PROGRAMMING AND OPTIMIZATION FOR INTEL® ARCHITECTURE

The Hands-On Workshop (HOW) Series

Session 1

Colfax International — *colfaxresearch.com*

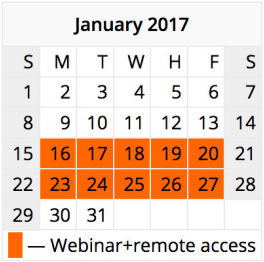
January 2017

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.

COURSE ROADMAP

- Module I. Programming
  - 01. Intel Architecture and Modern Code – Jan 16
  - 02. Xeon Phi, Coprocessors, Omni-Path – Jan 17
- Module II. Expresssing Parallelism
  - 03. Expressing Parallelism with Vectors – Jan 18
  - 04. Multi-threading with OpenMP – Jan 19
  - 06. Distributed Computing, MPI – Jan 20
- Module III. Optimization
  - 06. Optimization Overview: N-body – Jan 23
  - 07. Scalar tuning, Vectorization – Jan 24
  - 08. Common Multi-threading Problems – Jan 25
  - 09. Multi-threading, Memory Aspect – Jan 26
  - 10. Access to Caches and Memory – Jan 27



HOW SERIES ONLINE

Course page:
colfaxresearch.com/how-17-01

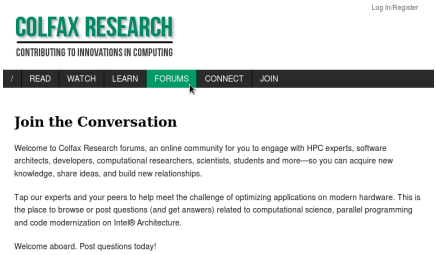
- Slides
- Code
- Video
- Chat

More workshops:
colfaxresearch.com/training

GET YOUR QUESTIONS ANSWERED

Chat (current):
colfaxresearch.com/how-17-01

Forums (technical):
colfaxresearch.com/discussion

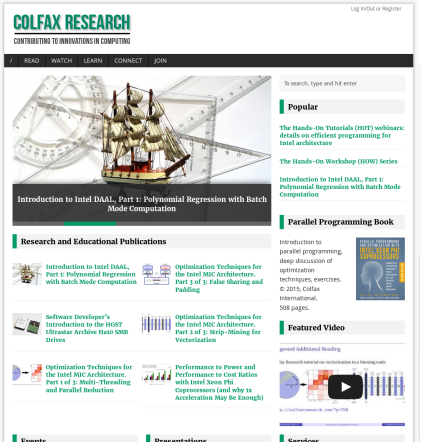


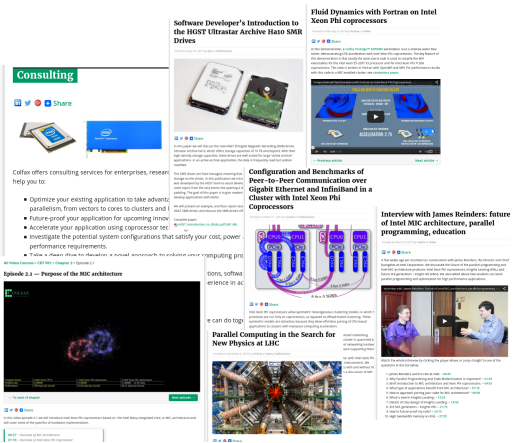
Email (organizational):
`training@colfaxresearch.com`

HANDS-ON EXERCISES AND REMOTE ACCESS

- All registrants receive an invitation from cluster@colfaxresearch.com
- Queue-based access to Intel Xeon E5, Intel Xeon Phi (KNC and KNL)
- Can access the cluster the entire 2 weeks of the workshop

