An Introduction to Knights Landing

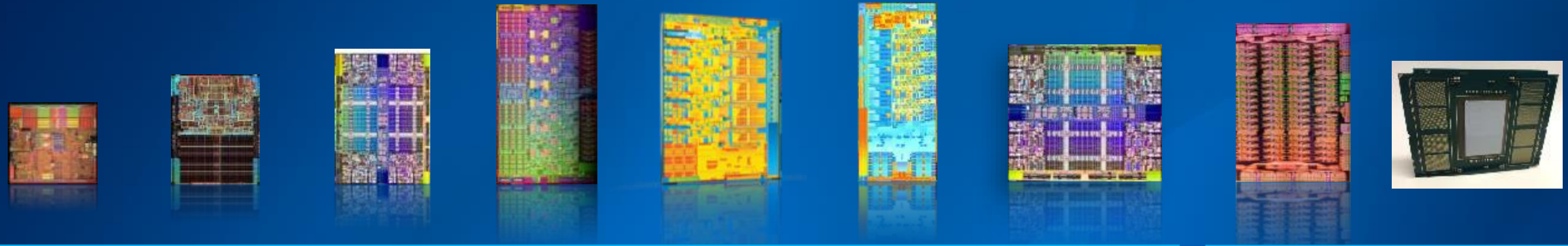
Jason Sewall & TCG Micro

September 2015

Agenda

- Single-node parallelism
- Introduction to Knights Landing
  - MCDRAM
  - AVX-512
- Preparing for Knights Landing
  - SDE
  - Hardware as a Proxy
- Summary

Increasing parallelism in Xeon and Xeon Phi

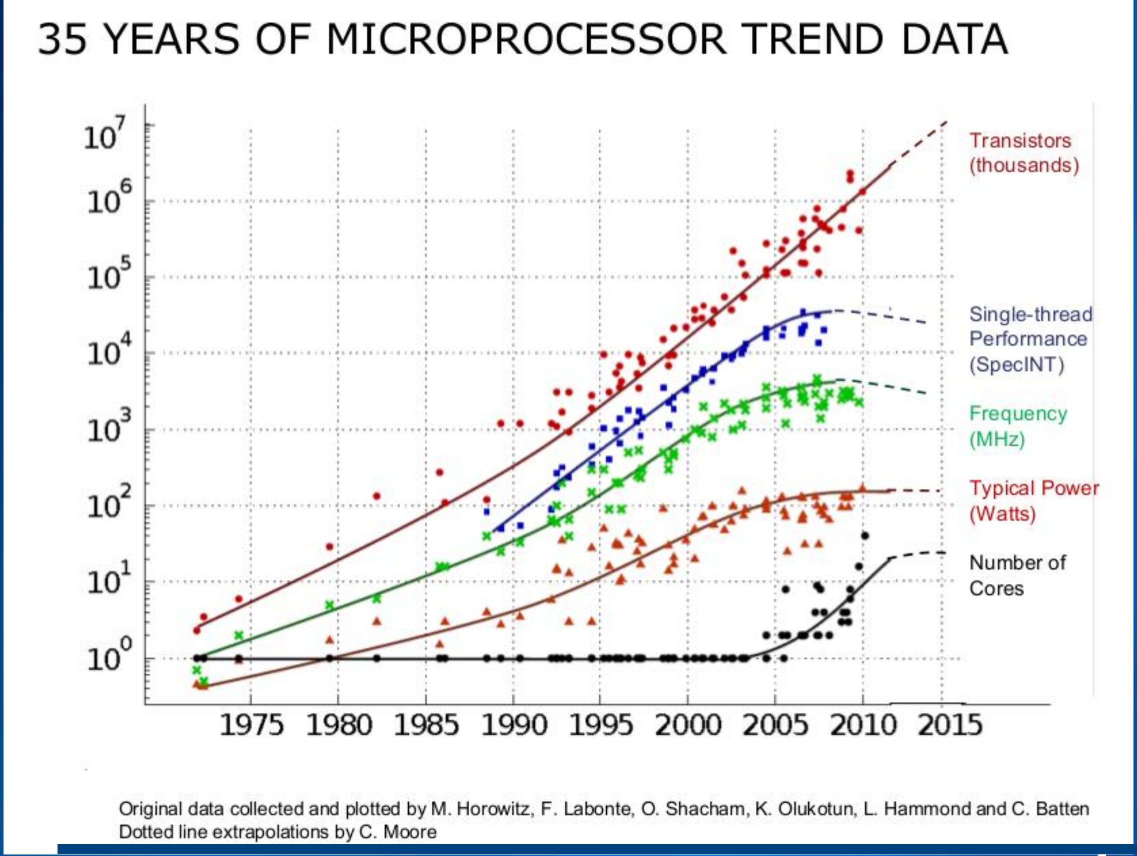


	Intel® Xeon® processor 64-bit series	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge EP	Intel® Xeon® processor code-named Ivy Bridge EP	Intel® Xeon® processor code-named Haswell EX	Intel® Xeon Phi™ coprocessor Knights Corner	Intel® Xeon Phi™ processor & coprocessor Knights Landing[1]
Core(s)	1	2	4	6	8	12	18	61	72
Threads	2	2	8	12	16	24	36	244	288
SIMD Width	128	128	128	128	256	256	256	512	512

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

1. Not launched.

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

Parallelism and Performance

Peak GFLOP/s in Single Precision

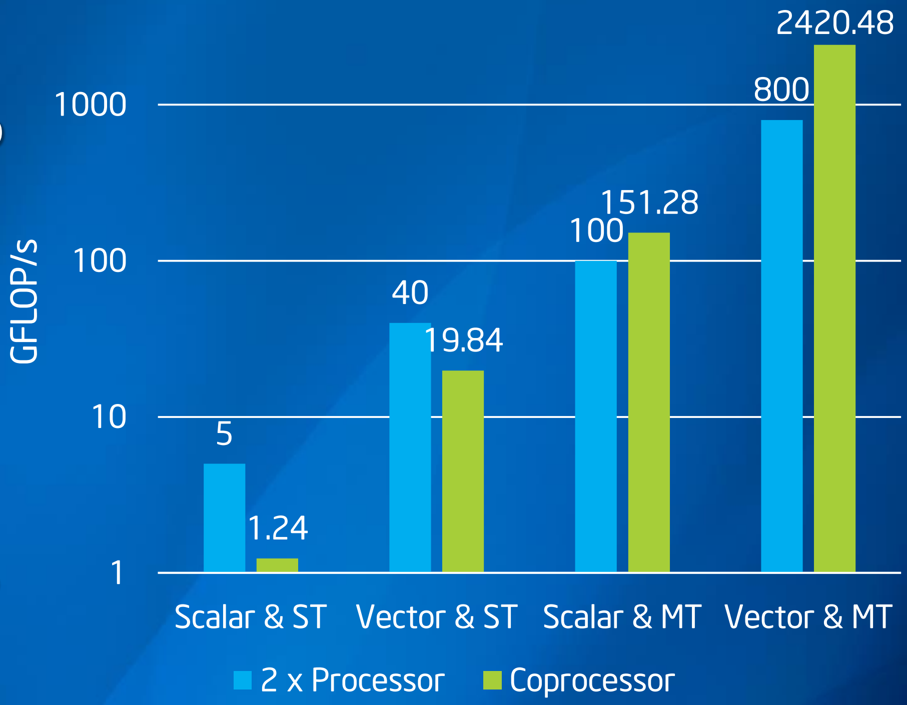
- Clock Rate x Cores x Ops/Cycle x SIMD

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



Note the logarithmic scale on the y-axis.
ST = Single Thread, MT = Multiple Threads

