SAPIENZA
UNIVERSITÀ DI ROMA

TotalView Debugger

Dr. Fabrizio Gala
Dipartimento di Scienze di Base e Applicate
Per l'Ingegneria – Sezione di Fisica
Via Scarpa 14-16 00141 Rome, Italy

SAPIENZA
UNIVERSITÀ DI ROMA

Outline

- Starting TotalView on CRESCO

- Basic Debugging
  - Root Window
  - Stepping Toolbar
  - Action Points
  - Visualizing Arrays
  - Visualizing C++ Sturctures

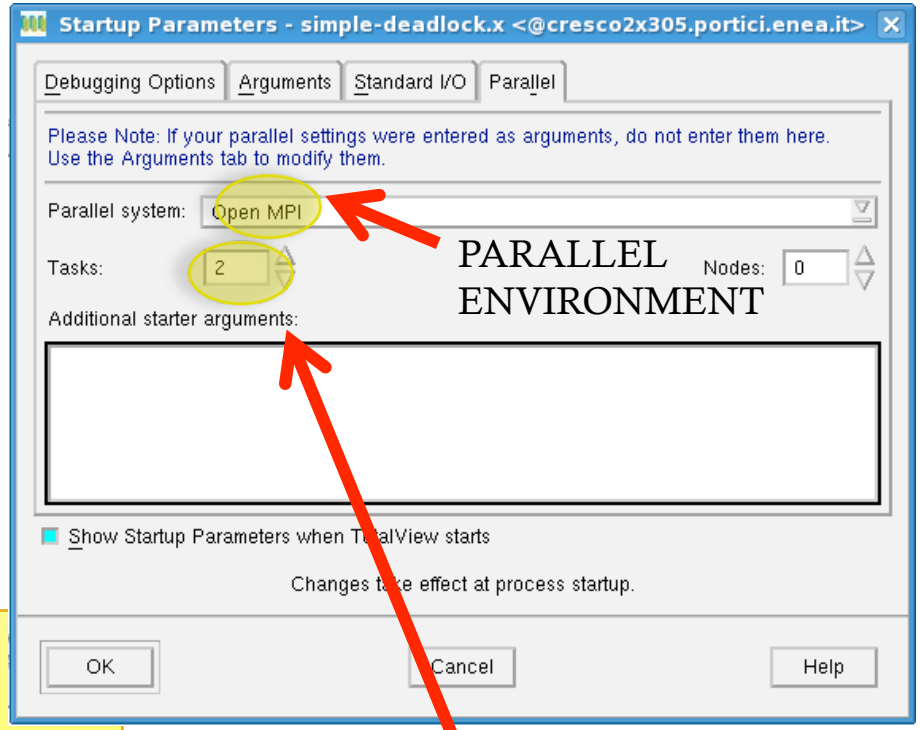
- MPI Debugging
  - Message Queue Graph

- Memory Menagement
  - Memory Debugging
  - Segmentation Faults
  - Memory Leaks

- Online Informations

Starting TotalView on CRESCO

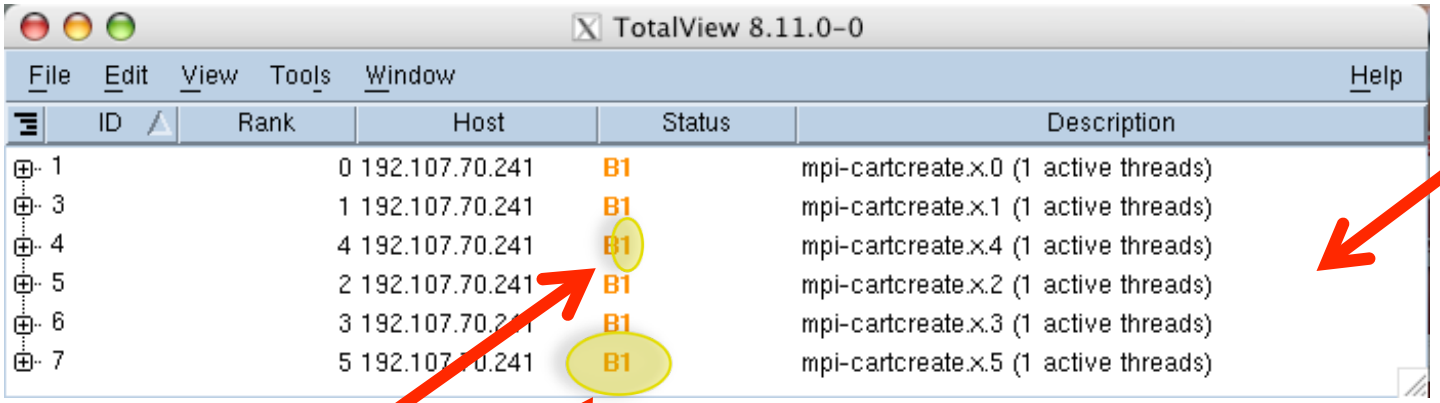
1) Open a terminal through **FARO**
2) Compile the code to debug
3) Load the **TotalView** module
4) Submit the job



1) http://www.cresco.enea.it/nx.html
2) mpicc **–g –O0** –o myprogram.x myprogram.c
3) ~/> module load **totalview**
4) **bsub –n #$_{procs}$ -I –a tv –q #$_{queue}$ totalview #$_{exe}$**

Root Window

The **Root Window** contains a list of all the processes and threads you are currently debugging.