COLFAX RESEARCH



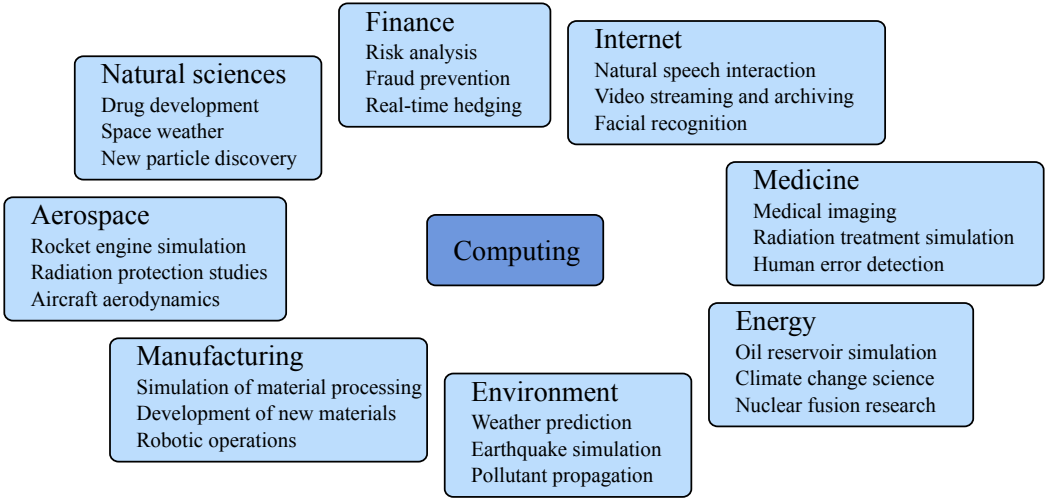


http://colfaxresearch.com/

§2. MODERN CHALLENGES

AREAS OF APPLICATION

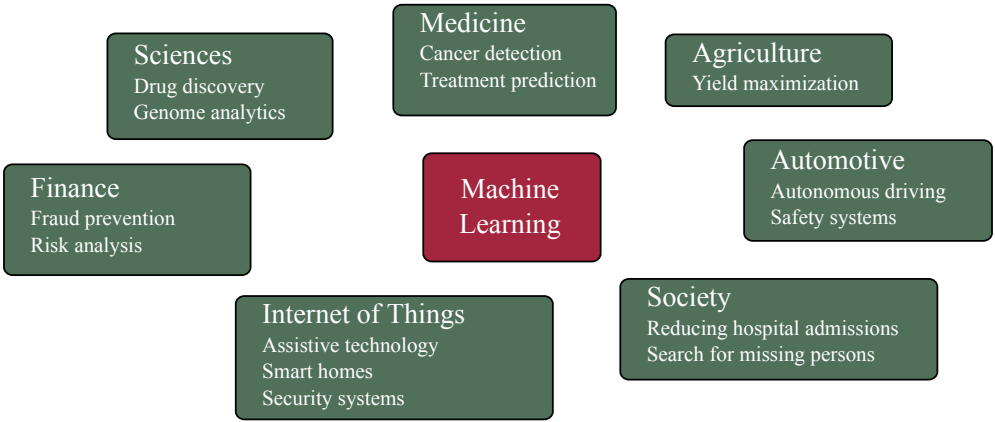
TRADITIONAL COMPUTING APPLICATIONS



ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

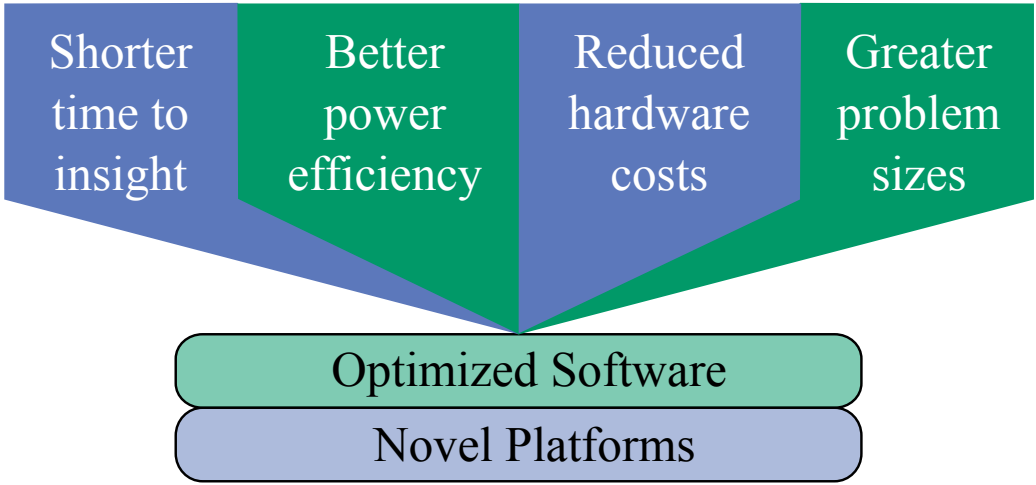
AI = reasoning, perception, action and adaptation by machines

ML = data-driven AI, intelligence without explicit programming



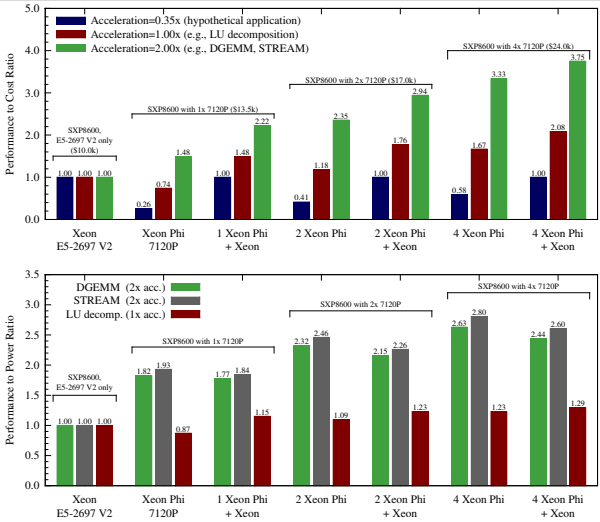
NEED FOR SPEED

PERFORMANCE OPTIMIZATION OUTCOMES



WHAT IS YOUR PERFORMANCE METRIC?

