



# Programming and Optimization for Intel<sup>®</sup> Architecture

The Hands-On Workshop (HOW) Series

Colfax International — @colfaxintl

June 2016 , Rev. 02d

# About This Document

This document represents the materials of a Web-based training “Programming and Optimization with Intel Architecture” developed and run by Colfax International.

© Colfax International, 2013–2016

Parallel Programming Boot Camp (1-Day) / Workshop (4-Days)



Instructor-led 1-day or 4-days training, at your office or at Colfax facility in Sunnyvale, CA

[Click here to learn more](#)

**1-Day Parallel Programming Boot Camp**  
 For software engineers and architects, providing an overview of parallel programming frameworks and optimization guidelines for multi-core CPUs (Intel® Xeon®) and many-core coprocessors (Intel® Xeon Phi™):

- Discussions about three layers of parallelism: SIMD, Threads, Cluster environment
- Tips for quick porting/development of HPC software applications
- Real-life examples of code and optimization techniques
- Hardware solution and corresponding software implementations, APIs, and frameworks

**4-Days Parallel Programming Workshop**  
 For the developer who wants to hit the ground running with the modern multi-core CPUs (Intel® Xeon®), many-core coprocessors (Intel® Xeon Phi™) and leading software development tools:

- Hardware installation
- MPSS tools and the Linux environment on the Intel® Xeon Phi™ coprocessor
- Exploring differences in serial vs. parallel programming / processing / hardware usage
- Accelerated clusters
- Optimizations of vector arithmetics, memory traffic, thread parallelism and communication
- Using the Intel® Math Kernel Library

Register Now!

[colfaxresearch.com/how-series](http://colfaxresearch.com/how-series)

# 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

- ① Why Intel Parallel Architectures?
  - ▶ Parallelism and specialization – June 20
  - ▶ Programming model continuity – June 20
- ② Programming models for Xeon Phi coprocessors
  - ▶ Native programming – June 20
  - ▶ Offload programming – June 21
- ③ Expressing Parallelism
  - ▶ Introduction to vectorization – June 22
  - ▶ Crash-course on OpenMP – June 23
- ④ Optimization – intro on June 24
  - ▶ Vectorization tuning – June 27
  - ▶ Multi-threading – June 28, 29
  - ▶ Memory traffic – June 30
- ⑤ Distributed Computing: MPI – July 1

June 2016						
S	M	T	W	H	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

July 2016						
S	M	T	W	H	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

 — 3:00pm UTC  
 Lecture+remote access

# HOW Online

Course page: [colfaxresearch.com/how-16-06](http://colfaxresearch.com/how-16-06)

- Slides (including this one), code downloads
- Video of recorded sessions
- Chat (during webinars or offline)



Additional resources:

- More workshops like this one: [colfaxresearch.com/training](http://colfaxresearch.com/training)
- Video courses: [colfaxresearch.com/video-courses](http://colfaxresearch.com/video-courses)

# Get Your Questions Answered

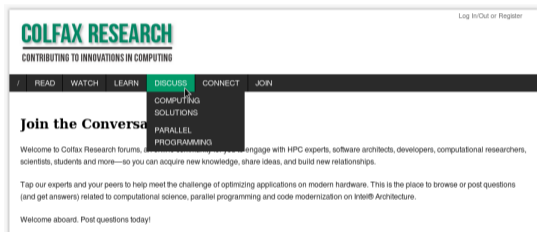
## Chat (current):

[colfaxresearch.com/how-16-06](http://colfaxresearch.com/how-16-06)



## Forums (technical):

[colfaxresearch.com/discussion](http://colfaxresearch.com/discussion)

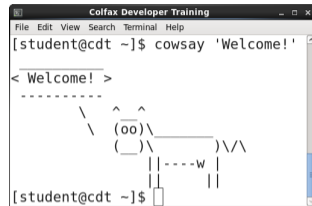


## Email (organizational):

[training@colfax-intl.com](mailto:training@colfax-intl.com)

# Hands-On Exercises and Remote Access

- 96 people receive a remote access token
- Can access the system the entire 3 weeks of the workshop
- Not among the 96? Stay tuned: follow along with instructor, use own system, or wait for a seat
- Use it or lose it: if you do not log in for a while, remote access token goes to next student on the list



```
Colfax Developer Training
File Edit View Search Terminal Help
[student@cdt ~]$ cowsay 'Welcome!'
< Welcome! >
-----
      \   ^__^
         (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

[student@cdt ~]$
```

## §2. Distributed Computing

# Computing Hardware

## Computing Platforms



Workstations

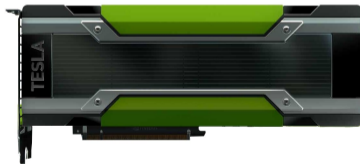


Servers



Clusters

## Computing Accelerators



GPGPUs



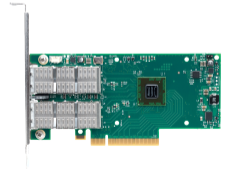
Coprocessors



FPGAs

# Clusters

Clusters often use Gigabit Ethernet for administration and InfiniBand for communication.



