

# Parallel Processing

Guy Tel-Zur

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

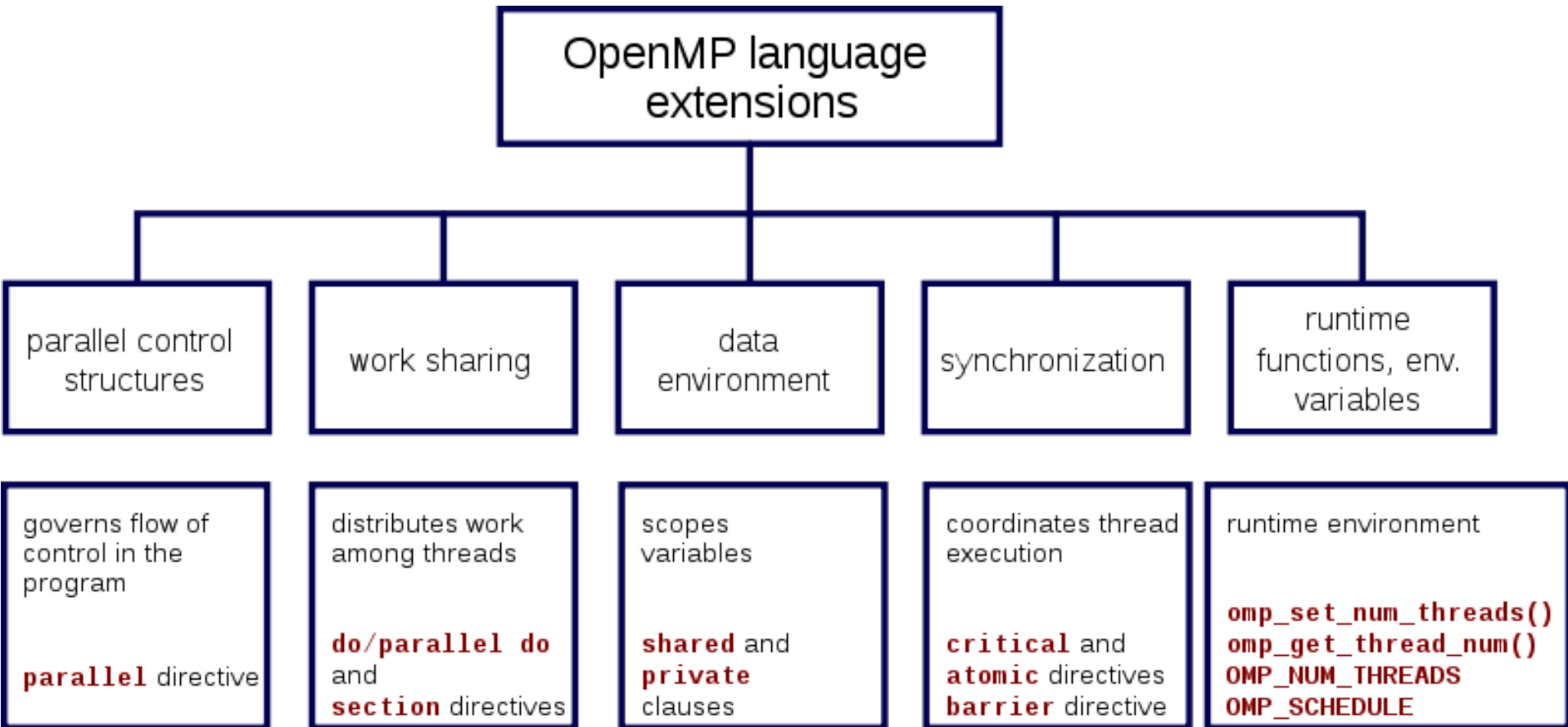
# **Agenda**

**Parallel programming in OpenMP**  
**- slides8**

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

# 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](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](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