- Performance per System
- Performance per Watt
- Performance/Cost Ratio

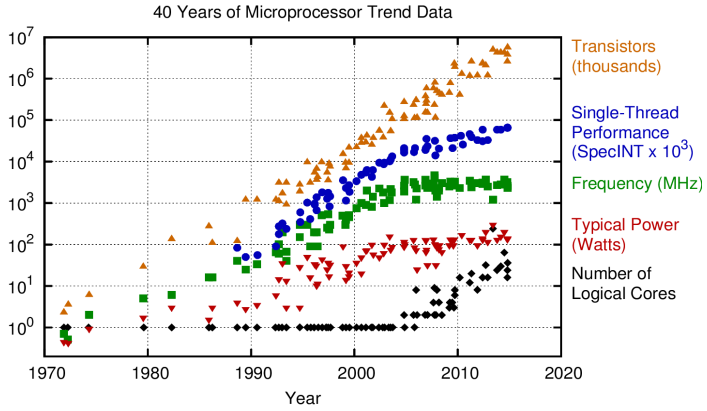


See this paper for details

§3. MODERN ARCHITECTURE

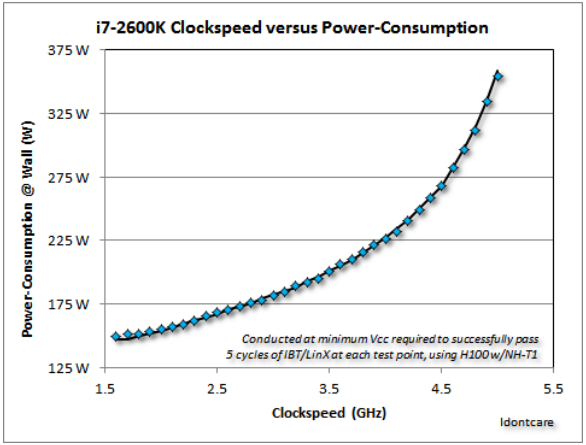
HOW PROCESSORS GET FASTER

40 YEARS OF MICROPROCESSOR DATA



Source: karlrupp.net

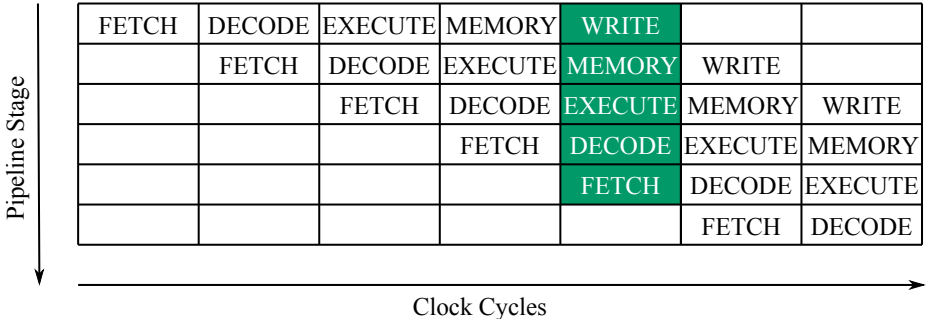
POWER WALL



Source: anandtech.com/...

INSTRUCTION-LEVEL PARALLELISM (ILP) WALL

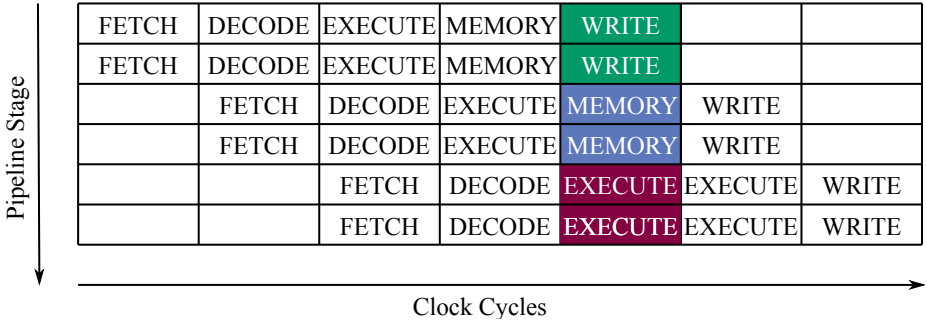
Pipelining – replication of hardware to run different stages of different instruction streams at the same time



Automatic search for pipelining opportunities requires extra resources

INSTRUCTION-LEVEL PARALLELISM (ILP) WALL: SUPERSCALAR EXECUTION

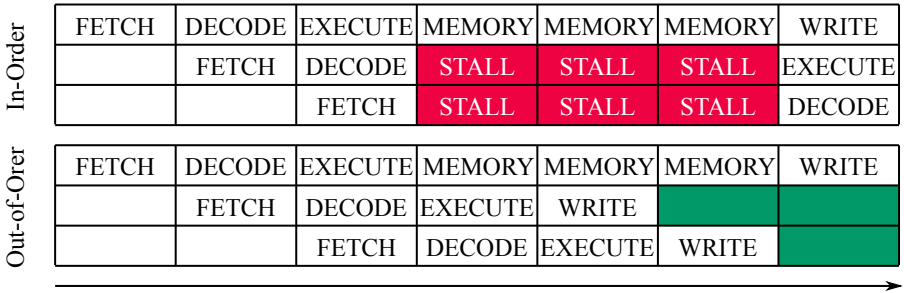
Superscalar Execution – hardware checks for independence of operations, pipelines multiple instructions in a cycle



Automatic search for independent instructions reqiures extra resources

MEMORY WALL: OUT-OF-ORDER EXECUTION

Out-of-order Execution – hardware re-orders instructions in a stream to minimize latencies

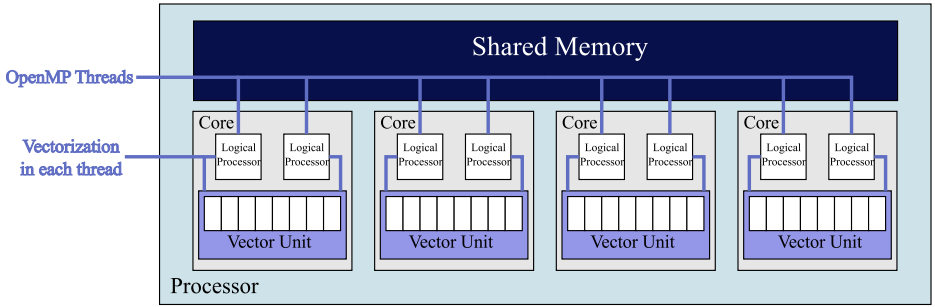


Limited by data locality access pattern in application.

PARALLELISM

CORES – multiple instructions on multiple data elements (MIMD)

VECTORS – single instruction on multiple data elements (SIMD)



Unbounded growth opportunity, but **not automatic**

PARALLELISM IS THE PATH FORWARD

- Clock speed has hit the power wall
- Automatic parallelism has hit the ILP wall
- Out-of-order execution cannot overcome the memory wall

The Show Must Go On

Hardware keeps evolving through parallelism.
Software must catch up!

INTEL ARCHITECTURE

INTEL COMPUTING PLATFORMS

General-Purpose Processors

Intel® Xeon®
Intel® Core™
Intel® Atom™, ...