# Parallel Programming Frameworks

## Data-Parallel :

intrinsic, vector classes, automatic vectorization, array notation, kernels

## Shared Memory :

Pthreads, Threading Building Blocks, OpenMP

## Message Passing :

TCP/IP, MPI

# MPI and Heterogeneous Computing

# Structure of MPI Applications: Hello World

```
1 #include "mpi.h"
2 #include <stdio.h>
3 int main (int argc, char *argv[]) {
4     MPI_Init (&argc, &argv); // Initialize MPI environment
5     if (ret != MPI_SUCCESS) {
6         MyErrorLogger("...");
7         MPI_Abort(MPI_COMM_WORLD, ret);
8     }
9     int i, rank, size, namelen;
10    char name[MPI_MAX_PROCESSOR_NAME];
11    MPI_Comm_size (MPI_COMM_WORLD, &size);
12    MPI_Comm_rank (MPI_COMM_WORLD, &rank);
13    MPI_Get_processor_name (name, &namelen);
14    printf ("Hello World from rank %d running on %s!\n", rank, name);
15    if (rank == 0) printf("MPI World size = %d processes\n", size);
16    MPI_Finalize (); // Terminate MPI environment
17 }
```

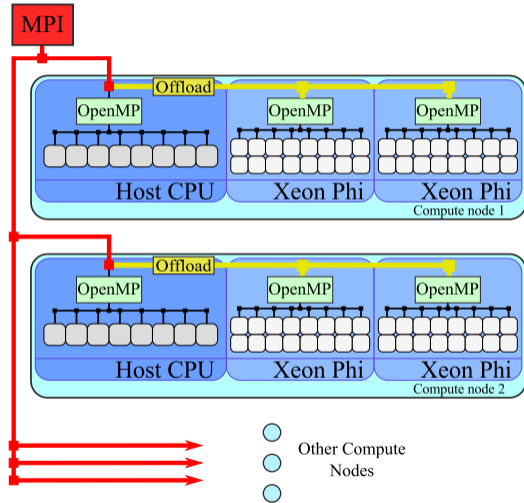
# Compiling and Running MPI Applications on Host

```
vega@lyra% mpiicpc -o HelloMPI HelloMPI.cc
vega@lyra% mpirun -host localhost -np 2 ./HelloMPI
Hello World from rank 1 running on lyra!
Hello World from rank 0 running on lyra!
MPI World size = 2 processes
```

- Set up MPI environment variables
- Use wrapper script `mpiicpc` to compile
- Use automated tool `mpirun` to launch

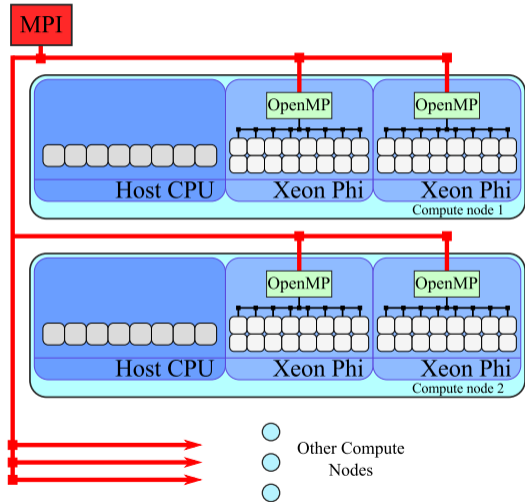
# Scaling Across a Cluster with Coprocessors

- We put MPI processes only on CPUs
- Subdivide data between coprocessors
- Concurrent offload from multiple host threads
- Synchronize data between nodes with MPI



# Scaling Across a Cluster with Coprocessors with MPI

- We put MPI processes only on CPUs
- Subdivide data between coprocessors
- Concurrent offload from multiple host threads
- Synchronize data between nodes with MPI



# Compiling and Running Native MPI Applications on Coprocessors

```
vega@lyra% export I_MPI_MIC=1
vega@lyra% mpiicpc -mmic -o HelloMPI.MIC HelloMPI.c
vega@lyra% scp HelloMPI.MIC mic0:~/
vega@lyra% mpirun -host mic0 -np 2 ~/HelloMPI.MIC
Hello World from rank 1 running on lyra-mic0!
Hello World from rank 0 running on lyra-mic0!
MPI World size = 2 processes
```

- Enable the MIC architecture in Intel MPI: `I_MPI_MIC=1`
- Copy or NFS-share MPI library & executables with coprocessor
- Use `mpiicpc` with `-mmic` to compile
- **Launch as if `mic0` is a remote host**

# Heterogeneous MPI Applications: Host + Coprocessors

```
vega@lyra% mpirun \  
> -host mic0 -n 2 ~/Hello.MIC : \  
> -host mic1 -n 2 ~/Hello.MIC : \  
> -host localhost -n 2 ~/Hello  
Hello World from rank 5 running on localhost!  
Hello World from rank 4 running on localhost!  
Hello World from rank 2 running on mic1!  
Hello World from rank 3 running on mic1!  
Hello World from rank 1 running on mic0!  
Hello World from rank 0 running on mic0!  
MPI World size = 6 ranks
```

- Specify Xeon executable for host processes
- Specify Xeon Phi executable for coprocessor processes

