



# PROGRAMMING AND OPTIMIZATION FOR INTEL<sup>®</sup> ARCHITECTURE

The Hands-On Workshop (HOW) Series  
Session 10

*Colfax International — [colfaxresearch.com](http://colfaxresearch.com)*

October 2016

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.

- ▶ HOW to Program Intel Architecture
  - 01. Parallelism, specialization, guided tour – Oct 24
  - 02. Programming Intel Xeon Phi (KNC, KNL) – Oct 25
- ▶ HOW to Express Parallelism
  - 03. Automatic vectorization – Oct 26
  - 04. Multi-threading with OpenMP – Oct 27
- ▶ HOW to Get Performance
  - 05. Comprehensive demo – Oct 28
  - 06. Scalar & vectorization tuning – Oct 31
  - 07. Multi-threading: common issues – Nov 1
  - 08. Multi-threading: memory aspect – Nov 2
  - 09. Memory traffic – Nov 3
- ▶ HOW to Scale
  - 10. Distributed Computing: MPI – Nov 4

October 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					
November 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			
■ — Webinar+remote access						

Course page: [colfaxresearch.com/how-16-10](http://colfaxresearch.com/how-16-10)

- ▶ 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-16-10](http://colfaxresearch.com/how-16-10)



## Forums (technical):

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

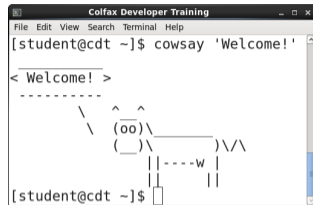
A screenshot of the Colfax Research website. At the top right, there is a "Log In/Register" link. The main header features the "COLFAX RESEARCH" logo in green, with the tagline "CONTRIBUTING TO INNOVATIONS IN COMPUTING" below it. A dark navigation bar contains the following menu items: READ, WATCH, LEARN, FORUMS (highlighted in green), CONNECT, and JOIN. Below the navigation bar, the heading "Join the Conversation" is displayed. The text below reads: "Welcome to Colfax Research forums, an online community for you to engage with HPC experts, software architects, developers, computational researchers, scientists, students and more—so you can acquire new knowledge, share ideas, and build new relationships." A paragraph follows: "Tap our experts and your peers to help meet the challenge of optimizing applications on modern hardware. This is the place to browse or post questions (and get answers) related to computational science, parallel programming and code modernization on Intel® Architecture." The final line says: "Welcome aboard. Post questions today!"

## Email (organizational):

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

# HANDS-ON EXERCISES AND REMOTE ACCESS

- ▶ 96 people receive a remote access token
- ▶ Virtualized Intel Xeon CPU, real Intel Xeon Phi coprocessor (1st gen, KNC), SW tools
- ▶ Can access the system the entire 2 weeks of the workshop



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

[student@cdt ~]$
```

- ▶ 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



## **§2. DISTRIBUTED COMPUTING**

## Computing Platforms



Workstations

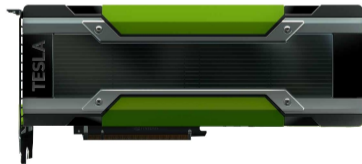


Servers

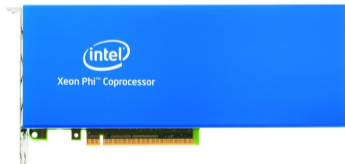


Clusters

## Computing Accelerators



GPGPUs

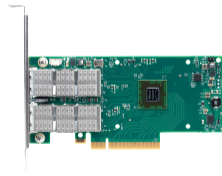
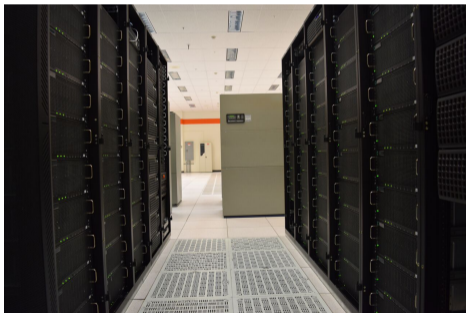


Coprocessors



FPGAs

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>
3 int main (int argc, char *argv[]) {
4     MPI_Init (&argc, &argv); // Initialize MPI environment
5     int rank, size, namelen;
6     char name[MPI_MAX_PROCESSOR_NAME];
7     MPI_Comm_rank (MPI_COMM_WORLD, &rank); // ID of current process
8     MPI_Get_processor_name (name, &namelen); // Hostname of node
9     MPI_Comm_size (MPI_COMM_WORLD, &size); // Number of processes
10    printf ("Hello World from rank %d running on %s!\n", rank, name);
11    if (rank == 0) printf("MPI World size = %d processes\n", size);
12    MPI_Finalize (); // Terminate MPI environment
13 }
```

