TDDC78 Lab Series

Sehrish Qummar, 2023
Credit to: August Ernstsson

Outline

- **Organization**:
  Workflow, demonstrations, reports, resources

- **Assignments**:
  Description of each lab and some hints

Organization

Lab Groups

- **Group A**: Sehrish Qummar (course assistant)
- **Group B**: Sehrish Qummar (course assistant)
- Send reports to your assigned assistant.
- Only one assistant guaranteed present per session.

Lab Assignments

- Lab 1: **Image filters**
  - a) Pthreads (shared memory)
  - b) MPI (distributed memory)
- Lab 2: **Heat solver**, OpenMP (shared memory)
- Miniproject: **Particle simulation**, MPI (distributed memory)
  - Written report and mandatory use of DDT, ITAC

Lab Structure

Title	Lab 1a	Lab 1b	Lab 2	Miniproject
Topic	Image filtering		Heat propagation	Particle simulation
Concepts	Pthreads	MPI	OpenMP	MPI
Tools (DDT / ITAC)	Encouraged	Encouraged	Encouraged	**Mandatory**
Demonstration	**Yes**	**Yes**	**Yes**	**Yes**
Written report	No	No	No	**Yes**
Sched. time	4 hours	4 hours	4 hours	6 hours
Soft deadline	12/4 A 13/4 B	26/4 A 17/4 B	5/5 A 3/5 B	17/5 A 16/5 B

Workflow

- Terminal on IDA computers -> log in to Sigma
  - ssh username@sigma.nsc.liu.se
- Also possible to use ThinLinc to access Sigma desktop env.
- Sometimes possible to develop locally (shared memory)
- Usage of own computer
  - Log in to Sigma as usual
  - Local development may require installing e.g. OpenMPI

Demonstrations

- Lab 1 a+b (separate or together), 2, and miniproject.
- Show and explain your code to the assistant.
  - **Illustrations** can help explaining!
- Performance measurements:
  Have **plots** ready from multiple runs to show scaling.
- Be prepared to do at least one test run live.

Miniproject

- Demonstrate your program as usual (You get a ”D” in WebReg)
- Write a report (aim for *at least* 5 pages including figures and code snippets) explaining your approach to solving the problem.
- Suggested outline on the course web page.
- Try to follow the PCAM model
- **An image says more than a thousand words!** Make illustrations that
  - Show your problem decomposition, etc
  - Show performance results
- Send via email to your assistant, title ”**TDDC78: Report**”
  (write LiU IDs and WebReg group number in email and document)

Information Resources

- Lab compendium
- Source files
- NSC + TDDC78 lecture, lesson slides
- NSC website + other online resources (e.g. MPI docs)
- Quick reference sheet (handout)

