SIAM Conference on Computational Science and Engineering
February 27-March 3, 2017
Hilton Atlanta, Atlanta, Georgia, USA

OPTIMIZING HPC STENCIL PERFORMANCE ON THE INTEL® XEON PHI™ PROCESSOR

Chuck Yount, Intel Corporation, chuck.yount@intel.com
with data contributed by

- Alexander Breuer & Josh Tobin, Univ. of CA, San Diego
- Alex Duran, Intel Corporation Iberia, Spain

MS84: Domain-Specific Abstractions for Full-Waveform Inversion
SIAM-CSE, February 27, 2017

Outline

Intel® Xeon Phi™ Processor

- Multi-core package
- New instruction-set architecture
- High-bandwidth on-package memory

Tuning Stencil Code for the Xeon Phi

- Overview of techniques
- Framework for rapid prototyping and tuning

Resources

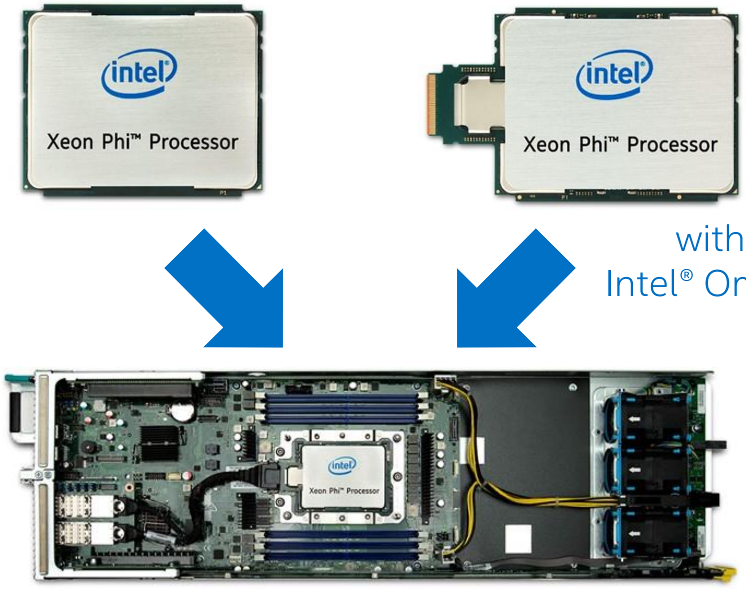
Summary

Intel® Xeon Phi™ Processor

Intel® Xeon Phi™ Product Family x200

(previously code-named Knights Landing, “KNL”)

Intel® Xeon Phi™ Processor

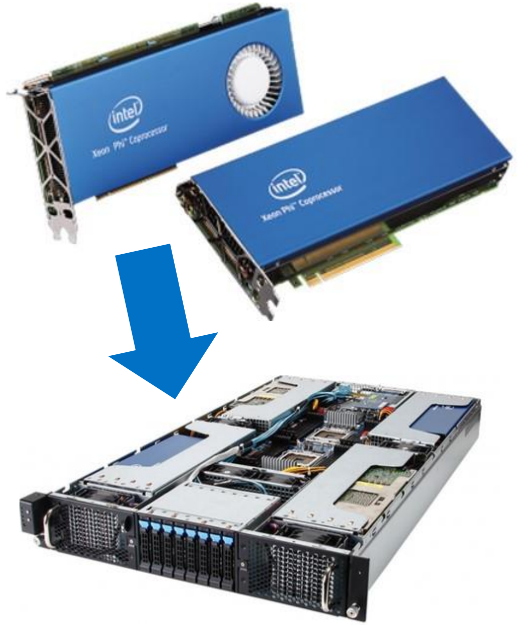


with integrated
Intel® Omni-Path Fabric

Host Processor in Groveport Platform

Self-boot Intel® Xeon Phi™ processor

Intel® Xeon Phi™ Coprocessor x200



Ingredient of Grantley & Purley Platforms

Requires Intel® Xeon® processor host

Intel® Xeon Phi™ Top500 Listings – Nov 2016

Intel Xeon Phi Processors (Knights Landing) now power 5 systems in the top 50:

#5	**Cori** (NERSC, USA); Cray XC – 14 PFLOPS
#6	**Oakforest PACS** (JCAHPC, Japan); Fujitsu CX1640 M1 – 13.5 PFLOPS
#12	**Marconi** (CINECA, Italy); Lenovo – 6.2 PFLOPS
#18	**Theta** (Argonne National Lab, USA); Cray XC40 – 5.1 PFLOPS
#33	**Camphor 2** (ACCMS, Kyoto University, Japan); Cray XC40 – 3.1 PFLOPS

KNL Architecture Overview

ISA

Intel® Xeon® Processor Binary-Compatible (w/Broadwell)

On-package memory

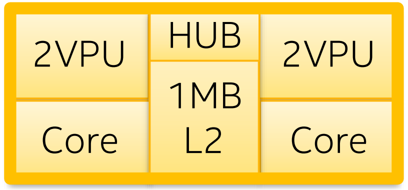
16GiB MCDRAM, ~490 GB/s Stream Triad

Platform Memory

Up to 384GiB DDR4-2400, ~90 GB/s Stream Triad

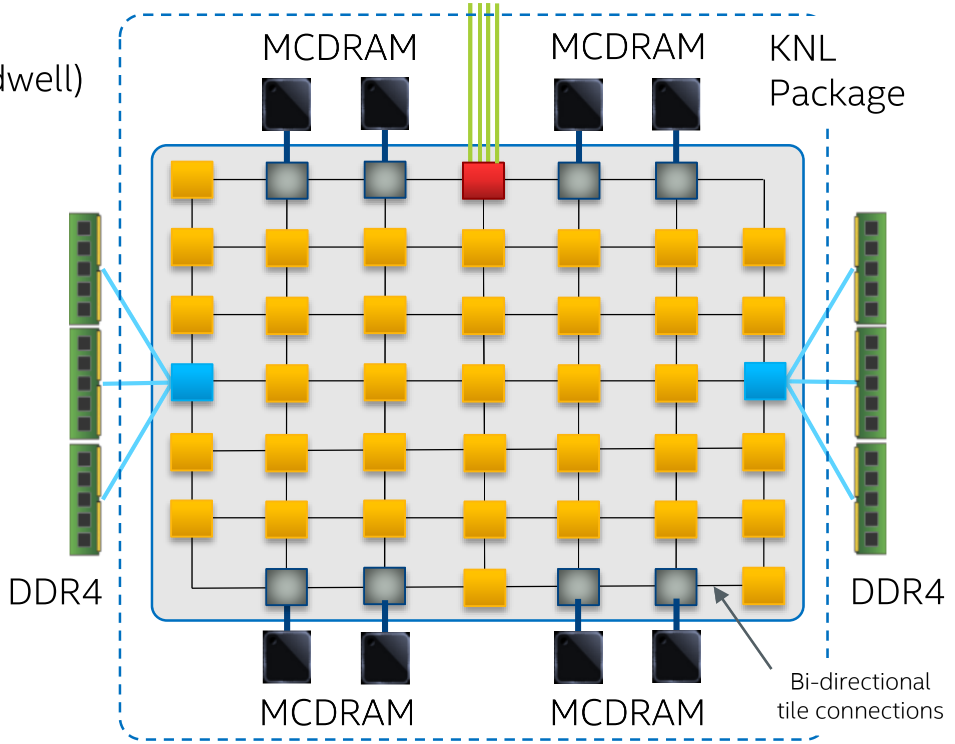
- ✓ 2D Mesh Architecture
- ✓ Out-of-Order Cores
- ✓ 3X single-thread vs. KNC

TILE: (up to 36)



