PROGRAMMING AND OPTIMIZATION FOR INTEL® ARCHITECTURE

The Hands-On Workshop (HOW) Series

Session 5

Colfax International — *colfaxresearch.com*

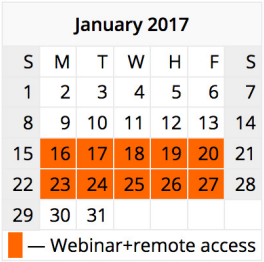
January 2017

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

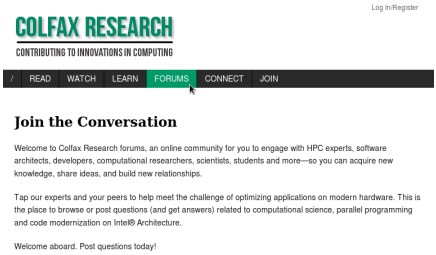
- Slides
- Code
- Video
- Chat

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

§2. DISTRIBUTED COMPUTING

COMPUTING HARDWARE

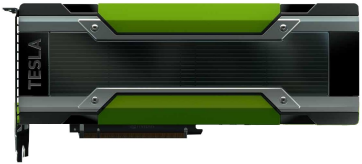
Computing Platforms

Workstations

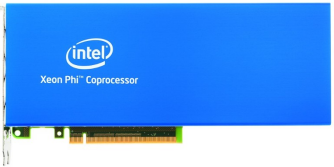
Servers

Clusters

Computing Accelerators



GPGPUs

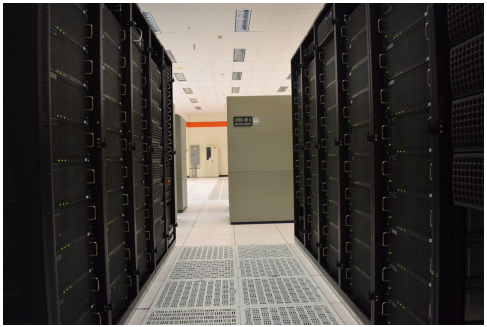


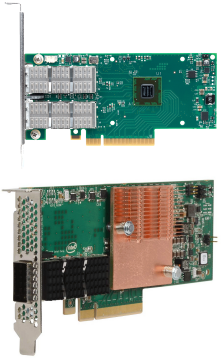
Coprocessors

FPGAs

CLUSTERS

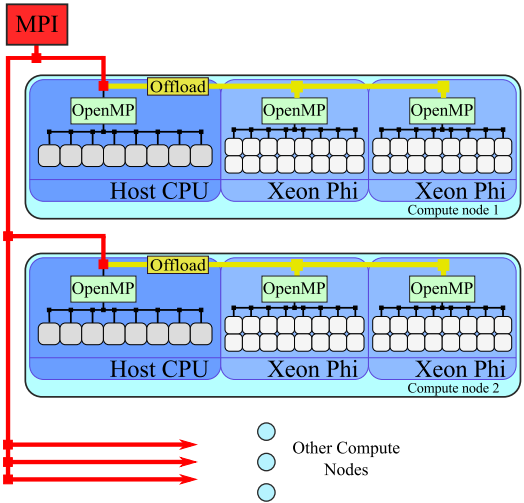
Clusters often use Gigabit Ethernet for administration and InfiniBand or Intel Omni-Path for communication.





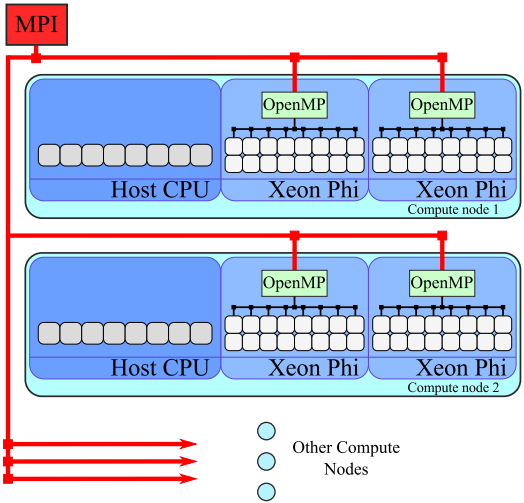
SCALING ACROSS A CLUSTER WITH COPROCESSORS