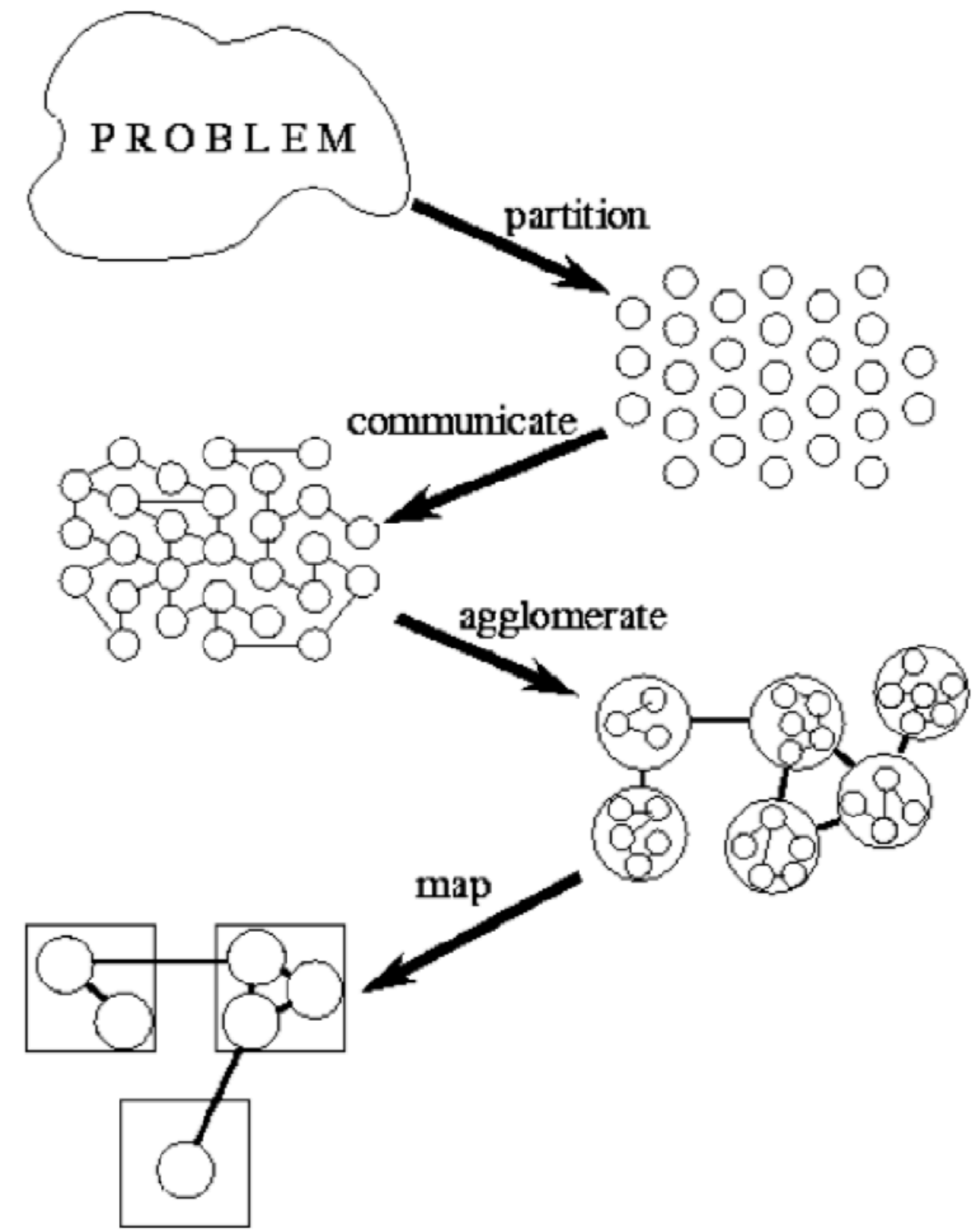
Suggestions

- Create Makefiles for compiling
- Create scripts for performance measurements
  (Somewhat outside the course scope, but it can be very powerful)
- Establish a good (automated?) plotting workflow
- Use Git for managing files across IDA and Sigma
  - LiU Gitlab: https://gitlab.liu.se

