Parallel Systems, Lab. 2
Shared Memory Programming

Guy Tel-Zur
guycomputing@gmail.com

Agenda

- OpenMP
- MPI+OpenMP
- Cilk Plus
- UPC

Working with Allineaś DDT

- Demo: /home/telzur/allinea/ddt_training
- Material is available at the course MTA web site.

Example 1: OpenMP

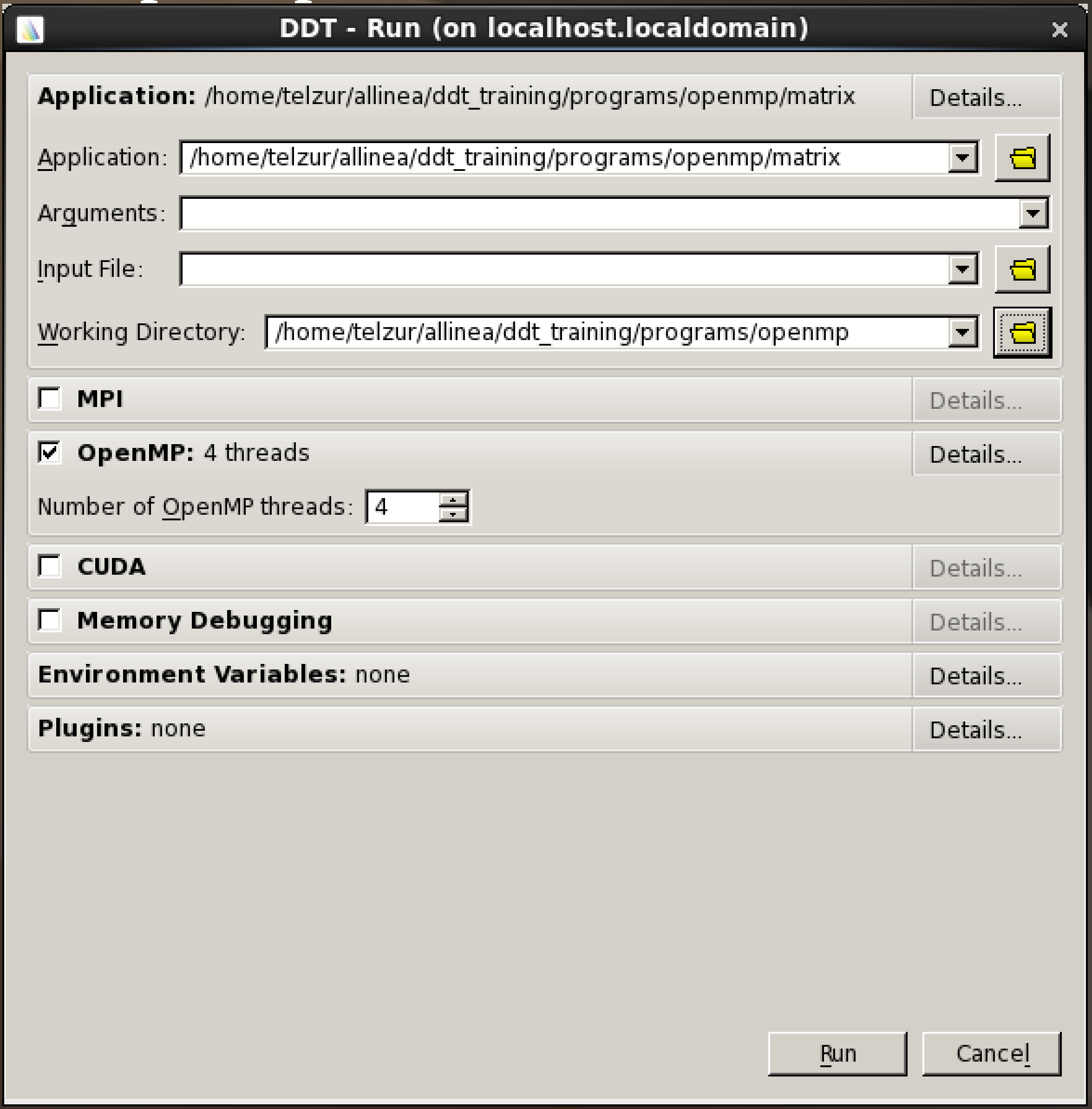
- cd to ¨openmp¨
- Open the matrix.c file in an editor and understand the code which is a parallel matrix multiply.
- Change the size from 16 to 4 so that the printout will be readable.
- Make:

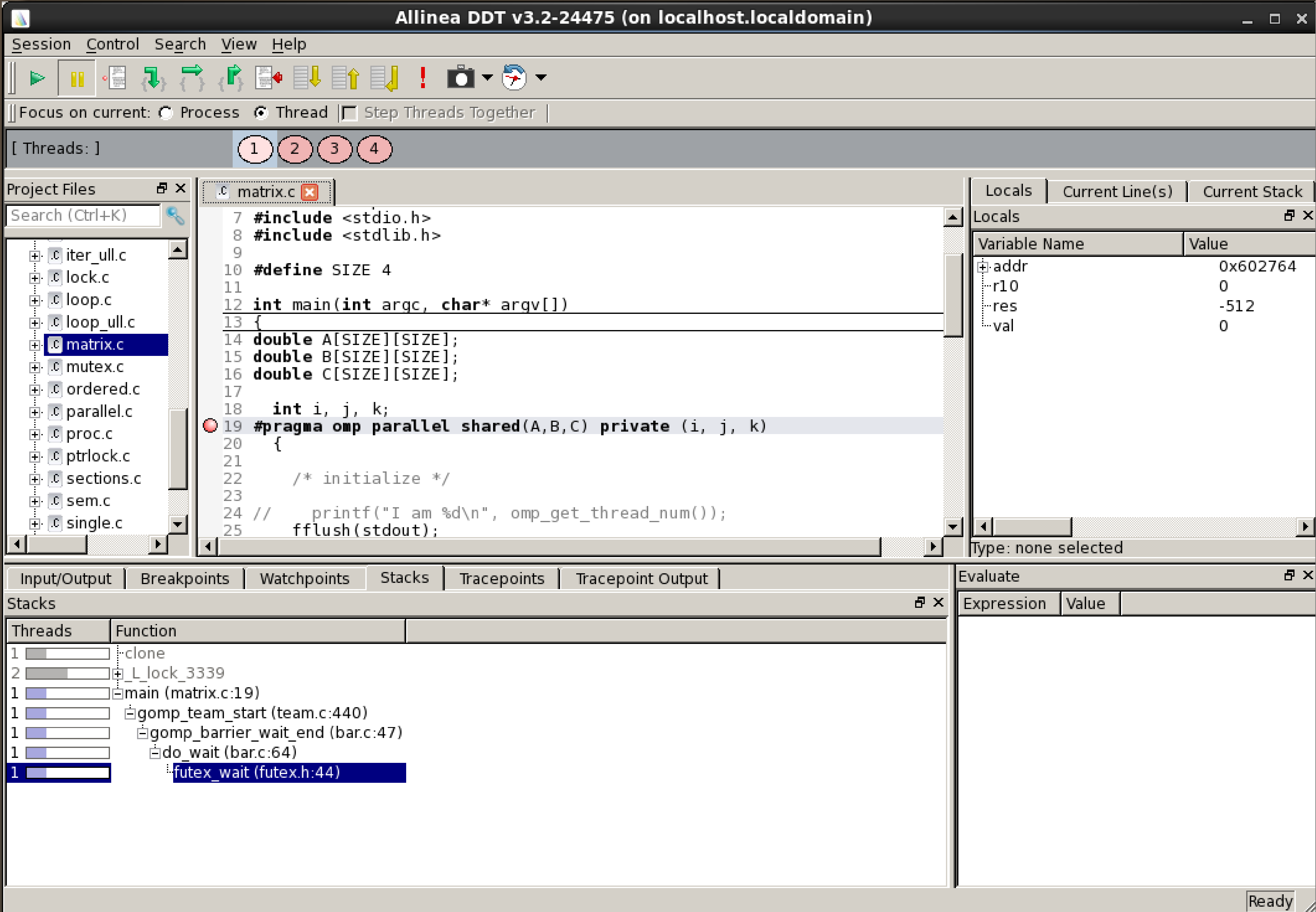
```
[root@localhost openmp]# make

gcc -c -g -O0 -fopenmp -o matrix.o matrix.c

gcc -g -O0 -fopenmp  -o matrix matrix.o
```