# Heterogeneous MPI Applications: Machine File

```
vega@lyra% cat hosts.txt
localhost:2
mic0:2
mic1:2
vega@lyra% export I_MPI_MIC_POSTFIX=.MIC
vega@lyra% mpirun -machinefile hosts.txt ~/Hello
Hello World from rank 0 running on localhost!
Hello World from rank 1 running on localhost!
Hello World from rank 2 running on mic1!
Hello World from rank 3 running on mic1!
Hello World from rank 4 running on mic0!
Hello World from rank 5 running on mic0!
MPI World size = 6 ranks
```

- Specify Xeon executable for host processes
- MIC executable obtained by appending `I_MPI_MIC_POSTFIX`

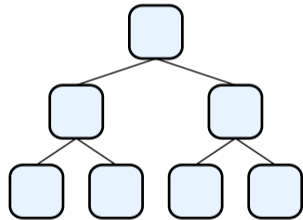
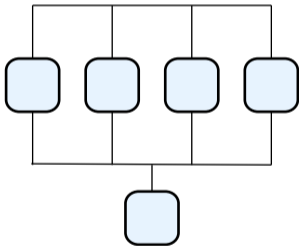
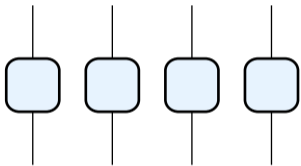
# Compiling and Running MPI applications

- 1 Compile and link with the MPI wrapper of the compiler:
  - ▶ `mpicc` for C,
  - ▶ `mpicxx` for C++,
  - ▶ `mpiifort` for Fortran 77 and Fortran 95.
- 2 Set up MPI environment variables *and* `I_MPI_MIC=1`
- 3 NFS-share or copy the MPI library and the application executable to the coprocessors
- 4 Launch with the tool `mpirun`
  - ▶ Colon-separated list of executables and hosts (argument `-host hostname`),
  - ▶ Alternatively, use the machine file to list hosts
  - ▶ Coprocessors have hostnames defined in `/etc/hosts`

# Communication Patterns

# Parallel Patterns

Embarrassingly parallel, reduction, fork-join.

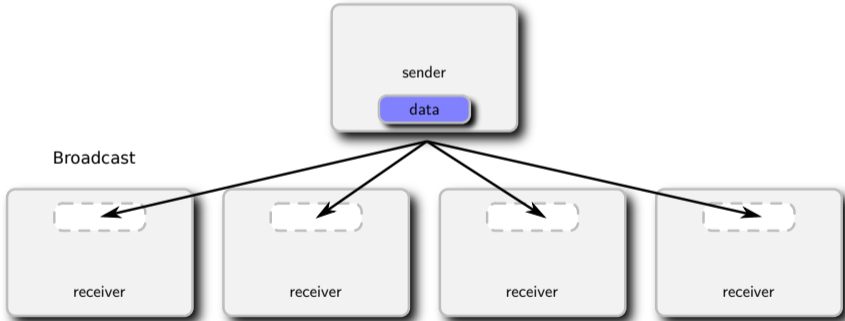


# Point to Point Communication

```
1  if (rank == sender) {
2
3     char outgoingMsg[messageLength];
4     strcpy(outgoingMsg, "/Jenny");
5     MPI_Send(&outgoingMsg, messageLength, MPI_CHAR, receiver, tag, MPI_COMM_WORLD);
6
7
8  } else if (rank == receiver) {
9
10    char incomingMsg[messageLength];
11    MPI_Recv (&incomingMsg, messageLength, MPI_CHAR, sender,
12            tag, MPI_COMM_WORLD, &stat);
13    printf ("Received message with tag %d: '%s'\n", tag, incomingMsg);
14
15
16 }
```

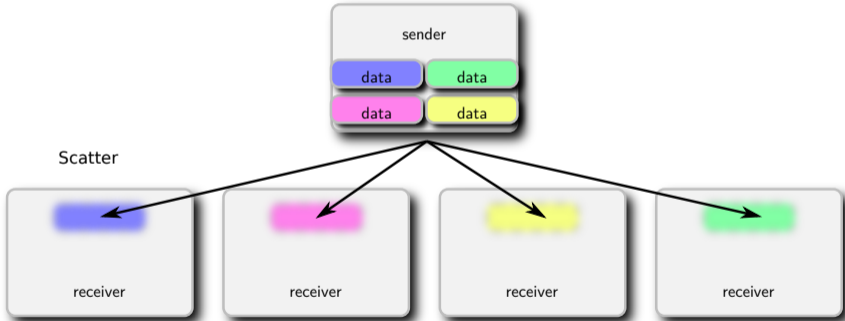
# Collective Communication: Broadcast

```
1 int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype,  
2 int root, MPI_Comm comm );
```



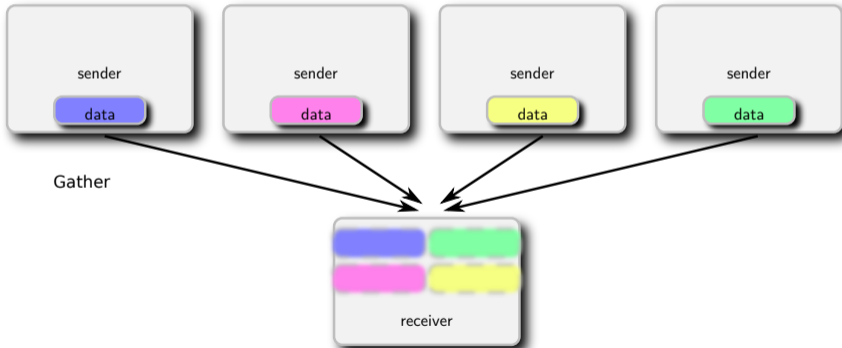
# Collective Communication: Scatter

```
1 int MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void *recvbuf,  
2   int recvcnt, MPI_Datatype recvtype, int root, MPI_Comm comm);
```