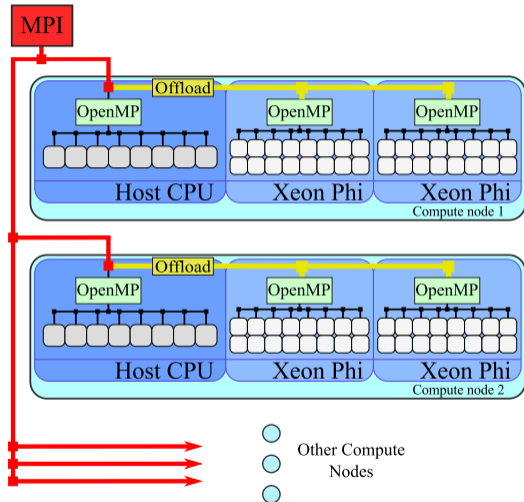
# COMPILING AND RUNNING MPI APPLICATIONS ON HOST

```
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

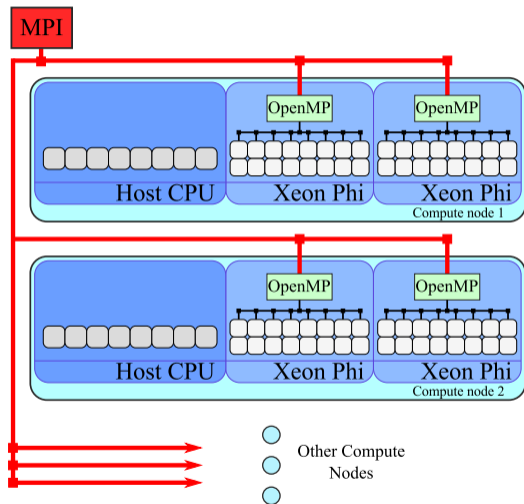
# SCALING ACROSS A CLUSTER WITH COPROCESSORS

- ▶ MPI processes only on CPUs
- ▶ Divide data between coprocessors
- ▶ Concurrent offload from multiple host threads
- ▶ Synchronize data between nodes with MPI



# SCALING ACROSS A CLUSTER WITH COPROCESSORS WITH MPI

- ▶ MPI processes only on CPUs
- ▶ Divide 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

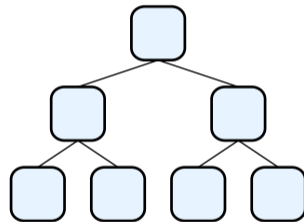
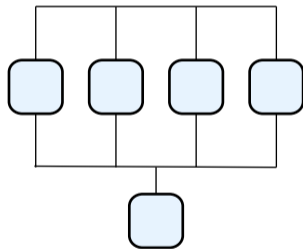
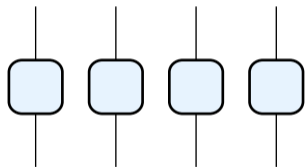
1. Compile and link with the MPI wrapper of the compiler:
  - `mpiicc` for C,
  - `mpiicpc` for C++,
  - `mpiifort` for Fortran 77 and Fortran 95.
2. Set up MPI environment variables *and* `I_MPI_MIC=1`
3. NFS-share or copy the MPI library and the application executable to the coprocessors
4. Launch 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.

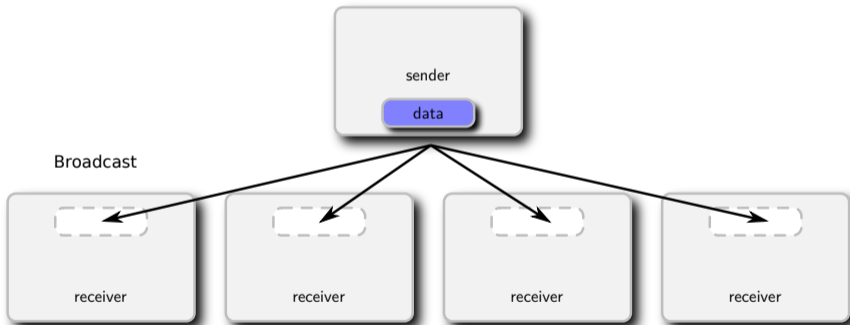


# POINT TO POINT COMMUNICATION

