MPI Programming on Intel® Xeon Phi™ coprocessor

Presenter: Zhou Shan SSG/DRD/CRT

Contributor: SSG/DRD SSG/DPD

Legal Disclaimers

Objectives

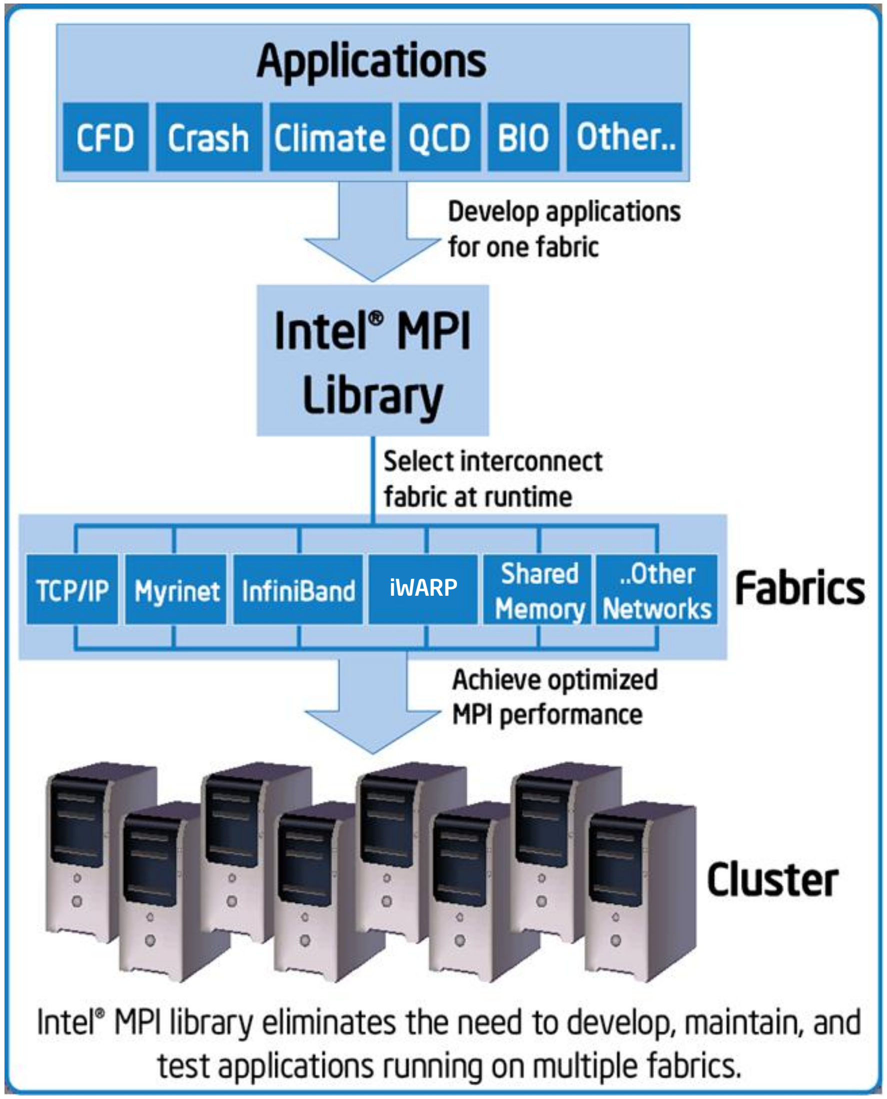
- Introduce MPI programming models on Intel® Xeon Phi™ product
- Basic implementation and fabric suppport information on Intel® Xeon Phi™ product

Agenda

- **Overview**
- MPI Programming Models
- MPI Communication Mechanisms
- Intel® Trace Analyzer and Collector
- Load Balance
- Summary

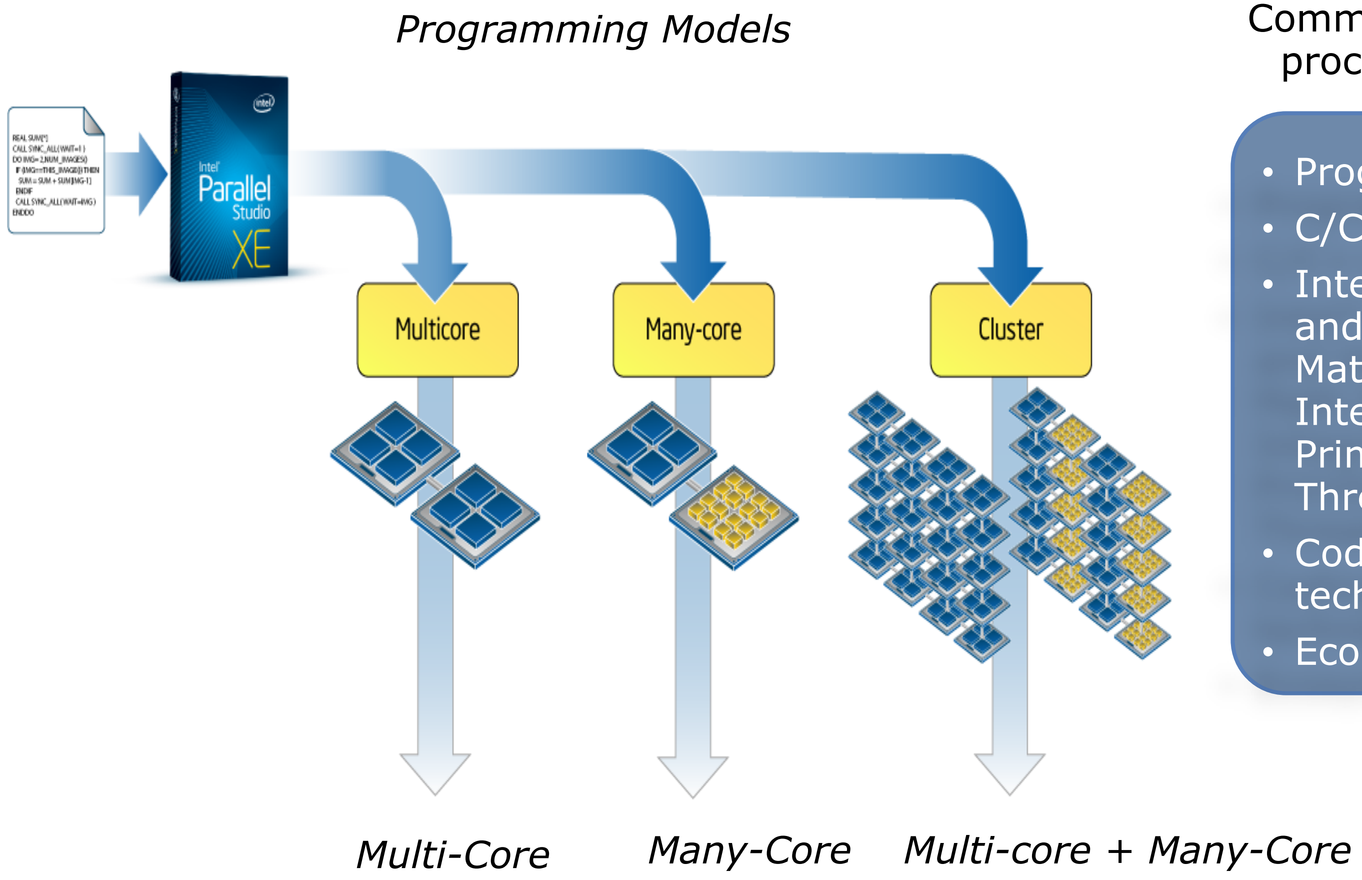
Intel® MPI Library Overview

- Optimized MPI application performance
  - Application-specific tuning
  - Automatic tuning
- Lower latency
  - Industry leading latency
- Interconnect Independence & Runtime Selection
  - Multi-vendor interoperability
  - Performance optimized support for the latest OFED capabilities through DAPL 2.0
- More robust MPI applications
  - Seamless interoperability with Intel® Trace Analyzer and Collector



Intel® MPI library eliminates the need to develop, maintain, and test applications running on multiple fabrics.

Intel® MIC Architecture Programming



Common with Intel® Xeon® processors

- Programming Models
- C/C++, Fortran compilers
- Intel® SW developer tools and libraries including Intel® Math Kernel Library, Intel® Integrated Performance Primitives and Intel® Threading Building Blocks
- Coding and optimization techniques and SW tools
- Ecosystem support

For illustration only, potential future options subject to change without notice.