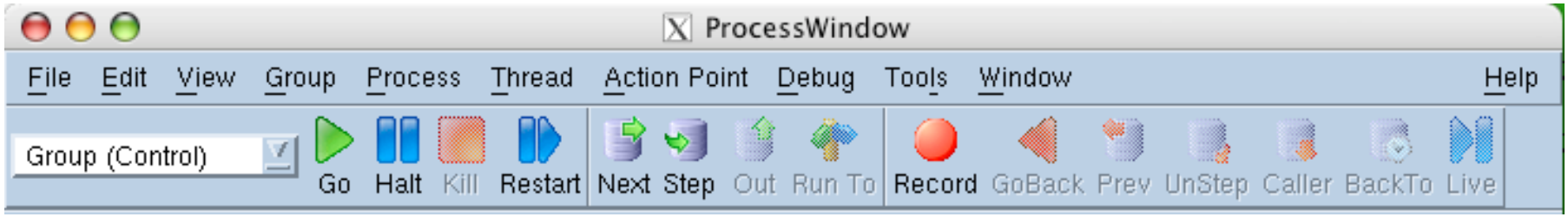
ROOT WINDOW

ACTION POINT ID NUMBER

STATUS INFO :

- T = sTopped
- B = Breakpoint
- E = Error
- W = Watchpoint
- R = Running
- M = Mixed

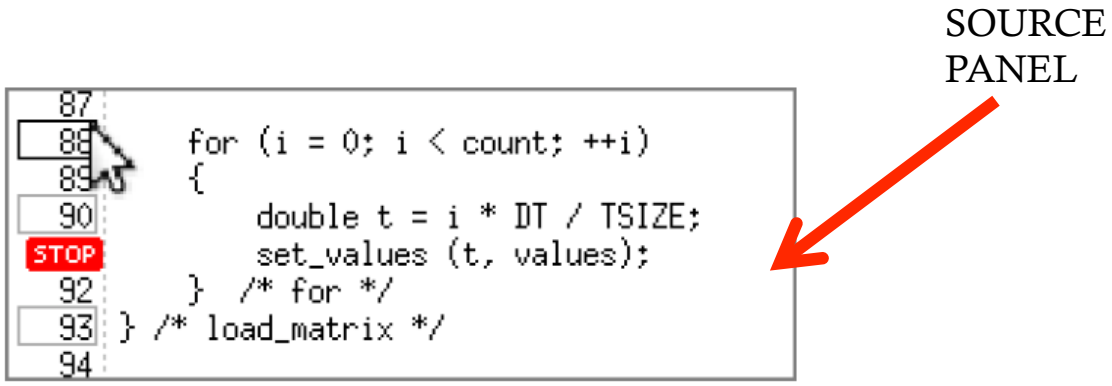
Stepping Toolbar



- Go : run the program
- Halt : pause the execution
- Kill : ends the process
- Restart : equal to Kill plus Go
- Next : executes all code on a line (i.e. a subroutine call)
- Step : executes the line
- Out : run up to return of a subroutine
- Run to : runs to a selected line

Action Points

To set an **action point**, click on the line number in the **source panel** and then choose what kind of action you need.



The basic **action points** are:

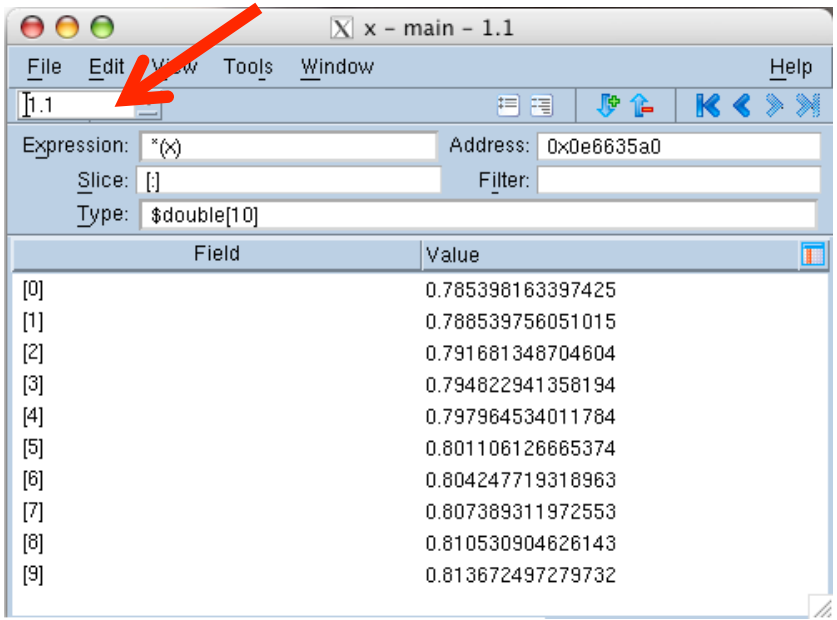
- **Breakpoint** : stops execution of the processes of threads that reach it,
- **Evaluation point** : executes a code fragment when it is reached,
- **Process Barrier Point** : hold each process when it reaches the barrier point until all process in the group have reached the barrir point

Visualizing Arrays

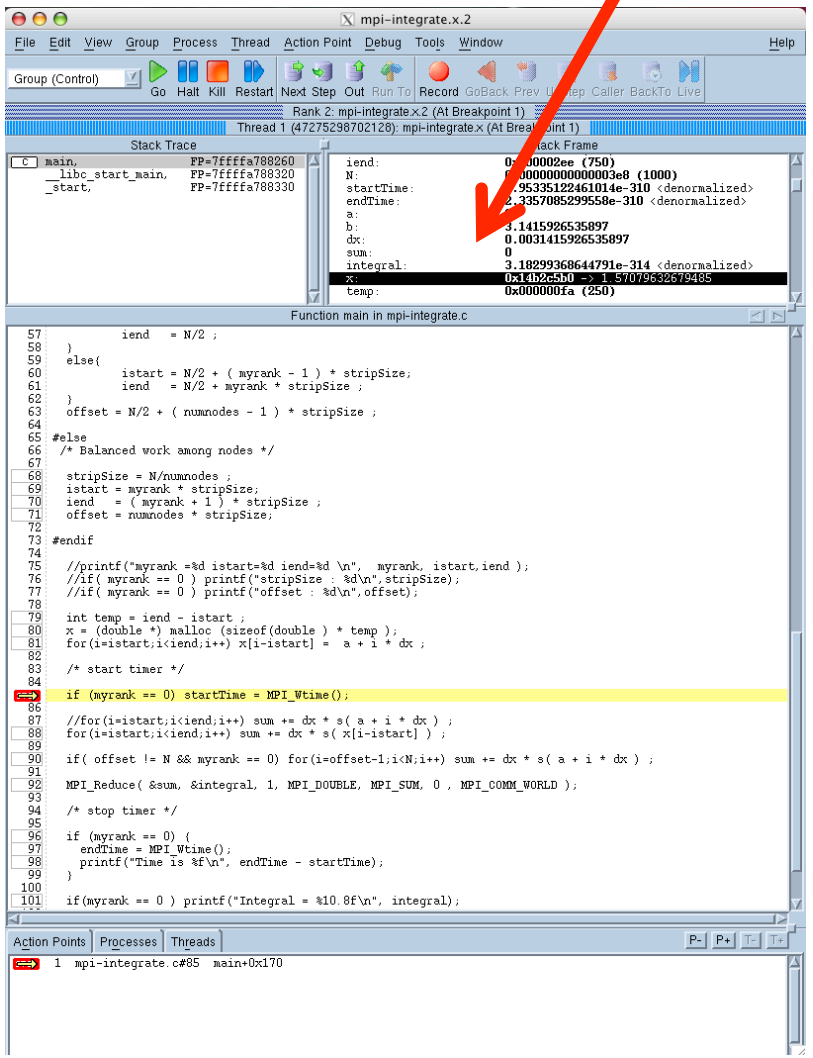
After having set a **breakpoint**, it is possible to have a look inside a variable, by simply selecting the variable in the **Stack Frame Panel** and then diving by a double-click.

This can be useful if one wants to explore a variable among different threads.