```
1  if (rank == sender) {
2
3     char outgoingMsg[messageLength];
4     strcpy(outgoingMsg, "Hi There!");
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 }
```

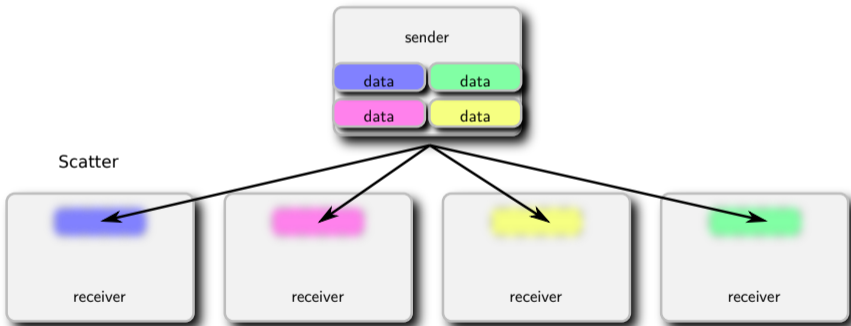
# COLLECTIVE COMMUNICATION: BROADCAST

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



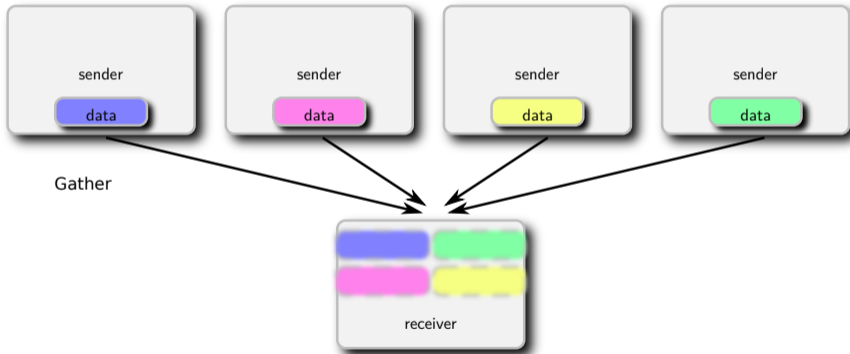
# COLLECTIVE COMMUNICATION: SCATTER

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



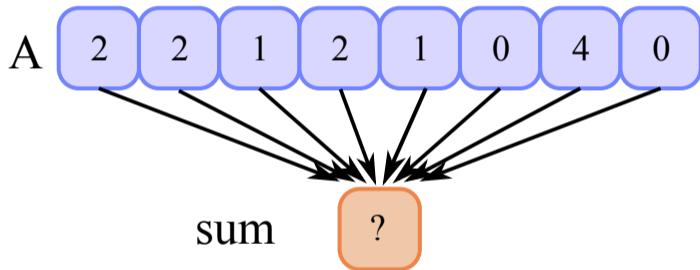
# COLLECTIVE COMMUNICATION: GATHER

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

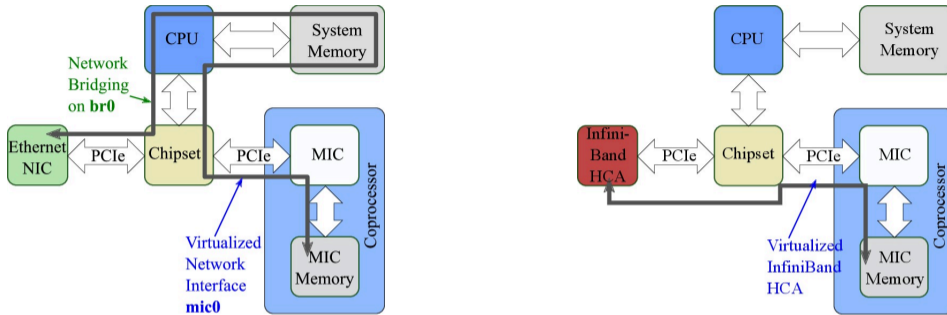


Available reducers: max/min, minloc/maxloc, sum, product, AND, OR, XOR (logical or bitwise).



## **COPROCESSORS AND INFINIBAND**

# PEER-TO-PEER COMMUNICATION BETWEEN COPROCESSORS

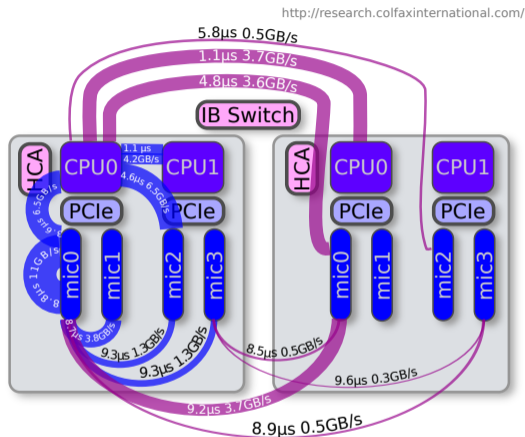


- ▶ Left: Gigabit Ethernet bridging on host allows to place coprocessors on the same subnet as hosts
- ▶ Right: Coprocessor Communication Link (CCL) – virtualization of an InfiniBand device on each coprocessor

# MPI FABRIC SELECTION

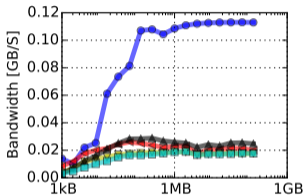
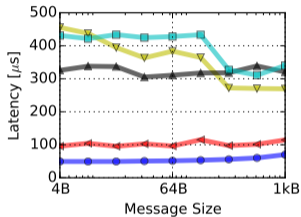
- ▶ MPI communication between CPU and coprocessors as efficient as offload
