



Hotspot-Guided Optimization with Intel VTune Amplifier XE

The Hands-On Workshop (HOW) Series “Tools”

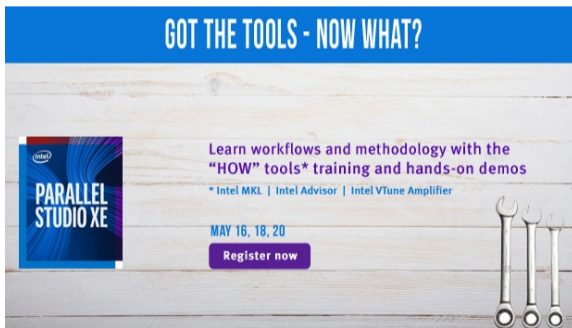
Andrey Vladimirov, PhD — Colfax International
colfaxresearch.com

Disclaimer

While best efforts have been used in preparing this training, Colfax International makes no representations or warranties of any kind and assumes no liabilities of any kind with respect to the accuracy or completeness of the contents and specifically disclaims any implied warranties of merchantability or fitness of use for a particular purpose. The publisher shall not be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused, or alleged to have been caused, directly or indirectly, by the information or programs contained herein. No warranty may be created or extended by sales representatives or written sales materials.

About the Series

Hands-On Workshop (HOW “Tools” Series): webinars on efficient programming for the Intel architecture with the help of dedicated software development tools



GOT THE TOOLS - NOW WHAT?

Learn workflows and methodology with the “HOW” tools* training and hands-on demos

* Intel MKL | Intel Advisor | Intel VTune Amplifier

MAY 16, 18, 20

[Register now](#)

colfaxresearch.com/how-tools-16-05

Learn More



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

STARTS MAY 23

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

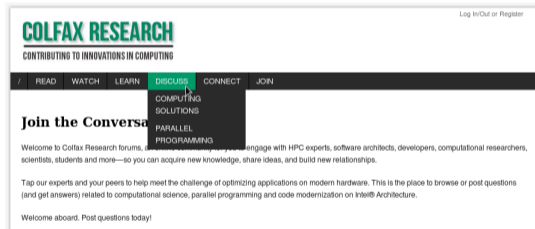
colfaxresearch.com/how-series

Get Your Questions Answered

Chat (for this course):
colfaxresearch.com/how-tools-16-05



Forums (general):
colfaxresearch.com/discussion



§2. Intel Architecture

Computing Platforms

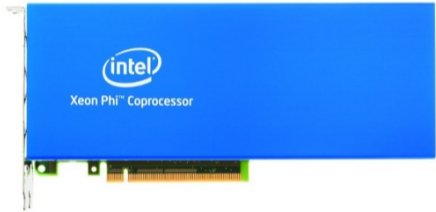
Intel Xeon Processor



Current: Broadwell
Upcoming: Skylake

Multi-Core Architecture

Intel Xeon Phi Coprocessor, 1st generation



Current: Knights Corner (KNC)

Intel Xeon Phi Processor, 2nd generation*



* socket and coprocessor versions

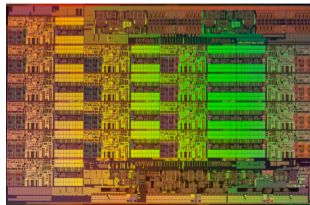
Upcoming: Knights Landing (KNL)

Intel Many Integrated Core (MIC) Architecture

Intel Xeon CPU: Purpose and Specifications

General-purpose platform for demanding computing applications.

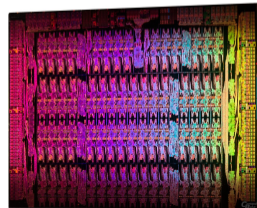
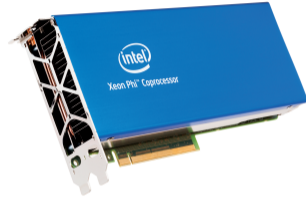
- Up to ~ 1 TFLOP/s in DP
- Up to ~ 2 TFLOP/s in SP
- Up to 768 GiB DDR4 RAM
- ~ 126 GB/s bandwidth
- Hardware-rich: forgiving of sub-optimal code



Intel Xeon Phi Processors (1st Gen)

Specialized platform for demanding computing applications.