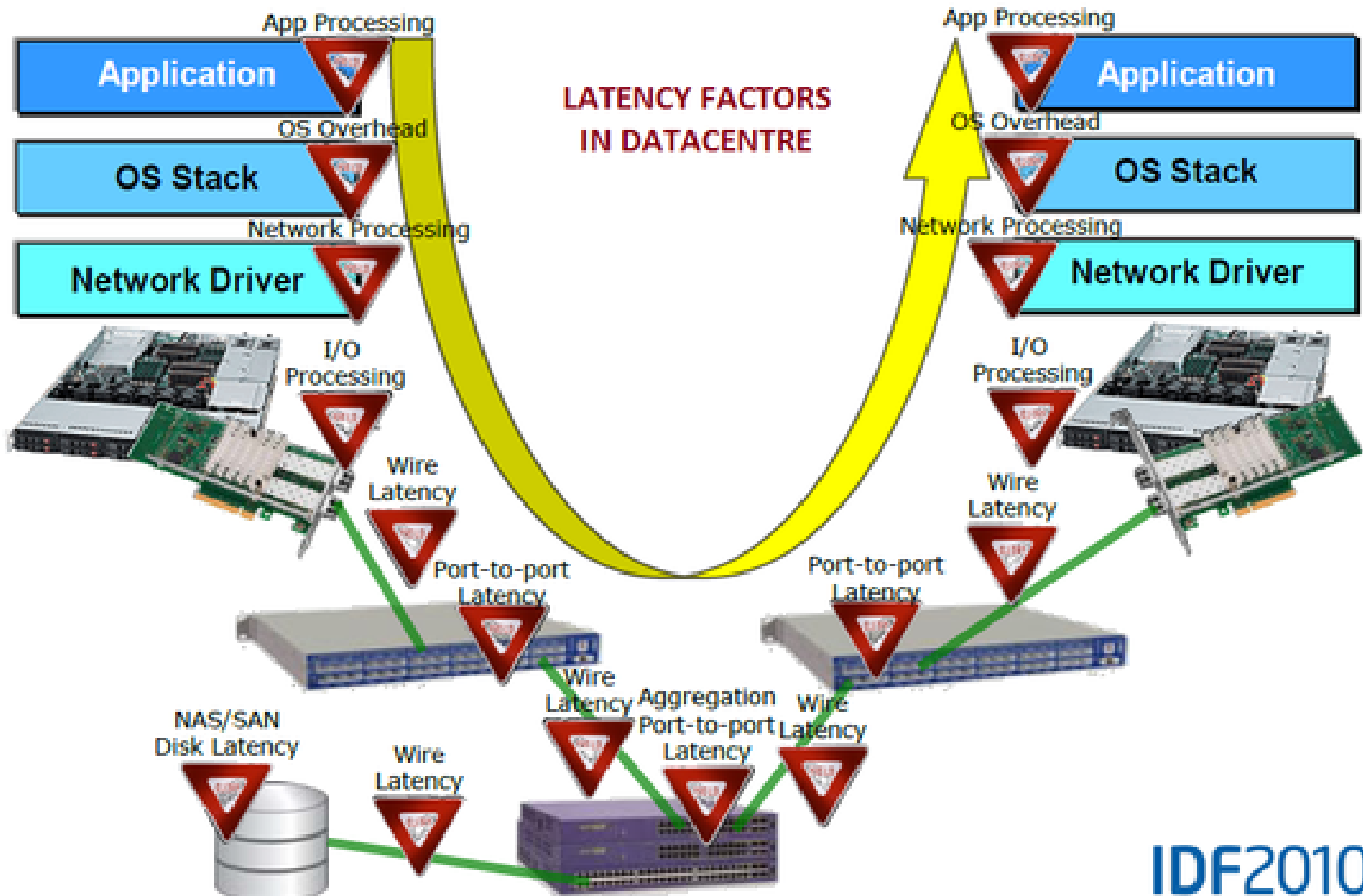
/usr/lib64/atlas/libcblas.so.3.0

/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

# Sending a Matrix 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/Ppxxx/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/PPxxx/lectures/08/code/Matrix2

