



PROGRAMMING AND OPTIMIZATION FOR INTEL[®] ARCHITECTURE

Hands-On Workshop (HOW) Series "Deep Dive"
Session 5

Colfax International — colfaxresearch.com

April 2017

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.

- ▶ **Module I. Programming**
 - 01. Intel Architecture and Modern Code – Apr 17
 - 02. Xeon Phi, Coprocessors, Omni-Path – Apr 18
- ▶ **Module II. Expressing Parallelism**
 - 03. Automatic vectorization – Apr 19
 - 04. Multi-threading with OpenMP – Apr 20
 - 05. Distributed Computing, MPI – Apr 21
- ▶ **Module III. Optimization**
 - 06. Optimization Overview: N-body – Apr 24
 - 07. Scalar tuning, Vectorization – Apr 25
 - 08. Common Multi-threading Problems – Apr 26
 - 09. Multi-threading, Memory Aspect – Apr 27
 - 10. Access to Caches and Memory – Apr 28

April 2017						
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-17-04

- ▶ Slides
- ▶ Code
- ▶ Video
- ▶ Chat

More workshops:

colfaxresearch.com/training




GET YOUR QUESTIONS ANSWERED: CHAT



colfaxresearch.com/how-17-04

GET YOUR QUESTIONS ANSWERED: FORUMS

	READ	WATCH	LEARN	FORUMS	CONNECT	JOIN
---	------	-------	-------	---------------	---------	------

Forum

Colfax Cluster
Discussion of Colfax Cluster usage policies, troubleshooting.

Developer Training, HOW Series
Questions about any of the Colfax trainings? Usage of training servers, experience with specific exercises, inquiries on what's inside, suggestions for future trainings - post them here.

Performance Optimization and Parallelism
Discuss with Colfax Research and colleagues any topics related to computational science, parallel programming, performance optimization and code modernization.

colfaxresearch.com/discussion

- ▶ 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





§2. DISTRIBUTED COMPUTING

Computing Platforms



Workstations

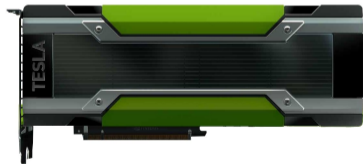


Servers

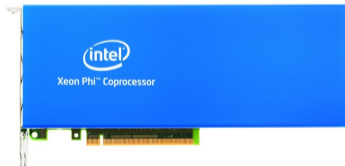


Clusters

Computing Accelerators



GPGPUs

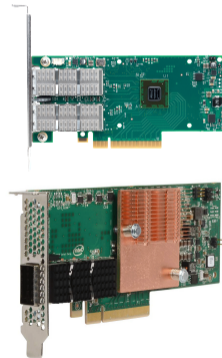
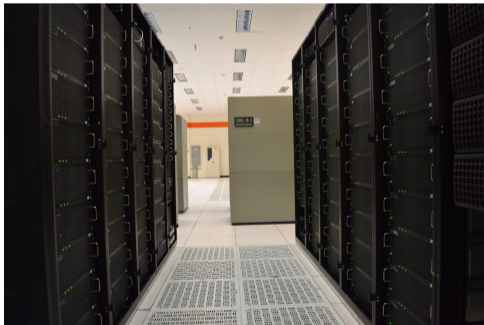


Coprocessors



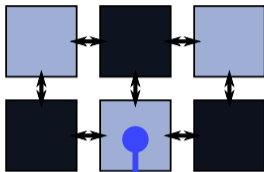
FPGAs

Clusters often use Gigabit Ethernet for administration and InfiniBand or Intel Omni-Path for communication.