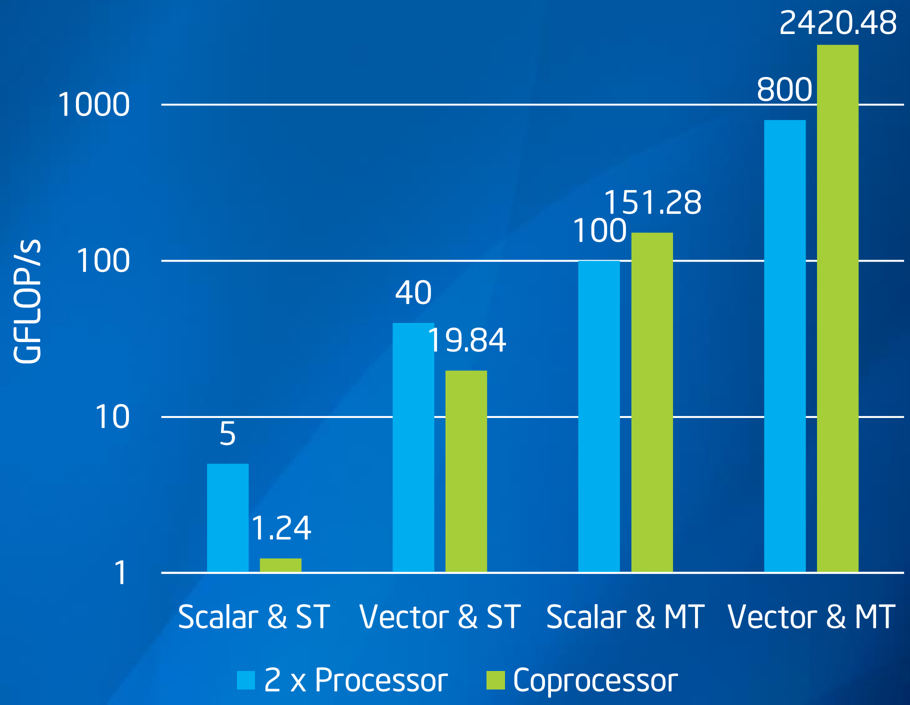
Parallelism and Performance

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”).



Note the logarithmic scale on the y-axis.
ST = Single Thread, MT = Multiple Threads

Parallel concepts

Parallel computing uses multiple computing units in parallel to

- Solve problems more quickly than a single processor ("strong scaling")
- Solve larger problems in the same time as a single processor ("weak scaling")
- Solve problems with higher fidelity

High-performance parallel computing is hard and requires

- Finding enough parallelism
- Deciding the optimal granularity, locality and load balance
- Coordination and synchronization

Real-world applications/algorithms are complex and often hierarchical: monolithic programming model is limited

Amdahl’s Law

$$S(N) = \frac{1}{(1-P) + \frac{P}{N}}$$

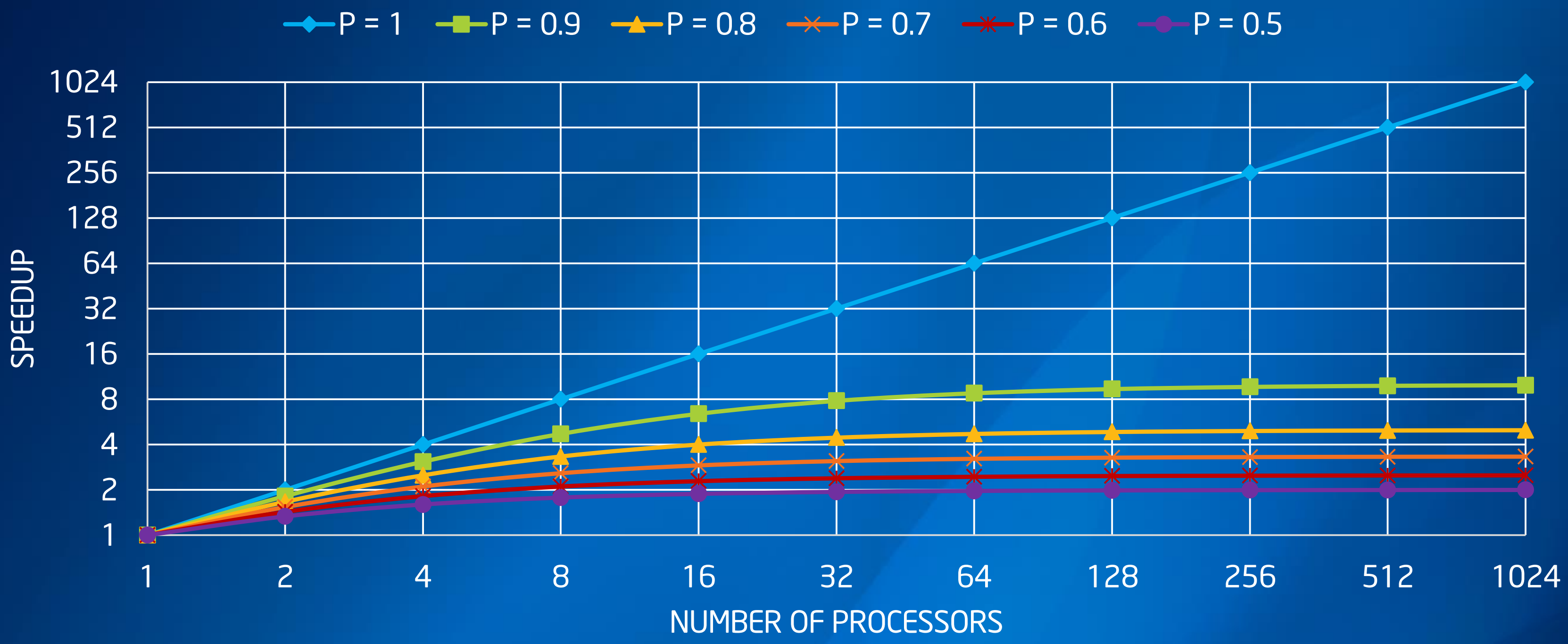
where:

- $S(N)$ = speedup on N processors
- P = fraction of code that can be parallelised
- N = number of processors

The speedup of “strong scaling” applications is governed by Amdahl’s Law.

As $N \rightarrow \infty$, $S(N) \rightarrow \frac{1}{(1-P)}$.

Impact of Amdahl’s Law



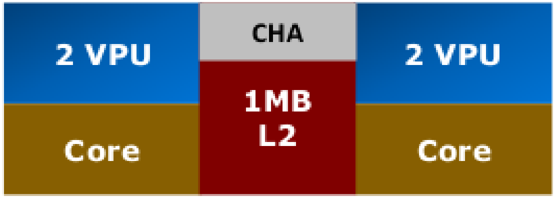
Amdahl’s Law in Practice

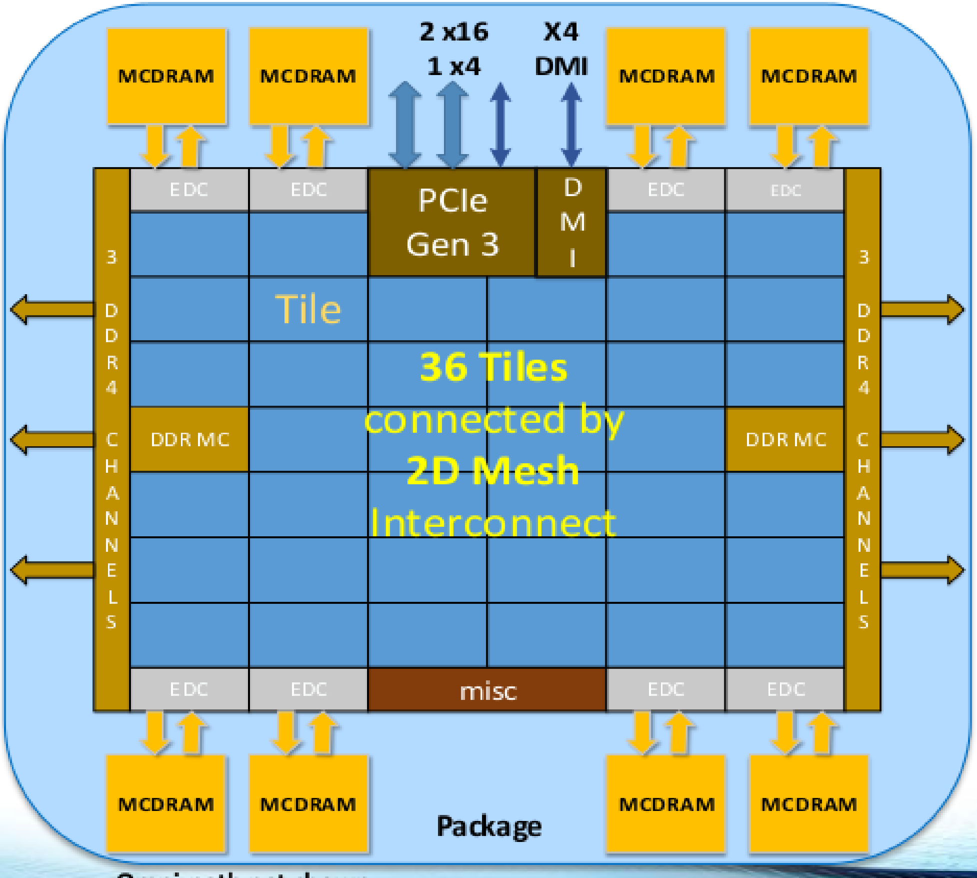
- Assumption that P and N are independent is unrealistic.
  - Strong Scaling:
    All-to-all communication costs increase with N.
    For sufficiently large N, applications will start to slow down again!
  - Weak Scaling:
    Increasing problem size may not linearly increase compute time.
- Key takeaway from both laws:
  maximize P to maximize efficiency and performance at scale.
- Parallelism “bolted on” to scalar applications will not scale.