Eliminates Need for Dual Programming Architecture

Agenda

Programming Models-Overview

MPI+Offload

- MPI ranks on Xeon only
- Messages into/out of Xeon
- MIC as accelerator
- Simplest

Co-processor-only

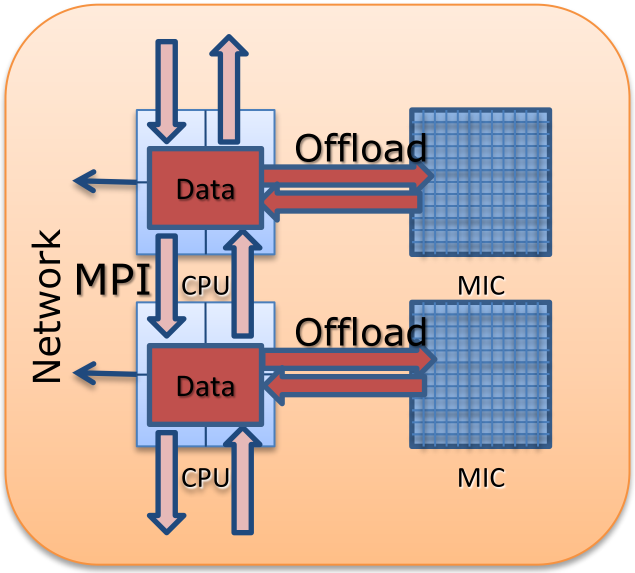
- MPI ranks on several MIC cards only
- Messages into/out of MIC thru Xeon

Symmetric

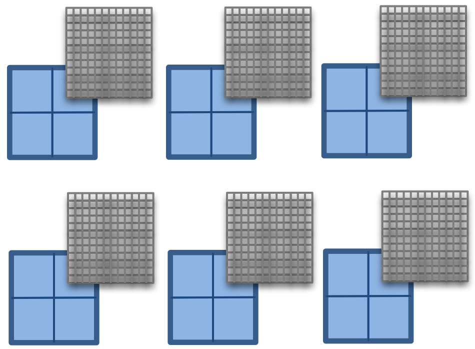
- MPI ranks on several MIC cards and Xeons
- Messages to/from any core

MPI+Offload Programming Model

- MPI ranks on Xeon processor(s) only
- All messages into/out of Xeon processors
- Offload models used to accelerate MPI ranks
- Intel ® Cilk ™ Plus , Intel® TBB, OpenMP, Pthreads used within MIC



Homogenous network of heterogeneous nodes



Suitable to start MIC enabling in HPC
• Seamless movement to heterogeneous computing
• Advantages of more cores and wider SIMD for certain applications

MPI+offload Programming Model

- Build the application with MPI+offload mode

```
# mpiicc –o test  test.c
```

- The code will be compiled by default for offload if offload construct is detected by compiler

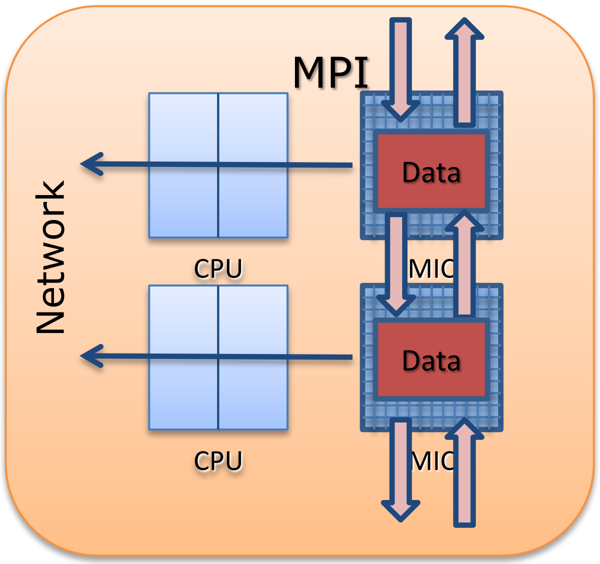
```
# switch off by  adding ”-no-offload” option when compiling
```

- Execute on host(s) as usual

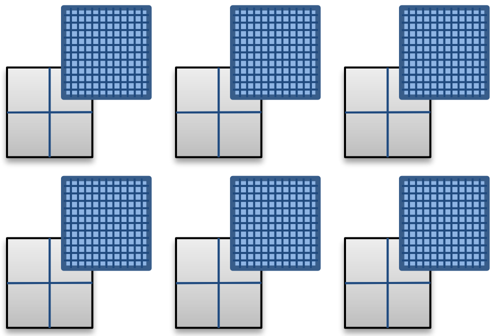
```
# mpiexec –n 2 ./test
```

- MPI processes will offload code to MIC card(s)

Co-processor-only Programming Model



Homogenous network
of many-core CPUs



- MPI ranks on MIC co-processor(s) only
- All messages into/out of MIC thru Xeon
- Intel ® Cilk ™ Plus , Intel® TBB, OpenMP, Pthreads used directly within MPI processes

Co-processor-only Programming Model

- Build the application in Co-processor-only mode for MIC architecture

```
# mpiicc –mmic –o test_MIC  test.c
```

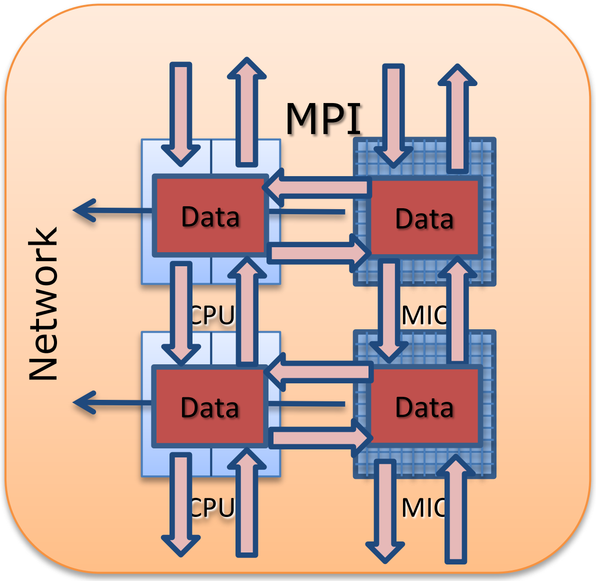
- Upload the MIC executable file to MIC card

```
# sudo scp test_MIC mic0:/tmp/test_MIC
```

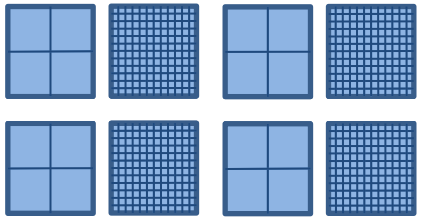
- Launch the application from host or login to the MIC card and launch it from the card by using mpiexec.hydra

```
# mpiexec –n 2 –wdir /tmp  -host mic0 /tmp/test_MIC
```

Symmetric Programming Model



Heterogeneous network of homogeneous CPUs



- MPI ranks on MIC card(s) and Xeon Processor(s)
- Messages to/from any core, MIC card(s) or Host CPU (s)
- Intel ® Cilk ™ Plus , Intel® TBB, OpenMP, Pthreads used directly within MPI processes

Symmetric Programming Model

- Build the application for Xeon and MIC Co-processor separately

```
# mpiicc –o test  test.c
# mpiicc –mmic –o test_MIC  test.c
```

- Upload the MIC executable file to MIC card

```
# sudo scp test_MIC mic0:/tmp/test_MIC
```

- Launch the application on the host and MIC card from host

```
# mpiexec –n 2 –host <hostname>  ./test
  : –wdir /tmp  -n 2 -host  mic0 /tmp/test_MIC
```

Agenda

- Overview
- MPI Programming Models
- MPI Communication Mechanisms
- Intel® Trace Analyzer and Collector
- Load Balance
- Summary

Communication Mechanisms

□ Multi-core cluster communication :