Specialized Processors

Intel® Xeon Phi™ processors
and coprocessors

Computing Accelerators

Intel® VCA (x86)
Intel® Nervana™ Platform
Intel® DLIA™ (FPGAs)

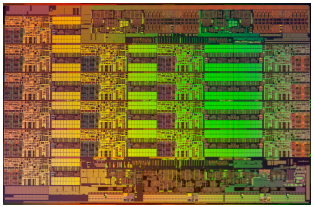
Network Interconnects

Intel® Omni-Path™ 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 3072 GiB DDR4 RAM*
- ~ 154 GB/s bandwidth*
- Hardware-rich: forgiving of sub-optimal code

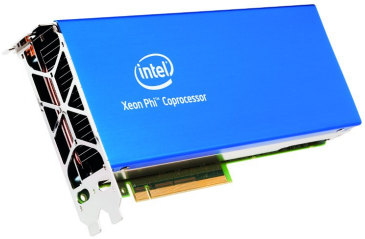


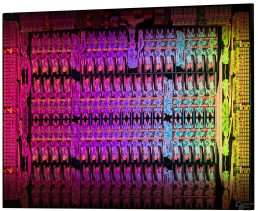
* 2-way Intel Xeon processor, Skylake architecture, top-of-the-line (e.g., E5-2699 V4)

INTEL XEON PHI PROCESSORS (1ST GEN)

Specialized platform for demanding computing applications.

- PCIe add-in card
- ~ 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

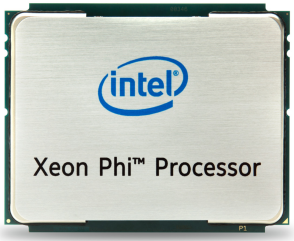


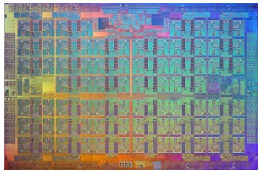


INTEL XEON PHI PROCESSORS (2ND GEN)

Specialized platform for demanding computing applications.

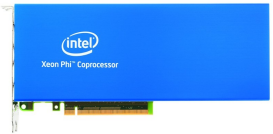
- Socket version or coprocessor
- 64-72 cores × 4 HT at 1.3-1.5 GHz
- 3+ TFLOP/s in DP (FMA)
- 6+ TFLOP/s in SP (FMA)
- ≤ 384 GiB DDR4 (> 90 GB/s)
- 16 GiB HBM (MCDRAM, > 400 GB/s)
- Binary-compatible with Xeon
- Common OS (RHEL/CentOS/SUSE/Windows)





BIRD'S EYE VIEW

- C/C++/Fortran
- Linux/Windows
- ≤3 TiB DDR4
- ≤44 cores (2-way)
- ≈3 GHz
- 2 HT/core
- 256-bit AVX

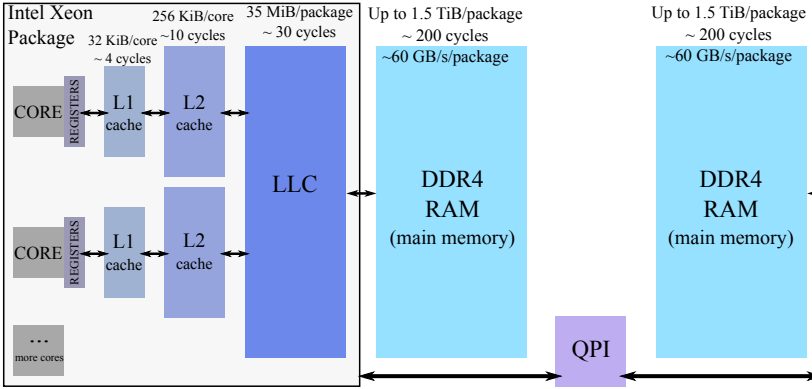


- C/C++/Fortran
- Special Linux
- ≤ 16 GiB GDDR5
- 57-61 cores
- ≈1.2 GHz
- 4 HW THR/core
- 512-bit IMCI

- C/C++/Fortran
- Linux
- MCDRAM+DDR4
- 64-72 cores
- 1.3-1.5 GHz
- 4 HT/core
- 512-bit AVX-512

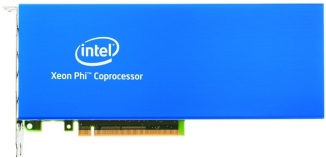
INTEL XEON CPU: MEMORY ORGANIZATION

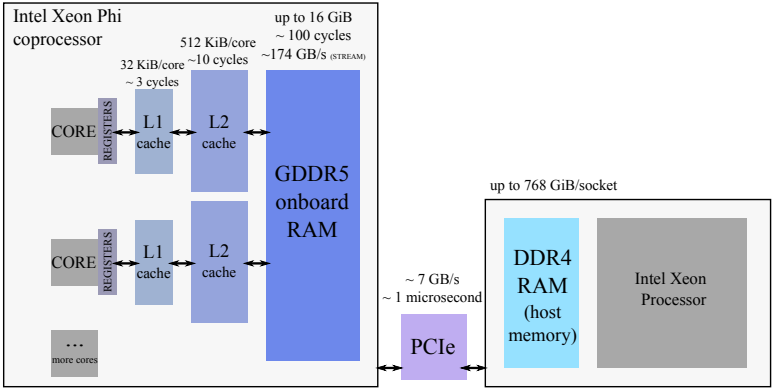
- Hierarchical cache structure
- Two-way processors have NUMA architecture



KNC MEMORY ORGANIZATION

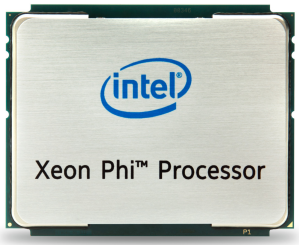
- Direct access to ≤ 16 GiB of cached GDDR5 memory on board
- No access to system DDR4, connected to host via PCIe

