Numerical Computation on Intel® Xeon Phi™ Coprocessors Using the Intel® Compilers and Math Libraries

October 15, 2013

Marius Cornea

Intel Corporation

Agenda

- Overview of the Intel® Xeon Phi™ product family
- Floating-Point support on Intel® Xeon Phi™ Coprocessors
- Intel® Compiler features and Math Libraries for Intel® Xeon Phi™ Coprocessors
- Intel® MKL support and usage models on Intel® Xeon Phi™
- Some performance and efficiency numbers
- Next-generation Intel Xeon Phi™ (Co)Processors
- Where to find more information

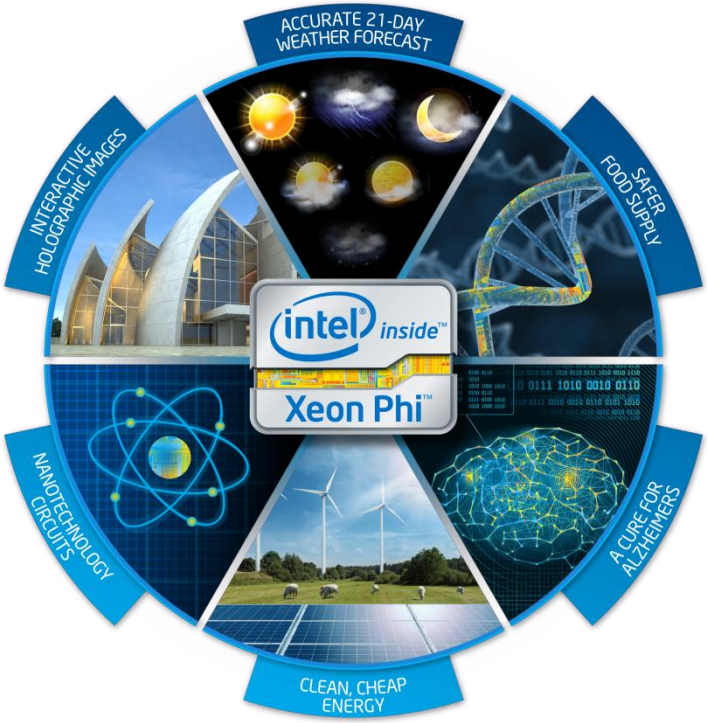
The Intel® Xeon Phi™ Coprocessor

Intel® Xeon Phi™ Product Family
based on Intel® Many Integrated Core (MIC) Architecture

Many Core compared to Multi-core
- Many smaller lower power Intel processor cores
- Wider vector processing units for greater (generally) floating-point performance/watt

Highly Parallel
- Higher aggregate performance relative to multi-core Intel® Xeon® processors
- Supports Data Parallel, Thread Parallel, Process Parallel programming
- Higher memory bandwidth relative to multi-core Intel® Xeon® processors

Highly Programmable
- Standards-based: C/C++/FORTRAN
- Abstract: no requirement to program to the underlying hardware
- More than an accelerator: fully addressable, independent node in a cluster
- Full support by Intel® Cluster Studio XE



Will help solve today's toughest challenges which also present the greatest compute complexity

A Coprocessor, Not an Accelerator

Optimization Notice

Intel® Xeon Phi™ Product Family

based on Intel® Many Integrated Core Architecture

Future Knights products

Future (co)processor

Knights Corner

1st Intel® MIC product

2012

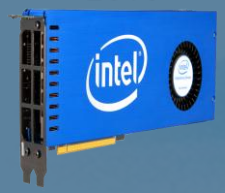
22 nm process

>50 Intel Architecture cores

Knights Ferry

Software Development Platform

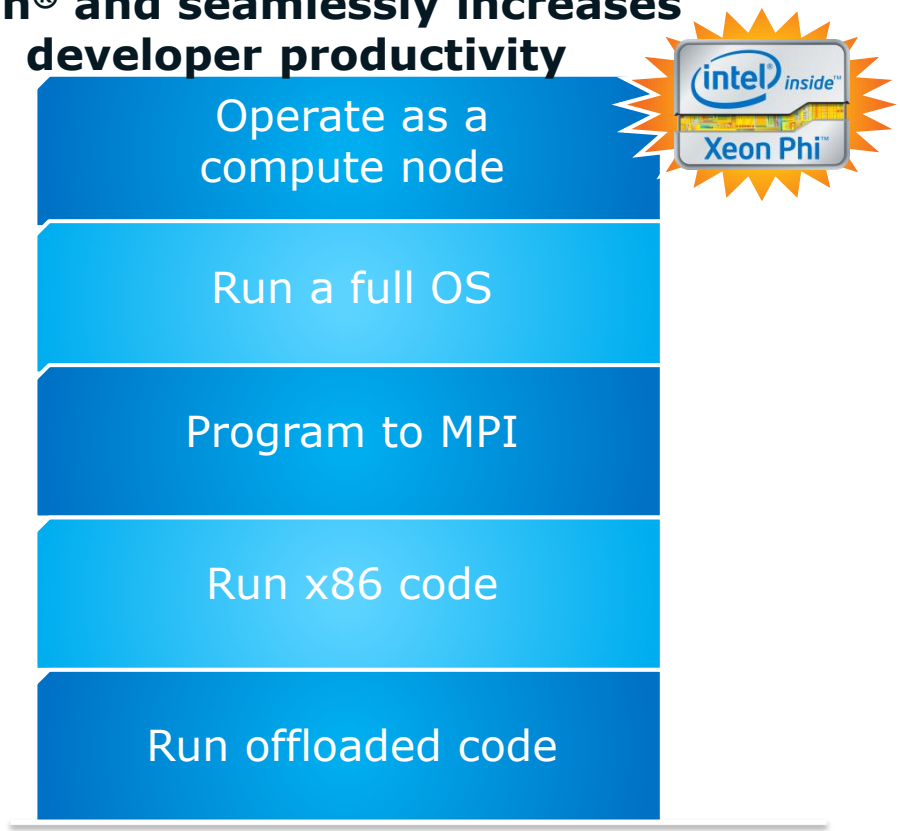
2010



Intel® Xeon Phi™ Coprocessors: General Purpose IA HW

Restrictive architectures

Intel® Xeon Phi™ shares a programming model with Intel® Xeon® and seamlessly increases developer productivity