- 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



PARALLEL PROGRAMMING FRAMEWORKS

Data-Parallel :

intrinsics, 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

```
#include "mpi.h"
#include <cstdio>
int main (int argc, char *argv[]) {
  MPI_Init (&argc, &argv);  // Initialize MPI envirnmnt
  int rank, size, namelen;
  char name[MPI_MAX_PROCESSOR_NAME];
  MPI_Comm_rank (MPI_COMM_WORLD, &rank); // ID of current process
  MPI_Get_processor_name (name, &namelen); // Hostname of node
  MPI_Comm_size (MPI_COMM_WORLD, &size); // Number of processes
  printf ("Hello World from rank %d running on %s!\n", rank, name);
  if (rank == 0) printf("MPI World size = %d processes\n", size);
  MPI_Finalize (); // Terminate MPI environment
}
```

MPICH site contains a list of MPI 3.2 routines

COMPILING AND RUNNING MPI APPLICATIONS ON LOCALHOST

Compilation:

```
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
```

Results:

```
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`

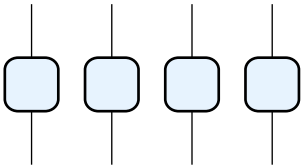
COMPILING AND RUNNING MPI APPLICATIONS

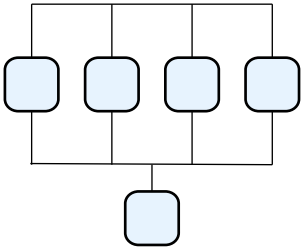
1. Compile and link with the MPI wrapper of the compiler:
   - `mpiicc` for C,
   - `mpiicpc` for C++,
   - `mpiifort` for Fortran 77 and Fortran 95.
2. For Xeon Phi coprocessors (KNC): `I_MPI_MIC=1`
3. Launch with the tool `mpirun`
   - Colon-separated list of hosts (`-host` *hostname*),
   - Alternatively, `-machinefile $PBS_NODEFILE`

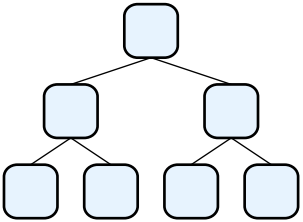
COMMUNICATION PATTERNS

PARALLEL PATTERNS

Embarrassingly parallel, reduction, fork-join.





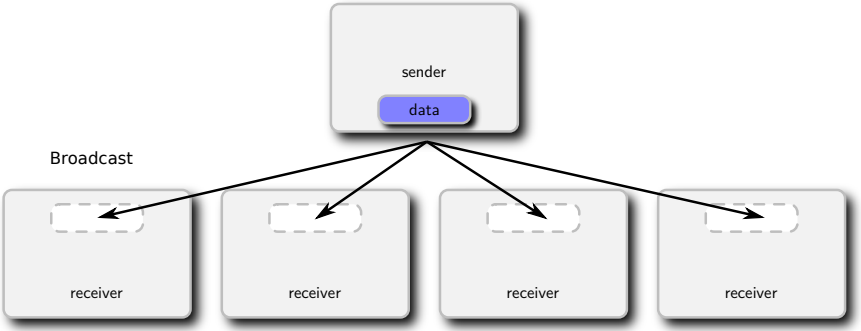


POINT TO POINT COMMUNICATION