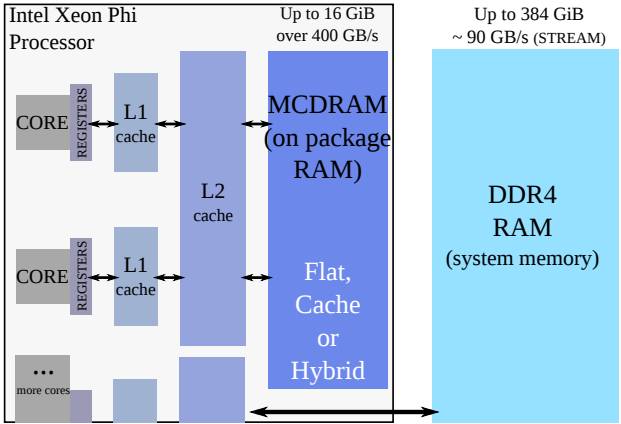




KNL MEMORY ORGANIZATION (BOOTABLE)

- Direct access to on-platform RAM and on-package HBM
- Use HBM as cache, in flat mode, or as hybrid

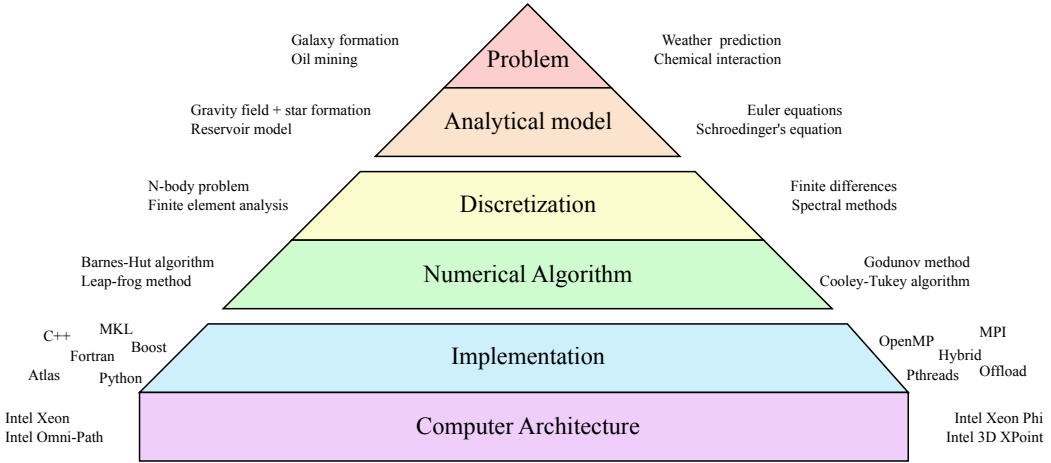




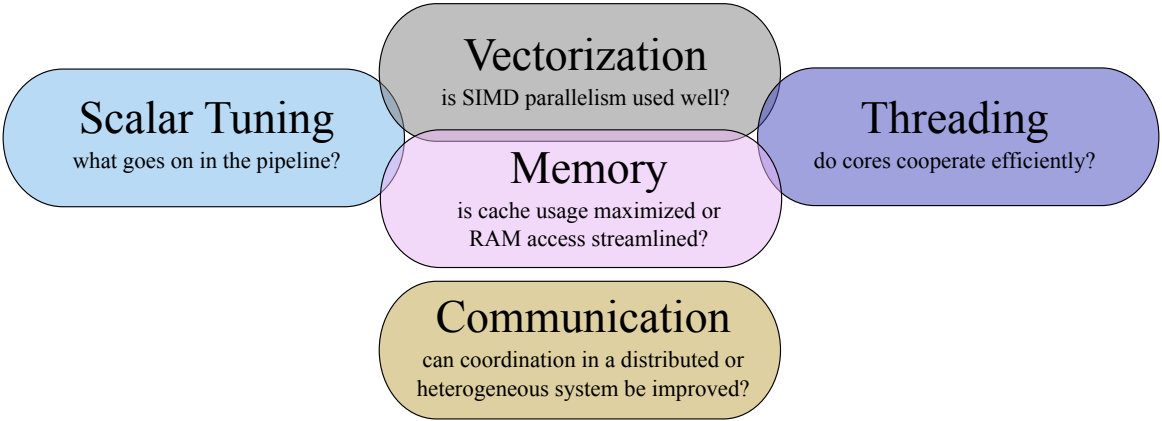
§4. MODERN CODE

OPTIMIZATION AND FUTURE-PROOFING

COMPUTING IN SCIENCE AND ENGINEERING

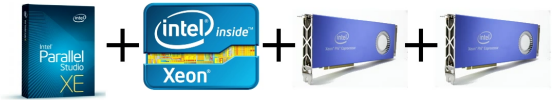


OPTIMIZATION AREAS



MOTIVATING EXAMPLES

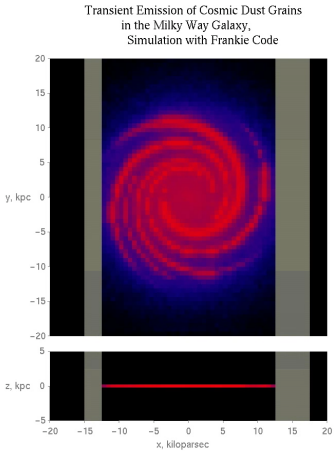
ASTROPHYSICAL CODE HEATCODE: AN OFFLOAD STORY