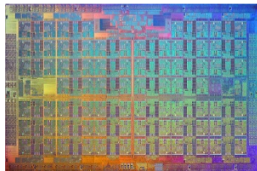
- PCIe end-point device
- ~ 1.2 TFLOP/s in DP
- ~ 2.4 TFLOP/s in SP
- Up to 16 GiB GDDR5 RAM
- ~ 176 GB/s bandwidth
- Heterogeneous clustering
- Runs special Linux distribution



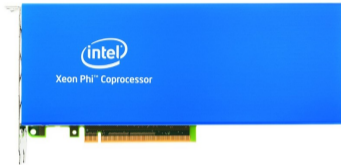
Intel Xeon Phi Processors (2nd Gen)

Specialized platform for demanding computing applications.

- Socket version or coprocessor
- 3+ TFLOP/s in DP
- 6+ TFLOP/s in SP
- Up to 16 GiB MCDRAM
- ≥ 400 GB/s MCDRAM bandwidth
- Up to 384 GiB DDR4 RAM
- ≥ 90 GB/s DDR4 bandwidth
- Supports common OS
- [Register for webinar](#)



“Standard Candle” Testbench



One Intel Xeon Phi 7120P
coprocessor (2012)
TDP: 300 W, RCP: \$4129

vs.



Two Intel Xeon E5-2697 v3
CPUs (2014)
TDP: 290 W, RCP: \$5404

See also [“Intel Xeon Product Family: Performance Brief”](#)

§3. Computation and Optimization

Optimization Methodology

Optimization Areas

- Scalar optimization
- Vectorization
- Multi-threading
- Memory access
- Communication

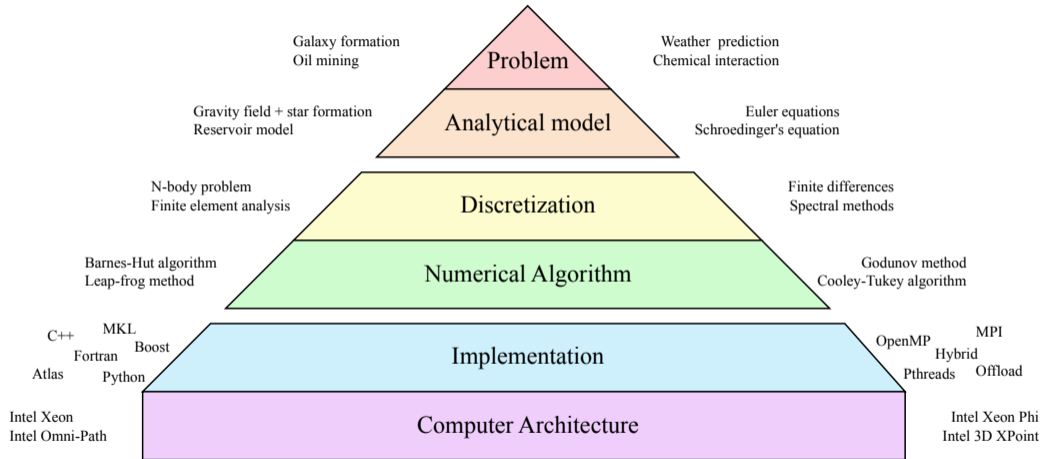
Details in [HOW Series](#) + [our book](#).

How to Navigate Them

- ① Focus on the algorithm
- ② Achieve parallel scalability
- ③ “Hotspot-guided optimization”:
iteratively use VTune +
optimization report

Performance and Algorithms

Computing in Science and Engineering



Algorithm Optimization Tips

- ❶ Instead of re-inventing the wheel, call standard library functions (e.g., instead of matrix-matrix multiplication with loops, use MKL)
- ❷ Identify bad asymptotic computational complexity (e.g., $O(N^3)$) and use improved algorithms
- ❸ Look for parallelism in the problem, expose it in the outer loops