What is Knights Landing?

Knights Landing Overview

TILE





Omni-path not shown

Chip: 36 Tiles interconnected by **2D Mesh**

Tile: 2 Cores + 2 VPU/core + 1 MB L2

Memory: MCDRAM: 16 GB on-package; High BW

DDR4: 6 channels @ 2400 up to 384GB

IO: 36 lanes PCIe Gen3. 4 lanes of DMI for chipset

Node: 1-Socket only

Fabric: Omni-Path on-package (not shown)

Vector Peak Perf: 3+TF DP and 6+TF SP Flops

Scalar Perf: ~3x over Knights Corner

Streams Triad (GB/s): MCDRAM : 400+; DDR: 90+

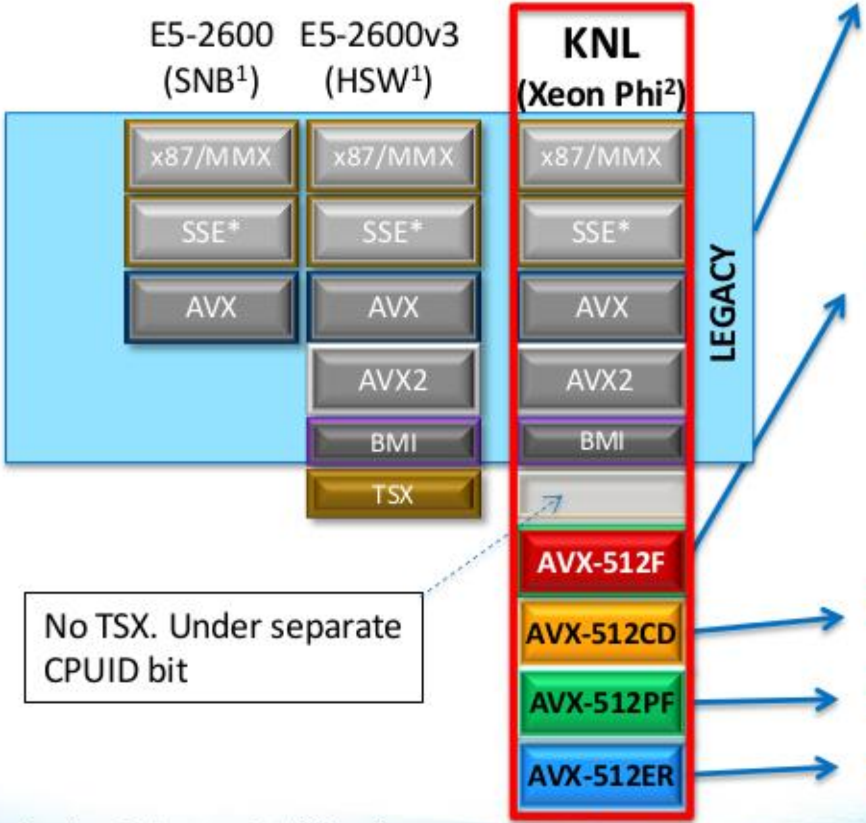
Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). [2]Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. 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.

Many Trailblazing Improvements in KNL

Improvements	What/Why
Self Boot Processor	No PCIe bottleneck
Binary Compatibility with Xeon	Runs all legacy software. No recompilation.
New Core: Atom™ based	~3x higher ST performance over KNC
Improved Vector density	3+ TFLOPS (DP) peak per chip
New AVX 512 ISA	New 512-bit Vector ISA with Masks
Scatter/Gather Engine	Hardware support for gather and scatter
New memory technology: MCDRAM + DDR	Large High Bandwidth Memory → MCDRAM Huge bulk memory → DDR
New on-die interconnect: Mesh	High BW connection between cores and memory
Integrated Fabric: Omni-Path	Better scalability to large systems. Lower Cost

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. 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 configuraton may affect actual performance.

KNL ISA



No TSX. Under separate CPUID bit

KNL implements all legacy instructions

- Legacy binary runs w/o recompilation
- KNC binary requires recompilation

KNL introduces AVX-512 Extensions

- 512-bit FP/Integer Vectors
- 32 registers, & 8 mask registers
- Gather/Scatter

Conflict Detection: Improves Vectorization

Prefetch: Gather and Scatter Prefetch

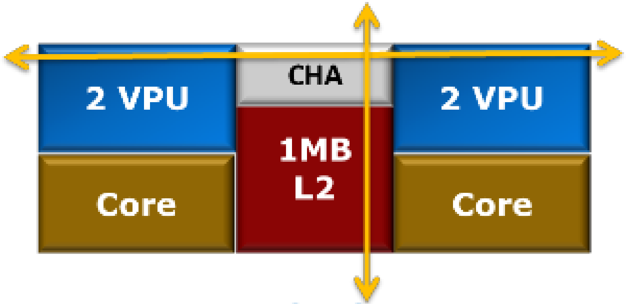
Exponential and Reciprocal Instructions

1. Previous Code name Intel® Xeon® processors
2. Xeon Phi = Intel® Xeon Phi™ processor

KNL Tile:

2 Cores, each with 2 VPU

1M L2 shared between two Cores



Core: Changed from Knights Corner (KNC) to KNL. Based on 2-wide OoO Silvermont™ Microarchitecture, but with *many* changes for HPC.

4 thread/core. Deeper OoO. Better RAS. Higher bandwidth. Larger TLBs.

2 VPU: 2x AVX512 units. 32SP/16DP per unit. X87, SSE, AVX1, AVX2 and EMU

L2: 1MB 16-way. 1 Line Read and ½ Line Write per cycle. Coherent across all Tiles

CHA: **C**aching/**H**ome **A**gent. Distributed Tag Directory to keep L2s coherent. MESIF protocol. 2D-Mesh connections for Tile