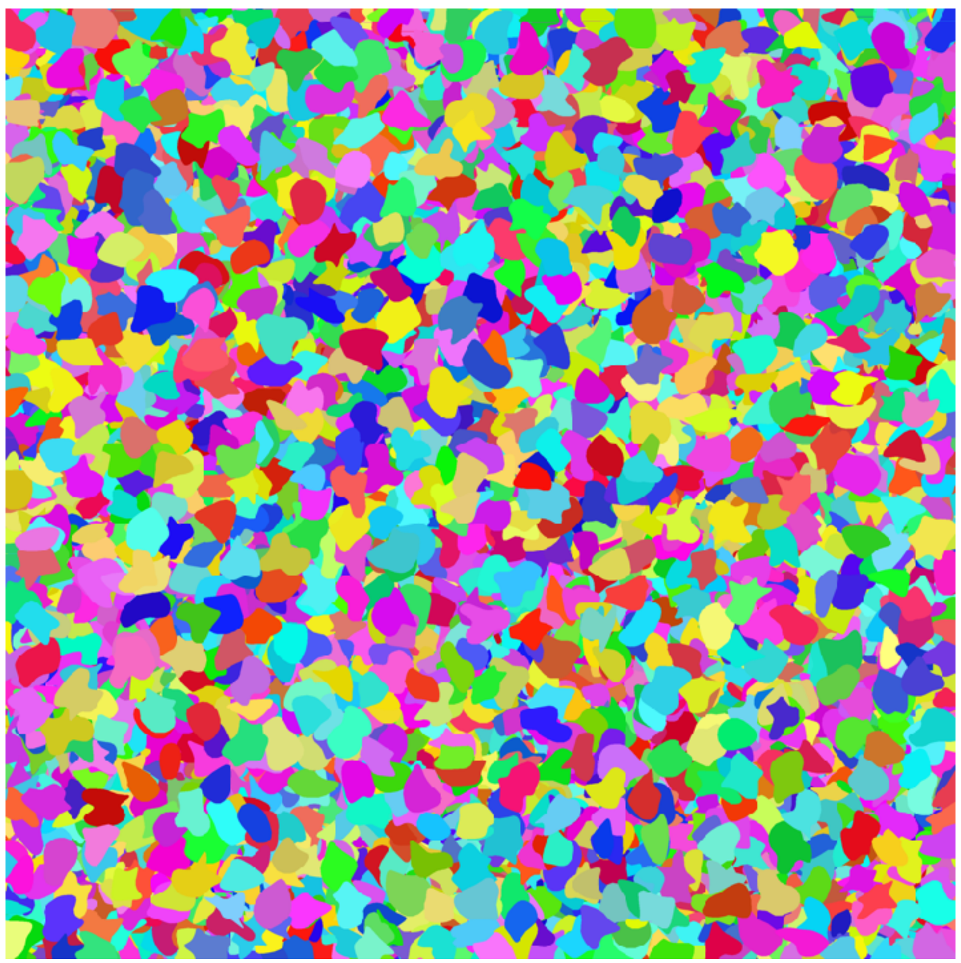
Assignments

"PCAM" model

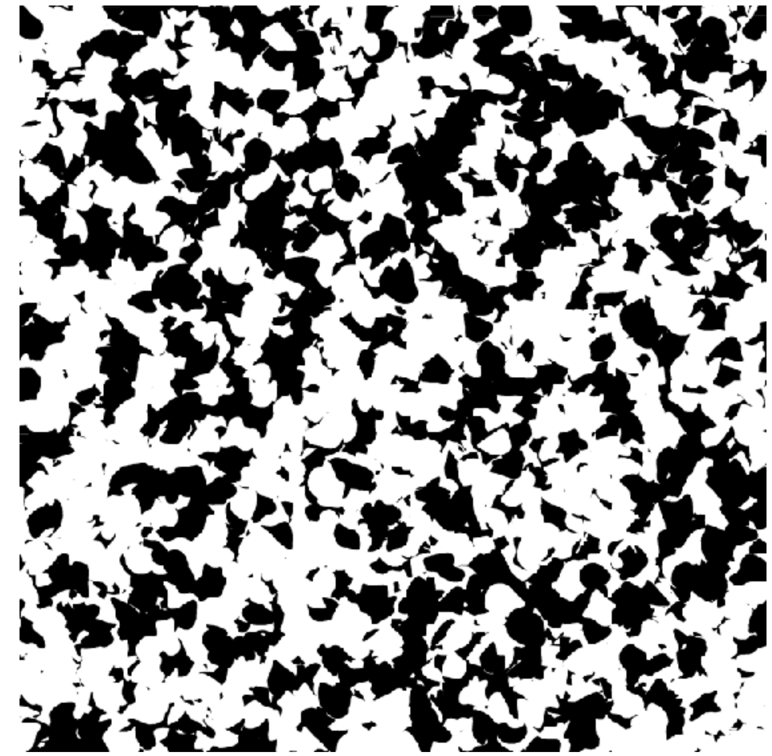
- Partitioning
  - Domain decomposition
  - Functional decomposition
- Communication + synchronization
- Agglomeration
- Mapping + Load balancing



