:
  - Latency hit to DDR.
  - Limited sustained bandwidth.
  - All memory is transferred DDR -> MCDRAM -> L2.
  - Less addressable memory.

Flat Mode

- Upside:
  - Maximum bandwidth and latency performance.
  - Maximum addressable memory.
  - Isolate MCDRAM for HPC application use only.
- Downside:
  - Software modifications required to use DDR and MCDRAM in the same application.
  - Which data structures should go where?
  - MCDRAM is a limited resource and tracking it adds complexity.

MCDRAM Functional Emulation

- Install jemalloc and memkind on a system with two CPU sockets.
- Configure the system to allocate in “far” memory (i.e. the other socket) by default, and in “near” memory (i.e. this socket) for hbw_malloc:
  - export MEMKIND_HBW_NODES=0
  - numactl –membind=1 –cpunodebind=0 ./application
- Do **not** optimize an application for this setup:
  - This is **not** an accurate model of the bandwidth and latency characteristics MCDRAM
  - but is a reasonable way to determine which data structures rely critically on bandwidth.

SUMMARY

Summary

- Knights Landing is a high-throughput successor to Knights Corner:
  - Socketable, bootable processor with access to large amounts of RAM
  - Greatly improved single-thread performance
  - Very high bandwidth, flexible MCDRAM
  - Optional on-chip interconnect (Omni Path)
- Much of Knights Landing's throughput comes from parallelism:
  - Codes will need to be modernized to fully exploit the features of the chip
  - Knights Corner has parallelism at similar scales and is the best proxy for performance

intel®

experience
what's inside™