



# arm

## Debugging GPU Codes on Daint with DDT

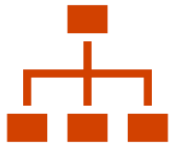
EuroHack18 - Lugano

# Arm Forge

An interoperable toolkit for debugging and profiling



Commercially supported  
by Arm



Fully Scalable



Very user-friendly

## The de-facto standard for HPC development

- Most widely-used debugging and profiling suite in HPC
- Fully supported by Arm on Intel, AMD, Arm, IBM Power, Nvidia GPUs, etc.

## State-of-the art debugging and profiling capabilities

- Powerful and in-depth error detection mechanisms (including memory debugging)
- Sampling-based profiler to identify and understand bottlenecks
- Available at any scale (from serial to petaflop applications)

## Easy to use by everyone

- Unique capabilities to simplify remote interactive sessions
- Innovative approach to present quintessential information to users

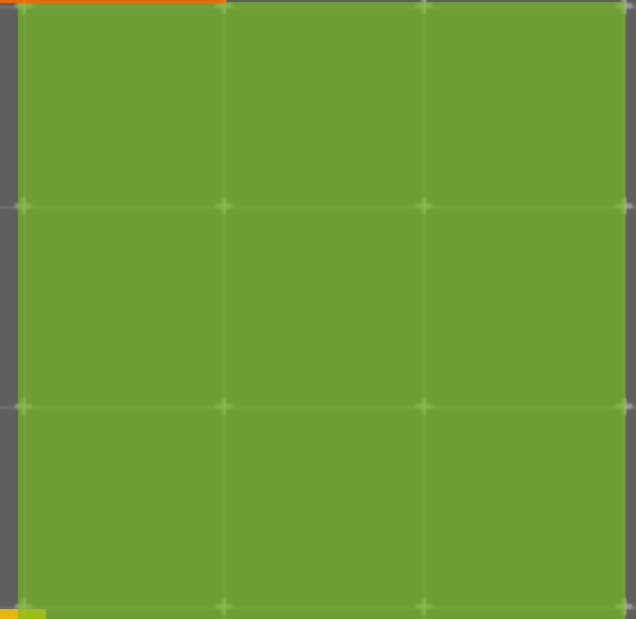
# Extra documentation

CSCS Documentation : <https://user.cscs.ch/computing/analysis/ddt/#using-ddt>

Arm Official Documentation : <https://developer.arm.com/docs/101136/latest/ddt>

Arm DDT Webinar : <https://www.youtube.com/playlist?list=PL1tk5lGm7zvR1CPR9KYZZEyRICYQYY-Xp>

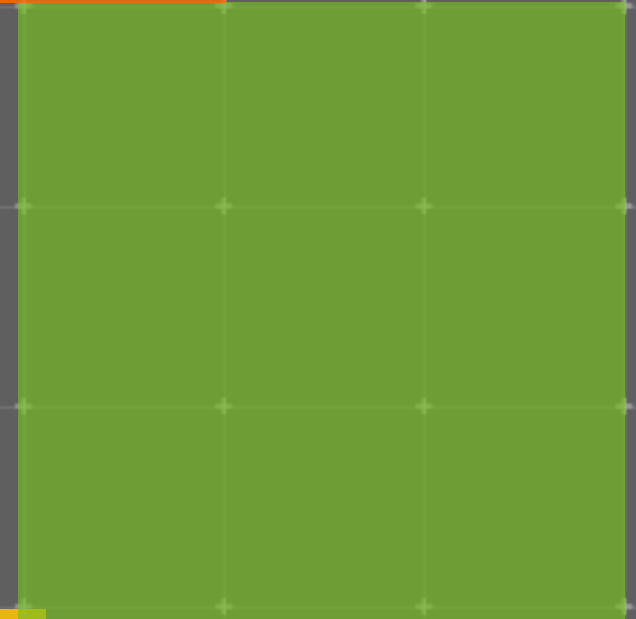
# Set up DDT



# Installation / Set Up

- Two ways of using the Graphical Interface of DDT :
  - X – Forwarding
  - Reverse Connect
- X-Forwarding :
  - No need to install any Client
- Reverse Connect :
  - When the system is far away
  - If X-forwarding becomes slow

# Reverse Connect



# Download DDT “Client” for Reverse - Connect

18.1.1 :

[http://content.allinea.com/downloads/arm-forge-18.1.1-Redhat-7.0-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.1.1-Redhat-7.0-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-18.1.1-Suse-12-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.1.1-Suse-12-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-18.1.1-Ubuntu-16.04-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.1.1-Ubuntu-16.04-x86_64.tar)

18.2.2 :

[http://content.allinea.com/downloads/arm-forge-18.2.2-Redhat-7.0-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.2.2-Redhat-7.0-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-18.2.2-Suse-12-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.2.2-Suse-12-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-18.2.2-Ubuntu-16.04-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-18.2.2-Ubuntu-16.04-x86_64.tar)

19.0 :

[http://content.allinea.com/downloads/arm-forge-19.0-preview2-Redhat-7.0-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-19.0-preview2-Redhat-7.0-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-19.0-preview2-Suse-12-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-19.0-preview2-Suse-12-x86_64.tar)

[http://content.allinea.com/downloads/arm-forge-19.0-preview2-Ubuntu-16.04-x86\\_64.tar](http://content.allinea.com/downloads/arm-forge-19.0-preview2-Ubuntu-16.04-x86_64.tar)

# Download Mac OS/X Client

18.1.1 :

[http://content.allinea.com/downloads/arm-forge-client-18.1.1-MacOSX-10.7.5-x86\\_64.dmg](http://content.allinea.com/downloads/arm-forge-client-18.1.1-MacOSX-10.7.5-x86_64.dmg)

18.2.2 :

[http://content.allinea.com/downloads/arm-forge-client-18.2.2-MacOSX-10.7.5-x86\\_64.dmg](http://content.allinea.com/downloads/arm-forge-client-18.2.2-MacOSX-10.7.5-x86_64.dmg)

19.0 :

[http://content.allinea.com/downloads/arm-forge-client-19.0-preview2-MacOSX-10.7.5-x86\\_64.dmg](http://content.allinea.com/downloads/arm-forge-client-19.0-preview2-MacOSX-10.7.5-x86_64.dmg)



# Install DDT “Client” for Reverse - Connect

```
wget http://content.allinea.com/downloads/arm-forge-18.1.1-Redhat-7.0-x86_64.tar
```

```
tar -xvf arm-forge-18.1.1-Redhat-7.0-x86_64.tar
```

```
cd arm-forge-18.1.1-Redhat-7.0-x86_64
```

```
./text-install
```

[ Accept the license and specify the path of the Install ]

```
export PATH=PATH:$<path_of_installation>/bin
```