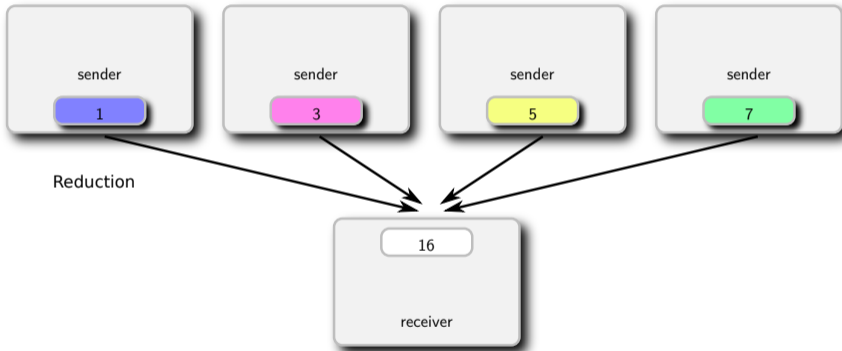
# Collective Communication: Gather

```
1 int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype,  
2 void *recvbuf, int recvcnt, MPI_Datatype recvtype,  
3 int root, MPI_Comm comm);
```



# Collective Communication: Reduction

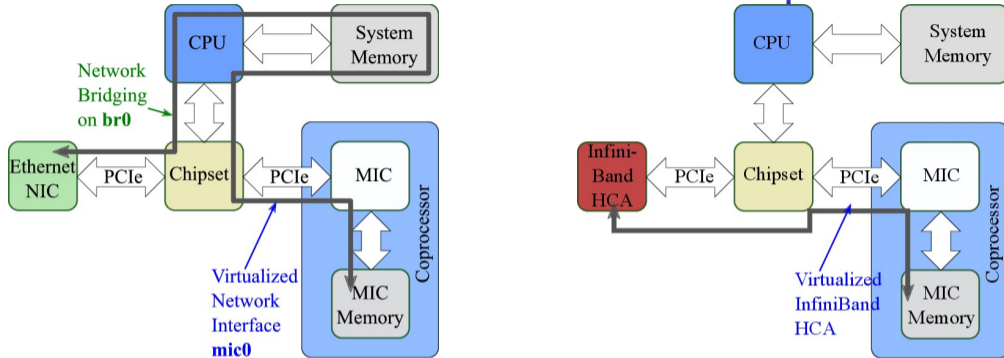
```
1 int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype,  
2 MPI_Op op, int root, MPI_Comm comm);
```



Available reducers: max/min, minloc/maxloc, sum, product, AND, OR, XOR (logical or bitwise).

# Coprocessors and InfiniBand

# Peer-to-Peer Communication between Coprocessors

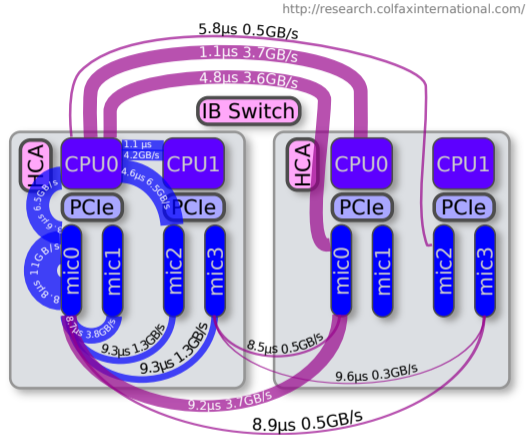


- Left: Gigabit Ethernet bridging on host allows to place coprocessors on the same subnet as hosts
- Right: Coprocessor Communication Link (CCL) – virtualization of an InfiniBand device on each coprocessor

# MPI Fabric Selection

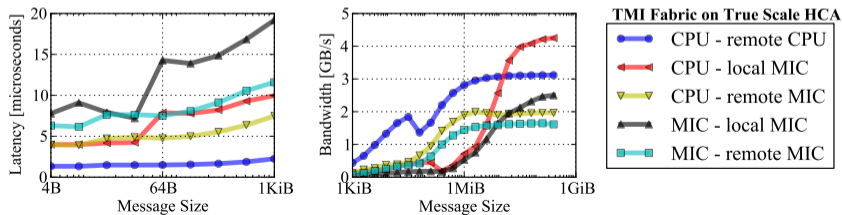
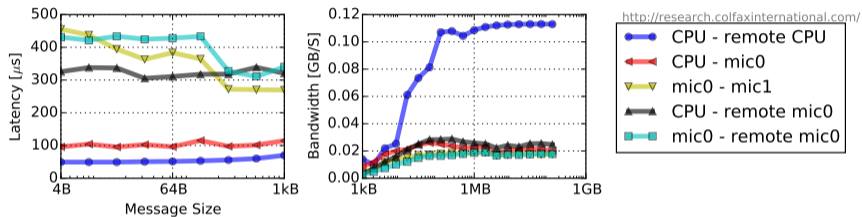
- MPI communication between CPU and coprocessors as efficient as offload
- Peer-to-peer communication not uniform, but better than with Gigabit Ethernet
- Control: environment variable `I_MPI_FABRICS`