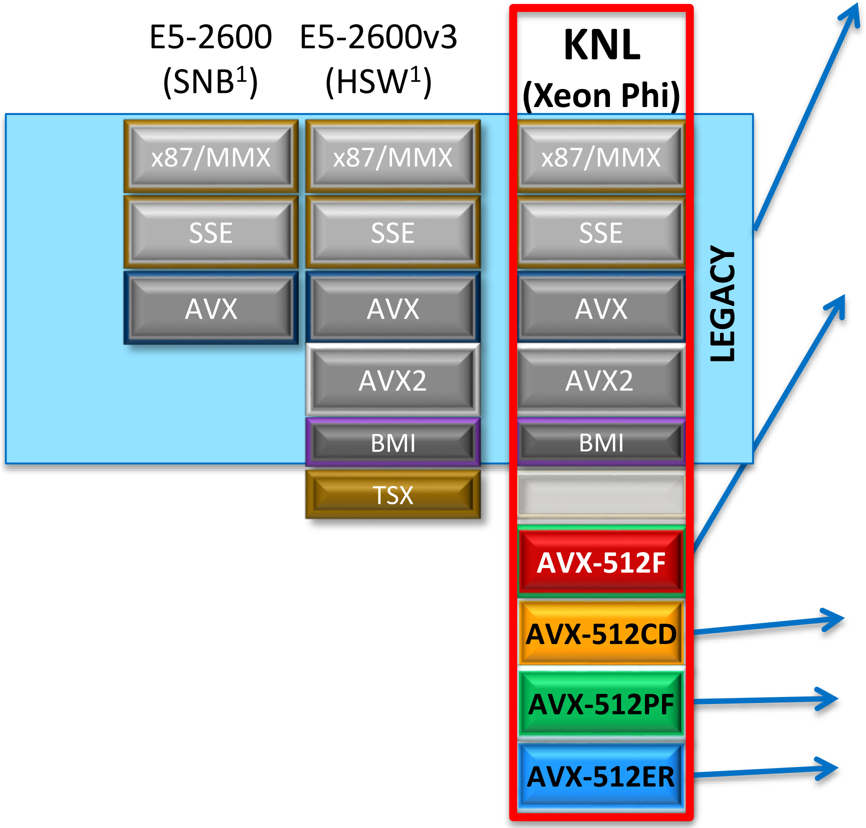
Enhanced Intel® Atom™ cores based on Silvermont Microarchitecture

Up to 72 cores with 4 hyper-threads each



🟧 Tile ▨ EDC (embedded DRAM controller) 🟦 IMC (integrated memory controller) 🟥 IIO (integrated I/O controller)

Instruction-Set



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 SIMD registers & 8 mask registers
- Gather/Scatter

Conflict Detection: Improves Vectorization

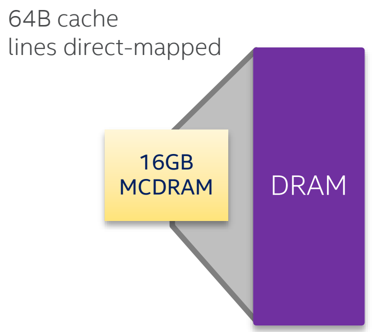
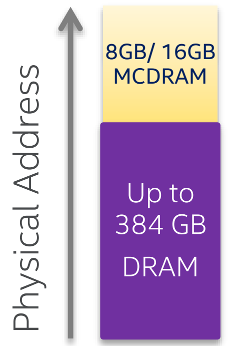
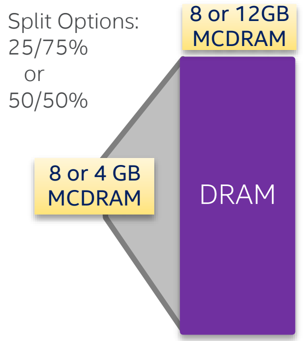
Prefetch: Gather and Scatter Prefetch

Exponential and Reciprocal Instructions

1. Previous code-names of Intel® Xeon® processors

Integrated On-Package Memory Usage Models

Model configurable at boot time and software exposed through NUMA[1]

	Cache Model Ideal for large data size (>16GB) cache blocking apps	**Flat Model** Maximum bandwidth for data reuse aware apps	**Hybrid Model** Maximum flexibility for varied workloads
			 Split Options: 25/75% or 50/50% 8 or 12GB MCDRAM 8 or 4 GB MCDRAM DRAM
Description	Hardware automatically manages the MCDRAM as a "L3 cache" between CPU and DDR memory	Manually manage how the app uses the integrated on-package memory and DDR for peak perf	Harness the benefits of both Cache and Flat models by segmenting the integrated on-package memory
Usage Model	▪ App and/or data set is very large and will not fit into MCDRAM ▪ Unknown or unstructured memory access behavior	▪ App or portion of an app or data set that can be "locked" into MCDRAM so it doesn't get flushed out	▪ Need to "lock" in a relatively small portion of an app or data set via the Flat model ▪ Remaining MCDRAM is configured as Cache

Feb. 27, 2017

1. NUMA = non-uniform memory access

STENCIL-CODE MODERNIZATION FOR THE INTEL® XEON PHI™ PROCESSOR

Stencil Computation

- Iterative kernels that update elements in one or more N-dimensional tensors using a fixed pattern of neighboring elements
- Fundamental algorithm in many HPC algorithms and scientific simulations, commonly used for solving differential equations using finite-difference methods (FDM)

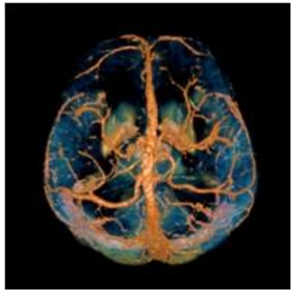
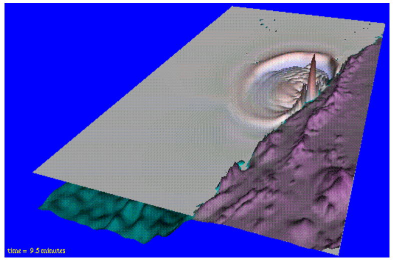
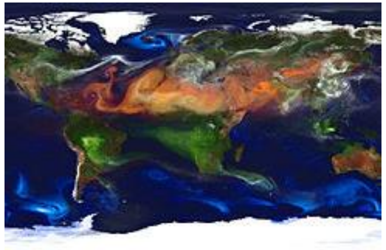


Image Processing



Seismic Modeling


Weather Simulation

Images from https://commons.wikimedia.org

Today’s Code Investment Carries Forward

To carry forward most key code modernizations

RECOMPILE

Parallelization, threading, vectorization, cache-blocking, MPI+OpenMP hybridization…

MKL | MPI | OpenMP | TBB

Knights Landing Enabled Performance Libraries & Runtimes

Intel® AVX-512

Cache Mode For High Bandwidth Memory

Knights Landing Enabled Compilers

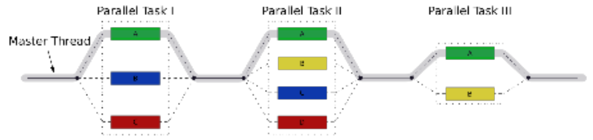
For additional gains

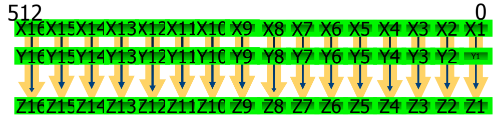
TUNE

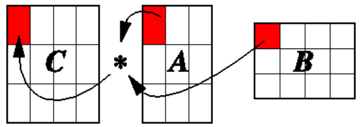
Exploit NEW features…

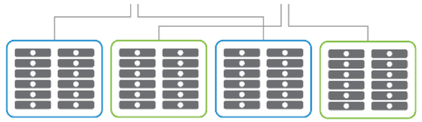
KNL Enhancements (memory, architecture, bandwidth, etc.)

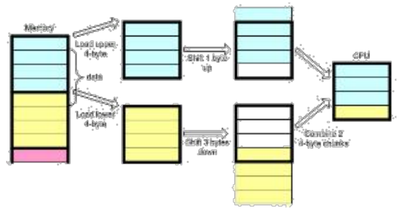
What is “Modernized” Code?