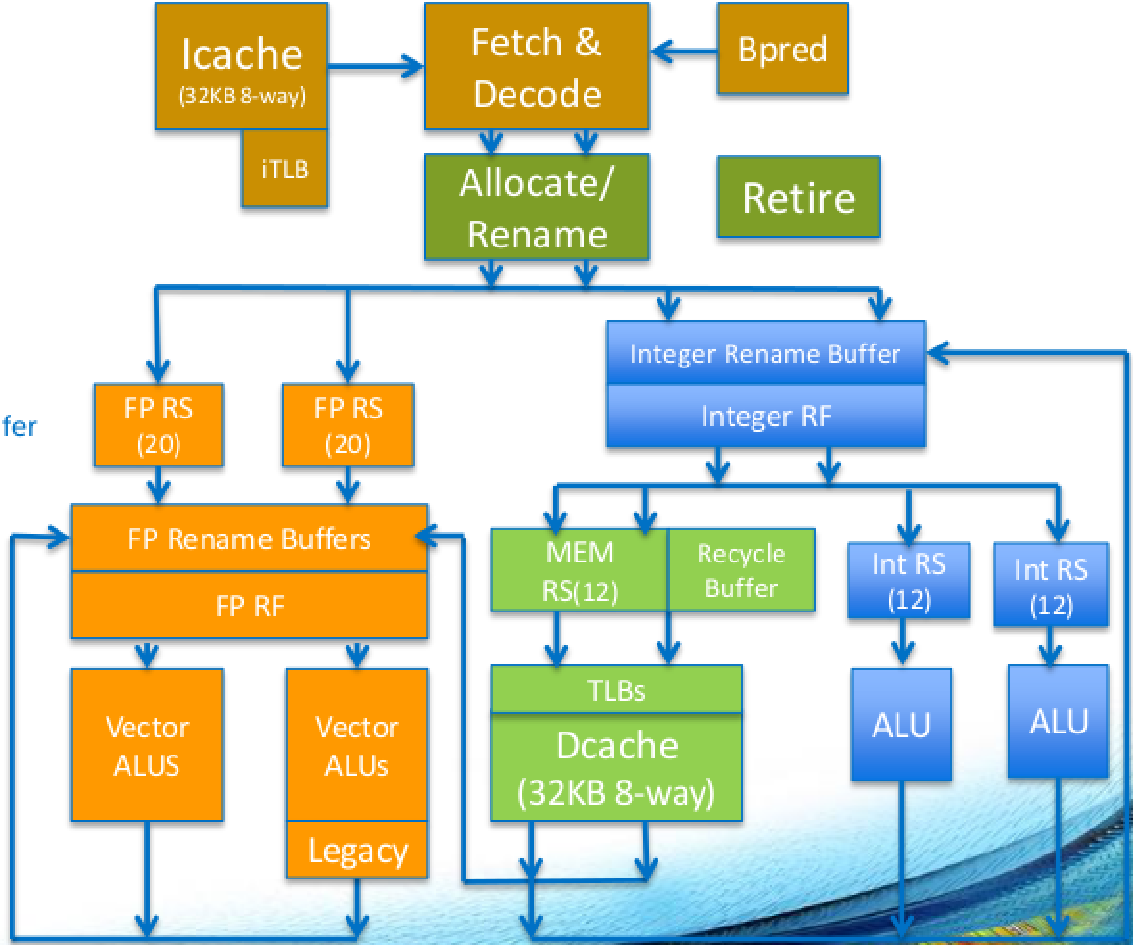
Core & VPU

- Out-of-order core w/ 4 SMT threads
- VPU tightly integrated with core pipeline

- 2-wide Decode/Rename/Retire
- ROB-based renaming. 72-entry ROB & Rename Buffers
- Up to 6-wide at execution
- Int and FP RS OoO.
- MEM RS inorder with OoO completion. Recycle Buffer holds memory ops waiting for completion.
- Int and Mem RS hold source data. FP RS does not.

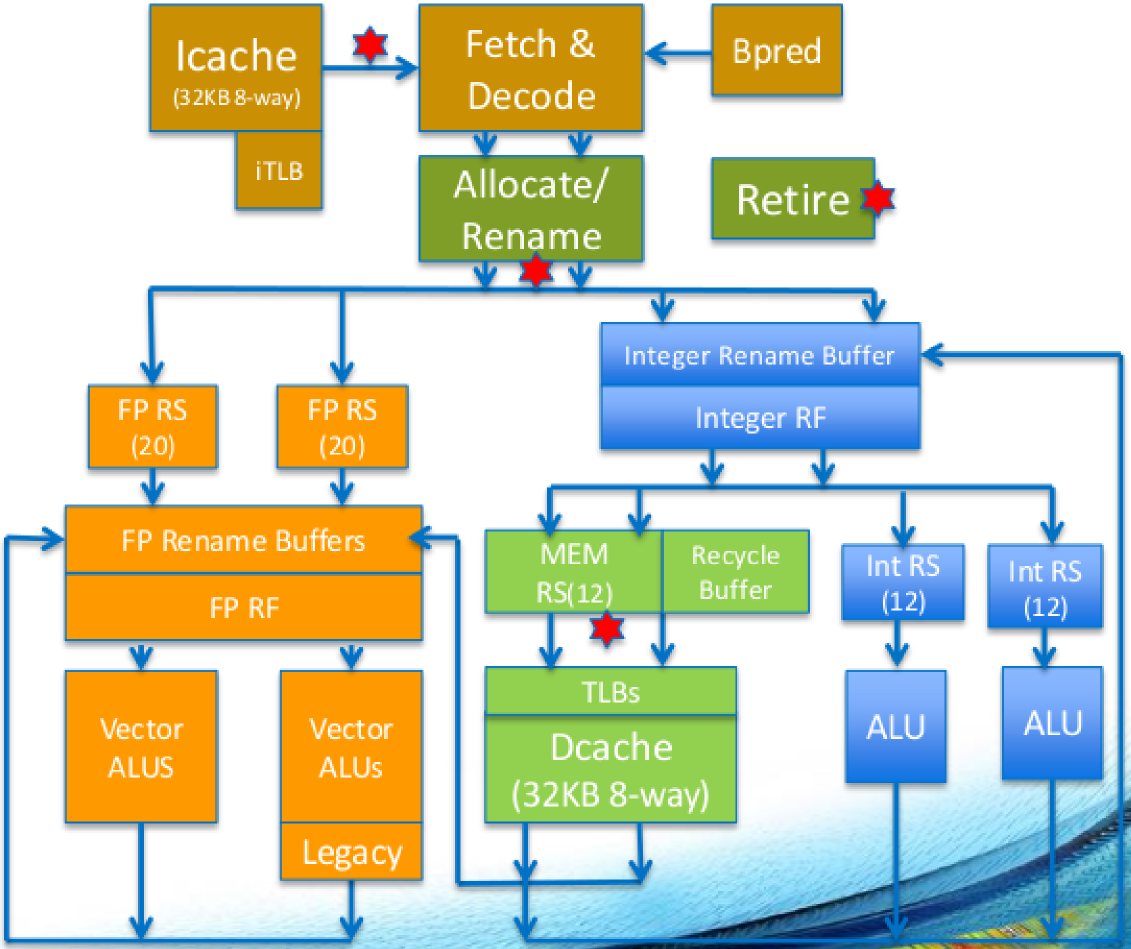
- 2x 64B Load & 1 64B Store ports in Dcache.
- 1st level uTLB: 64 entries
- 2nd level dTLB: 256 4K, 128 2M, 16 1G pages

- L1 Prefetcher (IPP) and L2 Prefetcher.
- 46/48 PA/VA bits
- Fast unaligned and cache-line split support.
- Fast Gather/Scatter support

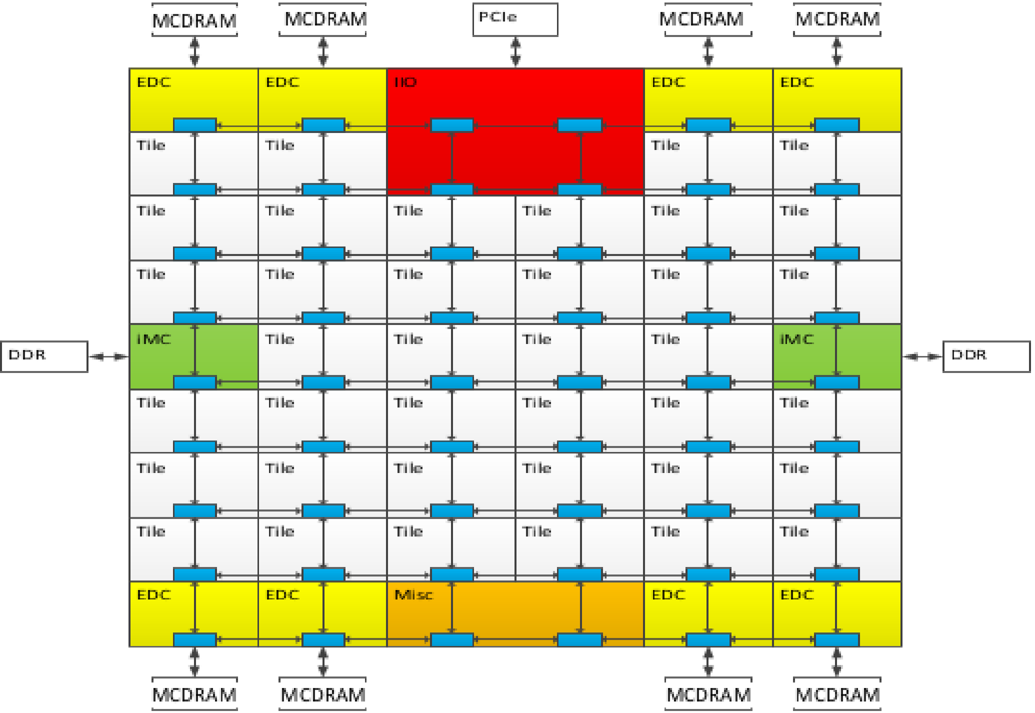


Threading

- 4 Threads per core. Simultaneous Multithreading.
- Core resources **shared** or **dynamically repartitioned** between active threads
  - ROB, Rename Buffers, RS: Dynamically partitioned
  - Caches, TLBs: Shared
  - E.g., 1 thread active → uses full resources of the core
- Several Thread Selection points in the pipeline. (★)
  - Maximize throughput while being fair.
  - Account for available resources, stalls and forward progress