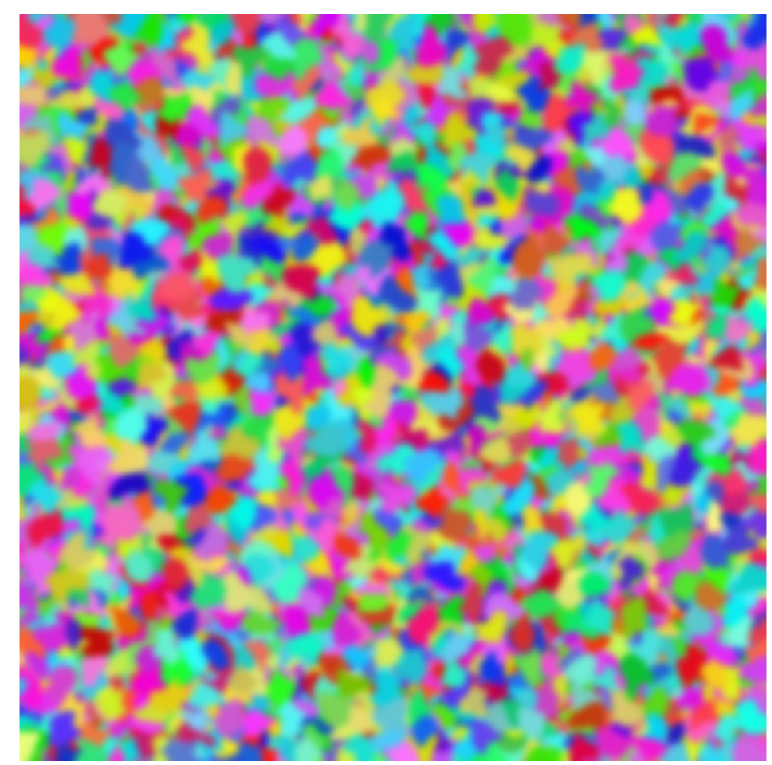
Lab 1: Image filters



Threshold →



Blur →



- Task partitioning.
  Consider different approaches.

Lab 1 a: Pthreads

```
struct thread_data {
    int threadId;
    char *msg;
};

struct thread_data thread_data_array[NUM_THREADS];

void *PrintHello(void *tParam) {
    struct thread_data *myData;
    myData = (struct thread_data *) tParam;
    taskId = myData->threadId;
    helloMsg = myData->msg;
}

int main (int argc, char *argv[]) {
    ...
    thread_data_array[t].threadId = t;
    thread_data_array[t].msg = msgPool[t];
    rc = pthread_create(&threads[t], NULL, PrintHello,
                        (void *) &thread_data_array[t]);
```

Lab 1 a: Pthreads

```
#include<pthread.h>

pthread_mutex_t count_mutex = ... ;
long count;

void increment_count() {
  pthread_mutex_lock(&count_mutex);
  count = count + 1;
  pthread_mutex_unlock(&count_mutex);
}

long get_count() {
  long c;
  pthread_mutex_lock(&count_mutex);
  c = count;
  pthread_mutex_unlock(&count_mutex);
  return (c);
}
```

Lab 1 b: MPI

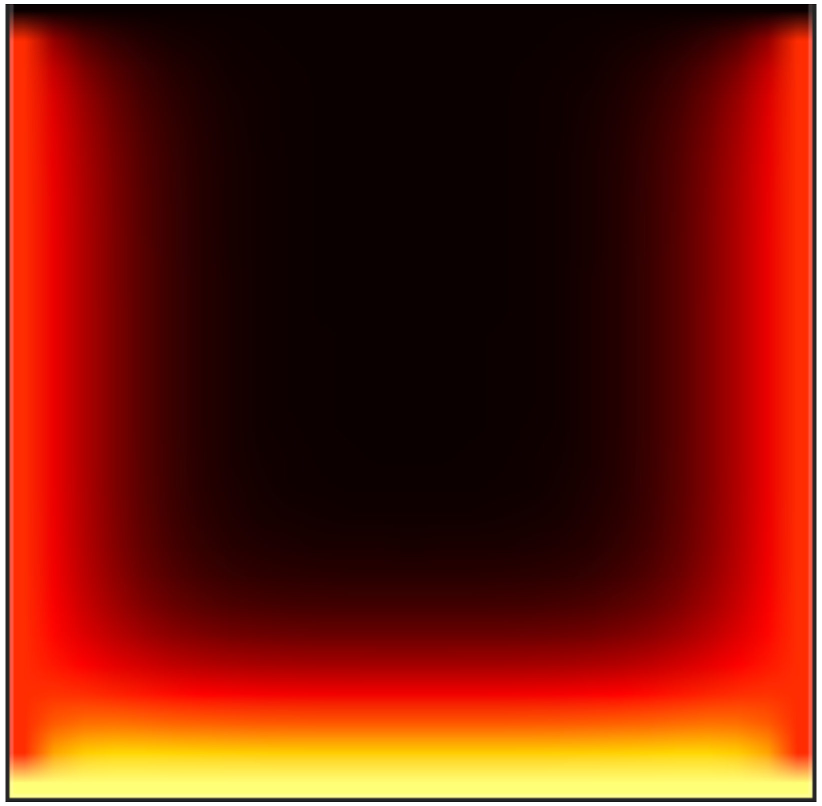
- MPI concepts: (Refer to lectures and documentation)
  - Define type (a Pixel type)
  - Send / Receive
  - Broadcast
  - Scatter / Gather

