



Xeon and Xeon Phi



Xeon (Broadwell)

Intel "Tick-Tock" Roadmap – Part I

Intel® Core™
MicroArchitecture

Micro Architecture
Codename "Nehalem"

2nd Generation
Intel® Core™
Micro Architecture

3rd Generation
Intel® Core™
Micro Architecture

Merom

Penryn

Nehalem

Westmere

Sandy Bridge

Ivy Bridge

NEW
Micro architecture
65nm

NEW
Process Technology
45nm

NEW
Micro architecture
45nm

NEW
Process Technology
32nm

NEW
Micro architecture
32nm

NEW
Process Technology
22nm

TOCK

TICK

TOCK

TICK

TOCK

TICK

2006
SSSE-3

2007
SSE4.1

2008
SSE4.2

2009
AES

2011
AVX

2012
RDRAND
etc

Intel "Tick-Tock" Roadmap – Part II

Future Release Dates & Features subject to Change without Notice !

4 th Generation Intel® Core™ Micro Architecture	TBD	TBD	TBD	TBD	TBD
Haswell NEW Micro architecture 22nm	Broadwell NEW Process Technology 14nm	Skylake NEW Micro architecture 14nm	TBD NEW Process Technology 10nm	TBD NEW Micro architecture 10nm	TBD NEW Process Technology 7nm
TICK	TOCK	TICK	TOCK	TICK	TOCK
2013	2015 ! 5 new Inst.	2017	???	???	???

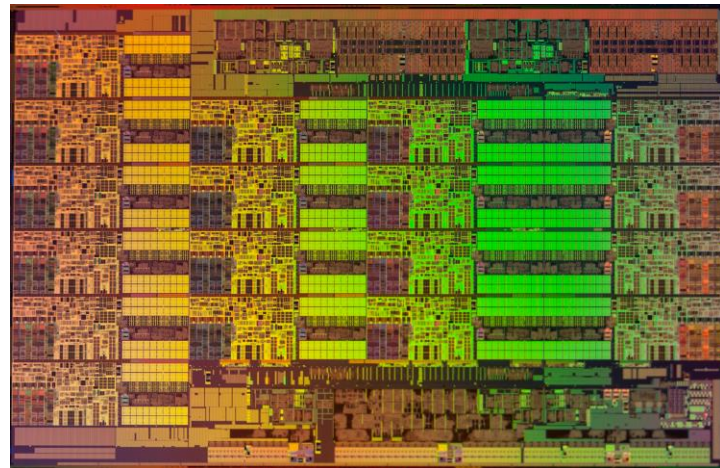
AVX-2

Current Intel® Xeon Platform - Broadwell

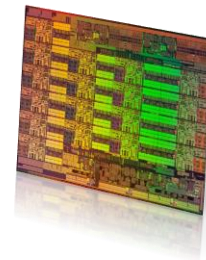
Xeon

Latest released – Broadwell (14nm process)

- Intel's Foundation of HPC Performance
- Up to 22 cores, Hyperthreading
- ~66 GB/s stream memory BW (4 ch. DDR4 2400)
- AVX2 – 256-bit (4 DP, 8 SP flops) -> >0.7 TFLOPS
- 40 PCIe lanes

