Parallel Processing

Guy Tel-Zur

Last update: ~~14/7/2015~~ ~~9/5/2016~~ 19/12/2016

Agenda

Parallel programming in OpenMP - slides8

Hands-On Introduction to OpenMP, Mattson and Meadows, from SC08

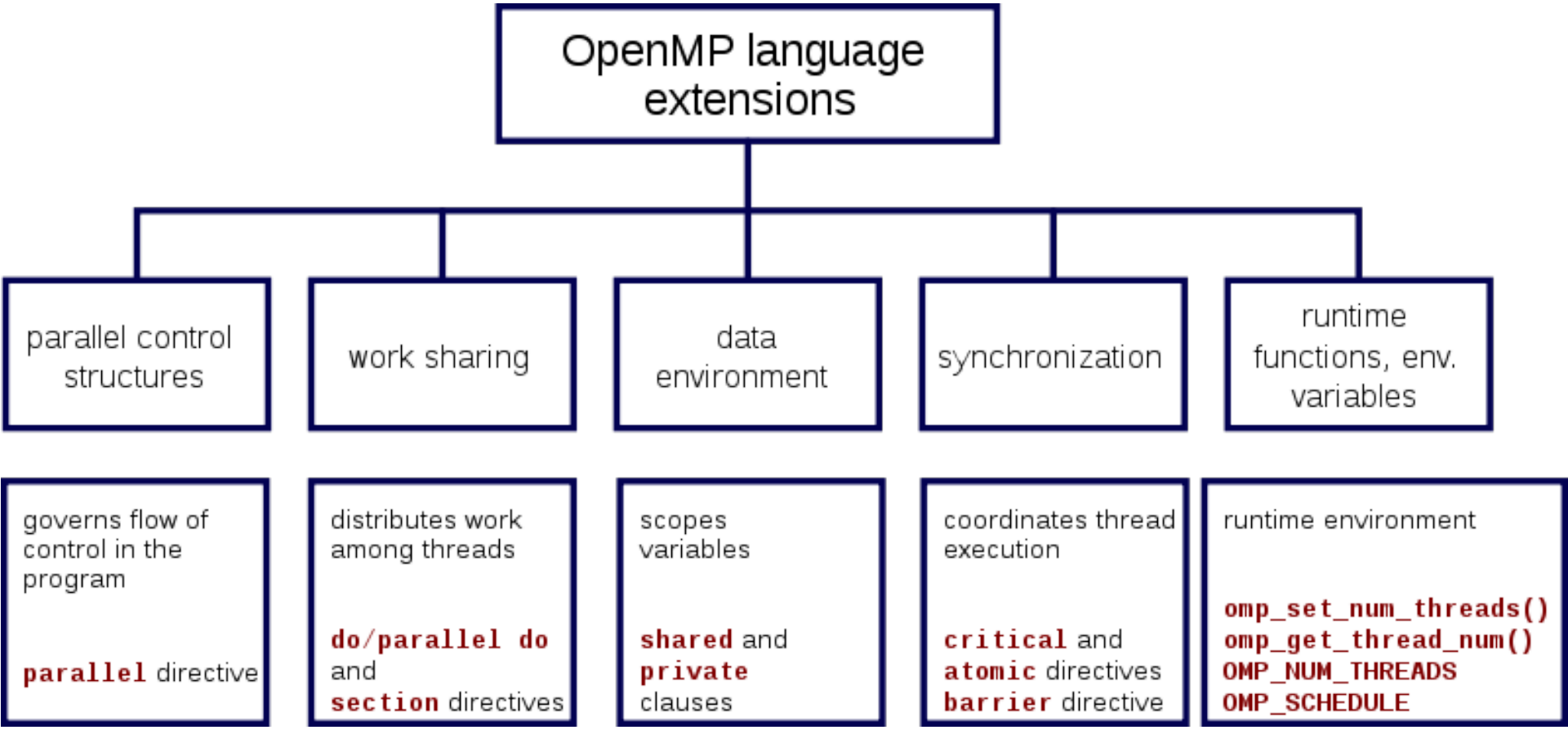
Hands-On Introduction to OpenMP, Mattson and Meadows, from SC08

Slides to skip:

2-3, 58-61, memory model: 70-80, OpenMP 3: 81-113, 128-142, 145-end

OpenMP

from Wikipedia



מומלץ להסתכל בסימוכין הנוספים באתר הויקיפדיה!