MPI Type

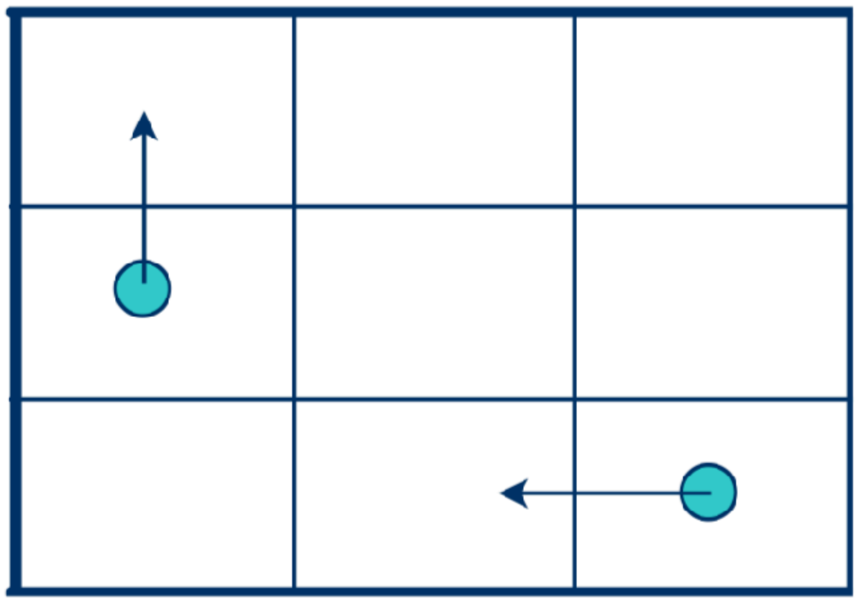
```
typedef struct {
    int id;
    double data[10];
} buf_t; // Composite type
buf_t item; // Element of the type

MPI_Datatype buf_t_mpi; // MPI type to commit
int block_lengths [] = { 1, 10 }; // Lengths of type elements
MPI_Datatype block_types [] = { MPI_INT, MPI_DOUBLE }; //Set types
MPI_Aint start, displ[2];

MPI_Get_address( &item, &start );
MPI_Get_address( &item.id, &displ[0] );
MPI_Get_address( &item.data[0], &displ[1] );
displ[0] -= start; // Displacement relative to address of start
displ[1] -= start; // Displacement relative to address of start MPI_Type_create_struct( 2, block_lengths, displ,
block_types, &buf_t_mpi );
MPI_Type_commit( &buf_t_mpi );
```

Lab 2: Heat solver

- **Problem**: Find stationary temperature distribution in a (NxN) square given some boundary temperature distribution
- **Solution**: Requires solving differential equation
    - Iterative Jacobi method
      Detailed algorithm in Compendium
- Primary concerns:
    - Shared memory, OpenMP
      (Refer to lectures)
    - Synchronize access
    - **O(N) extra memory**