Our publication with details:  
<http://xeonphi.com/papers/p2p>



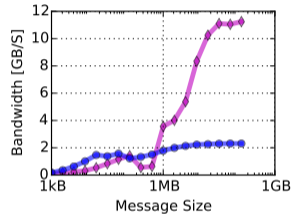
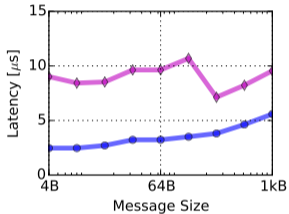
# Gigabit Ethernet versus Intel True Scale Interconnects

- Ethernet+TCP between coprocessors slower than the hardware limit
- InfiniBand approaches the hardware limit from CPU to coprocessors

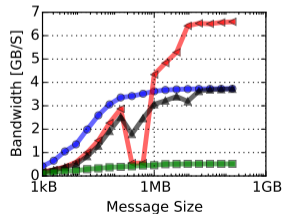
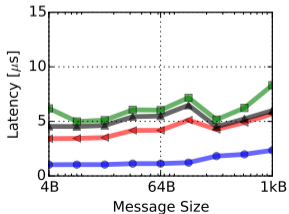
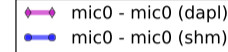


# Intra-Device and Intra-Node Communication

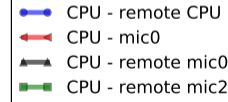
- Fabrics for messages within a single coprocessor:
- shm provides better latency, dap1 – greater bandwidth



<http://research.colfaxinternational.com/>



<http://research.colfaxinternational.com/>



# Intra-Device and Intra-Node Communication

## Intel® Omni-Path Fabric

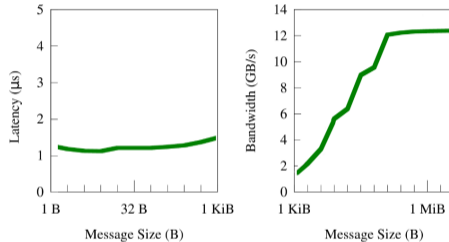
```

File Edit View Search Terminal Help
[ryoc010-n003 ~]$ cat /sys/module/hfi1/parameters/descct_intr
64
[ryoc010-n003 ~]$ cat /sys/module/hfi1/parameters/sdma_descq_cnt
2048
[ryoc010-n003 ~]$ mpirun -np 2 -ppn 1 -host c010-n003,c010-n004 \
-PSM -genv I_MPI_PIN_PROCESSOR_LIST=0 ~/osu_bw

# OSU MPI Bandwidth Test v5.0
# Size      Bandwidth (MB/s)
1           1.77
2           3.54
4           7.22
8           14.74
16          26.96
32          57.22
64          114.45
128         232.93
256         405.45
512         753.79
1024        1386.73
2048        2264.14
4096        3389.89
8192        5612.19
16384       6375.83
32768       8975.61
65536       9558.37
131072      12061.21
262144      12222.53
524288      12306.10
1048576     12338.90
2097152     12354.31
4194304     12379.14
[ryoc010-n003 ~]$

```

Intel Omni-Path Fabric Performance (OSU benchmark)\*



\* Pre-production A0 hardware and Alpha-level software

[www.colfax-intl.com](http://www.colfax-intl.com)



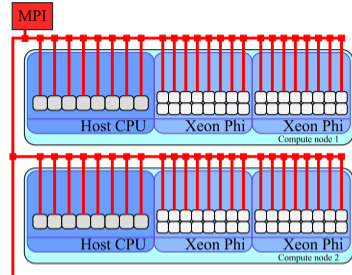
(slide from Colfax's presentation at SC'15)

# Inter-Operation with OpenMP

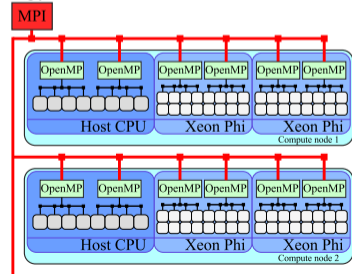
# Hybrid MPI+OpenMP

Using OpenMP inside of MPI processes:

- Reduces the memory footprint
- Decreases the number of MPI ranks, which reduces communication
- May incur thread synchronization overhead
- Optimal number of threads in MPI processes must be established empirically



VS.