More OpenMP references

OpenMP in Visual C++
http://msdn.microsoft.com/en-us/library/tt15eb9t(VS.80).aspx

Quick Reference Card:
http://openmp.org/mp-documents/OpenMP3.1-CCard.pdf

http://www.plutospin.com/files/OpenMP_reference.pdf

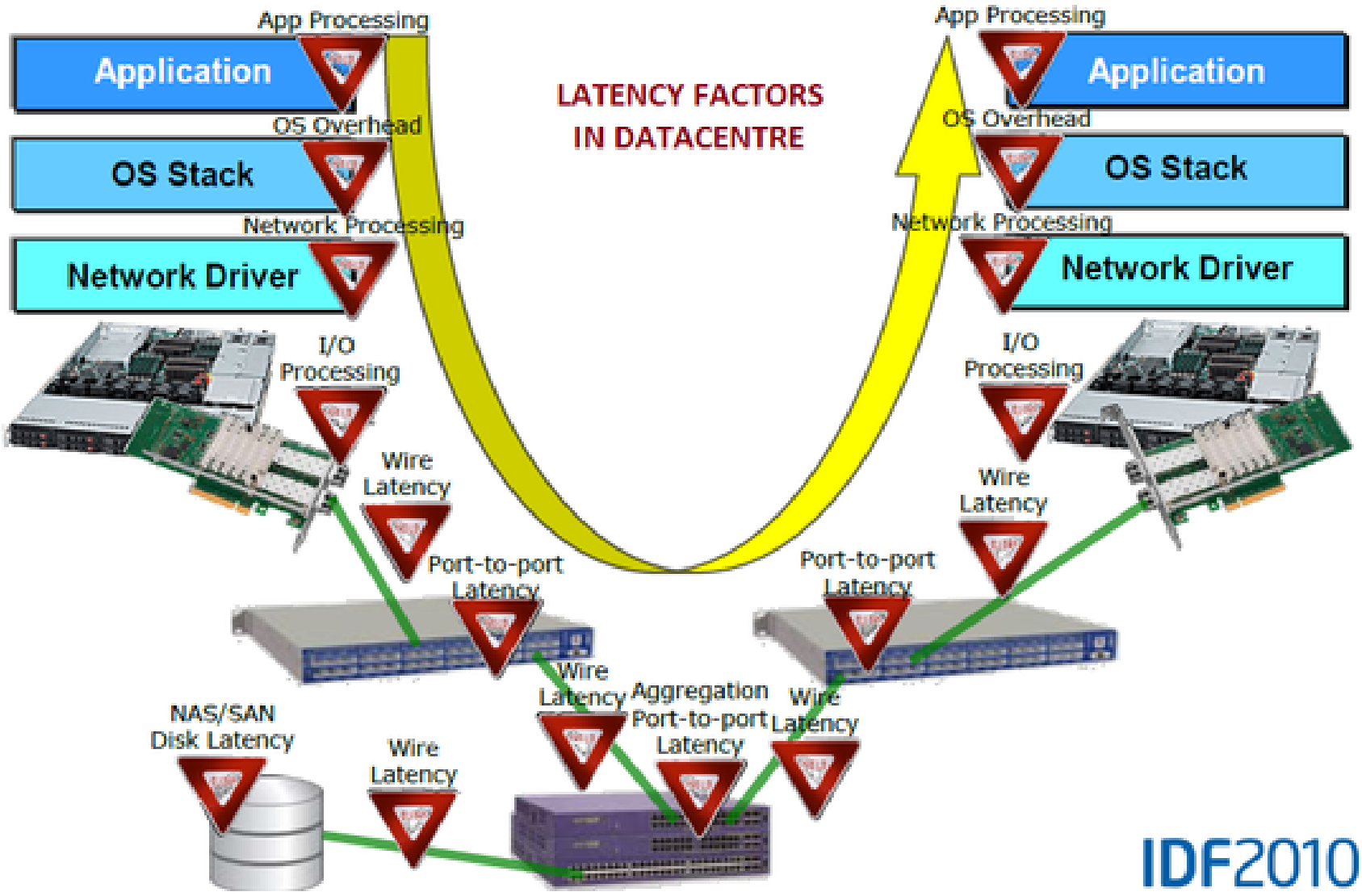
Location of Linear Algebra libraries in the hobbit cluster

```
/usr/lib64/libblas.a
/usr/lib64/atlas/libcblas.a
/usr/lib64/atlas/libcblas.so
/usr/lib64/atlas/libcblas.so.3
/usr/lib64/atlas/libcblas.so.3.0
/usr/include/cblas.h
```

hobbit2, 6-10:

```
/usr/local/lib/libscalapack.a
```

Latency Factors in Data Center



Reference:

