Parallel Processing

Guy Tel-Zur

Last update: ~~14/7/2015~~ ~~9/5/2016~~ ~~19/12/2016~~ ~~3/12/2018~~, ~~7/12/2020~~, 12/12/2022

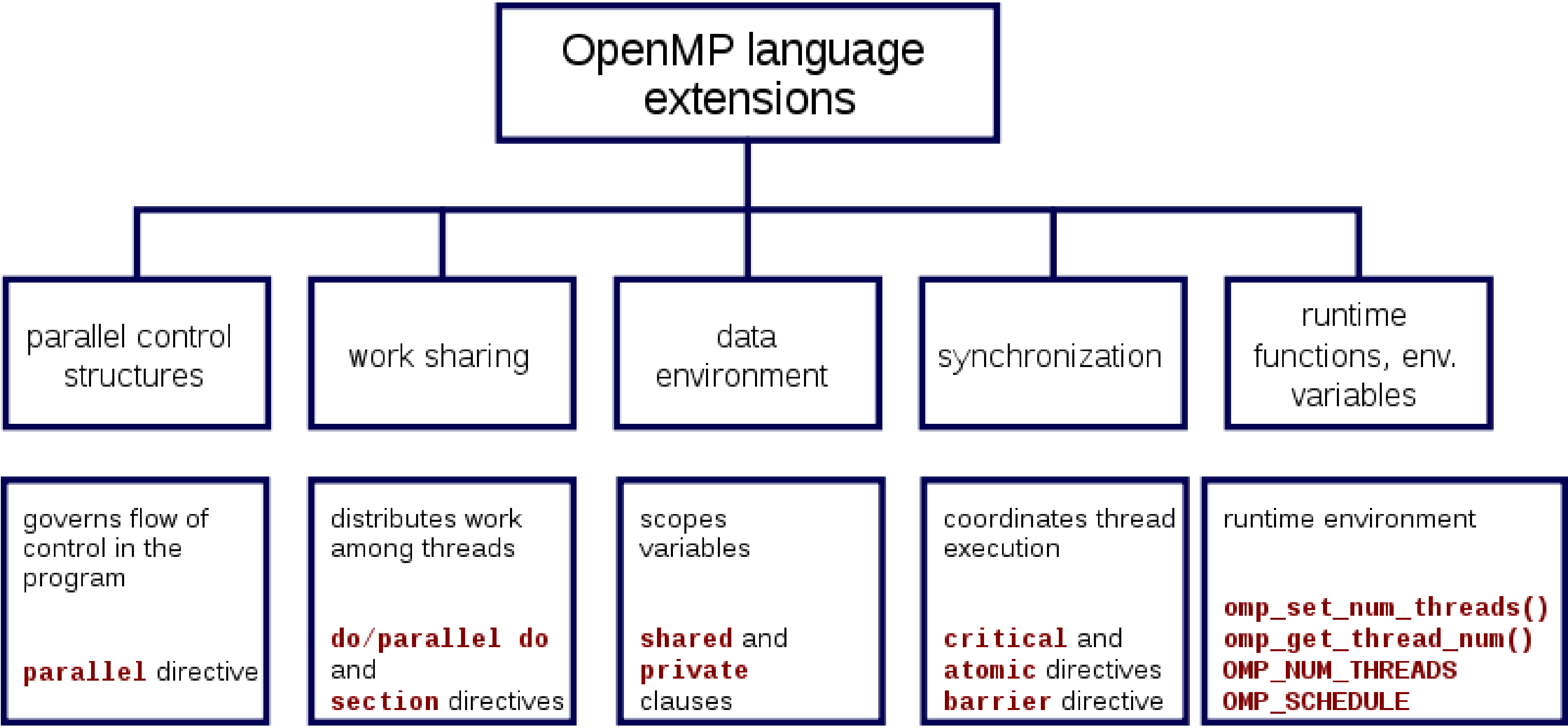
Agenda

Parallel programming in OpenMP
- slides8

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

OpenMP

from Wikipedia



OpenMP language extensions

parallel control structures

governs flow of control in the program

parallel directive

work sharing

distributes work among threads

do/parallel do and **section** directives

data environment

scopes variables

shared and **private** clauses

synchronization