Show: `m_size.c` and `m_size2.c`



C guy1.c

&lt;&gt; C/C++ Extension Release Notes

Release Notes: 1.29.1

```
1 // Demo of reading a matrix text file into master node,  
2 // then the master scatters the rows to the tasks  
3 #include <stdio.h>  
4 #include <stdlib.h>  
5 #include "mpi.h"  
6  
7 int main(int argc, char *argv[]) {  
8  
9     #define MAXLINE 1024  
10    int X=4, Y=4, N;  
11    int i, j;  
12    int ROOT=0; // master task  
13    char line[MAXLINE];  
14    float temp[X][Y];  
15    float *temperature;  
16    float recvb[X];  
17    int myid, numprocs;  
18  
19    N = X * Y;  
20  
21    temperature=(float *)malloc(sizeof(float)*X*Y);  
22  
23    MPI_Init(&argc, &argv);  
24    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);  
25    MPI_Comm_rank(MPI_COMM_WORLD, &myid);  
26
```

Demo 1

```
26
27 if (numprocs != Y) {
28     printf("This demo works with exactly 4 tasks. Exiting\n");
29     MPI_Abort;
30     exit(1);
31 }
32
33 if (myid == ROOT) {
34     FILE *fp1;
35     fp1=fopen("temp.dat","r");
36     for (j=0;j<Y;j++)
37     {
38         fgets(line,MAXLINE,fp1);
39         sscanf(line,"%f %f %f %f",&temp[0][j],&temp[1][j],&temp[2][j],&temp[3
40     }
41     fclose(fp1);
42
43     for (j=0;j<Y;j++)
44     for (i=0;i<X;i++) {
45         printf("%f ",temp[i][j]);
46         temperature[j*X+i]=temp[i][j];
47     }
48
49     printf("\n Verifying temperature array:\n");
50     for (i=0;i<N;i++)
51     printf("%f ",temperature[i]);
52     printf("\n");
53
54 } // endif myid==ROOT
55
56 MPI_Scatter(temperature, X, MPI_FLOAT, recvb, X, MPI_FLOAT, ROOT, MPI_COMM_WO
57
58 printf("myid=%d %f %f %f %f\n",myid,recvb[0],recvb[1],recvb[2],recvb[3]);
59
60 MPI_Finalize();
61 return 0;
62 }
```

# A demo using Alinea's DDT

The screenshot shows the Alinea DDT 4.2.2-39982 IDE. The main window displays the source code for `guy1.c`. The code includes a loop for printing a temperature array, a verification loop, and MPI-related functions. A breakpoint is set at line 60. The right-hand side of the IDE features a 'Locals' window showing the current state of variables: `myid` is 2, and `recvb` is an array with values [18, 19, 20, 21]. At the bottom, a 'Breakpoints' table lists two breakpoints at lines 53 and 60.

```
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 Verifying temperature array:\n");
52 for (i=0;i<N;i++)
53     printf("%f ",temperature[i]);
54 printf("\n");
55 } // endif myid==ROOT
56
57
58 MPI_Scatter(temperature, X, MPI_FLOAT, recvb, X, MPI_FLOAT,
59
60 printf("myid=%d %f %f %f %f\n",myid,recvb[0],recvb[1],recvb[2],recvb[3]);
61
62 MPI_Finalize();
63 return 0;
64 }
65
```

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

Processes	Threads	File	Line	Function	Condition	Start After	Trigger Every	Stop After	
<input checked="" type="checkbox"/>	All	all	guy1.c	53	main		0	1	Forever
<input checked="" type="checkbox"/>	All	all	guy1.c	60	main		0	1	Forever