Sending a Matrix in MPI

Demo1: reading a matrix from a file by the master process and then sending its rows to the workers
program location:
/home/telzur/Documents/Teaching/BGU/PP/PPxxx/lectures/08/code

Show: `guy1.c` which uses `temp.dat`

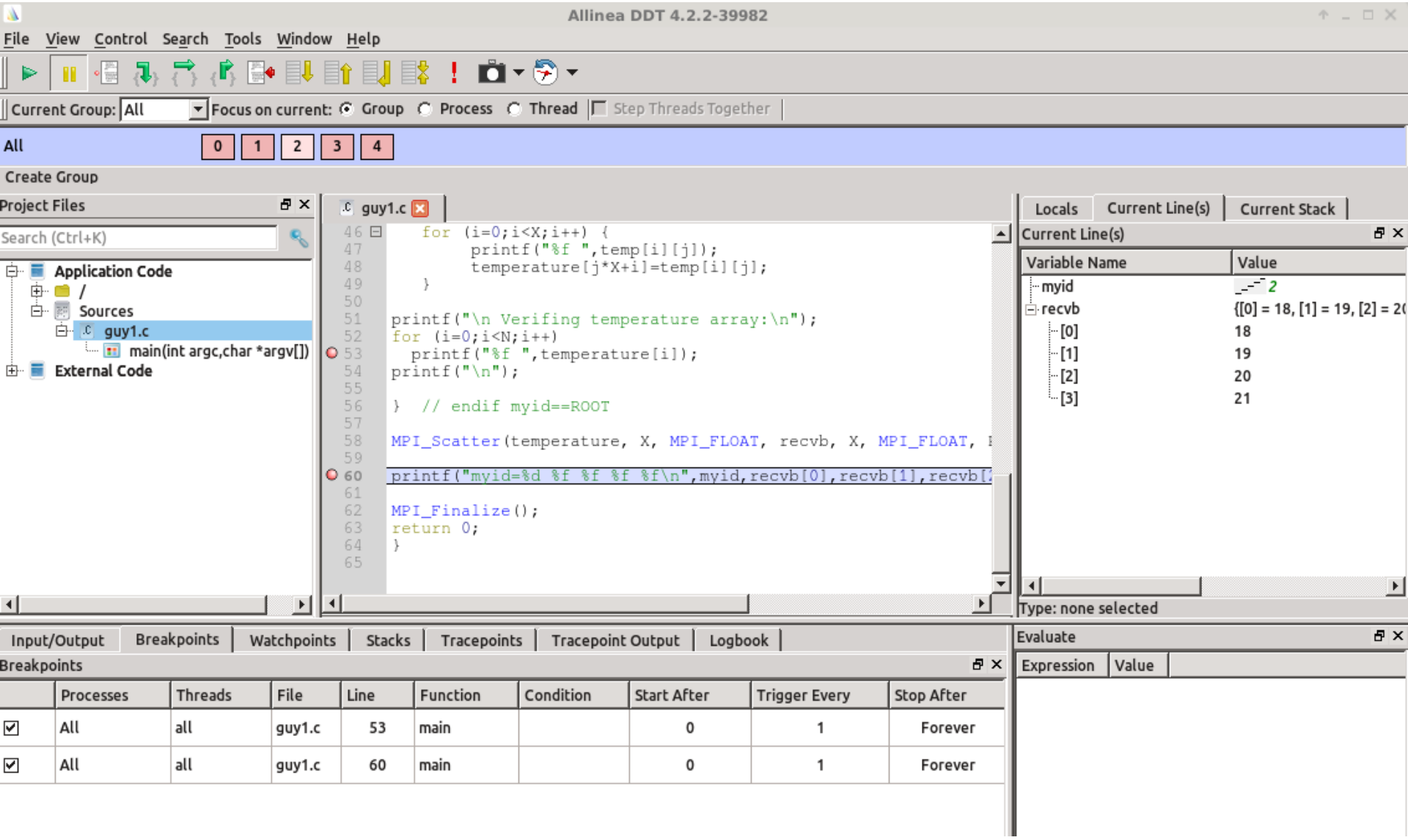
Demo 2: Matrix size limit (static vs. dynamic memory allocation)
Programs location:
/home/telzur/Documents/Teaching/BGU/PP/PPxxx/lectures/08/code
Show: `m_size.c` and `m_size2.c`

