



Intel® Xeon Phi™ basics and architecture

September 22nd-23rd 2015
University of Copenhagen,
Denmark

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

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.

Copyright ©, Intel Corporation. All rights reserved. Intel, the Intel logo, Xeon, Xeon Phi, Core, VTune, and Cilk are trademarks of Intel Corporation in the U.S. and other countries.

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

SOFTWARE AND SERVICES

Intel Technologies for HPC

Processors

Intel® Xeon® Processor



Coprocessor

Intel® Many Integrated Core



Network & Fabric



I/O & Storage



Software & Services



SOFTWARE AND SERVICES

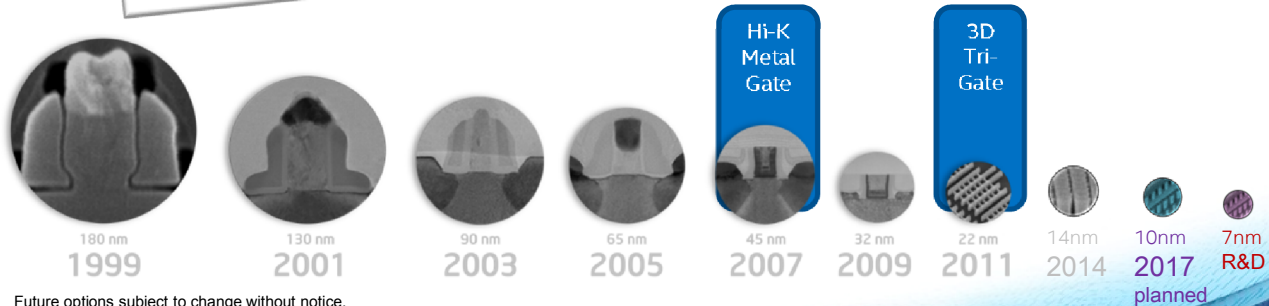
Transforming the economics of HPC



Executing to Moore's Law

Predictable Silicon Track Record – well and alive at Intel.

Enabling new devices with higher performance and functionality while controlling power, cost, and size

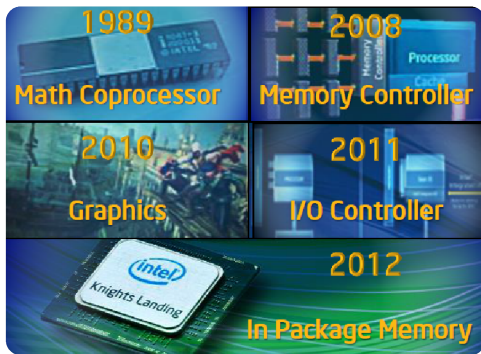


Future options subject to change without notice.

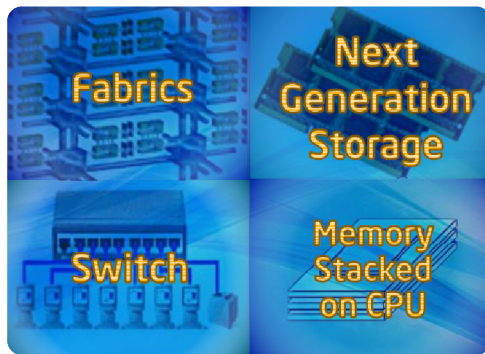
SOFTWARE AND SERVICES

Driving innovation and integration

Enabled by Leading Edge Process Technologies



Integrated Today



Coming in the near Future

SOFTWARE AND SERVICES

Intel® Xeon Phi™ Coprocessor Product Family

Based on Intel® Many Integrated Core (MIC) Architecture



2013 Knights Corner

Intel® Xeon Phi™ x100 product family

- 22 nm process
- Coprocessor
- Over 1 TF DP Peak
- Up to 61 Cores
- Up to 16GB GDDR5



2016 Knights Landing

The processor version of the next generation Intel Xeon Phi product family

- 14 nm process
- Processor & Coprocessor
- Over 3 TF DP Peak
- Up to 72 Cores
- On Package High-Bandwidth Memory
- 3x single-thread performance
- Out-of-order core
- Integrated Intel® Omni-Path



FUTURE

Knights Hill

Next generation of Intel® MIC Architecture Product Line

- 10 nm process
- 2nd Generation Integrated Intel® Omni-Path
- In planning –



intel
inside™
XEON PHI™

SOFTWARE AND SERVICES

Per Intel's announced products or planning process for future products

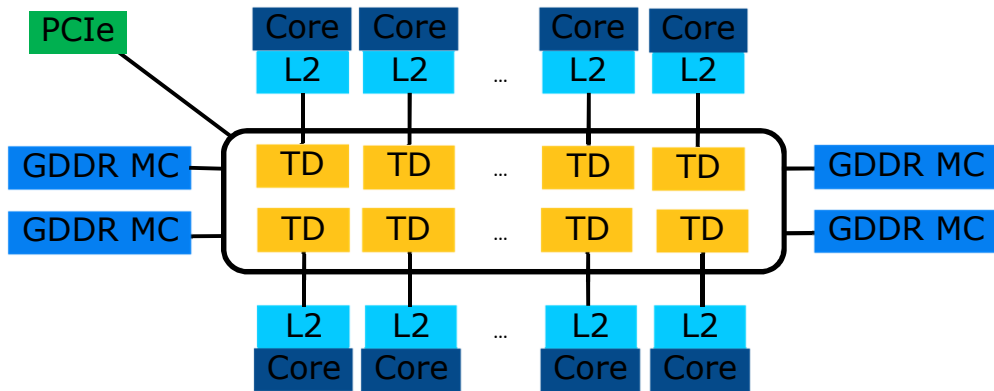


Hardware architecture

Intel® Xeon Phi™, Knights Corner (KNC)