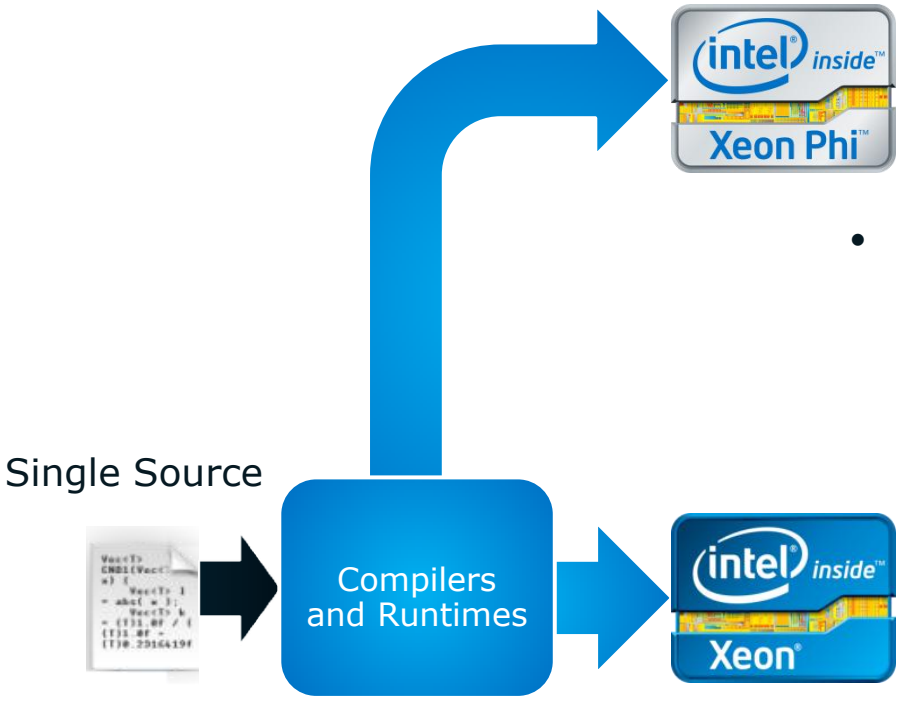


Custom HW Acceleration

Intel® Xeon Phi™ Coprocessor

Restrictive architectures limit the ability for applications to use arbitrary nested parallelism, functions calls and threading models

Intel® Xeon Phi™ Coprocessor

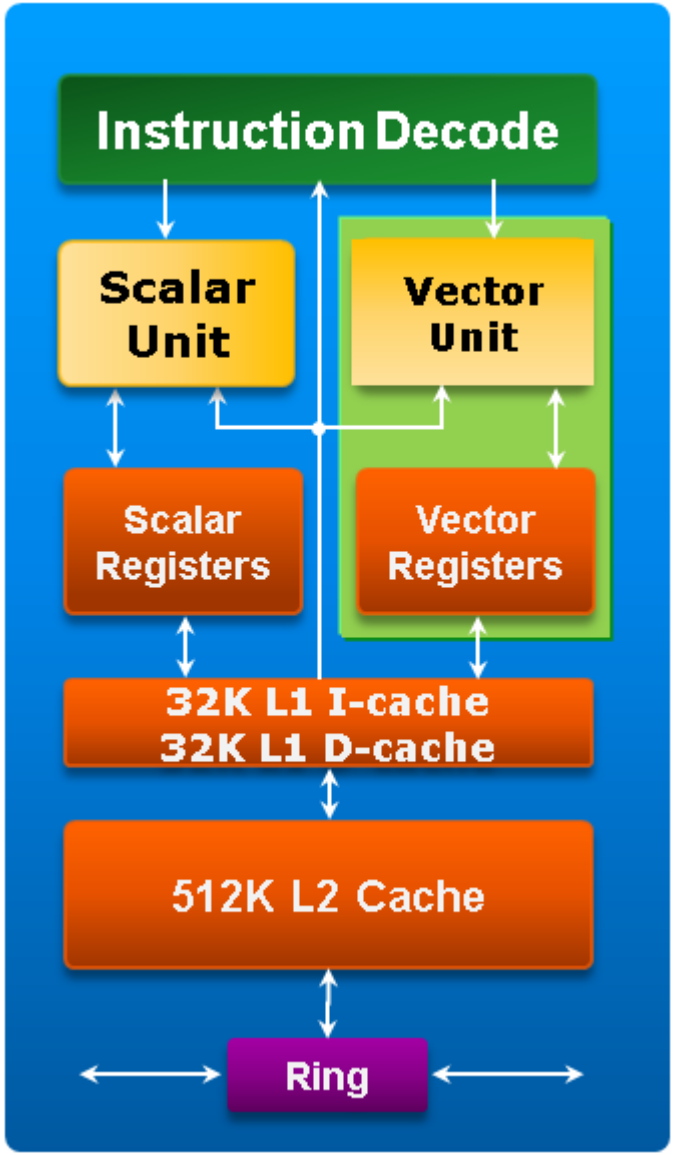


- Up to 2.2x higher **memory bandwidth** than on an Intel® Xeon® processor E5 family-based server
- Up to 4x better **performance per watt** than with an Intel® Xeon® processor E5 family-based server

- **First product in the Intel Many Integrated Core Architecture** (MIC)
  - Launched in 2012.
  - Up to 61 IA cores w/ shared memory; ~1.1 GHz; up to 244 threads; short in-order pipeline
  - Running Linux
  - Pentium-like x86 cores with 512-bit vector units
  - ~1 TFLOPS peak DP, ~2 TFLOPS SP
- **8 to 16 GB GDDR5 memory, 240 to 350 GB/sec bandwidth**
- **Linux OS**, IP addressable
- **Standard programming languages** and tools: C++, Fortran, MPI, OpenMP

