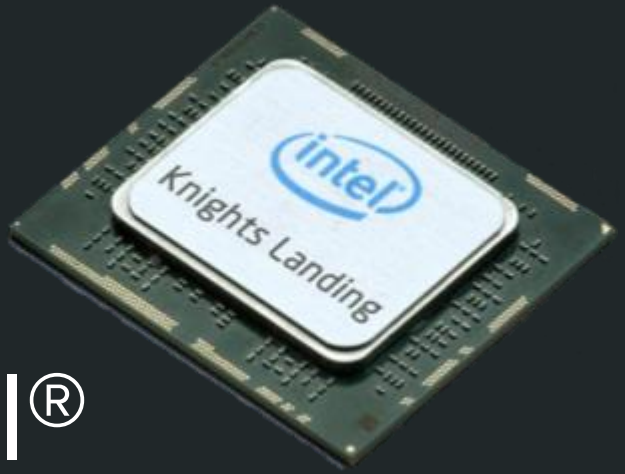
knights Landing Intel®



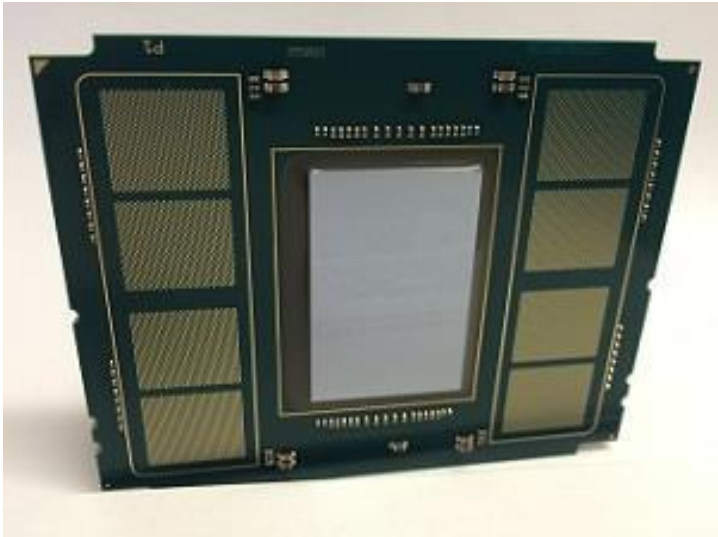
BOULLOUD, CHEN, DE SOUZA et SU

Plan

- KNL Overview
- Pipeline
- Mode memory
- Tools
- Applications

Introduction

- Target for **HPC** and **supercomputing**
- Significant improvement in **scalar** and **vector** performance
- Integration of **Memory on package**
- Integrated **fabric on package**



KNL Overview

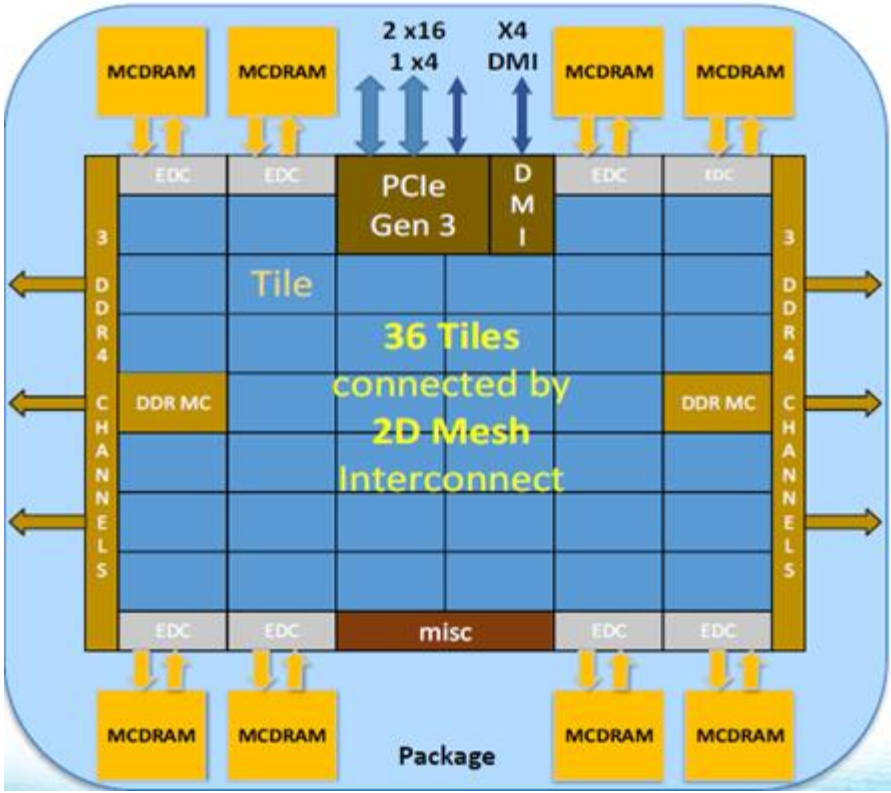
- 36 Tiles interconnected by **2D Mesh**
- Memory: 16 GB on-package **MCDRAM**