```
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 }
```

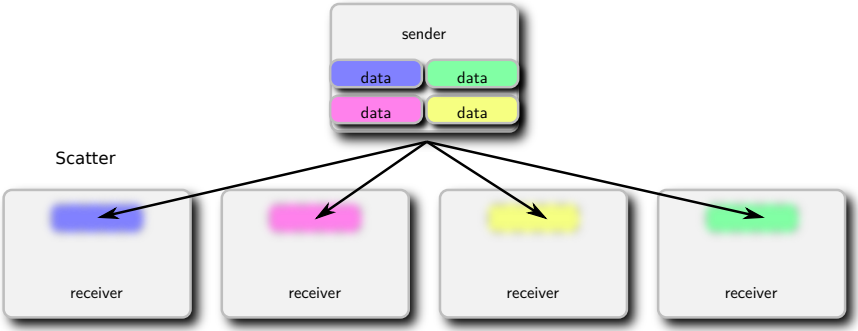
COLLECTIVE COMMUNICATION: BROADCAST

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



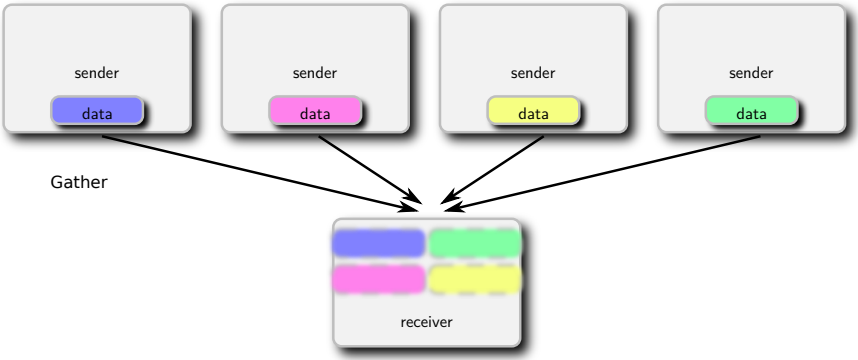
COLLECTIVE COMMUNICATION: SCATTER

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



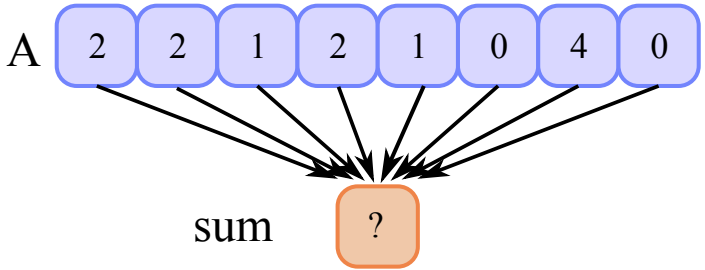
COLLECTIVE COMMUNICATION: GATHER

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



COLLECTIVE COMMUNICATION: REDUCTION

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

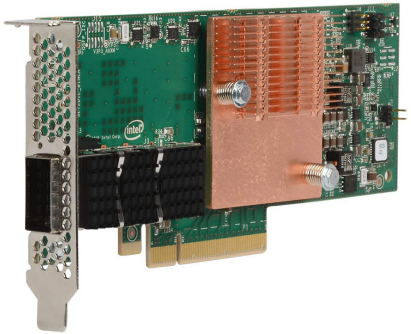


Available reducers: max/min, minloc/maxloc, sum, product, AND, OR, XOR (logical or bitwise).

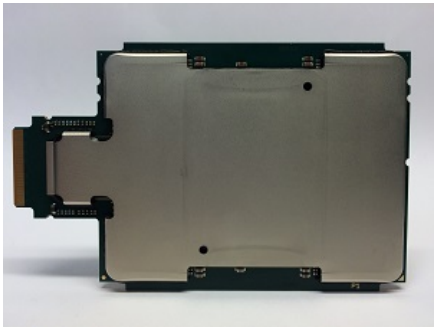
INTEL OMNI-PATH ARCHITECTURE

INTEL'S HPC COMMUNICATION FABRIC

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



Discrete

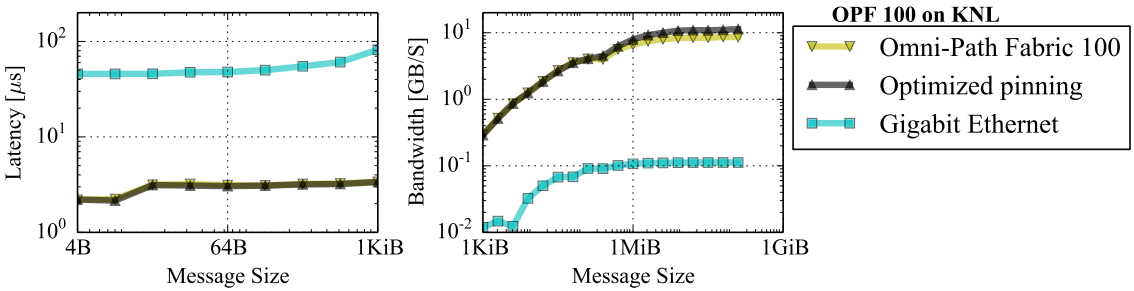


Integrated

INTEL'S HPC COMMUNICATION FABRIC

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
```



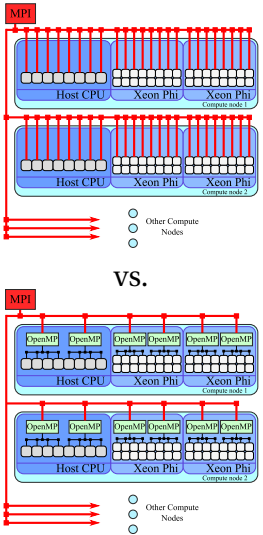
Optimal pinning `I_MPI_PIN_PROCESSOR_LIST=7` may be done automatically by a future version of Intel MPI