Demo 1

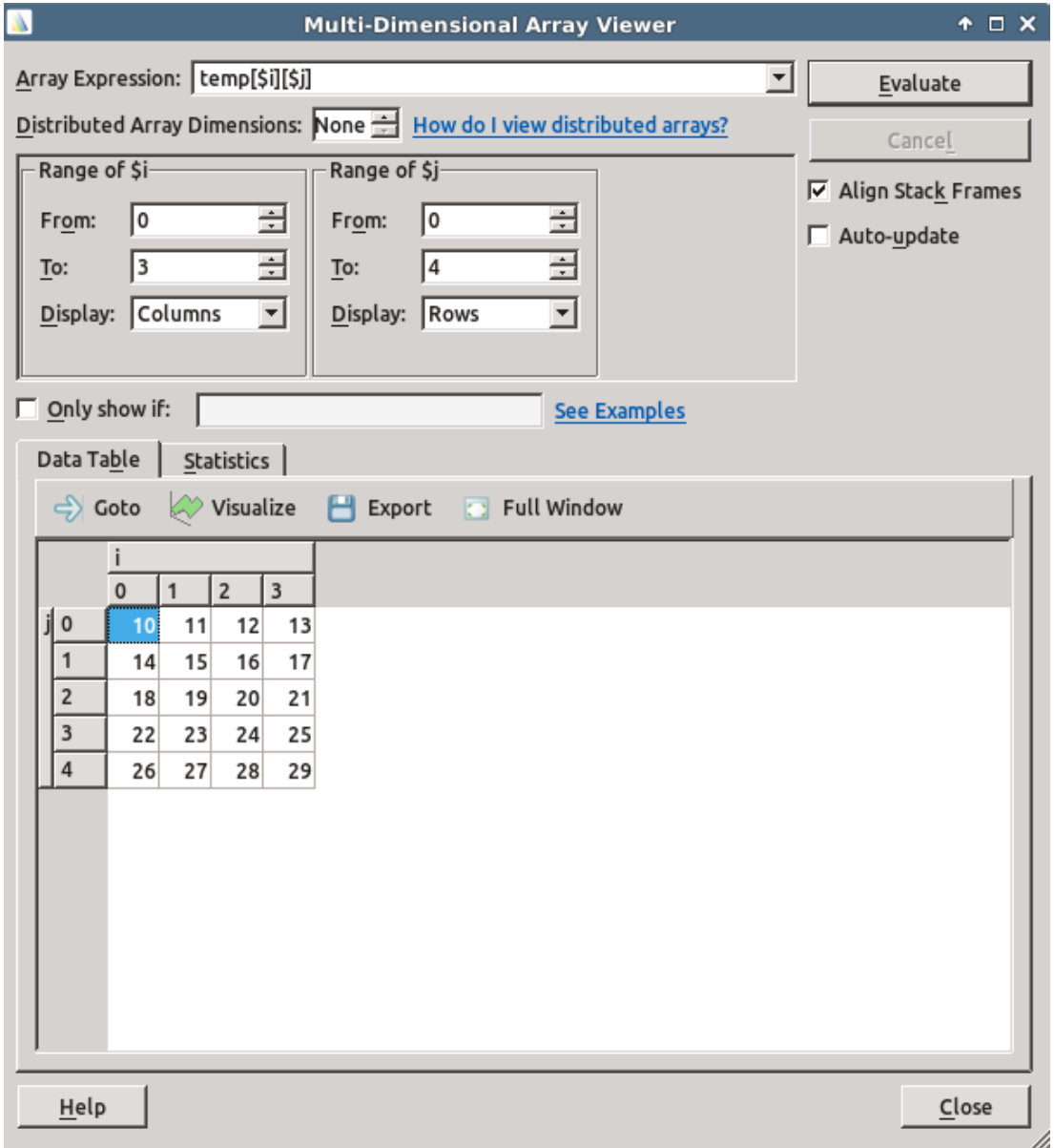
```
// Demo of reading a matrix text file into master node, then the master scatters the rows to
// the tasks
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

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

#define MAXLINE 1024
int X=4,Y=4,N;
int i,j;
int ROOT=0;  // master task
char line[MAXLINE];
float temp[X][Y];
float *temperature;
float recvb[X];
int myid, numprocs;

N = X * Y;

temperature=(float *)malloc(sizeof(float)*X*Y);

MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);

if (numprocs != Y) {
   printf("This demo works with  exactly 4 tasks. Exiting\n");
   MPI_Abort;
   exit(1);
}
```