  6 channels (**DIMM**) 384 GB **DDR4**
- Streams Triad(GB/S): MCDRAM : 450+

  DDR : 90+
- IO: 36 lanes **PCIe Gen3**

  4 lanes of **DMI** for chipset
- Performance : 3+ TF Double precision

  6+ TF Single precision



Intel Xeon Phi Family

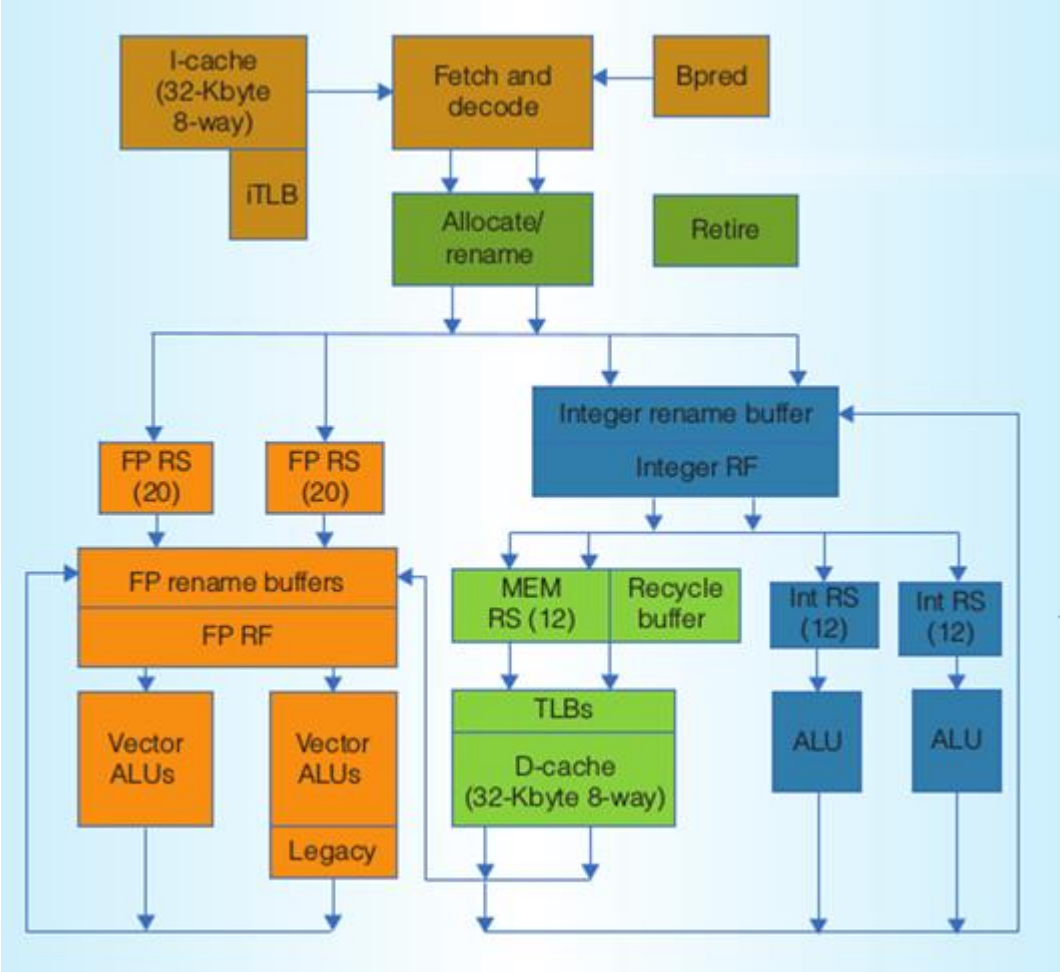
	7210	7210F	7230	7230F	7250	7250F	7290	7290F
Cores	64	64	64	64	68	68	72	72
Frequency	1.3GHz	1.3GHz	1.3GHz	1.3GHz	1.4GHz	1.4GHz	1.5GHz	1.5GHz
Max Frequency	1.5GHz	1.5GHz	1.5GHz	1.5GHz	1.6GHz	1.6GHz	1.7GHz	1.7GHz
Cache	32 MB L2	32 MB L2	32 MB L2	32 MB L2	34 MB L2	34 MB L2	36 MB L2	36 MB L2
TDP	215 W	230 W	215 W	230 W	215 W	230 W	245 W	260 W