SOFTWARE AND SERVICES

Architectural overview

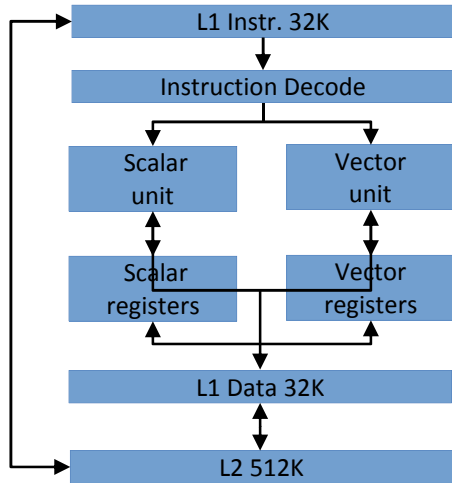


- Up to 61 cores
- 8-16 GB GDDR5 memory (ECC)
- PCIe Gen2 (client) x16 per dir.

SOFTWARE AND SERVICES

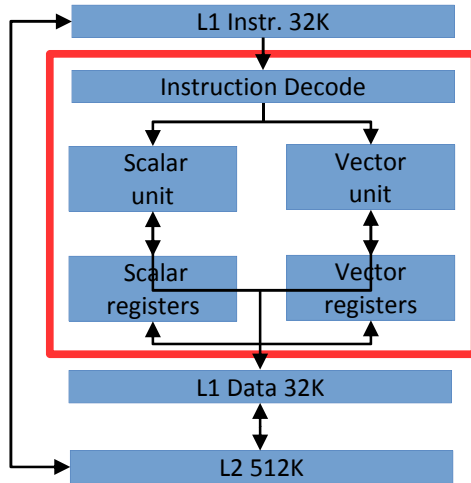
- Hardware cache coherency
- 8 memory controllers
 - 16 GDDR5 channels
 - Up to 5.5GT/s

Intel® Xeon Phi™ Core (KNC)



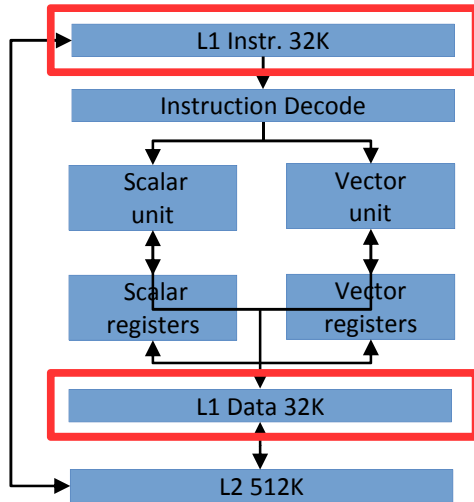
- Pentium (P54C) scalar instruction set (X87)
 - In order-operation
- 64bit addressing
- 512bit vector unit
- 4 HW threads/core
- Two pipelines:
 - Scalar
 - Vector/Scalar

Intel® Xeon Phi™ Core (KNC)



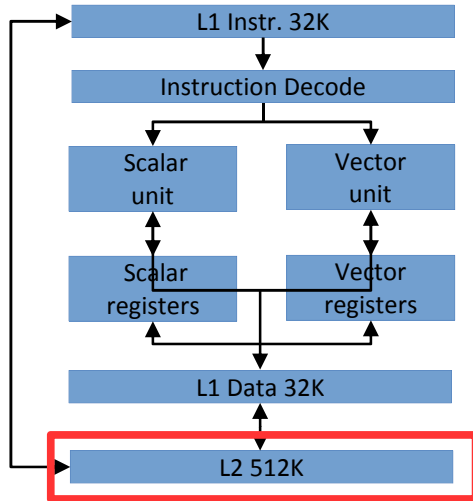
- 2 issue (1 scalar/1 vector)
- 2 cycle decoder: no-back to back cycle issue from *the same* context (thread)
 - At least two HW contexts (thread/proc) to fully utilize the core
- Most vector instructions have 4 clock latency

Intel® Xeon Phi™ Core (KNC)



- L1 caches
 - 32K I-cache per core
 - 32K D-cache per core
 - 8 way associative
- 3 cycle access latency
- Up to 8 outstanding requests
- 64byte cache line
- Fully coherent (MESI)

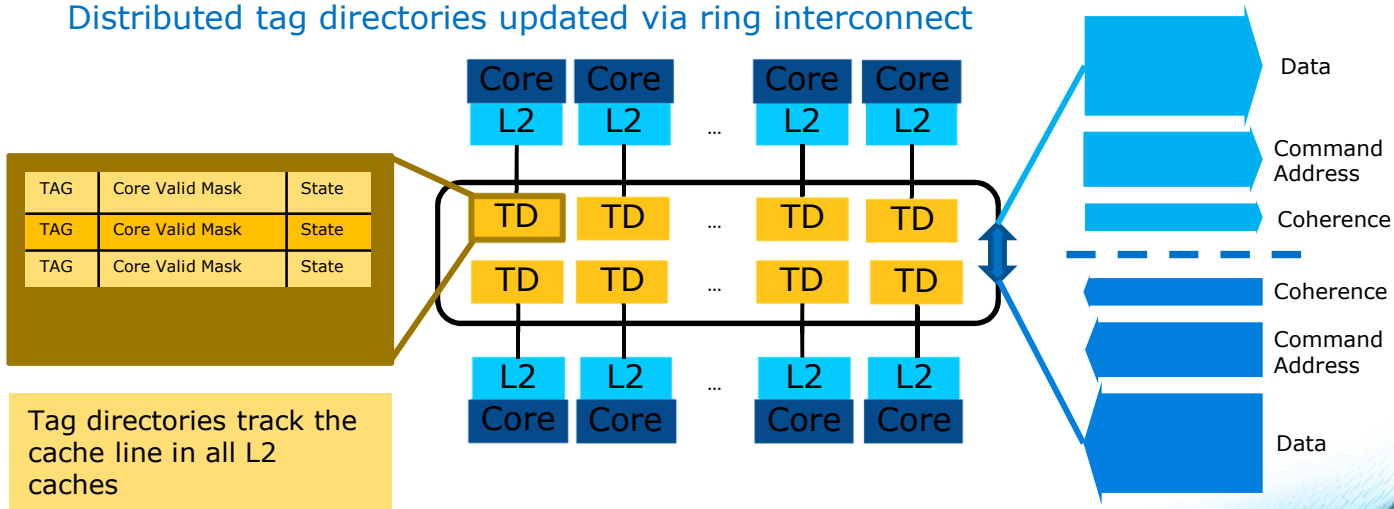
Intel® Xeon Phi™ Core (KNC)