# Configure a Remote Connection:

## How to establish a connection to Daint :

```
ssh hckxx@ela.cscs.ch  
ssh daint
```

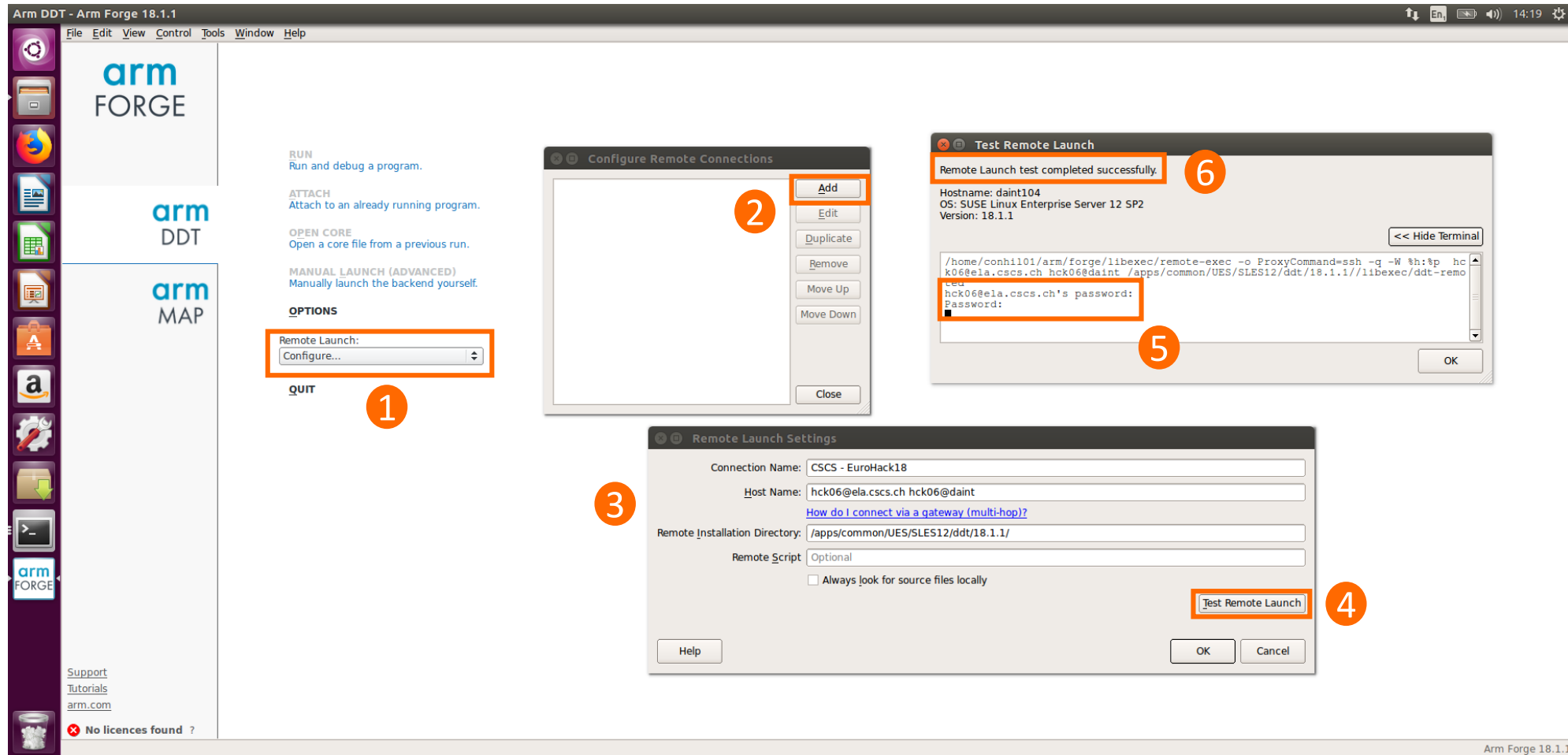
## How to set up Remote connection to Daint for DDT :

Click on Remote Launch / Configure / Add

The remote Installation directory can be :  
(select the same version than the one you have installed)

```
/apps/common/UES/SLES12/ddt/18.1.1/  
/users/hck06/DDT/18.2/  
/users/hck06/DDT/19.0/
```

# Configure a Remote Connection:



# Establish a Reverse-Connect connection:

## On Daint :

```
ssh hckxx@ela.cscs.ch
ssh daint
```

```
module load daint-gpu
```

```
module load ddt -> For 18.1.1
```

```
export PATH=$PATH:/users/hck06/DDT/18.2.2/bin
```

```
export PATH=$PATH:/users/hck06/DDT/19.0/bin
```

```
[...]
```

[Resources Allocation]

```
ddt --connect <mpi_exec_launch_command>
```

## Examples :

```
ddt --connect srun -n 1 ./main.exe
```

```
ddt --connect srun -reservation=hackathon -C
```

```
gpu -n 1 ./main.exe
```

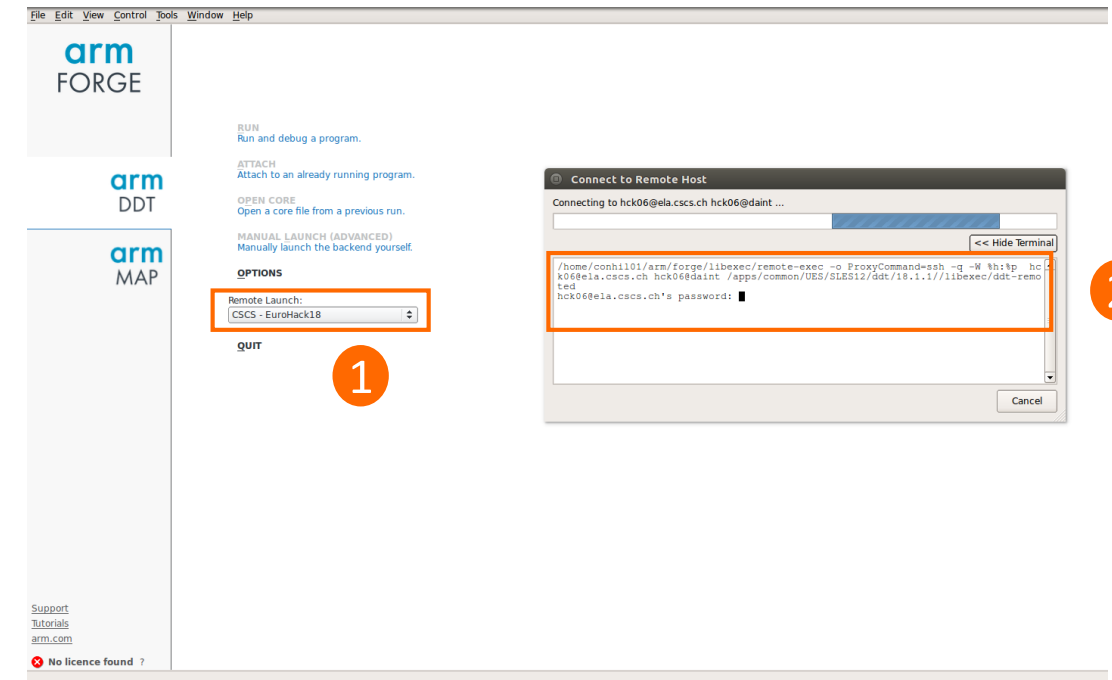


Just one line needed  
depending on the version  
installed locally (laptop)

## On your laptop :

Remote Launch / Select the Connection that you have set up

Enter passwords if required



# Establish a Reverse-Connect connection:

The screenshot displays the Arm Forge IDE interface. On the left is a sidebar with logos for 'arm FORGE', 'arm DDT', and 'arm MAP'. The main window has a menu bar (File, Edit, View, Control, Tools, Window, Help) and a central area with several buttons: 'RUN' (Run and debug a program), 'ATTACH' (Attach to an already running program), 'OPEN CORE' (Open a core file from a previous run), 'MANUAL LAUNCH (ADVANCED)' (Manually launch the backend yourself), 'OPTIONS', 'Remote Launch:' (with a dropdown menu showing 'CSCS - EuroHack18'), and 'QUIT'. A 'Reverse Connect Request' dialog box is open in the foreground, displaying an information icon, the text 'A new Reverse Connect request is available from daint101 for Arm DDT.', the command line '--connect srun -n 1 ./hello\_c', and the question 'Do you want to accept this request?'. The dialog has 'Help', 'Accept', and 'Reject' buttons. The status bar at the bottom shows 'Arm Forge 18.1.1 Connected to: hck06@ela.cscs.ch hck06@daint'.

arm FORGE

arm DDT

arm MAP

Support  
tutorials  
m.com

ence Serial: 6784 ?