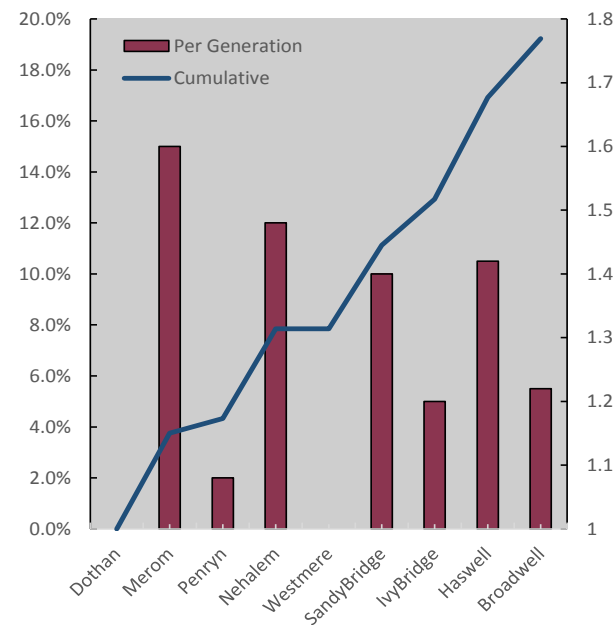


Intel® Xeon® Processors

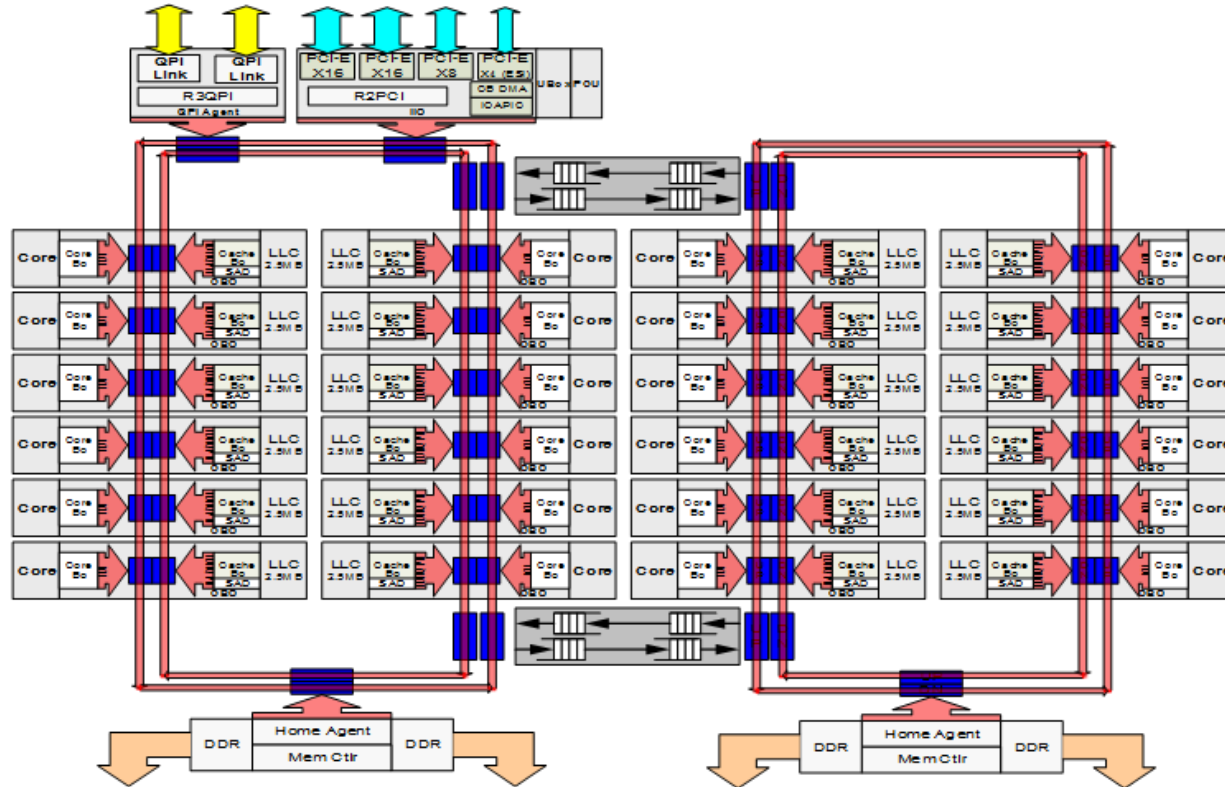


Feature	Xeon E5-2600 v3 (Haswell-EP, 22nm)	Xeon E5-2600 v4 (Broadwell-EP, 14nm)
Cores Per Socket	Up to 18	Up to 22
Threads Per Socket	Up to 36 threads	Up to 44 threads
Last-level Cache (LLC)	Up to 45 MB	Up to 55 MB
QPI Speed (GT/s)	2x QPI 1.1 channels 6.4, 8.0, 9.6 GT/s	
PCIe* Lanes/Controllers/Speed(GT/s)	40 / 10 / PCIe* 3.0 (2.5, 5, 8 GT/s)	
Memory Population	4 channels of up to 3 RDIMMs or 3 LRDIMMs	+ 3DS LRDIMM^{&}
Max Memory Speed	Up to 2133	Up to 2400
TDP (W)	160 (Workstation only), 145, 135, 120, 105, 90, 85, 65, 55	