Intel Xeon Phi in TOP500

Rank	Country	System	Cores	Rmax	Rpeak	Power
5	USA	Cori Intel Xeon Phi 7250	622336	14014.7	27880.7	3939
6	Japan	Oakforest Intel Xeon Phi 7250	556104	13554.6	24913.5	2718
12	Italy	Marconi Intel Xeon Phi 7250	241808	6223	10833	
18	USA	Theta Intel Xeon Phi 7230	207360	5095.8	8626.2	1087
33	Japan	Camphor2 Intel Xeon Phi 7250	122400	3057.3	5483.5	748.1
106	USA	TX-Green Intel Xeon Phi 7210	42472	1032.8	1725.2	
144	USA	Stampede Intel Xeon Phi 7250	34272	842.9	1535.4	515.5
375	Germany	QPACE3 Intel Xeon Phi 7210	18432	447.1	766.8	77
397	USA	SciPhi Intel Xeon Phi 7230	16896	425.9	702.9	111
456	France	Sequana Intel Xeon Phi 7250	14960	380.5	670.2	103

Pipeline

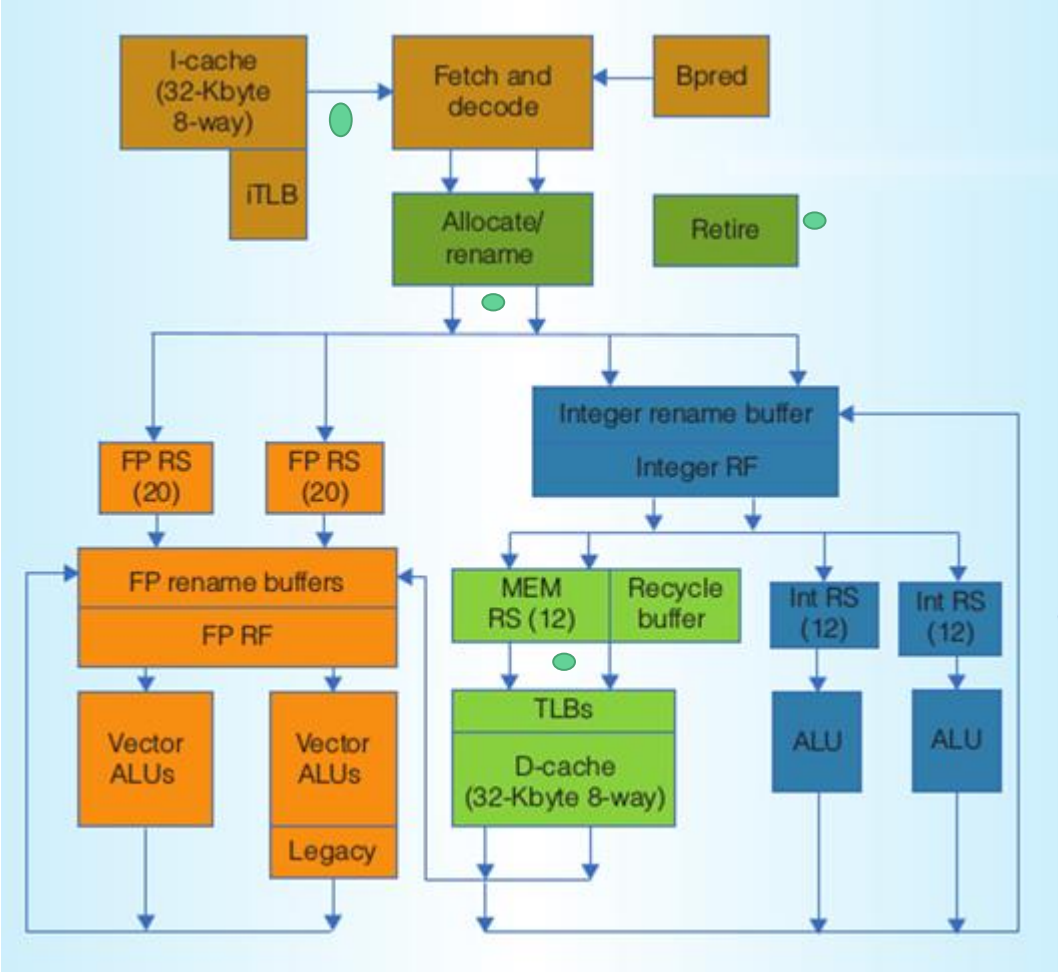
- Architecture Based on Intel® Atom (Silvermont Microarchitecture)
- 2 ALU for Int and 2 ALUs for Vector
- Decode/Rename/Retire 2-wide (16 bytes per cycle [2 instructions])
- 72-entry for Rename and Reorder buffer
- RS with 12 entries for Int and MEM and for FP 20 entries
- TLBs with 64 entries in 1st level and 256 4K, 128 2M, 16 1G pages for 2nd level
- 2x 64B Load & 1 64B Store ports in D-cache
- VPU tightly integrated with core pipeline