Porting to Intel® Xeon Phi™ coprocessors

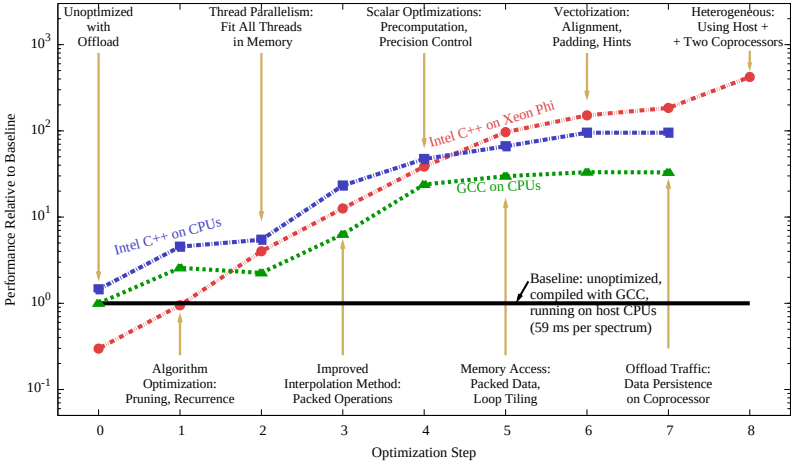
- We ported Frankie code using explicit offload model
- Same code & optimization methods for Xeon Phi™
- Simultaneous calculations on CPU and coprocessors with automatic load balancing was easy to implement
- With two Intel® Xeon Phi™ coprocessors, performance for high-res calculations is 3.2x better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

Goal achieved!



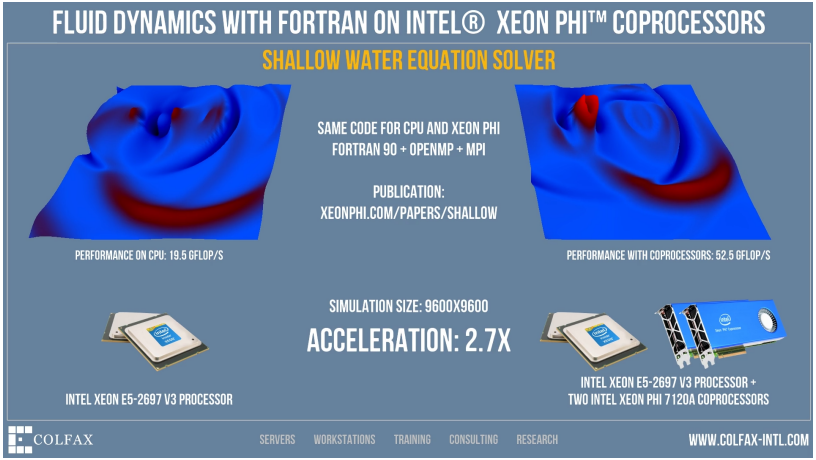
http://xeonphi.com/papers/heatcode

ASTROPHYSICAL CODE HEATCODE: AN OFFLOAD STORY



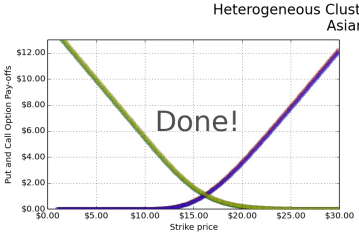
http://xeonphi.com/papers/heatcode

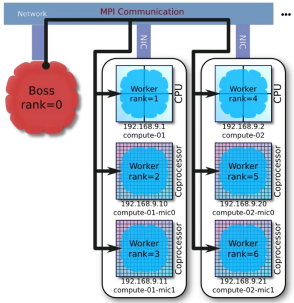
COMPUTATIONAL FLUID DYNAMICS: LEGACY CODE

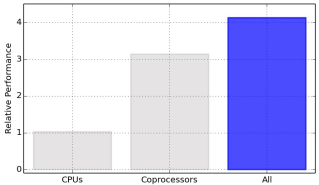


http://xeonphi.com/papers/shallow

ASIAN OPTION PRICING: HETEROGENEOUS CLUSTERING



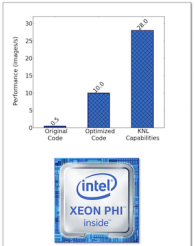




http://xeonphi.com/papers/heterogeneous

MACHINE LEARNING: OPTIMIZED MIDDLEWARE

INTEL® XEON PHI™ PROCESSORS — MACHINE LEARNING



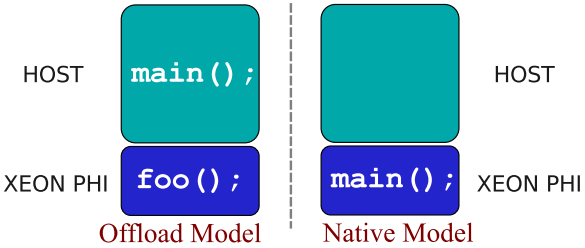
NEURALTALK2 — OPEN SOURCE IMAGE TAGGING CODE (KARPATHY & FEI-FEI, STANFORD)

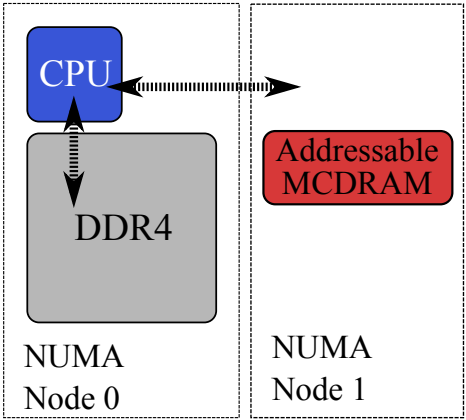
COLFAX Customized Solutions

http://colfaxresearch.com/isc16-neuraltalk

WHAT YOU ARE GOING TO LEARN

HETEROGENEOUS AND NUMA ARCHITECTURES



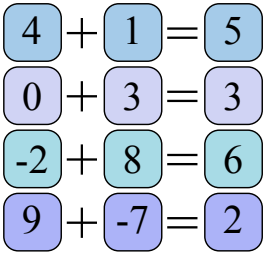


Session 2: handling memory organization in Intel Xeon Phi processors

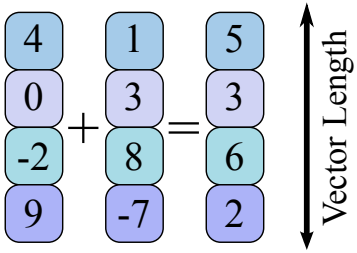
DATA PARALLELISM AND VECTOR INSTRUCTIONS

Vectors – form of SIMD architecture (Single Instruction Multiple Data).