coordinates thread execution

critical and **atomic** directives **barrier** directive

runtime functions, env. variables

runtime environment

omp_set_num_threads()
omp_get_thread_num()
OMP_NUM_THREADS
OMP_SCHEDULE

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

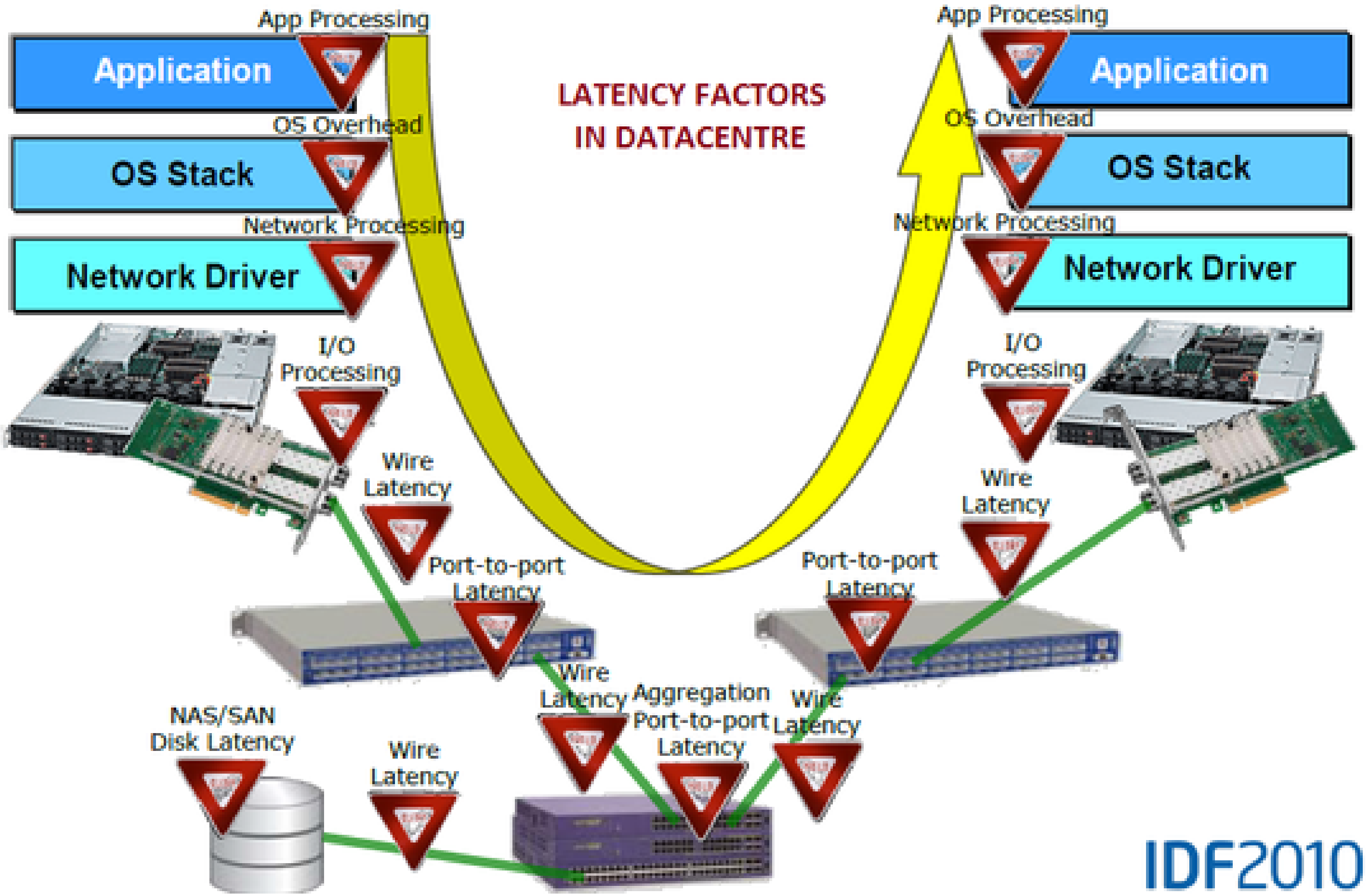
http://www.plutospin.com/files/OpenMP_reference.pdf

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: Intel's 10 Gigabit Ethernet boost pushes out Infiniband

Two demos

Dealing with Matrices in MPI

Demo 1: 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/lectures/08/code/Matrix1