File Edit View Control Tools Window Help

**RUN**  
Run and debug a program.

**ATTACH**  
Attach to an already running program.

**OPEN CORE**  
Open a core file from a previous run.

**MANUAL LAUNCH (ADVANCED)**  
Manually launch the backend yourself.

**OPTIONS**

Remote Launch:  
CSCS - EuroHack18

**QUIT**

**Reverse Connect Request**

A new Reverse Connect request is available from daint101 for Arm DDT.

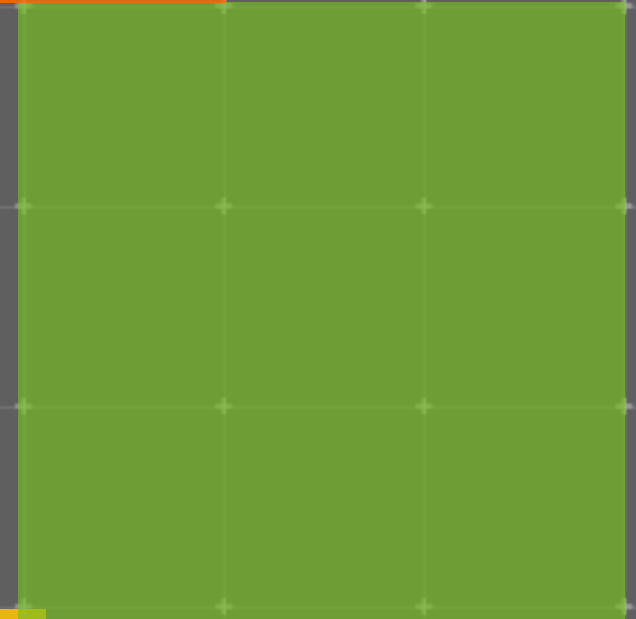
Command Line: --connect srun -n 1 ./hello\_c

Do you want to accept this request?

Help Accept Reject

Arm Forge 18.1.1 Connected to: hck06@ela.cscs.ch hck06@daint

# X-Forwarding



# Using X-Forwarding :

## On Daint :

```
ssh -X hckxx@ela.cscs.ch  
ssh -X daint
```

Just one line needed  
depending on the version

```
module load ddt -> For 18.1.1  
export PATH=$PATH:/users/hck06/DDT/18.2.2/bin  
export PATH=$PATH:/users/hck06/DDT/19.0/bin
```

[...]

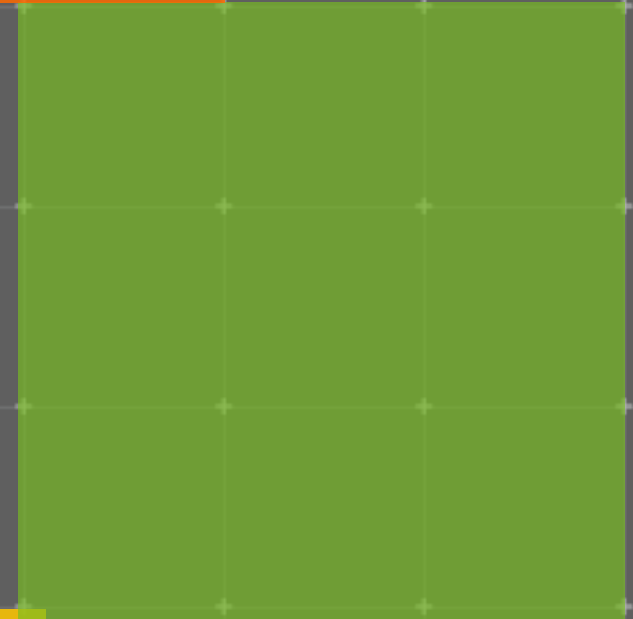
[Resources Allocation]

```
ddt <mpi_exec_launch_command>
```