Scalar Instructions



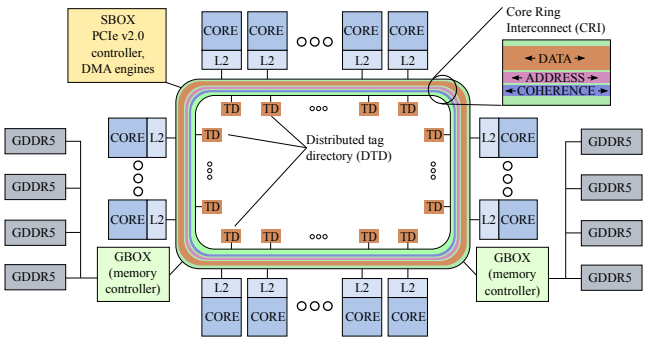
Vector Instructions

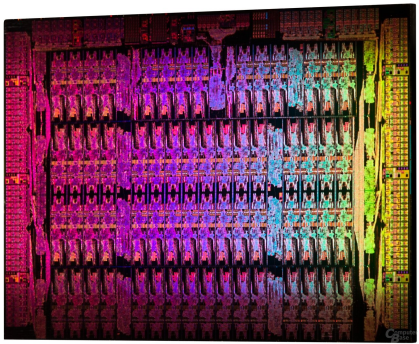


Session 3: automatic vectorization with Intel compilers

TASK PARALLELISM AND CORES

Cores implement MIMD (Multiple Instruction Multiple Data) arch

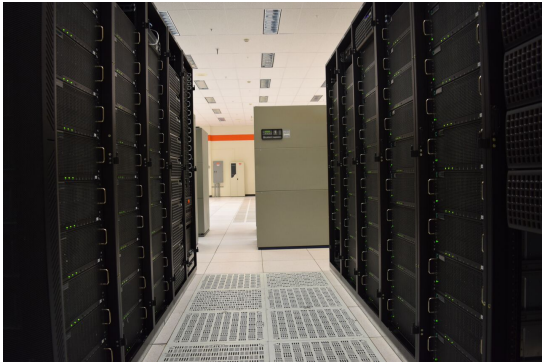


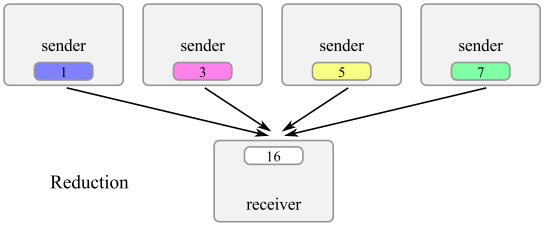


Session 4: multi-threading with OpenMP

DISTRIBUTED-MEMORY PROGRAMMING

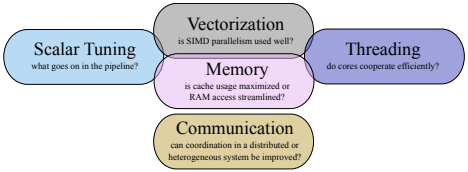
Clusters form distributed-memory systems with network interconnects

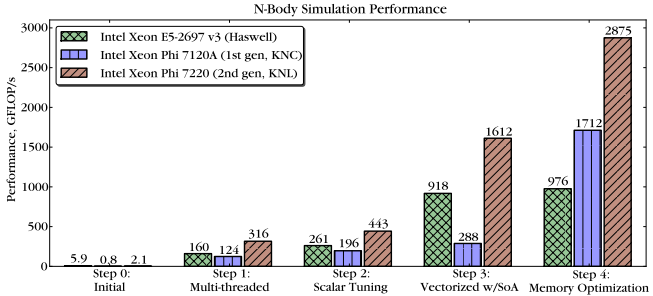




Session 5: Message Passing Interface (MPI)

OPTIMIZATION OVERVIEW

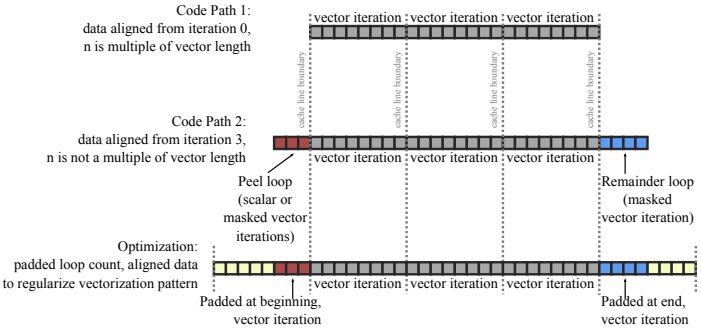




Session 6: optimization overview, case study

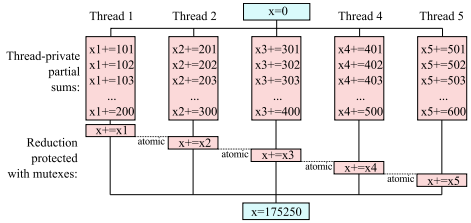
SCALAR TUNING, OPTIMIZATION OF VECTORIZATION

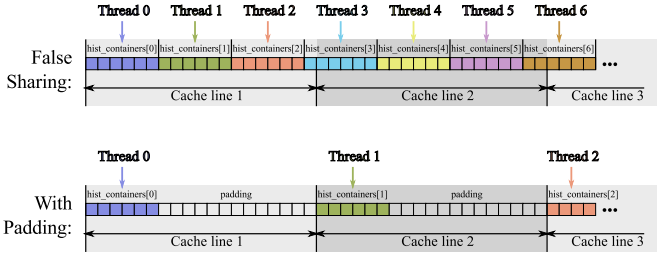
```
for (i = 0; i < n; i++)  A[i] = ...
```