THREAD INDEX



STACK FRAME PANEL



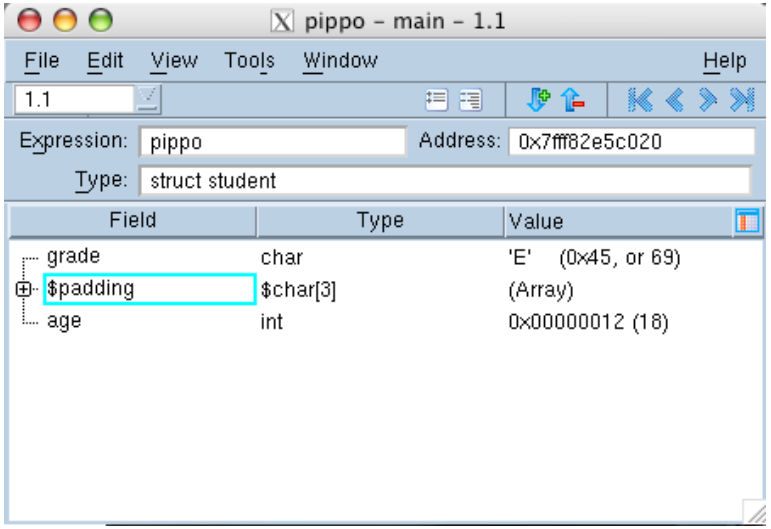
SAPIENZA
UNIVERSITÀ DI ROMA

Visualizing C++ Structures

To calculate the size of any **object type**, the compiler must take into account any address alignment that may be need to meet efficiency or achitectural constraints.

The compiler accomplishes this task by inserting unused **padding bytes** between members as needed to satisfy the alignment requirements. There may also be padding at the end of a structure to ensure proper alignment in case the structure is ever used as an element of an array.

Selecting **View > Padding** gives:



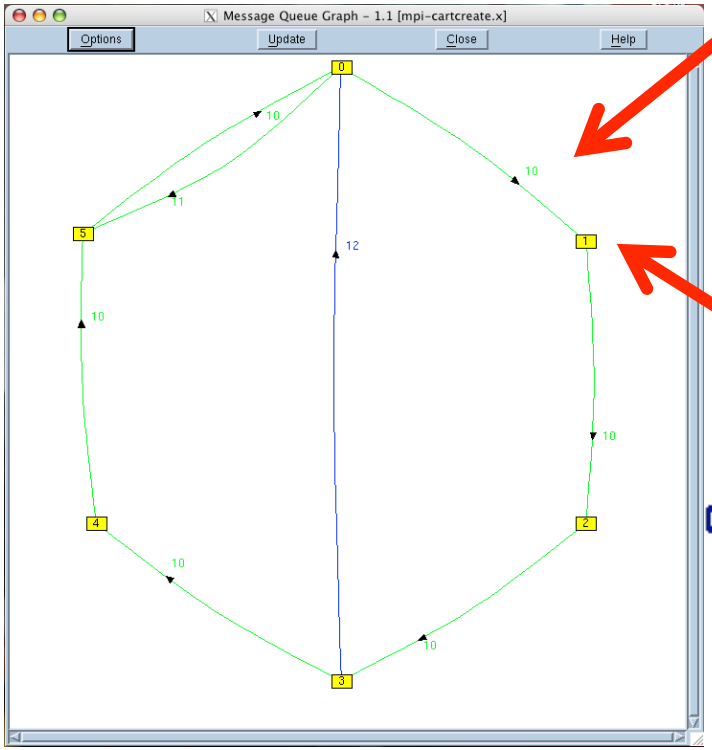

```
#include <iostream>

using namespace std;

struct student{
  char grade; /* char is 1 byte long */
  int age;    /* int is 4 bytes long */
};


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

student pippo;

pippo.grade = 'E';
pippo.age   = 18 ;

BARR printf("%c \n", pippo.grade);
printf("%d \n", pippo.age);
printf("Struct Size: %zu\n", sizeof (pippo));  /* Here the code prints 8 instead of 4 */


return 0 ;
}
```

CODE

Message Queue Graph (1)

Tools > Message Queue Graph displays a window that shows a graphic representation of the state of message queue information. This graph shows you the state of your program at a particular instant.

WARNING:
WORKS only ON CRESCO 1 or 2



MPI MESSAGE TAG

PROC ID

STATUS INFO :

- GREEN Pending Sends
- BLUE Pending Receives
- RED Unespected Msgs

```
if(myrank == 0 ) MPI_Isend(&b, 1, MPI_DOUBLE, (numnodes-1) , 11, MPI_NEW_COMM_WORLD,&request);
if(myrank == 0 ) MPI_Irecv(&c, 1, MPI_DOUBLE, numnodes/2  , 12, MPI_NEW_COMM_WORLD, &request);

MPI_Send(&a, 1, MPI_DOUBLE, rank_right, 10, MPI_NEW_COMM_WORLD);
BARR MPI_Recv(&b, 1, MPI_DOUBLE, rank_left , 10, MPI_NEW_COMM_WORLD, MPI_STATUS_IGNORE);

for(i=0;i<numnodes;i++) if( i == mycoord ) printf("%d %f %f\n",myrank, a,b);

if(myrank == numnodes/2 ) MPI_Isend(&c, 1, MPI_DOUBLE,  0 , 12, MPI_NEW_COMM_WORLD,&request);
if(myrank == numnodes-1 ) MPI_Irecv(&b, 1, MPI_DOUBLE,  0 , 11, MPI_NEW_COMM_WORLD, &request);
```

PSEUDO CODE

Message Queue Graph (2)

- Pending messages often indicate that a process can't keep up with the amount of work it is expected to perform.These messages indicate places where you may be able to improve your program's efficiency.