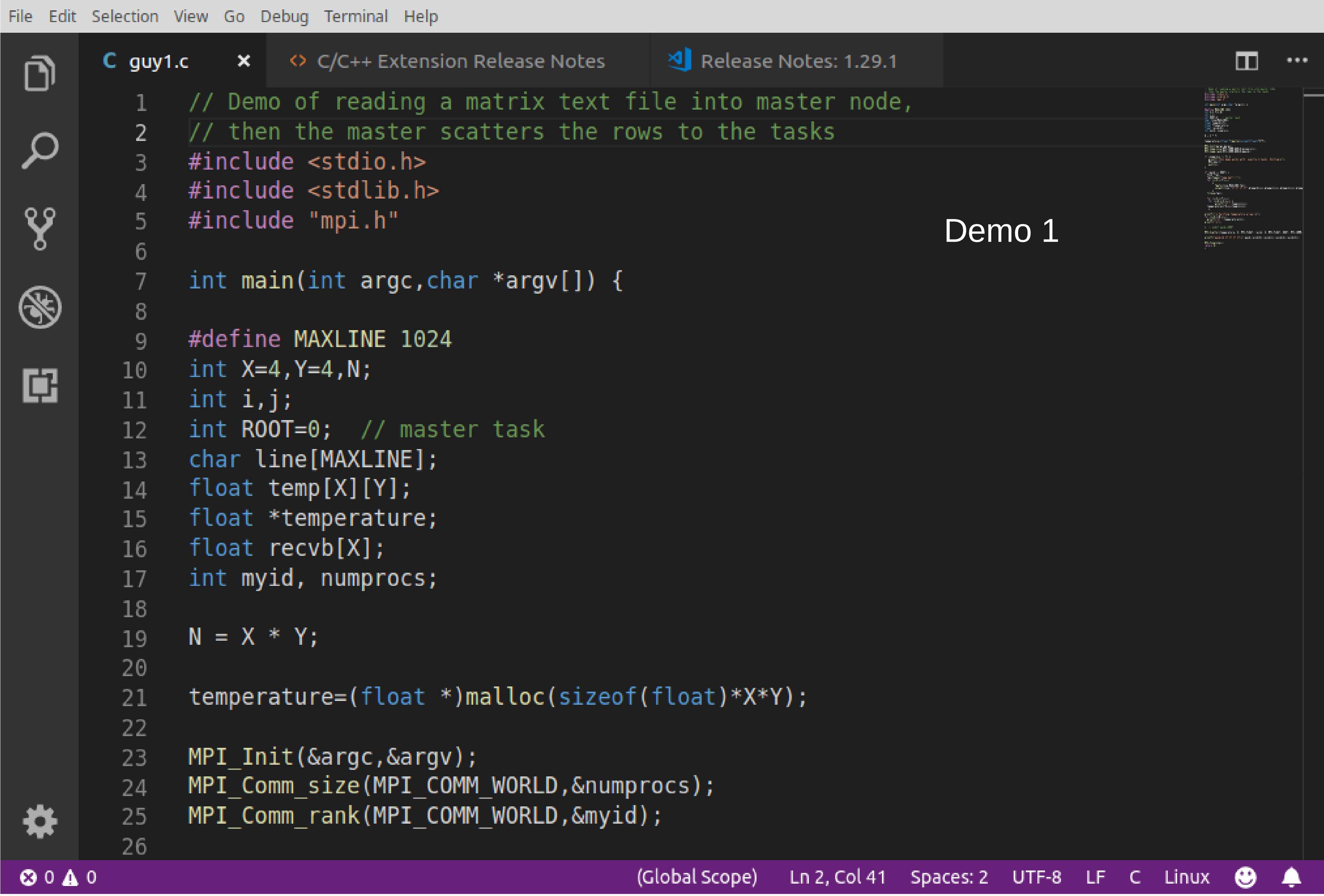
Show: `guy1.c` which uses `temp.dat`

Demo 2: Matrix size limit (static vs. dynamic memory allocation)