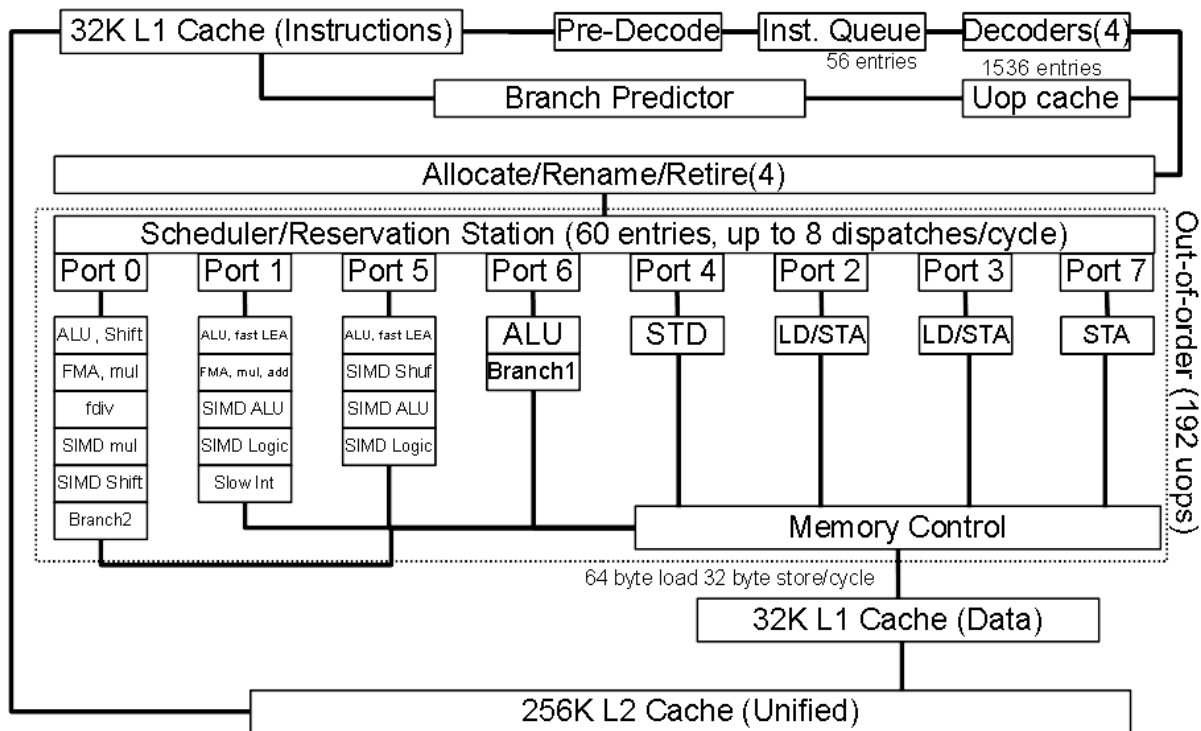
Core Single Thread IPC Performance



On-Chip Interconnect Architecture










Broadwell/Haswell Core Pipeline

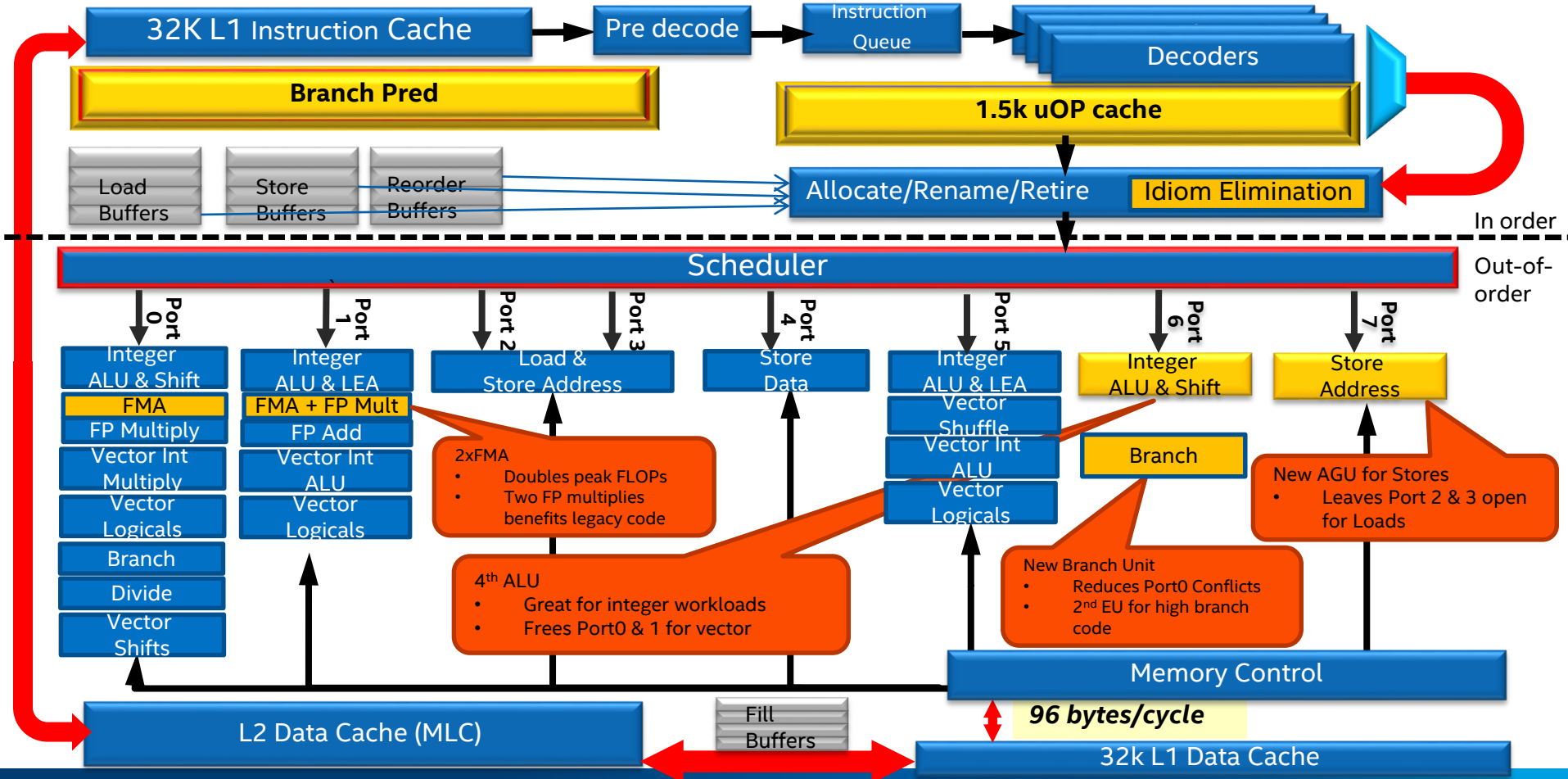


Haswell/Broadwell Buffer Sizes

Extract more parallelism in every generation

	Nehalem	Sandy Bridge	Haswell	
Out-of-order Window	128	168	192	
In-flight Loads	48	64	72	
In-flight Stores	32	36	42	
Scheduler Entries	36	54	60	
Integer Register File	N/A	160	168	
FP Register File	N/A	144	168	
Allocation Queue	28/thread	28/thread	56	

Haswell and Broadwell Core Microarchitecture



Intel® Xeon® Processor E5 v4 Family: Core Improvements

Extract more parallelism in scheduling uops

- Reduced instruction latencies (ADC, CMOV, PCLMULQDQ)
- Larger out-of-order scheduler (60->64 entries)
- New instructions (ADCX/ADOX)

Improved performance on large data sets

- Larger L2 TLB (1K->1.5K entries)
- New L2 TLB for 1GB pages (16 entries)
- 2nd TLB page miss handler for parallel page walks

Improved address prediction for branches and returns

- Increased Branch Prediction Unit Target Array from 8 ways to 10

Floating Point Instruction performance improvements

- Faster vector floating point multiplier (5 to 3 cycles)
- 1024 Radix divider for reduced latency, increased throughput
- Split Scalar dividers for increased parallelism/bandwidth
- Faster vector Gather

**Broadwell:
What's new**



All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. Intel may make changes to specifications and product descriptions at any time, without notice.

Core Cache Size/Latency/Bandwidth

Metric	Nehalem	Sandy Bridge	Haswell
L1 Instruction Cache	32K, 4-way	32K, 8-way	32K, 8-way
L1 Data Cache	32K, 8-way	32K, 8-way	32K, 8-way
Fastest Load-to-use	4 cycles	4 cycles	4 cycles
Load bandwidth	16 Bytes/cycle	32 Bytes/cycle (banked)	64 Bytes/cycle
Store bandwidth	16 Bytes/cycle	16 Bytes/cycle	32 Bytes/cycle
L2 Unified Cache	256K, 8-way	256K, 8-way	256K, 8-way
Fastest load-to-use	10 cycles	11 cycles	11 cycles
Bandwidth to L1	32 Bytes/cycle	32 Bytes/cycle	64 Bytes/cycle
L1 Instruction TLB	4K: 128, 4-way 2M/4M: 7/thread	4K: 128, 4-way 2M/4M: 8/thread	4K: 128, 4-way 2M/4M: 8/thread
L1 Data TLB	4K: 64, 4-way 2M/4M: 32, 4-way 1G: fractured	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way	4K: 64, 4-way 2M/4M: 32, 4-way 1G: 4, 4-way
L2 Unified TLB	4K: 512, 4-way	4K: 512, 4-way	4K+2M shared: 1024, 8-way

New Instructions in Haswell/Broadwell

Group	Description	Count *	
AVX-2	SIMD Integer Instructions promoted to 256 bits	170 / 124	
	Gather		Load elements using a vector of indices, vectorization enabler
	Shuffling / Data Rearrangement		Blend, element shift and permute instructions
FMA	Fused Multiply-Add operation forms (FMA-3)	96 / 60	
Bit Manipulation and Cryptography	Improving performance of bit stream manipulation and decode, large integer arithmetic and hashes	15 / 15	
TSX=RTM+HLE	Transactional Memory	4/4	
Others	MOVBE: Load and Store of Big Endian forms INVPCID: Invalidate processor context ID	2 / 2	

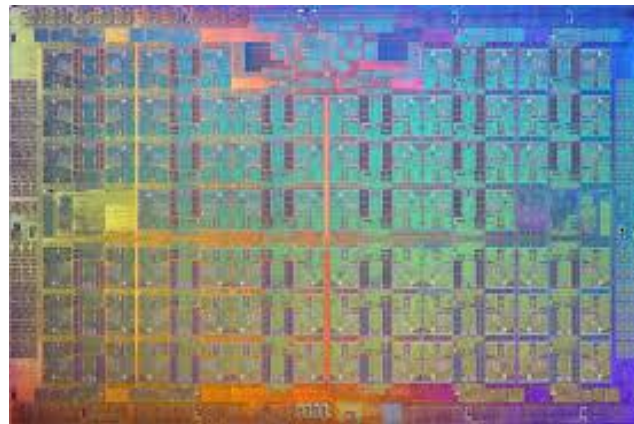
* Total instructions / different mnemonics



Xeon Phi (Knights Landing)

HIGH-LEVEL ARCHITECTURE & INSTRUCTION SET

Current Xeon Phi™ Platform – Knights Landing

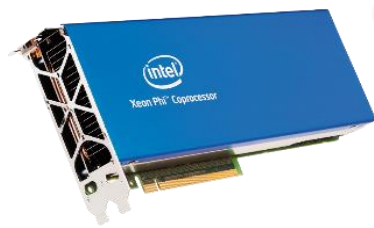


Xeon Phi

Knights Landing (14nm process),

- Optimized for highly parallelized compute intensive workloads
- Common programming model & S/W tools with Xeon processors, enabling efficient app readiness and performance tuning
- up to 72 cores, 490 GB/s stream BW, on-die 2D mesh
- AVX512– 512-bit (8 DP, 16 SP flops) -> >3 TFLOPS
- 36 PCIe lanes