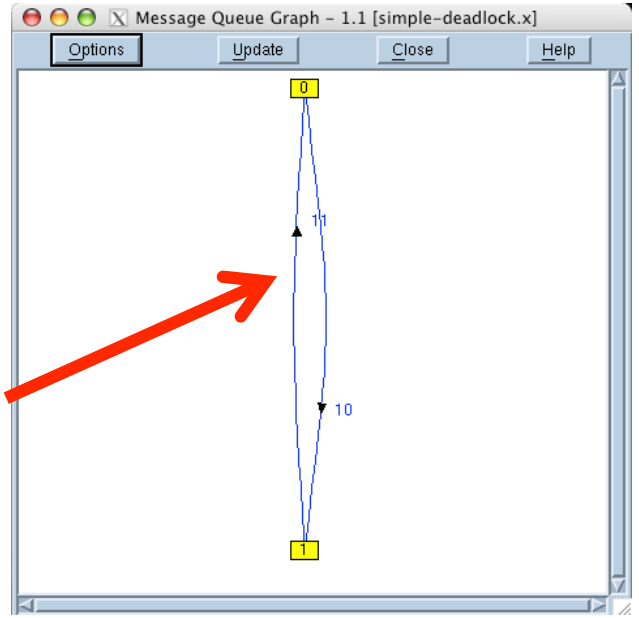
- Unexpected messages can indicate that something is wrong with your program because the receiving process doesn't know how to process the message.

- Loops in the graph may indicate **deadlocks**.(Cycles can be highlited by **Options > Cycle Detection**)

```
if(myrank == 0) {

      a[0] = 2.0 ;
      a[1] = 4.0 ;

      MPI_Recv(b, 2, MPI_DOUBLE, 1, 11, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      MPI_Send(a, 2, MPI_DOUBLE, 1, 10, MPI_COMM_WORLD);

}
else {

      a[0] = 3.0 ;
      a[1] = 5.0 ;

      MPI_Recv(b, 2, MPI_DOUBLE, 0, 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      MPI_Send(a, 2, MPI_DOUBLE, 0, 11, MPI_COMM_WORLD);

}
BARR
```

PSEUDO CODE

DEADLOCK EXAMPLE



SAPIENZA
UNIVERSITÀ DI ROMA

Memory Debugging (1)

Select the **Debug > Enable Memory Debugging** command before starting the program.

When the program makes a memory request, **MemoryScape** records the stack frames that existed when the action occurred.

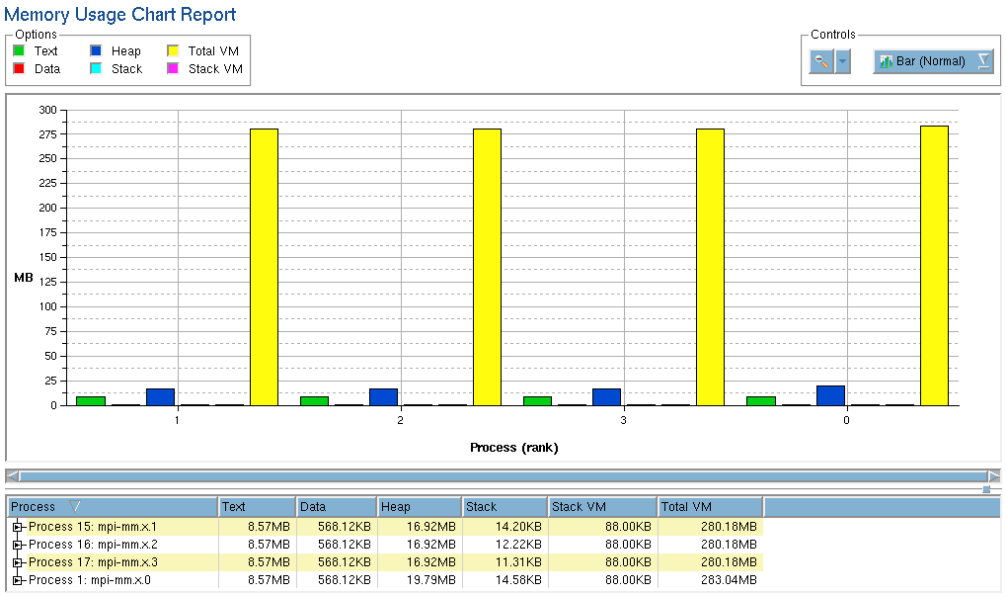
MemoryScape stops the program execution when one of the following events occurs:

- Freeing memory that is already freed,
- Freeing the wrong address
- Freeeing an interior pointer,
- Misaligning blocks

WARNING:
NOT WORKING correclty with OpenMPI compiled with intel

Memory Debugging (2)

Memory Usage Chart Report displays the memory occupied by each process:



Process	Text	Data	Heap	Stack	Stack VM	Total VM
Process 15: mpi-mm.x.1	8.57MB	568.12KB	16.92MB	14.20KB	88.00KB	280.18MB
Process 16: mpi-mm.x.2	8.57MB	568.12KB	16.92MB	12.22KB	88.00KB	280.18MB
Process 17: mpi-mm.x.3	8.57MB	568.12KB	16.92MB	11.31KB	88.00KB	280.18MB
Process 1: mpi-mm.x.0	8.57MB	568.12KB	19.79MB	14.58KB	88.00KB	283.04MB

TEXT: The amount of memory used for storing the program's machine code instructions.

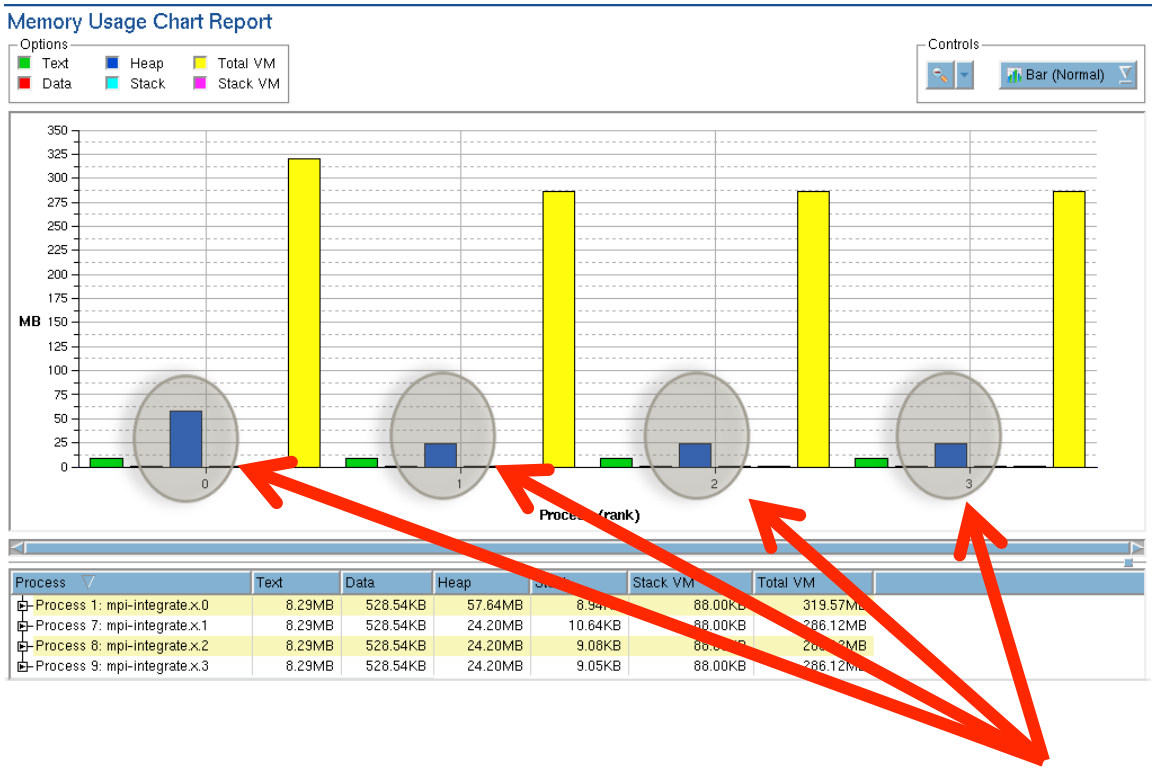
DATA: The amount of memory used for storing uninitialized and initialized data.

HEAP: The amount of memory currently being used for data created at runtime (i.e. allocated pointers).

STACK: The amount of memory used by the currently executing routine and all the routines in its backtrace.