PARALLEL PROGRAMMING LAYERS

CLUSTER COMPUTING
in distributed memory



```
MPI_Sendrecv(data, k,
             MPI_DOUBLE, data2,
             ... );
```

MULTITHREADING
in shared memory



```
#pragma omp parallel for
for (j = 0; j < m; j++)
    ComputeSubset(j);
```

VECTORIZATION
of floating-point math



```
#pragma omp simd
for (i = 0; i < n; i++)
    A[i] += B[i];
```



MPI AND HETEROGENEOUS COMPUTING



- Specification for message passing
- Multiple implementations exist

- Portable
- Efficient
- Designed for computing

- Distributed-memory computing
- Multiprocessing in shared memory

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 }
```

MPICH site contains a list of [MPI 3.2 routines](#)

COMPILING AND RUNNING MPI APPLICATIONS ON LOCALHOST

```
u100@c005% mpiicpc -o HelloMPI HelloMPI.cc
```

Command file mympi:

```
#PBS -l nodes=1  
cd $PBS_O_WORKDIR  
mpirun -host localhost -np 2 ./HelloMPI
```

Results:

```
u100@c005% qsub mympi  
2000  
u100@c005% cat mympi.o2000  
Hello World from rank 1 running on c005-n001!  
Hello World from rank 0 running on c005-n001!  
MPI World size = 2 processes
```

RUNNING MPI APPLICATIONS ON SEVERAL HOSTS

Command file mydistmpi:

```
#PBS -l nodes=2
cd $PBS_O_WORKDIR
cat $PBS_NODEFILE
mpirun -machinefile $PBS_NODEFILE ./HelloMPI
```

```
u100@c005% qsub mydistmpi
2001
u100@c005% cat mydistmpi.o2001
c005-n001
c005-n002
Hello World from rank 1 running on c005-n002!
Hello World from rank 0 running on c005-n001!
MPI World size = 2 processes
```

COMPILING AND RUNNING NATIVE MPI APPLICATIONS ON COPROCESSORS

Compilation

```
u100@c005% mpiicpc -mmic -o HelloMPI.MIC HelloMPI.c
```

Command file mymic:

```
#PBS -l nodes=1:coprocessor  
cd $PBS_O_WORKDIR  
scp HelloMPI.MIC mic0:~/  
export I_MPI_MIC=1  
mpirun -host mic0 -np 2 ~/HelloMPI.MIC
```

Results:

```
Hello World from rank 1 running on c005-n001-mic0!  
Hello World from rank 0 running on c005-n001-mic0!  
MPI World size = 2 processes
```

HETEROGENEOUS MPI APPLICATIONS: HOST + COPROCESSORS

Command file myhet:

```
#PBS -l nodes=1:coprocessor
cd $PBS_O_WORKDIR
scp HelloMPI.MIC mic0:~/
export I_MPI_MIC=1
mpirun -host localhost -np 1 ./HelloMPI : -host mic0 -np 1 ~/HelloMPI.MIC
```

Results:

```
Hello World from rank 0 running on c005-n001!
Hello World from rank 1 running on c005-n001-mic0!
MPI World size = 2 processes
```

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

HETEROGENEOUS MPI APPLICATIONS: MACHINE FILE

Xeon Phi coprocessors may be configured to be IP-addressable on cluster network and share file system paths with hosts.

Machine file `hosts.txt`:

```
c005-n101:1
c005-n102:1
c005-n101-mic0:1
c005-n102-mic0:1
```

Job submission:

```
vega@lyra% export I_MPI_MIC_POSTFIX=.MIC
vega@lyra% mpirun -machinefile hosts.txt ~/Hello
```

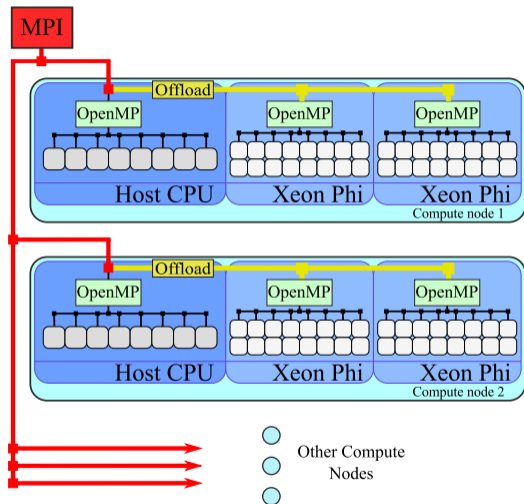
- ▶ Specify Xeon executable for host processes
- ▶ MIC executable obtained by appending `I_MPI_MIC_POSTFIX`

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. For Xeon Phi coprocessors (KNC): `I_MPI_MIC=1`
3. Launch with the tool `mpirun`
 - Colon-separated list of hosts (`-host hostname`),
 - Alternatively, `-machinefile $PBS_NODEFILE`

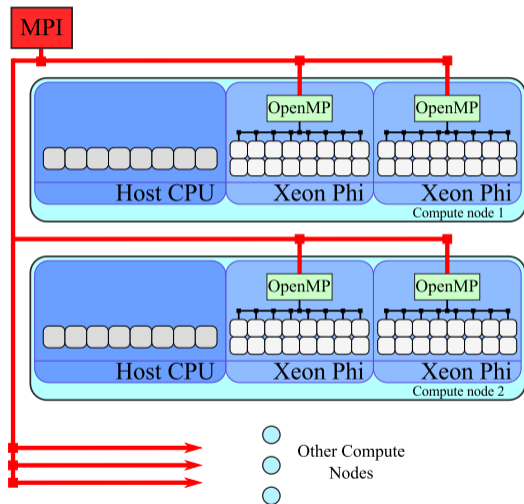
SCALING ACROSS A CLUSTER WITH COPROCESSORS

- ▶ 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





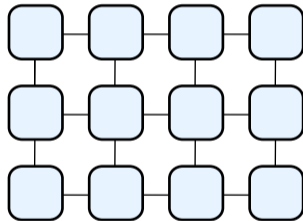
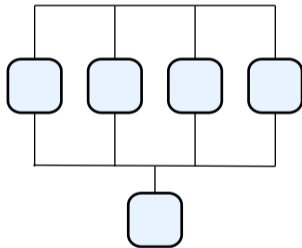
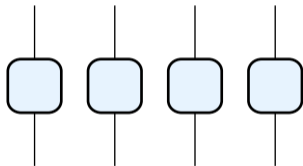
COMMUNICATION PATTERNS

POINT TO POINT COMMUNICATION

```
1  if (rank == sender) {  
2  
3  char outMsg[msgLen];  
4  strcpy(outMsg, "Hi There!");  
5  MPI_Send(&outMsg, msgLen, MPI_CHAR, receiver, tag, MPI_COMM_WORLD);  
6  
7  } else if (rank == receiver) {  
8  
9  char inMsg[msgLen];  
10 MPI_Recv (&inMsg, msgLen, MPI_CHAR, sender, tag, MPI_COMM_WORLD, &stat);  
11 printf ("Received message with tag %d: '%s'\n", tag, inMsg);  
12  
13 }
```

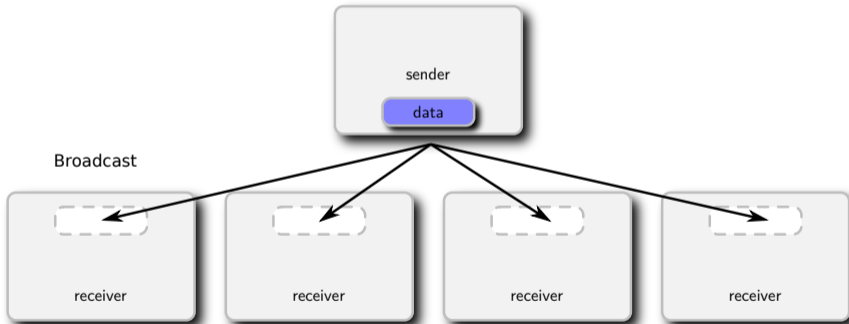
PARALLEL PATTERNS

Common parallel patterns call for collective communication



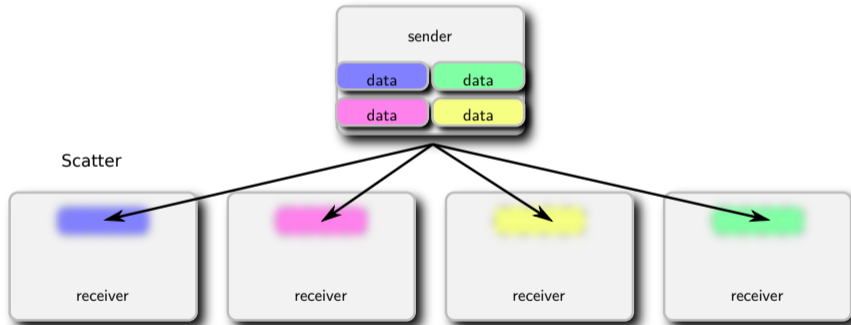
COLLECTIVE COMMUNICATION: BROADCAST

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



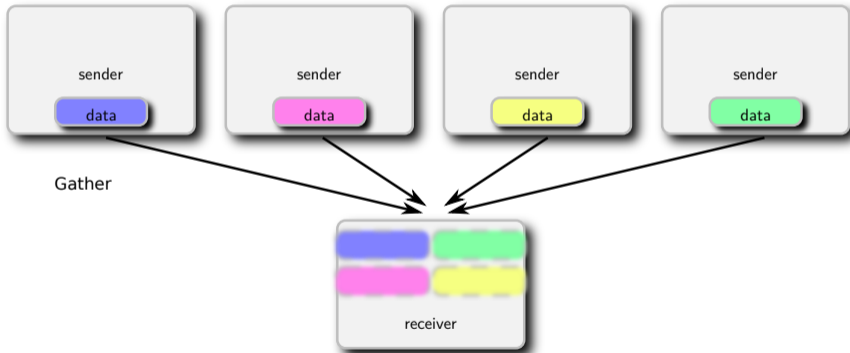
COLLECTIVE COMMUNICATION: SCATTER

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



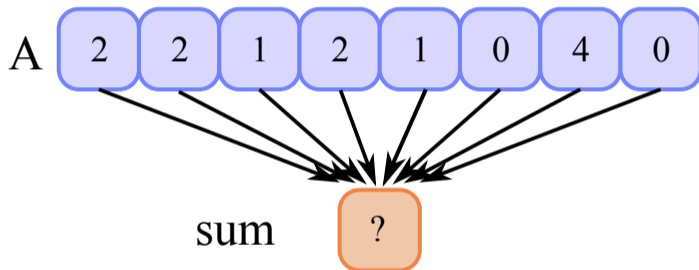
COLLECTIVE COMMUNICATION: GATHER