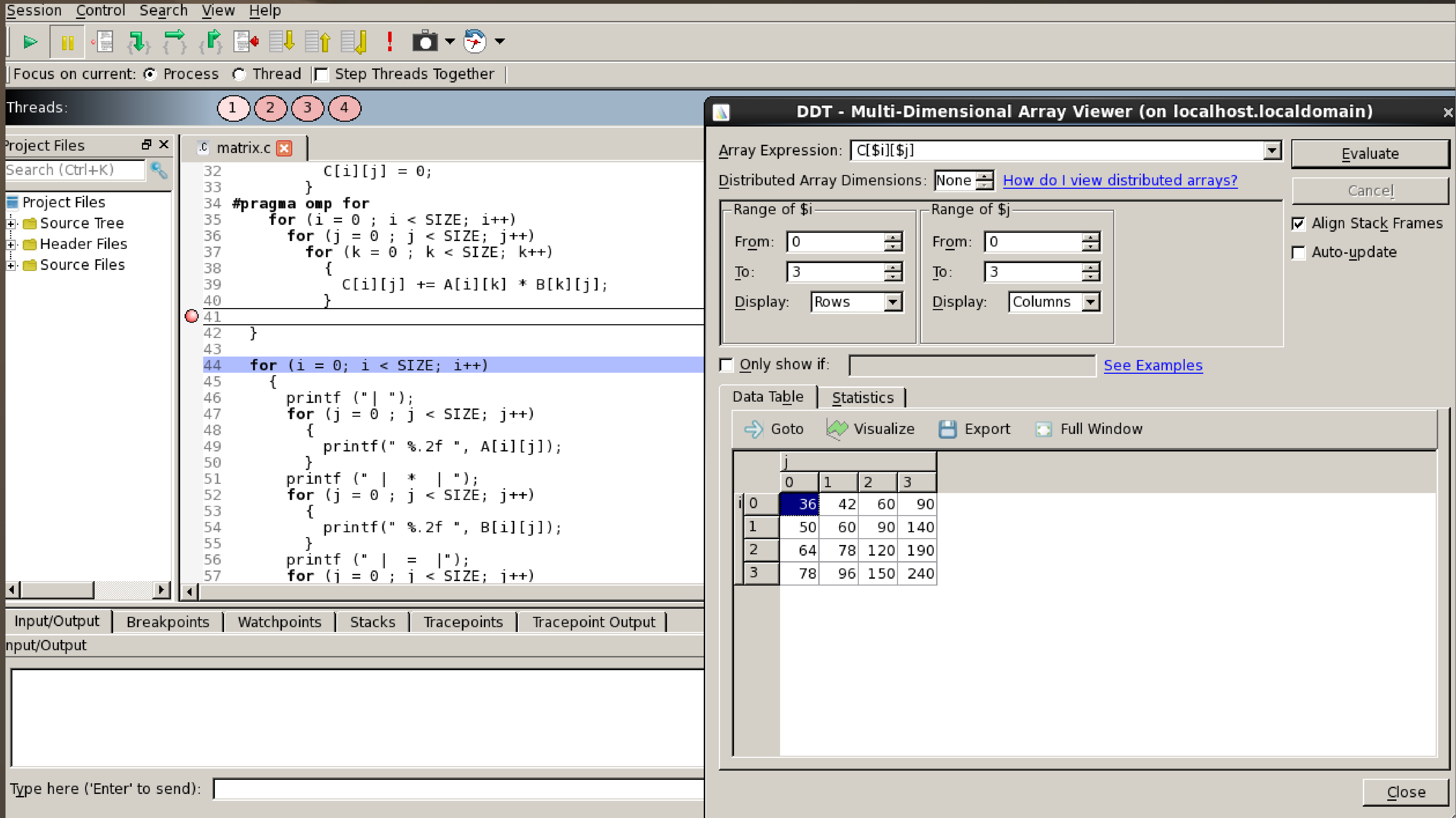
```
[root@localhost openmp]# ll
total 40
-rw-------. 1 19258   23   365 Mar  7  2012 Makefile
-rwxr-xr-x. 1 root  root 12461 Jan 14 19:19 matrix
-rw-------. 1 19258   23  1197 May 15  2012 matrix.c
-rw-r--r--. 1 root  root  8360 Jan 14 19:19 matrix.o
-rw-------. 1 19258   23  1237 Mar  7  2012 nompmatrix.c
```