# Hybrid MPI+OpenMP

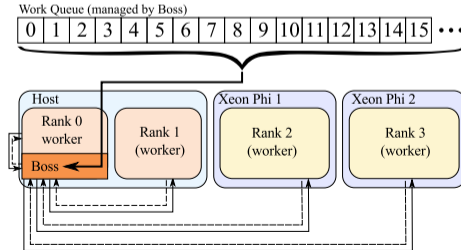
- For MPI calls from multiple MPI threads, use `-mt_mpi`
- MPI pins processes to cores and sets OpenMP affinity for them.
- To tune pinning: `I_MPI_PIN`, `I_MPI_PIN_DOMAIN`
- To diagnose process pinning: `I_MPI_DEBUG=4`
- More information in the [MPI Reference Manual](#)

# Multi-Threading within MPI Processes

```

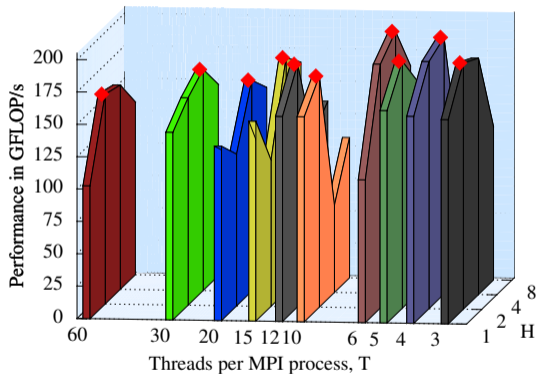
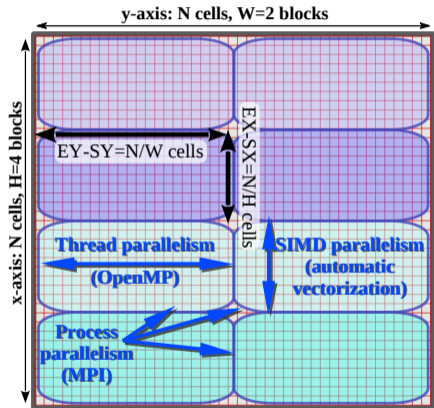
1  if(myRank == 0) { // Rank 0 has both a boss and a worker inside:
2      const int nThreads = omp_get_max_threads(); omp_set_nested(1);
3      #pragma omp parallel sections num_threads(2)
4          {
5          #pragma omp section
6              { DistributeWork(nOptions, option, mpiWorldSize); } // Boss
7          #pragma omp section
8              { omp_set_num_threads(nThreads-1); // Worker in rank 0:
9              ReceiveWork(option, payoff, myRank, optioncount); } // ...

```



# Example of OpenMP and MPI Inter-Operation

Number of threads per process may be a tuning parameter:



Case study: [this paper](#)

# Example: Asian Options, Heterogeneous Distributed Computing

# Example: the Monte Carlo Method of Asian Option Pricing

## 1) Simulate Random-Walk of Asset Price

$$dS(t) = \mu S(t)dt + \sigma S(t)dB(t)$$

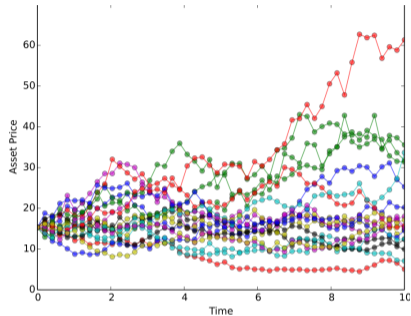
Using the Solution

$$S(t) = S(0)e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma\sqrt{t}N(0,1)}$$

## 2) Perform Asian Option Price Averaging

$$\langle S \rangle_{\text{arithm}} = \frac{1}{N} \sum_{i=0}^{N-1} S(t_i),$$

$$\langle S \rangle_{\text{geom}} = \exp\left(\frac{1}{N} \sum_{i=0}^{N-1} \log S(t_i)\right)$$



## 3) Compute Discounted Pay-off

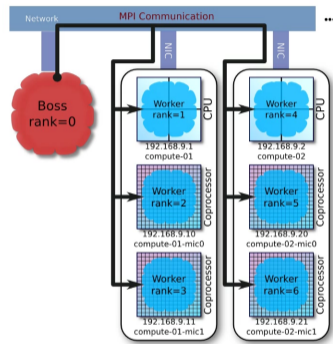
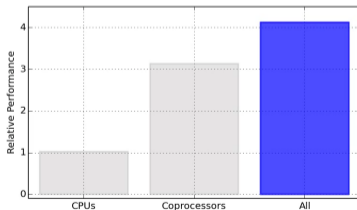
$$P_{\text{put}} = e^{-rT} \mathbb{E}(\max\{0; K - \langle S \rangle\}),$$

$$P_{\text{call}} = e^{-rT} \mathbb{E}(\max\{0; \langle S \rangle - K\})$$

More information: [this paper](#)

# Asian Option Pricing: Heterogeneous Clustering

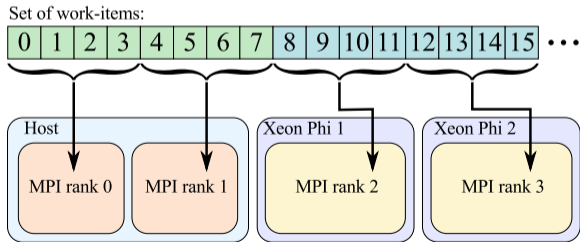
Heterogeneous Clustering with Homogeneous Code:  
Asian Option Pricing



