MPI Programming on Intel Xeon Phi™ Coprocessors

Silvio Stanzani , Raphael Cóbe , Rogério Iope

UNESP - Núcleo de Computação Científica

silvio@ncc.unesp.br , rmcobe@ncc.unesp.br , rogerio@ncc.unesp.br

Exploiting the parallel universe

Instruction Level Parallelism

- Single thread (ST) performance
- Automatically exposed by HW/tools
- Effectively limited to a few instructions

Data Level Parallelism

- Single thread (ST) performance
- Exposed by tools and programming models
- Operate on 4/8/16 elements at a time

Task Level Parallelism

- Multi thread/task (MT) performance
- Exposed by programming models
- Execute tens/hundreds/thousands task concurrently

Process Level Parallelism

- Multi Process (MP) performance
- Exposed by programming models
- Execute tens/hundreds/thousands of process concurrently across several nodes

Agenda

- **Message Passing Interface (MPI)**
- MPI Programming
- Profiling MPI
- Combining MPI/OpenMP/Offload/Vecorization (Affinity)

Agenda

- **Message Passing Interface (MPI)**
- MPI Programming
- Profiling MPI
- Combining MPI/OpenMP/Offload/Vecorization (Affinity)

Message Passing Interface (MPI)

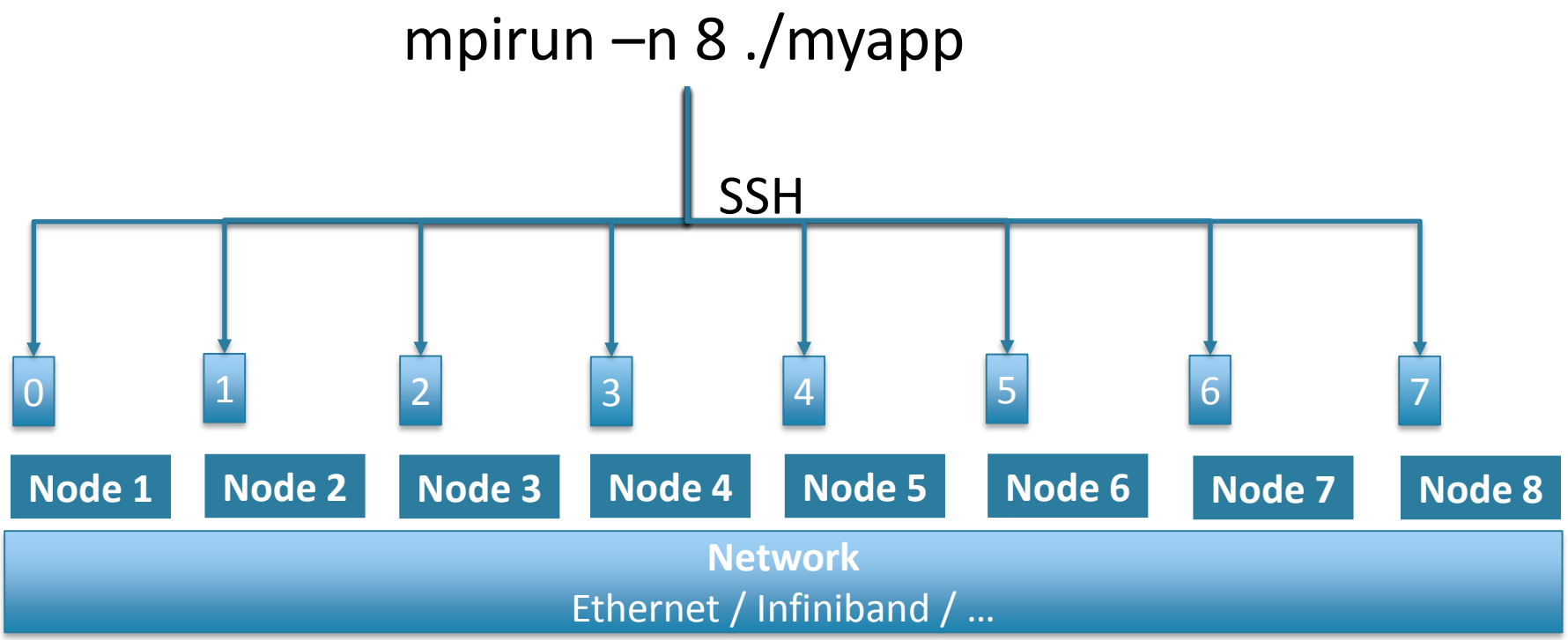
- Rank
  - Each process executing concurrently is identified as rank
- Communicators
  - A logical communication channel that support the message exchange between Ranks;
- Library
  - MPI_Init(&argc, &argv)
    - create a communication channel
  - MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    - Obtain rank number
  - MPI_Comm_size(MPI_COMM_WORLD, &size);
    - Obtain amount of ranks
  - MPI_Finalize();
    - Destroy the communication channel
  - Mpi_send();
    - Send a Message (single const, arrays of pointers, etc…)
  - MPI_recv();
    - Receive a Message (single const, arrays of pointers, etc…)
  - …