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

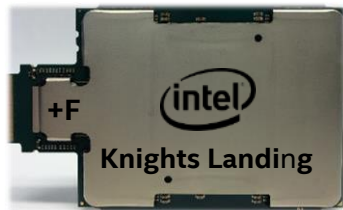
Flexibility
Limitless configurations

Density
3+ KNL with fabric in 1U

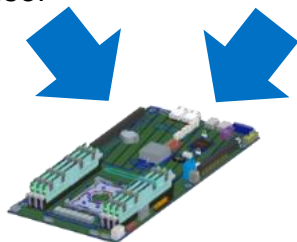
Knights Landing (Host or PCIe)



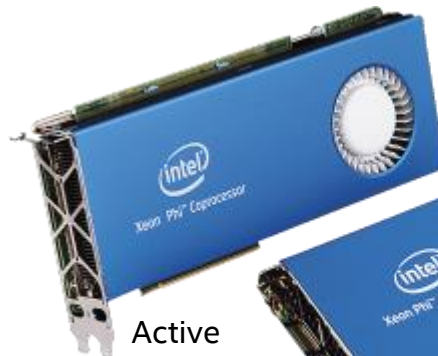
Host Processor



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

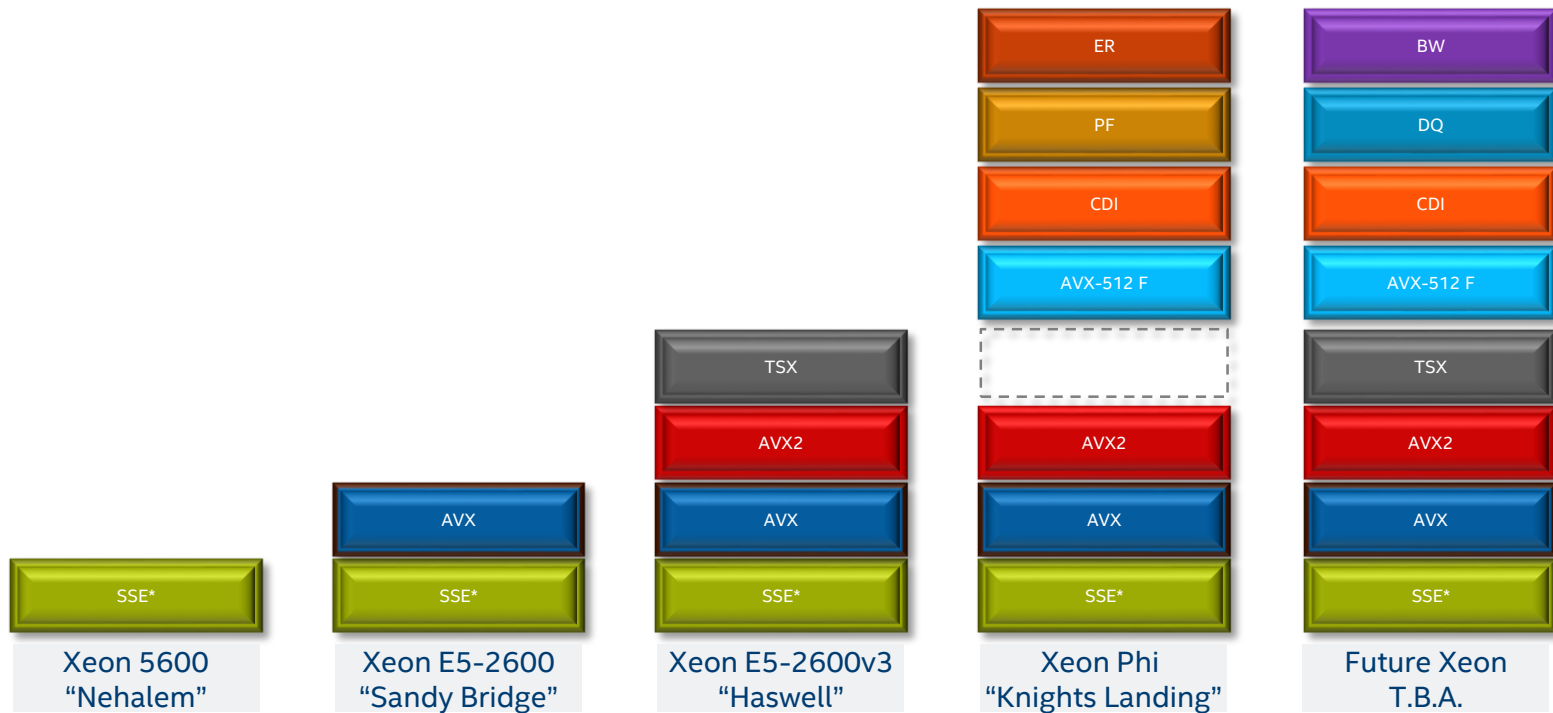


Baseline	Perf/W/\$	Up to 40% better ¹
PCIe	I/O	Fabric
Up to 16GB	Mem Capacity	Up to 384GB
Unique	Manageability	Standard CPU
Up to 4 in 1U	Density	>4 in 1U
<100 PF	Scalability	>100 PF
Partial	Utilization	Full