512
0
X16 X15 X14 X13 X12 X11 X10 X9 X8 X7 X6 X5 X4 X3 X2 X1
Y16 Y15 Y14 Y13 Y12 Y11 Y10 Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1
Z16 Z15 Z14 Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1

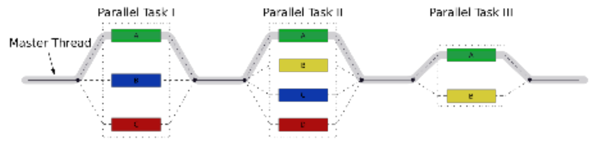






	What	Defined	Tools of the trade
1	**Thread Scaling**	Increase concurrent thread use across coherent shared memory	OpenMP, TBB, Cilk Plus
2	**Vector Scaling**	Use wide-vector (512-bit) instructions	Vector loops, vector functions, array notations
3	**Cache Blocking**	Use algorithms to reduce memory bandwidth pressure and improve cache hit rate	Blocking algorithms
4	**Fabric Scaling**	Tune workload to increased node count	MPI
5	**Data Layout**	Optimize data layout for unconstrained performance	AoS→SoA, directives for alignment

Thread Scaling for Stencils



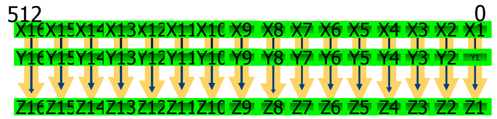
Algorithm characteristics

- Threading stencils is often straight-forward within a single grid and time-step
  - Many stencils update elements in one grid at a given time-step based only on elements in other grids or the same grid at previous time-steps
  - Some updates across multiple grids may be independent within a time-step
  - In these cases, all elements can be updated in parallel trivially
- Threading across multiple dependent grids and time-steps is more challenging
  - Requires more complex synchronization to observe dependencies

Techniques

- Example techniques to implement dependent threading include temporal wave-fronts and diamond tiling
- Threading software
  - Older code tends to use multiple single-threaded MPI tasks even on a single node
  - Often does not scale well to many threads available on KNL socket (up to 288)
  - More modern code uses OpenMP or MPI+OpenMP or MPI w/shared memory on a node
  - More advanced threading may include task scheduling to avoid global synchronization

Vector Scaling for Stencils


512
0
X16 X15 X14 X13 X12 X11 X10 X9 X8 X7 X6 X5 X4 X3 X2 X1
Y16 Y15 Y14 Y13 Y12 Y11 Y10 Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1
Z16 Z15 Z14 Z13 Z12 Z11 Z10 Z9 Z8 Z7 Z6 Z5 Z4 Z3 Z2 Z1

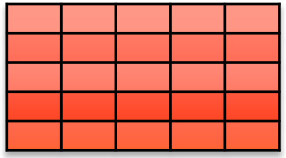
Algorithm characteristics

- The nature of “stencils” is application of a fixed pattern to multiple elements
- Elements within a grid are usually independent as discussed earlier
- Thus, SIMD vectorization can be applied in a straight-forward manner

Techniques

- Straight-forward vectorization along one dimension often results in many cache-line reads, many unused elements, and low reuse between vectors
- “Vector-folding” is a technique to vectorize in two or more dimensions, increasing reuse and thus decreasing memory-bandwidth requirements
  - Speed-ups of >1.5x have been observed in several real-world stencils
  - See HPCC’15 paper “Vector Folding: improving stencil performance via multi-dimensional SIMD-vector representation”

Cache Blocking for Stencils



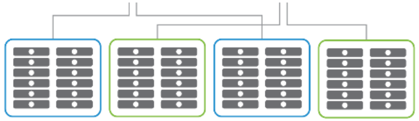
Algorithm characteristics

- Stencil codes are very often memory-bound
- Stencil equations typically input multiple neighboring values (increasing with accuracy)
- These factors make cache-blocking critical for high performance

Techniques

- Most cache-blocking is implemented via simple loop-tiling with each OpenMP thread working on separate tiles
- Advanced techniques leverage sharing of KNL caches between threads
  - Each L1 data cache is shared by 4 hyper-threads in a core
  - Each L2 cache is shared by 2 cores in a tile
  - Tiles can be sub-divided into slabs or similar partitions, and threads that share these caches can cooperate within them, increasing reuse and decreasing evictions
- To leverage the MCDRAM cache shared by all cores, an addition level of tiling may be used
  - See PMBS'16 paper "Effective Use of Large High-Bandwidth Memory Caches in HPC Stencil Computation via Temporal Wave-Front Tiling"
- In addition, prefetching data into L1 and/or L2 cache may improve performance

Fabric Scaling for Stencils



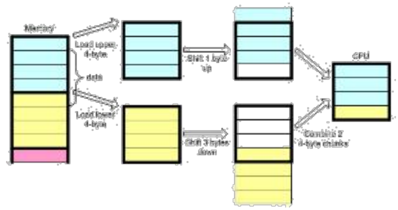
Algorithm characteristics

- As with threading and SIMD, independence of solutions within a time-step facilitate partitioning across nodes
- Access to multiple neighboring values that are common across partitions requires synchronizing data

Techniques

- Use of “halo” or “ghost” regions is the most common solution to reduce communications within a time-step as shared data is accessed
  - Halos must be exchanged between nodes to keep data consistent
  - Application of tiling across time-steps requires more sophisticated exchanges, usually consisting of exchanging more data but less often
- MPI is the most common software used, but other alternatives are in the works
- Global synchronization can cause under-utilization of nodes on large clusters and/or on clusters with nodes of heterogeneous performance

Data Layout for Stencils



Algorithm characteristics

- Many problems consist of multi-dimensional domains across multiple grids, which could be implemented naïvely with a multi-dimensional array-of-structures (AoS)
- Access to many neighboring elements and/or grids may cause translation look-aside buffer (TLB) pressure when multiple pages are accesses

Techniques

- Structure-of-arrays layout (SoA) is typical for multi-grid problems to enable vectorization
- Options to reduce TLB pressure
  - Using huge pages, e.g., via transparent huge pages (THP) in Linux
  - Reordering index nesting in grids, e.g., TXYZ → XYTZ
  - Tiling the layout itself
- To benefit from vector-folding discussed earlier, a data-layout transformation to a grid of folded vectors is essential

Y.A.S.K. – Yet Another Stencil Kernel

A *framework* to implement and tune stencil code for Intel® Xeon® processors and Intel® Xeon Phi™ processors and coprocessors