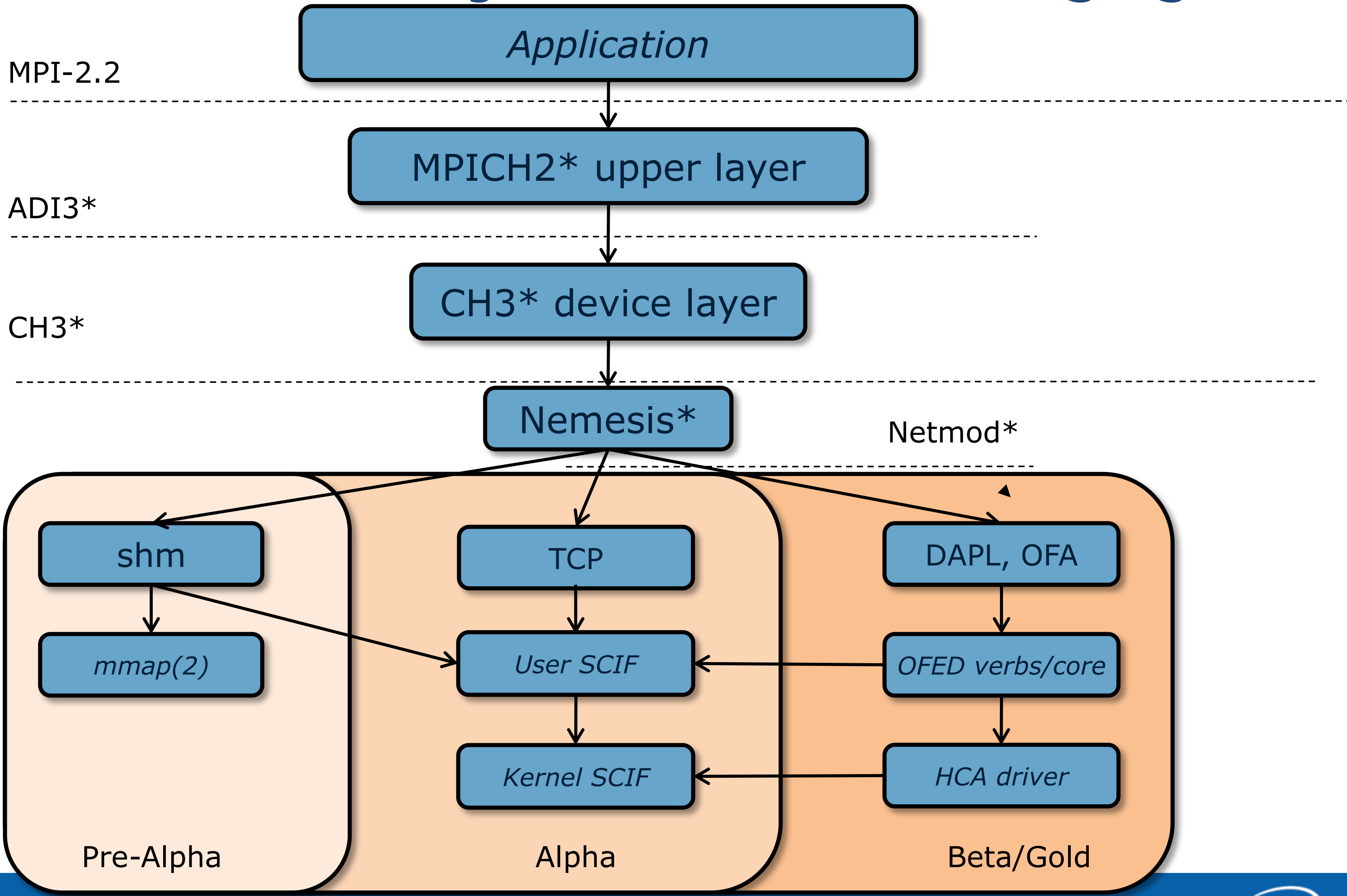
- Inter node, by Infiniband, Ethernet, etc
- Intra node
  - Inter sockets(QPI)
  - Intra socket

□ Multi-core + Many-core cluster communication

- All of the communication mechanisms of multi-core cluster
- Two additional communication

- Communication between host and co-processor
- Inter co-processors communication

Intel® MPI Library Architecture & Staging



SCIF: symmetric communications Interface
HCA: Host Channel Adapter
OFA - OpenFabrics Alliance
DAPL - Direct Access Programming Library

Network Fabrics for Heterogenous Cluster

- Default select the best available network fabric which can be found
- The best fabric is usually based on Infiniband(dapl, ofa) for inter node and shared memory(shm) for intra node
- Available farics for Intel® Xeon Phi™ coprocessor

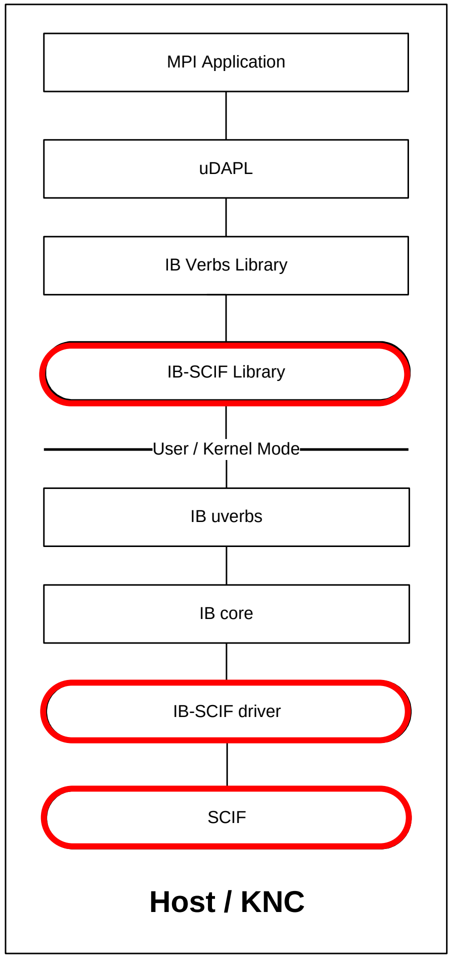
  - shm, tcp, ofa, dapl

  - Availability checked in the order, shm:dapl, shm:ofa, shm:tcp(intra:inter)
- I_MPI_FABRICS can be used to select a different communication mechanism explicitly
- Set I_MPI_SSHM_SCIF=1 to enable shm fabric between host and MIC card

MPSS software to support MPI

OFED over SCIF

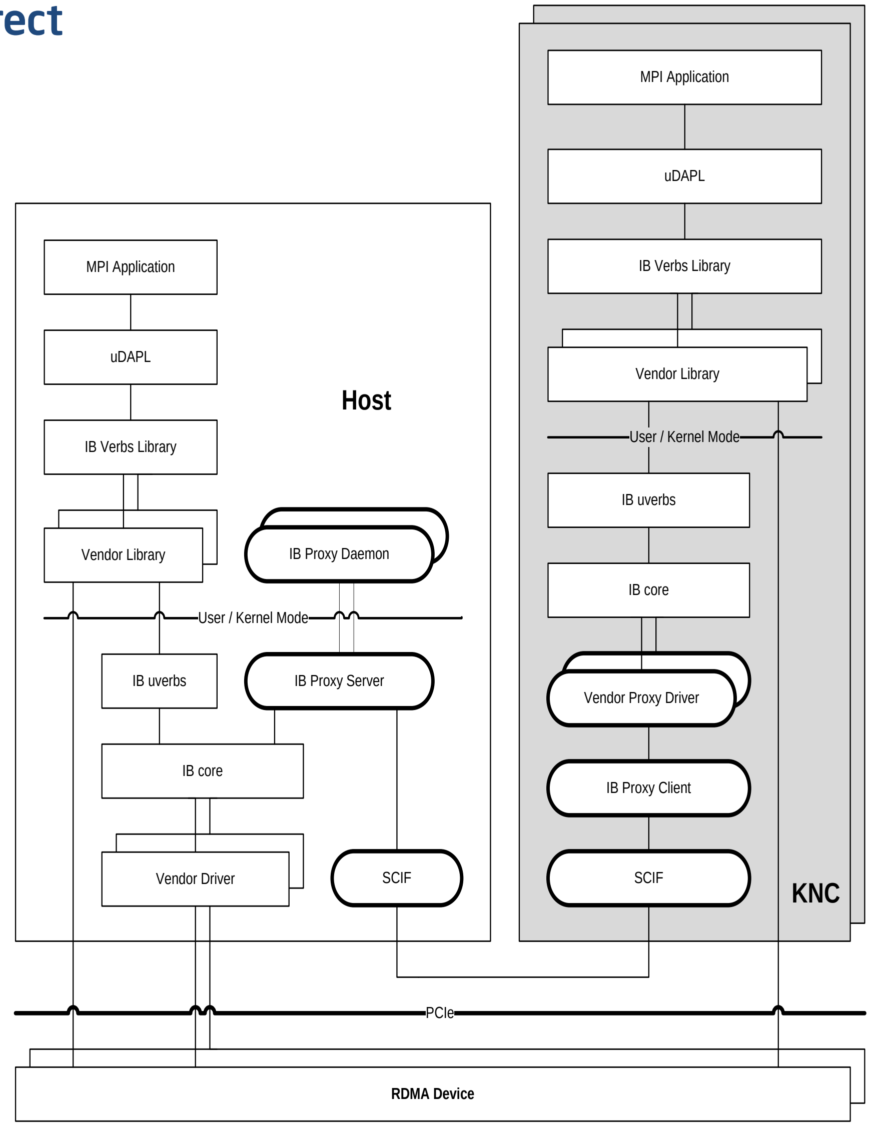
- OFA Software (OFED) is the industry standard code used for messaging on high-end HPC clusters (>60% of Top500)
  - Supports Intel MPI and all open source MPIs
  - Is in Linux and all the various Linux distros
- OFED for MIC uses the core OFA software modules from the Open Fabrics Alliance
- OFED/SCIF is a new hardware specific driver and library that plugs into the OFED core mid-layer
  - SCIF is the lowest level communications driver between the Xeon and the MIC cards
  - OFED/SCIF allows communication between the Xeon and the MIC or between MIC cards on the same system