Compiling and Running MPI applications

- Compile MPI sources using Intel® MPI scripts
  - mpiicc –o test test.c
- For native execution on MIC add “–mmic“ flag
  - mpiicc –mmic –o test.mic test.c
- Running MPI applications
  - mpirun [clauses]
    - -host : hostname to execute process
    - -n : amount of process (Ranks)
    - Binary [parameters]
    - Example:
      - mpirun -host localhost -n 3 test : -host mic0 -n 3 test.mic : -host mic1 -n 4 test.mic

Message Passing Interface (MPI)

Mpirun supports the execution of a MPI application across several nodes using SSH (Secure Shell)



Infiniband: computer-networking communications standard used in high-performance computing

- Very high throughput;
- Very low latency.

Heterogeneous communication

Intra node

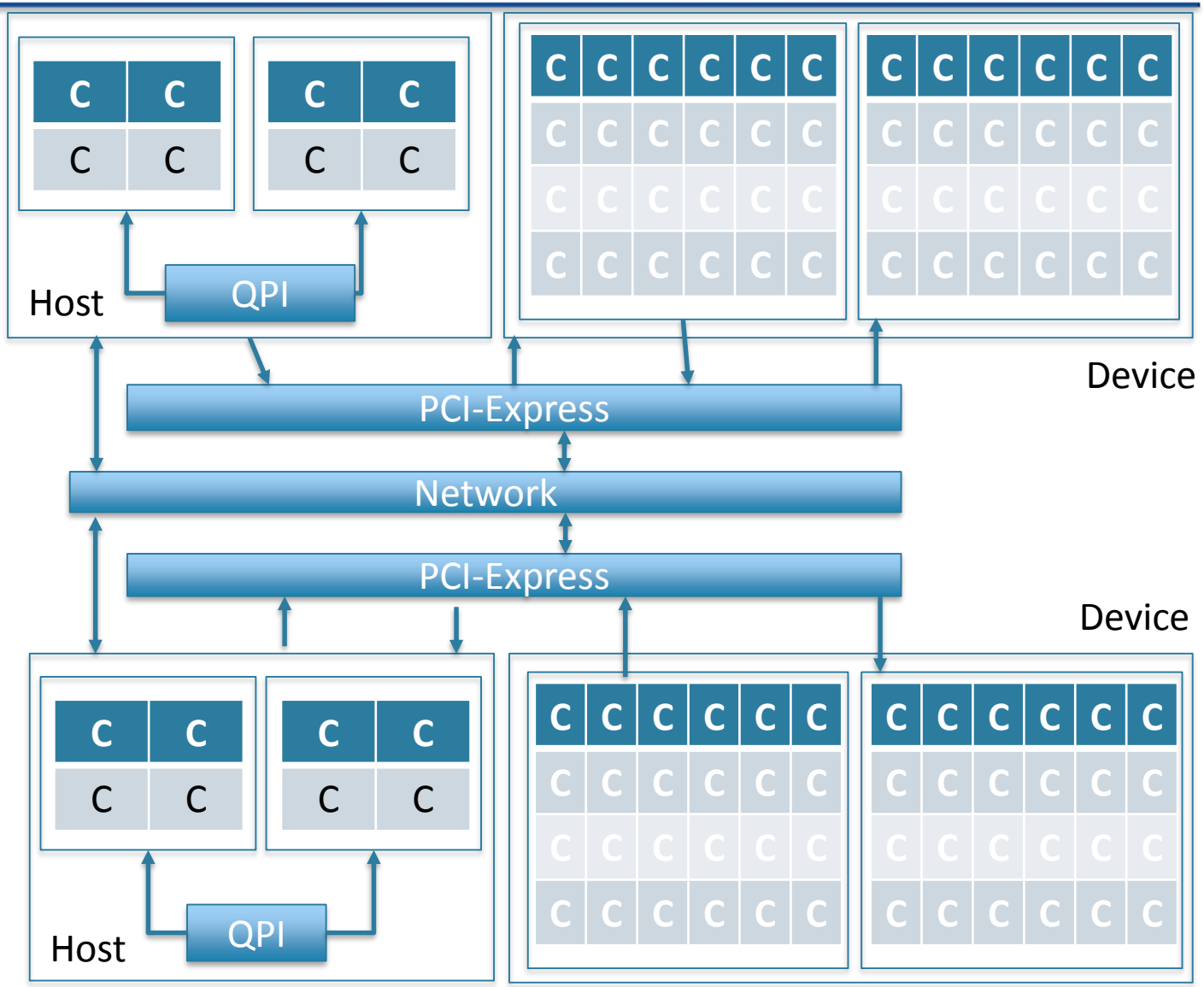
- Intra socket
- Inter sockets (Quick Path Interconnect)
- Host-MIC
- Inter MIC