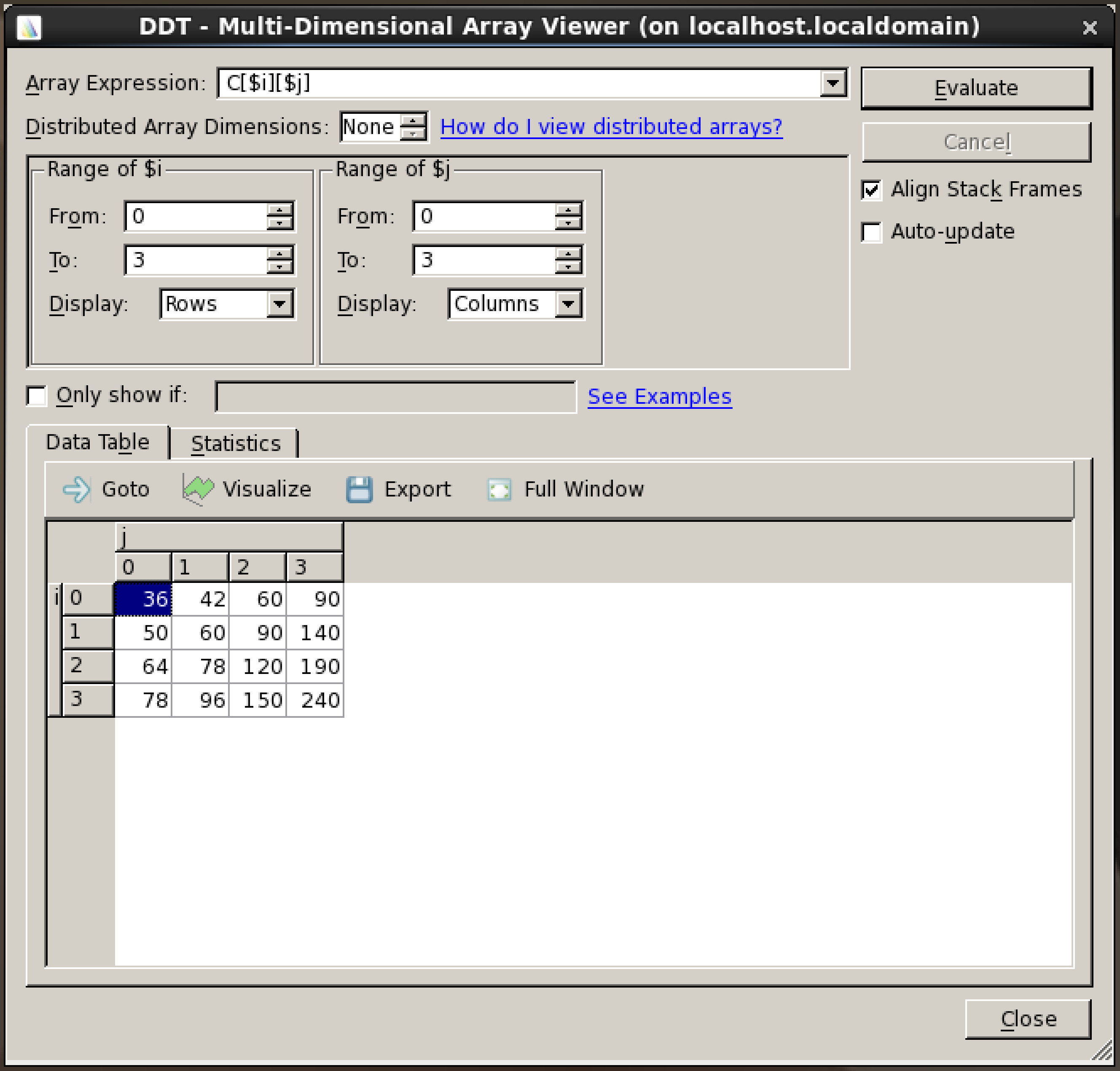
```
[root@localhost openmp]# export OMP_NUM_THREADS=2
[root@localhost openmp]# ./matrix
|  0.00  1.00  2.00  3.00  |  *  |  0.00  1.00  4.00  9.00  |  =  | 36.00  42.00  60.00  90.00  |
|  1.00  2.00  3.00  4.00  |  *  |  1.00  2.00  5.00  10.00  |  =  | 50.00  60.00  90.00  140.00  |
|  2.00  3.00  4.00  5.00  |  *  |  4.00  5.00  8.00  13.00  |  =  | 64.00  78.00  120.00  190.00  |
|  3.00  4.00  5.00  6.00  |  *  |  9.00  10.00  13.00  18.00  |  =  | 78.00  96.00  150.00  240.00  |
[root@localhost openmp]#
```

Bash: export OMP_NUM_THREADS=4
csh: setenv OMP_NUM_THREADS 4
[root@localhost openmp]# ddt -start ./matrix

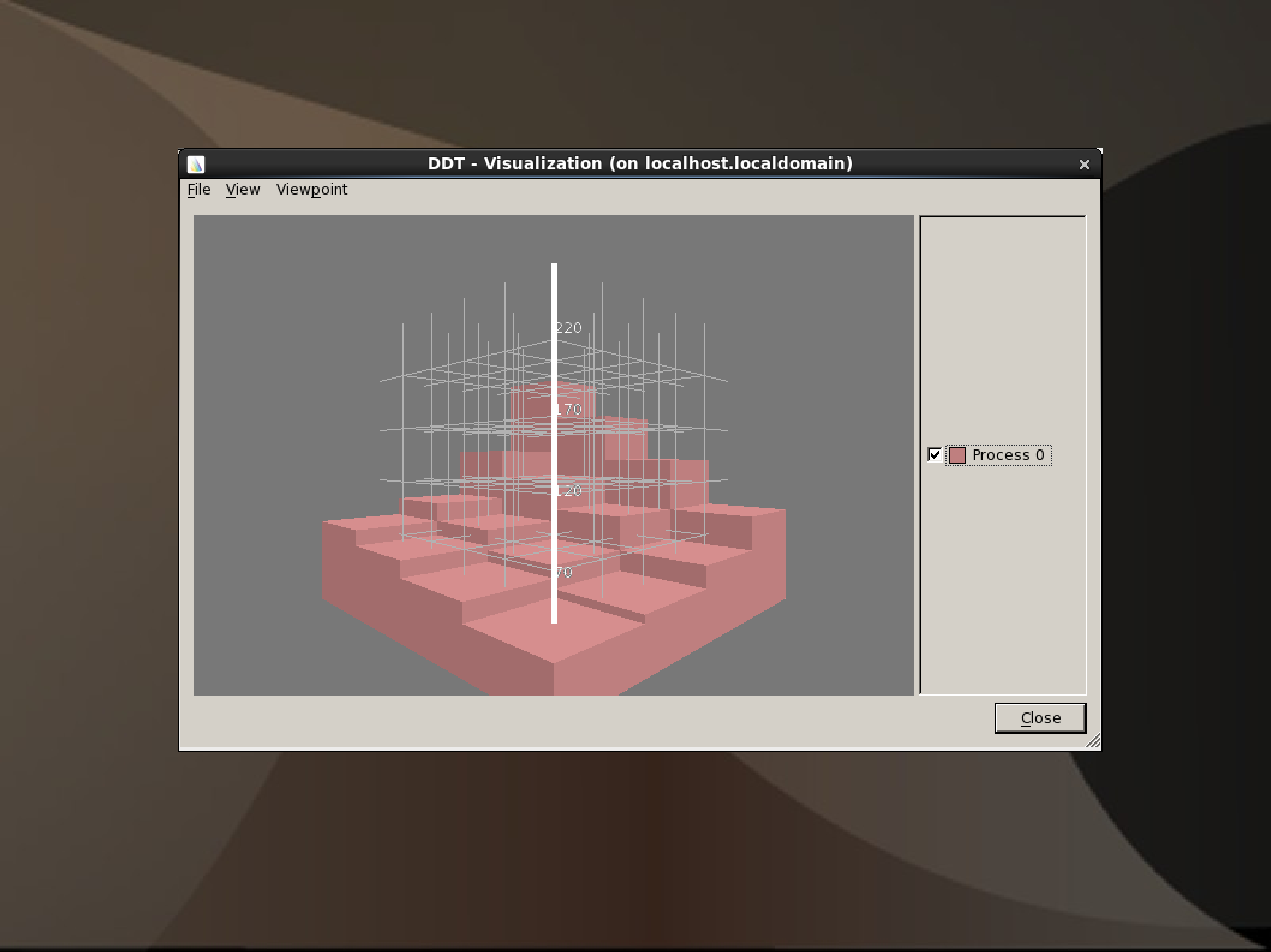



Allinea DDT v3.2-24475 (on localhost.localdomain)
Session Control Search View Help
Focus on current: Process Thread Step Threads Together
[Threads:] 1 2 3 4
Project Files
Search (Ctrl+K)
iter_ull.c
lock.c
loop.c
loop_ull.c
matrix.c
mutex.c
ordered.c
parallel.c
proc.c
ptrlock.c
sections.c
sem.c
single.c
matrix.c
7 #include <stdio.h>
8 #include <stdlib.h>
9
10 #define SIZE 4
11
12 int main(int argc, char* argv[])
13 {
14 double A[SIZE][SIZE];
15 double B[SIZE][SIZE];
16 double C[SIZE][SIZE];
17
18 int i, j, k;
19 #pragma omp parallel shared(A,B,C) private (i, j, k)
20 {
21
22 /* initialize */
23
24 // printf("I am %d\n", omp_get_thread_num());
25 fflush(stdout);
Locals Current Line(s) Current Stack
Locals
Variable Name Value
addr 0x602764
r10 0
res -512
val 0
Type: none selected
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output
Stacks
Threads Function
1 clone
2 _L_lock_3339
1 main (matrix.c:19)
1 gomp_team_start (team.c:440)
1 gomp_barrier_wait_end (bar.c:47)
1 do_wait (bar.c:64)
1 futex_wait (futex.h:44)
Evaluate
Expression Value
Ready