- ▶ Peer-to-peer communication not uniform, but better than with Gigabit Ethernet
- ▶ Control: environment variable `I_MPI_FABRICS`

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

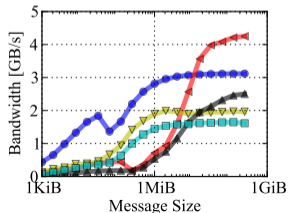
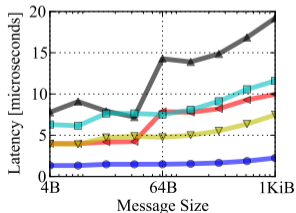
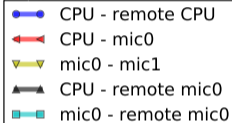


# GIGABIT ETHERNET VERSUS INTEL TRUE SCALE INTERCONNECTS

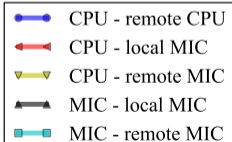
- ▶ Ethernet+TCP between coprocessors slower than the hardware limit
- ▶ InfiniBand approaches the hardware limit from CPU to coprocessors



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

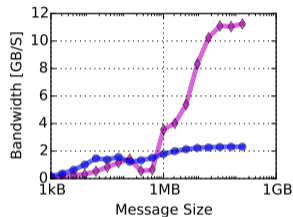
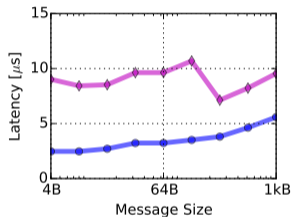


**TMI Fabric on True Scale HCA**

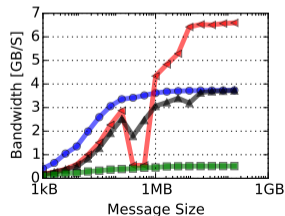
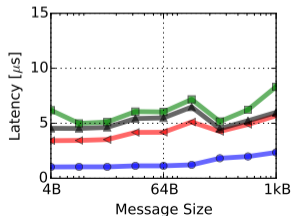
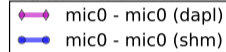


# INTRA-DEVICE AND INTRA-NODE COMMUNICATION

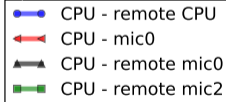
- Fabrics for messages within a single coprocessor:
- shm provides better latency, dap1 – greater bandwidth



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



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



# INTEL OMNI-PATH ARCHITECTURE (OPA)

## Intel® Omni-Path Fabric

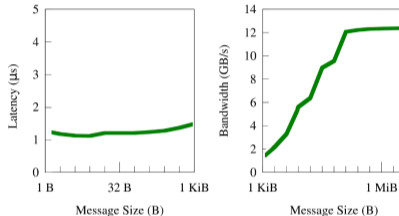
```