## Example :

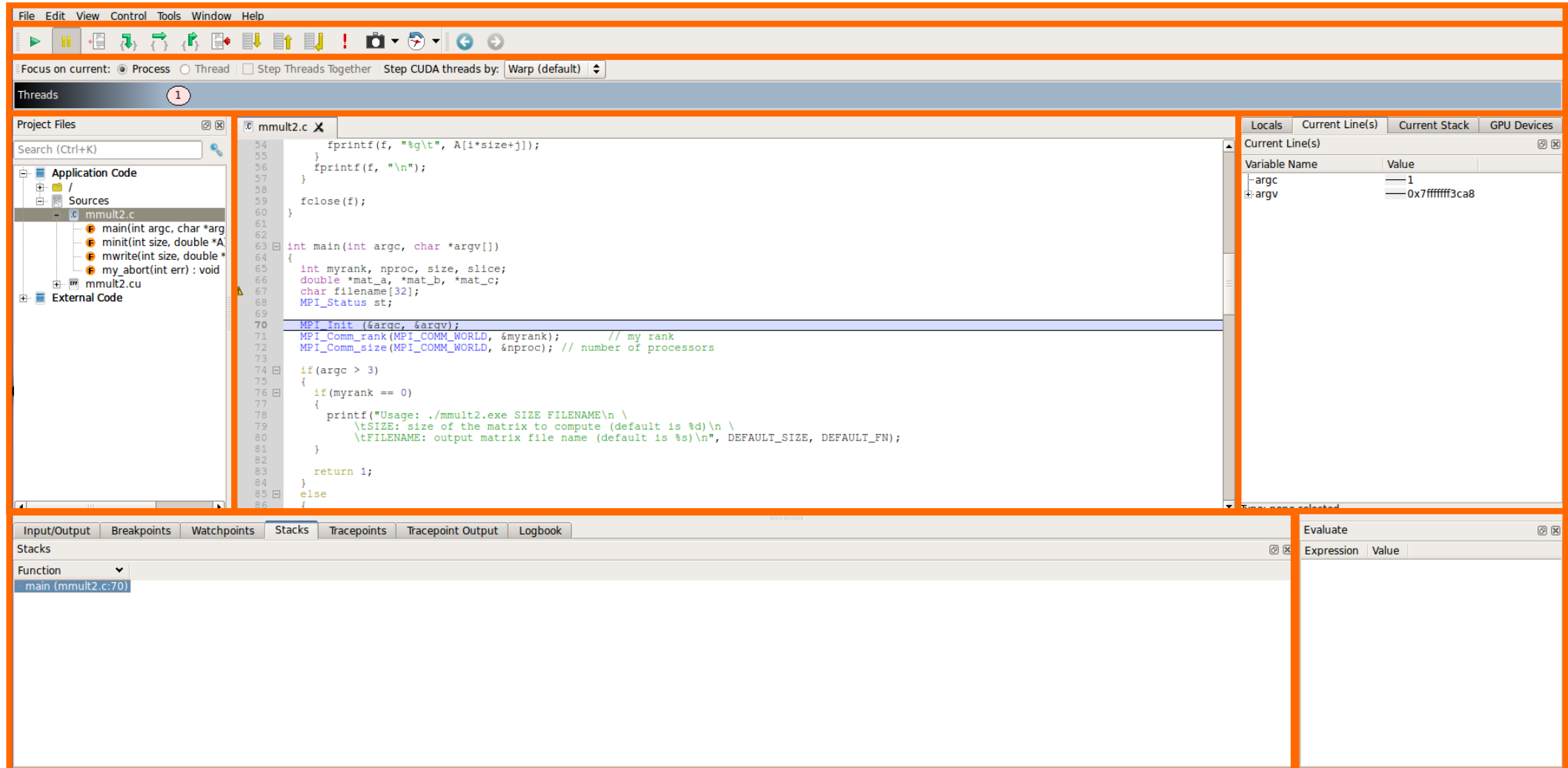
```
ddt srun -n 1 ./main.exe
```

# Key Features

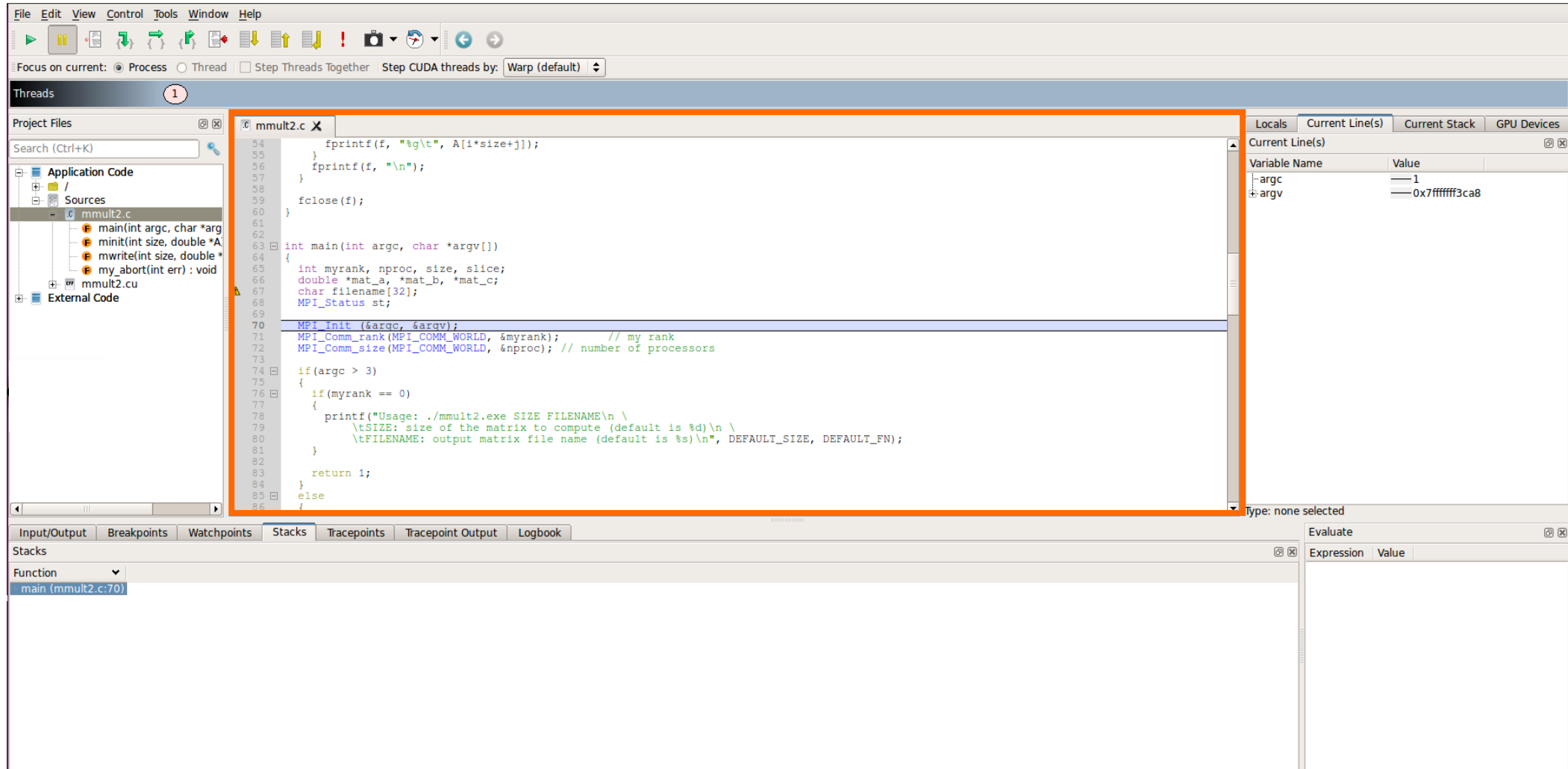




# User Interface



# User Interface – Source code viewer



# User Interface – Play/ Pause / Step

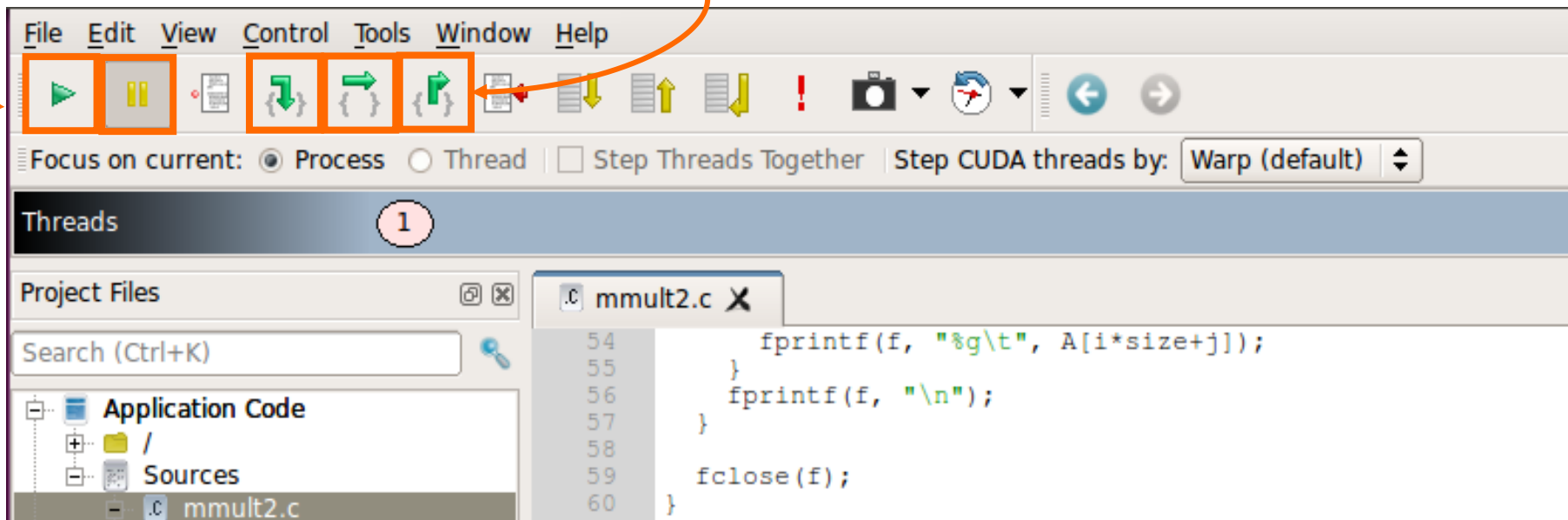
Play : Run everything. Use typically at the beginning or after Pause