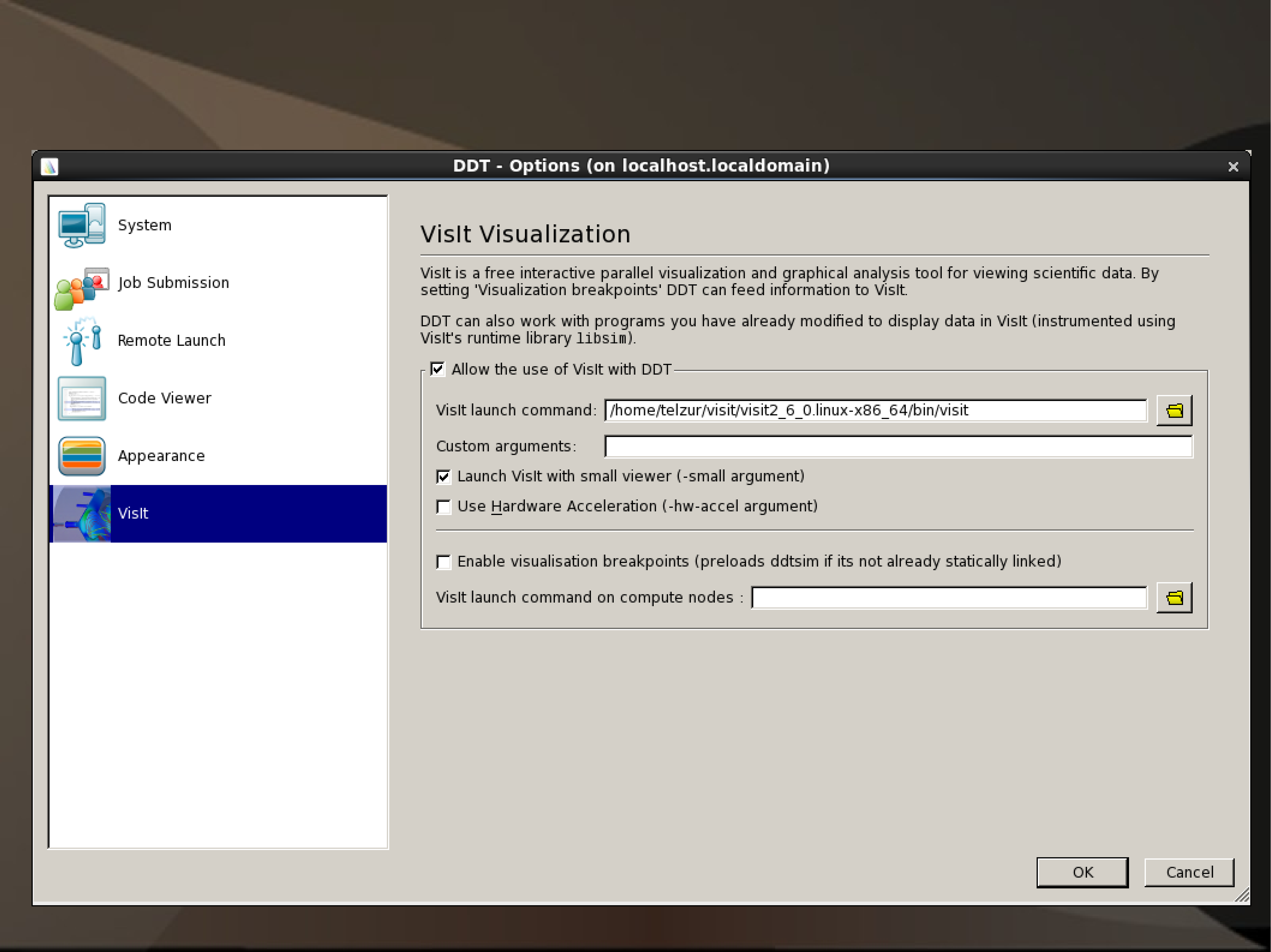

Session Control Search View Help
Focus on current: Process Thread Step Threads Together
Threads: 1 2 3 4
Project Files
Search (Ctrl+K)
Project Files
Source Tree
Header Files
Source Files
matrix.c
C[i][j] = 0;
}
#pragma omp for
for (i = 0 ; i < SIZE; i++)
for (j = 0 ; j < SIZE; j++)
for (k = 0 ; k < SIZE; k++)
{
C[i][j] += A[i][k] * B[k][j];
}
}
for (i = 0; i < SIZE; i++)
{
printf ("| ");
for (j = 0 ; j < SIZE; j++)
{
printf(" %.2f ", A[i][j]);
}
printf (" | * | ");
for (j = 0 ; j < SIZE; j++)
{
printf(" %.2f ", B[i][j]);
}
printf (" | = |");
for (j = 0 ; j < SIZE; j++)
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output
Input/Output
Type here ('Enter' to send):
DDT - Multi-Dimensional Array Viewer (on localhost.localdomain)
Array Expression: C[$i][$j]
Evaluate
Cancel
Distributed Array Dimensions: None
How do I view distributed arrays?
Align Stack Frames
Auto-update
Range of $i
From: 0
To: 3
Display: Rows
Range of $j
From: 0
To: 3
Display: Columns
Only show if:
See Examples
Data Table Statistics
Goto Visualize Export Full Window
j
0 1 2 3
i 0 36 42 60 90
1 50 60 90 140
2 64 78 120 190
3 78 96 150 240
Close



i \ j	0	1	2	3
0	36	42	60	90
1	50	60	90	140
2	64	78	120	190
3	78	96	150	240


DDT - Visualization (on localhost.localdomain)
File
View
Viewpoint
220
170
120
70
Process 0
Close