Inter node

- Inter Host
- Intel MIC
- Mixed host and MIC

Network

- Ethernet
- Infiniband
- …



Agenda

- Message Passing Interface (MPI)
- **MPI Programming**
- Profiling MPI
- Combining MPI/OpenMP/Offload/Vecorization (Affinity)

Hello World - MPI

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
  int rank, size;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  char hostname[1024];
  gethostname(hostname, 1024);

  printf("Rank: %d of total: %d Hostname: %s\n", rank, size, hostname);

  MPI_Finalize();

  return 0;
}
```

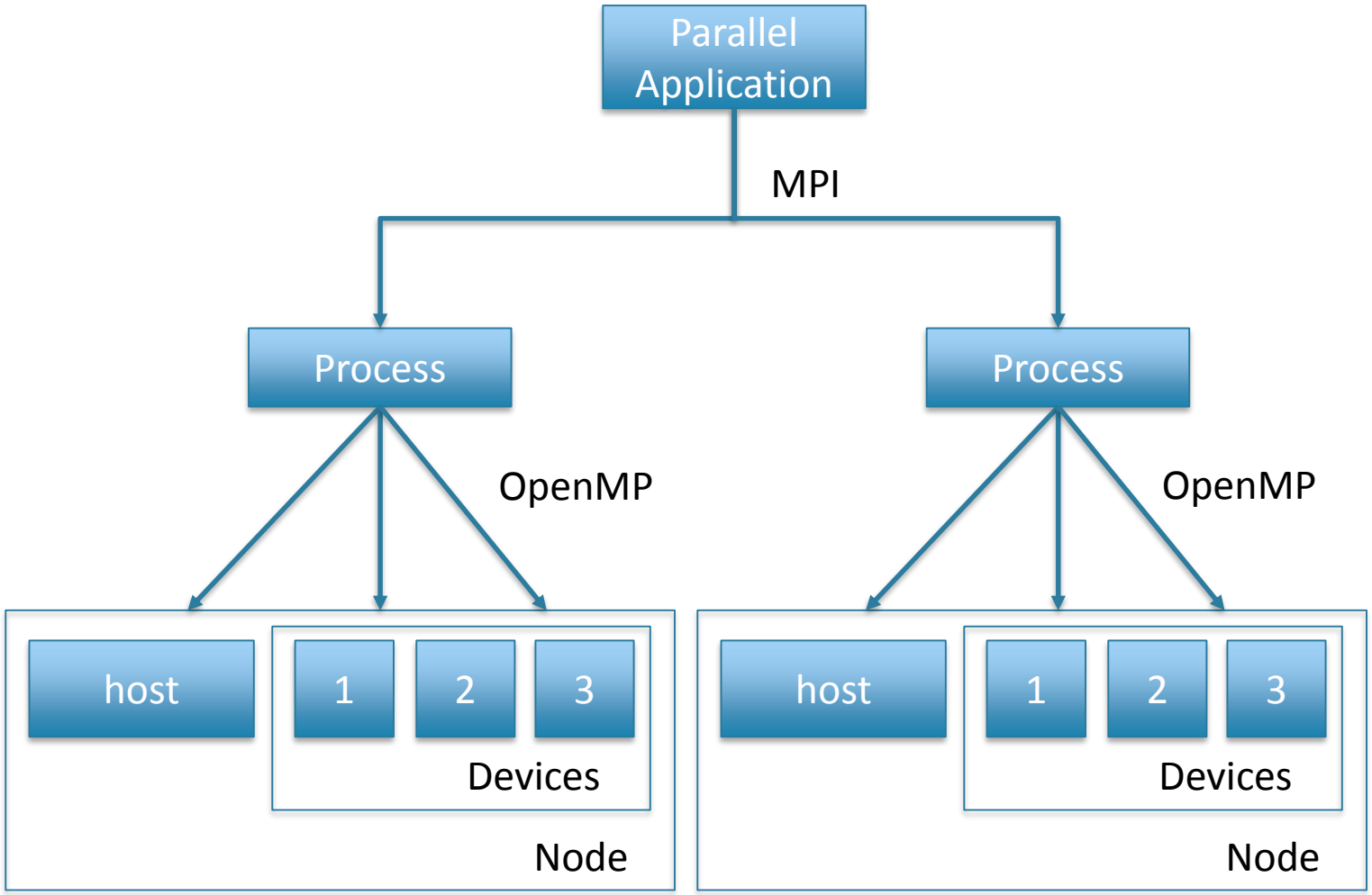
Vector Operation (OpenMP Version)