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



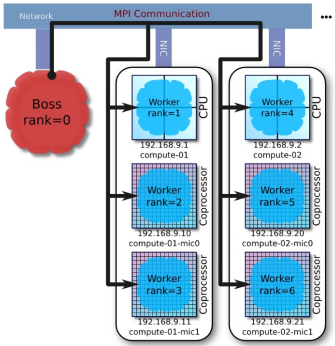
HYBRID MPI+OPENMP

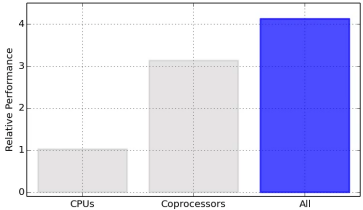
- For MPI calls from multiple MPI threads, use `-mt_mpi`
- MPI pins processes to cores and sets OpenMP affinity for them.
- To tune pinning: `I_MPI_PIN`, `I_MPI_PIN_DOMAIN`
- To diagnose process pinning: `I_MPI_DEBUG=4`
- More information in the MPI Reference Manual

LOAD BALANCING

ASIAN OPTION PRICING: HETEROGENEOUS CLUSTERING

Heterogeneous Clustering with Homogeneous Code:
Asian Option Pricing



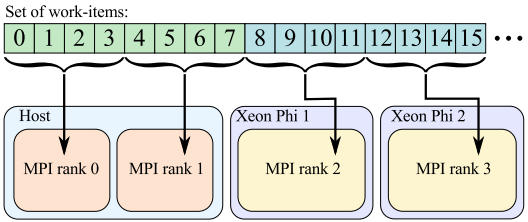


COLFAX
Customized Solutions

http://xeonphi.com/papers/heterogeneous

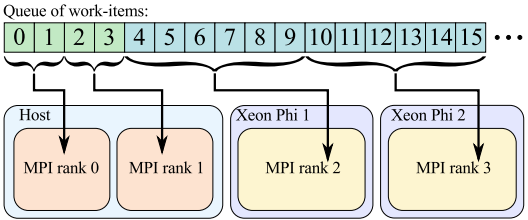
HETEROGENEOUS CALCULATION WITHOUT LOAD BALANCING

```
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]);
```

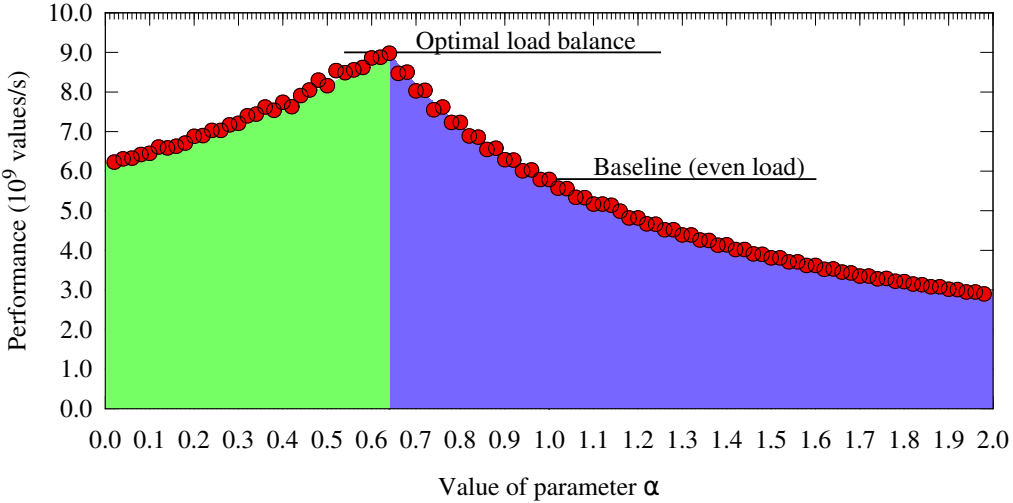


STATIC LOAD BALANCING

```
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)); }
```