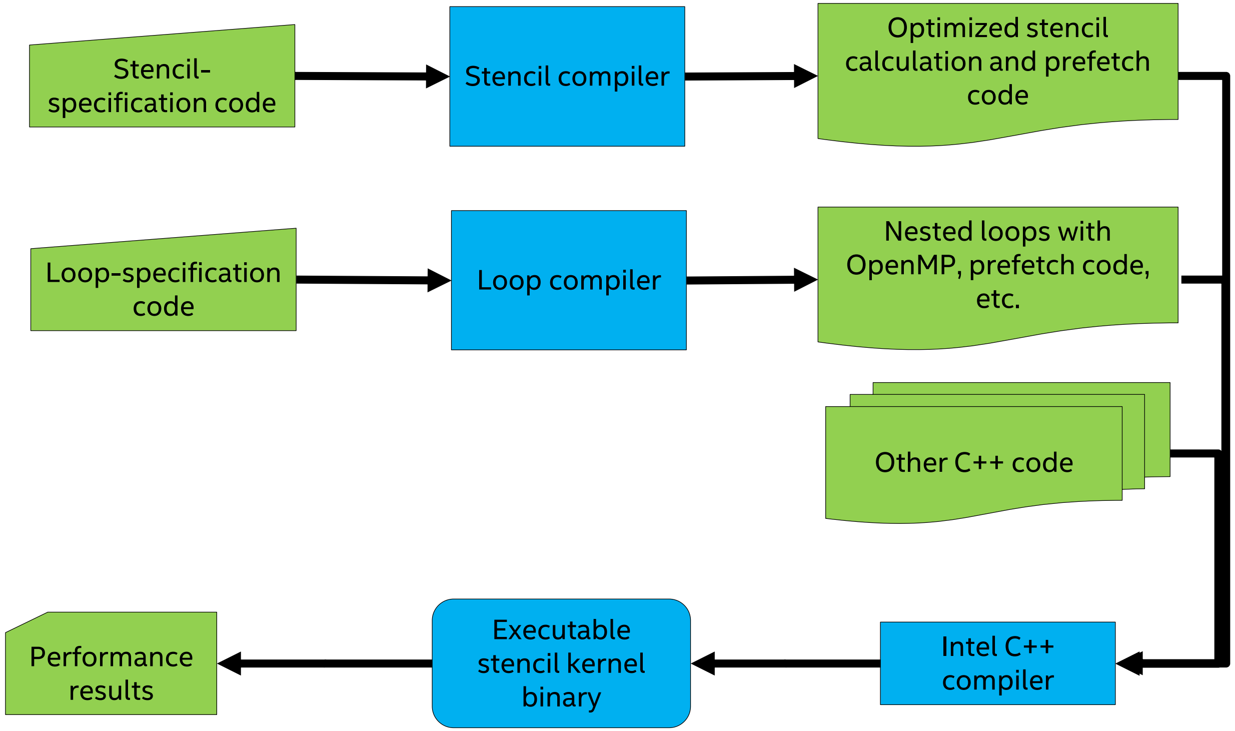
Original impetus

- Tool to evaluate the benefits of vector-folding on multiple stencil kernels
- Expanded to include most of the optimizations described earlier

Goals

- Create high-performing kernel code from a straightforward specification of stencil equations
- Provide a simple kernel-driver to test stencil performance
  - Expose [most] optimization trade-off choices without requiring code changes
  - Automate searching through the optimization design space
- Provide ability to integrate code into larger applications

High-Level Flow


Stencil-specification code
Stencil compiler
Optimized stencil calculation and prefetch code
Loop-specification code
Loop compiler
Nested loops with OpenMP, prefetch code, etc.
Other C++ code
Intel C++ compiler
Executable stencil kernel binary
Performance results

Example 1: Iso3DFD Stencil

$$
\begin{aligned}
p(t+1,i,j,k) \leftarrow\ & 2p(t,i,j,k) - p(t-1,i,j,k) + \\
& v(i,j,k)\Bigg(c_0 p(t,i,j,k) + \\
& \sum_{r=1}^{8} c_r \Big[p(t,i-r,j,k) + p(t,i+r,j,k) + \\
& p(t,i,j-r,k) + p(t,i,j+r,k) + \\
& p(t,i,j,k-r) + p(t,i,j,k+r)\Big]\Bigg)
\end{aligned}
$$

- 51-point stencil
- 16th order accurate in space, 2nd order in time
- 61 FP ops

YASK Input Specification for Iso3DFD Stencil

```
#include "StencilBase.hpp"
class Iso3dfdStencil : public StencilRadiusBase {
protected:
    Grid pressure;    // time-varying 3D pressure grid.
    Grid vel;         // constant 3D vel grid.
    Param coeff;      // stencil coefficients.
public:
    Iso3dfdStencil(StencilList& stencils, int radius=8) :
      StencilRadiusBase("iso3dfd", stencils, radius) {
        INIT_GRID_4D(pressure, t, x, y, z);
        INIT_GRID_3D(vel, x, y, z);
        INIT_PARAM_1D(coeff, r, radius + 1);   }
```

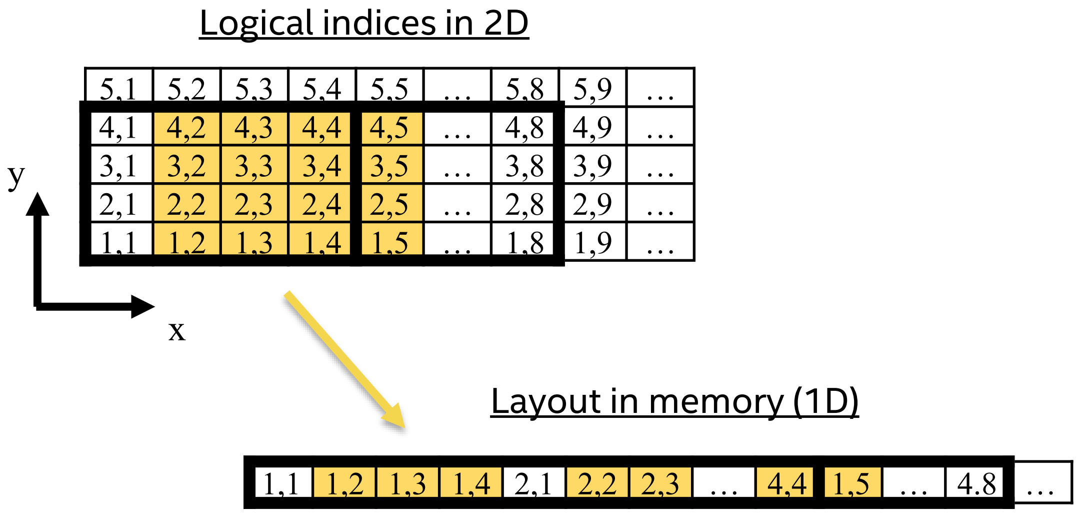
Declare grids and coefficient array

```
    virtual void define(const IntTuple& offsets) {
        GET_OFFSET(t); GET_OFFSET(x); GET_OFFSET(y); GET_OFFSET(z);
        GridValue np = pressure(t, x, y, z) * coeff(0);
        for (int r = 1; r <= _radius; r++) {
          np += coeff(r) *
            (pressure(t, x-r, y, z) + pressure(t, x+r, y, z) +
             pressure(t, x, y-r, z) + pressure(t, x, y+r, z) +
             pressure(t, x, y, z-r) + pressure(t, x, y, z+r));   }
        np = (2.0 * pressure(t, x, y, z))
          - pressure(t-1, x, y, z)  // subtract pressure at t-1.
          + (np * vel(x, y, z));    // add velocity term.
        pressure(t+1, x, y, z) IS_EQUIV_TO v;
    }
};
```

Define equation for pressure at t+1

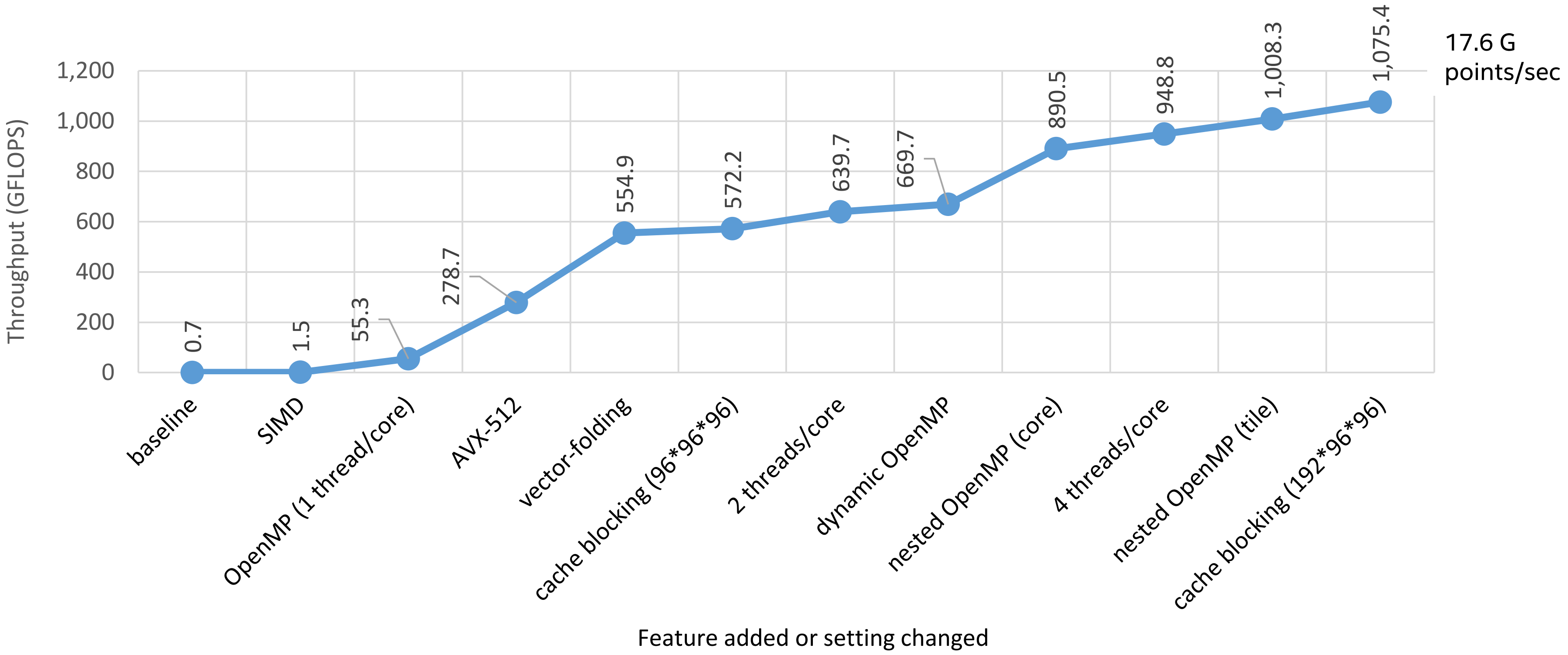
Example Stencil-Compiler Feature: Automatic Vector Folding