Memory Debugging (3)

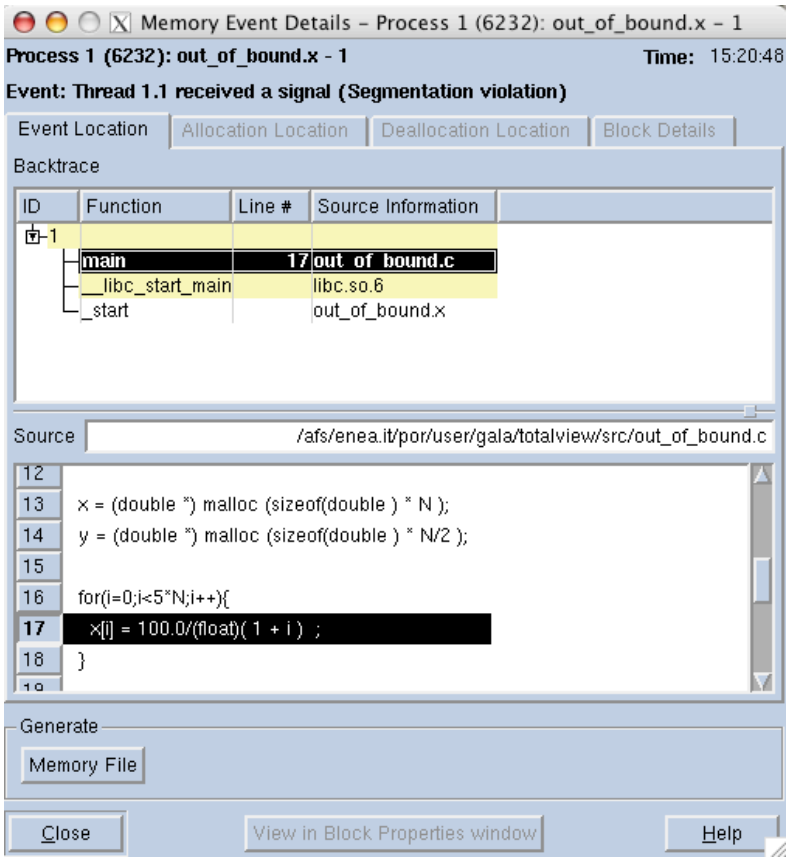
Looking at how the program is using memory may help in understanding where parallelization can be improved.



UNBALANCED MEMORY OCCUPANCY AMONG PROCESSES

Segmentation Faults

When a **Segmentation Fault** occurs, **MemoryScape** keeps track of the event, you can have a look at it in **Debug > Memory Event Details**)




```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <malloc.h>

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

  const int N = 10000 ;
  int i ;

  double    *x, *y;

  x = (double *) malloc (sizeof(double ) * N );
  y = (double *) malloc (sizeof(double ) * N/2 );

  for(i=0;i<5*N;i++){
        x[i] = 100.0/(float)( 1 + i )  ;
  }

  for(i=0;i<N/2;i++) y[i] = ( x[i] + x[N-i] ) / (float)2 ;

  for(i=0;i<N/2;i++) printf("%d %lf \n", i,y[i]);

  free(x);
  free(y);

  return 0;
}
```

CODE

SAPIENZA
UNIVERSITÀ DI ROMA

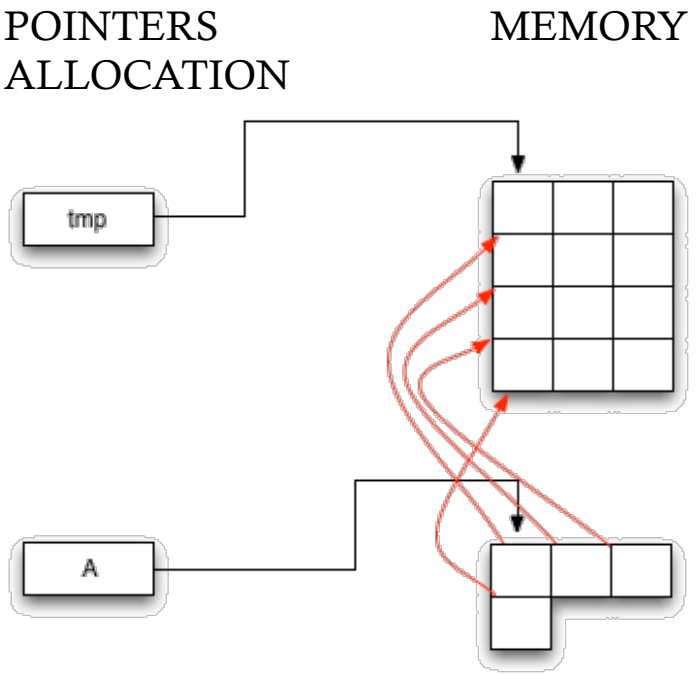
Memory Leaks (1)

A **memory leak** occurs when the program incorrectly menages memory allocations.