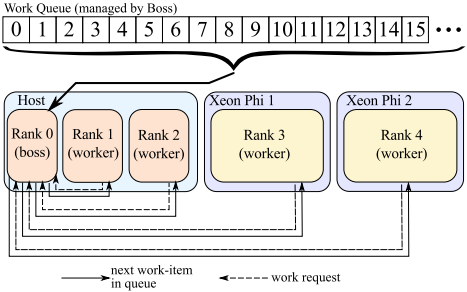


STATIC LOAD BALANCING: PARAMETER TUNING

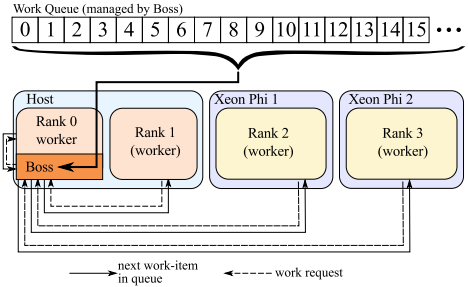


DYNAMIC LOAD BALANCING

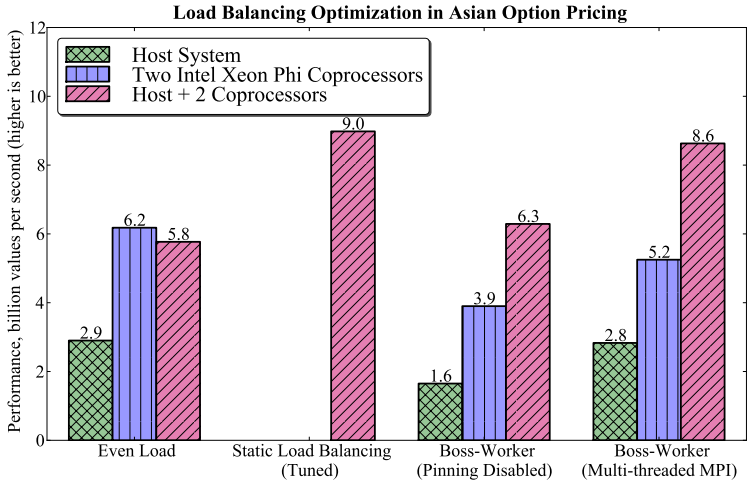
```
1  if (myRank == 0) // Boss's branch
2    DistributeWork(nOptions, option, mpiWorldSize);
3  else // Workers' branch
4    ReceiveWork(option, payoff, myRank);
```



or



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`:

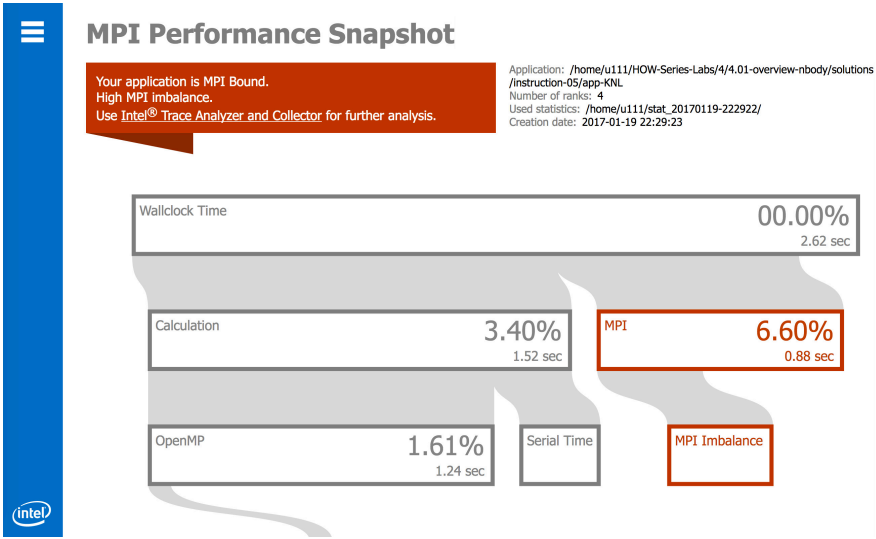
```
#PBS -l nodes=4

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
```

READING INTEL MPI PERFORMANCE SNAPSHOT



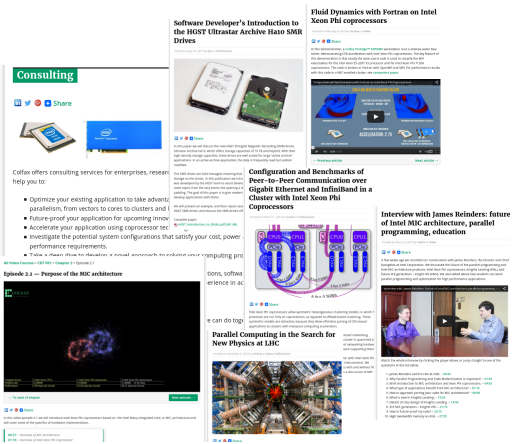
SUMMARY ON MPI

- 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

MPICH site contains a list of MPI 3.2 routines

Next session: performance optimization: overview, case study

COLFAX RESEARCH



http://colfaxresearch.com/