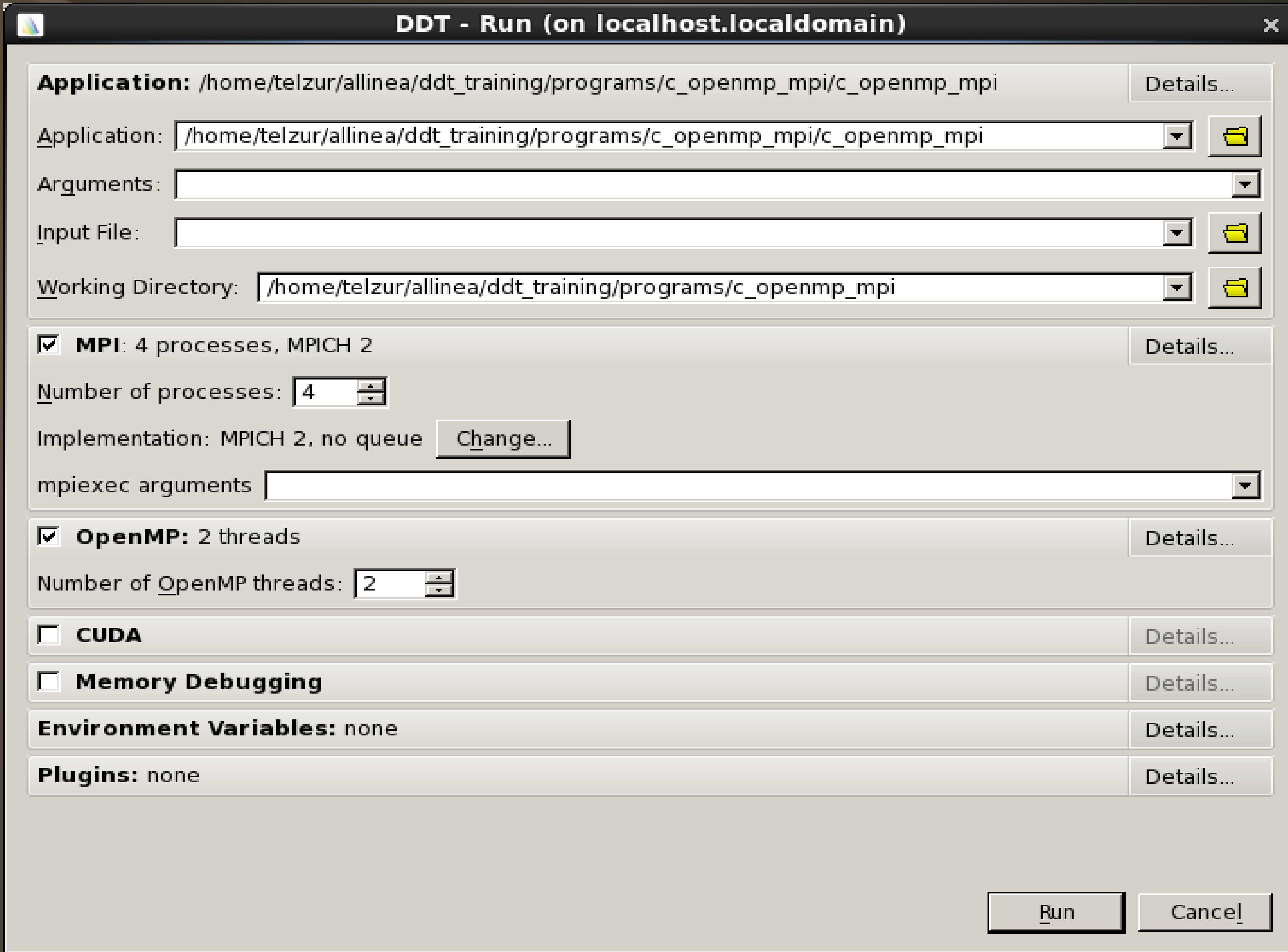
VisIt Integration with DDT

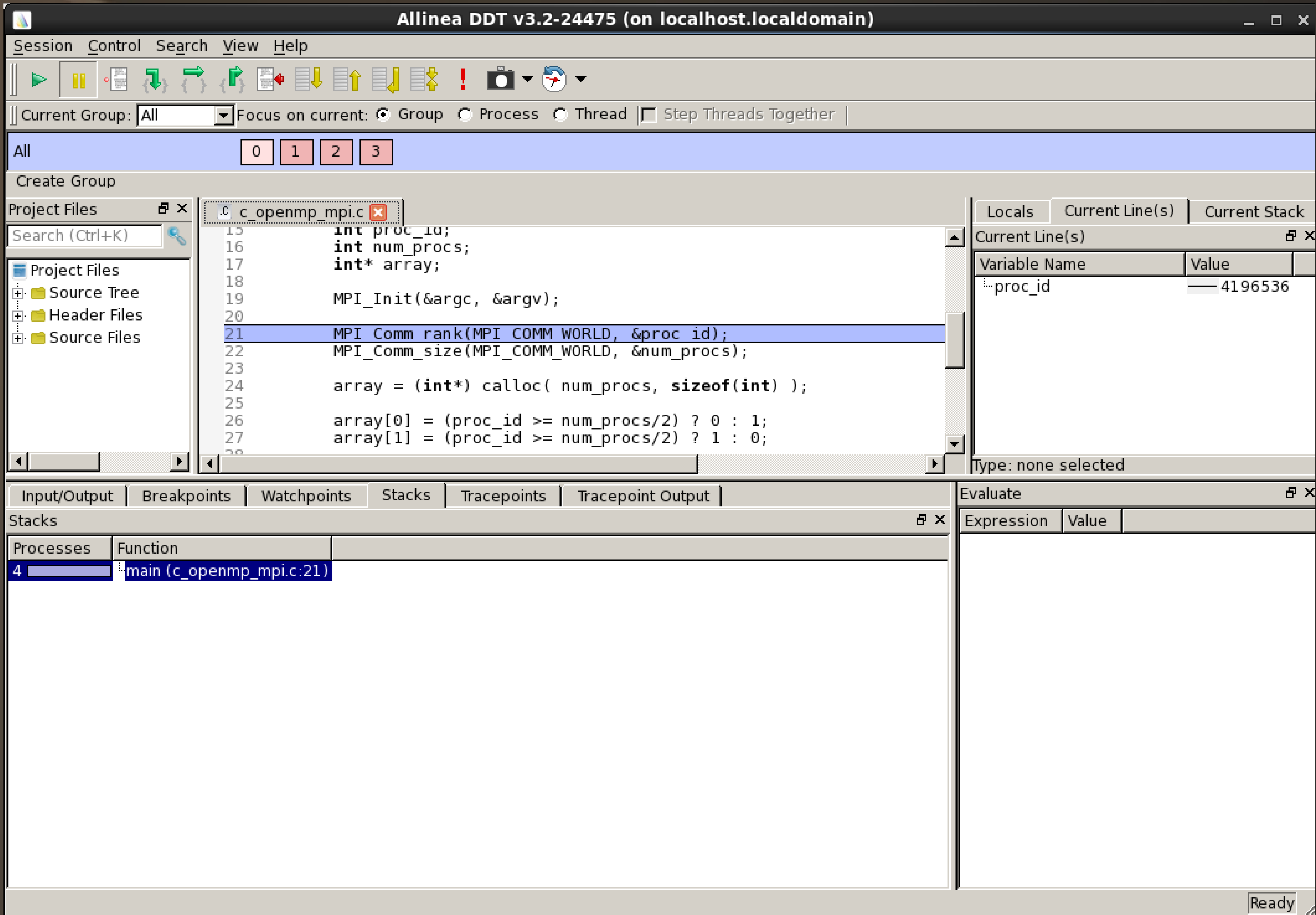

DDT - Options (on localhost.localdomain)
System
Job Submission
Remote Launch
Code Viewer
Appearance
VisIt
VisIt Visualization
VisIt is a free interactive parallel visualization and graphical analysis tool for viewing scientific data. By setting 'Visualization breakpoints' DDT can feed information to VisIt.
DDT can also work with programs you have already modified to display data in VisIt (instrumented using VisIt's runtime library libsim).
Allow the use of VisIt with DDT
VisIt launch command: /home/telzur/visit/visit2_6_0.linux-x86_64/bin/visit
Custom arguments:
Launch VisIt with small viewer (-small argument)
Use Hardware Acceleration (-hw-accel argument)
Enable visualisation breakpoints (preloads ddtsim if its not already statically linked)
VisIt launch command on compute nodes :
OK
Cancel