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

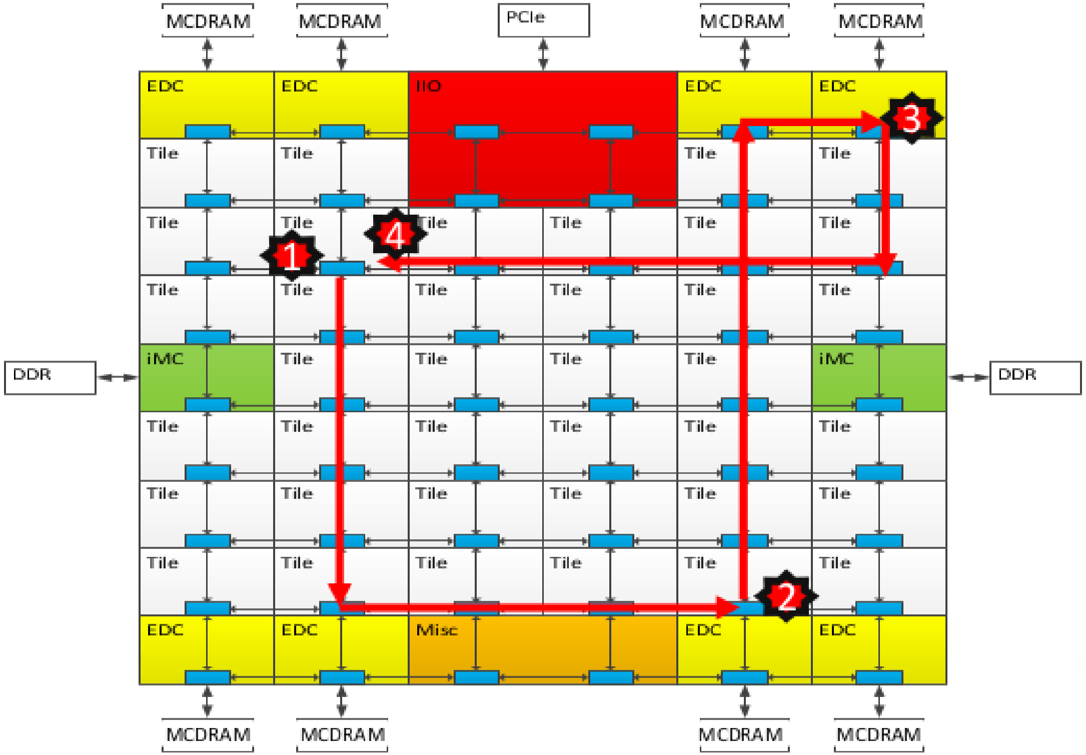
Cache Coherent Interconnect

- 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



Address uniformly hashed across all distributed directories

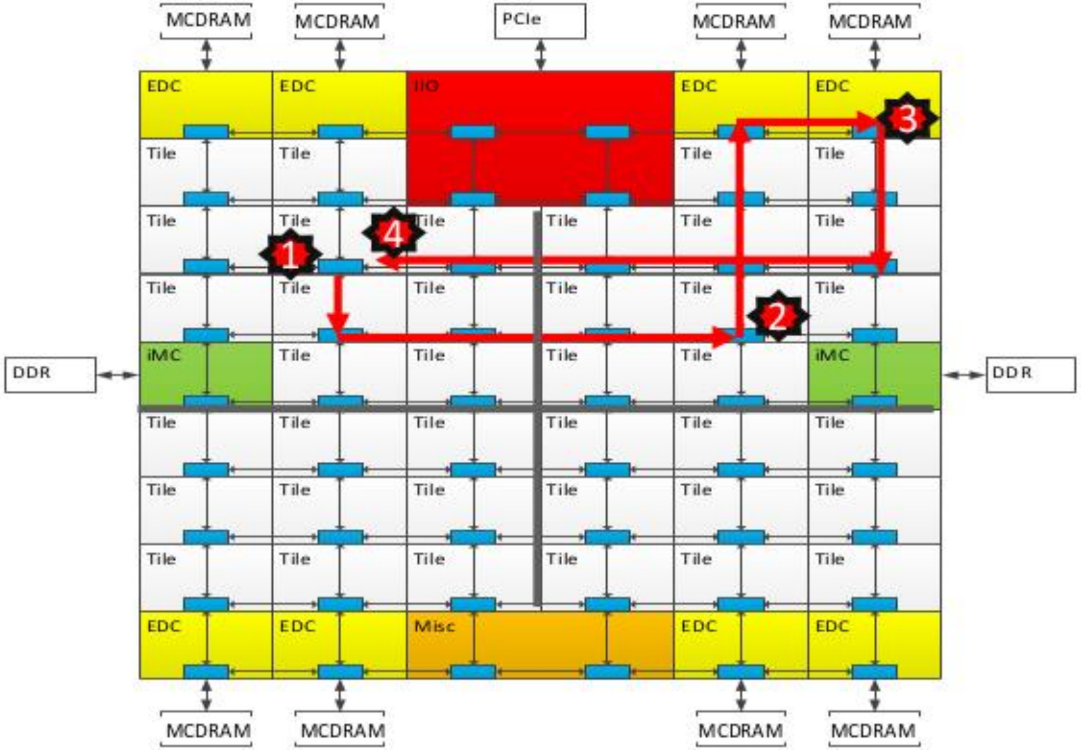
No affinity between Tile, Directory and Memory

Most general mode. Lower performance than other modes.

Typical Read L2 miss

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



Chip divided into four virtual Quadrants

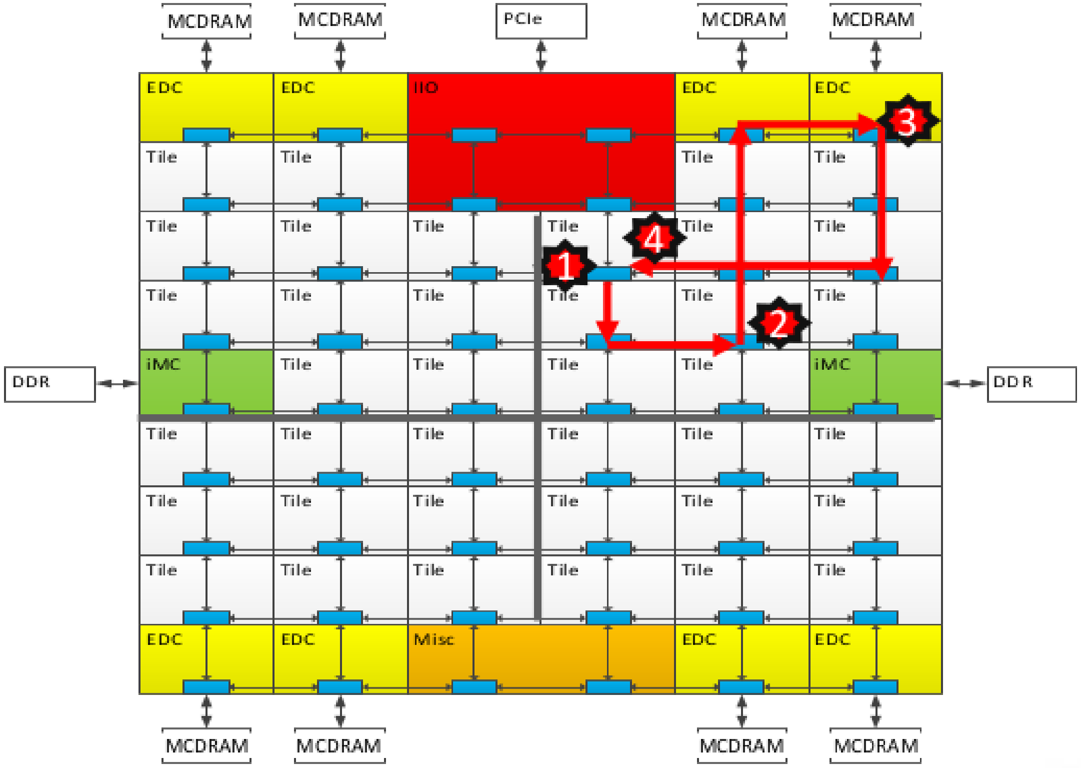
Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

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

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

Cluster Mode: Sub-NUMA Clustering (SNC)



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.