¹Results 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



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

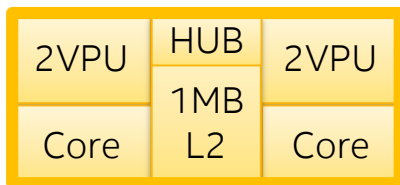
Platform Memory

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

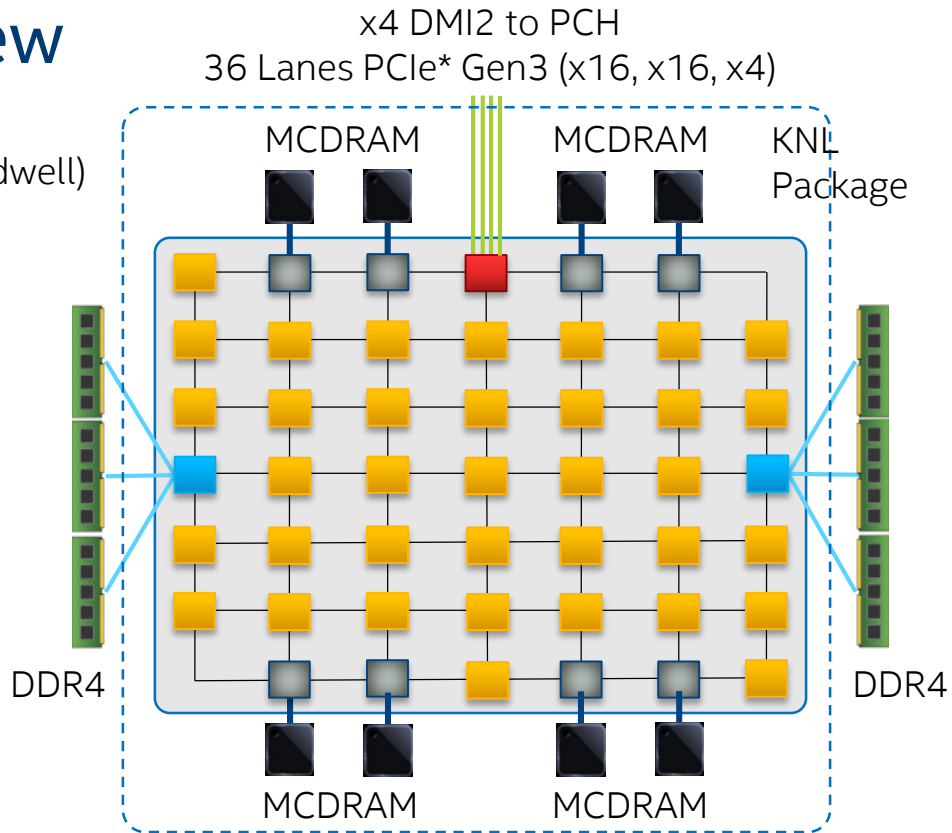
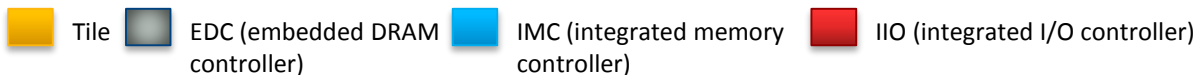
Fixed Bottlenecks

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

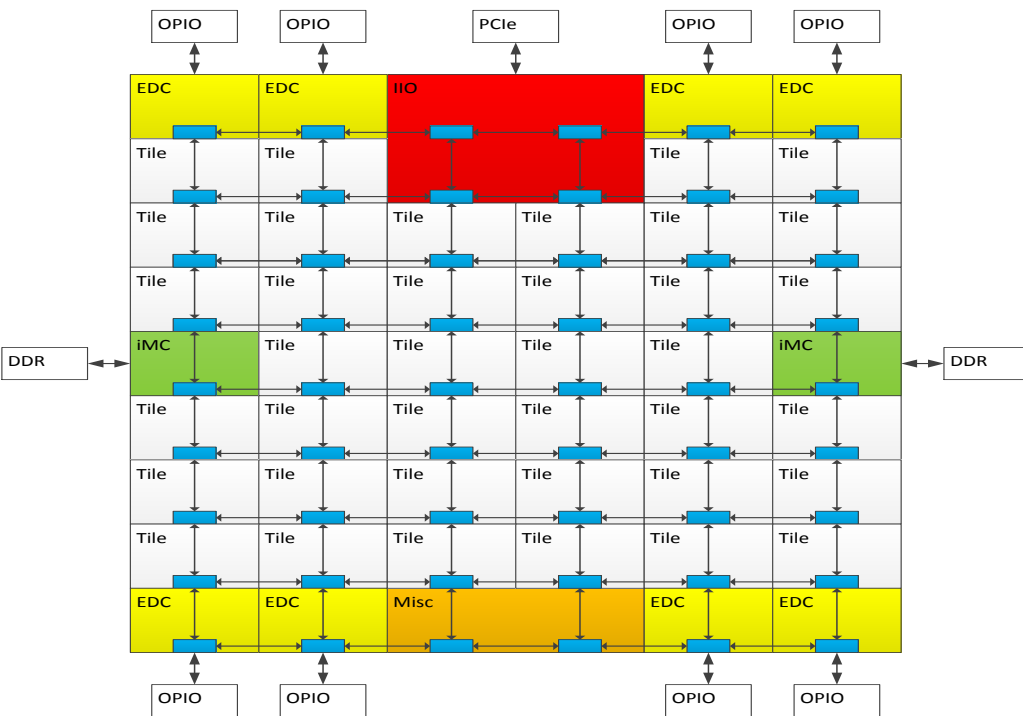
TILE:
(up to
36)