FP = Floating Point RS = Reservation Stations

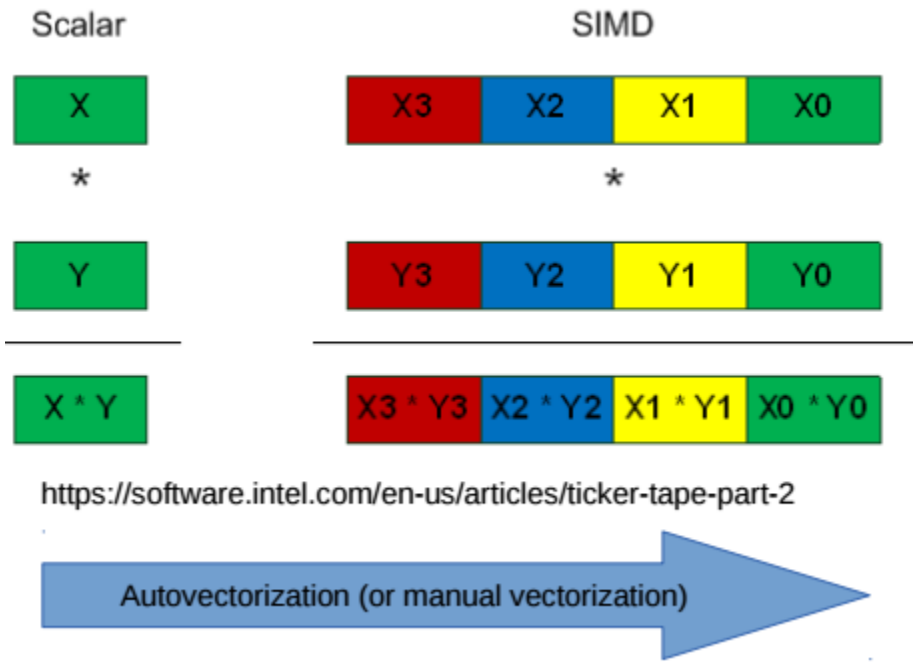
Pipeline

- 4 Threads per core.
- Simultaneous Multithreading
- Core resources divised in Dynamically partitioned for Reorder Buffer, Rename Buffers and RS and shared for caches et TLBs
- Up to 6-wide at execution
- Int and FP RS can enter in Out-of-Order , MEN and Int hold source data
- ~3x higher ST performance over KNC*



* in performance tests SYSmark and MobileMark

KNL : vectorial computation



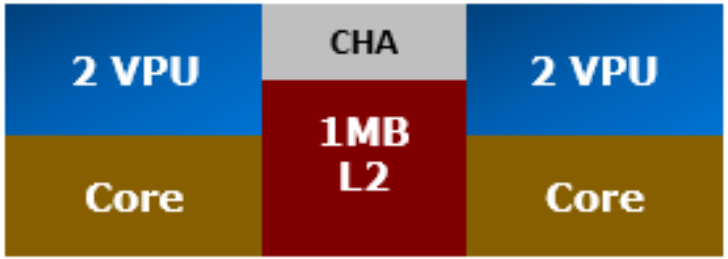
KNL Tile : 2 cores + 2VPU/core + 1MBL2

Core : Based on 2-wide OoO Silvermont™ Microarchitecture, but with many changes for HPC. 4 thread/core. Deeper OoO. Higher bandwidth. Larger TLBs.

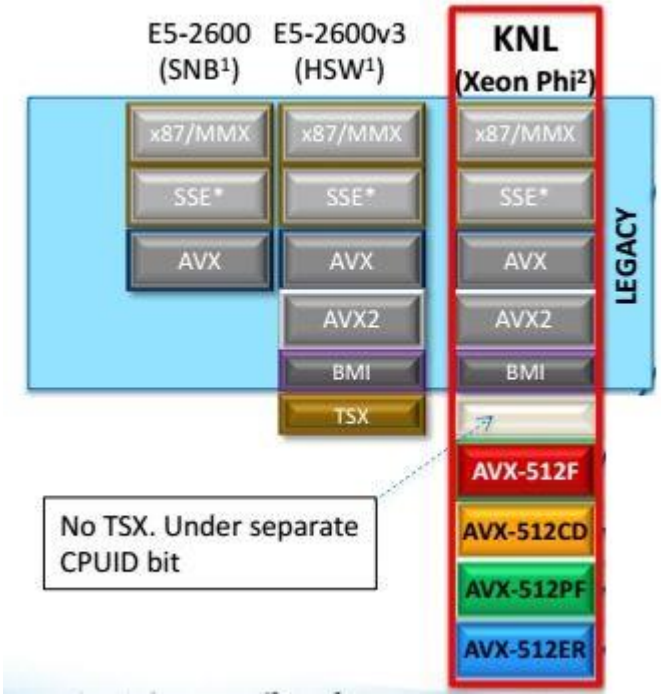
2 VPU : 2x AVX512 units. 32SP/16DP per unit. X87, SSE, AVX1, AVX2 and EMU