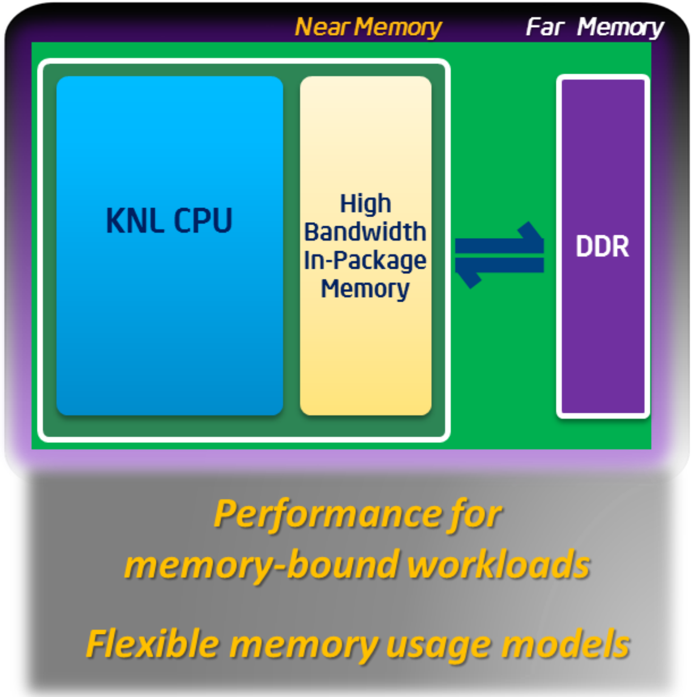
SW needs to NUMA optimize to get benefit.

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

MCDRAM

MCDRAM Overview

- Large numbers of cores can consume large amounts of memory bandwidth.
- Knights Landing is equipped with 6 bidirectional DDR4 memory channels and MCDRAM.
- Memory requests are serviced by a mesh network maintaining cache coherence.
- MCDRAM can be configured as:
  1) a third level cache;
  2) as a flat, distinct region of memory; or
  3) as a hybrid, somewhere in-between.


Near Memory
Far Memory
KNL CPU
High Bandwidth In-Package Memory
DDR
Performance for memory-bound workloads
Flexible memory usage models

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.
Diagram is for conceptual purposes only and only illustrates a CPU and memory – it is not to scale, and is not representative of actual component layout.

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

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.

MCDRAM in Flat Mode

Upside

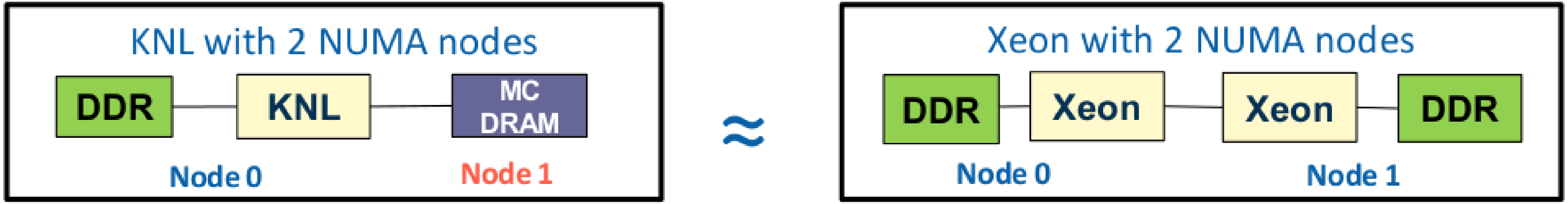
- Maximum bandwidth and latency performance.
- Maximum addressable memory.
- Isolate MCDRAM for HPC application use only.

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.

Flat MCDRAM: SW Architecture

MCDRAM exposed as a separate NUMA node



Memory allocated in DDR by default → Keeps non-critical data out of MCDRAM.

Apps explicitly allocate critical data in MCDRAM. Using two methods:

- **"Fast Malloc"** functions in High BW library (https://github.com/memkind)
  - Built on top to existing *libnuma* API
- **"FASTMEM"** Compiler Annotation for Intel Fortran

Flat MCDRAM with existing NUMA support in Legacy OS

A New ISA: AVX-512

- Knights Landing is the first micro-architecture to support AVX-512.
  - AVX-512F, AVX-512 CDI, AVX-512 ERI, AVX-512 PFI

- Differences from AVX2:
  - 32 x 512 bit SIMD registers (zmm0 – zmm31)
  - Dedicated mask registers (k0 – k7)
  - New instructions: gather/scatter (F), expand/compress (F), conflict detection (CDI), exponential and reciprocal (ERI), prefetch (PFI)

- Differences from IMCI (Knights Corner ISA):
  - Backwards compatible with SSE/AVX.
  - New instructions: conflict detection (CDI)

Conflict Detection Instructions (CDI)

- Sparse updates (e.g. `A[B[i]]++`) are common, but hard to vectorize.

- Naively vectorized, loops look like this:

```
for (i=0; i<N; i+=16)
{
  index = vload &B[i]                // Load 16 B[i]
  old_val = gather A, index          // Grab A[B[i]]
  new_val = vadd old_val, some_value // Compute new values
  scatter A, index, new_val          // Update A[B[i]]
}
```

- Code is **wrong** if any values within “index” are duplicated.

Conflict Detection Instructions (CDI)

- AVX-512 CDI introduces three new instructions:
  - `vpconflict{d,q} zmm1 {k1}, zmm2/mem`
    Compares (for equality) each element in zmm2 with “earlier” elements and outputs bit vector.
  - `vpbroadcastm{b2q,w2d} zmm1 {k0}, to_do`
  - `vplzcnt{d,q}`
  - `vptestnm{d,q} k2 {k1}, zmm1, zmm2/mem`
    (from AVX-512F)

Manipulate bit vector from vpconflict to construct a useful mask.

Conflict Detection Instructions (CDI)

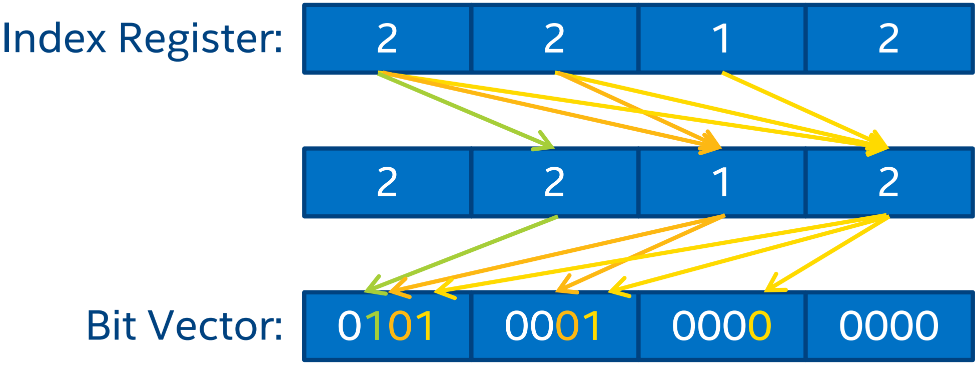
- Vectorization with these instructions looks like this:

```
for (int i = 0; i < N; i += 16)
{
  __m512i indices = vload &index_array[i]
  vpconflictd comparisons, indices // comparisons = __m512i
  __mmask to_do = 0xffff;

  do
  {
      vpbroadcastmd tmp, to_do // tmp = __m512i
      vptestnmd mask {to_do}, comparisons, tmp
      do_work(mask); // gather-compute-scatter
      to_do ^= mask;
  } while(to_do);
}
```

Do work for element iff no conflicts on **remaining** earlier elements.