Miniproject



- Moving particles
- Validate the pressure law: pV = nRT (how?)
- Dynamic interaction patterns:
  # of particles that fly across borders is not static.
- Approximations: when to check for collisions?
  Your baseline sequential comparison needs to apply the same approximations!
- You need advanced domain decomposition.
  Motivate your choice!
- Use debugging tools, tracing, software counters to convince yourselves that the approach is good

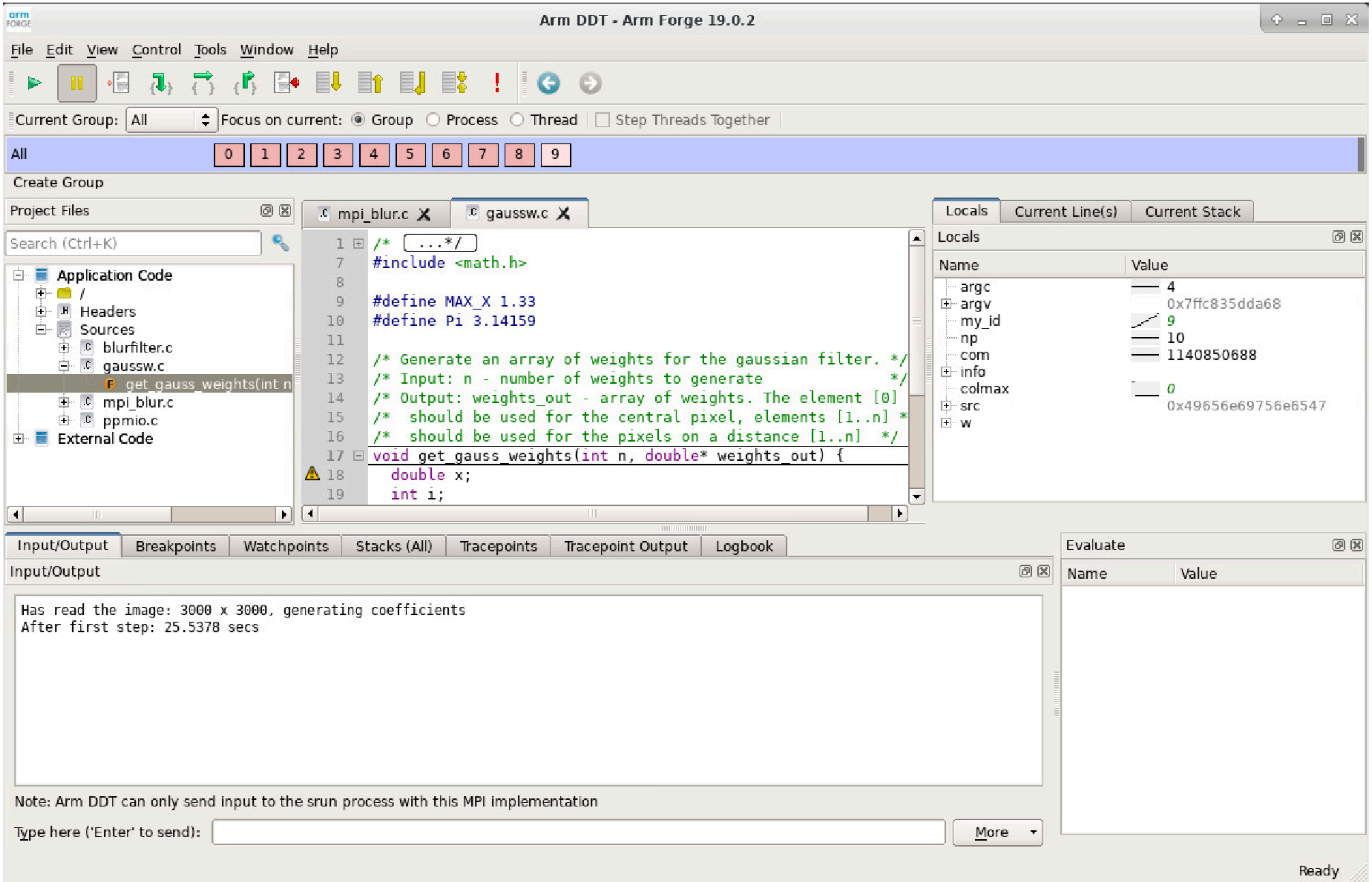
MPI Topologies (1)

```
int dims[2];  // 2D matrix / grid
dims[0] = 2;  // 2 rows
dims[1] = 3;  // 3 columns

MPI_Dims_create( nproc, 2, dims );
int periods[2];
periods[0] = 1; // Row-periodic
periods[1] = 0; // Column-non-periodic
int reorder = 1; // Re-order allowed

MPI_Comm grid_comm;
MPI_Cart_create( MPI_COMM_WORLD, 2, dims, periods,
                 reorder, &grid_comm );
```