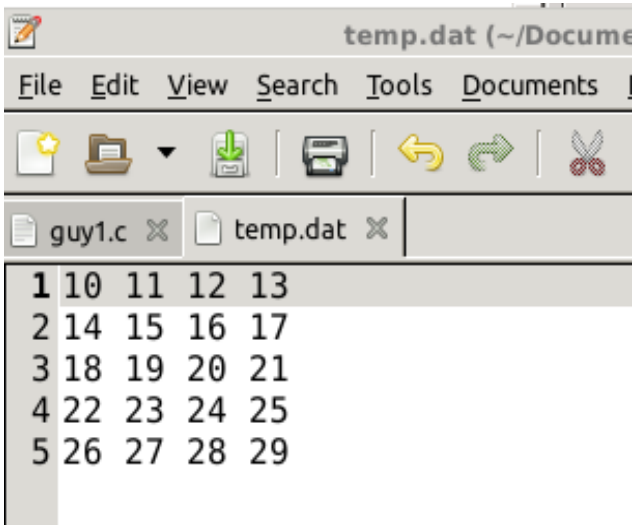
```
if (myid == ROOT) {
  FILE *fp1;
  fp1=fopen("temp.dat","r");
  for (j=0;j<Y;j++)
      {
        fgets(line,MAXLINE,fp1);
        sscanf(line,"%f %f %f %f",&temp[0][j],&temp[1][j],&temp[2][j],&temp[3][j]);
      }
  fclose(fp1);

  for (j=0;j<Y;j++)
   for (i=0;i<X;i++) {
        printf("%f ",temp[i][j]);
        temperature[j*X+i]=temp[i][j];
   }

printf("\n Verifing temperature array:\n");
for (i=0;i<N;i++)
  printf("%f ",temperature[i]);
printf("\n");

}  // endif myid==ROOT

MPI_Scatter(temperature, X, MPI_FLOAT, recvb, X, MPI_FLOAT, ROOT, MPI_COMM_WORLD);

printf("myid=%d %f %f %f %f\n",myid,recvb[0],recvb[1],recvb[2],recvb[3]);

MPI_Finalize();
return 0;
}
```