Example: 2D "4x4" vector folding



- Unaligned 4×4 vector (1,2 … 4,5) is shaded
- To read, two aligned vectors (1,1 … 4,4 and 1,5 … 4,8) are loaded, then 12 elements from the first and 4 from the second are assembled into a SIMD register using a permute instruction

Example Optimizations Applied to Iso3DFD



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 complete information visit **www.intel.com/benchmarks**. **Intel measurements** as of Oct., 2016 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4. See complete configuration details on "Configuration" slide.

Automatic Tuner

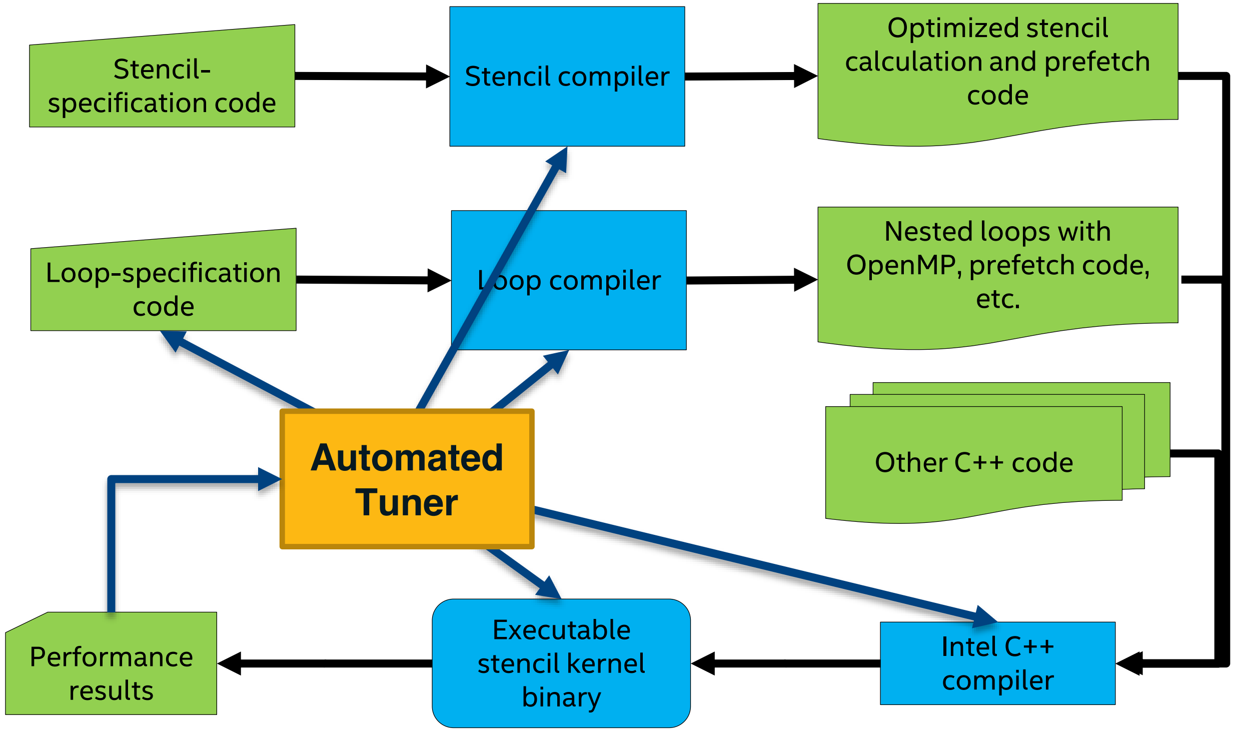
Challenge

- Dozens of possible optimization strategies
- Some of these can take hundreds of values (e.g., cache-block dimensions)
- Leads to combinatorial explosion in size of possible design space

Solution

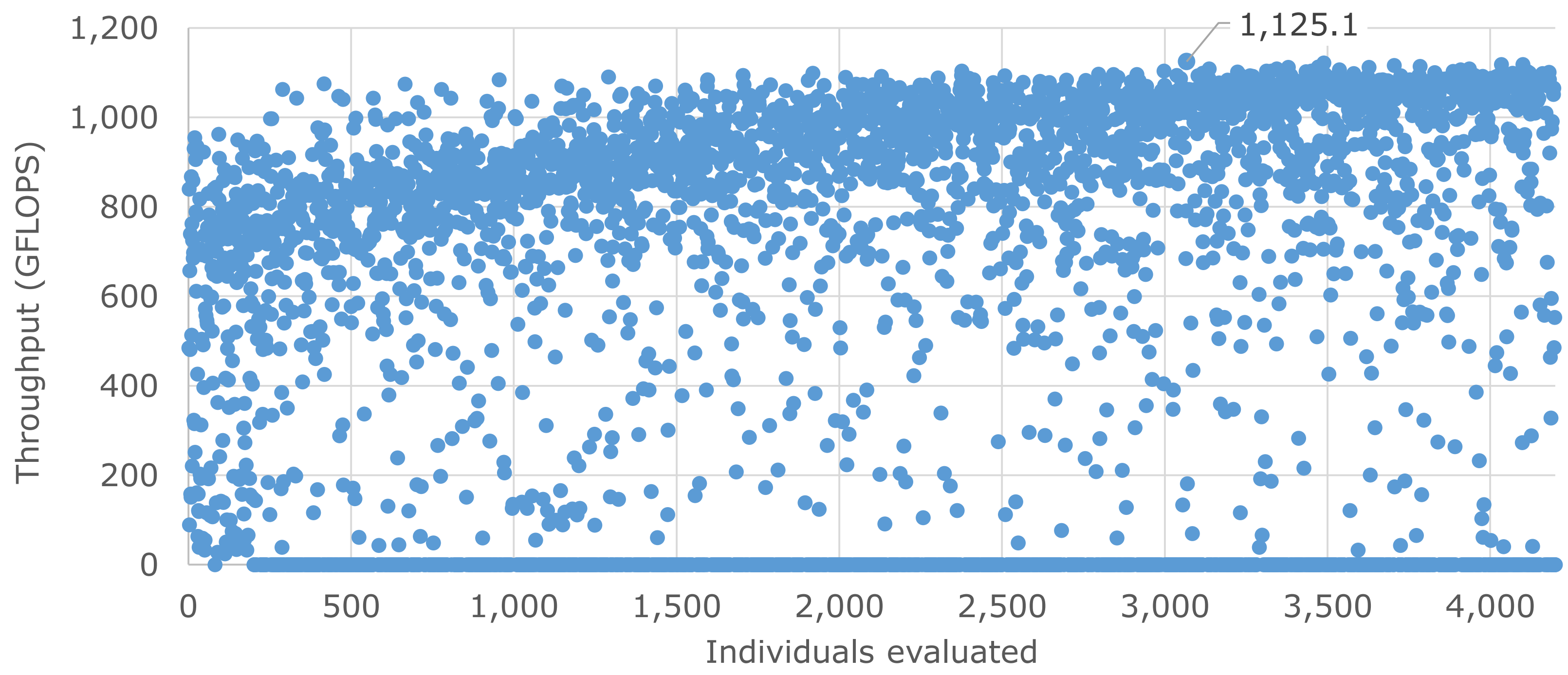
- Use genetic algorithm to select optimizations and tune parameters
- Tuner repeats the following sequence until convergence
  - Chooses optimization strategies and parameters based on random values (first generation) or mutation and crossover (subsequent generations)
  - Runs stencil compiler, loop compiler, C++ compiler, and kernel itself
  - Inputs resulting performance as fitness