Intel® Xeon Phi™ Coprocessor Vector Instructions

- **64 bit execution environment** , with basic support for float64 and int64 logical operations
- **32KB L1 data cache, 32KB L1 instruction cache, 512KB L2 cache/core**
- **32** 512-bit wide **vector registers (ZMM)** w/ native support for 32 and 64 bit FP and integer data; conversions for native types
- **Ternary instructions** (in most cases)
- **Vector mask support:** 8 vector mask registers: conditional execution over the 16/8 elements in a vector instruction; vectorizing loops w/ conditional statements; updating vector masks
- **Coherent memory model:** follows the *Intel® 64* architecture standard
- **Gather/Scatter** instructions manipulate irregular data patterns of memory
- **Swizzles:** permuting and/or replicating 32 or 64-bit vector elements



Intel® Xeon Phi™ Floating-Point Support

- The **same floating-point data types** as the Intel® Xeon processor
  - Single (32-bit) and double (64-bit) precision supported in HW
  - Extended (80-bit) precision supported w/ the x87 instruction set
- **Denormalized numbers** and gradual underflow are supported
- The same set of **rounding modes** is supported as for Intel Xeon processors
- The **MXCSR (control and status register)** provides:
  - Exception flags to indicate SIMD floating-point exceptions signaled by floating-point instructions operating on ZMM registers: IE, DE, ZE, OE, UE, PE
  - Rounding behavior and control: DAZ, FZ and RC
  - Exception Suppression: DUE – Disable Unmasked Exc. (always 1)
- When **SIMD floating-point exceptions** occur, exception reporting using exception flags in the MXCSR register is supported, but traps (unmasked exceptions) are not; SIMD FP exceptions are precise
- **Fused multiply-add** (**FMA**) – ternary w/ 3 sources; one is dest.
- Suppress All Exceptions Attribute (**SAE**)
- Some instructions support a **'fixed' rounding mode** through an immediate or encoded in the instruction swizzle attribute (higher precedence than the MXCSR)
- **Floating-point division and square root** are implentened in SW

Intel® Xeon Phi™ Coprocessor Family Reference Table

SKU #	Form Factor, Thermal	Peak Double Precision	Max # of Cores	Clock Speed (GHz)	GDDR5 Memory Speeds (GT/s)	Peak Memory BW	Memory Capacity (GB)	Total Cache (MB)	Board TDP (Watts)	Process
SE10P (special edition)	PCIe Card(1)	1073 GF	61	1.1	5.5	352	8	30.5	300	22nm
SE10X (special edition)	PCIe Card(3)	1073 GF	61	1.1	5.5	352	8	30.5	300	
7120X	PCIe Card(3)	1208 GF	61	1.238	5.5	352	16	30.5	300	
7120P	PCIe Card(1)	1208 GF	61	1.238	5.5	352	16	30.5	300	
5120D	PCIe Card(4)	1011 GF	60	1.053	5.0	352	8	30	245	
5110P	PCIe Card(1)	1011 GF	60	1.053	5.0	320	8	30	225	
3100A	PCIe Card(2)	1003 GF	57	1.1	5.0	240	6	28.5	300	
3100P	PCIe Card(1)	1003 GF	57	1.1	5.0	240	6	28.5	300	

(1)PC = Passively Cooled
(2)AC = Actively Cooled
(3)NTS = No Thermal Solution
(4)DF = Dense Form, no Thermal Solution

PCIe Card, Actively Cooled

PCIe Card, Passively Cooled

Notice: This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information. Contact your local Intel sales office or your distributor to obtain the latest specification before placing your product order.
Knights Corner 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. All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel Compilers for Intel® Xeon® Processors and Intel® Xeon Phi™ Coprocessors

Intel® Compiler Floating-Point Programming Objectives

- **Intel provides C/C++ and Fortran compilers**
- **Objectives are the same for Intel® Xeon Phi™ Coprocessors and for Intel® Xeon® Processors**
- **Accuracy:** produce results "close" to the correct value
  - Measure differences as relative errors, possibly in ulp-s
- **Performance**
  - Produce the most efficient code possible
- **Reproducibility**
  - Produce consistent results
    - From one run to the next
    - From one set of build options to another
    - From one compiler to another
    - From one platform to another
- **Developers must make tradeoffs between accuracy, performance, and reproducibility**

Intel® C/C++ Compiler Floating-Point Semantics

The most important compiler switch for FP semantics:

–fp-model

- Lets you choose the floating-point semantics at a coarse granularity
- Lets you specify the compiler rules for:
  - **Value safety** (main focus)
  - FP expression evaluation
  - FPU environment access
  - Precise FP exceptions
  - FP contractions (fused multiply-add, "fma")

- Also pragmas in C99 standard, e.g.

  #pragma STDC FENV_ACCESS (ON|OFF)

Intel® C/C++ Compiler Floating-Point Semantics

-fp-model

- fast [=1] allows value-unsafe optimizations (default)
- fast=2 allows additional approximations
- precise value-safe optimizations only
- source | double | extended imply "precise" unless overridden
  see "FP Expression Evaluation" for more detail
- except enable floating-point exception semantics
- strict precise + except + disable fma +
  don't assume default floating-point environment

-fp-model precise -fp-model source

- recommended for best reproducibility
- also for ANSI/ IEEE standards compliance, C++ & Fortran
- "source" is default with "precise" on Intel 64 Linux

Denormalized numbers and Flush-to-Zero (FTZ)

- Denormals extend the (lower) range of IEEE floating-point values, at the cost of:
  - Reduced precision
  - Reduced performance (can be 100 X for ops with denormals)
- The Intel® Compiler provides –ftz/-no-ftz switches for this
- If your application creates but does not depend on denormal values, setting these to zero may improve performance ("abrupt underflow", or "flush-to-zero",)
  - Done in SSE or AVX hardware, therefore fast
  - Happens by default at –O1 or higher
  - -no-ftz or –fp-model precise will prevent flush-to-zero
    - Must compile main() with this switch to have an effect
    - -fp-model precise –ftz to get "precise" without denormals
  - Not available for x87, denormals always generated
    - (unless trapped and set to zero in software – very slow)

Intel® C/C++ Compiler Math Libraries

- Math Libraries in the Intel® C/C++ Compiler
  - LIBM
  - SVML
  - DFP
  - BFP754

- The compiler generates code that is fully compliant with the ANSI language standards and the IEEE-754 standard for floating-point arithmetic.