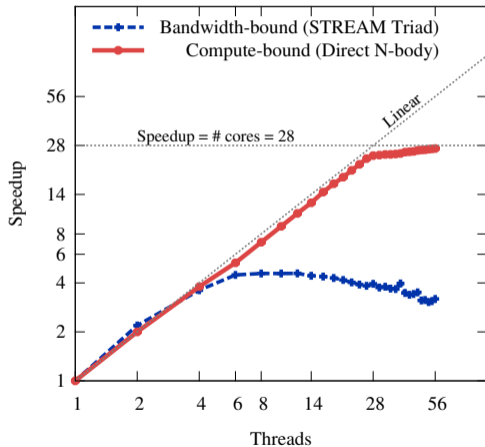
Scalability Tests

Study Thread Scalability with OpenMP

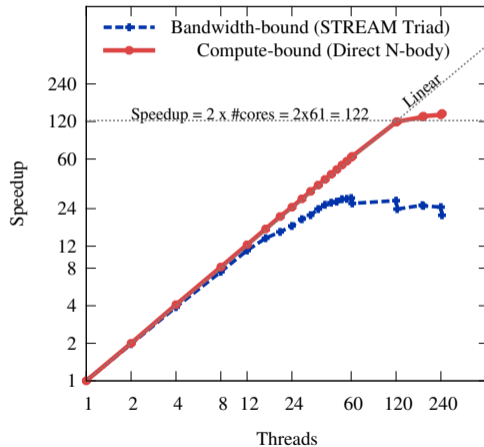
- 1 Set `KMP_AFFINITY=compact`
If hyper-threading is enabled, use `compact, 1` or `scatter`
 - 2 Vary `OMP_NUM_THREADS` from 1 to # of logical CPUs
 - 3 On Xeon Phi, use `KMP_PLACE_THREADS`
- Expectation depends on problem: compute- or bandwidth-bound
 - VTune to detect location of imbalance, synchronization, overhead

Expected Thread Scalability

Performance on the CPU architecture



Performance on the MIC architecture

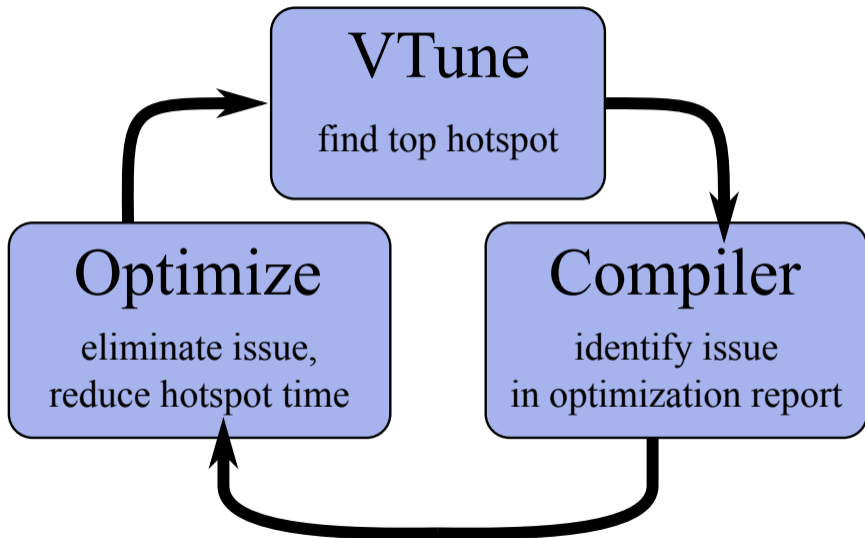


Study Vector Scalability

- 1 Compile code with `-no-vec -no-simd`
 - 2 Alternatively, use `#pragma novector` in code
- Speedup equal to vector width = good
 - Less than vector width = no vectorization or memory-bound
 - VTune to detect most important loops for vectorization

Hotspot-Guided Optimization

Hotspot-Guided Optimization



Using Intel VTune Amplifier XE

- 1 Compile code with `-g -O2` or `-g -O3`
 - 2 Set environment variables or use a wrapper script
 - 3 Tweak code input for a short representative run
- Advanced hotspots – good starting point
 - Focus on optimization report for detected hotspots

Clues in Optimization Report

- 1 Compile code with `-qopt-report`
- 2 For more verbosity, use up to `-qopt-report=5`

Things to look for:

- Failed vectorization
- Peeling in short vector loops
- Strided load/store on CPU = gather/scatter on MIC
- Type conversions
- Multiversioning in vectorized loops
- Suggestions for loop permutation

VTune to detect most important locations.