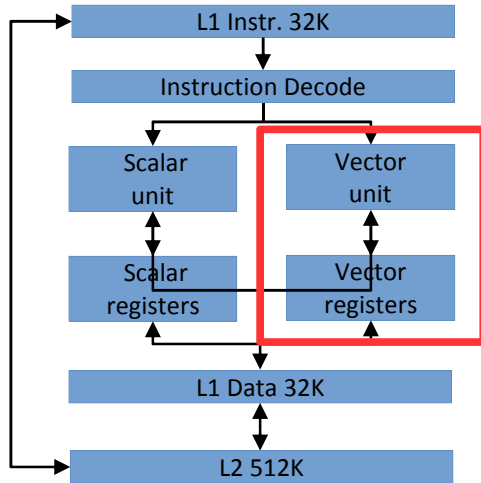
- L2 cache
 - 512K Unified per core
- 8 way associative
- 11 cycle raw access latency
- Up to 32 outstanding requests
- Streaming HW prefetcher
- Fully coherent (MESI)

Cache coherency

Distributed tag directories updated via ring interconnect



Intel® Xeon Phi™ Core (KNC)



- Vector unit width 512 bits
- 32 512-bit vector registers per context
 - 16 floats or 8 doubles
 - 8 vector mask registers for per lane conditional operations
- ALU support for
 - int32/float32 operations, float64 arithmetic, int64 logic ops
 - Ternary ops including Fused-Multiply-Add
 - Broadcast/swizzle support, float16 up-convert
- Most ops: 4-cycle latency 1-cycle throughput
 - Matches 4-cycle round robin of integer unit
- Mostly IEEE 754 2008 compliant

Architectural comparison

Intel® Xeon®

- General instruction streams
 - High single-thread perf.
 - High memory capacity
- Core/memory aggr. via sockets and nodes
- Instruction set extensions
 - SIMD e.g., Intel® AVX/AVX2
 - Virtualization, AES, etc.

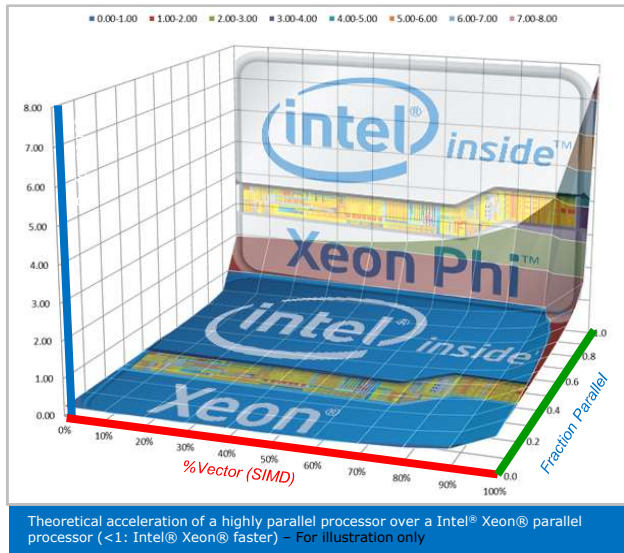
Intel® Xeon Phi™

- General instruction streams
 - Highly parallel workloads
 - High memory bandwidth
- Up to 61 cores/die, aggr. via PCIe and nodes
- SIMD (512-bit registers)
 - Gather/scatter, FMA, masked instructions

Architectural comparison (in numbers)

	Intel® Xeon® E5-2670 v3	Intel® Xeon Phi™ 7120
Cores	12	61
Clock rate (Ghz)	2.3 (3.1 with turbo)	1.24 (1.3 with turbo)
Memory (GB)	32 (typical), 768 (maximum)	16
Cache (L1,L2,L3)	32kB, 256kB, 30Mb (shared)	32kB, 512kB, -
Peak perf (DP Gflops)	441.6	1210.24
Memory BW (GB/sec)	68	352