```
int main(int argc, char *argv[])
{
  int nn, i, j;

  double *a;

  nn=20;

  a = (double*) malloc(nn*sizeof(double));

  for( i=0; i<nn; i++)
    a[i] = 2;

  #pragma omp for
  for( j=0; j<nn; j++)
    a[j] = a[j] + cos(a[j]);

  for( j=0; j<nn; j++)
    printf("a[%d]=%f\n", j, a[j]);

  free(a);
  return 0;
}
```

Vector Operation (MPI Version)

```
#include <mpi.h>

int main(int argc, char *argv[])
{
  int rank, size, nn, i, j, contRank, begin, end, rest, amountPerRank ;

  double *a;

  nn=20;
  a = (double*) malloc(nn*sizeof(double));

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Status status;

  if (rank == 0) {
    for( i=0; i<nn; i++) {
      a[i] = 2;
    }
    for( contRank=1; contRank<size; contRank++) { // rank 0 sends the
vectors
      MPI_Send(&a[0], nn, MPI_DOUBLE, contRank, 1 ,
MPI_COMM_WORLD);
    }
  } else {
    MPI_Recv(&a[0], nn, MPI_DOUBLE, 0, 1, MPI_COMM_WORLD,
&status);
  }
  if (rank != 0) {
    amountPerRank = (nn / (size-1));
    rest = (nn % (size-1));
    begin=(rank-1)*amountPerRank;
    end=amountPerRank*rank;

    if (rank == (size -1))
      end=end+rest;

    printf("rank %d begin %d end %d rest %d amountPerRank %d \n",
rank, begin,end, rest , amountPerRank);

    for( j=begin; j < end ; j++) {
      a[j] = a[j] + cos(a[j]);
    }

    for( j=begin; j < end ; j++) {
        printf("Rank: %d of total: %d a[%d]=%f\n", rank, size, j, a[j]);
    }
  }