MPSS software to support MPI

MIC- direct

- MIC-Direct allows MPI speed path operations direct access to a RDMA NIC, i.e., InfiniBand HCA, iWarp RNIC
  - MIC shares RDMA NIC with Host
  - No changes to MPIs are needed
- Communication setup operations are proxied to the Xeon, e.g, allocate QPs, register memory, map HCA registers



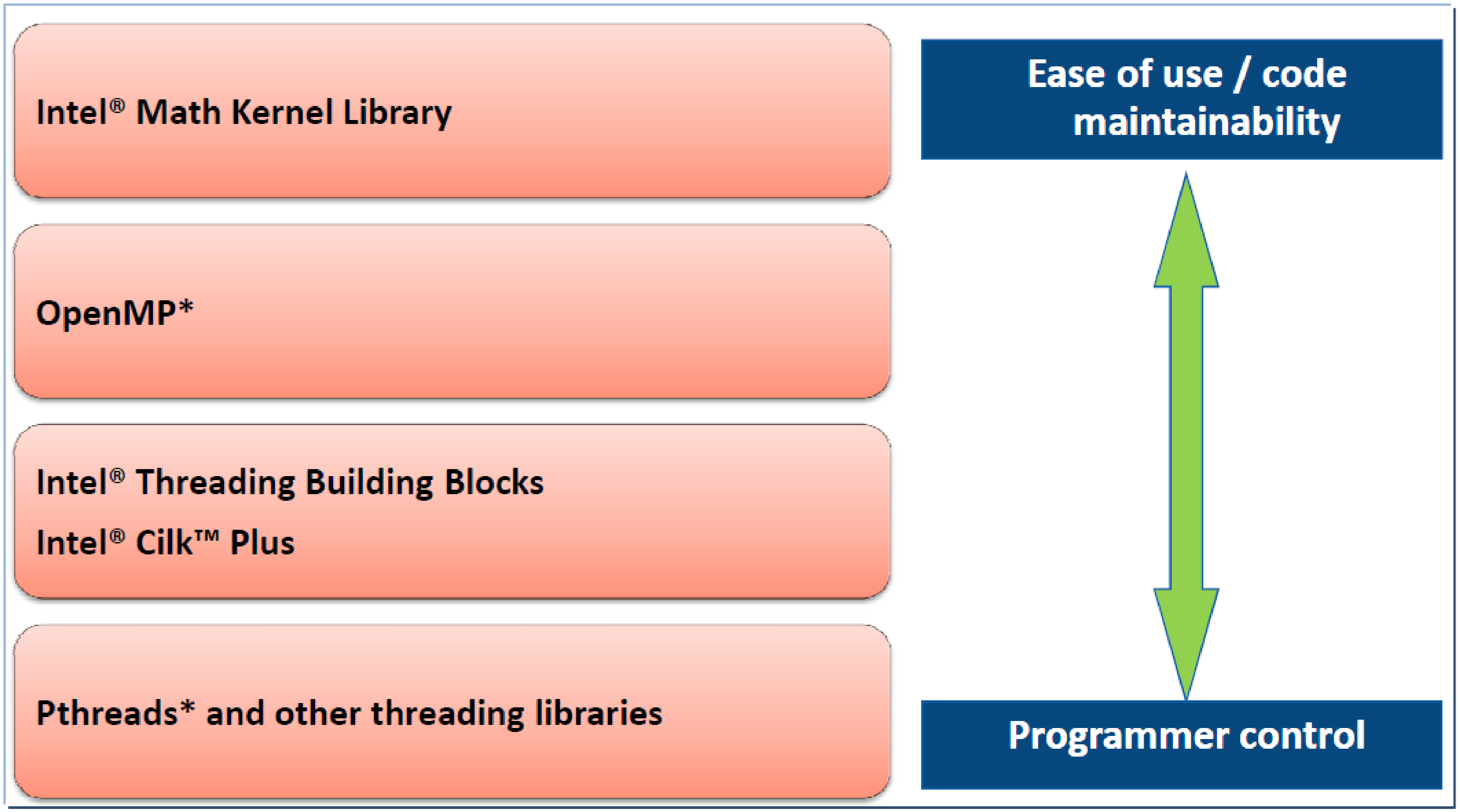
Pure MPI Programming

- MPI is the portable cluster solution
- Enlarge Cluster Scale and improve performance by
  - Increasing core number per node
  - Increasing node number
- Performance improved need more MPI processes
- Potential limitations
  - Memory requirement per MPI process, would exceeds the node memory
  - Limited scalability due to exhausted interconnects
  - Load imbalance

Hybrid programming - overview

- Combine MPI programming model with many multi-threading models
- Overcome MPI limitations by using multi-threads
  - Potential memory gains in threaded code
  - Better scalability, less MPI communication
  - Better load balance strategies
- Maximize performance by exploitation of hardware

Options for Thread Parallelism


Intel® Math Kernel Library
OpenMP*
Intel® Threading Building Blocks
Intel® Cilk™ Plus
Pthreads* and other threading libraries
Ease of use / code maintainability
Programmer control

Unified programming to support Intel® Xeon and Intel® Xeon Phi™ coprocessor

MPI Process Mapping

- By mapping MPI processes would cause impact on performance, especially for hybrid mode
- Intel® MPI provides many environment variable for programmer to control process pinning

  - For pure MPI: I_MPI_PIN_PROCESSOR_LIST
  - For hybrid code,

    split logical processors into subsets,

    pin omp threads inside domain with affinity set

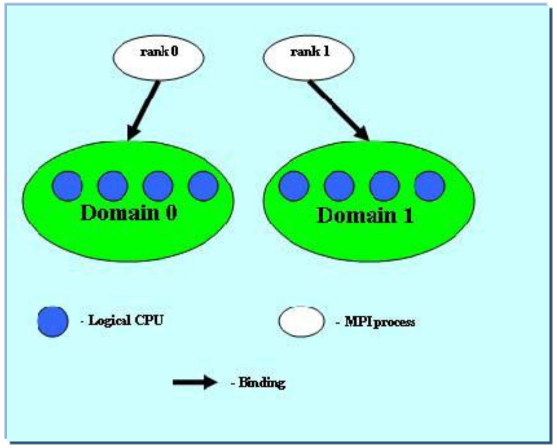
  I_MPI_PIN_DOMAIN=<size>[:<layout>]

  <size> =

omp	Adjust to OMP_NUM_THREADS
auto	#CPU/#MPI procs
<n>	Number

<layout>=

platform	According to BIOS number
compact	close to each other
scatter	far away from each other



- Also extends this support to MIC

MPI + Offload Support

- Avoid the interference between the offload from different MPI processes, such as