```
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);
```



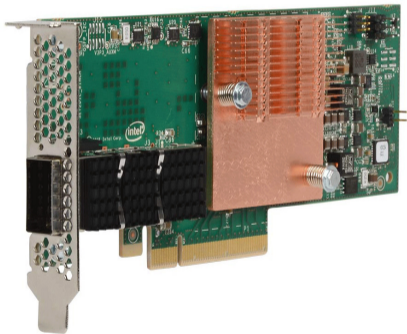
Available reducers: max/min, minloc/maxloc, sum, product, AND, OR, XOR (logical or bitwise).



INTEL OMNI-PATH ARCHITECTURE

INTEL'S HPC COMMUNICATION FABRIC

Intel Omni-Path Architecture - low-latency, high-bandwidth, scalable communication fabric for HPC applications.



Discrete

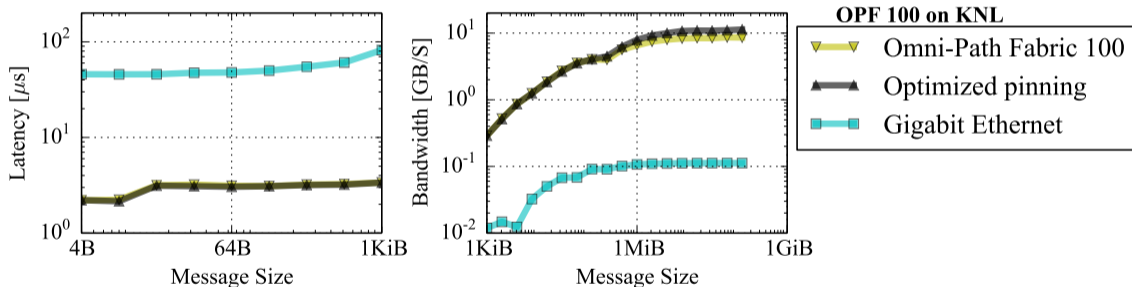


Integrated

INTEL OMNI-PATH FABRIC 100 WITH INTEL XEON PHI PROCESSORS (KNL)

Switching between OPA and 1GbE on Colfax Cluster:

```
I_MPI_FABRICS=tmi mpirun -machinefile $PBS_NODEFILE IMB-MPI1 PingPong
I_MPI_FABRICS=tcp mpirun -machinefile $PBS_NODEFILE IMB-MPI1 PingPong
```



Optimal pinning `I_MPI_PIN_PROCESSOR_LIST=7` may be done automatically by a future version of Intel MPI

INTEL OMNI-PATH FABRIC 100 WITH INTEL XEON PROCESSORS

First generation: 100 Gbps bandwidth, \approx 1 microsecond latency

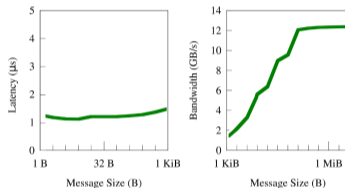
Intel® Omni-Path Fabric

```

File Edit View Search Terminal Help
[ryoc010-n003 ~]# cat /sys/module/hfi1/parameters/descr_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     12386.10
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

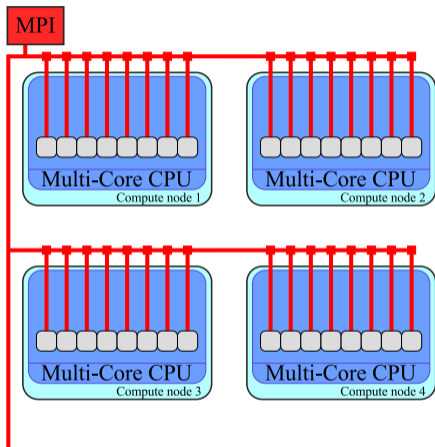
www.colfax-intl.com



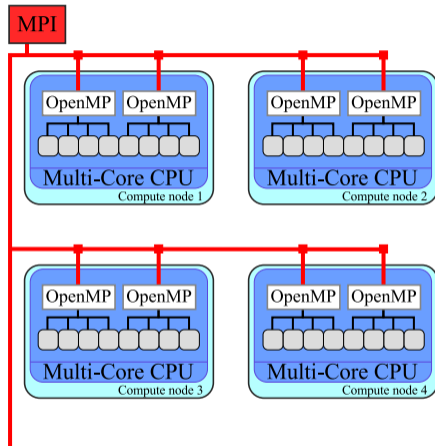
- ▶ Rely on MPI for platform-independent communication
- ▶ Intel MPI: set `I_MPI_FABRICS=tmi`.



INTER-OPERATION WITH OPENMP



Works for low core counts

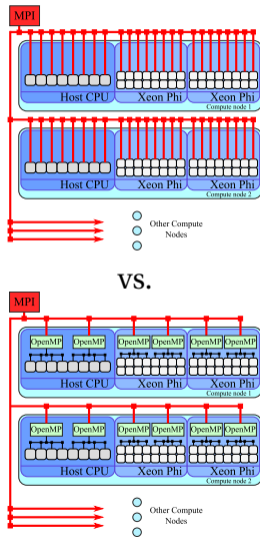


Necessary for multi-core CPUs

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



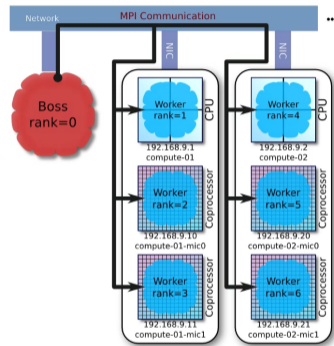
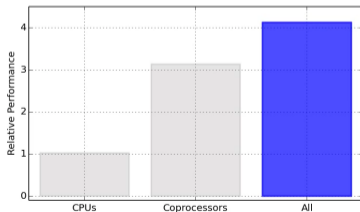
- ▶ MPI pins processes to cores and sets OpenMP affinity for them.
- ▶ To tune pinning: `I_MPI_PIN, I_MPI_PIN_DOMAIN`
- ▶ To diagnose process pinning: `I_MPI_DEBUG=4`
- ▶ For MPI calls from multiple OpenMP threads, use `-mt_mpi`
- ▶ More information in the [MPI Reference Manual](#)