<http://xeonphi.com/papers/heterogeneous>

# Heterogeneous Calculation without Load Balancing

```
1  const double optionsPerProcess = double(nOptions)/double(mpiWorldSize);  
2  const int myFirstOption = int(optionsPerProcess*(myRank));  
3  const int myLastOption = int(optionsPerProcess*(myRank+1));  
4  
5  // Static, even load distribution: assign options to ranks  
6  for (int i = myFirstOption; i < myLastOption; i++)  
7      ComputeOptionPayoffs(option[i], payoff[i]);
```

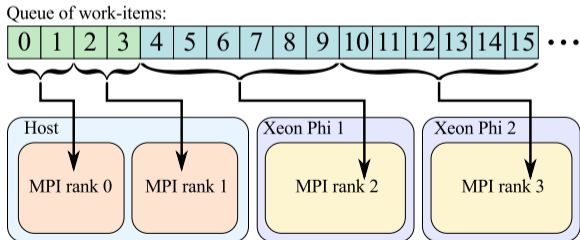


# Static Load Balancing

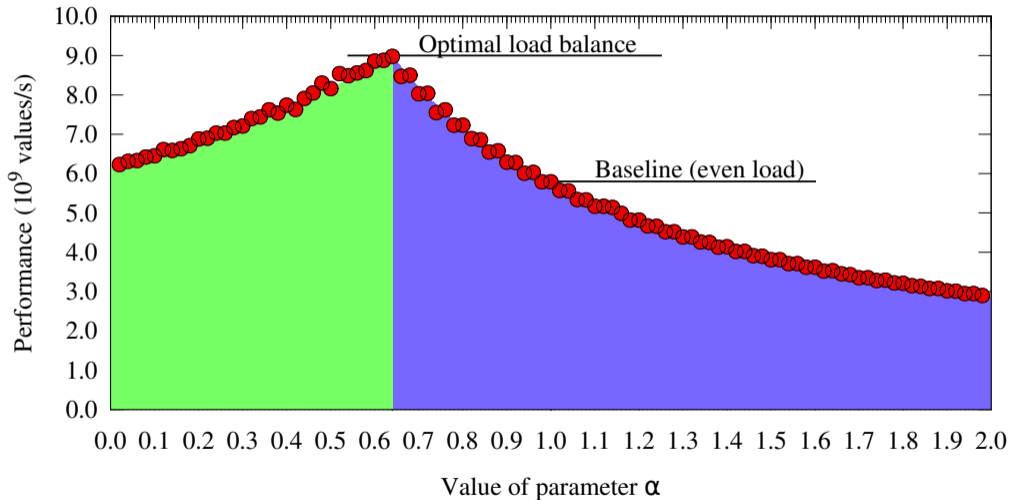
```

1  if (rankTypes[myRank] == 0) { // I am a MIC-based rank
2      double optionsPerProc = double(lastOptForCPUs)/double(cpuRanks.size());
3      myFirstOpt = int(optionsPerProc*(myGroupRank));
4      myLastOpt = int(optionsPerProc*(myGroupRank+1.0));
5  } else { // I am a CPU-based rank
6      double optionsPerProc = double(nOpts-lastOptForCPUs)/double(micRanks.size());
7      myFirstOpt=lastOptForCPUs+int(optionsPerProc*(myGroupRank));
8      myLastOpt=lastOptForCPUs+int(optionsPerProc*(myGroupRank+1.0)); }

```



# Static Load Balancing: Parameter Tuning



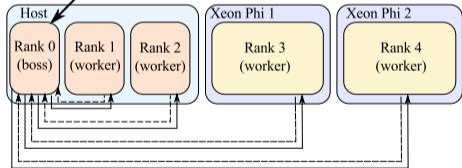
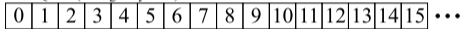
# Dynamic Load Balancing

```

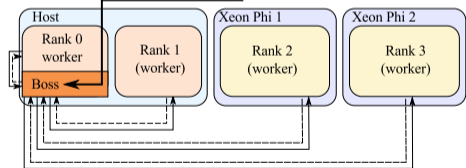
1  if (myRank == 0) // Boss's branch
2      DistributeWork(nOptions, option, mpiWorldSize);
3  else // Workers' branch
4      ReceiveWork(option, payoff, myRank);

```

Work Queue (managed by Boss)

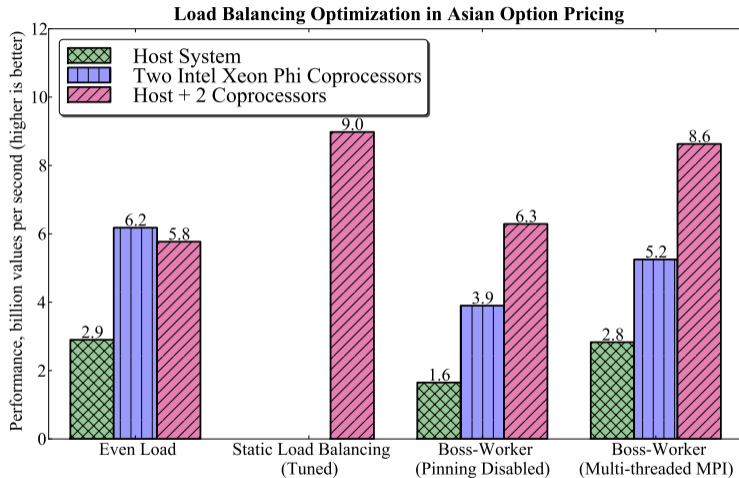


Work Queue (managed by Boss)



or

# Performance with Different Scheduling Modes



Refer to the book for explanation on the last two results.