- Don't offload to the same card for different MPI process

- Using MPI+openMP and set thread affinity manually for every MPI process

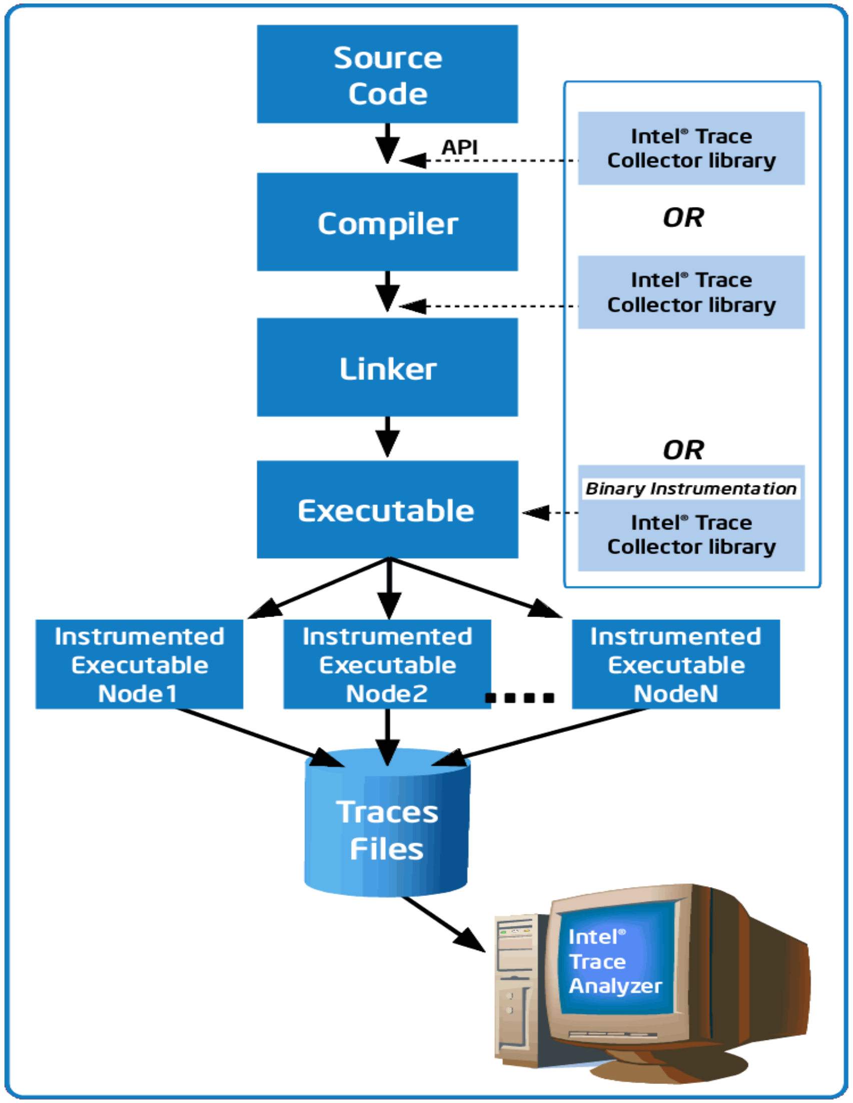
```
#export OMP_NUM_THREADS=4
#mpiexec –env KMP_AFFINITY=[1,4] –n 1 –host mic0
         –env KMP_AFFINITY=[5,6] –n 1 –host mic0
           .......
```

Agenda

- Overview
- MPI Programming Models
- MPI Communication Mechanisms
- **Intel® Trace Analyzer and Collector**
- Load Balance
- Summary

Intel® Trace Analyzer and Collector Overview

- Intel® Trace Analyzer and Collector helps the developer:
  - Visualize and understand parallel application behavior
  - Evaluate profiling statistics and load balancing
  - Identify communication hotspots
- Features
  - **Event-based** approach
  - **Low overhead**
  - Excellent scalability on time and processors
  - Comparison of multiple profiles
  - Powerful aggregation and filtering functions
  - Fail-safe MPI tracing
  - Provides **API to instrument** user code
  - **MPI correctness checking**
  - **Idealizer**


Source Code
API
Intel® Trace Collector library
OR
Compiler
Intel® Trace Collector library
Linker
OR
Binary Instrumentation
Intel® Trace Collector library
Executable
Instrumented Executable Node1
Instrumented Executable Node2
Instrumented Executable NodeN
Traces Files
Intel® Trace Analyzer

Build Application with ITAC support

Co-processor-only mode

- Build MIC executable with *-trace*

```
# mpiicc -mmic -trace -o test_MIC test.c
```

- Upload it to the MIC
- Run this binary remotely from the host
- Copy trace file (if no NFS mounted) from the card to a host system

Symmetric mode

- Build Intel®64 and MIC executables with *-trace*

```
# mpiicc -mmic -trace -o test_MIC test.c
# mpiicc       -trace -o test  test.c
```

- Upload the MIC executable to the card:
- Run the application from the host so that rank 0 will generate trace file

Offload mode

- Build Intel®64 executables with *-trace*

```
# mpiicc    -trace -o test  test.c
```

- Run the application from the host to generate trace file

Run Application with ITAC support

Co-processor-only mode

- Run with –trace flag to create a trace file

```
# mpiexec –trace –n 2  -wdir /tmp –host mic0 /tmp/test_MIC
```

Symmetric mode

- Run with –trace flag to create a trace file

```
# mpiexec  –trace –n 2 –host  michost ./test  :
                   -wdir /tmp –n 2 –host mic0 /tmp/test_MIC
```

Offload mode

- Run with –trace flag to create a trace file

```
 # mpiexec    -trace  -n 2-o  ./test
```

Set VT_LOGFILE_FORMAT=stfsingle to create a singeletrace

ITAC Analysis

- Start the ITAC analysis GUI with the trace file (or load it)

```
# traceanalyzer  test.single.stf
```

- Start the analysis, usually by inspection of the Flat Profile (default chart), the Event Timeline, and the Message Profile

– Select “Charts->Event Timeline”

– Select “Charts->Message Profile”

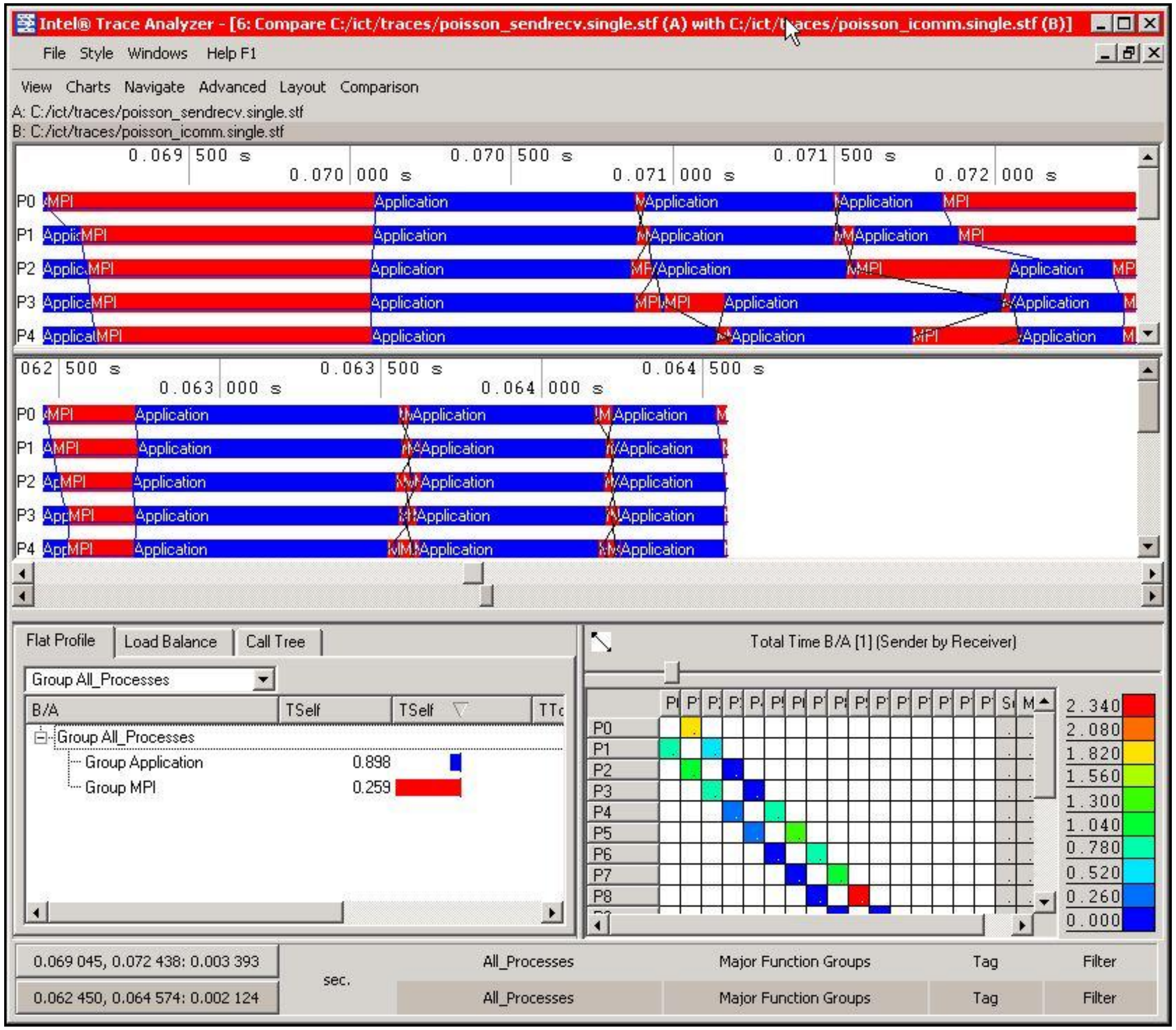
– Zoom into the Event Timeline

  Click into it, keep pressed, move to the right, and release the mouse

  See menu Navigate to get back

– Right click the “Group MPI->Ungroup MPI”.