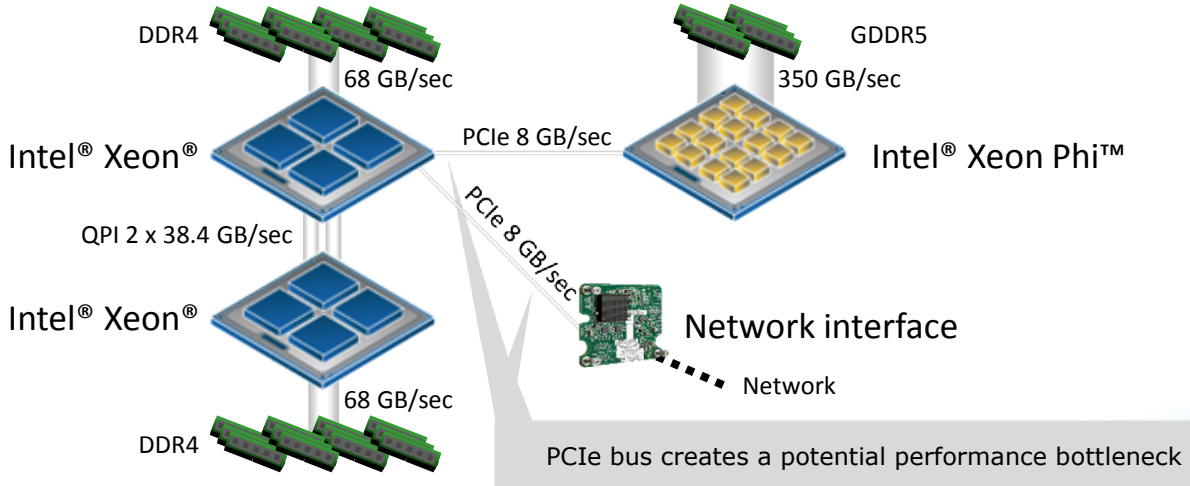
Highly Parallel Applications



- Efficient...
 - Vectorization
 - Threading
 - Parallel execution
- ...drives higher performance for *suitable* scalable applications

SOFTWARE AND SERVICES

Coprocessor system topology today



SOFTWARE AND SERVICES

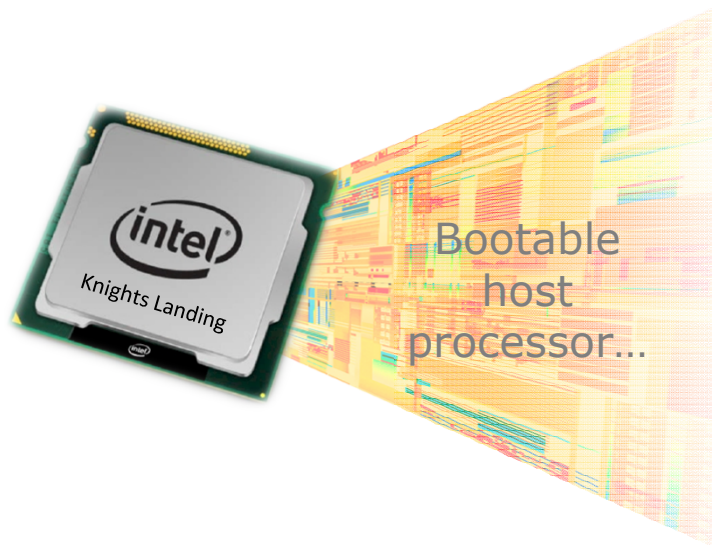


Outlook on future hardware architecture

Intel® Xeon Phi™, Knights Landing (KNL)

SOFTWARE AND SERVICES

Future Intel® Xeon Phi™ Processor: Knights Landing



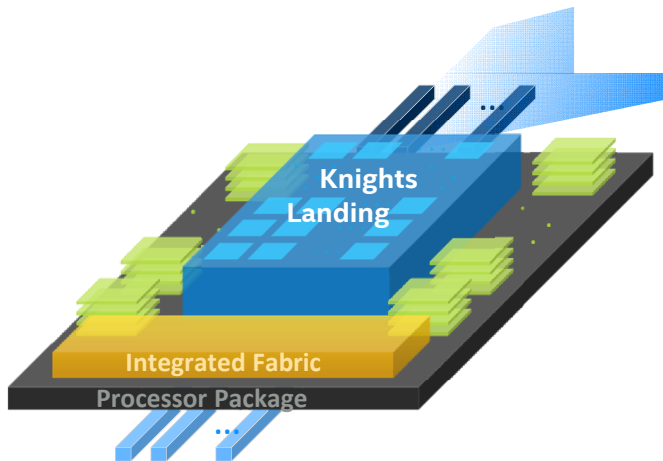
Bootable
host
processor...

- **Unconstrained** by PCIe* offload bottlenecks
- Up to **72 cores** (Silvermont based)
 - 3x single thread performance over KNC
- Excellent compute density and power efficiency
 - >3 Tflops peak DP performance
- Integrated **high bandwidth memory** and **fabric**

SOFTWARE AND SERVICES

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel® Xeon Phi™ Core (KNL)

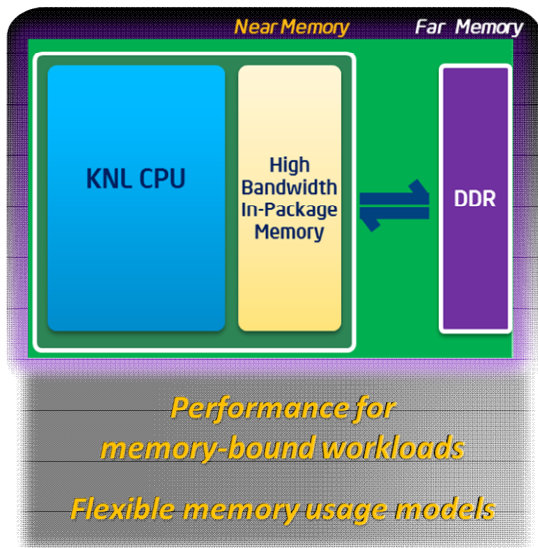


- Silvermont based core
 - **Out-of-order** architecture
 - Binary compatible with Intel® Xeon® (AVX-512)
- 4 HW threads/core
- AVX-512 vector instructions
 - Prefetch instructions
 - Conflict Detection instructions
 - Exponential and Reciprocal instructions
- Advanced branch prediction in hardware
- 2D mesh architecture

SOFTWARE AND SERVICES

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

KNL integrated on-package memory



- Up to 5x STREAM bandwidth over DDR4 (>400GB/sec)
- Cache model
 - Hardware automatically manages the integrated on-package memory as cache
- Flat model
 - Programmer manages the integrated on-package memory and external DDR for peak performance
- Hybrid model

SOFTWARE AND SERVICES

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 CPU and memory – it is not to scale, and is not representative of actual component layout.