Enhanced Intel® Atom™ cores based on Silvermont™ Microarchitecture



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

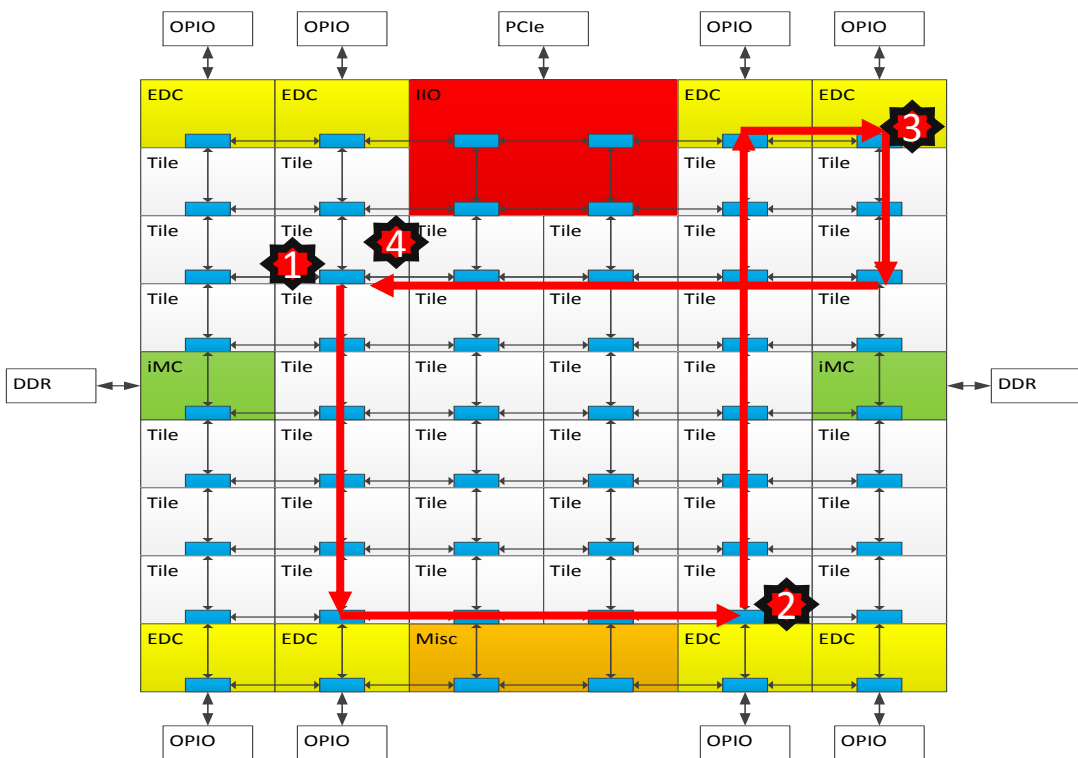
Cache Coherent Interconnect

- 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

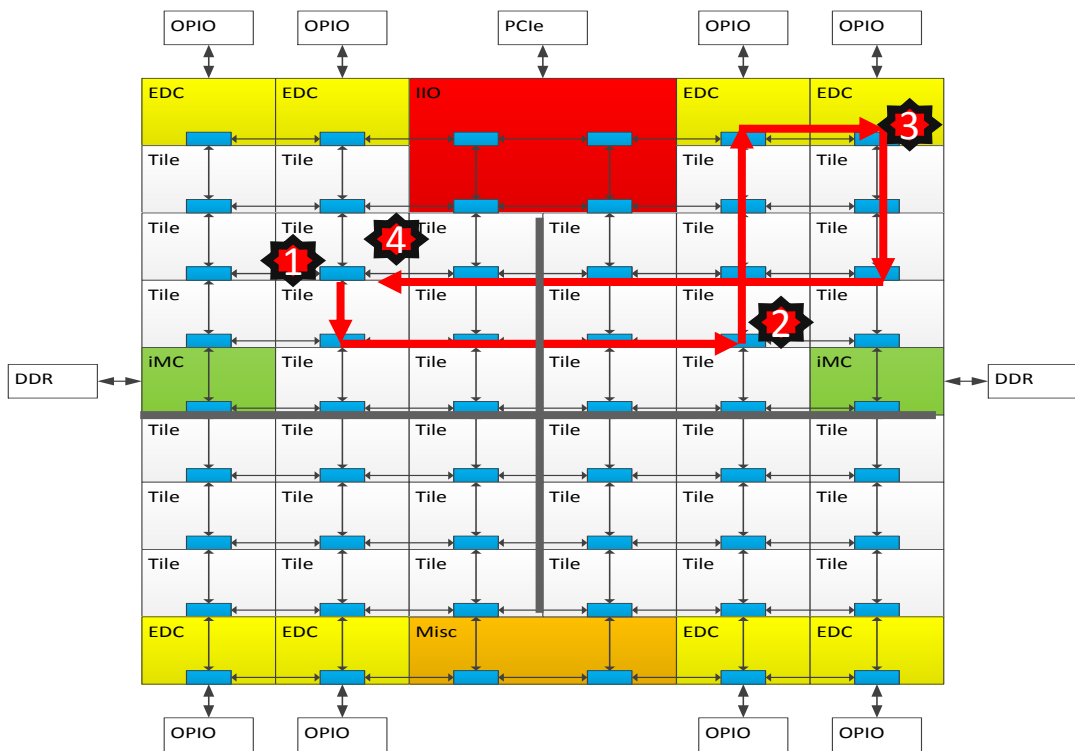
No affinity between Tile, Directory and Memory

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

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



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)

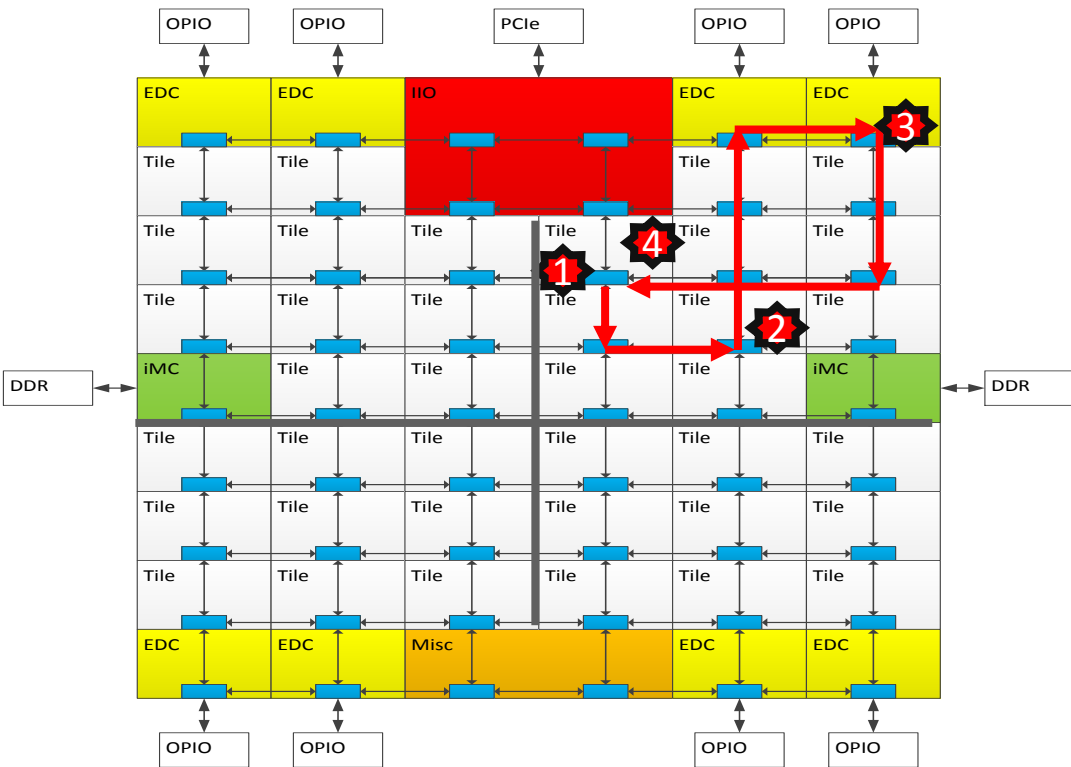
Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