High-Level Flow with Tuner



Stencil-specification code
Stencil compiler
Optimized stencil calculation and prefetch code
Loop-specification code
Loop compiler
Nested loops with OpenMP, prefetch code, etc.
Other C++ code
Automated Tuner
Performance results
Executable stencil kernel binary
Intel C++ compiler

Example Optimizations Applied with Automatic Tuner to Iso3dfd



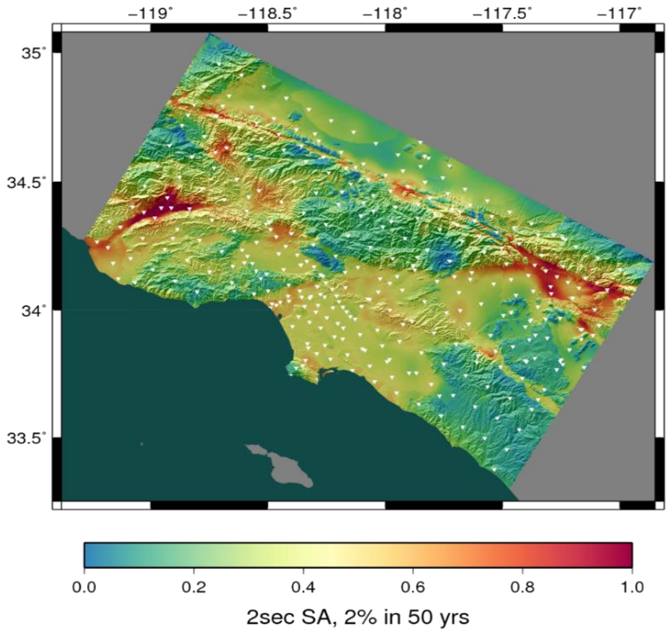
18.4 G points/sec

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 complete information visit www.intel.com/benchmarks. **Intel measurements** as of Oct., 2016 on Intel® Xeon Phi™ processor 7250 with 16 GiB MCDRAM, 96 GiB DDR4. See complete configuration details on "Configuration" slide.

Example 2: AWP-ODC-OS

AWP-ODC: Anelastic Wave Propagation-Olsen, Day, Cui

- Software that simulates seismic wave propagation after a fault rupture
- Widely used by the Southern California Earthquake Center (SCEC) community
- In recent years has primarily run on GPU accelerated supercomputers
- First ever open source release this year (BSD-2 license), including port to Intel Xeon Phi processor, under development by San Diego Supercomputer Center (SDSC) at Univ. of CA, San Diego (UCSD)



- CyberShake Study 15.4 hazard map for 336 sites around Southern California
- Warm colors represent areas of high hazard

Content on this slide courtesy of UCSD

AWP-ODC Numerics

Finite Difference code

- Staggered-grid scheme
- Fourth-order accurate in space and second-order accurate in time

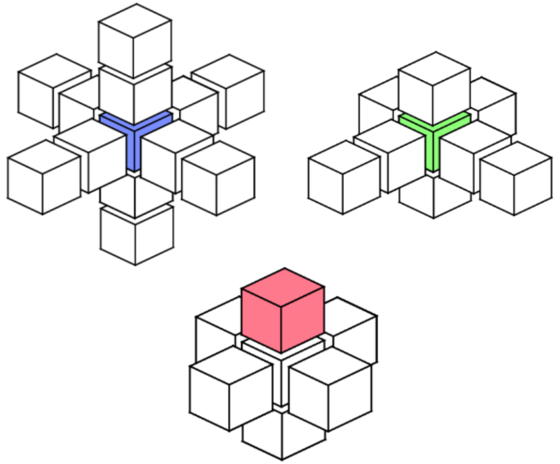
Fifteen grids updated in every time-step

- 3 velocity grids
- 6 stress grids
- 6 grids for auxiliary memory-variables required for accurate high-frequency simulation

Fifteen stencils

- Nine 13-point stencils
- Six 9-point stencils

Free-surface boundary computation every time-step



AWP-ODC stencils, starting from top left: velocity/stress update, memory variable stencil, boundary stencil

Preliminary AWC-ODC-OS performance

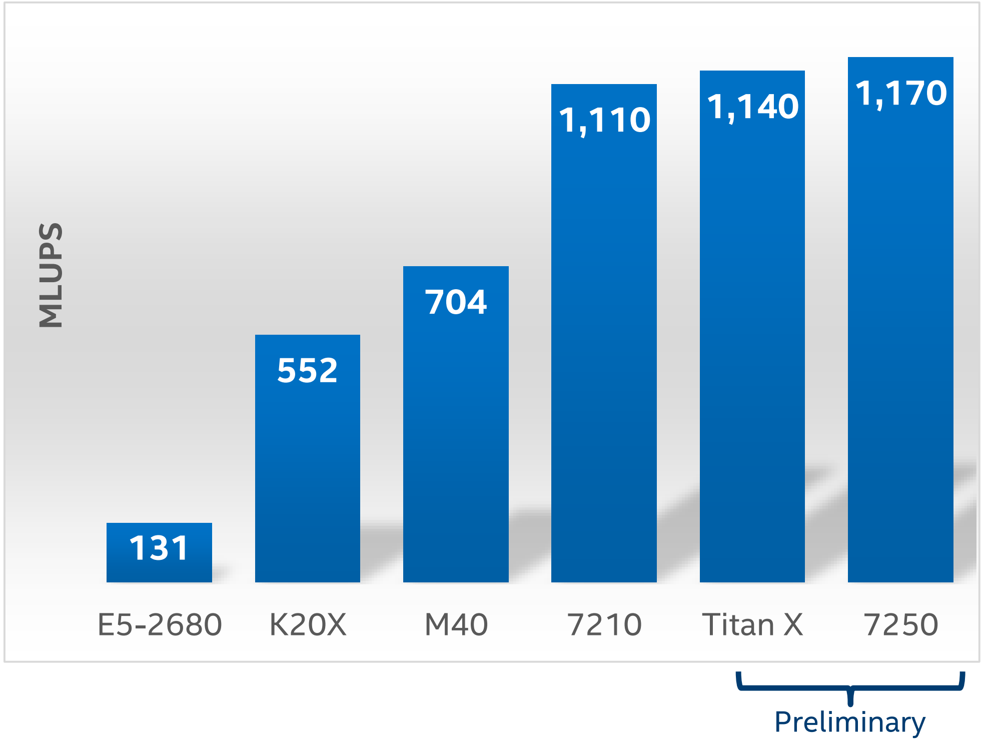
Numerics: Velocity + Frequency-dependent viscoelasticity and free-surface boundary conditions

Domain sizes chosen to use most of the available memory for each platform (MCDRAM for Intel Xeon Phi processors)

Performance measured in number of Lattice Update Points per Second, updating 15 grids

Compared time-to-solution per grid-point for:

- Single-socket Intel® Xeon® processor E5-2680 v3
- Intel Xeon Phi processor 7210 and 7250
- NVIDIA K20X, M40 and Titan X*



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 complete information visit **www.intel.com/benchmarks**. Measurements created by **UC San Diego San Diego Supercomputer Center (SDSC)** as of Oct., 2016. See complete configuration details on "Configuration" slide. *Other names and brands may be claimed as the property of others

RESOURCES

Worldwide Training

Colfax training with access to a 36-node cluster

Intel® Modern Code

Intel® Modern Code

Drive faster breakthroughs through faster code: Get more results on your hardware today and carry your code forward to the future.