- Compiler options give the user control over the tradeoffs between optimizations for performance, accuracy, reproducibility of results and strict conformance with these standards

- Compiler options that would unmask floating-point exceptions are unsupported on Intel(R) MIC architecture.

Compiler Switches for FP Math Libs Functions

- Accuracy vs. speed for math functions can be controlled with:
  –fp-model precise, by adding -fast-transcendentals –no-prec-div –no-prec-sqrt
  -fimf-precision=<high|medium|low> [:func1,func2,…] <- list of math functions affected by this
  -**fimf-max-error** <- need to specify a value here
  -fimf-accuracy-bits <- need to specify a value here
  -fimf-absolute-error <- need to specify a value here
  -fimf-domain-exclusion= <value>; the bits of <value> indicate domains for which the compiler need not generate special code

1	extreme values excluded (close to singularities or infinities; denormals)
2	NaNs are excluded (so the function may not work 'correctly' for NaNs)
4	infinities
8	denormals
16	zeros

  e.g. -fimf-domain-exclusion=31 excludes all of these, for all functions
  can be restricted to specific functions, e.g. -fimf-domain-exclusion=15:/sqrt,sqrtf gives fast, inlined versions of single & double precision square root (will not work well for 1 – extreme values, 2 – NaNs, 4 – infinities, and 8 – denormals; ok 16 - for zeros)
  -fp-model-fast=2 implies –fimf-domain-exclusion=15

- The compiler is aggressive in inlining faster functions with limited accuracy or domains when these switches are specified

Compiler FP Support for Intel® Xeon Phi™

Floating-point exception flags are set by Intel® Initial Multi-Core Instructions (Intel® IMCI)

- the flags can be read, but unmasking & trapping are not supported
- attempts to unmask will result in seg fault
- -fpe0 (Fortran) and -fp-trap (C) are disabled
- -fp-model except or strict will yield (slow!) x87 code that supports unmasking and trapping of floating-point exceptions

Denormals are supported by Intel IMCI (but slow, like host)

- Needs –no-ftz or –fp-model precise (like host)

512 bit vector transcendental math functions available (SVML)

- Fast versions of division and square root with -fp-model fast=2
  - Sets –fimf-domain-exclusion to exclude special cases
- Both SVML and fast inlined divide and sqrt sequences available
- Fast versions of powers and logs to base 2
- See Differences in floating-point arithmetic between Intel(R) Xeon processors and the Intel Xeon Phi(TM) coprocessor for details and status

Comparing Floating-Point Results between Intel® Xeon processors & Intel® Xeon Phi™ Coprocessors

Different architectures – expect some differences

- Different optimizations
- Use of fused multiply-add (FMA)
- Different implementations of math functions

How to minimize differences (e.g. for debugging):

- Build with –fp-model precise (both architectures)
- Build with –no-fma (Intel® MIC architecture)
- Select high accuracy math functions
  - (e.g. -fimf-precision=high; default with –fp-model precise)
- Choose reproducible parallel reductions
  - Or run sequentially, if you have the patience…
- Remember, the true uncertainty of your result is probably much greater!

19

Programming for Intel® Xeon Phi™

Intel® Xeon Phi™ Programming: Levels of Parallelism

61 cores, 4 hardware threads per core.

– Launch one or several MPI processes with several threads per process (e.g. Open MP).

– Run at least 2 threads per core for efficiency.

512-bit vector units: capable of 8 simultaneous arithmetic operations in DP, 16 in SP

– Use SIMD instructions to benefit from vector units.

– Compiler may generate vector instructions to execute several loop iterations in parallel.

Peak performance - execute vector FMA in all cores, e.g.:

- DP: 61 Cores × 1.1 GHz × 8 FMA/clk × 2 FPOps/FMA = 1.07 TFLOPS
- SP: 61 Cores × 1.1 GHz × 16 FMA/clk × 2 FPOps/FMA = 2.15 TFLOPS

Note: Overall peak of Xeon Phi is ~ 5x peak of CPU, but single-threaded scalar performance is ~ 0.1x CPU core; a fast sequential operation on CPU can become a bottleneck on Xeon Phi; need to make everything parallel the coprocessor, or use offload mode and perform sequential operations on host CPU

Software Development Tools/Environments

Intel® MPSS – Intel® Manycore Platform Software Stack

Compilers

- Intel C++ Composer XE 2013
- Intel® Fortran Composer XE 2013

Libraries packaged with the compilers

- Intel® Math Kernel Library (Intel® MKL) (optimized)
- Intel® Threading Building Blocks (Intel® TBB)
- Intel® Integrated Performance Primitive (Intel® IPP)

Libraries packaged separately

- Intel® MPI for Linux*
- Intel® Trace Collector and Analyzer
- Intel® SDK for OpenCL* Applications XE 2013

Debugger

- Intel® Debugger for Intel® 64 and Intel® MIC Architecture
- Intel® C++ Eclipse* Product Extension w/ Debugging

Profiling Tools

- Intel® VTune™ Amplifier XE 2013 for Linux
- Intel® Inspector XE 2013
- Intel® Advisor XE 2013

Intel® Xeon Phi™ Programming

- **Native mode**: use as a 61-core CPU under Linux:
  C++ / Fortran + OpenMP / TBB / Cilk Plus

- **Symmetric mode**: use as a node of cluster, can host several MPI processes:
  C++ / Fortran + MPI + OpenMP / TBB / Cilk Plus

- **Offload mode**: use as a coprocessor, run kernels launched from the host CPU:
  C++ / Fortran + Offload compiler directives + OpenMP / TBB / Cilk Plus

Optimization recommendations for CPUs also apply to Xeon Phi: multi-threading, vectorization, cache-friendliness, etc.