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

CHA : Caching/Home Agent



KNL ISA



- 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 registres, & 8 masks registres
  - Gather/Scatter
- Conflict Detection : Improves vectorization
- Prefetch : Gather and Scatter prefetch
- Exponential and Reciprocal instructions

KNL AVX-512 CD

```
for(i=0; i<16; i++) { A[B[i]]++;}
```

```
index = vload &B[i]           // Load 16 B[i]
old_val = vgather A, index    // Grab A[B[i]]
new_val = vadd old_val, +1.0  // Compute new values
vscatter A, index, new_val    // Update A[B[i]]
```

Code is wrong if any values within B[i] are duplicated

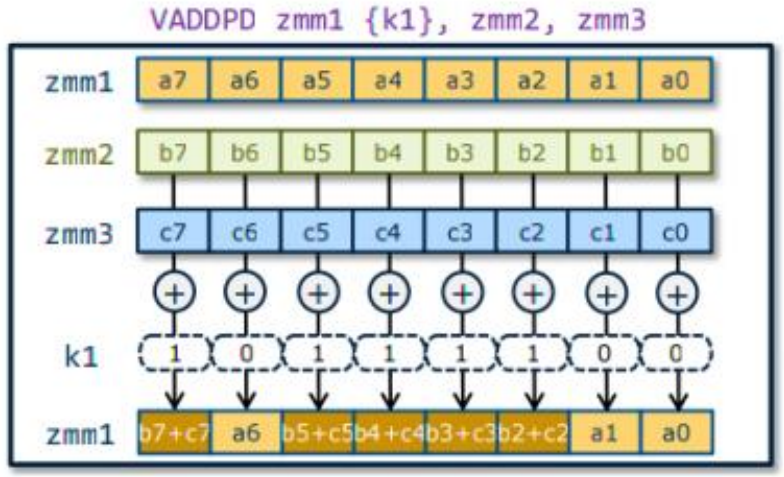
```
index = vload &B[i]                          // Load 16 B[i]
pending_elem = 0xFFFF;                       // all still remaining
do {
   curr_elem = get_conflict_free_subset(index, pending_elem)
   old_val = vgather {curr_elem} A, index    // Grab A[B[i]]
   new_val = vadd old_val, +1.0              // Compute new values
   vscatter A {curr_elem}, index, new_val    // Update A[B[i]]
   pending_elem = pending_elem ^ curr_elem   // remove done idx
} while (pending_elem)
```

AVX-512 Conflict Detection

- VPCONFLICT{D,Q} zmm1{k1}, zmm2/mem
- VPBROADCASTM{W2D,B2Q} zmm1, k2
- VPTESTNM{D,Q} k2{k1}, zmm2, zmm3/mem
- VPLZCNT{D,Q} zmm1 {k1}, zmm2/mem

AVX-512 Mask Registres

- Suppress individual elements read from memory
  - Hence not signaling any memory fault
- Avoid actual independent operations within an instruction happening
  - And hence not signaling any FP faut
- Avoid the individual destination elements being updated,



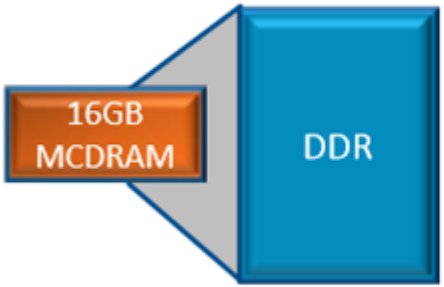
https://gcc.gnu.org/wiki/cauldron2014?action=AttachFile&do=get&target=Cauldron14_AVX-512_Vector_ISA_Kirill_Yukhin_20140711.pdf

AVX-512 has 8 mask registers (64-bits each)