Pause : Stops running current kernel

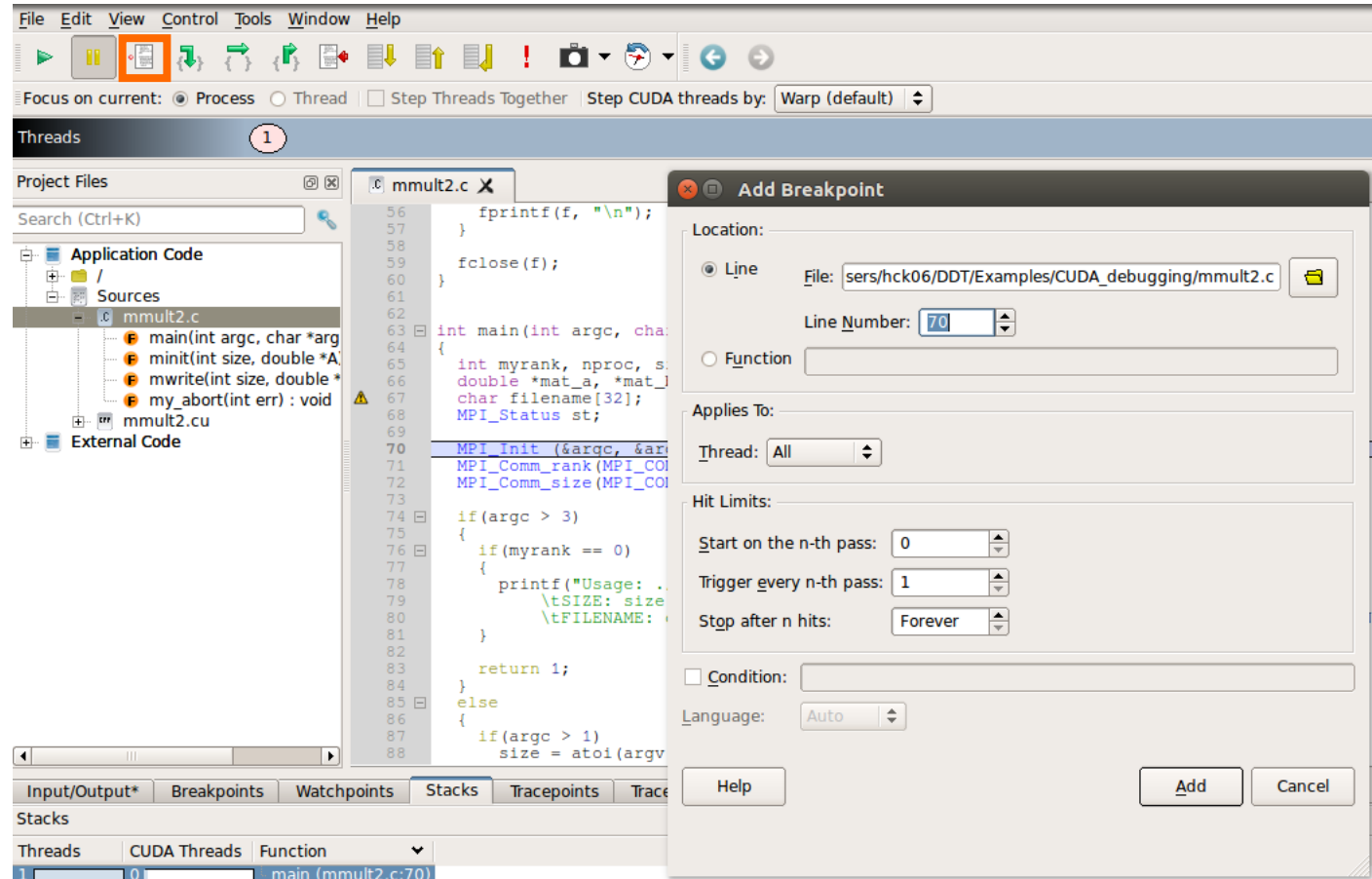
Step In : Enter a function call and display source code of the function

Step Over : Execute current line of code

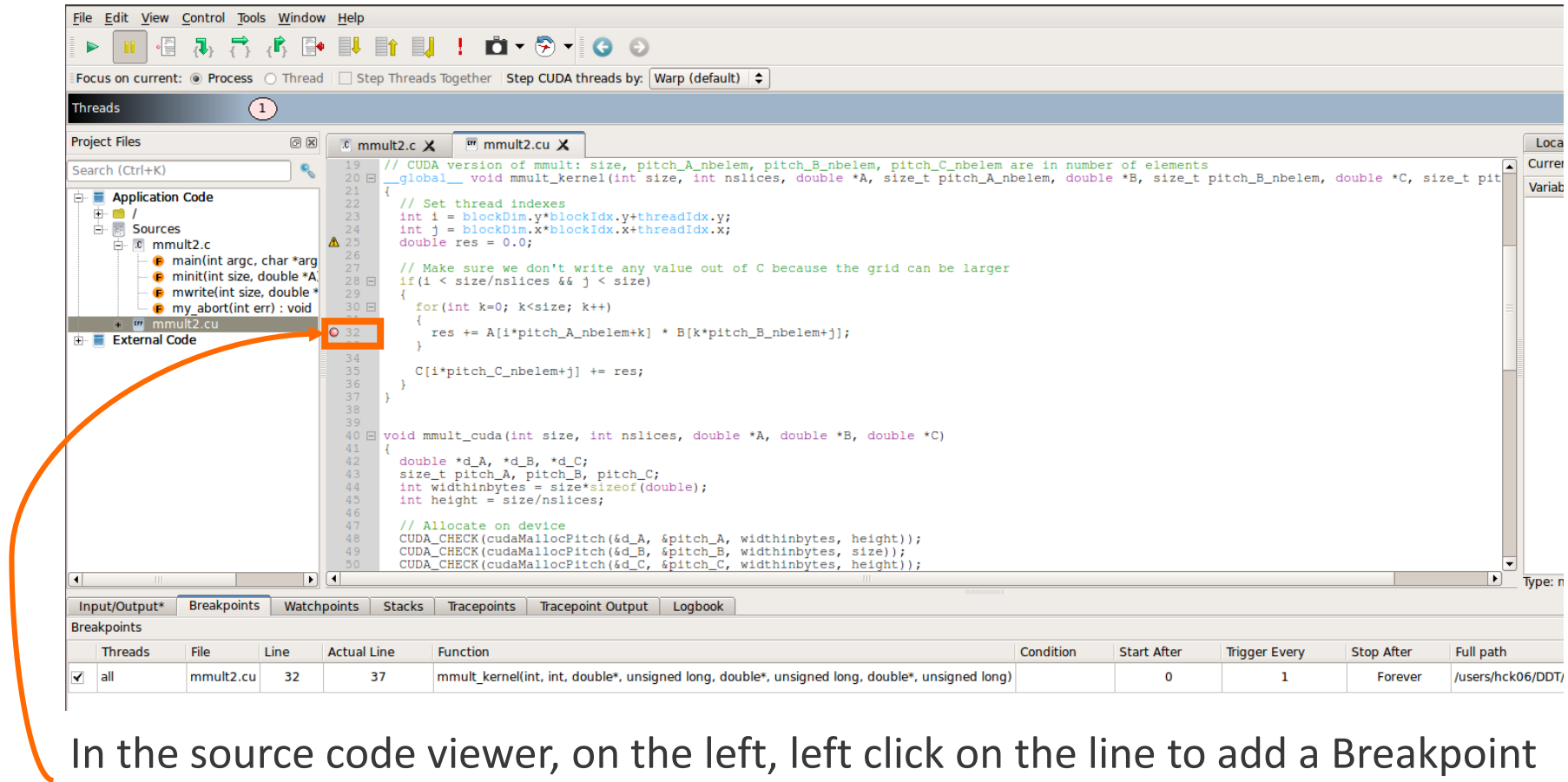
Step Out : Comes back one stage above current stack



# User Interface – Add Breakpoints – Way 1



# User Interface – Add Breakpoints – Way 2



In the source code viewer, on the left, left click on the line to add a Breakpoint  
Typical next action : Play

The list of the Breakpoints is available on the bottom panel. You can uncheck it to deactivate it.  
Tip : Right click on it and `Jump to` displays the source code around the Breakpoint.