- Differences between Xeon Phi and CPU that make it more challenging:
  - Many more threads: 120 to 240 vs. 4 to 16
  - Wider vector units: 512-bit vs. 128 to 256-bit

Intel® Xeon Phi™ Programming: Using the Offload Compiler w/ Explicit Memory Copy Model

- Example of reduction operation: ans = a[0] + a[1] + … + a[n-1]
- In C/C++:

```
float reduction(float *data, int size) {
  float ret = 0.f;
  for (int i=0; i<size; ++i) { ret += data[i];}
  return ret;
}
```

- Serial Reduction with Offload:

```
float reduction(float *data, int size) {
  float ret = 0.f;
  #pragma offload target(mic) in(data:length(size))
  for (int i=0; i<size; ++i) { ret += data[i]; }
  return ret;
}
```

- Vector Reduction with Offload in C/C++:

```
float reduction(float *data, int size) {
  float ret = 0;
  #pragma offload target(mic) in(data:length(size))
  ret = __sec_reduce_add(data[0:size]); //Intel® Cilk™ Plus
                                        //Extended Array Notation
  return ret;
}
```

- An implicit memory copy model can also be used

Intel® Xeon Phi™ Programming: Native Compilation

- Applications can also be run natively on the Intel® Xeon Phi™ Coprocessor
  - The coprocessor is treated as a standalone multicore computer
  - Build the binary on the host system, and copy the binary and other related binaries or data to the Intel® Xeon Phi™ Coprocessor's file system (or make them visible via NFS)

- Example:
  - icc –mmic –vec-report3 –openmp openmp_sample.c
  - scp a.out mic0:/tmp/a.out
  - scp /opt/intel/composerxe/lib/mic/libiomp5.so mic0:/tmp/libiomp5.so
  - ssh mic0
  - export LD_LIBRARY_PATH=/tmp
  - ulimit –s unlimited
  - cd /tmp
  - ./a.out

Intel® Xeon Phi™ Programming: Parallel Programming Options

- Parallel Programming Options
  - Intel Threading Building Blocks (Intel® TBB)
  - OpenMP*
  - Intel® Cilk Plus
  - pthreads*
- Example: Using OpenMP in Offloaded Reduction Code

```
float OMP_reduction(float *data, int size) {
  float ret = 0;
  #pragma offload target(mic) in(size) in(data:length(size))
  {
    #pragma omp parallel for reduction(+:ret)
    for (int i=0; i<size; ++i) { ret += data[i]; }
  }
  return ret;
}
```