  MPI_Barrier(MPI_COMM_WORLD);
  MPI_Finalize();

  free(a);
  return 0;
}
```

OpenMP x MPI

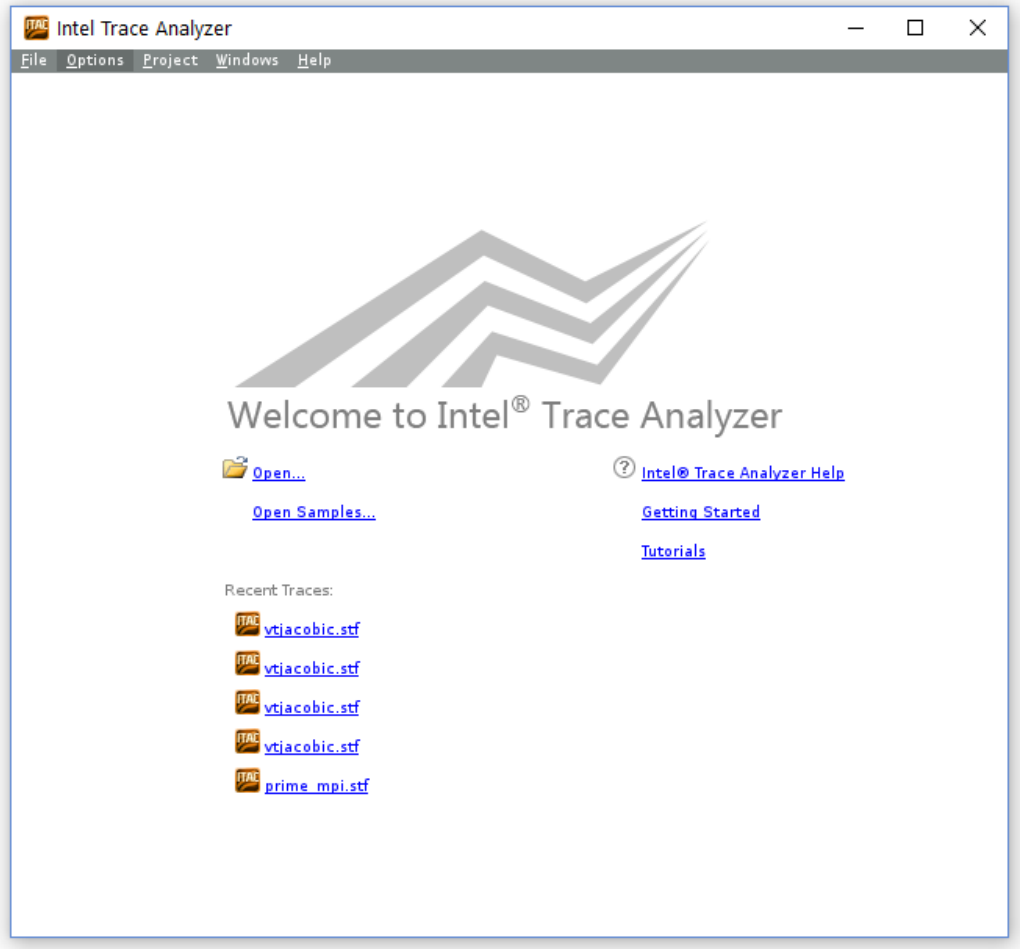

Parallel Application
MPI
Process
Process
OpenMP
OpenMP
host
1
2
3
Devices
Node
host
1
2
3
Devices
Node

Agenda

- Message Passing Interface (MPI)
- MPI Programming
- **Profiling MPI**
- Combining MPI/OpenMP/Offload/Vecorization (Affinity)

Profiling MPI

- MPI Performance Snapshot
  - -mps option in mpirun
  - Generate a simple report
- Intel Trace Analyzer
  - -trace
  - -generate itac report



Itac Example

/home/silvio/traces/3/vtjacobic.stf

Summary: vtjacobic.stf

Total time: **146** sec. Resources: **40** processes, **2** nodes.

Ratio

This section represents a ratio of all MPI calls to the rest of your code in the application.

Serial Code - 96.6 sec	66.1 %
MPI calls - 49.4 sec	33.8 %

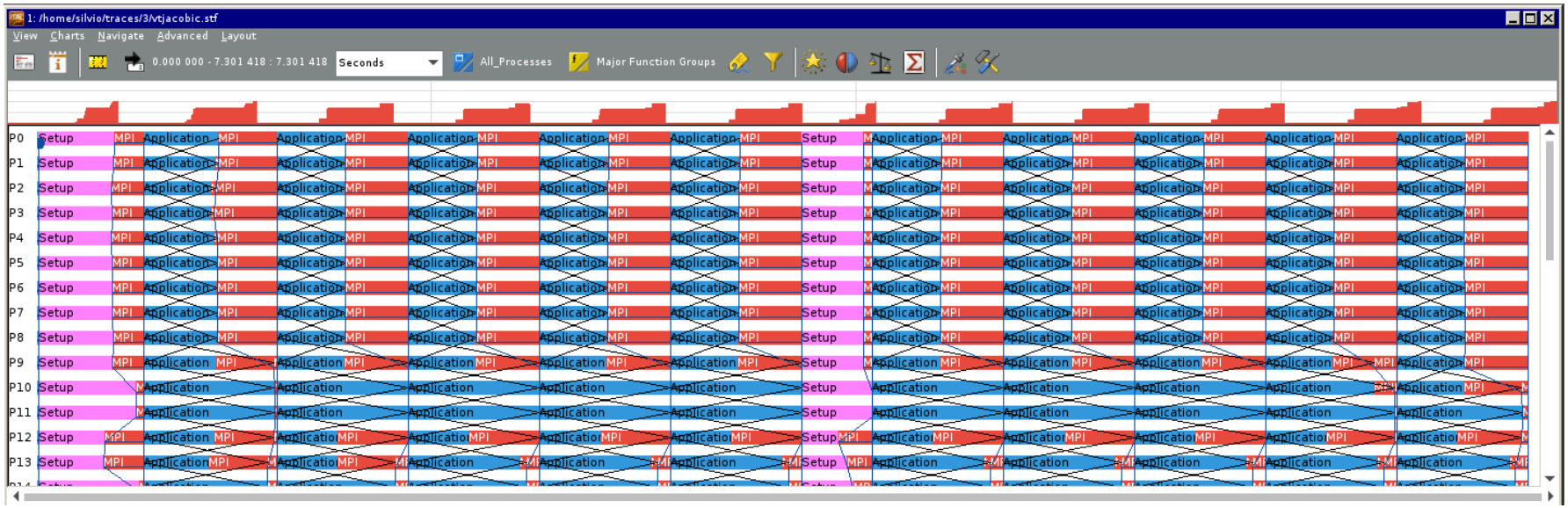
Itac Example

Continue >

Top MPI functions

This section lists the most active MPI functions from all MPI calls in the application.

Itac Example



NAS Parallel Benchmarks (NPB)

- http://www.nas.nasa.gov/publications/npb.html

- Several Benchmarks that mimic the computation and data movement in CFD applications;

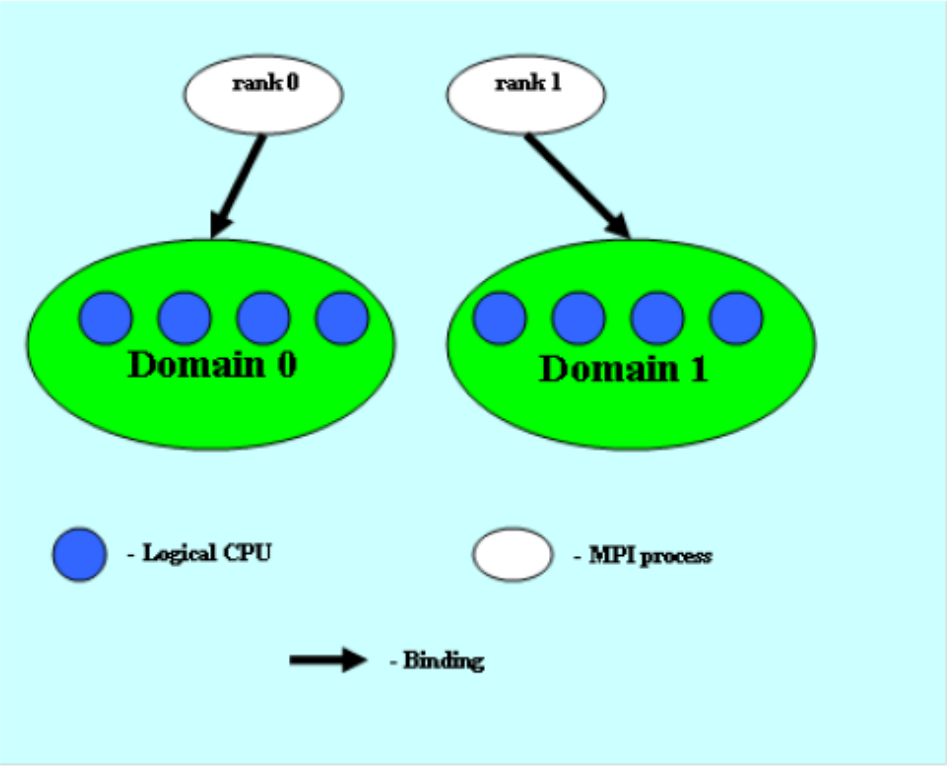
- Comparing the execution of NAS using Ethernet and Infiniband;

Agenda

- Message Passing Interface (MPI)
- MPI Programming
- Profiling MPI
- **Combining MPI/OpenMP/Offload/Vecorization (Affinity)**

Intel® MPI Support of Hybrid Codes (Affinity)