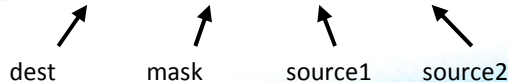
Intel® IMCI (Intel® Initial Many Core Instructions)

Intel® Xeon Phi™, Knights Corner (KNC)

SOFTWARE AND SERVICES

Vector Instruction Format

- 3 operand form with explicit destination register
`instruction destination, source1, source2`
 - → Source registers are not destroyed
 - → Very compact code
- (Most) MIC instructions can be masked
`instruction destination {mask}, source1, source2`
 - → Result of masking is non-destructive, i.e. destination is preserved
- Example: `vaddps zmm1{k1}, zmm2, zmm3`



Examples of Intel® IMCI

- Ternary Operands

- `vop zmm1, zmm2, zmm3` , $zmm1 = zmm2:::vop:::zmm3$
- `vop zmm1, zmm2, [ptr]` , $zmm1 = zmm2:::vop:::MEM[ptr]$

- Fused operation Multiply-Add, Multiply-subtract

- `vfmadd132ps zmm1, zmm2, zmm3` , $zmm1 = zmm1 * zmm3 + zmm2$
- `vfmadd213ps zmm1, zmm2, zmm3` , $zmm1 = zmm2 * zmm1 + zmm3$
- `vfmadd231ps zmm1, zmm2, zmm3` , $zmm1 = zmm2 * zmm3 + zmm1$
- Standard IEEE 754-2008R 0.5 ulps not 1 ulps as two operations

- Prefetching

- Memory Prefetching minimize the likelihood of L1, L2 cache misses
- Intel® Xeon Phi Coprocessor has a hardware prefetcher
- L1 prefetch: `vprefetch1 ptr, hint`
- L2 prefetch: `vprefetch2 ptr, hint`

Extended Math Unit (EMU)

- **Single precision** transcendental functions via minmax quadratic polynomial approximation
- Elementary functions
 - Reciprocal: $1/x$
 - Reciprocal square root: $1/\text{sqrt}(x)$
 - Logarithm: $\text{log}_2(x)$
 - Exponential: $\text{exp}_2(x)$
- Derived functions
 - Power: $x^y = \text{exp}_2(y * \text{log}_2(x))$
 - Square root: $\text{sqrt}(x) = x * 1/\text{sqrt}(x)$
 - Division $\text{div}(x/y) = x * 1/y$
 - Natural logarithm
 $\ln(x) = \text{log}_2(x) * 1/\text{log}_2(e)$

Function	Latency	Throughput
exp2()	8	2
log2()	4	1
rcp()	4	1
rsqrt()	4	1
sqrt()	8	2
pow()	16	4
div()	8	2
ln()	8	2



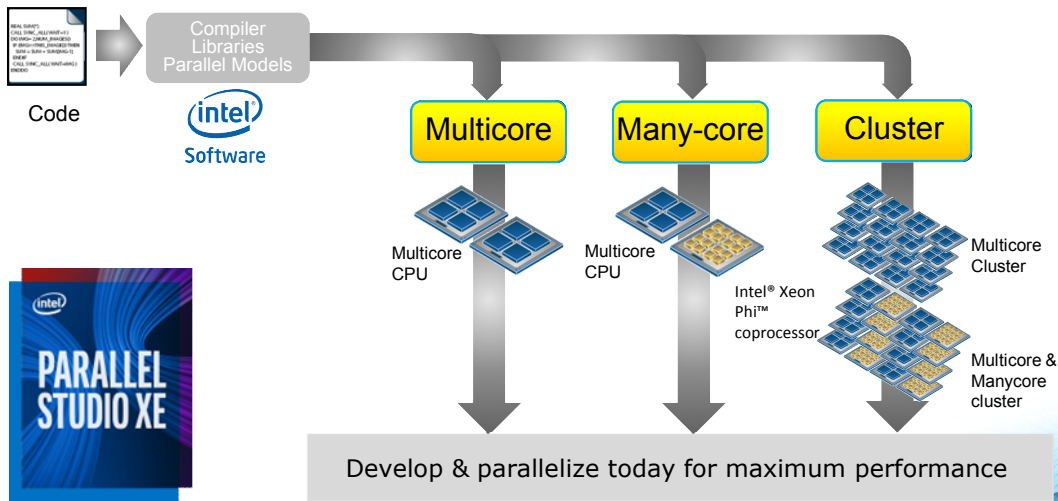
System architecture

Intel® Xeon Phi™, Knights Corner (KNC)