LOAD BALANCING

ASIAN OPTION PRICING: HETEROGENEOUS CLUSTERING

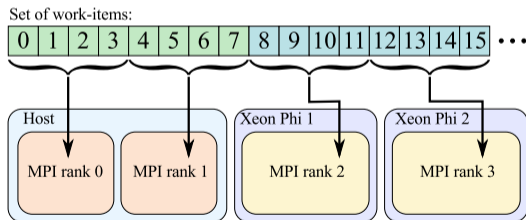
Heterogeneous Clustering with Homogeneous Code:
Asian Option Pricing



<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]);
```

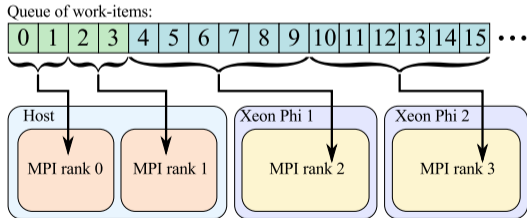


STATIC LOAD BALANCING

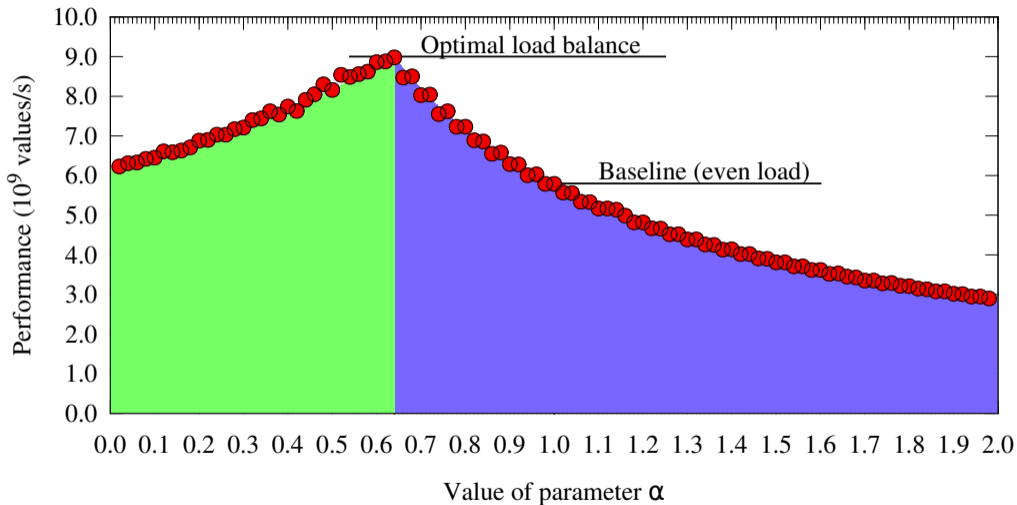
```

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



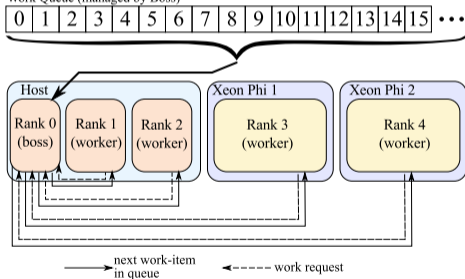
DYNAMIC LOAD BALANCING

```

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

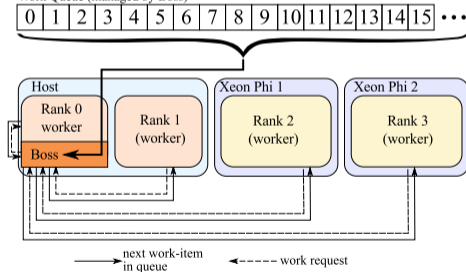
```

Work Queue (managed by Boss)