File Edit View Search Terminal Help
[ryo@c010-n003 ~]$ cat /sys/module/hfi1/parameters/descct_intr
64
[ryo@c010-n003 ~]$ cat /sys/module/hfi1/parameters/sdma_descq_cnt
2048
[ryo@c010-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
[ryo@c010-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)

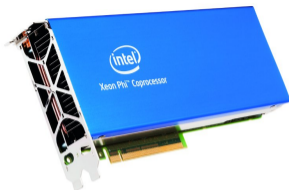


(slide from Colfax's presentation at SC'15)

# INTEL XEON PHI PROCESSORS WITH INTEGRATED FABRIC

## KNLF: KNL with Fabric

- ▶ Fabric integrated on CPU
  - Intel Omni-Path Architecture
- ▶ Socket mount processor



\*KNC image

## KNL Coprocessor

- ▶ PCIe add-in card
  - Requires host
- ▶ Multiple KNLs in a system

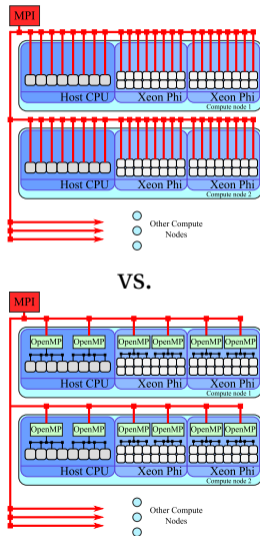


## **INTER-OPERATION WITH OPENMP**

# 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



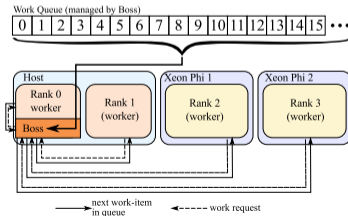
- ▶ For MPI calls from multiple MPI threads, use `-mt_mpi`
- ▶ MPI pins processes to cores and sets OpenMP affinity for them.
- ▶ To tune pinning: `I_MPI_PIN, I_MPI_PIN_DOMAIN`
- ▶ To diagnose process pinning: `I_MPI_DEBUG=4`
- ▶ More information in the [MPI Reference Manual](#)