Intel® Parallel Computing Centers

Intel® Parallel Computing Centers (Intel® PCC)

Universities, institutions, and labs that are leaders in their field, focusing on modernizing apps.

***FREE**...Worldwide Training and Teaching Resources*

Parallel Programming Reference Books

PARALLEL PROGRAMMING AND OPTIMIZATION WITH INTEL XEON PHI COPROCESSORS
High Performance Parallelism Pearls
Structured Parallel Programming
High Performance Parallelism Pearls
Intel® Xeon Phi™ Coprocessor High Performance Programming
INTEL XEON PHI PROCESSOR HIGH PERFORMANCE PROGRAMMING KNIGHTS LANDING EDITION
MK

Commercial ISVs Embracing Intel® Xeon Phi™ processor Family

Click on the images to learn more

IXPUG Community Forum

The Intel® Xeon Phi™ User's Group (IXPUG) is an independent global users group whose mission is to provide a forum for the free exchange of information that enhances the usability and efficiency of scientific and technical applications running on large High Performance Computing (HPC) systems using the Intel® Xeon Phi™ processor. IXPUG is administered by representatives of member sites that operate large Phi-based HPC systems.

- **IXPUG Monthly Tuning Meetings:** conference calls to inform the Intel® Xeon Phi™ processor community of relevant updates and share techniques, results, and methodologies.

YASK Resources

Software available

- YASK: https://github.com/01org/yask (MIT OS license) with several example stencils:
  - Simple symmetric 3D shapes on one grid
  - Simple heat-transfer a la miniGhost
  - Iso3DFD & AWP examples shown earlier (plus elastic version of AWP)
  - Standard-staggered grid (SSG)
  - Full-staggered grid (FSG), with and without absorbing boundary conditions
- AWP-ODC-OS: https://github.com/HPGeoC/awp-odc-os (BSD OS license)

Related collateral

- YASK article: https://software.intel.com/en-us/articles/recipe-building-and-running-yask-yet-another-stencil-kernel-on-intel-processors
- High Performance GeoComputing Lab: http://hpgeoc.sdsc.edu
- Source of AWP-ODC-OS information and data provided by UCSD: https://anl.app.box.com/v/IXPUG2016-presentation-13
- WOLFHPC'16 paper "YASK–Yet Another Stencil Kernel: a framework for HPC stencil code-generation and tuning"

Summary

Intel® Xeon Phi™ Processor (Knights Landing)

- Up to 72 cores of 4 hyper-threads each
- New AVX-512 instruction set architecture
- High-bandwidth on-package MCDRAM

Tuning Stencil Code for the Xeon Phi

- Thread scaling, vector scaling, cache blocking, fabric scaling, data layout
- YASK framework for rapid prototyping and tuning

Resources

- Training
- Experimental cluster
- Community forum

Legal Disclaimers

BACKUP

Experimental Configurations

Configuration details: YASK HPC Stencils, iso3DFD Kernel

Intel® Xeon Phi™ processor 7250: Intel® Xeon Phi™ processor 7250, 68 cores, 272 threads, 1400 MHz core freq. (Turbo On), MCDRAM 16 GiB, DDR4 96GiB 2400 MHz, quad cluster mode, MCDRAM flat memory mode, Red Hat* Enterprise Linux Server release 6.7

Configuration details: YASK HPC Stencils, AWP-ODC Kernel

Intel® Xeon® processor E5-2680 v3: Single Socket Intel® Xeon® processor E5-2680 v3, 2.5 GHz (Turbo Off) , 12 Cores, 12 Threads (HT off), DDR4 128GiB, CentOS* 6.7

Intel® Xeon Phi™ processor 7210: Intel® Xeon Phi™ processor 7210, 64 cores, 256 threads, 1300 MHz core freq. (Turbo On), MCDRAM 16 GiB, DDR4 96GiB 2133 MHz, quad cluster mode, MCDRAM flat memory mode, CentOS* 7.2

Intel® Xeon Phi™ processor 7250: Intel® Xeon Phi™ processor 7250, 68 cores, 272 threads, 1400 MHz core freq. (Turbo On), MCDRAM 16 GiB, DDR4 96GiB 2400 MHz, quad cluster mode, MCDRAM flat memory mode, CentOS* 7.2

NVIDIA Tesla* K20X (Kepler): Part number 900-22081-0030-000, 1x GK110 CPU, 2688 cores, 732 MHz core freq, 6GiB 2.6GHz GDDR5

NVIDIA M40 (Maxwell): Part number TCSM40M-PB, 3072 cores, 948 MHz base freq, 12 GiB GDDR5

NVIDIA Titan X (Pascal): 3072 cores, 1000 MHz base freq, 12 GiB GDDR5

Top500 Details (first 5)

Source: https://www.top500.org/statistics/sublist/

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
5	DOE/SC/LBNL/NERSC United States	**Cori** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	622,336	14,014.7	27,880.7	3,939
6	Joint Center for Advanced High Performance Computing Japan	**Oakforest-PACS** - PRIMERGY CX1640 M1, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Fujitsu	556,104	13,554.6	24,913.5	2,718.7
12	CINECA Italy	**Marconi Intel Xeon Phi** - CINECA Cluster, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Lenovo	241,808	6,223.0	10,833.0	
18	DOE/SC/Argonne National Laboratory United States	**Theta** - Cray XC40, Intel Xeon Phi 7230 64C 1.3GHz, Aries interconnect Cray Inc.	207,360	5,095.8	8,626.2	1,087
33	Academic Center for Computing and Media Studies (ACCMS), Kyoto University Japan	**Camphor 2** - Cray XC40, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray Inc.	122,400	3,057.3	5,483.5	748.1

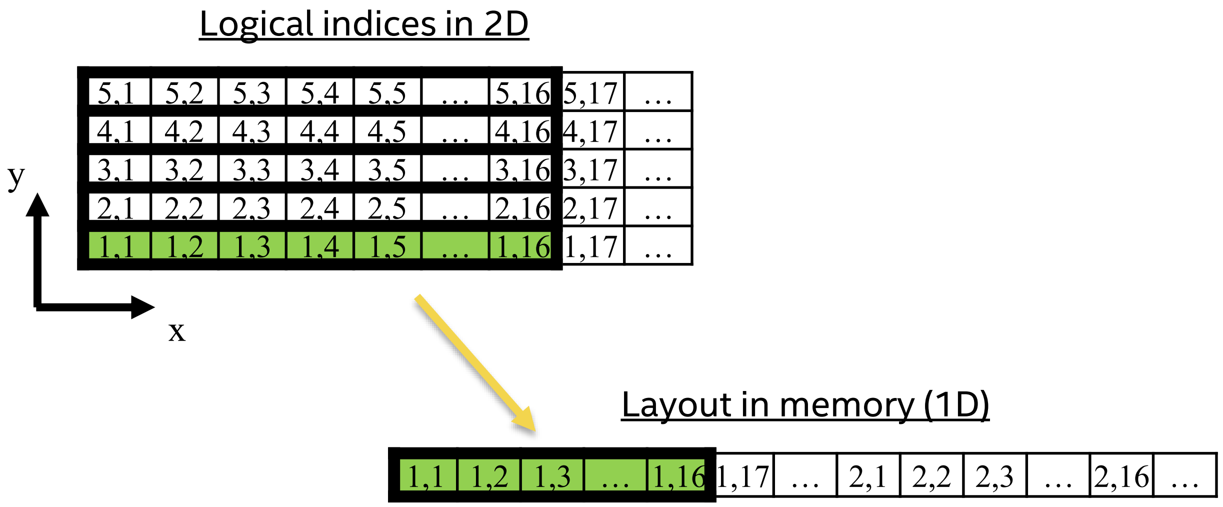
Top500 Details (next 5)