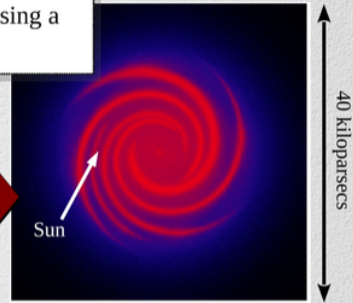
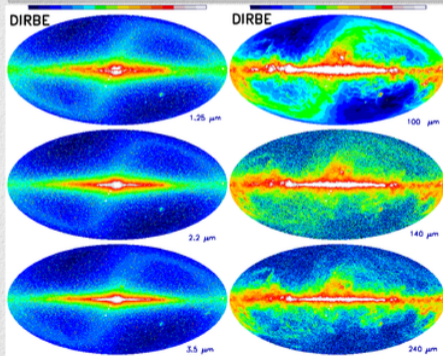
Optimization Report Example

```
LOOP BEGIN at TransientHeatingFunctionsXeonPhi.cc(162,4)
  remark #15389: vectorization support: reference bMatrix has unaligned access
  remark #15389: vectorization support: reference _M_data has unaligned access
  remark #15381: vectorization support: unaligned access used inside loop body
  remark #15305: vectorization support: vector length 8
  remark #15399: vectorization support: unroll factor set to 2
  remark #15309: vectorization support: normalized vectorization overhead 1.118
  remark #15417: vectorization support: number of FP up converts: single prec..
  remark #15300: LOOP WAS VECTORIZED
  remark #15442: entire loop may be executed in remainder
  remark #15450: unmasked unaligned unit stride loads: 2
  remark #15475: --- begin vector loop cost summary ---
  remark #15476: scalar loop cost: 10
  remark #15477: vector loop cost: 2.120
  remark #15478: estimated potential speedup: 3.890
  remark #15487: type converts: 1
  remark #15488: --- end vector loop cost summary ---
LOOP END
```

§4. Case Study: HEATCODE

Case Study: Building a 3D Model of the Milky Way Galaxy using 2D Sky Surveys

Goal: build a 3D model of the Milky Way Galaxy using a large volume of 2D data from sky surveys.



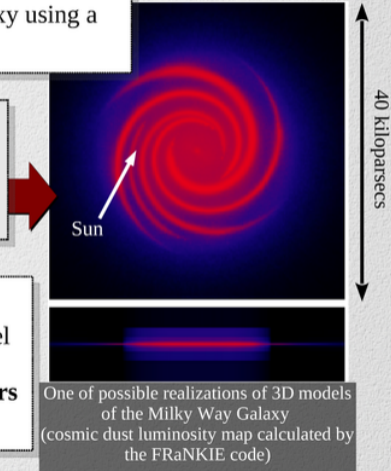
One of possible realizations of 3D models of the Milky Way Galaxy (cosmic dust luminosity map calculated by the FRaNKIE code)

Case Study: Building a 3D Model of the Milky Way Galaxy using 2D Sky Surveys

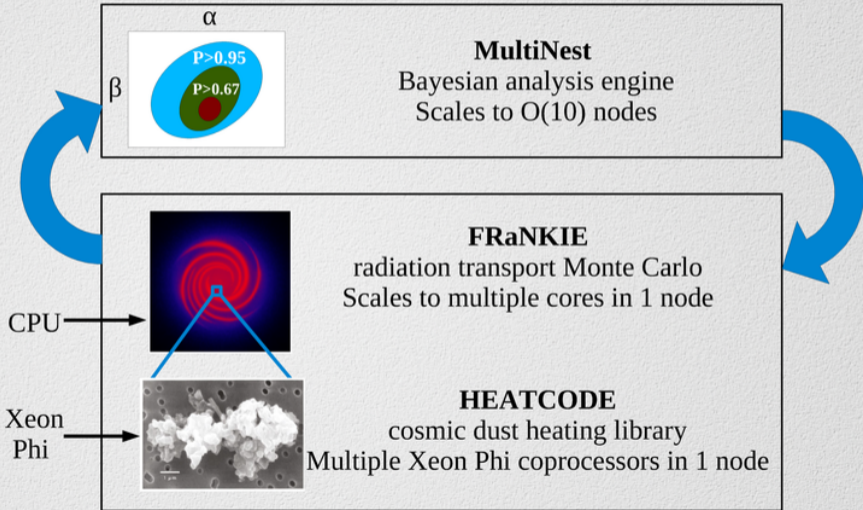
Goal: build a 3D model of the Milky Way Galaxy using a large volume of 2D data from sky surveys.