Mode memory

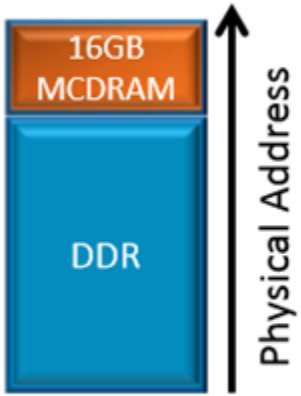
Three Modes. Selected at boot

Cache Mode



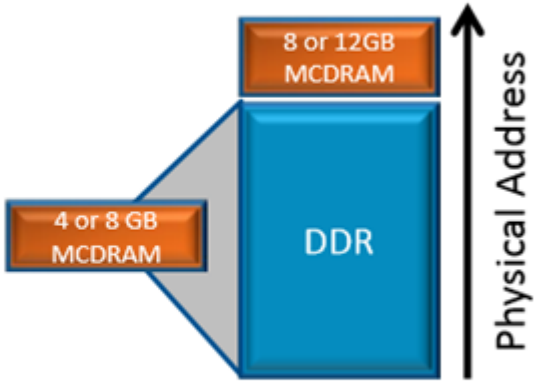
- SW-Transparent, Mem-side cache
- Direct mapped. 64B lines.
- Tags part of line
- Covers whole DDR range