Example 2: MPI + OpenMP

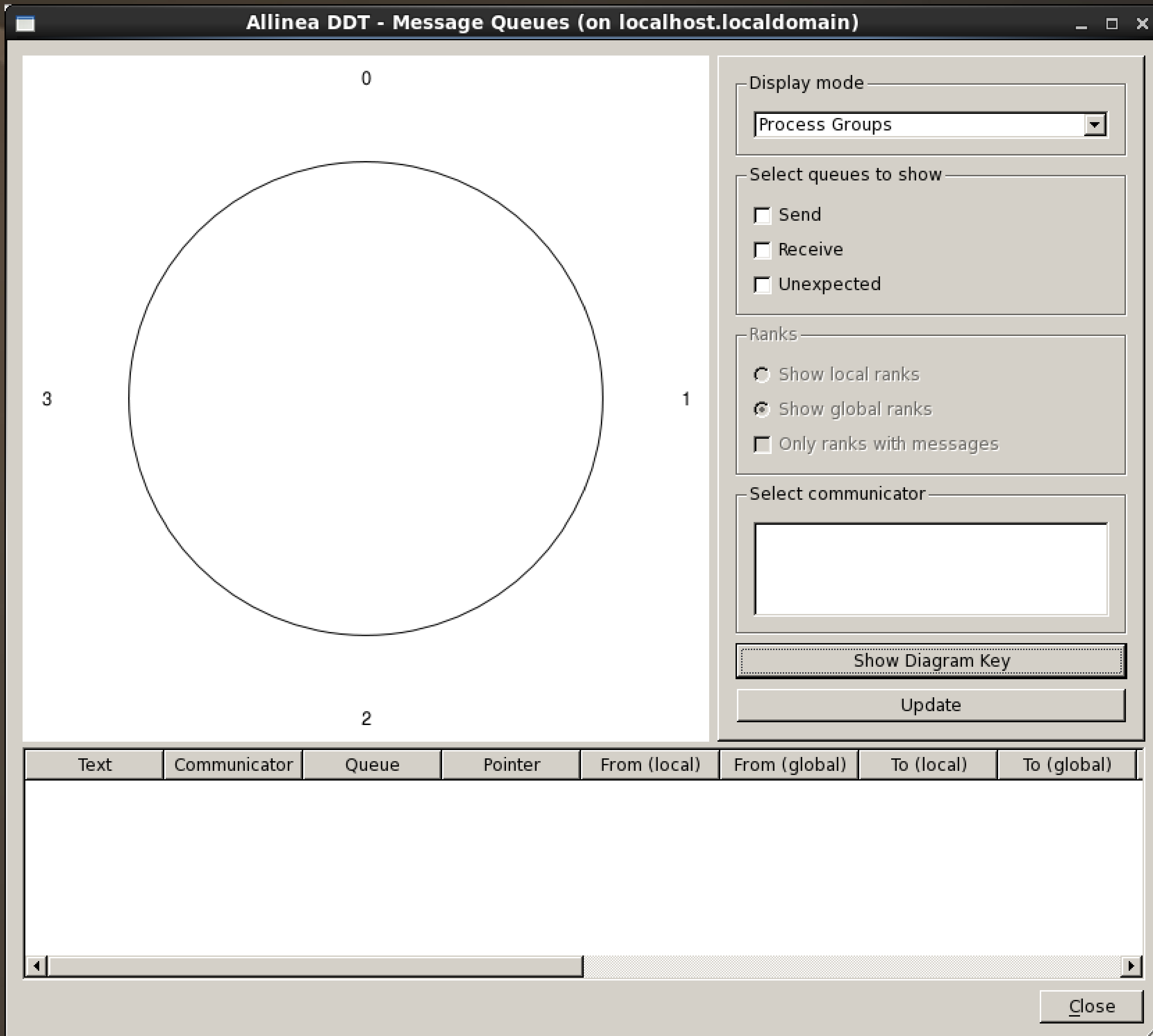
- cd to: c_openmp_mpi
- Correct the Makefile to gcc: -fopenmp
- Built using ¨make¨:

```
[root@localhost c_openmp_mpi]# make
mpicc -c -g -fopenmp -o c_openmp_mpi.o c_openmp_mpi.c
mpicc -g -fopenmp  -o c_openmp_mpi c_openmp_mpi.o
```


DDT - Run (on localhost.localdomain)
Application: /home/telzur/allinea/ddt_training/programs/c_openmp_mpi/c_openmp_mpi
Details...
Application: /home/telzur/allinea/ddt_training/programs/c_openmp_mpi/c_openmp_mpi
Arguments:
Input File:
Working Directory: /home/telzur/allinea/ddt_training/programs/c_openmp_mpi
MPI: 4 processes, MPICH 2
Details...
Number of processes: 4
Implementation: MPICH 2, no queue
Change...
mpiexec arguments
OpenMP: 2 threads
Details...
Number of OpenMP threads: 2
CUDA
Details...
Memory Debugging
Details...
Environment Variables: none
Details...
Plugins: none
Details...
Run
Cancel


Allinea DDT v3.2-24475 (on localhost.localdomain)
Session Control Search View Help
Current Group: All
Focus on current: Group Process Thread
Step Threads Together
All
0 1 2 3
Create Group
Project Files
Search (Ctrl+K)
Project Files
Source Tree
Header Files
Source Files
c_openmp_mpi.c
16 int num_procs;
17 int* array;
18
19 MPI_Init(&argc, &argv);
20
21 MPI_Comm_rank(MPI_COMM_WORLD, &proc_id);
22 MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
23
24 array = (int*) calloc(num_procs, sizeof(int));
25
26 array[0] = (proc_id >= num_procs/2) ? 0 : 1;
27 array[1] = (proc_id >= num_procs/2) ? 1 : 0;
Locals
Current Line(s)
Current Stack
Current Line(s)
Variable Name
Value
proc_id
4196536
Type: none selected
Input/Output
Breakpoints
Watchpoints
Stacks
Tracepoints
Tracepoint Output
Stacks
Processes
Function
4
main (c_openmp_mpi.c:21)
Evaluate
Expression
Value
Ready

- Add Breakpoints. Hoover with the mouse over the variables like ¨thread_id¨


Allinea DDT - Message Queues (on localhost.localdomain)
0
1
2
3
Display mode
Process Groups
Select queues to show
Send
Receive
Unexpected
Ranks
Show local ranks
Show global ranks
Only ranks with messages
Select communicator
Show Diagram Key
Update
Text
Communicator
Queue
Pointer
From (local)
From (global)
To (local)
To (global)
Close

Part 2: CilkPlus

- Reference material: http://cilkplus.org/cilk-plus-tutorial !!!
- Please read this tutorial and understand it.
- open: http://cilkplus.org/download#block-views-code-samples
  and download the tutorial files: cilk-fib.tgz tutorial-src.tgz
- Unzip it and cd to the ¨keywords¨ folder
- CilkPlus Keywords tutorial: http://cilkplus.org/tutorial-cilk-plus-keywords

Fibonacci

- Understand the code ¨fib.cpp¨
- Build the binary:

1. `setenv LD_LIBRARY_PATH {$LD_LIBRARY_PATH}:/opt/gcc-cilkplus-install/lib`

2. `/opt/gcc-cilkplus-install/bin/gcc -fcilkplus -o fib -L /opt/gcc-cilkplus-install/lib -lcilkrts fib.cpp`

Fibonacci

```
[guyte@parallel1 keywords]$ ./fib
Fibonacci number #39 is 63245986.
Calculated in 7.290 seconds using 32 workers.
```