Method: Bayesian inference. Simulate the Galaxy, assess the fit to data, refine 3D model parameters, rinse & repeat.

Challenge: modeling the process of stochastic heating of cosmic dust by starlight, in each voxel of a 3D grid, is very time consuming. With unoptimized code, **hundreds of CPU-years** for each run.



Software Stack for Modeling Galactic 3D Structure



Accelerating Radiation Transport Models for the Milky Way

Solution: use a computing accelerator, optimize existing code.

Calculation of Stochastic Heating and Emissivity of Cosmic Dust Grains with Optimization for the Intel Many-Core Architecture

Troy A. Porter¹, Andrey E. Vladimirov^{1,2}

¹Physics Laboratory, Stanford University, 452 Lomita Mall, Stanford, CA 94305-4085, USA
²Colfax International, 750 Palomar Ave, Sunnyvale, CA 94085, USA

Result: HEATCODE
 (HEterogeneous Architecture library for sTOchastic COsmic Dust Emissivity)

(open source, code soon to be published)

<http://arxiv.org/abs/1311.4627>

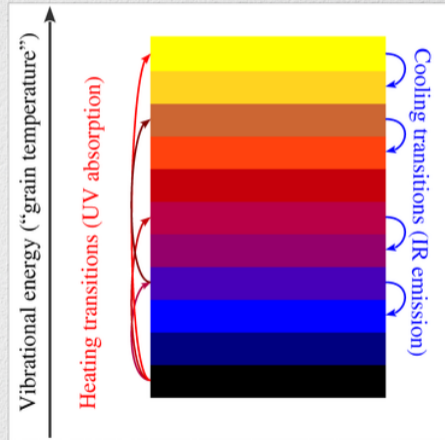
...light. Their absorption of starlight produces emission spectra from the near- to far-infrared. The absorption depends on the size and properties of the dust grains, and spectrum of the heating radiation field. The emission is dominated by very small grains. Modeling the absorption of starlight by these particles is computationally expensive and a significant bottleneck for self-consistent radiation transport codes treating the heating and emission of dust by stars. In this paper we summarize the formalism for computing the stochastic emissivity of cosmic dust, which was

Hundreds of CPU-years

Hundreds of CPU-days

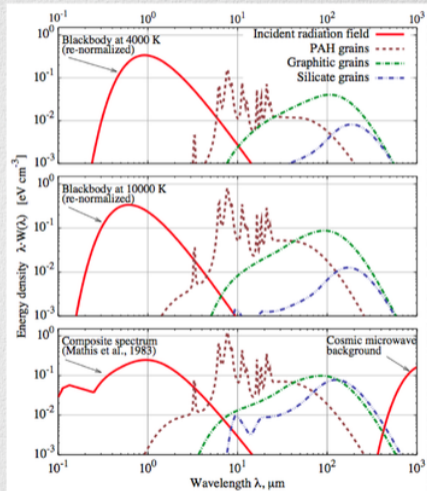
Stochastic Dust Grain Heating

- Small grains ($\leq 0.1 \mu\text{m}$) are important
- Absorption and re-emission is stochastic (non-thermal)
- Grains undergo “temperature” spikes, characterized by temperature distribution
- Evaluation is computationally expensive



Calculation of Stochastic Dust Emissivity

- **Input:** incident electromagnetic radiation field
- **Intermediate:** “temperature” distribution of grains of all sizes
- **Output:** spectrum of re-emitted photons
- Method and absorption cross sections: Draine et al. (2001), *ApJ*, 551, 807



Matrix Formalism for Stochastic Dust Emissivity

- **Stage 1:**

Interpolate (in log space) and convolve the incident RF with the photon absorption cross sections

$$T_{ul} = I(\lambda)\sigma(\lambda)\frac{\lambda^3\Delta E_{ul}}{hc^2} \quad \text{for } u > l.$$

$$I(\lambda)\sigma(\lambda) \equiv \Omega(\lambda)$$

$$\log\left[\frac{\Omega(\lambda)}{\Omega(\lambda_{j-1})}\right] = \frac{\log(\lambda/\lambda_{j-1})}{\log(\lambda_j/\lambda_{j-1})} \log\left[\frac{\Omega(\lambda_j)}{\Omega(\lambda_{j-1})}\right]$$

transcendental operations

- **Stage 2:**

form and solve a quasi-triangular system of linear algebraic equations for the “temperature” distribution