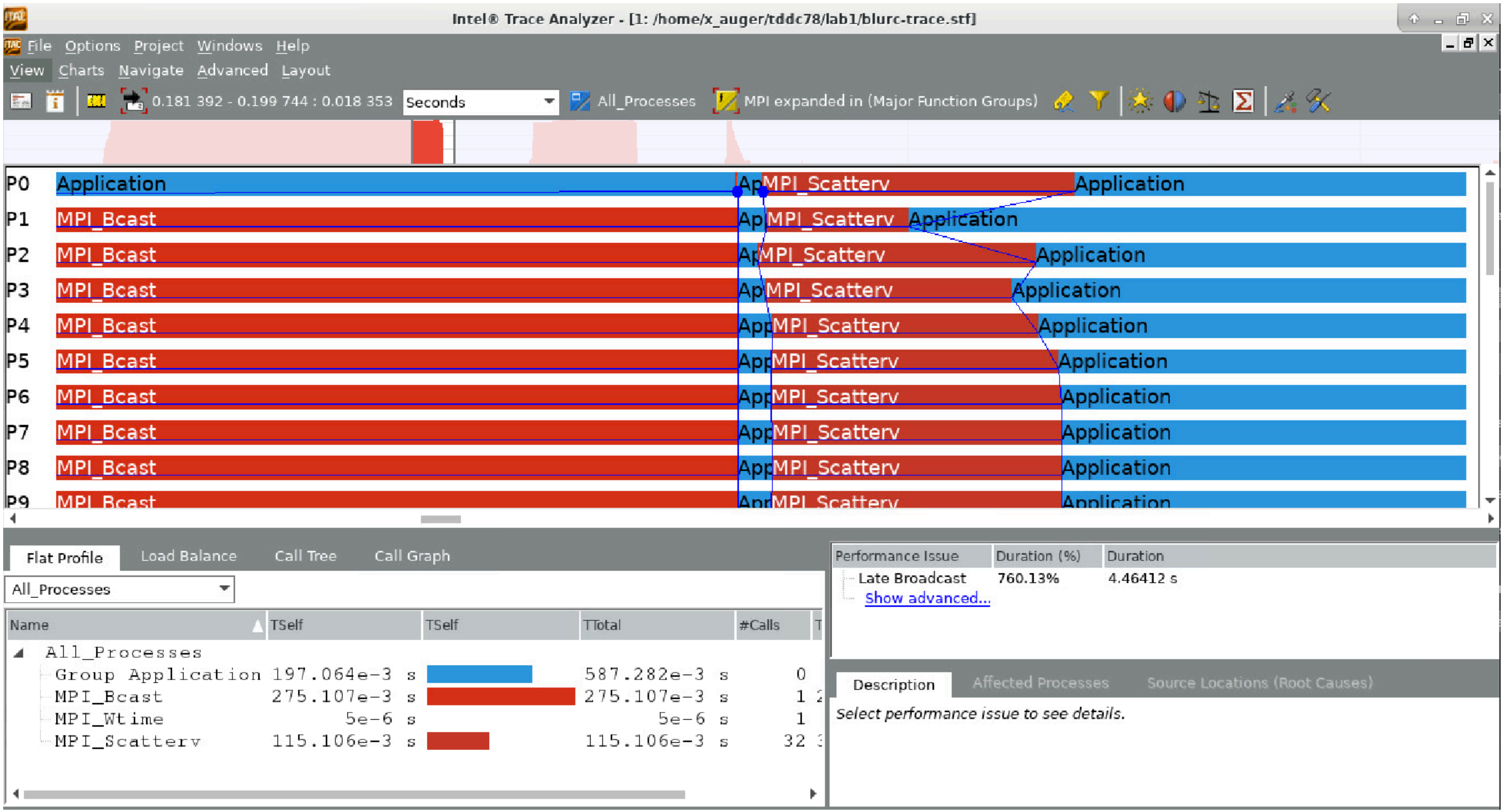
MPI Topologies (2)

```
int my_coords[2]; // Cartesian Process coordinates
int my_rank;      // Process rank
int right_nbr[2];
int right_nbr_rank;

MPI_Cart_get(  grid_comm, 2, dims, periods, my_coords);
MPI_Cart_rank( grid_comm, my_coords, &my_rank);

right_nbr[0] = my_coords[0]+1;
right_nbr[1] = my_coords[1];
MPI_Cart_rank( grid_comm, right_nbr, &right_nbr_rank);
```