Flat Mode



- MCDRAM as regular memory
- SW-Managed
- Same address space

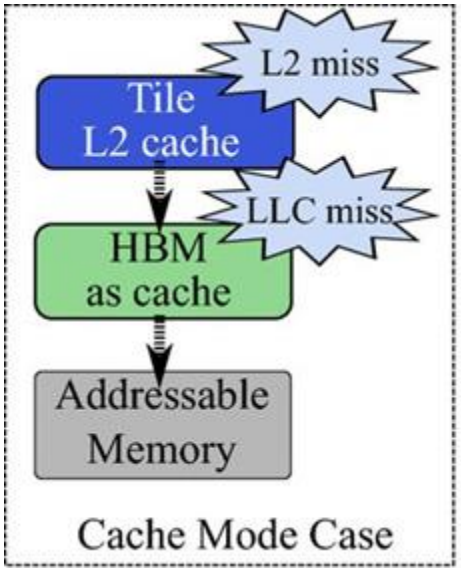
Hybrid Mode

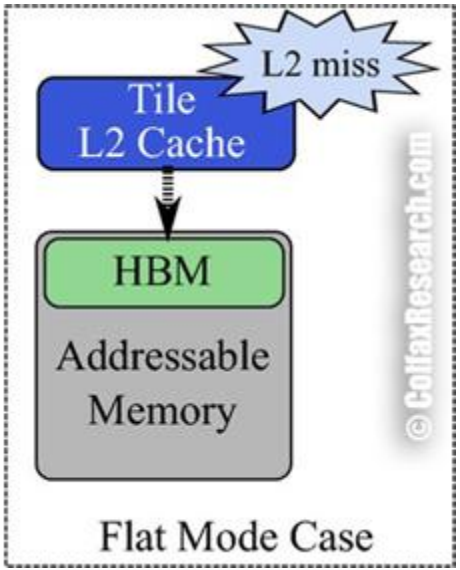


- Part cache, Part memory
- 25% or 50% cache
- Benefits of both

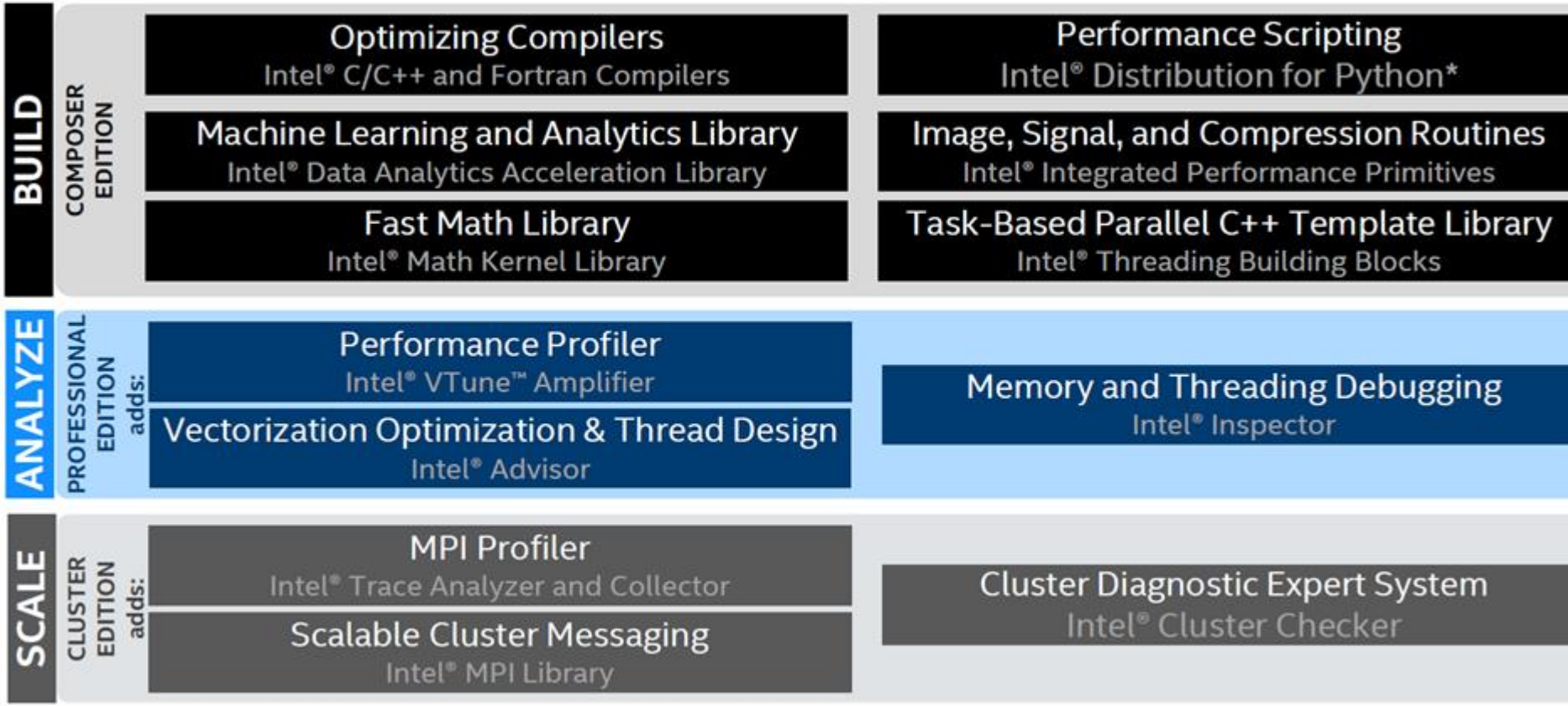
Advantages

- *Cache mode*
  - No work required
  - but lower performance (misses in HBM as cache)
- *Flat mode*
  - May offer better performance
  - but requires modifications of the code
- *Hybrid mode*
  - Benefit of both *Flat mode* and *Cache mode*