CilkView

```
[guyte@parallel1 keywords]$ cilkview ./fib
cilkview: generating scalability data
Cilkview Scalability Analyzer V2.0.0, Build 2516
Fibonacci number #39 is 63245986.
Calculated in 38.540 seconds using 1 workers.

Whole Program Statistics
1) Parallelism Profile
   Work :                                      33,160,809,023 instructions
   Span :                                      4,551,601 instructions
   Burdened span :                             5,491,601 instructions
   Parallelism :                               7285.53
   Burdened parallelism :                      6038.46
   Number of spawns/syncs:                     102,334,154
   Average instructions / strand :                    108
   Strands along span :                               77
   Average instructions / strand on span :            59,111
   Total number of atomic instructions :              102,334,158
   Frame count :                               307,002,462

2) Speedup Estimate
     2 processors:   1.90 - 2.00
     4 processors:   3.80 - 4.00
     8 processors:   7.60 - 8.00
    16 processors:   15.20 - 16.00
    32 processors:   30.40 - 32.00
    64 processors:   60.80 - 64.00
   128 processors:   121.60 - 128.00
   256 processors:   238.85 - 256.00
```

CilkView

```
Cilk Parallel Region(s) Statistics - Elapsed time: 38.389 seconds
1) Parallelism Profile
   Work :                                      33,156,266,667 instructions
   Span :                                      9,245 instructions
   Burdened span :                             949,245 instructions
   Parallelism :                               3586399.86
   Burdened parallelism :                      34929.09
   Number of spawns/syncs:                     102,334,154
   Average instructions / strand :                    108
   Strands along span :                               38
   Average instructions / strand on span :            243
   Total number of atomic instructions :              102,334,158
   Frame count :                               307,002,462
   Entries to parallel region :                       1

2) Speedup Estimate
     2 processors:   1.90 - 2.00
     4 processors:   3.80 - 4.00
     8 processors:   7.60 - 8.00
    16 processors:   15.20 - 16.00
    32 processors:   30.40 - 32.00
    64 processors:   60.80 - 64.00
   128 processors:   121.60 - 128.00
   256 processors:   243.20 - 256.00
```

CilkScreen

- For detection of Race conditions

```
[guyte@parallel1 keywords]$ cilkscreen ./fib
Cilkscreen Race Detector V2.0.0, Build 2516
Fibonacci number #39 is 63245986.
Calculated in 482.280 seconds using 1 workers.
No errors found by Cilkscreen
```

Reducers

- Tutorial: http://cilkplus.org/tutorial-cilk-plus-reducers
- cd to reducers
- Build:

```
[guyte@parallel1 reducers]$ /opt/gcc-cilkplus-install/bin/gcc -fcilkplus
-o reducers -L /opt/gcc-cilkplus-install/lib -lcilkrts cilk-reducers-
demo.cpp

cilk-reducers-demo.cpp: In constructor ‘mutex::mutex()’:

cilk-reducers-demo.cpp:84:14: warning: extended initializer lists only
available with -std=c++11 or -std=gnu++11 [enabled by default]

       m_mutex(PTHREAD_MUTEX_INITIALIZER)

cilk-reducers-demo.cpp:84:40: warning: extended initializer lists only
available with -std=c++11 or -std=gnu++11 [enabled by default]

       m_mutex(PTHREAD_MUTEX_INITIALIZER)
```

Reducers

```
[guyte@parallel1 reducers]$ ./reducers
Letters from locked list:  a n g d b j w e h l y q x z i r o k f s p m v t c u
Letters from reducer_list: a b c d e f g h i j k l m n o p q r s t u v w x y z
```

Array Notation

- Tutorial: http://cilkplus.org/tutorial-array-notation
- Code examples: ...cilk/tutorial-src/array-notations

Add Arrays

```
/*
 * add-arrays.cpp
 *Demonstrate two arrays to produce a third using both for loop and
 * Array Notation
 */
#include <stdio.h>
#include <string.h>
const int array_size=10;
int main(int argc, char **argv)
{
    int a[array_size], b[array_size], c[array_size], d[array_size];

    // Initialize arrays a & b using for loop
    for (int i = 0; i < array_size; i++)
    {
        a[i] = i;
        b[i] = i + array_size;
    }
    // Add arrays using for loop
    for (int i = 0; i < array_size; i++)
        c[i] = a[i] + b[i];
    // Add the arrays using Array Notation.  Since the array is statically allocated, we can use default
    //values for the start index (0) and number of elements (all of them).
    d[:] = a[:] + b[:];
    // Verify the results - The arrays should be identical
    if (0 == memcmp(c, d, sizeof(c)))
        printf("Success\n");
    else
        printf("Failed\n");
return 0;
}
```

- Compile and execute:

```
[guyte@parallel1 array-notations]$ /opt/gcc-cilkplus-install/bin/gcc -fcilkplus -o add-arrays -L /opt/gcc-cilkplus-install/lib -lcilkrts add-arrays.cpp

[guyte@parallel1 array-notations]$ ./add-arrays

Success
```

Part3: UPC

- Reference material for this part in at the course web site, see lecture #12.