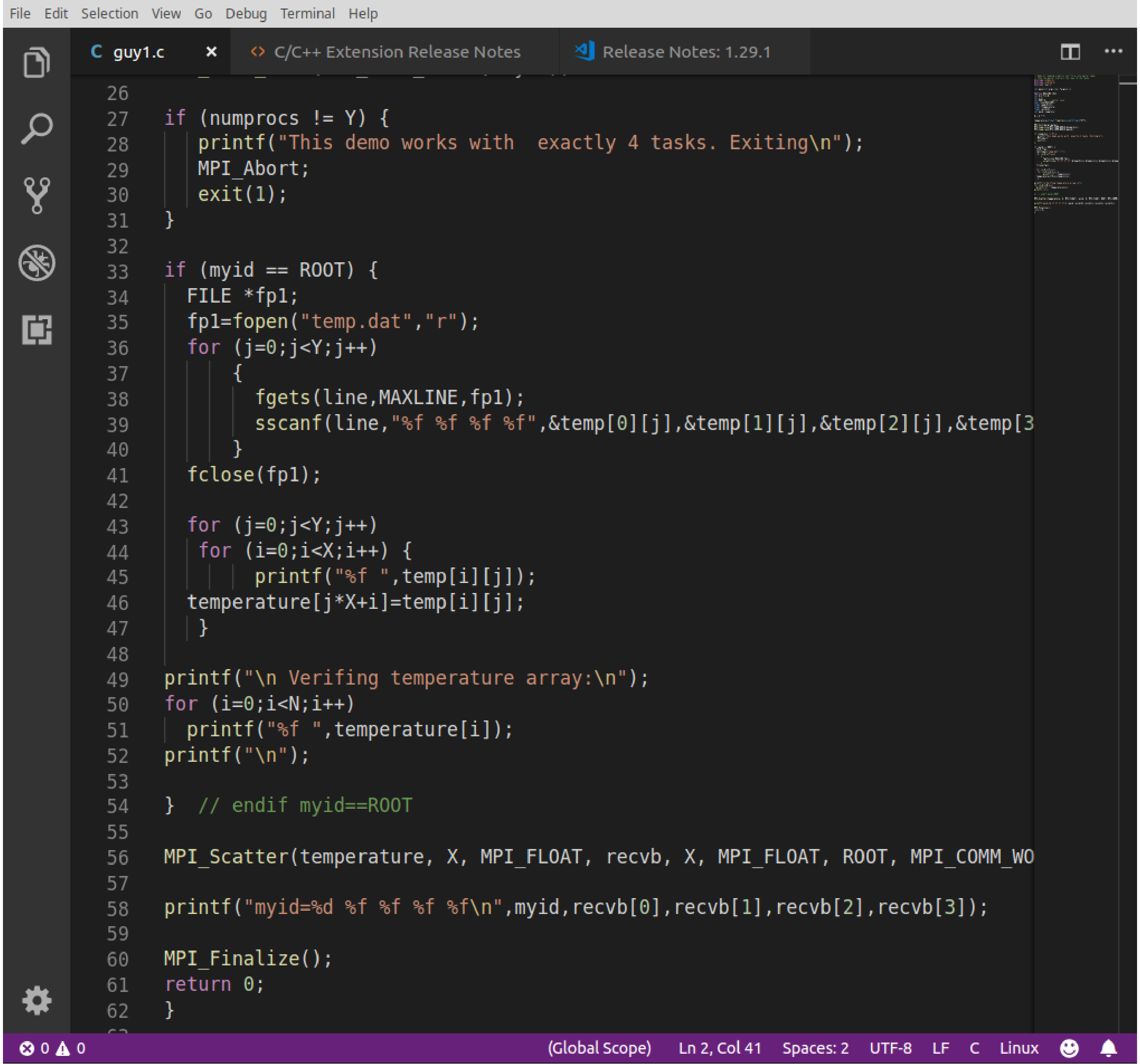
Programs location: /home/telzur/Documents/Teaching/BGU/PP/lectures/08/code/Matrix2