SOFTWARE AND SERVICES

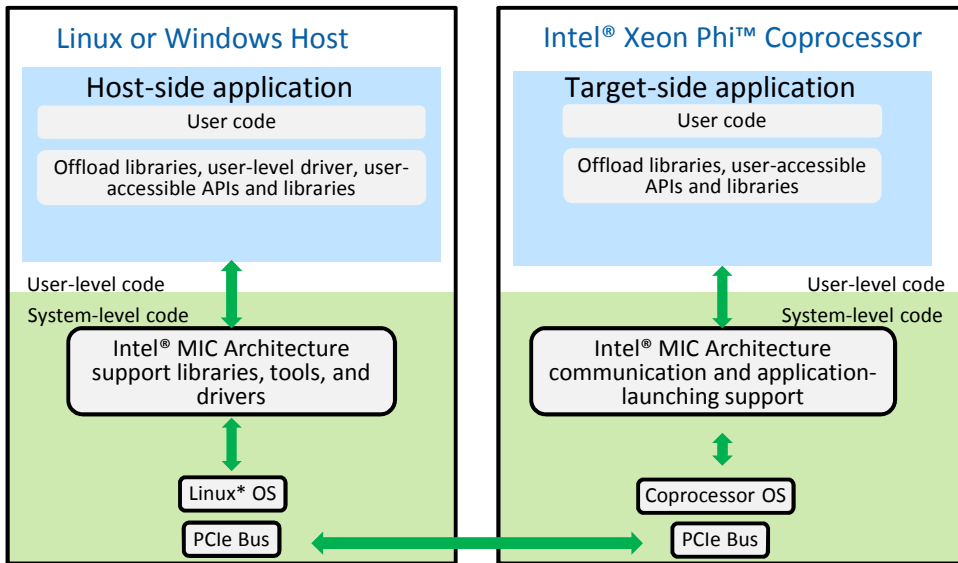
Enabling and advancing parallelism

Intel tools, libraries and parallel models extend to multicore, many-core and heterogeneous computing

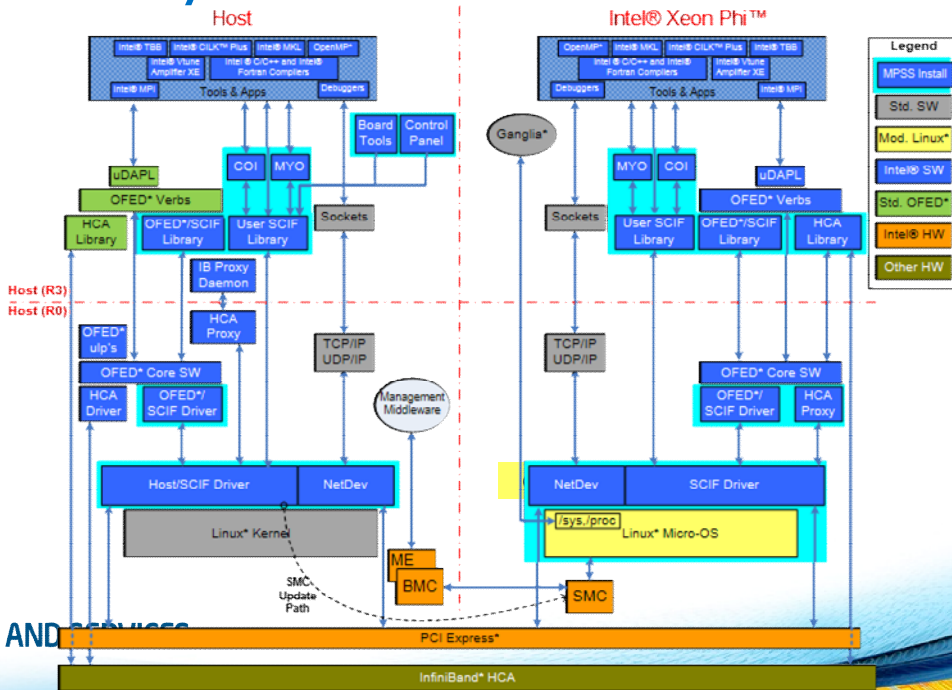


SOFTWARE AND SERVICES

System architecture overview



Detailed system architecture overview



SOFTWARE AND SERVICES

InfiBand* HCA

Intel® Xeon Phi™ infrastructure

- Operating System (OS)
 - Embedded Linux* based on Conjure/Yocto (very few customizations)
 - You can assume at least a BusyBox environment
- Other infrastructure
 - Intel® Manycore Platform Software Stack (Intel® MPSS)
 - Intel® Coprocessor Offload Infrastructure (Intel® COI)
 - Intel® Symmetric Communications Infrastructure (Intel® SCI)



Programming environment

Intel® Xeon Phi™, Knights Corner (KNC)

SOFTWARE AND SERVICES

Development environment

- Standard Intel® development environment is available:
 - Intel® Composer: **C**, **C++** and **Fortran** Compilers
 - **Standard runtime libraries**, including pthreads*
 - **OpenMP***
 - Intel® **MPI** Library support for the Intel® Xeon Phi™ Coprocessor
 - Parallel Programming Models
 - Intel® Threading Building Blocks (**Intel® TBB**)
 - **Intel® Cilk™ Plus**
 - Tools
 - Intel support for **gdb**, Intel® VTune™ Amplifier XE
 - **Intel Performance Libraries** (e.g. Intel Math Kernel Library)
 - Three versions: host-only, coprocessor-only, heterogeneous

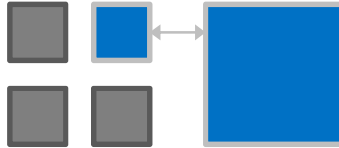
Programming models

Native



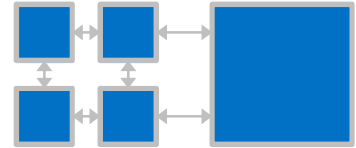
- Target Code: Highly parallel (threaded and vectorized) throughout
- Potential Bottleneck: Serial/scalar code

Offload



- Target Code: Mostly serial, but with expensive parallel regions
- Potential Bottleneck: PCIe data transfers