# User Interface – Local Variables

CUDA Threads (mmult\_kernel) Block 19 Thread 21 Grid size: 1x64x1 Block size: 64x1x1

Project Files

Search (Ctrl+K)

Application Code

Sources

mmult2.c

mmult2.cu

External Code

```
18 // CUDA version of mmult: size, pitch_A_nbelem, pitch_B_nbelem, pitch_C_nbelem are in number of elements
19 __global__ void mmult_kernel(int size, int nslices, double *A, size_t pitch_A_nbelem, double *B, size_t pitch_B_nbelem, double *C, size_t pitch_C_nbelem)
20 {
21     // Set thread indexes
22     int i = blockDim.y*blockIdx.y+threadIdx.y;
23     int j = blockDim.x*blockIdx.x+threadIdx.x;
24     double res = 0.0;
25
26     // Make sure we don't write any value out of C because the grid can be larger
27     if(i < size/nslices && j < size)
28     {
29         for(int k=0; k<size; k++)
30         {
31             res += A[i*pitch_A_nbelem+k] * B[k*pitch_B_nbelem+j];
32         }
33
34         C[i*pitch_C_nbelem+j] += res;
35     }
36 }
37
38 void mmult_cuda(int size, int nslices, double *A, double *B, double *C)
39 {
40     double *d_A, *d_B, *d_C;
41     size_t pitch_A, pitch_B, pitch_C;
42     int widthinbytes = size*sizeof(double);
43     int height = size/nslices;
44     // ...
45 }
```

Locals

Variable Name	Value
A	0x10216200000
B	0x10216208000
C	0x10216210000
i	19
j	21
nslices	1
pitch_A_nbelem	512
pitch_B_nbelem	512
pitch_C_nbelem	512
res	0
size	64

Type: none selected

Value of local variables i and j depending on the thread and block

CUDA Threads (mmult\_kernel) Block 28 Thread 32 Grid size: 1x64x1 Block size: 64x1x1

Project Files

Search (Ctrl+K)

Application Code

Sources

mmult2.c

mmult2.cu

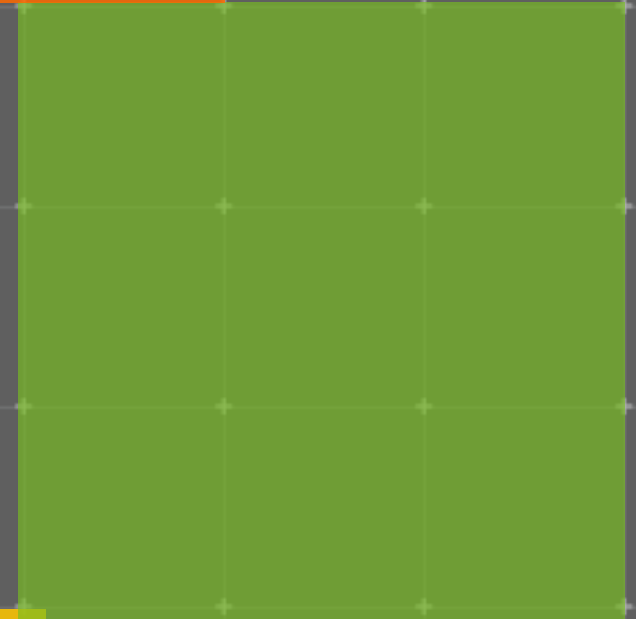
External Code

```
18 // CUDA version of mmult: size, pitch_A_nbelem, pitch_B_nbelem, pitch_C_nbelem are in number of elements
19 __global__ void mmult_kernel(int size, int nslices, double *A, size_t pitch_A_nbelem, double *B, size_t pitch_B_nbelem, double *C, size_t pitch_C_nbelem)
20 {
21     // Set thread indexes
22     int i = blockDim.y*blockIdx.y+threadIdx.y;
23     int j = blockDim.x*blockIdx.x+threadIdx.x;
24     double res = 0.0;
25
26     // Make sure we don't write any value out of C because the grid can be larger
27     if(i < size/nslices && j < size)
28     {
29         for(int k=0; k<size; k++)
30         {
31             res += A[i*pitch_A_nbelem+k] * B[k*pitch_B_nbelem+j];
32         }
33
34         C[i*pitch_C_nbelem+j] += res;
35     }
36 }
37
38 void mmult_cuda(int size, int nslices, double *A, double *B, double *C)
39 {
40     double *d_A, *d_B, *d_C;
41     size_t pitch_A, pitch_B, pitch_C;
42     int widthinbytes = size*sizeof(double);
43     int height = size/nslices;
44     // ...
45 }
```

Locals

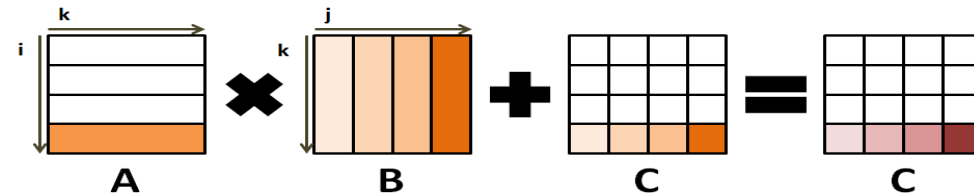
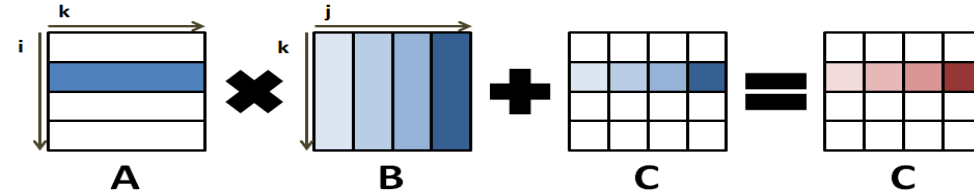
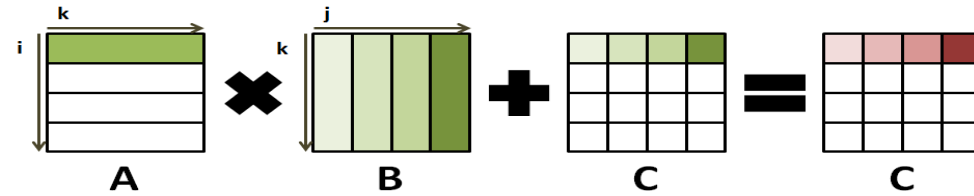
Variable Name	Value
A	0x10216200000
B	0x10216208000
C	0x10216210000
i	28
j	32
nslices	1
pitch_A_nbelem	512
pitch_B_nbelem	512
pitch_C_nbelem	512
res	0
size	64

# Example CUDA Debugging



# Matrix Multiplication Example

$$C = A \times B + C$$



Available in

`/users/hck06/DDT/Examples/GPU_debugging`



# Run

```
[.. Connect to Piz Daint]
module load daint-gpu
module load craype-accel-nvidia60
```

```
[... Copy Test Case from /users/hck06/DDT/Examples ]
```

```
cd DDT/Examples/GPU_debugging
make
salloc -p debug -C gpu --nodes=1 --time=00:30:00 --gres=gpu:1
srun -n 1 ./mmult2.exe
```

```
hck06@daint103:~/DDT/Examples/GPU_debugging> srun -n 1 ./mmult2.exe
0: Size of the matrices: 8192x8192
0: Initializing matrices...
0: Sending matrices...
0: Processing...
CUDA error
Rank 0 [Thu Sep 27 19:24:31 2018] [c8-1c0s13n0] application called MPI_Abort(MPI_COMM_WORLD, 77) - process 0
srun: error: nid03508: task 0: Aborted
srun: Terminating job step 9862146.1
hck06@daint103:~/DDT/Examples/GPU_debugging>
```