$$\sum_{j \neq i} T_{ij}P_j - \sum_{j \neq i} T_{ji}P_i = 0$$

$$T_{ij} = 0, \quad \text{if } i < j - 1$$

$$B_{fj} = \sum_{k=f}^M T_{kj} \quad (f > j)$$

$$X_f = \frac{1}{T_{(f-1)f}} \sum_{j=0}^{f-1} B_{fj}X_j$$

sparse memory access

- **Stage 3:**

convolve the “temperature” distribution with the grain size distribution and emissivity function

$$\nu F_a(\nu) = \sigma(\nu) \sum_{i=0}^M P_i(a)\Lambda(\nu, E_i)$$

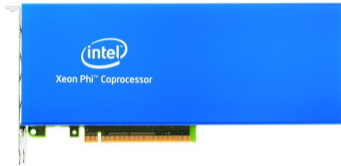
$$\Lambda(\nu, E_i) = \begin{cases} 0, & \text{if } E_i < h\nu, \\ \frac{2h\nu^4}{c^2} \frac{P_i}{\exp(h\nu/kT_i) - 1} & \end{cases}$$

$$\nu F(\nu) = \int_{a_{\min}}^{a_{\max}} \nu F_a(\nu) Q(a) da$$

dense linear algebra

§5. Practical Optimization

“Standard Candle” Testbench



One Intel Xeon Phi 7120P
coprocessor (2012)
TDP: 300 W, RCP: \$4129

vs.



Two Intel Xeon E5-2697 v3
CPUs (2014)
TDP: 290 W, RCP: \$5404

See also [“Intel Xeon Product Family: Performance Brief”](#)

Focust Point 1: Algorithm

- Before: 3 nested loops — $O(N^3)$ complexity
- After: 2 nested loops — $O(N^2)$ complexity
- Used recurrent relation to improve algorithm

Speedup in HEATCODE: 3.7x

Focus Point 2: Thread Scalability

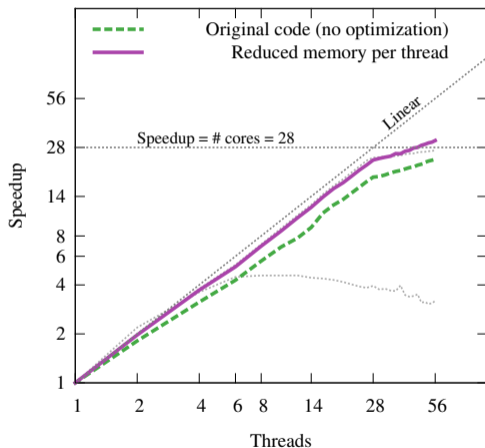
- Minimize synchronization
- Tune scheduling for load balancing
- Expose all of the parallelism
- Make threading co-exist with vectorization

Speedup in HEATCODE: 1.2x (total: 4.4x)

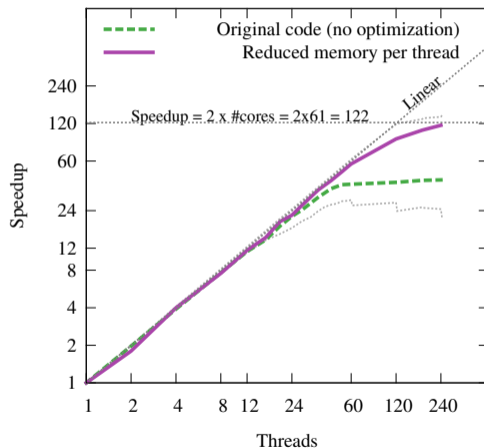
More details in the HOW series [Session 4](#) and [Session 7](#)

HEATCODE Scalability Tuning

Performance of HEATCODE the CPU architecture



Performance of HEATCODE the MIC architecture



Fruit 1: Algebraic Optimization

- **Precompute + look up vs Compute on time** — tradeoff
- Rule of thumb: 100 FLOPs = 1 memory access
- Precomputation helped in HEATCODE

Speedup in HEATCODE: 4.3x (total: 19x)

Fruit 2: Scalar Tuning

- All double precision = OK
- All single precision = GREAT
- Mixed precision = BAD

Speedup in HEATCODE: 1.8x (total: 34x)

More details in the HOW series [Session 6](#).