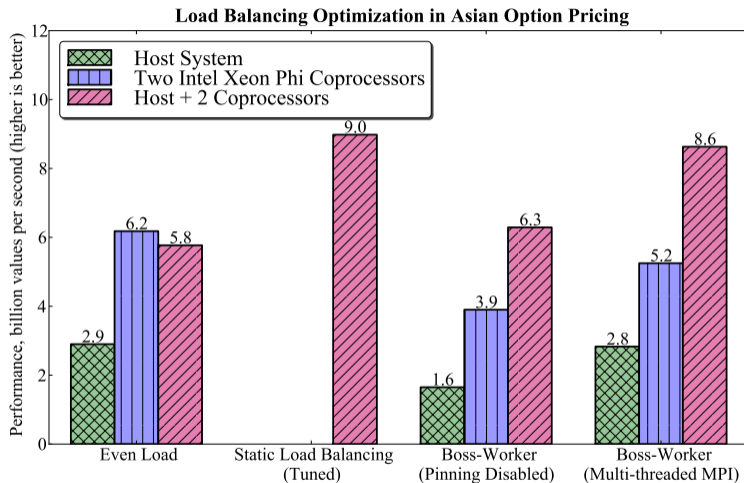


OR

Work Queue (managed by Boss)



PERFORMANCE WITH DIFFERENT SCHEDULING MODES



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



INTEL MPI PERFORMANCE SNAPSHOT

USING INTEL MPI PERFORMANCE SNAPSHOT

Part of Intel Trace Analyzer and Collector (ITAC), invoke with `-mps` argument of `mpirun`:

```
#PBS -l nodes=4

cd $PBS_O_WORKDIR
source /opt/intel/itac_latest/bin/mpsvars.sh
mpirun -mps -machinefile $PBS_NODEFILE ./myApplication
```

Produces `stat_*` directory with `.bin` files in it. Analyze them with `mps`.

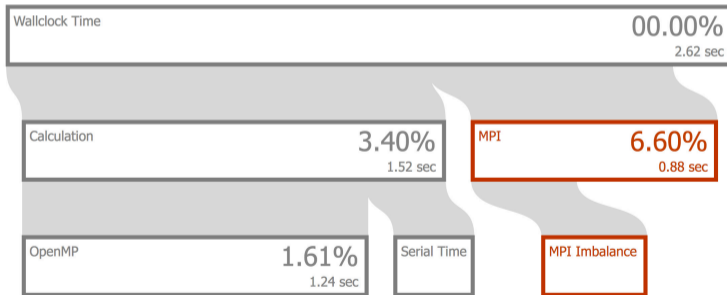
```
u111@c005% ls ./stat_*
/home/u111/myproject/stat_20170119-222922:
stat-0.bin  stat-1.bin  stat-2.bin  stat-3.bin
u111@c005% mps stat_20170119-222922
u111@c005% mps -g stat_20170119-222922
```



MPI Performance Snapshot

Your application is MPI Bound.
High MPI imbalance.
Use [Intel® Trace Analyzer and Collector](#) for further analysis.

Application: /home/u111/HOW-Series-Labs/4/4.01-overview-nbody/solutions
/instruction-05/app-KNL
Number of ranks: 4
Used statistics: /home/u111/stat_20170119-222922/
Creation date: 2017-01-19 22:29:23





§3. SUMMARY

MPI CONCEPTS AND FUNCTIONS

Communication modes – (non-)blocking, (a)synchronous, ready mode

Message buffers – application, system, user space

Communicators, Groups – creation, manipulation, usage

Data types – built-in, derived

Collective communication – patterns

Hybrid programming – co-existence with threads: safety, performance

One-sided communication – remote memory access

Click for links from the [MPI Forum](#).

More information in [our book](#) and [MPI tutorial from the LLNL](#)

SUMMARY ON MPI

- ▶ Framework for distributed-memory programming
- ▶ Hides from developer complexity of programming a variety of fabrics
- ▶ Collective communication may use functions of the fabric
- ▶ Intel Omni-Path Architecture is a native solution for Xeon Phi
- ▶ Intel tool for tuning load balance, communication: ITAC

Next session: performance optimization: overview, case study

MC² SERIES LEARN HOW THEY DID IT



Scientists, engineers and developers are achieving breakthrough computational performance through code modernization.

Join us and hear experts discuss performance optimization methods used in real-life applications in the all-new Modern Code Contributed (MC²) Talks - a series of free webinars.

[Learn from the experts - register now >](#)

mc2series.com