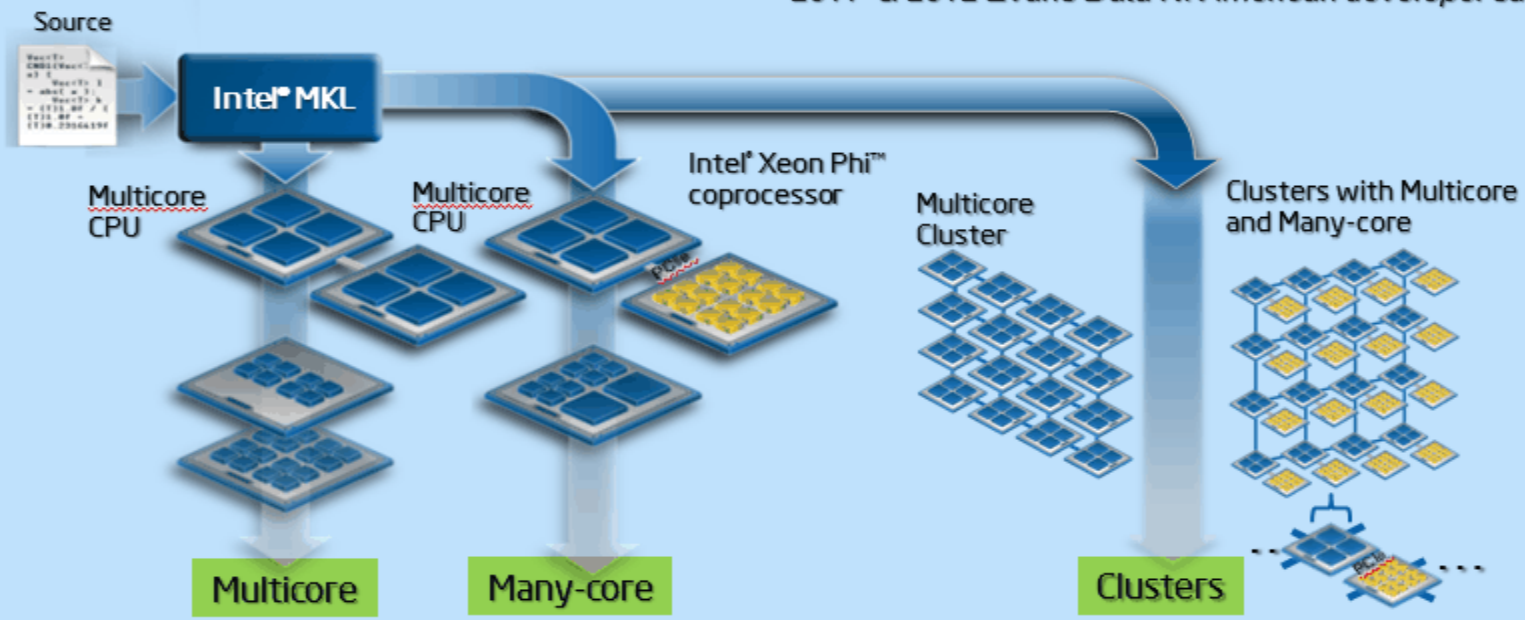
Intel® MKL

Using Intel® MKL on Intel® Xeon Phi™ Coprocessors

Intel® MKL is industry's leading math library *

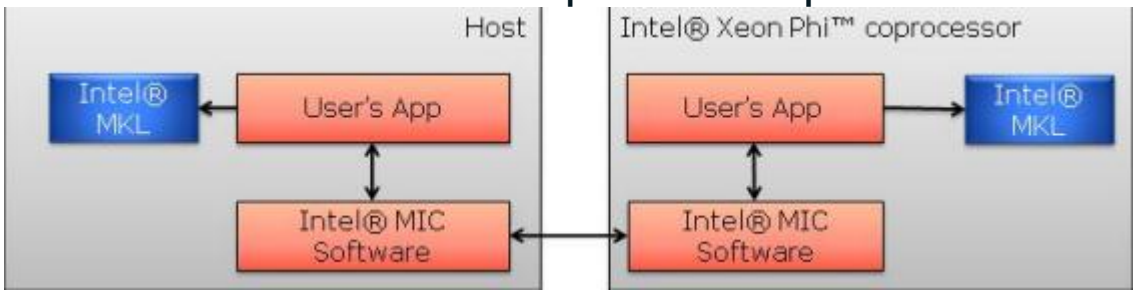
Linear Algebra	Fast Fourier Transforms	Vector Math	Vector Random Number Generators	Summary Statistics	Data Fitting
•BLAS •LAPACK •Sparse solvers •ScaLAPACK	•Multidimensional (up to 7D) •FFTW interfaces •Cluster FFT	•Trigonometric •Hyperbolic •Exponential, Logarithmic •Power / Root •Rounding	•Congruential •Recursive •Wichmann-Hill •Mersenne Twister •Sobol •Neiderreiter •Non-deterministic	•Kurtosis •Variation coefficient •Quantiles, order statistics •Min/max •Variance-covariance •…	•Splines •Interpolation •Cell search

* 2011 & 2012 Evans Data N. American developer surveys



Using Intel® MKL on Intel® Xeon Phi™ Coprocessors

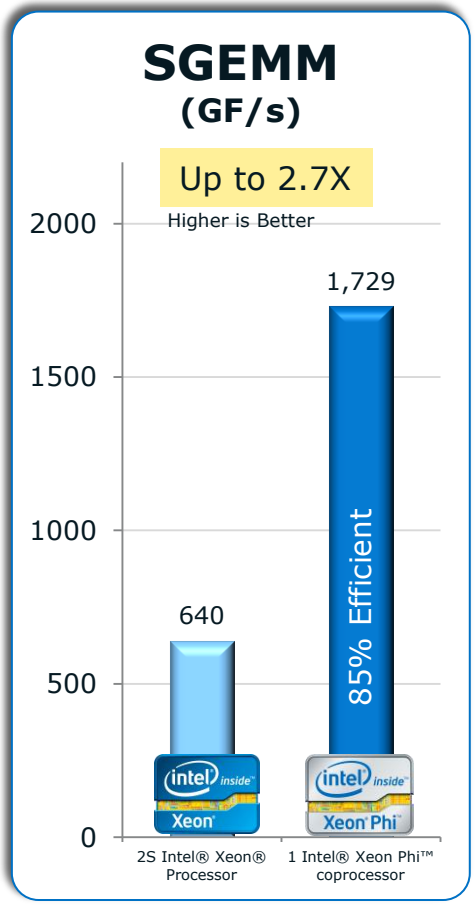
- Native Acceleration (NAcc) mode - most common for offload users
  - Data transferred by programmer through offload compiler pragmas and semantics to be used by Intel MKL calls within an offloaded region or function.
  - It applies to BLAS, LAPACK, FFT, VML, VSL, (Sparse Matrix Vector), and required Intel MKL Service functions
  - It can also be used in native Intel® Xeon Phi™ code – MKL shared libraries must be copied to coprocessor before execution