Symmetric



- Target Code: Highly parallel and performs well on both platforms
- Potential Bottleneck: Load imbalance

Programming models

- MPI
 - Used for “native” and “symmetric” execution
 - Can launch ranks across processors and coprocessors
- OpenMP
 - Used for “native”, “offload” and “symmetric” execution
 - OpenMP 4.0 standard supports device constructs for offloading
- Many real-life HPC codes use a native MPI/OpenMP hybrid
 - Balance task granularity by tuning combination of ranks/threads (e.g.16 MPI ranks x 15 OpenMP threads)

Standards and existing code

- Existing source code
 - In most cases, code can be simply recompiled
 - All IA/x86 assumptions hold incl. legacy instructions
 - Cross-compiled code can be used in offload section e.g., Intel® Threading Building Blocks
- Targeting Intel® Xeon Phi™ does not waste effort
 - Tuning takes effort, but leverages existing standards
 - Optimizations usually lead to improved performance on Intel® Xeon®

Intel® Parallel Studio XE 2016



Composer Edition	Professional Edition	Cluster Edition
Intel® C++ Compiler Intel® Fortran Compiler Intel® Data Analytics Acceleration Library Intel® Threading Building Blocks Intel® Integrated Performance Primitives Intel® Math Kernel Library Intel® Cilk™ Plus & Intel® OpenMP*	Intel® C++ Compiler Intel® Fortran Compiler Intel® Data Analytics Acceleration Library Intel® Threading Building Blocks Intel® Integrated Performance Primitives Intel® Math Kernel Library Intel® Cilk™ Plus & Intel® OpenMP*	Intel® C++ Compiler Intel® Fortran Compiler Intel® Data Analytics Acceleration Library Intel® Threading Building Blocks Intel® Integrated Performance Primitives Intel® Math Kernel Library Intel® Cilk™ Plus & Intel® OpenMP*
	Intel® Advisor XE Intel® Inspector XE Intel® VTune™ Amplifier XE	Intel® Advisor XE Intel® Inspector XE Intel® VTune™ Amplifier XE Intel® MPI Library Intel® Trace Analyzer and Collector
Bundle or Add-on: Rogue Wave IMSL* Library	Add-on: Rogue Wave IMSL* Library	Add-on: Rogue Wave IMSL* Library

Additional configurations including, floating and academic, are available at: <http://intel.ly/perf-tools>

SOFTWARE AND SERVICES

Software architecture

Multicore Options

Intel® Math Kernel Library MPI*

Intel® Threading Building Blocks
Intel® Cilk™ Plus

OpenMP*

Pthreads*

Vector Options

Intel® Math Kernel Library

Auto vectorization

Semi-auto vectorization:
#pragma (vector, ivdep, simd)

Array Notation: Fortran,
Intel® Cilk™ Plus

C/C++ Vector Classes
(F32vec16, F64vec8)

OpenCL*

Intrinsics

Ease of use



Fine control

Execution models

- Intel MKL Automatic Offload (AO)
 - Transparent data transfer and execution management
 - Limited to key functions (sufficient FLOP/Byte ratio)
 - Automatically uses host and (multiple) targets
 - No code changes required
- Compiler Assisted Offload (CAO)
 - Explicit control of data transfer / persistence
 - Intel Compiler offload pragmas/directives
 - Language Extension for Offload (LEO)
 - OpenMP* 4.0 device constructs
 - Can be used together with Automatic Offload
- Native Execution
 - Coprocessors used as independent nodes

Intel® Math Kernel Library (Intel® MKL)

- Single -and multi-threaded libraries
- Cluster support for important domains
- Support for large problem sizes (ILP)
- Conditional Numerical Reproducibility (CNR)
- Support for Intel® Xeon Phi™ coprocessors
 - Automatic offload, and compiler-assisted offload
 - Manycore-hosted execution, cluster support, etc.
- Enabled early for future hardware
 - KLN support: AVX-512 instruction set

SOFTWARE AND SERVICES

Intel® Math Kernel Library (Intel® MKL)

Linear Algebra

- BLAS
- LAPACK
- ScaLAPACK
- Sparse BLAS
- Sparse Solvers
- Iterative
- PARDISO* SMP & Cluster

Fast Fourier Transforms

- Multidimensional
- FFTW interfaces
- Cluster FFT

Vector Math

- Trigonometric
- Hyperbolic
- Exponential
- Log
- Power
- Root

Vector RNGs

- Congruential
- Wichmann-Hill
- Mersenne Twister
- Sobol
- Neiderreiter
- Non-deterministic

Summary Statistics

- Kurtosis
- Variation coefficient
- Order statistics
- Min/max
- Variance-covariance

And More...

- Splines
- Interpolation
- Trust Region
- Fast Poisson Solver

Intel® MKL Automatic Offload (AO)

- Control automatic offload (hybrid execution!)
 - Environment variable: `MKL_MIC_ENABLE=1`
 - Remember: sufficient problem size needed (Byte/FLOP ratio)
 - Service functions take precedence (work division, etc.)
- Supported functions
 - BLAS level 3: `xGEMM`, `xTRMM`, `xTRSM`
 - LAPACK: Cholesky, LU, QR
- Offload report (also applies to CAO)
 - `OFFLOAD_REPORT=<0|1|2>`, or call
 - `mk1_mic_set_offload_report(...)`



Programmability and performance

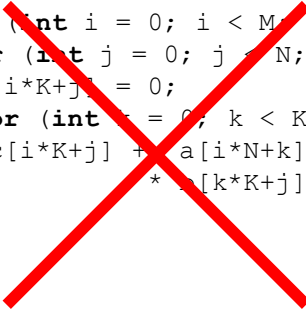
Intel® Xeon Phi™, Knights Corner (KNC)

SOFTWARE AND SERVICES

How and where to optimize?