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

KNL Core and VPU

Out-of-order core w/ 4 SMT threads

VPU tightly integrated with core pipeline

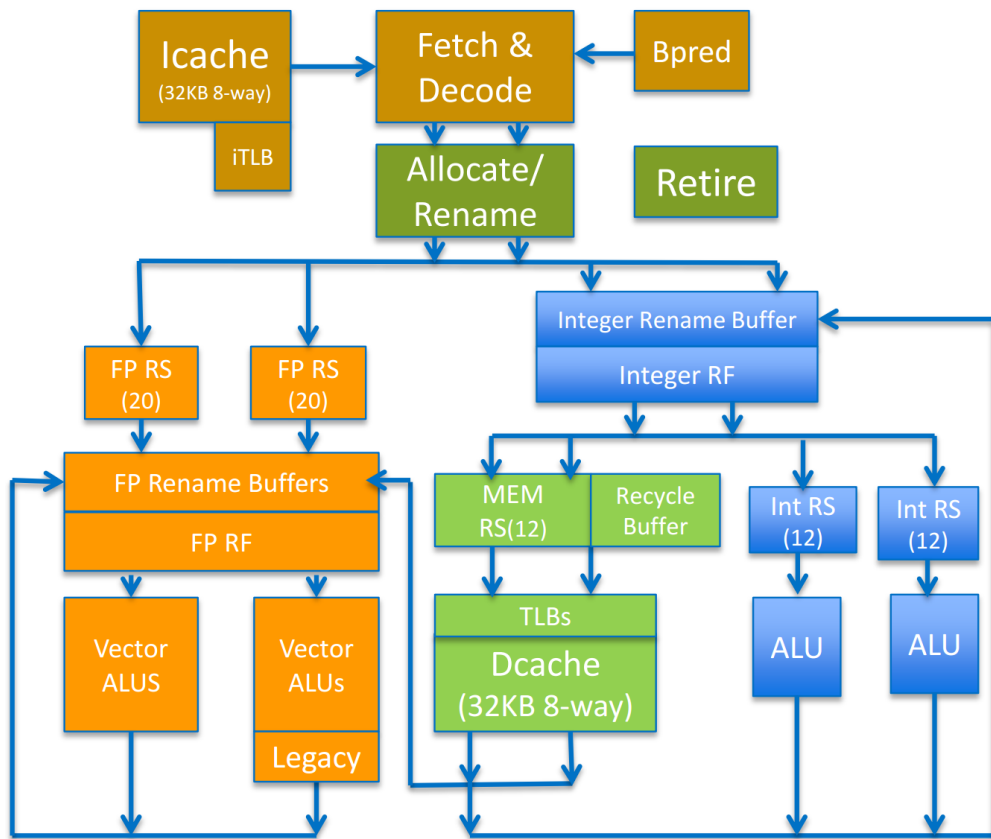
2-wide decode/rename/retire

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

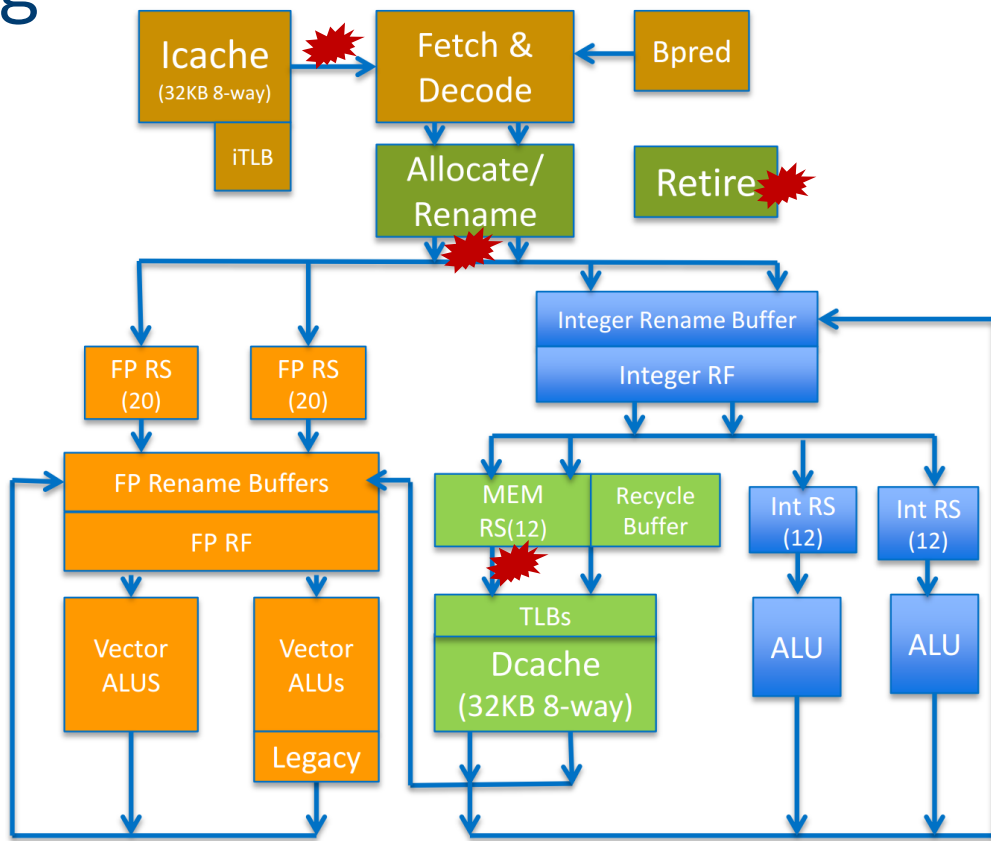
Resources dynamically partitioned

- Re-order Buffer
- Rename buffers
- Reservation station

Resources shared

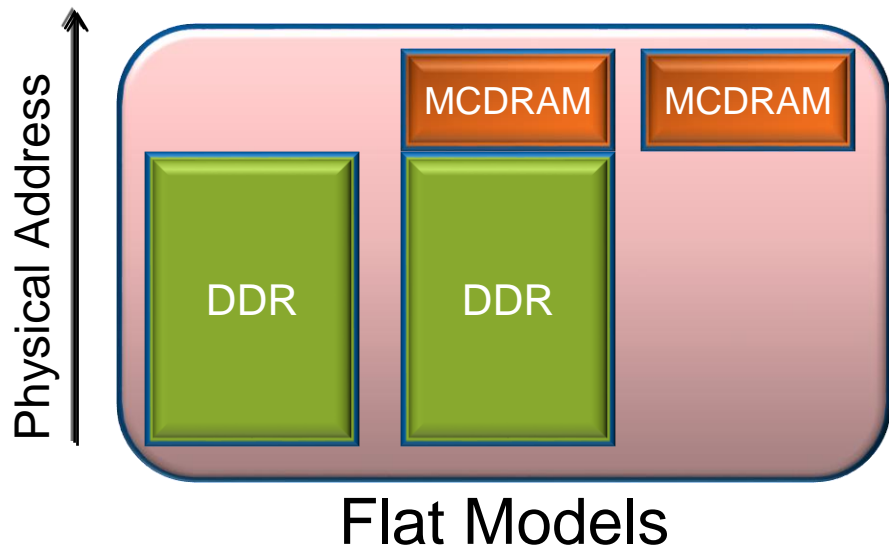
- Caches
- TLB

 Thread selection point

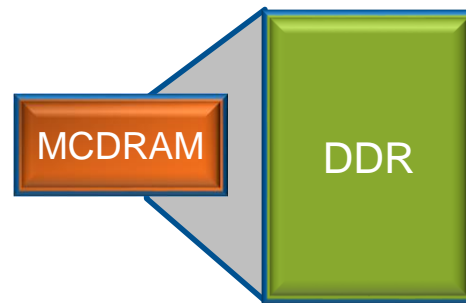


KNL Memory Modes

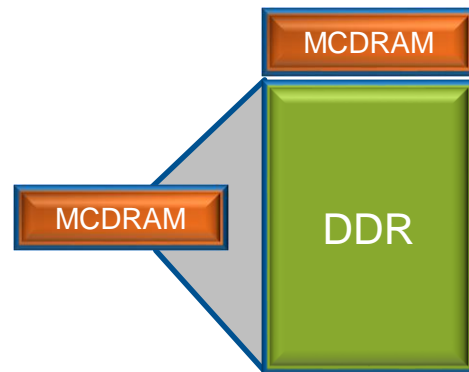
- Mode selected at boot
- MCDRAM-Cache covers all DDR