Show: Static allocation: `m_size.c` and Dynamic allocation: `m_size2.c`

Demo 1

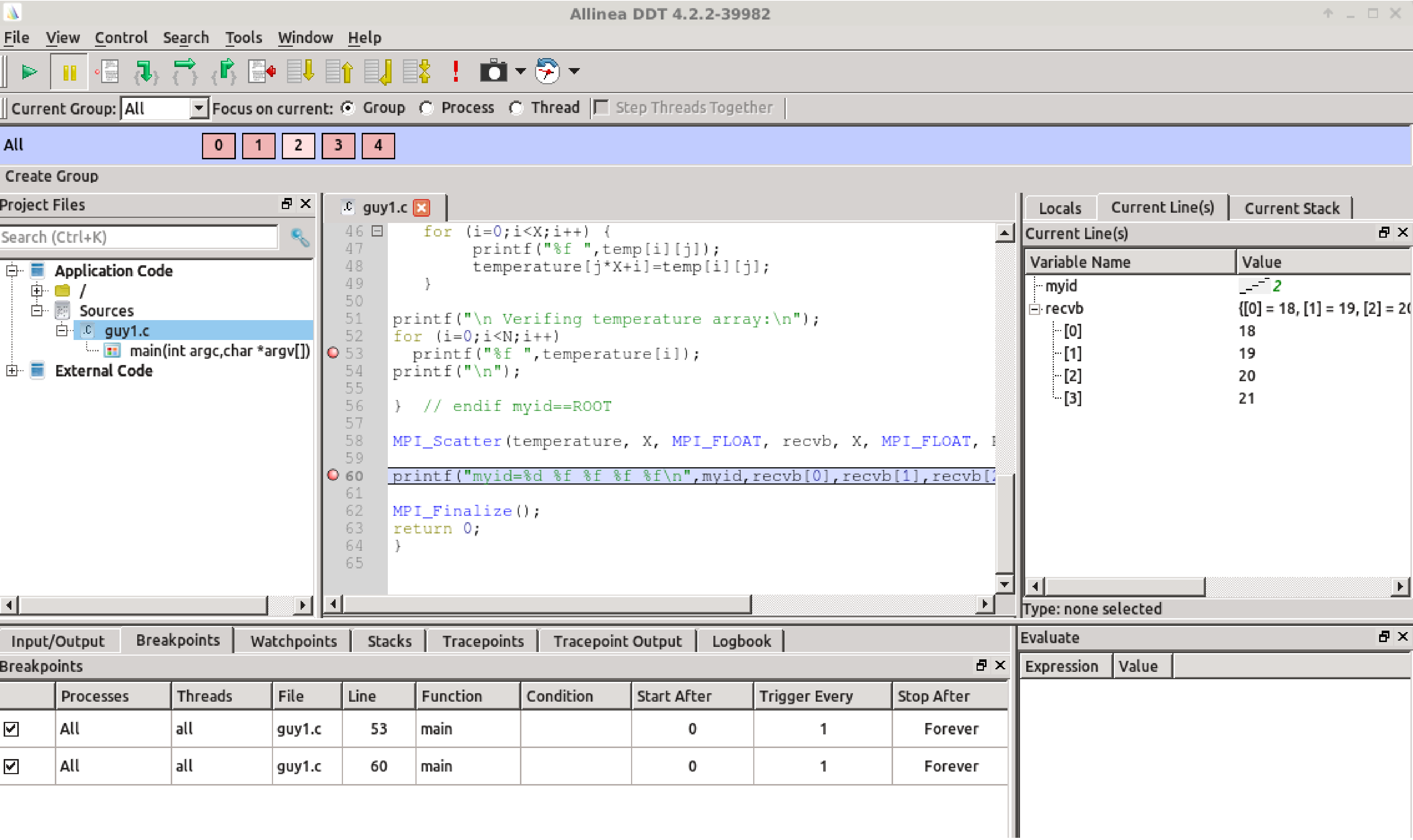


```c
// Demo of reading a matrix text file into master node,
// then the master scatters the rows to the tasks
#include <stdio.h>
#include <stdlib.h>
#include "mpi.h"

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
      }
  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_WO

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 4.2.2-39982
File View Control Search Tools Window Help
Current Group: All
Focus on current: ● Group ○ Process ○ Thread □ Step Threads Together
All 0 1 2 3 4
Create Group
Project Files
Search (Ctrl+K)
Application Code
/
Sources
guy1.c
main(int argc,char *argv[])
External Code
guy1.c
Locals Current Line(s) Current Stack
Current Line(s)
Input/Output Breakpoints Watchpoints Stacks Tracepoints Tracepoint Output Logbook
Evaluate
Type: none selected