A demo using Allinea's DDT

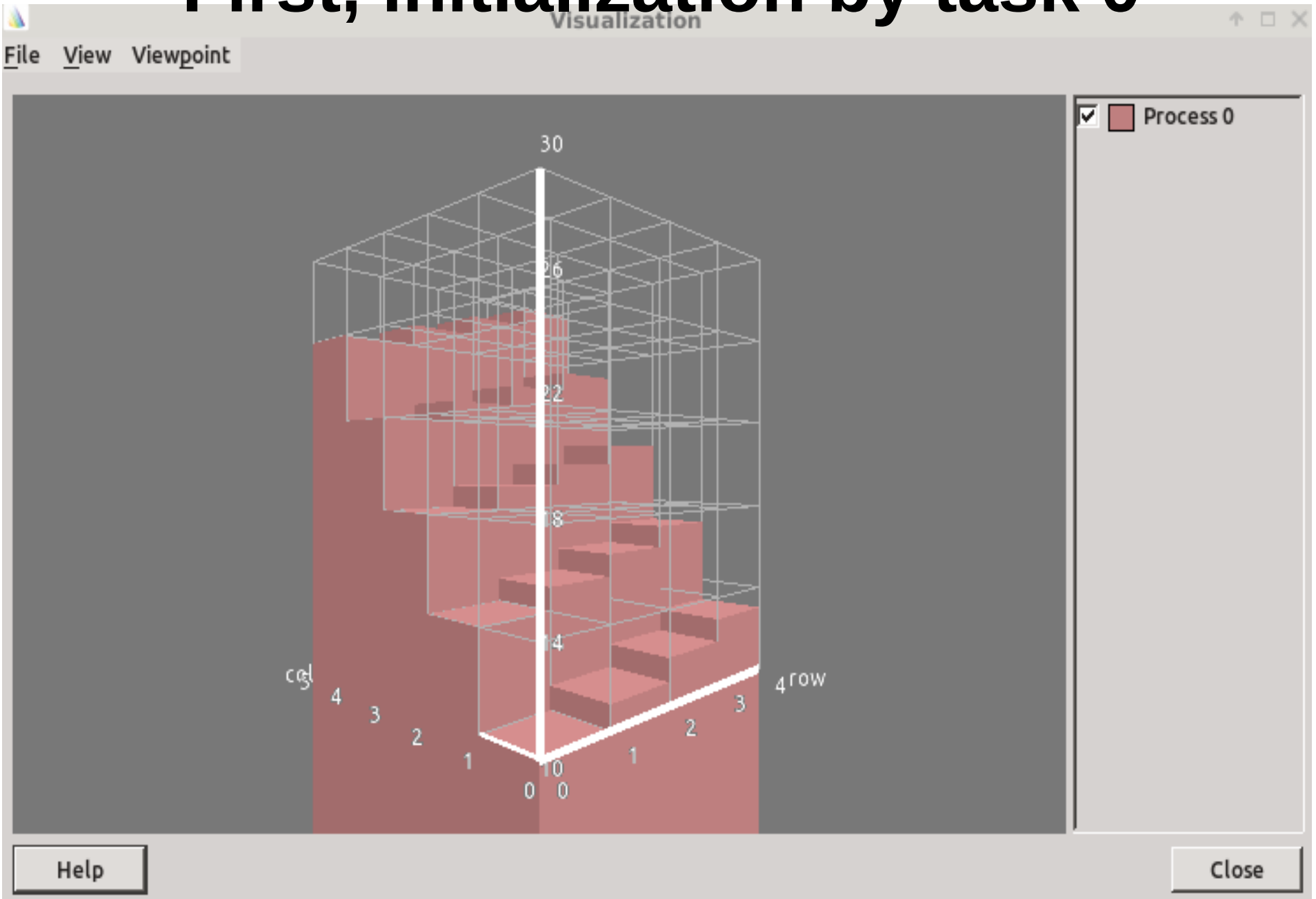


Allinea DDT

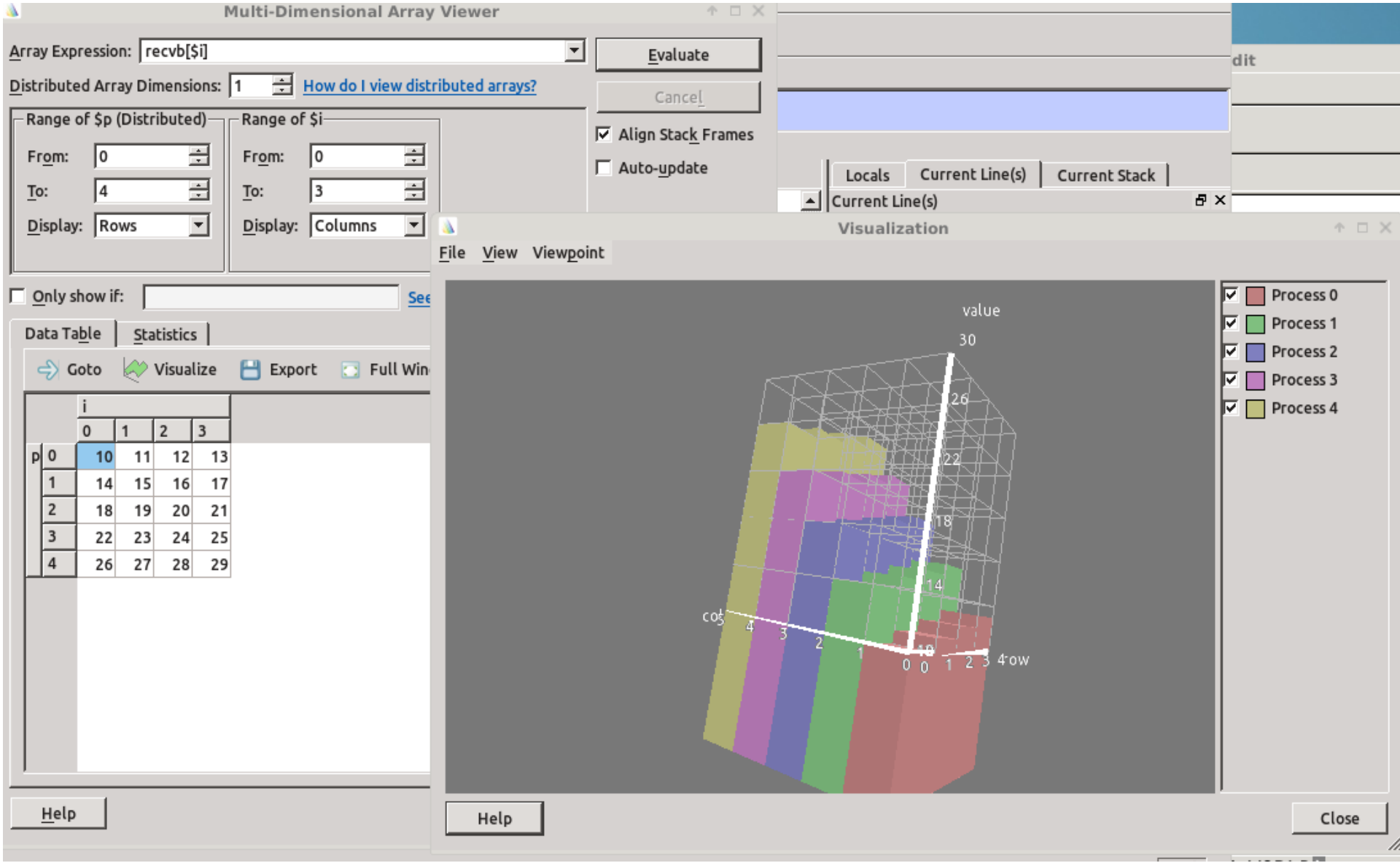

Multi-Dimensional Array Viewer
Array Expression: temp[$i][$j]
Evaluate
Distributed Array Dimensions: None
How do I view distributed arrays?
Cancel
Range of $i
From: 0
To: 3
Display: Columns
Range of $j
From: 0
To: 4
Display: Rows
Align Stack Frames
Auto-update
Only show if:
See Examples
Data Table
Statistics
Goto
Visualize
Export
Full Window
i
0 1 2 3
j
0 10 11 12 13
1 14 15 16 17
2 18 19 20 21
3 22 23 24 25
4 26 27 28 29
Help
Close


temp.dat (~/Docume
File Edit View Search Tools Documents
guy1.c
temp.dat
1 10 11 12 13
2 14 15 16 17
3 18 19 20 21
4 22 23 24 25
5 26 27 28 29

First, initialization by task 0



Then, rows are scattered to tasks



Demo 2

```
// m_size.c, this code is to clarify limitation
of Matrix size


#define SIZE 800 // on my laptop 800 is still ok
but it crushes on
  //  SIZE >> 800

int main() {
 int i,j;
 float M[SIZE][SIZE];
    for (i=0;i<SIZE;i++)
        for (j=0;j<SIZE;j++)
            M[i][j]=i;
        return 0;
}
```

```
// m_size2.c
#include<stdlib.h>
#define ROW 80000
#define COL 9000

int main() {
    int i,j;
    float** M;
    // Create 2D array of pointers:
    M= (float**) malloc(ROW*sizeof(float*));
    for (i = 0; i < ROW; i++)
        M[i] = (float*) malloc(COL*sizeof(float*));

    // Computation...
    for (i = 0; i < ROW; ++i)
        for (j = 0; j < COL; ++j)
            M[i][j] = i*j;

    // Free allocated memory
    for (i = 0; i < ROW; i++)
        free(M[i]);
    free(M);
    return 0;
}
```