# Load DDT

```
18.1.1: module load daint-gpu  
       module load ddt
```

```
18.2.2: export PATH=$PATH:/users/hck06/DDT/18.2.2/bin
```

```
19.0  : export PATH=$PATH:/users/hck06/DDT/19.0/bin
```

# Compilation flags for debugging

```
make clean; make DEBUG=1
```

```
Compiler : -O0 -g  
nvcc : -g -G -O0 --cudart shared
```

Optional Flags  
Necessary for Memory  
Debugging only

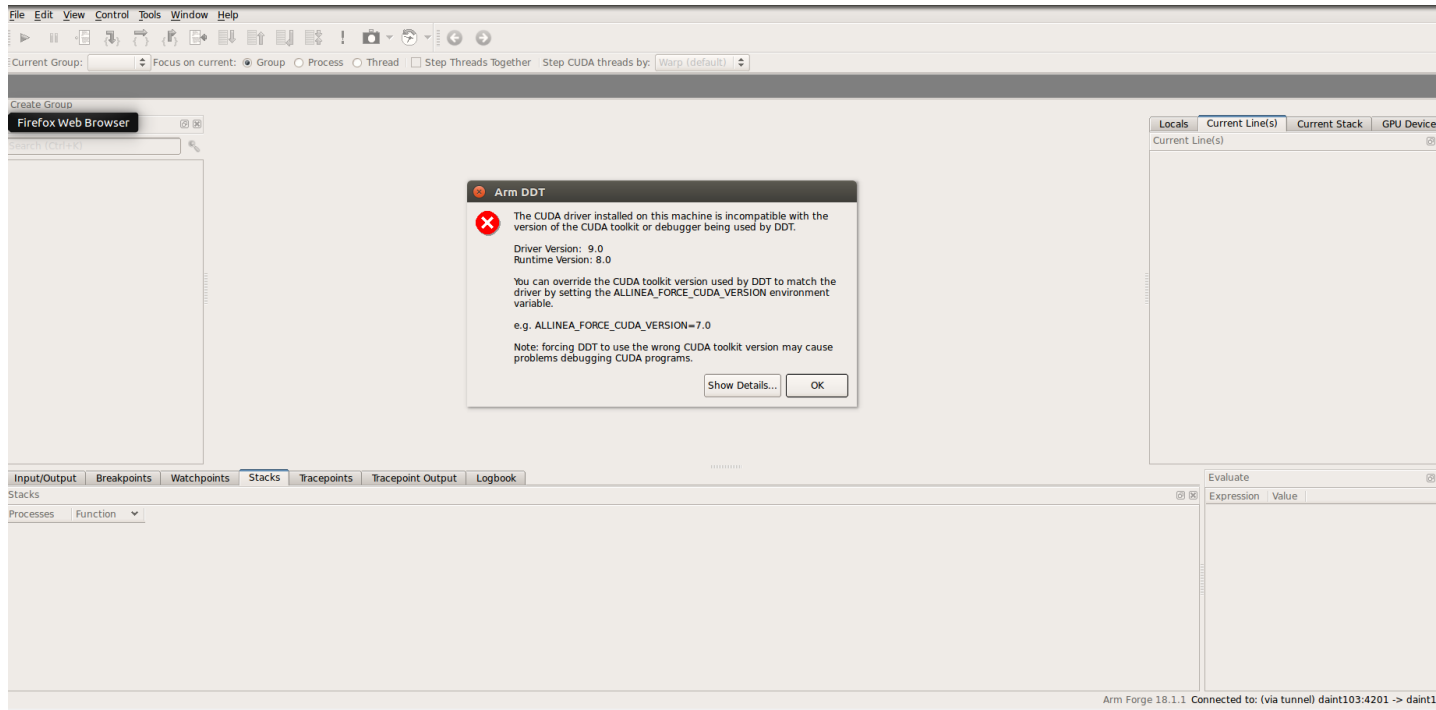
```
Linking : -L/apps/common/UES/SLES12/ddt/18.1.1/lib/64 -Wl,--allow-  
multiple-definition,--undefined=malloc,--undefined=_ZdaPv -  
ldmallocthcxx
```

For DDT 18.2.2, use `-L/users/hck06/DDT/18.2.2/lib/64`

For DDT 19.0, use `-L/users/hck06/DDT/19.0/lib/64`

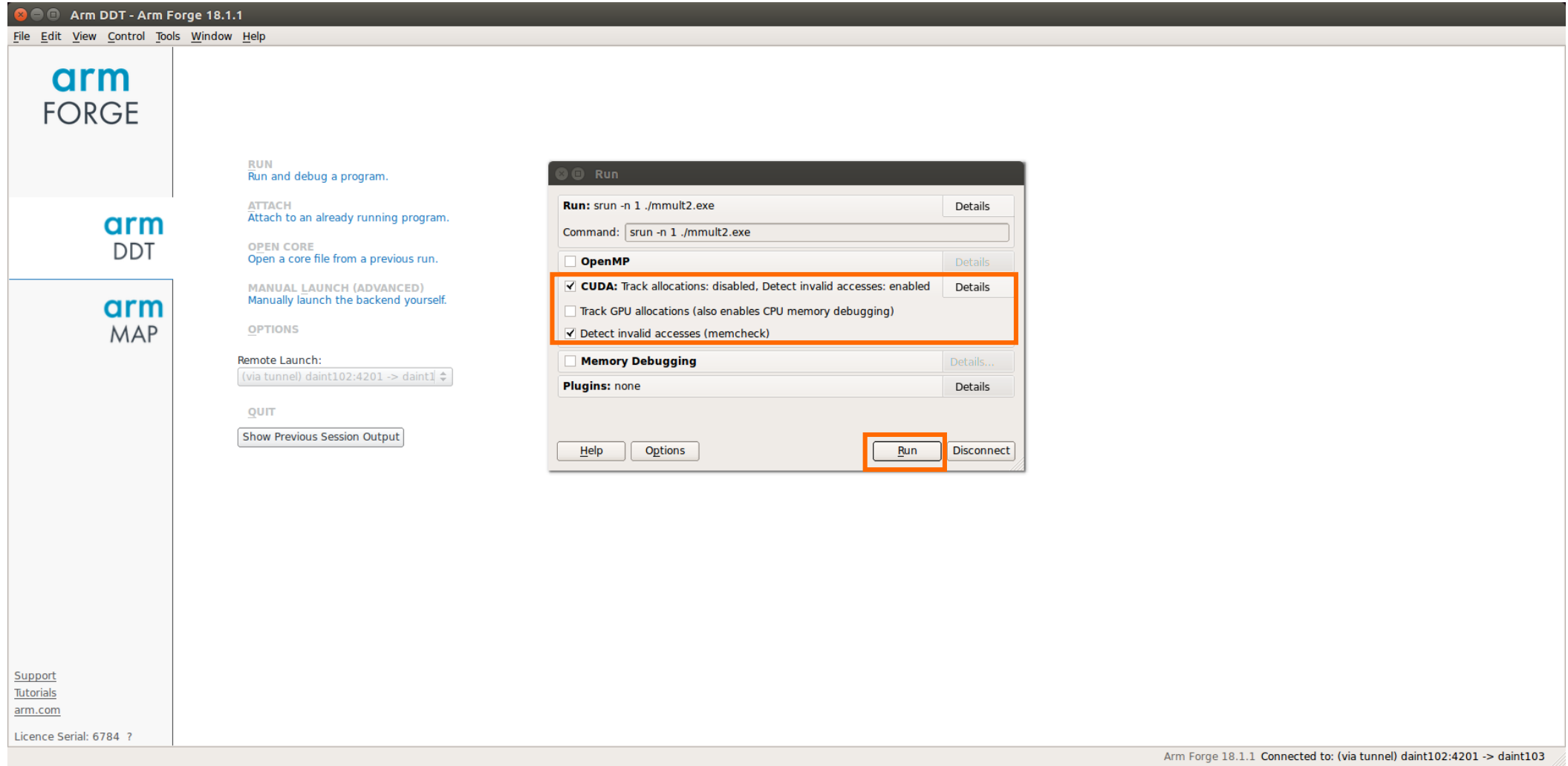
```
ddt --connect srun --reservation=hackathon -C gpu -n 1 ./mmult2.exe
```