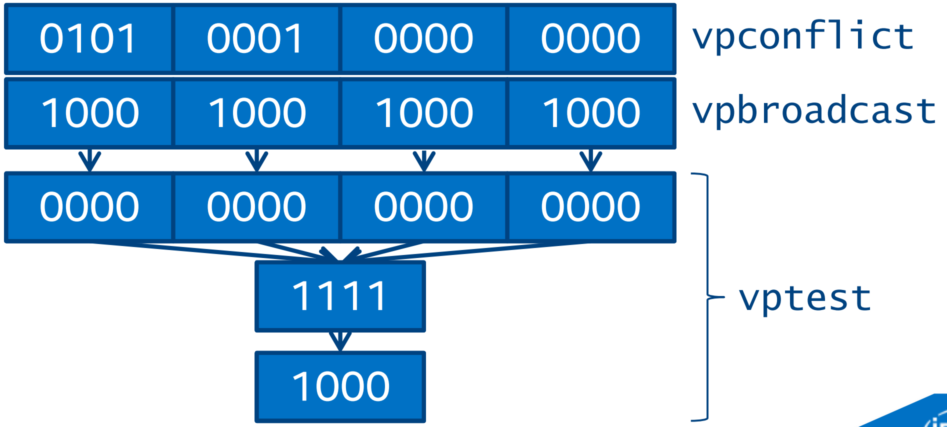
Conflict Detection Instructions (CDI) - Example



Index Register:	2	2	1	2
	2	2	1	2
Bit Vector:	0101	0001	0000	0000

1) Compare (for equality) each element in zmm2 with “**earlier**” elements and output bit vector.

2) Combine bit vector and todo to work out which elements can be updated in **this iteration**.



0101	0001	0000	0000	vpconflict
1000	1000	1000	1000	vpbroadcast
0000	0000	0000	0000	vptest

1111

1000

3) Loop until todo is 0000.

Conflict Detection Instructions (CDI) – Compiler

- The Intel® compiler (15.0 onwards) will recognise potential run-time conflicts and generate vpconflict loops **automatically**:

```
for (int i = 0; i < N; ++i)
{
     histogram[index[i]]++;
}
```

- Such loops would originally have resulted in:

```
remark #15344: loop was not vectorized: vector dependence prevents vectorization
remark #15346: vector dependence: assumed FLOW dependence between histogram line 22 and histogram line 22
remark #15346: vector dependence: assumed ANTI dependence between histogram line 22 and histogram line 22
```

- If you **know** that conflicts cannot occur, you should still specify this: (e.g. #pragma ivdep, #pragma simd, #pragma omp simd)

AVX-512 – Summary

- Performance impact of AVX-512:
  - 2x vector width (vs AVX2)
  - Improved gather/scatter efficiency (vs mov/insertps/extractps sequence and IMCI)
  - Improved control divergence management via masking (vs AVX2)

- Programmability impact of AVX-512:
  - Auto-vectorization of loops with potential dependencies (via vconflict)
  - Improved vectorization of outer loops with complicated flow (via masking)

Preparing for Knights Landing

Preparing for Knights Landing

- 1st Knights Landing systems appearing by end of 2015
- 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, AVX-512
  - High-bandwidth MCDRAM
- Plenty of **parallelism** needed for best performance.

Proxies for Knights Landing

- Two pronged approach:

1. Check functional correctness with software tools:
   - Intel® Composer Studio XE 2015 compiles for Knights Landing (-xMIC-AVX512)
   - Intel® Software Development Emulator (SDE) functionally emulates; rough perf. analysis
   - memkind & hbw_malloc will harness MCDRAM

2. Test performance/scalability with the best real-world proxy:
   - Current generation Intel® Xeon Phi™ coprocessor ("Knights Corner")

Intel® Software Development Emulator (SDE)

Intel® Software Development Emulator (SDE)

- Freely available instruction emulator:
  - http://www.intel.com/software/sde

- 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.

- Produces a profile of the dynamic instruction mix; useful for tuning/assessing vectorization content.

Running SDE

- SDE invocation is very simple:

```
$ sde <sde-opts> -- <binary> <command args>
```

- CPUID is the same as the host (by default) but can be overridden:
  - -knl for Knights Landing
  - -hsw for Haswell

- SDE can summarize the types of instructions that were executed:

```
$ sde <sde-opts> -omix <output-file> -- <binary> <command args>
```

- The output file contains lots of information on the instructions executed by the program as seen in the following slides

SDE – Instruction Mix Output File

- Information for each thread in the program:
  - Basic block execution counts, addresses, disassembly, etc.
  - Function-level profile
  - Instruction mix by category, e.g.: category-AVX512
  - Dynamic and static instruction counts
- Global static instruction counts by category
- Global dynamic instruction counts by category

SDE – Basic Block Statistics

```
# ==============================================
# STATS FOR TID 0 EMIT# 1
# ==============================================
# EMIT_TOP_BLOCK_STATS FOR TID 0 EMIT # 1 EVENT=ICOUNT
BLOCK: 00000   PC: 0000000000410c23   ICOUNT: 15983666400   EXECUTIONS: 270909600   #BYTES: 272   %:    15   cumltv%:
15  FN: swUpdatePress2ndTiltedZ_DDz1_8  IMG: swell-TTC2-12x12x8 Source: swUpdatePress2ndTilted-orig.tc 274,273
XDIS 0000000000410c23:  SSE 430F101498             movups xmm2, xmmword ptr [r8+r11*4]
XDIS 0000000000410c28:  SSE 420F101C98             movups xmm3, xmmword ptr [rax+r11*4]
XDIS 0000000000410c2d:  SSE 0F59D1                 mulps xmm2, xmm1
XDIS 0000000000410c30:  SSE 410F59DF               mulps xmm3, xmm15
XDIS 0000000000410c34: BASE 4C8BBC2468010000       mov r15, qword ptr [rsp+0x168]
XDIS 0000000000410c3c:  SSE 0F58D3                 addps xmm2, xmm3
XDIS 0000000000410c3f:  SSE 430F105C9D00           movups xmm3, xmmword ptr [r13+r11*4]
```

- ICOUNT: total dynamic instructions executed by this basic block. Basic blocks are sorted by ICOUNT from highest to lowest.

- EXECUTIONS: number of times a basic block is invoked.

- %: percent of total instructions that come from this basic block.

- cumltv%: cumulative % of instruction count up to this basic block.

SDE – Basic Block Statistics

- Look for unpack *ss or *sd instructions (i.e., scalar instructions) in top basic blocks
- Are they SSE, AVX, AVX2, AVX512 instructions?
  - For KNL, we want as many AVX512 instructions as possible
  - Sometimes, non AVX512 instructions come from math libraries and some math functions are not yet optimized for AVX512.
- Are there gathers/scatters (non-unit stride)?
  - Can code be transformed to remove gathers/scatters?

SDE - Summary

- SDE is good for:
  - Correctness testing.
  - Investigating instruction inefficiencies (e.g. expensive instruction sequences, poor vectorization, vector length issues)
  - Quantifying expected vector speed-up (i.e. reduction in # instructions)
  - Investigating dynamic mask values.

- SDE is not good for:
  - Performance projections.
  - Identifying some performance bottlenecks.
    Instruction mix gives only counts and does not account for execution bottlenecks (e.g., memory issues).