ITAC Profile Example



Intel® Trace Analyzer - [6: Compare C:/ict/traces/poisson_sendrecv.single.stf (A) with C:/ict/traces/poisson_icomm.single.stf (B)]

A: C:/ict/traces/poisson_sendrecv.single.stf
B: C:/ict/traces/poisson_icomm.single.stf

Group All_Processes

B/A	TSelf	TSelf	TT
Group All_Processes			
Group Application		0.898	
Group MPI		0.259	

0.069 045, 0.072 438: 0.003 393	sec.	All_Processes	Major Function Groups	Tag	Filter
0.062 450, 0.064 574: 0.002 124		All_Processes	Major Function Groups	Tag	Filter

Compare the event timelines of two communication profiles

Blue = computation
Red = communication

Chart showing how the MPI processes interact

Intel® Trace Analyzer and Collector

Works on systems from 2 processes to more than a **thousand** processes

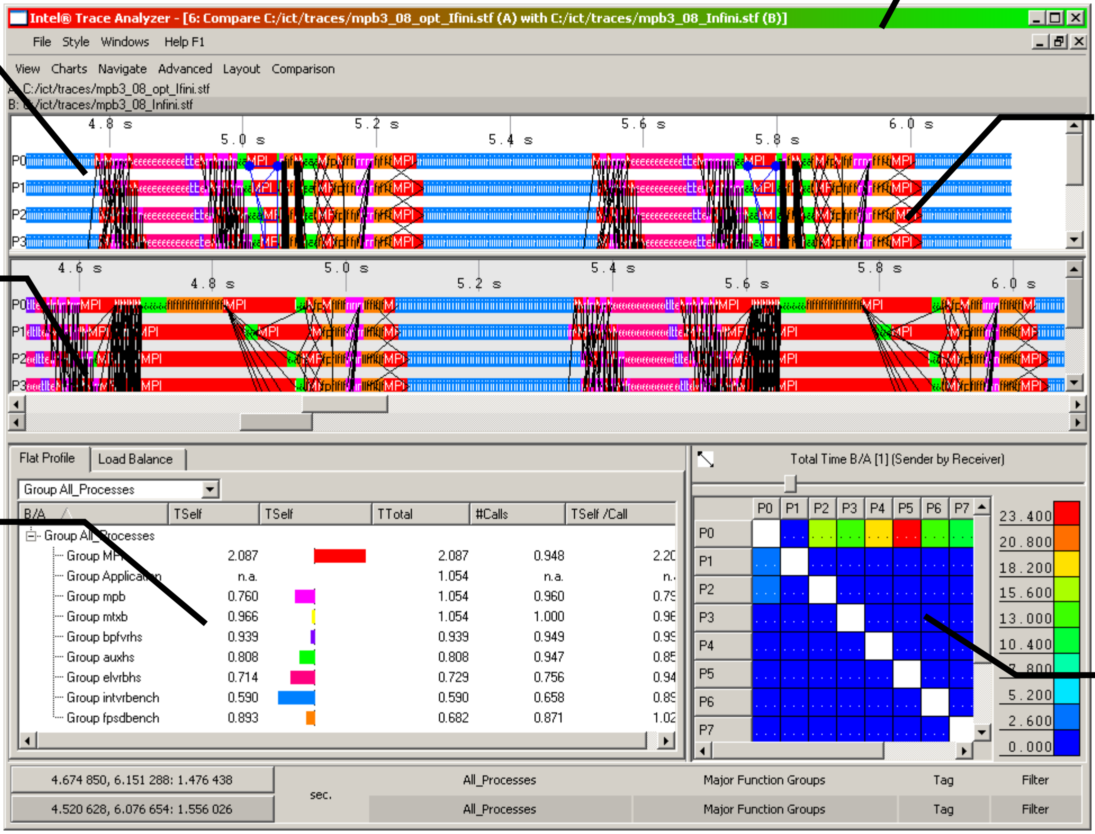
Timeline of **optimized** application run

Timeline of **initial** application run

Shorter RED bars means less MPI traffic and increased performance

Comparison of function and process profile data

Network usage profile data for MPI messages


Intel® Trace Analyzer - [6: Compare C:/ict/traces/mpb3_08_opt_Ifini.stf (A) with C:/ict/traces/mpb3_08_Infini.stf (B)]
File Style Windows Help F1
View Charts Navigate Advanced Layout Comparison
A: C:/ict/traces/mpb3_08_opt_Ifini.stf
B: C:/ict/traces/mpb3_08_Infini.stf
4.8 s 5.0 s 5.2 s 5.4 s 5.6 s 5.8 s 6.0 s
4.6 s 4.8 s 5.0 s 5.2 s 5.4 s 5.6 s 5.8 s 6.0 s
P0 P1 P2 P3
MPI
Flat Profile
Load Balance
Group All_Processes
Total Time B/A [1] (Sender by Receiver)
P0 P1 P2 P3 P4 P5 P6 P7
23.400
20.800
18.200
15.600
13.000
10.400
7.800
5.200
2.600
0.000

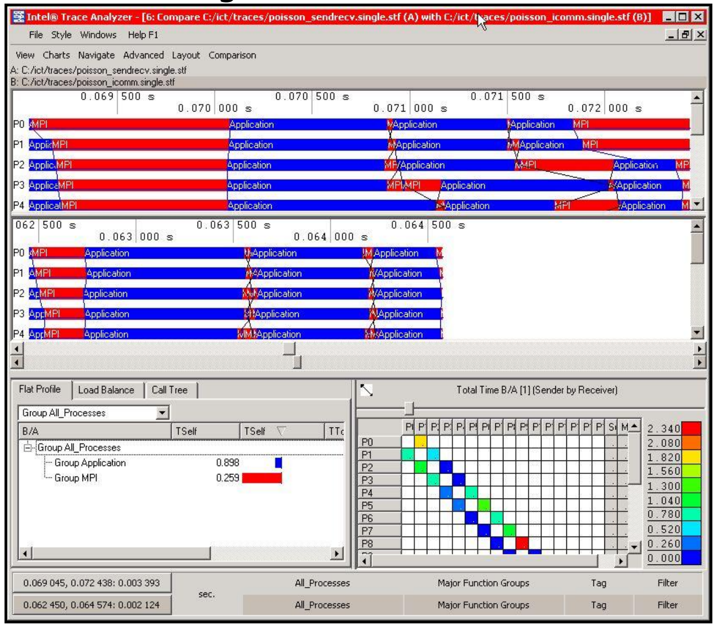
B/A	TSelf	TSelf	TTotal	#Calls	TSelf /Call
Group All_Processes					
Group MPI	2.087	🟥	2.087	0.948	2.20
Group Application	n.a.		1.054	n.a.	n.
Group mpb	0.760	🟪	1.054	0.960	0.7
Group mtxb	0.966	🟨	1.054	1.000	0.9
Group bpfvrhs	0.939	🟪	0.939	0.949	0.9
Group auxhs	0.808	🟩	0.808	0.947	0.8
Group elvrbhs	0.714	🟥	0.729	0.756	0.94
Group intvrbench	0.590	🟦	0.590	0.658	0.8
Group fpsdbench	0.893	🟧	0.682	0.871	1.02

4.674 850, 6.151 288: 1.476 438	sec.	All_Processes	Major Function Groups	Tag	Filter
4.520 628, 6.076 654: 1.556 026		All_Processes	Major Function Groups	Tag	Filter