Fruit 3: Memory Traffic

- Loop tiling: cache blocking or unroll-and-jam
- Cache-oblivious parallel recursion
- Loop fusion

Speedup in HEATCODE: 2.4x (total: 82x)

More details in the HOW series [Session 9](#).

Fruit 4: Vectorization

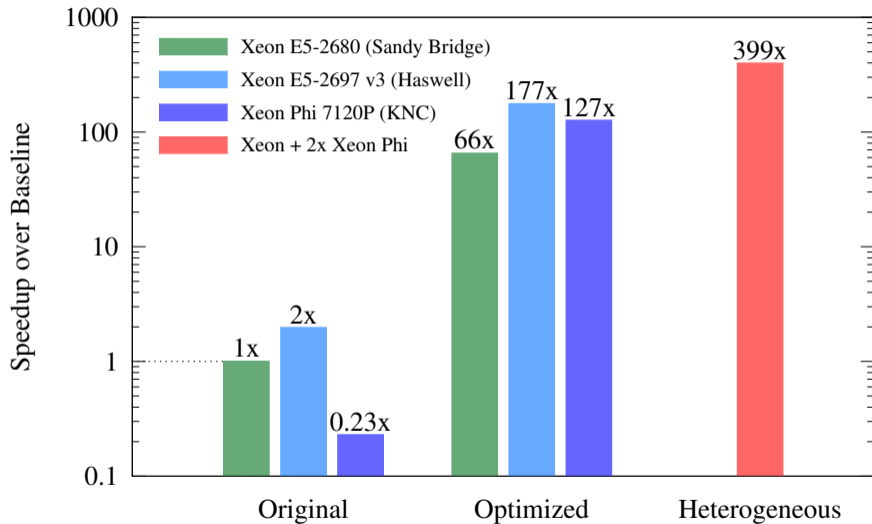
- Data alignment
- Alignment hint
- Guiding the compiler with `#pragma simd`

Speedup in HEATCODE: 1.1x (total: 91x)

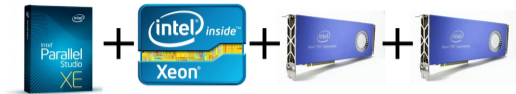
More details in the HOW series [Session 6](#).

§6. Summary

Performance Results



Illustration



Porting to Intel® Xeon Phi™ coprocessors

- We ported Frankie code using explicit offload model
- Same code & optimization methods for Xeon Phi™
- Simultaneous calculations on CPU and coprocessors with automatic load balancing was easy to implement
- With two Intel® Xeon Phi™ coprocessors, performance for high-res calculations is 3.2x better than with two Intel® Xeon® E5 processors alone.
- **RESULT:** estimated target project calculation time is now 2 weeks (down from 6+ years)

Goal achieved!

Transient Emission of Cosmic Dust Grains
in the Milky Way Galaxy,
Simulation with Frankie Code

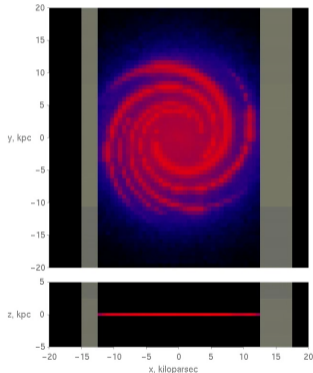


Illustration: youtu.be

Paper: <http://xeonphi.com/papers/heatcode>

Optimization Methodology

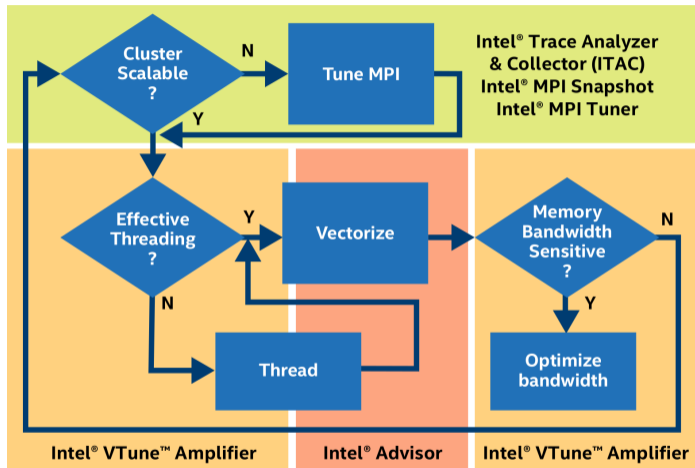
Optimization Areas

- Scalar optimization
- Vectorization
- Multi-threading
- Memory access
- Communication