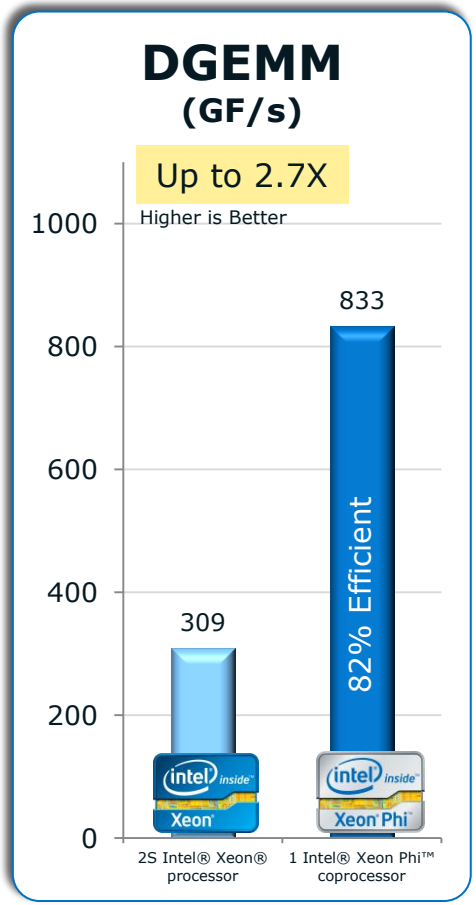


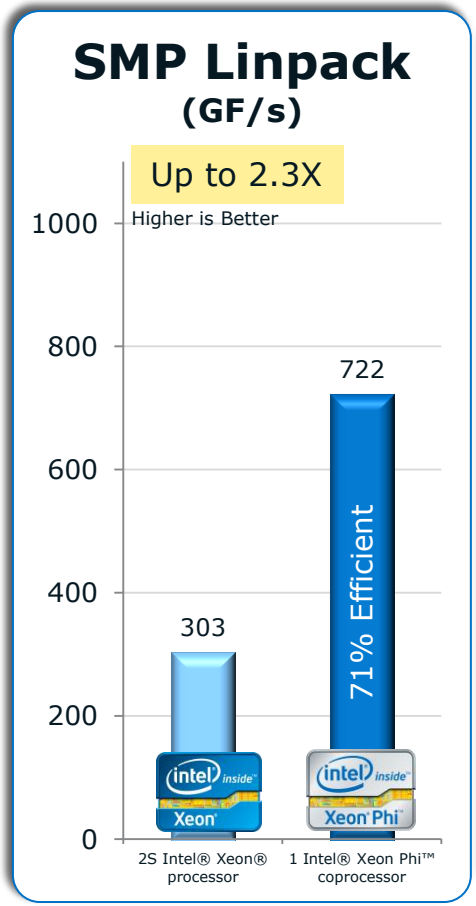
- Intel® MKL Automatic Offload Model
  - Some of the host Intel® MKL functions are Automatic Offload-aware; must be preceded by a call to mkl_mic_enable(), automatically divides work at runtime between the host and the coprocessor
  - No code changes required; automatically uses both host and target; transparent data transfer and execution management

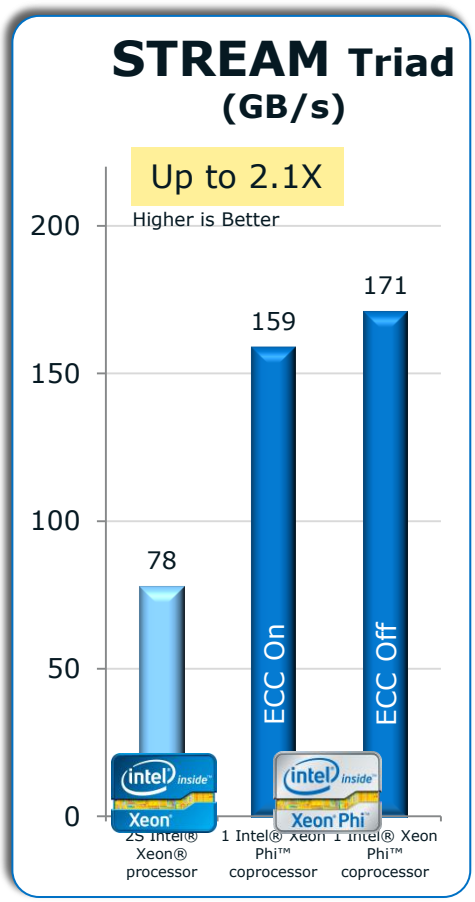
Some Performance Numbers

Synthetic Benchmark Results (Intel® MKL) (5110P)









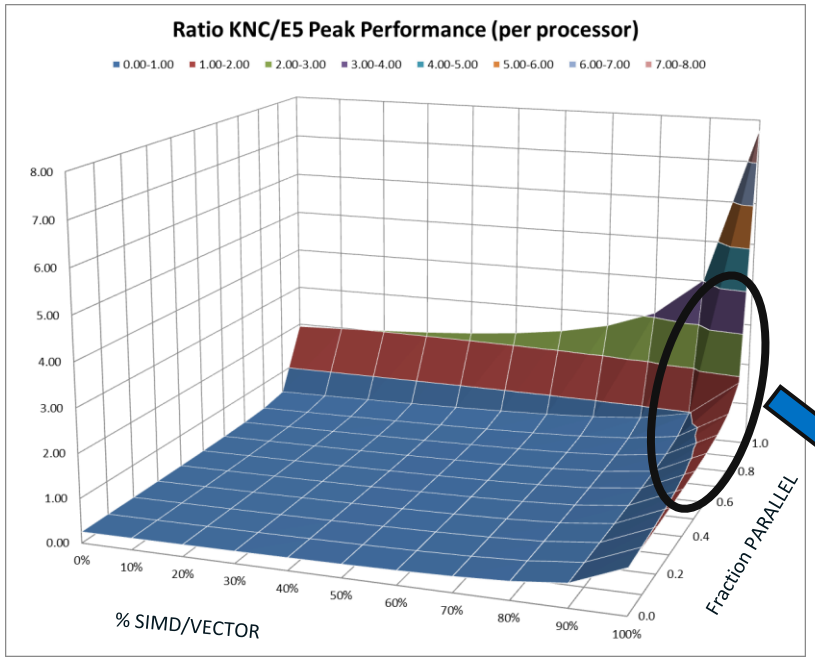
Coprocessor results: Benchmark run 100% on coprocessor, no help from Intel® Xeon® processor host (aka native)

Notes

1. **Intel® Xeon® Processor E5-2670 used for all SGEMM Matrix = 13824 x 13824 , DGEMM Matrix 7936 x 7936, SMP Linpack Matrix 30720 x 30720**
2. **Intel® Xeon Phi™ coprocessor 5110P (ECC on) with "Gold Release Candidate" SW stack SGEMM Matrix = 11264 x 11264, DGEMM Matrix 7680 x 7680, SMP Linpack Matrix 26872 x 28672**

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. Source: Intel Measured results as of October 26, 2012 Configuration Details: Please reference slide speaker notes. **For more information go to http://www.intel.com/performance**

Intel® Xeon Phi™ Coprocessor: Increases Application Performance up to 10x



Application Performance Examples

Customer	Application	Performance Increase[1] vs. 2S Xeon*
Los Alamos	Molecular Dynamics	Up to 2.52x
Acceleware	8th order isotropic variable velocity	Up to 2.05x
Jefferson Labs	Lattice QCD	Up to 2.27x
Financial Services	BlackScholes SP Monte Carlo SP	Up to 7x Up to 10.75x
Sinopec	Seismic Imaging	Up to 2.53x[2]
Sandia Labs	miniFE (Finite Element Solver)	Up to 2x[3]
Intel Labs	Ray Tracing (incoherent rays)	Up to 1.88x[4]