# MULTI-THREADING WITHIN MPI PROCESSES

```

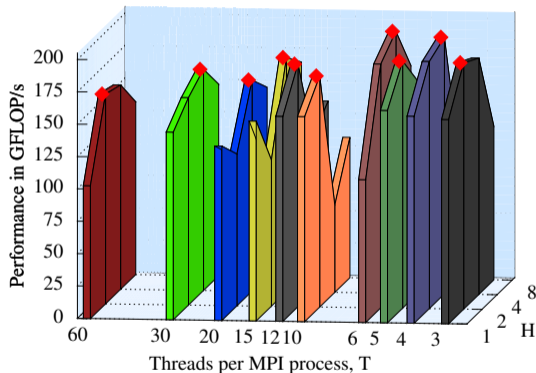
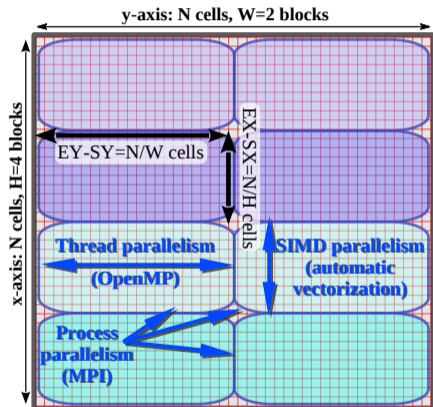
1  if(myRank == 0) { // Rank 0 has both a boss and a worker inside:
2      const int nThreads = omp_get_max_threads();  omp_set_nested(1);
3      #pragma omp parallel sections num_threads(2)
4          {
5      #pragma omp section
6          { DistributeWork(nOptions, option, mpiWorldSize); } // Boss
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

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

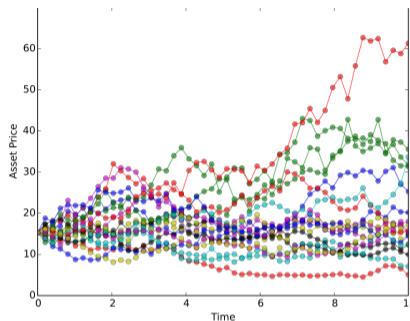
Using the Solution

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

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

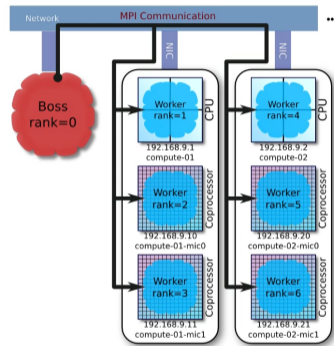
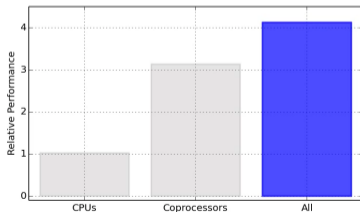
$$P_{\text{put}} = e^{-rT} \mathbb{E} (\max \{0; K - \langle S \rangle\}),$$

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

More information: [this paper](#)

# ASIAN OPTION PRICING: HETEROGENEOUS CLUSTERING

Heterogeneous Clustering with Homogeneous Code:  
Asian Option Pricing

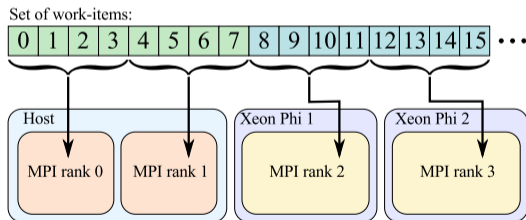


<http://xeonphi.com/papers/heterogeneous>

EXAMPLE: ASIAN OPTIONS, HETEROGENEOUS DISTRIBUTED COMPUTING

# 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]);
```