1. Choose a library that solves the problem
or
2. Choose an appropriate algorithm and optimize your own code
 - a) Across SIMD lanes
 - b) Across multiple threads
 - c) Across multiple nodes

```
for (int i = 0; i < M; ++i) {  
    for (int j = 0; j < N; ++j) {  
        c[i*K+j] = 0;  
        for (int k = 0; k < K; ++k) {  
            c[i*K+j] + a[i*N+k]  
                * b[k*K+j];  
        }  
    }  
}
```



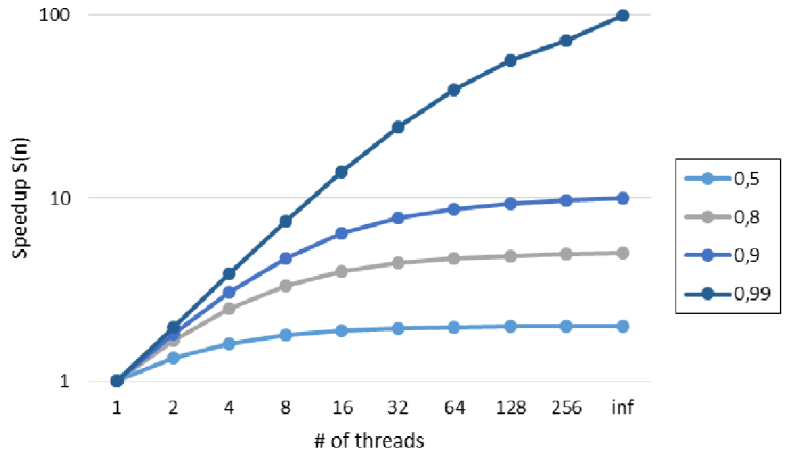
Intel Performance Library!

Multithreading: Amdahl's law

- Speedup with n threads is limited by the *parallelizable* fraction P of the program

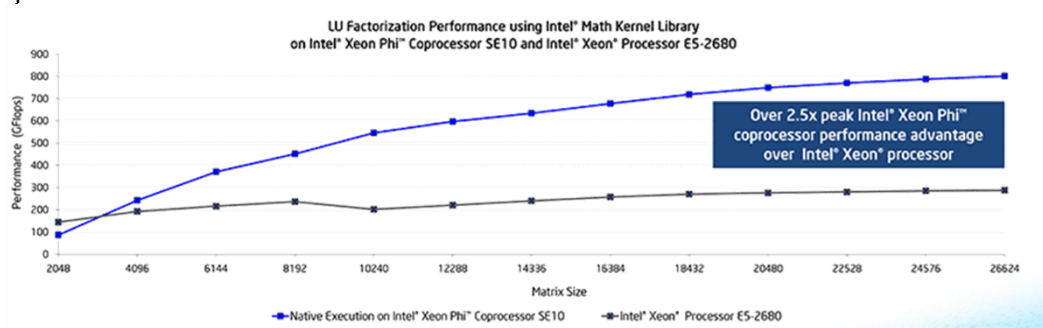
$$\rightarrow S(n) = \frac{1}{(1-P) + \frac{P}{n}}$$

- Up to 240 threads may be needed by Intel® Xeon Phi™!



Performance

- xGEMM, STREAM, and SMP Linpack
<http://www.intel.com/content/www/us/en/benchmarks/xeon-phi-product-family-performance-brief.html>
- xGEMM, Cholesky / LU / QR Decomposition, SMP Linpack, etc.
<http://software.intel.com/en-us/intel-mkl#pid-12768-1295>
- Example:



Configuration Info - Software Versions: Intel® Math Kernel Library (Intel® MKL) 11.0.1, Intel® Manycore Platform Software Stack (MPSS) 2.1.4348; Hardware: Crown Pass Software Development System, Intel® Xeon® Processor E5-2680 v2, Eight Core CPUs
80081MLC 2-7044 135GB DDR3 RAM 11333MHz Intel® Xeon Phi™ Coprocessor SE10, Step B1, 61 cores (30.5MB total cache, 1.1GHz), 8GB COORS Memory; Operating System: RHEL 6.1 GA x86_64

Real application performance

- A number of real applications have reported a speedup on Intel® Xeon Phi™
- For references, see Intel® Xeon Phi™ Coprocessor – Applications and Solutions Catalog
<https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-applications-and-solutions-catalog>

Conclusions

- Vectorization and thread parallelism are keys to good performance on an Intel® Xeon Phi™
- Code modernization benefits both the processor and the coprocessor
- Some of the limitations of the current coprocessor generation will be removed with Knight's Landing

References

- James Jeffers, James Reinders, *"Intel® Xeon Phi™ Coprocessor High Performance Programming"*, Morgan Kaufmann, 2013.
- Alexander Supalov, Andrey Semin, Michael Klemm, Chris Dahnken, *"Optimizing HPC Applications with Intel® Cluster Tools"*, Apress Open, 2015.
- PRACE Xeon Phi Best Practice Guide
<http://www.prace-ri.eu/Best-Practice-Guide-Intel-Xeon-Phi-HTML>
- Xeon Phi Developer's Quick Start Guide
<http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide>



References

- James Jeffers, James Reinders, "*High Performance Parallelism Pearls Volume One, 1st Edition. Multicore and Many-core Programming Approaches*", Morgan Kaufmann, 2014.
- James Jeffers, James Reinders, "*High Performance Parallelism Pearls Volume Two, 1st Edition. Multicore and Many-core Programming Approaches*", Morgan Kaufmann, 2015.