How to Navigate Them

- ① Focus on the algorithm
- ② Achieve parallel scalability
- ③ Hotspot-guided optimization:
iteratively use VTune +
optimization report

Alternative Optimization Methodology



Source: [Parallel Universe Magazine](#), issue 23

§7. Resources

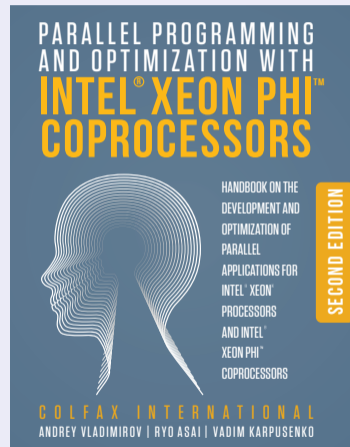
Supplementary Materials: Textbook

ISBN: 978-0-9885234-0-1 (2nd edition, 508 pages, Electronic or Print)

Parallel Programming and Optimization with Intel® Xeon Phi™ Coproprocessors

Handbook on the Development and
Optimization of Parallel Applications
for Intel® Xeon® Processors
and Intel® Xeon Phi™ Coprocessors

© Colfax International, 2015



<http://xeonphi.com/book>

COLFAX RESEARCH
CONTRIBUTING TO INNOVATIONS IN COMPUTING

Log In/Out or Register

READ WATCH LEARN CONNECT JOIN

To search, type and hit enter

Popular

The Hands-On Tutorials (HOT) webinars: details on efficient programming for Intel architecture

The Hands-On Workshop (HOW) Series

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Research and Educational Publications

Introduction to Intel DAAL, Part 1: Polynomial Regression with Batch Mode Computation

Optimization Techniques for the Intel MIC Architecture, Part 3 of 3: False Sharing and Padding

Software Developer's Introduction to the H88T Ultrastar Archive H88T SMR Drives

Optimization Techniques for the Intel MIC Architecture, Part 2 of 3: Strip-Mining for Vectorization

Optimization Techniques for the Intel MIC Architecture, Part 1 of 3: Multi-Threading and Parallel Reduction

Performance to Power and Performance to Cost Ratios with Intel Xeon Phi Coprocessors (and why 1x Acceleration May Be Enough)

Events

Presentations

Case Studies

Consulting

Share

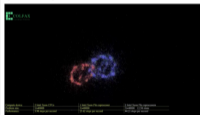


Colfax offers consulting services for enterprises, research and education to help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and beyond
- Future-proof your application for upcoming innovations, including the Intel Xeon Phi coprocessor
- Investigate the potential system configurations that satisfy your cost, power and performance requirements.
- Take a clean sheet to evaluate a novel approach to collaborate across computing platforms, software and hardware

All Video Courses - COP 981 - Chapter 2 - Episode 1.1

Episode 2.1 - Purpose of the MIC architecture



In this episode (2.1) we will introduce how the MIC architecture is used in the Intel Xeon Phi coprocessor. We will also discuss the architecture of the Intel Xeon Phi coprocessor.

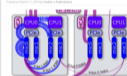
Share

Software Developer's Introduction to the H88T Ultrastar Archive H88T SMR Drives



Share

Configuration and Benchmarks of Peer-to-Peer Communication over Gigabit Ethernet and InfiniBand in a Cluster with Intel Xeon Phi Coprocessors



Share

Parallel Computing in the Search for New Physics at LHC



Share

Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors



Share

Interview with James Reinders: future of Intel MIC architecture, parallel programming, education



Share



Share

Learn More



THE "HOW" SERIES

DEEP DIVE

WITH CODE MODERNIZATION EXPERTS

STARTS MAY 23

*10x 2-hour sessions | 24-hour 2-weeks remote access to a system | Filling up fast, register now!

Interested? Sign-up at:

colfaxresearch.com/how-series

Slides, Code, Video

You can download slides, code and watch the video recording of this webinar here (requires registration for a free Colfax Research account):

colfaxresearch.com/how-tools-16-05

Next webinar on May 18, 2016: “Practical usage of Intel Math Kernel Library: performance tuning tips and usage with coprocessors”:

Register