- Define I_MPI_PIN_DOMAIN to split logical processors into non-overlapping subsets
- Mapping rule: 1 MPI process per 1 domain



Pin OpenMP threads inside the domain with **KMP_AFFINITY** (or in the code)

I_MPI_PIN_DOMAIN

- I_MPI_PIN_DOMAIN=<multi-core-shape>

<mc-shape>	Define domains through multi-core terms.
core	Each domain consists of the logical processors that share a particular core. The number of domains on a node is equal to the number of cores on the node.
socket | sock	Each domain consists of the logical processors that share a particular socket. The number of domains on a node is equal to the number of sockets on the node. This is the recommended value.
node	All logical processors on a node are arranged into a single domain.
cache1	Logical processors that share a particular level 1 cache are arranged into a single domain.
cache2	Logical processors that share a particular level 2 cache are arranged into a single domain.
cache3	Logical processors that share a particular level 3 cache are arranged into a single domain.
cache	The largest domain among cache1, cache2, and cache3 is selected.

I_MPI_PIN_DOMAIN

- I_MPI_PIN_DOMAIN=<Explicit shape <size>:<layout>>

<size>	Define a number of logical processors in each domain (domain size)
omp	The domain size is equal to the OMP_NUM_THREADS environment variable value. If the OMP_NUM_THREADS environment variable is not set, each node is treated as a separate domain.