# Alinea 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

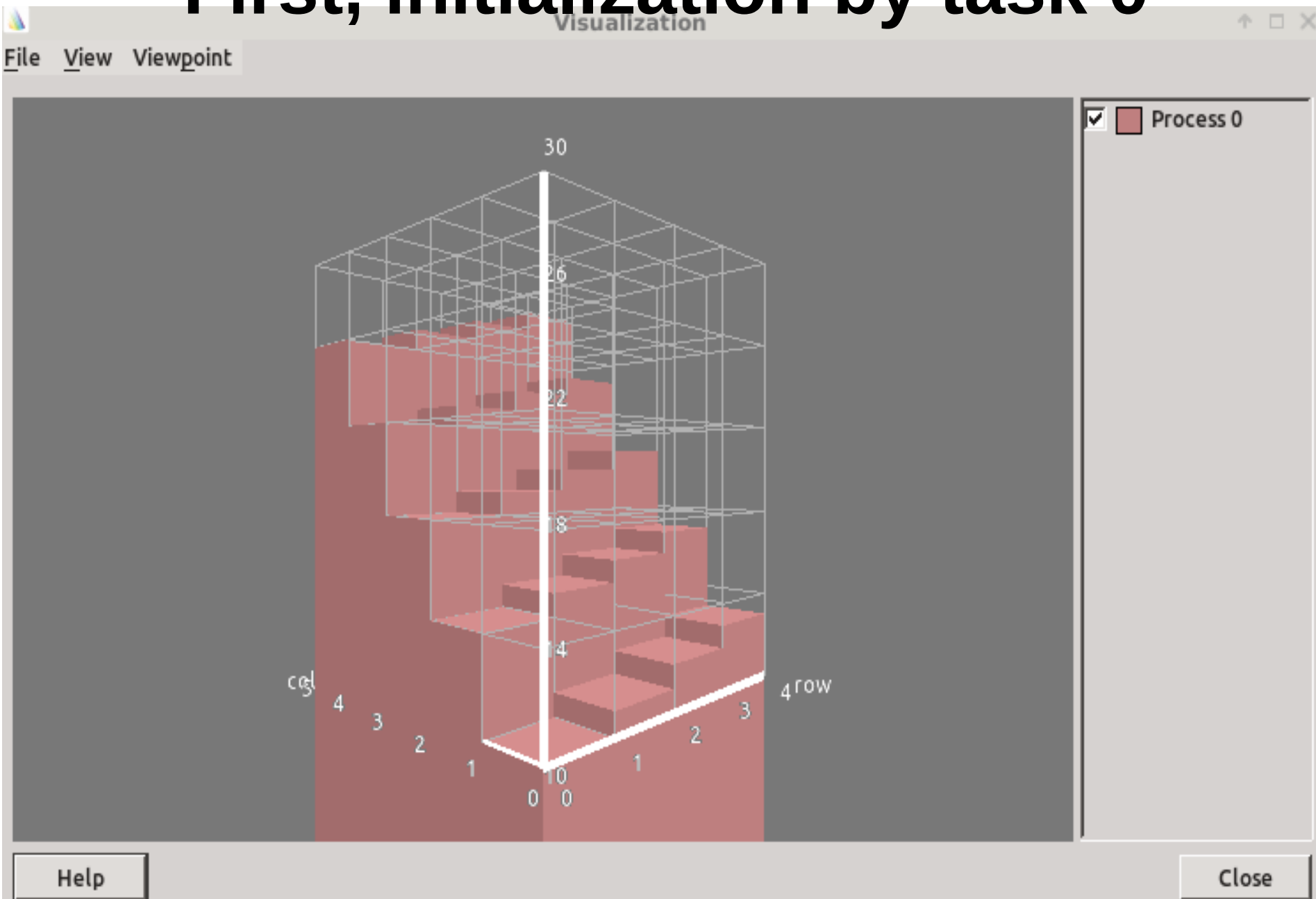
temp.dat (~/Docume)

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

Multi-Dimensional Array Viewer

Array Expression: `recvb[$i]` Evaluate Cancel

Distributed Array Dimensions: 1 [How do I view distributed arrays?](#)

Range of  $\$p$  (Distributed) Range of  $\$i$

From: 0 To: 4 Display: Rows

From: 0 To: 3 Display: Columns

Align Stack Frames  
 Auto-update

Locals Current Line(s) Current Stack

Current Line(s)

Visualization

Only show if:  See

Data Table Statistics

Goto Visualize Export Full Win

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
4	26	27	28	29

value

30

26

22

18

14

10

6

2

row

0 1 2 3 4

col

4 3 2 1 0

- Process 0
- Process 1
- Process 2
- Process 3
- Process 4

Help Close

# Demo 2: static vs. dynamic memory allocation

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

```
#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;
}
```