Source: https://www.top500.org/statistics/sublist/

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
106	MIT/Lincoln Laboratory United States	**TX-Green** - S7200AP Cluster, Intel Xeon Phi 7210 64C 1.3GHz, Intel Omni-Path Dell	41,472	1,032.8	1,725.2	
144	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede-KNL** - Intel S7200AP Cluster, Intel Xeon Phi 7250 68C 1.4GHz, Intel Omni-Path Dell / Intel	34,272	842.9	1,535.4	515.5
375	SFB/TR55 at Fujitsu Technology Solutions GmbH Germany	**QPACE3** - PRIMERGY CX1640 M1, Intel Xeon Phi 7210 64C 1.3GHz, Intel Omni-Path Fujitsu	18,432	447.1	766.8	77
397	Thomas Jefferson National Accelerator Facility United States	**SciPhi XVI** - KOI Cluster, Intel Xeon Phi 7230 64C 1.3GHz, Intel Omni-Path Koi Computers	16,896	425.9	702.9	111
456	Atos France	**Sequana_BXI** - Bull Sequana, Intel Xeon Phi 7250 68C 1.4GHz, Bull BXI 1.1 Bull, Atos Group	14,960	380.5	670.2	103

Example Stencil-Compiler Feature: Automatic Vector Folding

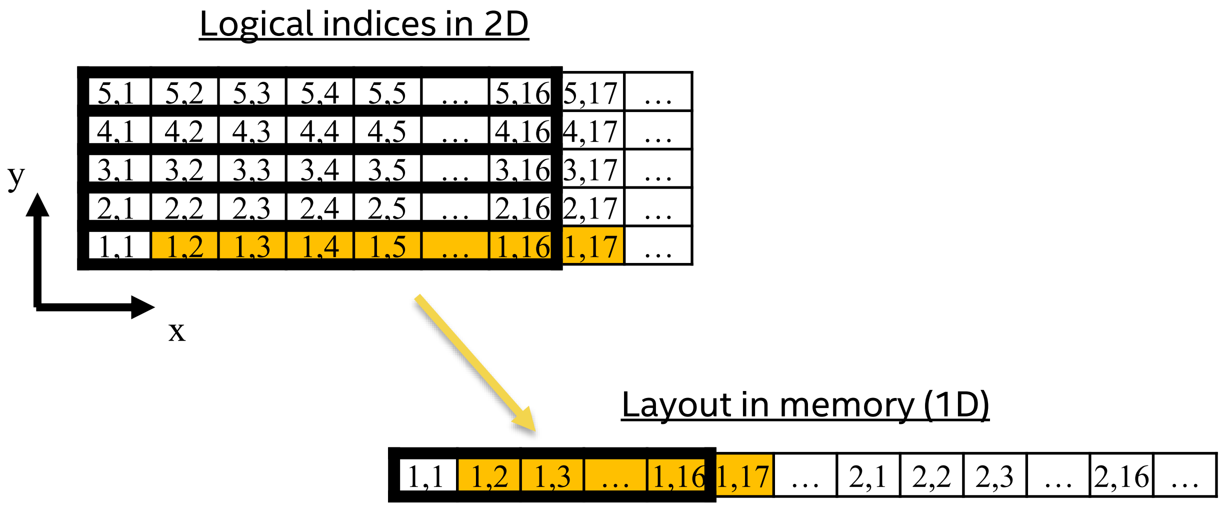
Background: Traditional 1D vectorization



- Traditional 1D vectorization layout (16×1)
- First <u>aligned</u> vector (1,1 … 1,16) is shaded
- Read with simple and efficient aligned vector load

Automatic Vector Folding

Background: Traditional 1D vectorization

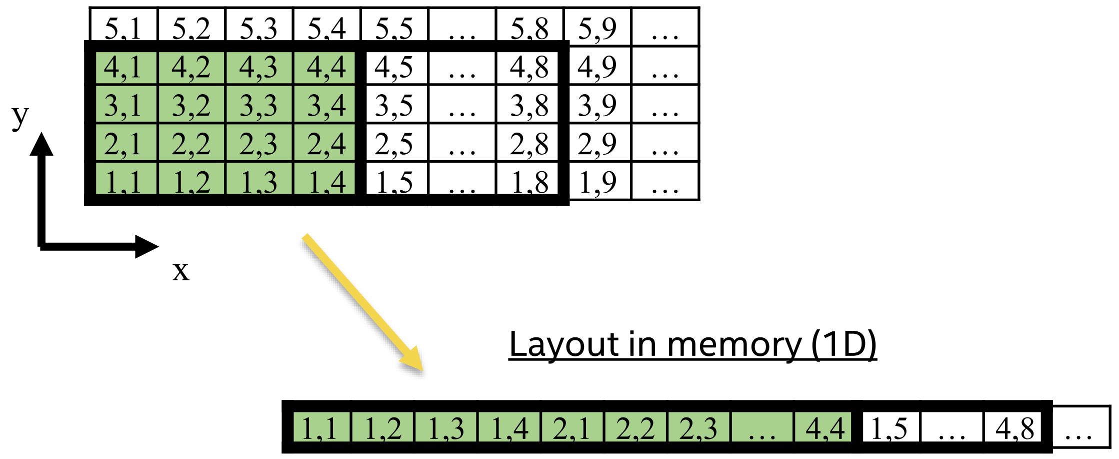


- Unaligned vector (1,2 … 1,17) is shaded
- Can use simple unaligned load or two aligned loads plus a simple shift instruction to create vector

Automatic Vector Folding

Example: 2D “4x4” vector folding

Logical indices in 2D



5,1	5,2	5,3	5,4	5,5	…	5,8	5,9	…
4,1	4,2	4,3	4,4	4,5	…	4,8	4,9	…
3,1	3,2	3,3	3,4	3,5	…	3,8	3,9	…
2,1	2,2	2,3	2,4	2,5	…	2,8	2,9	…
1,1	1,2	1,3	1,4	1,5	…	1,8	1,9	…

Layout in memory (1D)

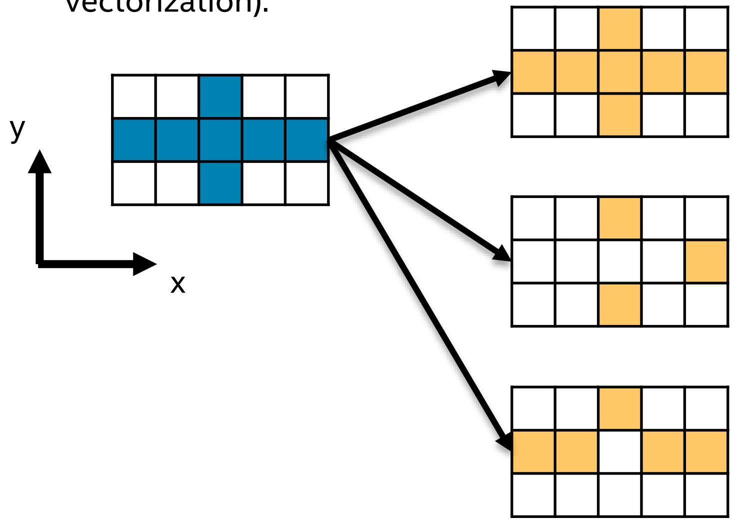
1,1	1,2	1,3	1,4	2,1	2,2	2,3	…	4,4	1,5	…	4,8	…

- 2D vector-folding layout (4×4)
- First aligned vector (1,1 … 4,4) is shaded
- Read with simple and efficient aligned vector load

Example Stencil-Compiler Feature: Automatic Prefetch Generation

This example stencil reads from 7 cache lines (after vectorization):

The stencil compiler generates the following prefetch functions:



Full prefetch function loads all 7 cache lines

X-direction prefetch function loads only these 3 leading cache lines

Y-direction prefetch function loads only these 5 leading cache lines