As a consequence, a memory leak can diminish the performance by reducing the amount of available memory. Eventually, in the worst case, too much of the available memory may become allocated and all or part of the system e stops working correctly, the application fails, or the system slows down unacceptably.

```
tmp = (double *) malloc (sizeof(double ) * N * N);
A = (double **) malloc (sizeof(double *) * N);
for (i = 0; i < N; i++)
  A[i] = &tmp[i * N];

 .......
 .......

free(A) ;
```

PSEUDO CODE

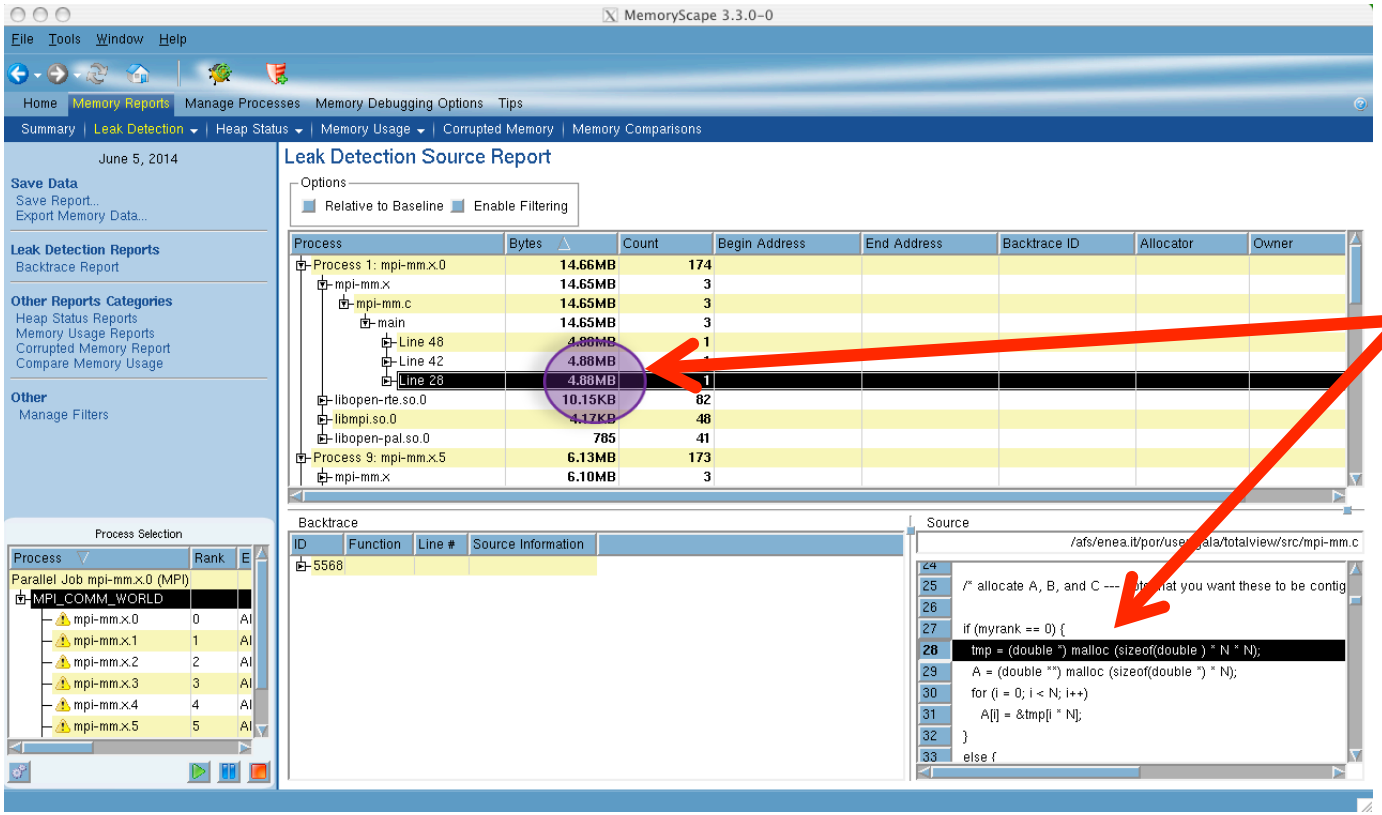


The instruction here frees only A, not tmp

MEMORY LEAK

Memory Leaks (2)

In **Leak Detection Reports > Source Report** in fact tells us :



$N = 800$

$sizeof(tmp) \simeq 8N^2 \sim 5$ Mb

sizeof (double)

SAPIENZA
UNIVERSITÀ DI ROMA

Online Information

TotalView documentation can be found here:

http://www.roguewave.com/support/product-documentation/totalview.aspx

THANKS TO Prof. G. ZOLLO and Dr. A. SCHIAVI

&

THANK YOU TO EVERYBODY
FOR THE ATTENTION