* Xeon = Intel® Xeon® processor;
* Xeon Phi = Intel® Xeon Phi™ coprocessor

- Intel® Xeon Phi™ coprocessor accelerates highly parallel & vectorizable applications. (graph above)
- Table provides examples of such applications

Notes:

1. 2S Xeon* vs. 1 Xeon Phi* (preproduction HW/SW & Application running 100% on coprocessor unless otherwise noted)
2. 2S Xeon* vs. 2S Xeon* + 2 Xeon Phi* (offload)
3. 8 node cluster, each node with 2S Xeon* (comparison is cluster performance with and without 1 Xeon Phi* per node) (Hetero)
4. Intel Measured Oct. 2012

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.

Source: Customer Measured results as of October 22, 2012 Configuration Details: Please reference slide speaker notes.

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

Future Intel® MIC Architecture Processors

Future Intel MIC Implementations – AVX-512

- Intel® Advanced Vector Extensions 512 (**Intel® AVX-512**) instructions announced in July
- The latest Intel® Architecture Instruction Set Extensions Programming Reference includes the definition of Intel® Advanced Vector Extensions 512 (Intel® AVX-512) instructions.
  - **512-bit SIMD support** - pack eight double precision or sixteen single precision floating-point numbers, or eight 64-bit integers, or sixteen 32-bit integers within the 512-bit vectors
  - Enables processing of twice the number of data elements that AVX/AVX2 can process with a single instruction and four times that of SSE.
- Intel AVX-512 will be first implemented in the future Intel® Xeon Phi™ processor and coprocessor known by the code name **Knights Landing**

Intel® Xeon® and Intel® Xeon Phi™ in the Top 500 Lists

HPC Top 500 List, Jun 2013

http://www.top500.org/

Rank	Name	Site	Rmax (PF)	Efficiency (%)	Processor	Accelerator/ Co-Processor
1	Tianhe-2 (MilkyWay-2)	National University of Defense Technology	33.86	61.68	Intel Xeon E5-2692 12C 2.200GHz	Intel Xeon Phi 31S1P
6	Stampede	Texas Advanced Computing Center/Univ. of Texas	5.17	60.66	Xeon E5-2680 8C 2.700GHz	Intel Xeon Phi SE10P
28	Conte	Purdue University	0.94	70.34	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi 5110P
65	Discover	NASA Center for Climate Simulation	0.42	66.35	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi 5110P
69	PARAM Yuva - II	Center for Development of Advanced Computing (C-DAC)	0.39	73.05	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi 5110P
71	Endeavor	Intel	0.38	75.55	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi
72	MVS-10P	Joint Supercomputer Center	0.38	71.76	Xeon E5-2690 8C 2.900GHz	Intel Xeon Phi SE10X
148	Maia	NASA/Ames Research Center/NAS	0.21	70.64	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi
249	RSC Tornado SUSU	South Ural State University	0.15	61.99	Xeon X5680 6C 3.330GHz	Intel Xeon Phi SE10X
397	Beacon	National Institute for Computational Sciences/Univ. of Tennessee	0.11	70.14	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi 5110P
399		University of Miami	0.11	74.87	Xeon E5-2670 8C 2.600GHz	Intel Xeon Phi

An Intel® Xeon® and Intel® Xeon Phi™ system is #3 on the Top Green500 List (green500.org)

Green500 Rank, Oct 2013	Performance/ Power Unit (GFLOPS/W)	Site	Computer	Total Power (kW)
3	2.50	NICS/Univ. of Tennessee	Intel Xeon proc. E5-2670 8C 2.600GHz, Infiniband* FDR, Intel Xeon Phi 5110P	44.89

Intel MKL derived MP Linpack #1 on the Top500

Where to Find More Information

References

- **Intel® Developer Zone: Intel® Xeon Phi™ Coprocessor,** http://software.intel.com/en-us/mic-developer
- Intel® Xeon Phi™ Coprocessor System Software Developers Guide - http://download-software.intel.com/sites/default/files/article/334766/intel-xeon-phi-systemssoftwaredevelopersguide_0.pdf
- "Consistency of Floating-Point Results using the Intel® Compiler - http://software.intel.com/en-us/articles/consistency-of-floating-point-results-using-the-intel-compiler/
- http://software.intel.com/en-us/articles/advanced-optimizations-for-intel-mic-architecture-low-precision-optimizations
- http://software.intel.com/sites/default/files/article/164389/fp-consistency-122712_1.pdf
- https://secure-software.intel.com/sites/default/files/article/326703/floating-point-differences-sept11.pdf
- http://software.intel.com/en-us/articles/run-to-run-reproducibility-of-floating-point-calculations-for-applications-on-intel-xeon
- http://software.intel.com/en-us/blogs/2013/avx-512-instructions

Legal Disclaimer & Optimization Notice

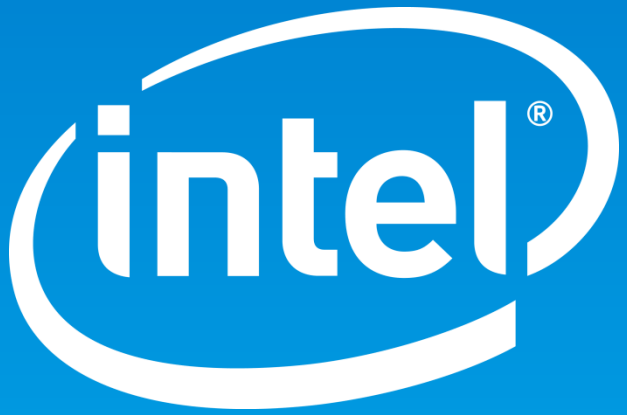
INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING 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.

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


intel