Scale Performance

Tune Hybrid Cluster MPI and Thread Performance

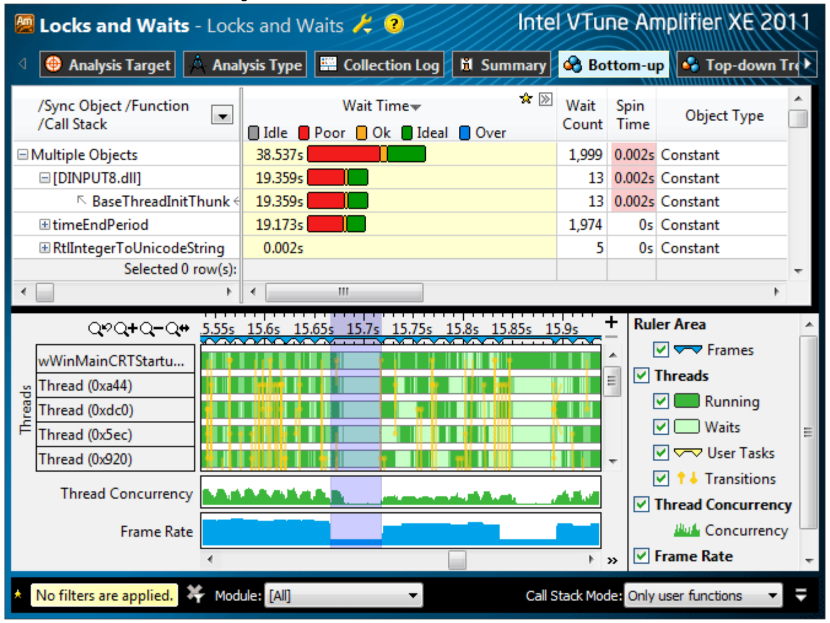
Intel® Trace Analyzer and Collector


Intel® Trace Analyzer - [6: Compare C:/ict/traces/poisson_sendrecv.single.stf (A) with C:/ict/traces/poisson_icomm.single.stf (B)]
File Style Windows Help F1
View Charts Navigate Advanced Layout Comparison
A: C:/ict/traces/poisson_sendrecv.single.stf
B: C:/ict/traces/poisson_icomm.single.stf
Flat Profile
Load Balance
Call Tree
Group All_Processes
Total Time B/A [1] (Sender by Receiver)

0.069 045, 0.072 438: 0.003 393	All_Processes
0.062 450, 0.064 574: 0.002 124	All_Processes

B/A	TSelf
Group All_Processes	
Group Application	0.898
Group MPI	0.259

Intel® VTune™ Amplifier XE



/Sync Object /Function /Call Stack	Wait Time	Wait Count	Spin Time	Object Type
Multiple Objects	38.537s	1,999	0.002s	Constant
[DINPUT8.dll]	19.359s	13	0.002s	Constant
BaseThreadInitThunk	19.359s	13	0.002s	Constant
timeEndPeriod	19.173s	1,974	0s	Constant
RtlIntegerToUnicodeString	0.002s	5	0s	Constant
Selected 0 row(s):				

Threads
wWinMainCRTStartu...
Thread (0xa44)
Thread (0xdc0)
Thread (0x5ec)
Thread (0x920)

No filters are applied. Module: [All] Call Stack Mode: Only user functions

Tune cross-node MPI

- Visualize MPI behavior
- Evaluate MPI load balancing
- Find communication hotspots

Tune single node threading

- Visualize thread behavior
- Evaluate thread load balancing
- Find thread sync. bottlenecks

Agenda

- Overview
- MPI Programming Models
- MPI Communication Mechanisms
- Intel® Trace Analyzer and Collector
- **Load Balance**
- Summary

Load Balance

- MPI in symmetric mode is like running on a heterogeneous cluster. Original load balanced codes may get imbalanced. Because

– Host and coprocessor computation performance are different

– Host and coprocessor internal communication speed is different

- There is no general solution.

– Approach 1: Adapt MPI mapping of (hybrid) code to performance characteristics: #m processes per host, #n process per coprocessor(s)

– Approach 2: Change code internal mapping of workload to MPI processes
such as uneven split of calculation grid for MPI processes on host vs.coprocessor(s)