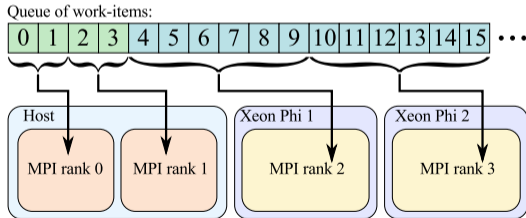


# STATIC LOAD BALANCING

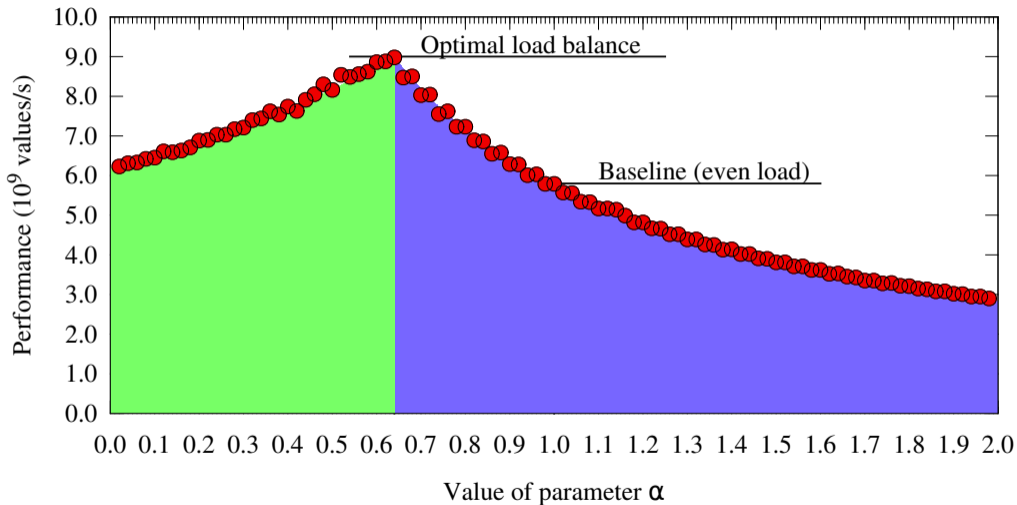
```

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



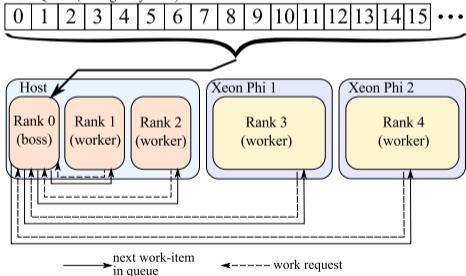
# DYNAMIC LOAD BALANCING