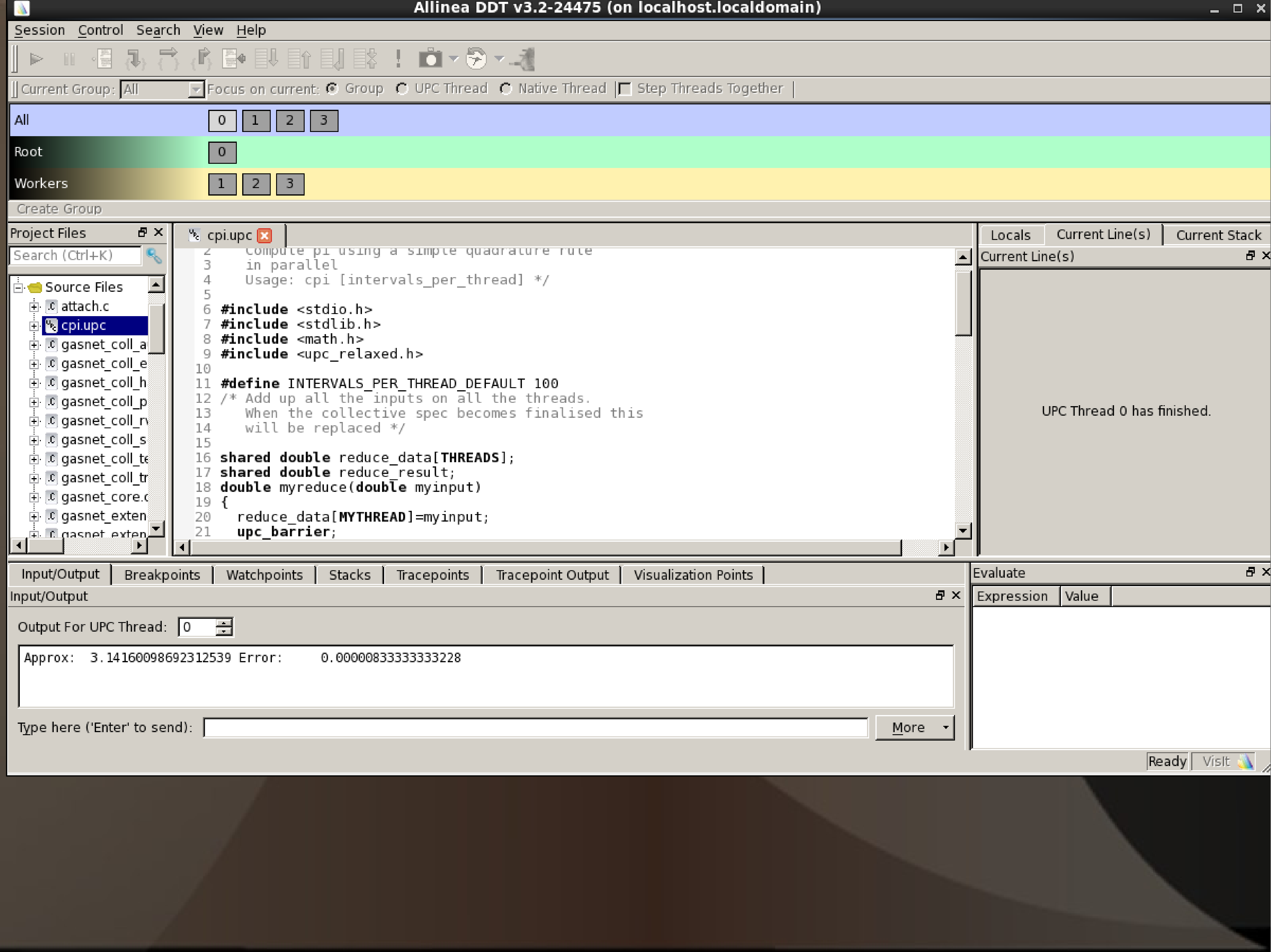
Part 3: UPC

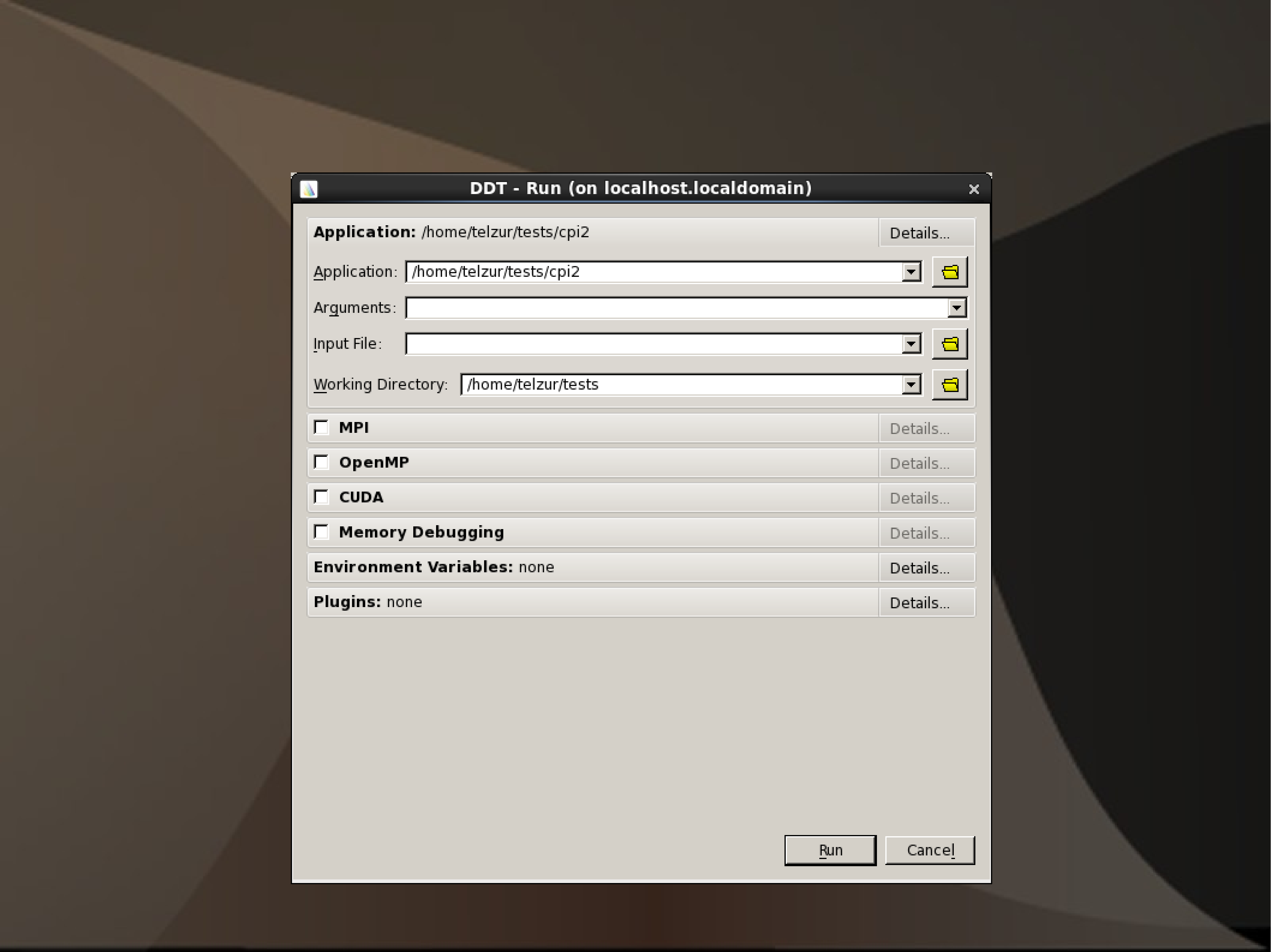
```
[root@localhost tests]# upcrun -n 4 ./cpiupc

UPCR: UPC thread 2 of 4 on localhost.localdomain (pshm node 0 of
1, process 2 of 4, pid=10544)

UPCR: UPC thread 1 of 4 on localhost.localdomain (pshm node 0 of
1, process 1 of 4, pid=10543)

UPCR: UPC thread 3 of 4 on localhost.localdomain (pshm node 0 of
1, process 3 of 4, pid=10545)

UPCR: UPC thread 0 of 4 on localhost.localdomain (pshm node 0 of
1, process 0 of 4, pid=10518)

Approx:  3.14159317442312647 Error:     0.00000052083333335
```

UPC in Allineaś DDT

- Compilation: `upcc -tv -o cpi cpi.upc`
-
- -tv means totalview debugging symbols (also what ever is taken by -g)
- -tv is also a must for DDT
- After configuring DDT to support UPC you can invoke your progrsm by:

```
ddt -start cpi
```


Allinea DDT v3.2-24475 (on localhost.localdomain)
Session Control Search View Help
Current Group: All
Focus on current: Group UPC Thread Native Thread Step Threads Together
All 0 1 2 3
Root 0
Workers 1 2 3
Create Group
Project Files
Search (Ctrl+K)
Source Files
attach.c
cpi.upc
gasnet_coll_a
gasnet_coll_e
gasnet_coll_h
gasnet_coll_p
gasnet_coll_r
gasnet_coll_s
gasnet_coll_te
gasnet_coll_tr
gasnet_core.c
gasnet_exten
cpi.upc
Compute pi using a simple quadrature rule
in parallel
Usage: cpi [intervals_per_thread] */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <upc_relaxed.h>
#define INTERVALS_PER_THREAD_DEFAULT 100
/* Add up all the inputs on all the threads.
When the collective spec becomes finalised this
will be replaced */
shared double reduce_data[THREADS];
shared double reduce_result;
double myreduce(double myinput)
{
reduce_data[MYTHREAD]=myinput;
upc_barrier;
Locals Current Line(s) Current Stack
Current Line(s)
UPC Thread 0 has finished.
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Visualization Points
Input/Output
Output For UPC Thread: 0
Approx: 3.14160098692312539 Error: 0.00000833333333228
Type here ('Enter' to send):
More
Evaluate
Expression Value
Ready Visit


DDT - Run (on localhost.localdomain)
Application: /home/telzur/tests/cpi2
Details...
Application: /home/telzur/tests/cpi2
Arguments:
Input File:
Working Directory: /home/telzur/tests
MPI
Details...
OpenMP
Details...
CUDA
Details...
Memory Debugging
Details...
Environment Variables: none
Details...
Plugins: none
Details...
Run
Cancel