```
46     for (i=0;i<X;i++) {
47           printf("%f ",temp[i][j]);
48           temperature[j*X+i]=temp[i][j];
49     }
50
51  printf("\n Verifing temperature array:\n");
52  for (i=0;i<N;i++)
53    printf("%f ",temperature[i]);
54  printf("\n");
55
56  }  // endif myid==ROOT
57
58  MPI_Scatter(temperature, X, MPI_FLOAT, recvb, X, MPI_FLOAT, I
59
60  printf("myid=%d %f %f %f %f\n",myid,recvb[0],recvb[1],recvb[
61
62  MPI_Finalize();
63  return 0;
64  }
65
```

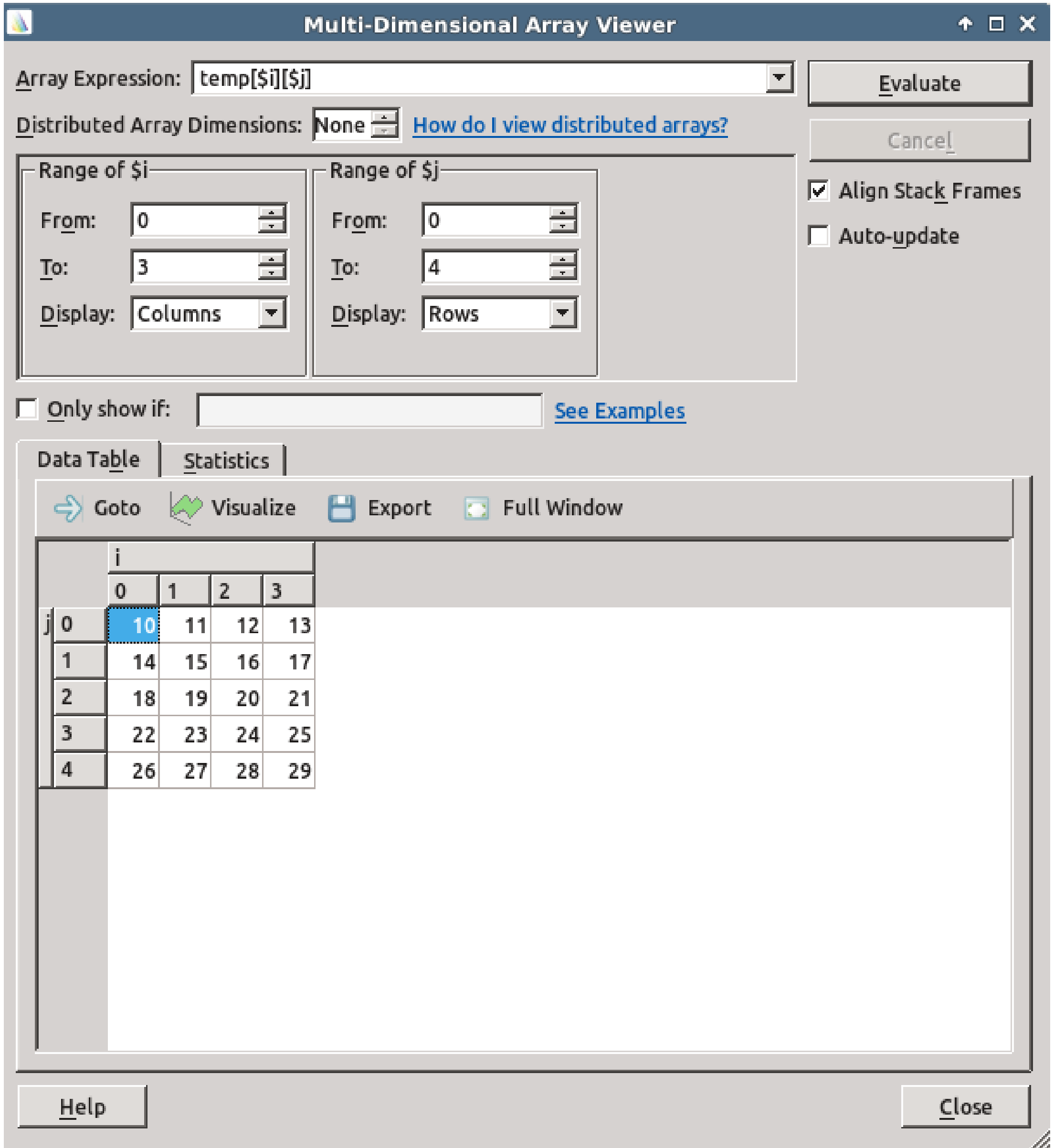
Variable Name	Value
myid	2
recvb	{[0] = 18, [1] = 19, [2] = 2(
[0]	18
[1]	19
[2]	20
[3]	21

Breakpoints