Session 7: precision control, regularizing vectorization patterns

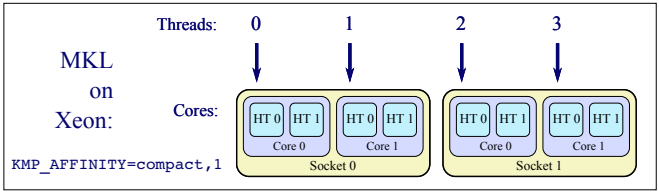
COMMON ISSUES IN MULTI-THREADING

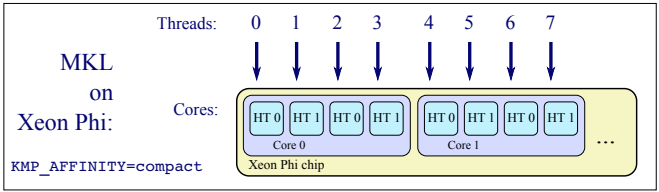


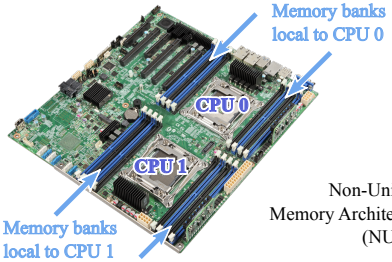


Session 8: minimizing synchronization, avoiding false sharing, strip-mining for parallelism

MULTI-THREADING, MEMORY ASPECT

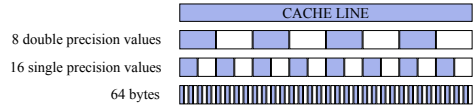


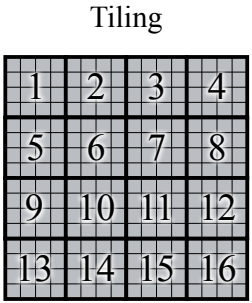


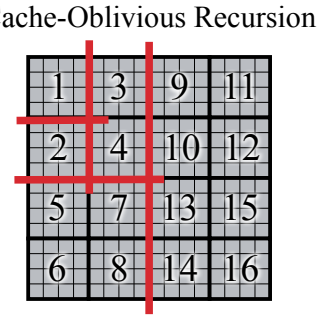


Session 9: thread affinity, NUMA locality, scheduling

CACHE AND MEMORY ACCESS







Session 10: loop transformations for locality, bandwidth secrets

§5. HANDS-ON DEMONSTRATION

ACCESS THE COLFAX CLUSTER

Home | Learn | Connect | Program | Compute | Log Out

Welcome to Colfax Cluster!

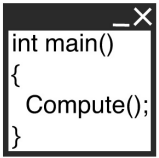
Learn

what to expect on the Colfax Cluster

Connect

from your home computer to the cluster

Program

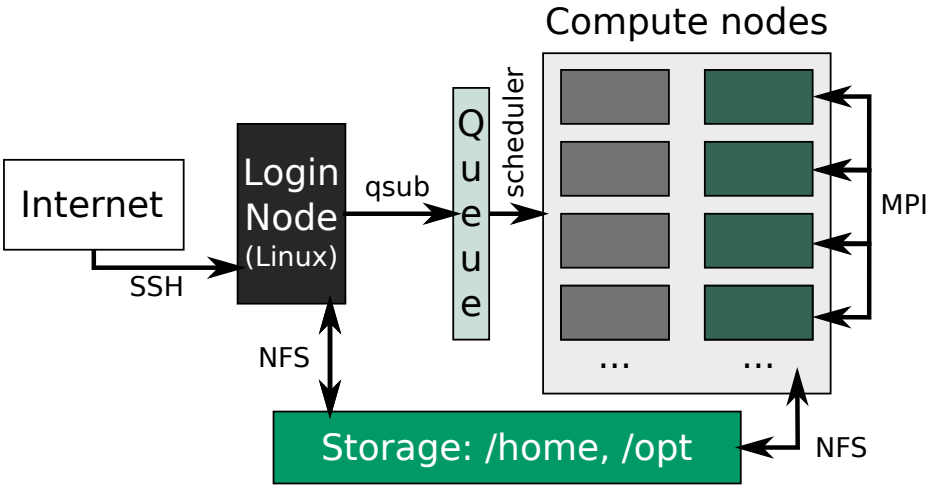


using modern code practices

Compute

with cluster job management tools

JOB MANAGEMENT IN THE CLUSTER



DOWNLOAD LABS

```
[u111@c005 ~]% git clone https://github.com/ColfaxResearch/HOW-Series-Labs.git
Cloning into 'HOW-Series-Labs'...
[u111@c005 ~]% ls HOW-Series-Labs/*/
HOW-Series-Labs/2/:
2.01-native-basic  2.03-offload-basic         2.05-shared-virtual-memory-basic
2.02-native-MPI    2.04-offload-asynchronous  2.06-shared-virtual-memory-complex-

HOW-Series-Labs/3/:
3.01-vectorization  3.03-OpenMP-reduction  3.05-Cilk-Plus-basics    3.07-Cilk-Plu
3.02-OpenMP-basics  3.04-OpenMP-tasks      3.06-Cilk-Plus-reducers  3.08-MPI-basi

HOW-Series-Labs/4/:
4.01-overview-nbody                          4.07-threading-affinity
4.02-vectorization-data-structures-coulomb   4.08-memory-tiling-matrix_x_vector
4.03-vectorization-tuning-lu-decomposition   4.09-memory-loop-fusion-statistics
4.04-threading-misc-histogram                4.10-offload-double-buffering-dgem
...
```

REVIEW AND WHAT'S NEXT

- Computers are getting faster through parallelism and specialization
- Intel Xeon E5 product family – general-purpose parallel processors
- Intel Xeon Phi product family – specialized parallel processors
- Coprocessor – either offload device or an additional compute node

Next session: details of Intel Xeon Phi processor and coprocessor programming.