auto	The domain size is defined by the formula size=#cpu/#proc, where #cpu is the number of logical processors on a node, and #proc is the number of the MPI processes started on a node
<n>	The domain size is defined by a positive decimal number *<n>*

<layout>	Ordering of domain members. The default value is compact
platform	Domain members are ordered according to their BIOS numbering (platform-depended numbering)
compact	Domain members are located as close to each other as possible in terms of common resources (cores, caches, sockets, etc.). This is the default value
scatter	Domain members are located as far away from each other as possible in terms of common resources (cores, caches, sockets, etc.)

Intel® MPI Environment Support

- Debug I_MPI_DEBUG
  - Print out debugging information when an MPI program starts running. I_MPI_DEBUG=<level>[,<flags>]

#debug

export I_MPI_DEBUG=4

#mapping MPI only compact

mpirun -env I_MPI_PIN_DOMAIN auto:compact -env I_MPI_DEBUG 4 -host mic0 -n 4 /tmp/mpitest

#mapping MPI only compact

mpirun -env I_MPI_PIN_DOMAIN auto:scatter -env I_MPI_DEBUG 4 -host mic0 -n 4 /tmp/mpitest

Prime numbers

- Prime numbers (https://people.sc.fsu.edu/~jburkardt/c_src/prime_mpi/prime_mpi.c)
- Comparing execution on one node or two nodes:
  - mpirun -trace -host 10.10.30.3 -n 60 /home/silvio/mpi/prime_mpi
  - mpirun -trace -host 10.10.30.3 -n 60 /home/silvio/mpi/prime_mpi : -host 10.10.30.2 -n 60 /home/silvio/mpi/prime_mpi