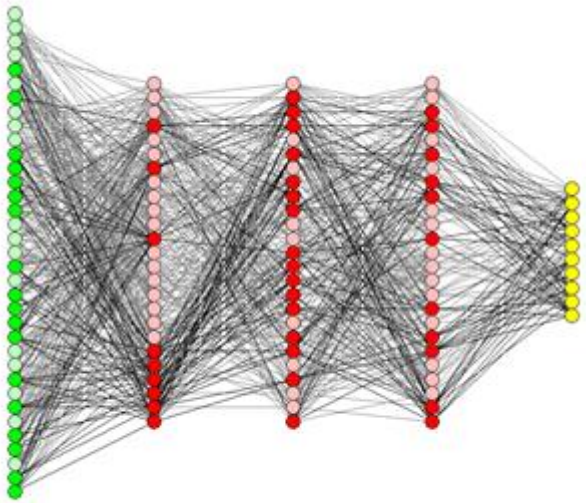


Tools

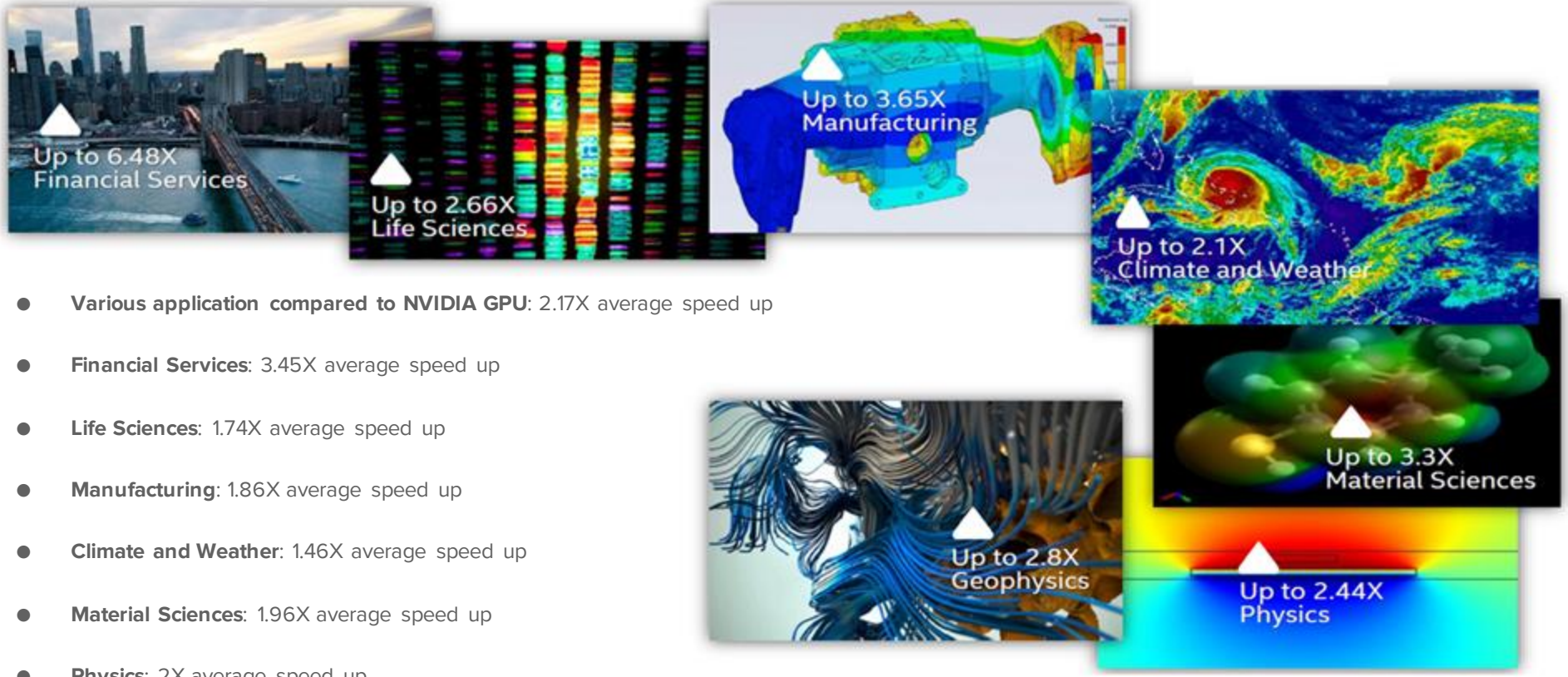

BUILD
COMPOSER EDITION
Optimizing Compilers
Intel® C/C++ and Fortran Compilers
Performance Scripting
Intel® Distribution for Python*
Machine Learning and Analytics Library
Intel® Data Analytics Acceleration Library
Image, Signal, and Compression Routines
Intel® Integrated Performance Primitives
Fast Math Library
Intel® Math Kernel Library
Task-Based Parallel C++ Template Library
Intel® Threading Building Blocks
ANALYZE
PROFESSIONAL EDITION adds:
Performance Profiler
Intel® VTune™ Amplifier
Vectorization Optimization & Thread Design
Intel® Advisor
Memory and Threading Debugging
Intel® Inspector
SCALE
CLUSTER EDITION adds:
MPI Profiler
Intel® Trace Analyzer and Collector
Scalable Cluster Messaging
Intel® MPI Library
Cluster Diagnostic Expert System
Intel® Cluster Checker

Typical Machine using KNL

- High Performance Computer (HPC)
- Machine Learning



Application



- **Various application compared to NVIDIA GPU**: 2.17X average speed up
- **Financial Services**: 3.45X average speed up
- **Life Sciences**: 1.74X average speed up
- **Manufacturing**: 1.86X average speed up
- **Climate and Weather**: 1.46X average speed up
- **Material Sciences**: 1.96X average speed up
- **Physics**: 2X average speed up
- **Geophysics**: 2.17X average speed up

References

http://www.enterprisetech.com/2016/06/22/intel-launches-knights-landing-phi-machine-learning-hpc/ June 22, 2016 byTiffany Trader

https://software.intel.com/en-us/articles/migrating-applications-from-knights-corner-to-knights-landing-self-boot-platforms May 11, 2016 by Michael Greenfield, Intel

https://www.hpcwire.com/2016/06/18/knights-landing-not/ Jun 18, 2016 byJames Reinders, Intel

https://colfaxresearch.com/knl-avx512/ May 11, 2016 in Publications

https://colfaxresearch.com/knl-mcdram/ May 11, 2016 in Publications

https://www.top500.org/lists/

Knights Corner: Your Path to Knights Landing, Sept 17, 2014 by James Reinders, Intel

Optimizing for Intel's Knights Landing and Other HPC Architectures, Jul 21, 2016 byHal Finkel

Knights Landing (KNL): 2nd Generation Intel® Xeon Phi™ Processor, byAvinash Sodani, Intel

The microarchitecture of Intel, AMD and VIA CPU, Des 12, 2016 by Agner Fog

Knights Landing Intel® Xeon Phi™ CPU: Path to Parallelism with General Purpose Programming, by Avinash Sodani, Intel