```

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

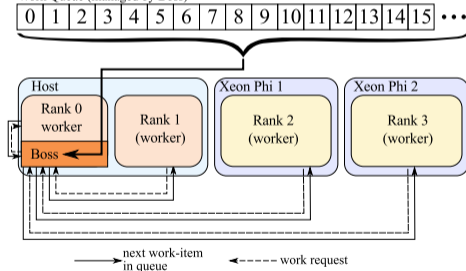
```

Work Queue (managed by Boss)

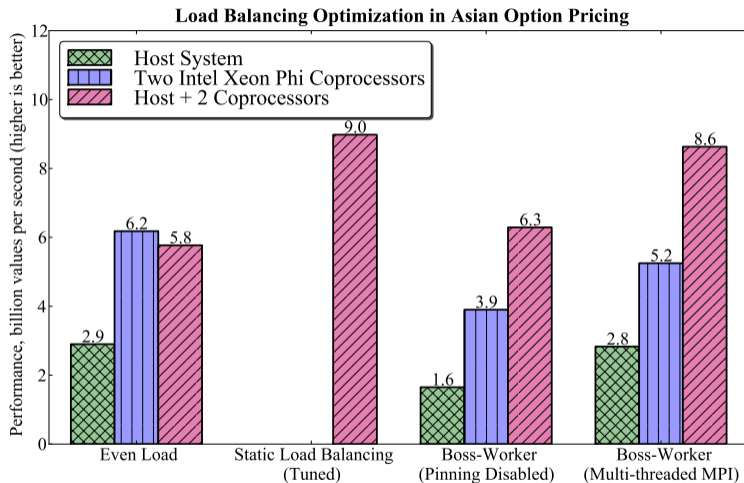


or

Work Queue (managed by Boss)



# 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

# HOW Series Archive

Let us know what you think via email!



**LEARN MORE**

# HOW SERIES: KNIGHTS LANDING

HOW SERIES "KNIGHTS LANDING":

PROGRAMMING AND OPTIMIZATION FOR  
INTEL XEON PHI X200 FAMILY

Free 2-hour video course



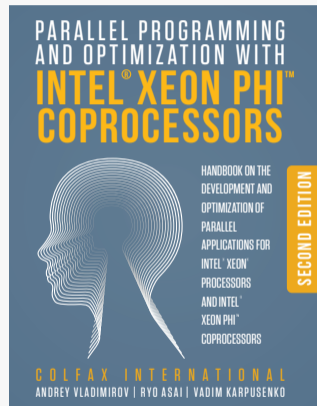
[colfaxresearch.com/how-knl/](http://colfaxresearch.com/how-knl/)

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

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**

### 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: False Sharing and Padding**

**Software Developer's Introduction to the HGST Ultrastar Archive H700 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 ix Acceleration May Be Enough)**

### Featured Video

See Research material on vectorization in a streaming video

▶

[View Full Screen](#)

## Consulting

Facebook Twitter LinkedIn Google+

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
- Future-proof your application for upcoming innovations
- Accelerate your application using coprocessor tech
- Investigate the potential system configurations that satisfy your cost, power, performance requirements.
- Take a clean slate to develop a novel approach to reduce your computing pro

48 Videos (Courses: 108 VHS | Chapters: 1, Episode 1.1)

**Episode 2.1 — Purpose of the MIC architecture**

▶

▶

▶

▶

▶

▶

▶

### Software Developer's Introduction to the HGST Ultrastar Archive H700 SMR Drives

This paper will discuss the new HGST Ultrastar Archive H700 SMR drives, which are the first 7.2 TB drives to be available in a 3.5-inch form factor. These drives are well suited for high volume and/or high capacity applications. These drives are well suited for high volume and/or high capacity applications. These drives are well suited for high volume and/or high capacity applications.

▶

▶

▶

▶

▶

▶

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

At the presentation, a Colfax researcher presents a series of slides on how to use Fortran on Intel Xeon Phi coprocessors. The slides focus on the use of Fortran on Intel Xeon Phi coprocessors. The slides focus on the use of Fortran on Intel Xeon Phi coprocessors. The slides focus on the use of Fortran on Intel Xeon Phi coprocessors.

▶

▶

▶

▶

▶

▶

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

This paper reports on the configuration and benchmarks of a cluster of Intel Xeon Phi coprocessors connected via Gigabit Ethernet and InfiniBand. The paper reports on the configuration and benchmarks of a cluster of Intel Xeon Phi coprocessors connected via Gigabit Ethernet and InfiniBand. The paper reports on the configuration and benchmarks of a cluster of Intel Xeon Phi coprocessors connected via Gigabit Ethernet and InfiniBand.

▶

▶

▶

▶

▶

▶

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

A few weeks ago we conducted an interview with James Reinders, the Director of Intel's Parallel Computing Group. In this interview, we discuss the future of Intel MIC architecture, parallel programming, and education. A few weeks ago we conducted an interview with James Reinders, the Director of Intel's Parallel Computing Group. In this interview, we discuss the future of Intel MIC architecture, parallel programming, and education.

▶

▶

▶

▶

▶

▶

http://colfaxresearch.com/