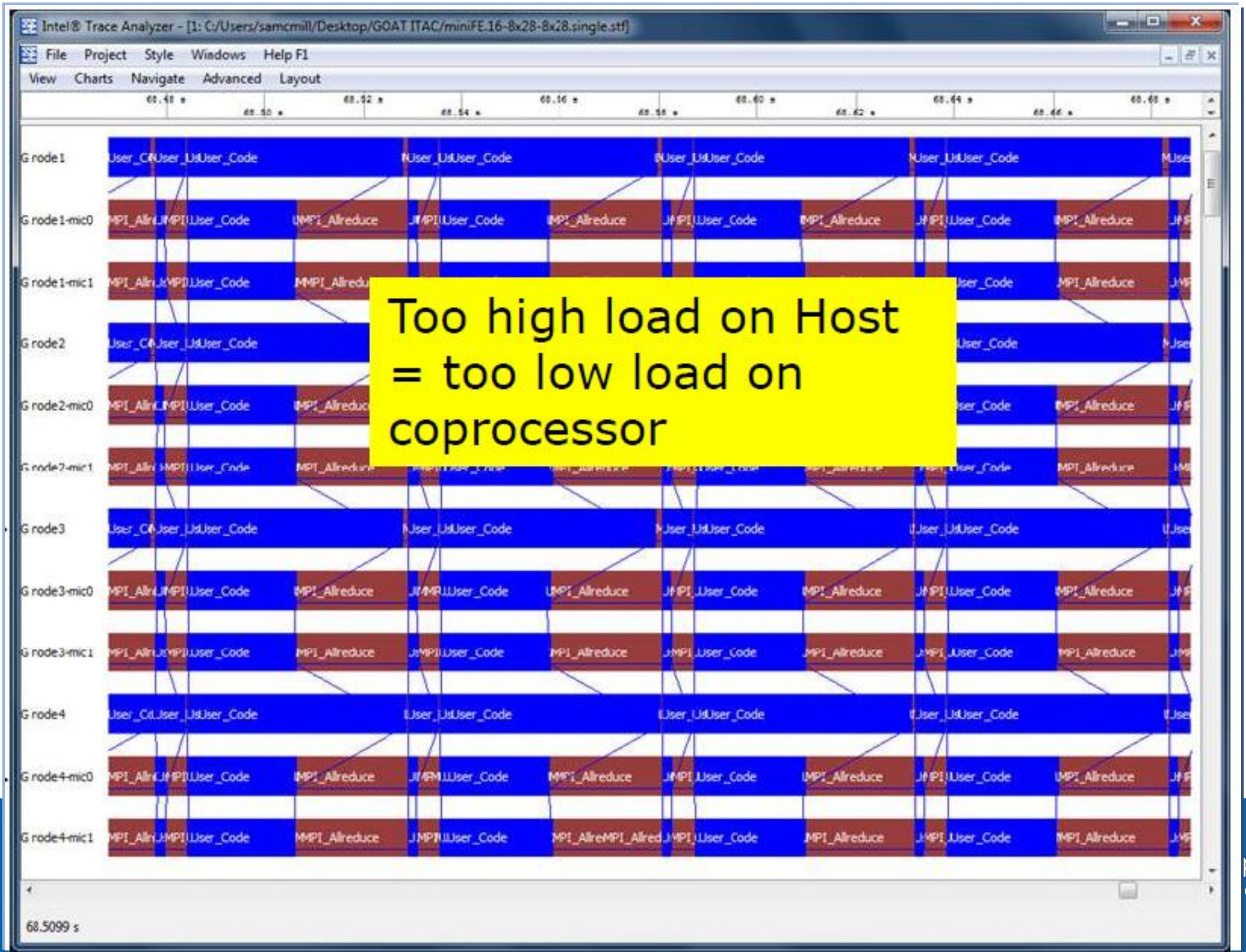
– Approach 3: …

- Analyze load balance of application with ITAC

– Ideal Interconnect Simulator

Improving Load Balance

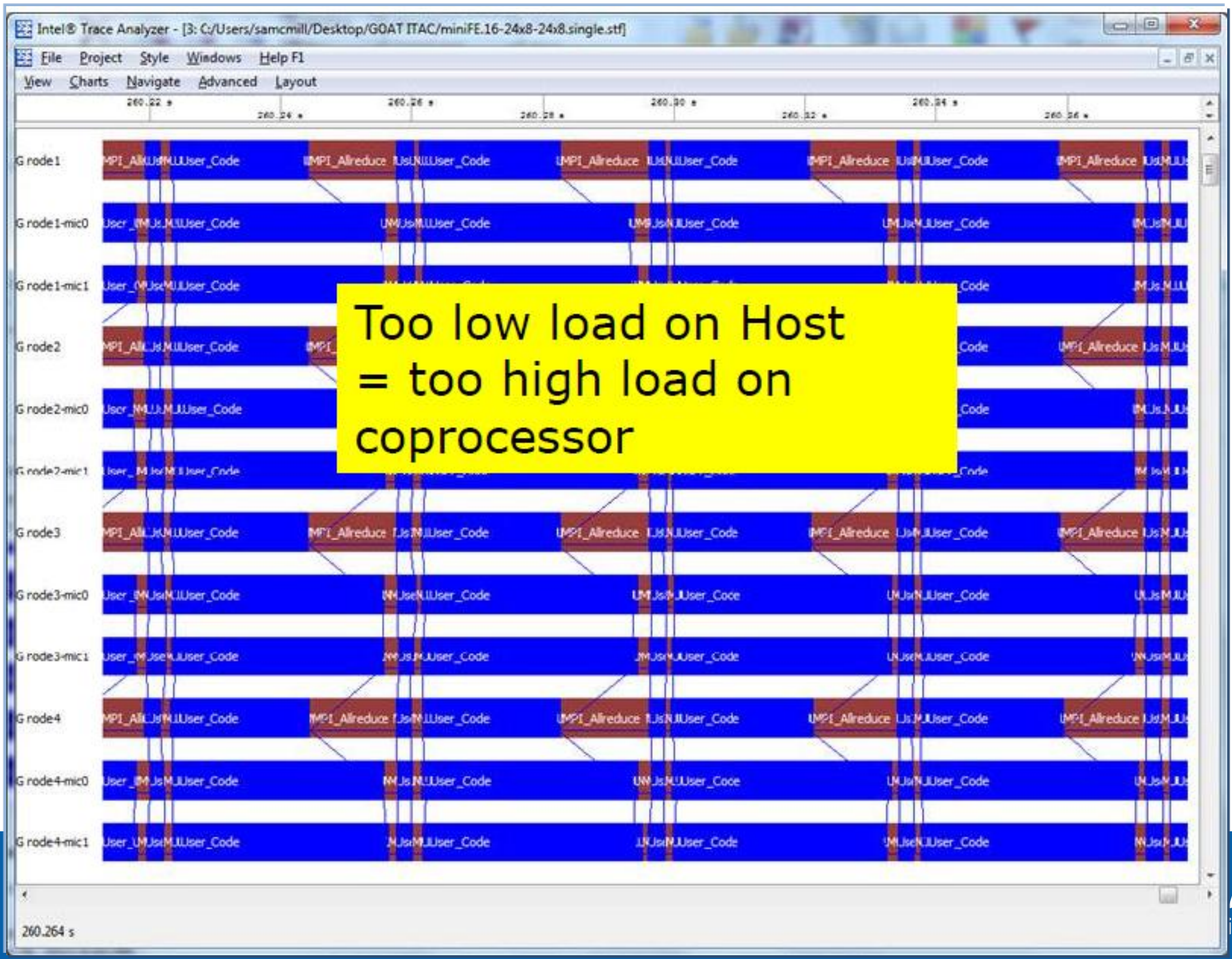
- Collapsed data per node and coprocessor card
- Host: 16 MPI ranks* 1 OMP thread
- MIC: 8 MPI ranks * 28 OMP threads


Intel® Trace Analyzer - [1: C:/Users/samcmill/Desktop/GOAT ITAC/miniFE.16-8x28-8x28.single.stf]
File Project Style Windows Help F1
View Charts Navigate Advanced Layout
G node1
G node1-mic0
G node1-mic1
G node2
G node2-mic0
G node2-mic1
G node3
G node3-mic0
G node3-mic1
G node4
G node4-mic0
G node4-mic1
User_Code
MPI_Allreduce
68.5099 s

Too high load on Host = too low load on coprocessor

Improving Load Balance(cont.)

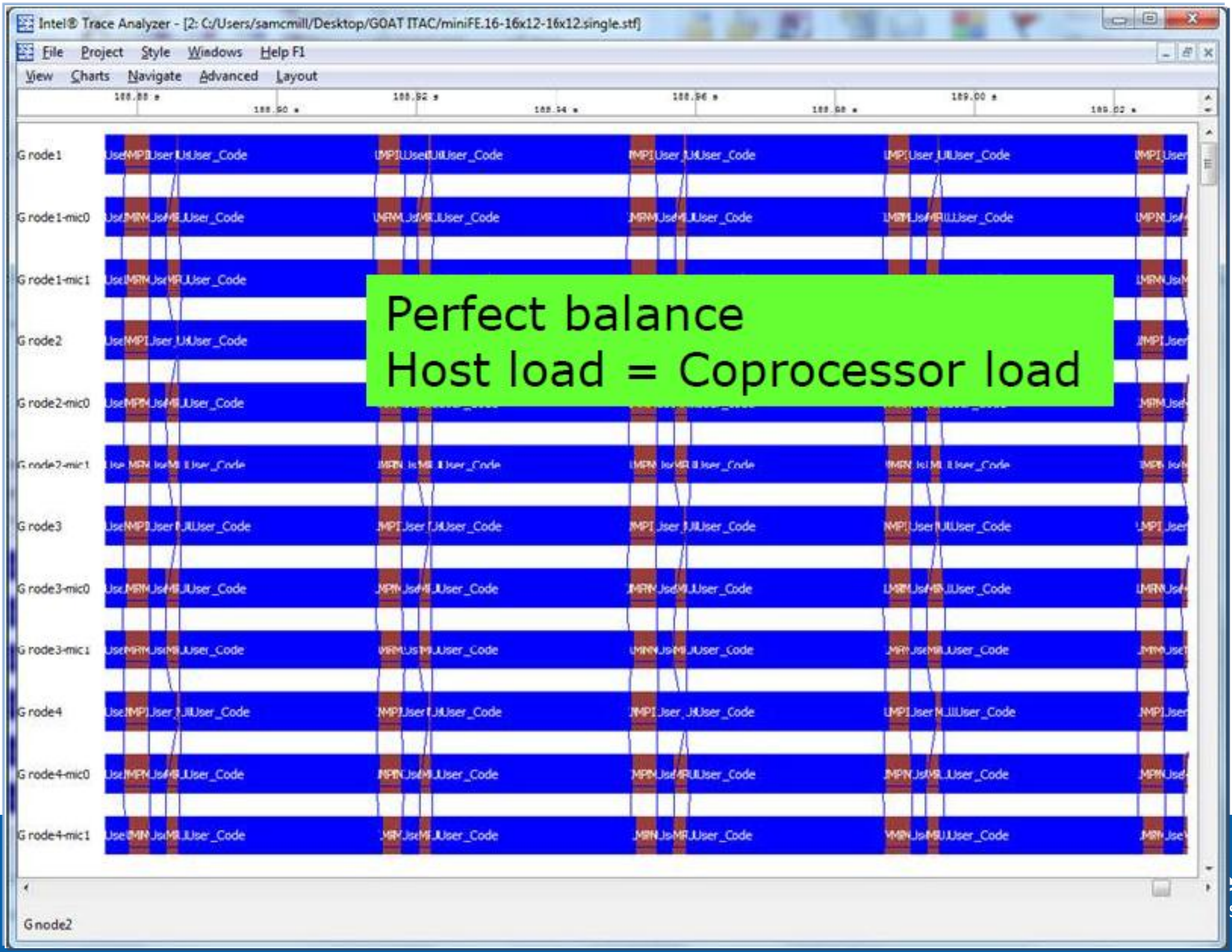
- Collapsed data per node and coprocessor card
- Host: 16 MPI ranks* 1 OMP thread
- MIC: 24 MPI ranks * 8 OMP threads


Intel® Trace Analyzer - [3: C:/Users/samcmill/Desktop/GOAT ITAC/miniFE.16-24x8-24x8.single.stf]
File Project Style Windows Help F1
View Charts Navigate Advanced Layout
G rode1
G rode1-mic0
G rode1-mic1
G rode2
G rode2-mic0
G rode2-mic1
G rode3
G rode3-mic0
G rode3-mic1
G rode4
G rode4-mic0
G rode4-mic1
MPI_Allreduce
User_Code
260.264 s

Too low load on Host = too high load on coprocessor

Improving Load Balance(cont.)

- Collapsed data per node and coprocessor card
- Host: 16 MPI ranks* 1 OMP thread
- MIC: 16 MPI ranks * 12 OMP threads


Intel® Trace Analyzer - [2: C:/Users/samcmill/Desktop/GOAT ITAC/miniFE.16-16x12-16x12.single.stf]
File Project Style Windows Help F1
View Charts Navigate Advanced Layout
G node1
G node1-mic0
G node1-mic1
G node2
G node2-mic0
G node2-mic1
G node3
G node3-mic0
G node3-mic1
G node4
G node4-mic0
G node4-mic1
User_Code
MPI
Gnode2

Perfect balance
Host load = Coprocessor load

Hands On Lab

Thanks!

More detailed info can be got from
http://software.intel.com/mic-developer