Hardware as a Performance Proxy

Hardware as a Performance Proxy

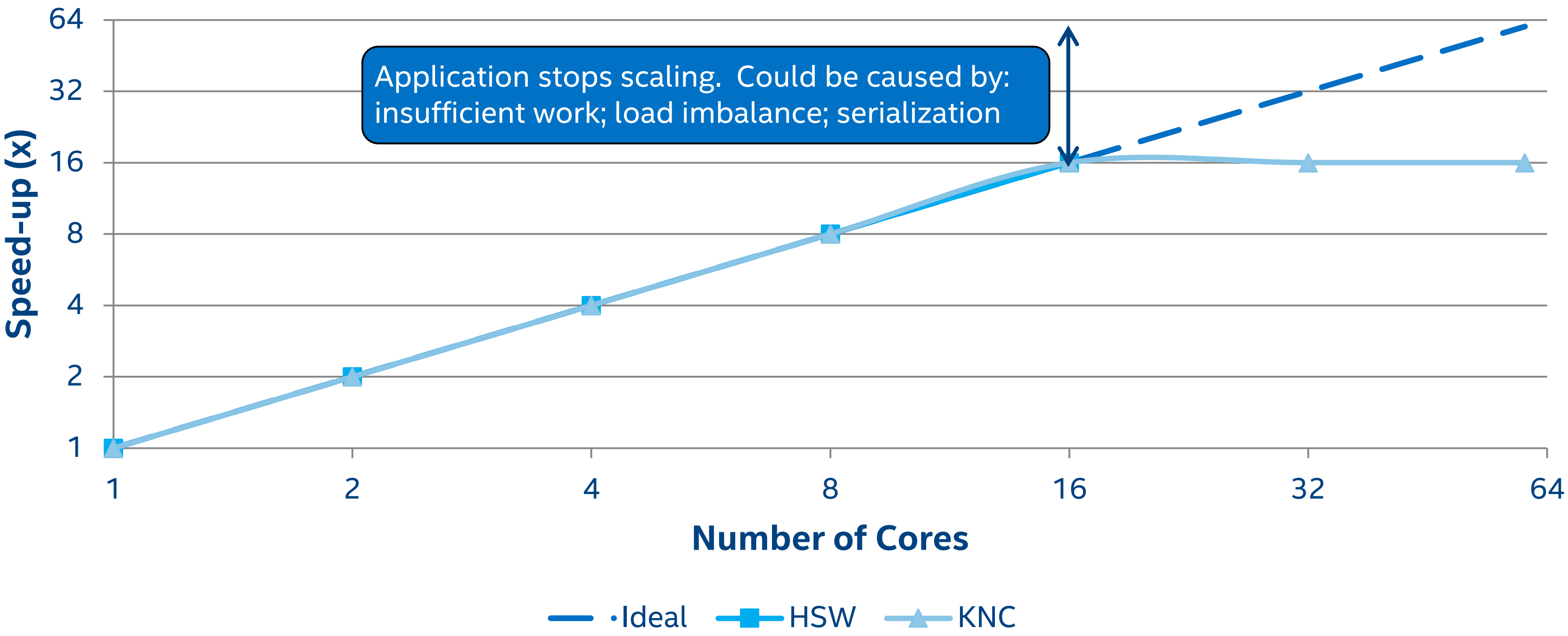
- Functional emulation and advanced APIs are invaluable for testing out new instructions and features.
- Performance testing is very important, too.
- How will your code run on Knights Landing?
  - Is it compute-bound? Memory bandwidth-bound?
  - How will the performance features of Knights Landing affect those limits?
- Is there an **existing processor** that is a good proxy for Knights Landing for your code?

Picking the Right Proxy

	Intel® Xeon® E5-2696v3	**Intel® Xeon Phi™ 7120**	**Knights Landing**
Cores/threads	14/28	61/244	60+/240+
Nom. Hz	2.6GHz	1.3GHz	N/A
STREAM BW	~50 GB/s	~170 GB/s	>400 GB/s (MCDRAM)
SIMD width	256 bits	512 bits	512 bits
LLC capacity	35MB	30.5MB	30+MB
DRAM cap.	768GB	16GB	384GB

- Knights Corner is the best proxy for codes that are:
  - Compute-bound
  - Bandwidth-bound
- Intel® Xeon® processors are the best proxy for codes that are:
  - Limited by memory capacity
- Capacity-limited codes may still benefit from Knights Landing – consider using a **reduced problem size** for tuning.

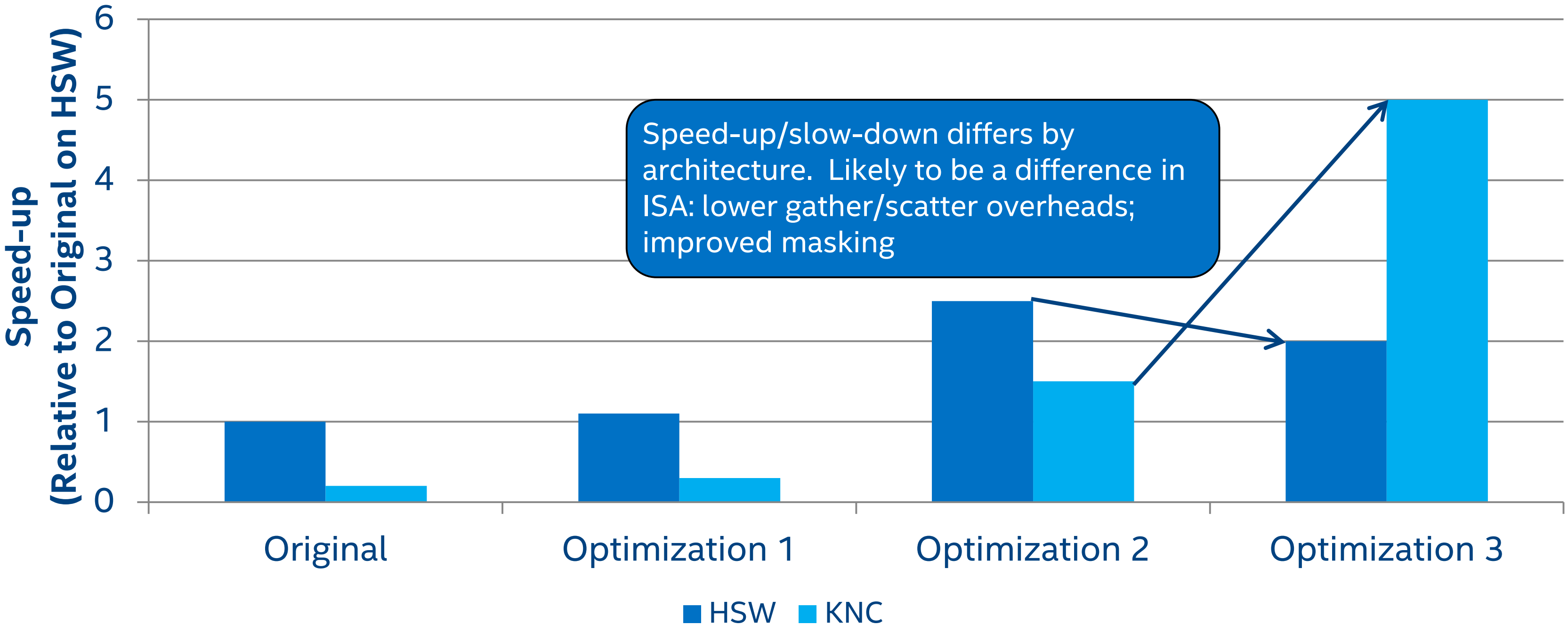
Picking the Right Proxy – Thread Scaling



Application stops scaling. Could be caused by: insufficient work; load imbalance; serialization

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Picking the Right Proxy – SIMD ISA



Speed-up/slow-down differs by architecture. Likely to be a difference in ISA: lower gather/scatter overheads; improved masking

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Picking the Right Proxy - Summary

- Knights Corner is almost always the **closest proxy** to Knights Landing
  - Native execution on KNC = self-hosted KNL
  - Offload to KNC = offload to KNL

- Performance of Knights Corner, like Knights Landing, is driven by:
  - **Thread Scalability**
(i.e. load balancing, divergence, inter-core communication costs)
  - **Efficient SIMD Vectorization**
(i.e. data layout/alignment, compiler smarts)
  - **High Bandwidth Memory**
(i.e. GDDR5)

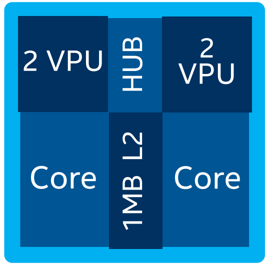
Performance Optimization

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.
- Memory:
  - Tile your algorithm to take advantage of data reuse
  - Layout data to avoid TLB misses (AOS vs. SOA, large pages)
  - Consider the need for software prefetches
  - Understand how threading affects cache and TLB pressure

Nested Parallelism and Locality

- 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)
      {
            …
      }
}
```

Cluster Modes

- KNL has three on-die cluster modes:
  - All-to-All
  - Quadrant (Default)
  - Sub-NUMA Clustering (SNC)
- SNC exposes each quadrant (9 tiles) as a separate NUMA region.
- Running **one MPI rank per NUMA region** (when in SNC mode) will ensure locality-of-access, and may improve 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
  - Power-efficient

- 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**

Legal Disclaimers

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Knights Landing and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user

Intel, Look Inside, Xeon, Intel Xeon Phi, Pentium, Cilk, VTune and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Legal Disclaimers

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 SSE3 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

Legal Disclaimers

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. For more information go to http://www.intel.com/performance.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more athttp://www.intel.com/go/turbo.

Estimated Results Benchmark Disclaimer:
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.

Software Source Code Disclaimer:
Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Legal Disclaimers

The above statements and any others in this document that refer to plans and expectations for the third quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company’s expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel’s and competitors’ products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the company’s most recent reports on Form 10-Q, Form 10-K and earnings release.

Rev. 7/17/13