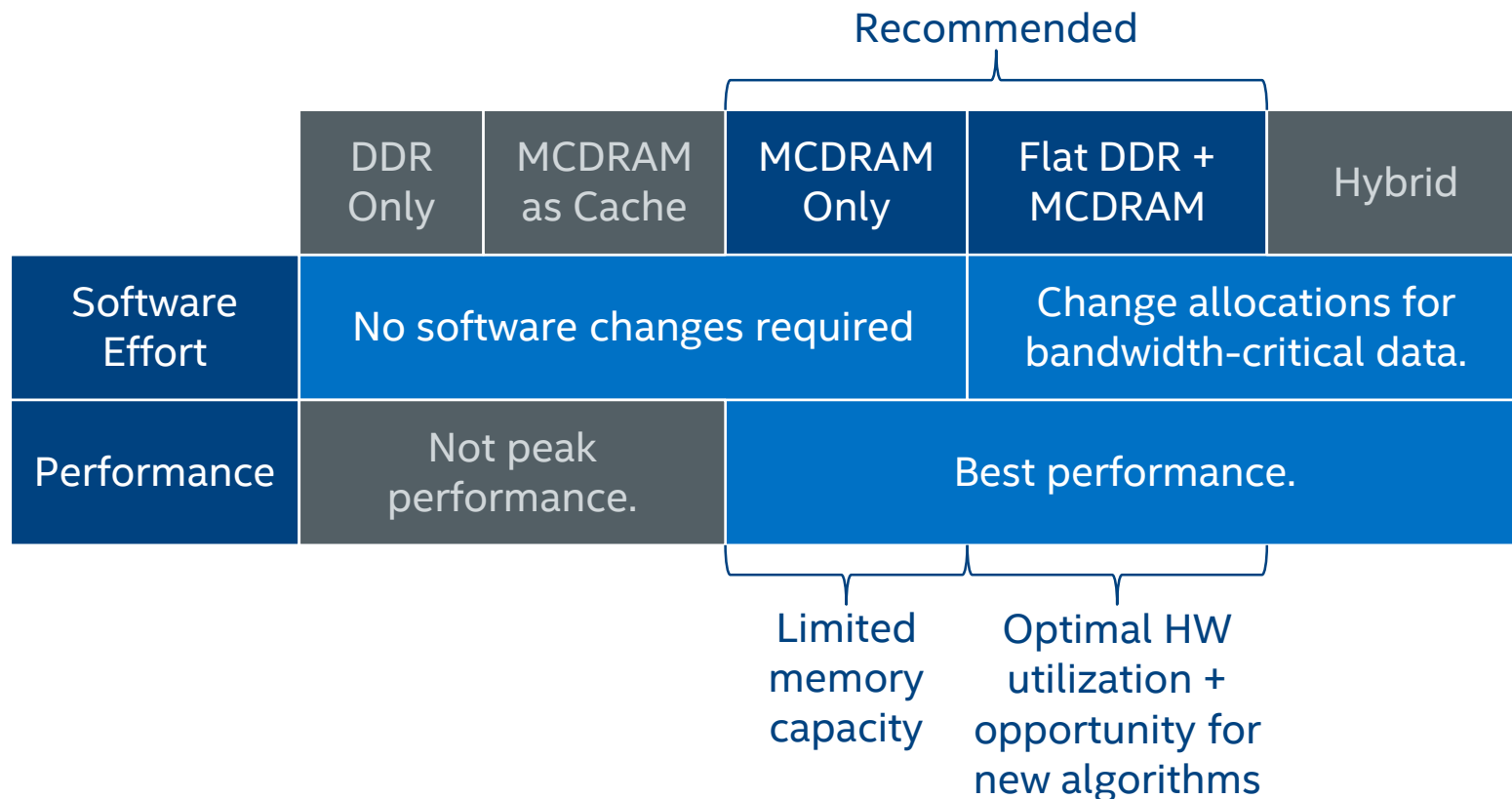
Cache Model



Hybrid Model



MCDRAM: Cache vs Flat Mode



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:

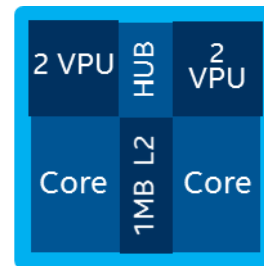
- Avoid cache-line splits; align data structures to 64 bytes.
- Avoid gathers/scatters; replace with shuffles/permutates for known sequences.
- Avoid mixing SSE, AVX and AVX512 instructions.

- 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

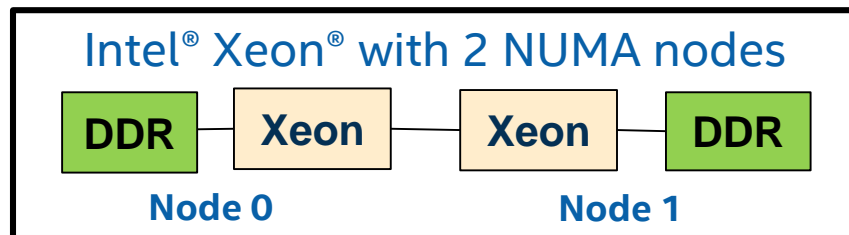
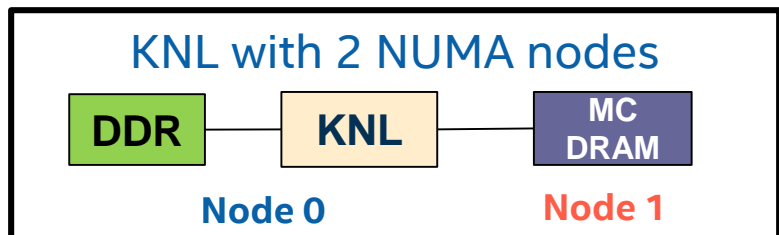
- Recall that KNL cores are grouped into tiles, with two cores sharing an L2.
- Effective capacity depends on locality:
 - 2 cores sharing no data => 2 x 512 KB
 - 2 cores sharing all data => 1 x 1 MB
- Ensuring good locality (e.g. through blocking or nested parallelism) is likely to improve performance.



```
#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 *)hbwmalloc(size);
    if (hbw_str == NULL) {
        perror("hbwmalloc()");
        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.