	Processes	Threads	File	Line	Function	Condition	Start After	Trigger Every	Stop After
☑	All	all	guy1.c	53	main		0	1	Forever
☑	All	all	guy1.c	60	main		0	1	Forever

Expression	Value

Allinea DDT



Array Expression: temp[$i][$j]

Distributed Array Dimensions: None

Range of $i
From: 0
To: 3
Display: Columns

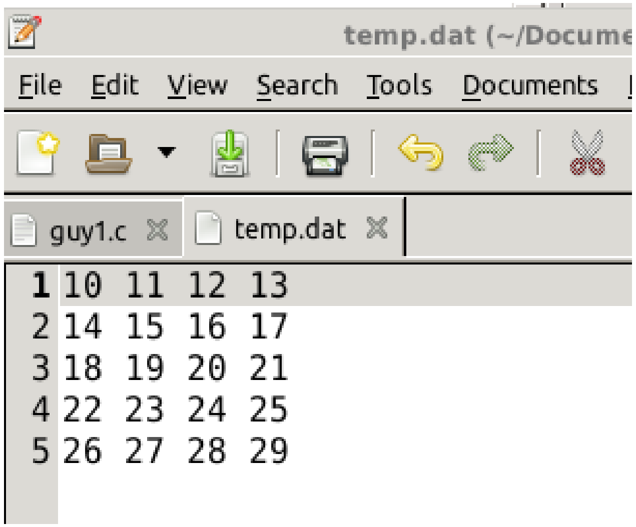
Range of $j
From: 0
To: 4
Display: Rows

☑ Align Stack Frames

□ Auto-update

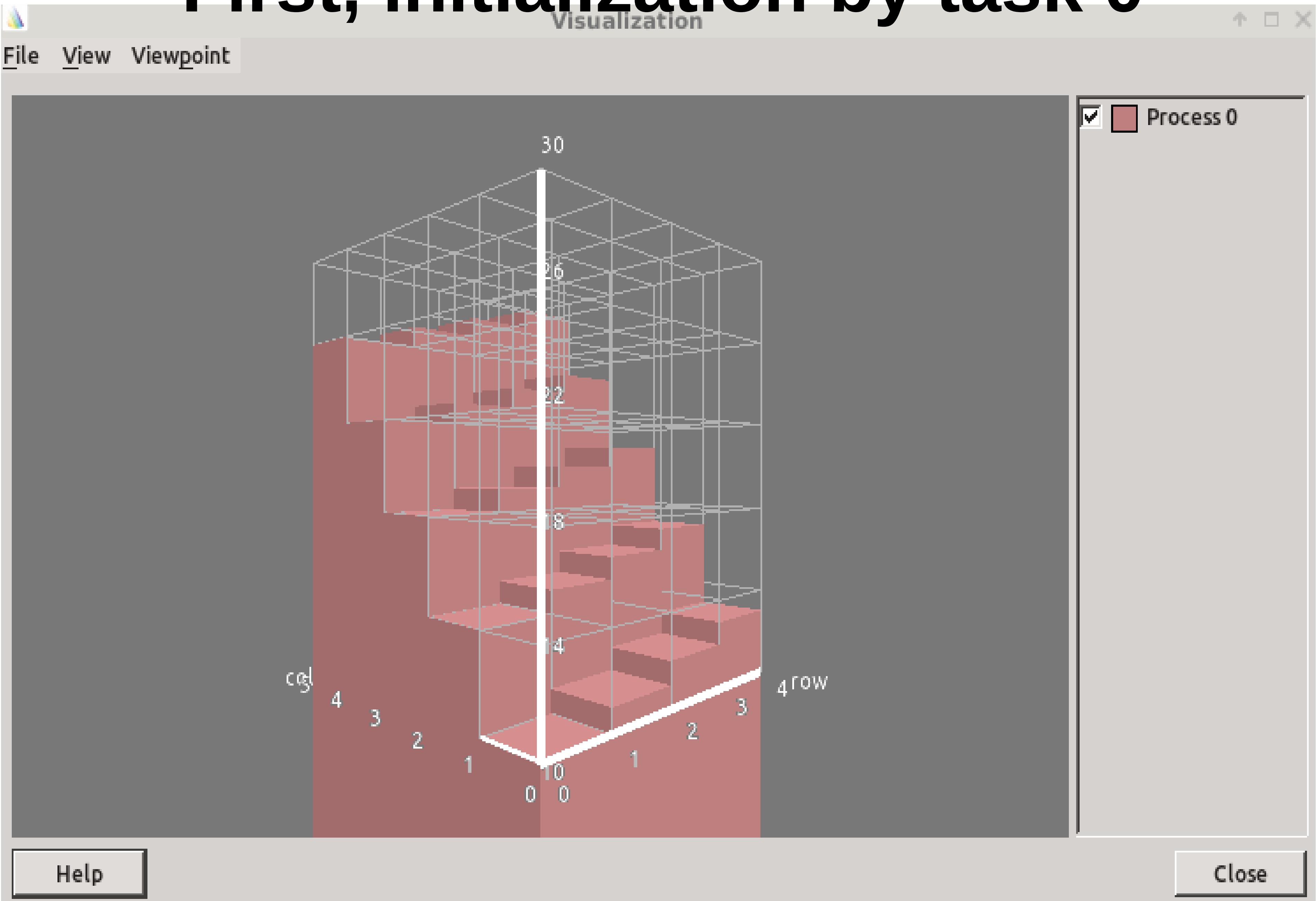
□ Only show if:

		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

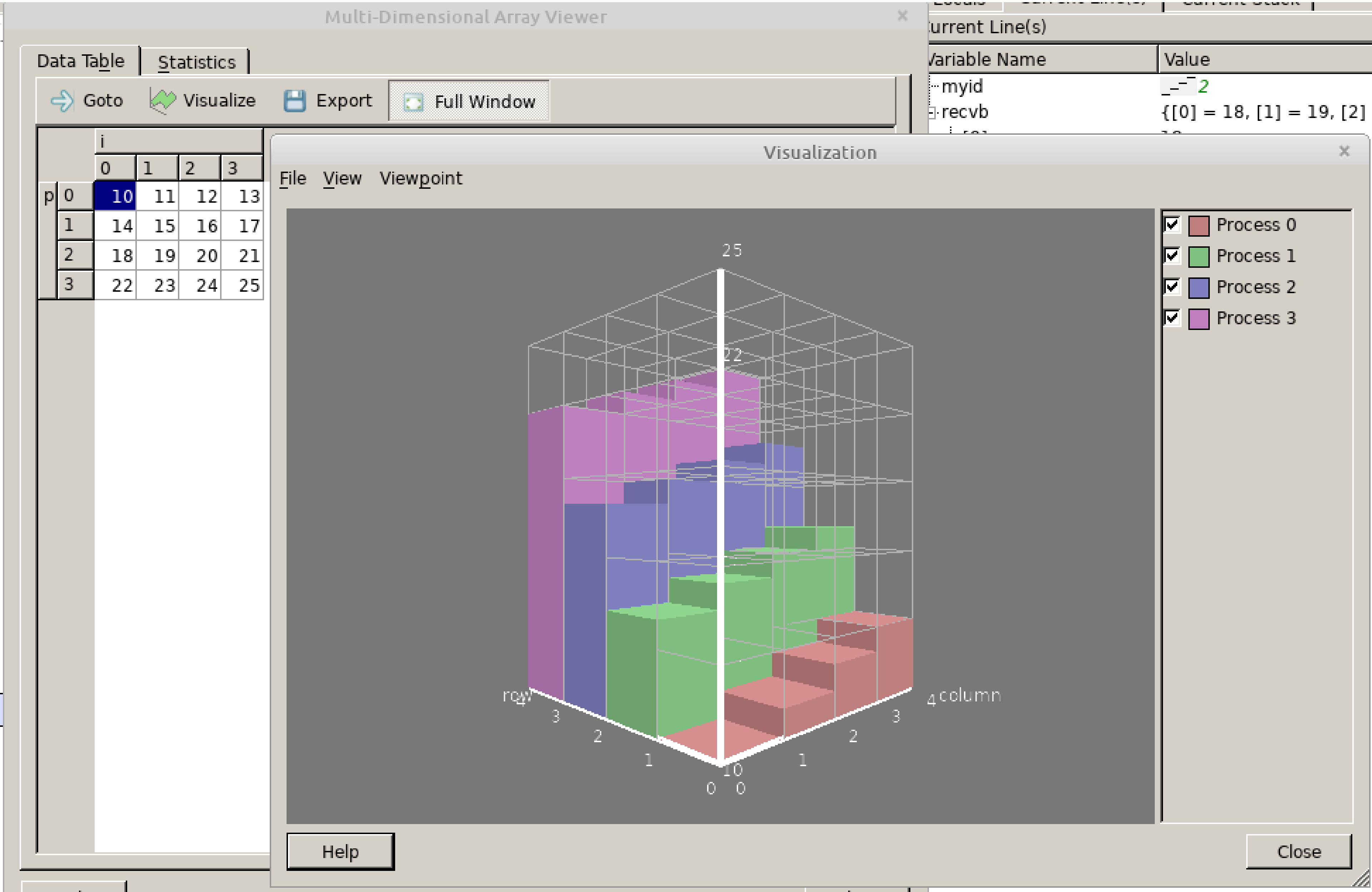


```
1 10 11 12 13
2 14 15 16 17
3 18 19 20 21
4 22 23 24 25
5 26 27 28 29
```

First, initialization by task 0



Then, rows are scattered to tasks



p \ i	0	1	2	3
0	10	11	12	13
1	14	15	16	17
2	18	19	20	21
3	22	23	24	25

Current Line(s)

Variable Name	Value
myid	2
recvb	{[0] = 18, [1] = 19, [2]

☑ Process 0
☑ Process 1
☑ Process 2
☑ Process 3

Demo 2: static vs. dynamic memory allocation

```
// m_size.c, this code demonstrates the
// limitation of static memory allocation for
// creating large Matrices
//  Folder: Matrix2

#define SIZE 800 // on my laptop 800 is still ok
but it crushes for
   //  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;
}
```