DDT


Arm DDT - Arm Forge 19.0.2
File Edit View Control Tools Window Help
Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All
0 1 2 3 4 5 6 7 8 9
Create Group
Project Files
Search (Ctrl+K)
Application Code
/
Headers
Sources
blurfilter.c
gaussw.c
get_gauss_weights(int n
mpi_blur.c
ppmio.c
External Code
mpi_blur.c
gaussw.c
/* ...*/
#include <math.h>
#define MAX_X 1.33
#define Pi 3.14159
/* Generate an array of weights for the gaussian filter. */
/* Input: n - number of weights to generate */
/* Output: weights_out - array of weights. The element [0]
/* should be used for the central pixel, elements [1..n] *
/* should be used for the pixels on a distance [1..n] */
void get_gauss_weights(int n, double* weights_out) {
double x;
int i;
Locals
Current Line(s)
Current Stack
Name
Value
argc
4
argv
0x7ffc835dda68
my_id
9
np
10
com
1140850688
info
colmax
0
src
0x49656e69756e6547
w
Input/Output
Breakpoints
Watchpoints
Stacks (All)
Tracepoints
Tracepoint Output
Logbook
Has read the image: 3000 x 3000, generating coefficients
After first step: 25.5378 secs
Note: Arm DDT can only send input to the srun process with this MPI implementation
Type here ('Enter' to send):
More
Evaluate
Ready

ITAC



Intel® Trace Analyzer - [1: /home/x_auger/tddc78/lab1/blurc-trace.stf]
File Options Project Windows Help
View Charts Navigate Advanced Layout
0.181 392 - 0.199 744 : 0.018 353
Seconds
All_Processes
MPI expanded in (Major Function Groups)
P0 Application MPI_Scatterv Application
P1 MPI_Bcast MPI_Scatterv Application
P2 MPI_Bcast MPI_Scatterv Application
P3 MPI_Bcast MPI_Scatterv Application
P4 MPI_Bcast MPI_Scatterv Application
P5 MPI_Bcast MPI_Scatterv Application
P6 MPI_Bcast MPI_Scatterv Application
P7 MPI_Bcast MPI_Scatterv Application
P8 MPI_Bcast MPI_Scatterv Application
Flat Profile
Load Balance
Call Tree
Call Graph
All_Processes
Name TSelf TSelf TTotal #Calls
All_Processes
Group Application 197.064e-3 s 587.282e-3 s 0
MPI_Bcast 275.107e-3 s 275.107e-3 s 1
MPI_Wtime 5e-6 s 5e-6 s 1
MPI_Scatterv 115.106e-3 s 115.106e-3 s 32
Performance Issue Duration (%) Duration
Late Broadcast 760.13% 4.46412 s
Show advanced...
Description
Affected Processes
Source Locations (Root Causes)
Select performance issue to see details.

How much parallelism?

- Always measure parallel code on 1 thread/process
  - Reference for speedup
  - Note: Not the same as measuring sequential code!
- Then measure on at least ”powers of 2” threads/procs.
  - 1, 2, 4, 8, 16, …
  - Shared memory: Up to all the available processor cores
  - Distributed memory: Up to at least 2 nodes, at most 4 nodes

Questions?