## §3. Summary

# What We Learned

- Session 1 Native programming for Xeon Phi
- Session 2 Offload programming for Xeon Phi
- Session 3 Expressing vectorization
- Session 4 Expressing thread parallelism
- Session 5 Optimization overview (N-body)
- Session 6 Optimizing vectorization
- Session 7 Optimizing multi-threading (common errors)
- Session 8 Optimizing multi-threading (advanced tweaks)
- Session 9 Optimizing memory access
- Session 10 Distributed computing

Play It Again

# HOW Series Archive

## Your Turn

Let us know what you think via email!

[Learn More](#)

# HOW “Tools” 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

PARALLEL STUDIO XE

The graphic features a blue header with the text 'GOT THE TOOLS - NOW WHAT?'. Below this is a light-colored wood-grain background. On the left is a book cover for 'PARALLEL STUDIO XE' with the Intel logo. To the right of the book is the text 'Learn workflows and methodology with the “HOW” tools\* training and hands-on demos' and a list of tools: '\* Intel MKL | Intel Advisor | Intel VTune Amplifier'. At the bottom right, three silver wrenches are arranged vertically.

[colfaxresearch.com/how-tools-16-06](http://colfaxresearch.com/how-tools-16-06)

# Developer's Guide to Knights Landing



[colfaxresearch.com/knl-webinar/](http://colfaxresearch.com/knl-webinar/)

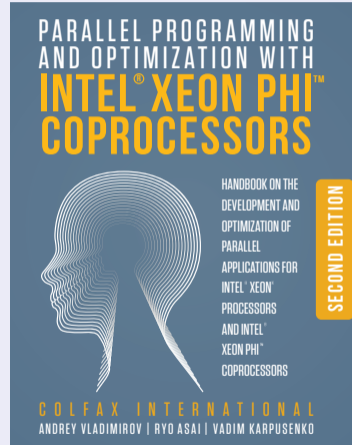
# Textbook

ISBN: 978-0-9885234-0-1 (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

## 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 an efficient programming for Intel architecture

**The Hands-On Workshop (HOW) Series**

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

### Parallel Programming Book

Introduction to parallel programming, deep discussion of optimization techniques, exercises.

© 2015, Colfax International. 508 pages.

### 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: Strip-Mining for Vectorization**



**Software Developer's Introduction to the HGST Ultrastar Archive HaaS 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 TX Acceleration May be Enough)**

### Featured Video

generated Additional Reading

has Research related to vectorization like a sharing code



<http://colfaxresearch.com/?p=704>

### Events

### Presentations

### Courses

## Consulting

Share

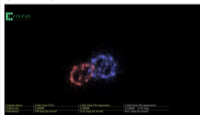


Colfax offers consulting services for enterprises, research help you to:

- Optimize your existing application to take advantage of parallelism, from vectors to cores to clusters and clouds
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor technologies
- Investigate the potential system configurations that satisfy your cost, power performance requirements.
- Take a clean sheet to develop a novel approach to solve your computing problem

All Video Courses - COP 901 - Chapter 2 - Episode 1.1

### Episode 2.1 - Purpose of the MIC architecture



Share

In this episode I will introduce you to the coprocessor based on the Intel Xeon Phi, an architecture and software stack for the specific of Parallel Computing.

### Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR Drives

Share



In this paper we will discuss the new HGST Ultrastar Archive HaaS SMR drives. Software Developer's Introduction to the HGST Ultrastar Archive HaaS SMR drives. High storage capacity, these drives are well suited for "long term archival" applications. In a public archive application, the data is frequently read but seldom modified.

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

Share



The Intel Xeon Phi has been designed, meaning that it is designed to be used in a cluster. In this publication we will discuss the network communication in a cluster. The goal of this paper is to provide a detailed description of the network configuration and benchmarks.

### Parallel Computing in the Search for New Physics at LHC

Share



Share

### Fluid Dynamics with Fortran on Intel Xeon Phi coprocessors

Share

In this document a Fortran program is used to solve the Navier-Stokes equations. The program is written in Fortran and is designed to run on Intel Xeon Phi coprocessors. The code is written in Fortran and is designed to run on Intel Xeon Phi coprocessors. The code is written in Fortran and is designed to run on Intel Xeon Phi coprocessors.



Share

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

Share

A few weeks ago we conducted an interview with James Reinders, the Intel and Intel Xeon Phi coprocessor lead. In this interview, we discussed the future of Intel MIC architecture, parallel programming, education, and more. James Reinders is the Intel Xeon Phi coprocessor lead and is also the Intel Xeon Phi coprocessor lead.



Share

James Reinders is the Intel Xeon Phi coprocessor lead and is also the Intel Xeon Phi coprocessor lead. In this interview, we discussed the future of Intel MIC architecture, parallel programming, education, and more. James Reinders is the Intel Xeon Phi coprocessor lead and is also the Intel Xeon Phi coprocessor lead.

<http://colfaxresearch.com/>