# Troubleshooting – DDT and Cuda version conflict



**ALLINEA\_FORCE\_CUDA\_VERSION=8.0** ddt --connect srunk -n 1 ./mmult2.exe

# DDT Launch



# You have an error !

File Edit View Control Tools Window Help

Focus on current: ☒ Process ☐ Thread ☐ Step Threads Together Step CUDA threads by: Warp (default)

Threads 1 2 3 K4

CUDA Threads (mmult\_kernel) Block 0 28 Thread 0 0 0 Go Grid size: 1x64x1 Block size: 64x1x1

Project Files

Search (Ctrl+K)

Application Code

Sources

mmult2.c

mmult2.cu

mmult\_cuda(int size, int nslices, double \*A, double \*B, double \*C, size\_t pitch\_A\_nbelem, size\_t pitch\_B\_nbelem, size\_t pitch\_C\_nbelem)

External Code

```
18 // CUDA version of mmult: size, pitch_A_nbelem, pitch_B_nbelem, pitch_C_nbelem are in number of elements
19
20 __global__ void mmult_kernel(int size, int nslices, double *A, size_t pitch_A_nbelem, double *B, size_t pitch_B_nbelem, double *C, size_t pitch_C_nbelem)
21 {
22     // Set thread indexes
23     int i = blockIdx.y*blockDim.y+threadIdx.y;
24     int j = blockIdx.x*blockDim.x+threadIdx.x;
25     double res = 0.0;
26
27     // Make sure we don't write any value out of bounds
28     if(i < size/nslices && j < size)
29     {
30         for(int k=0; k<size; k++)
31         {
32             res += A[i*pitch_A_nbelem+k] * B[k*pitch_B_nbelem+j];
33         }
34         C[i*pitch_C_nbelem+j] += res;
35     }
36 }
37
38 void mmult_cuda(int size, int nslices, double *A, double *B, double *C, size_t pitch_A_nbelem, size_t pitch_B_nbelem, size_t pitch_C_nbelem)
39 {
40     double *d_A, *d_B, *d_C;
41     size_t pitch_A, pitch_B, pitch_C;
42     int widthinbytes = size*sizeof(double);
43     int height = size/nslices;
44     // ...
45 }
```

Program Stopped

Process 0:

Kernel 1 Thread <<<(0,28,0).(0,0,0)>>> stopped in mmult\_kernel (mmult2.cu:32) with signal CUDA\_EXCEPTION\_1 (Lane Illegal Address). Your program will probably be terminated if you continue. You can use the stack controls to see what the process was doing at the time.

☒ Always show this window for signals

Continue Pause

Locals Current Line(s) Current Stack GPU Devices

Current Line(s)

Variable Name	Value
A	0x10216200000
B	0x10216208000
i	28
j	0
k	0
pitch_A_nbelem	512
pitch_B_nbelem	512
res	0

Type: none selected

Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output Logbook

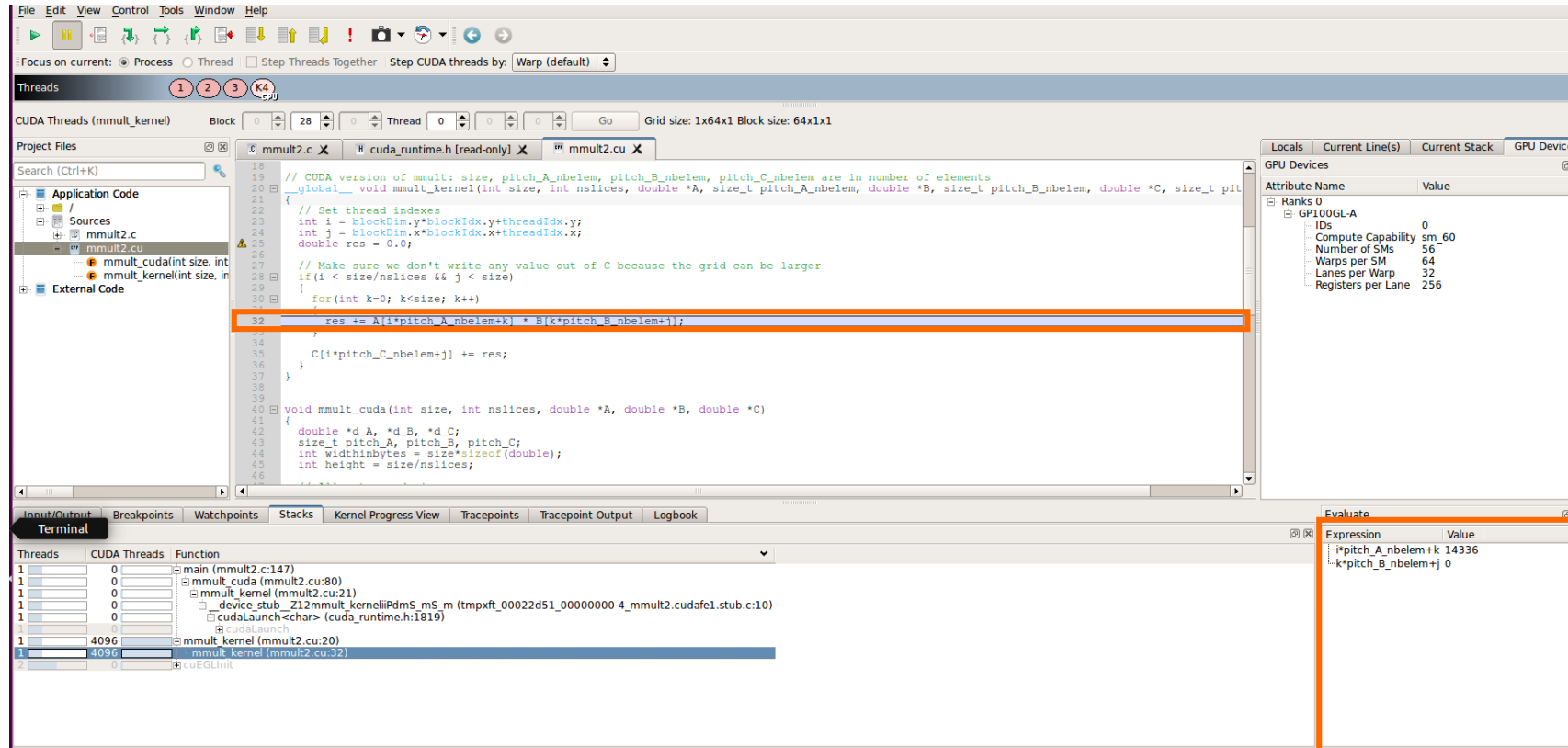
Stacks

Threads	CUDA Threads	Function
1	0	main (mmult2.c:147)
1	4096	mmult_kernel (mmult2.cu:20)
1	4096	mmult_kernel (mmult2.cu:32)
2	0	cuEGLInit

Evaluate

Expression	Value
------------	-------

# Debug - Matrix Multiplication: $C = A \times B + C$

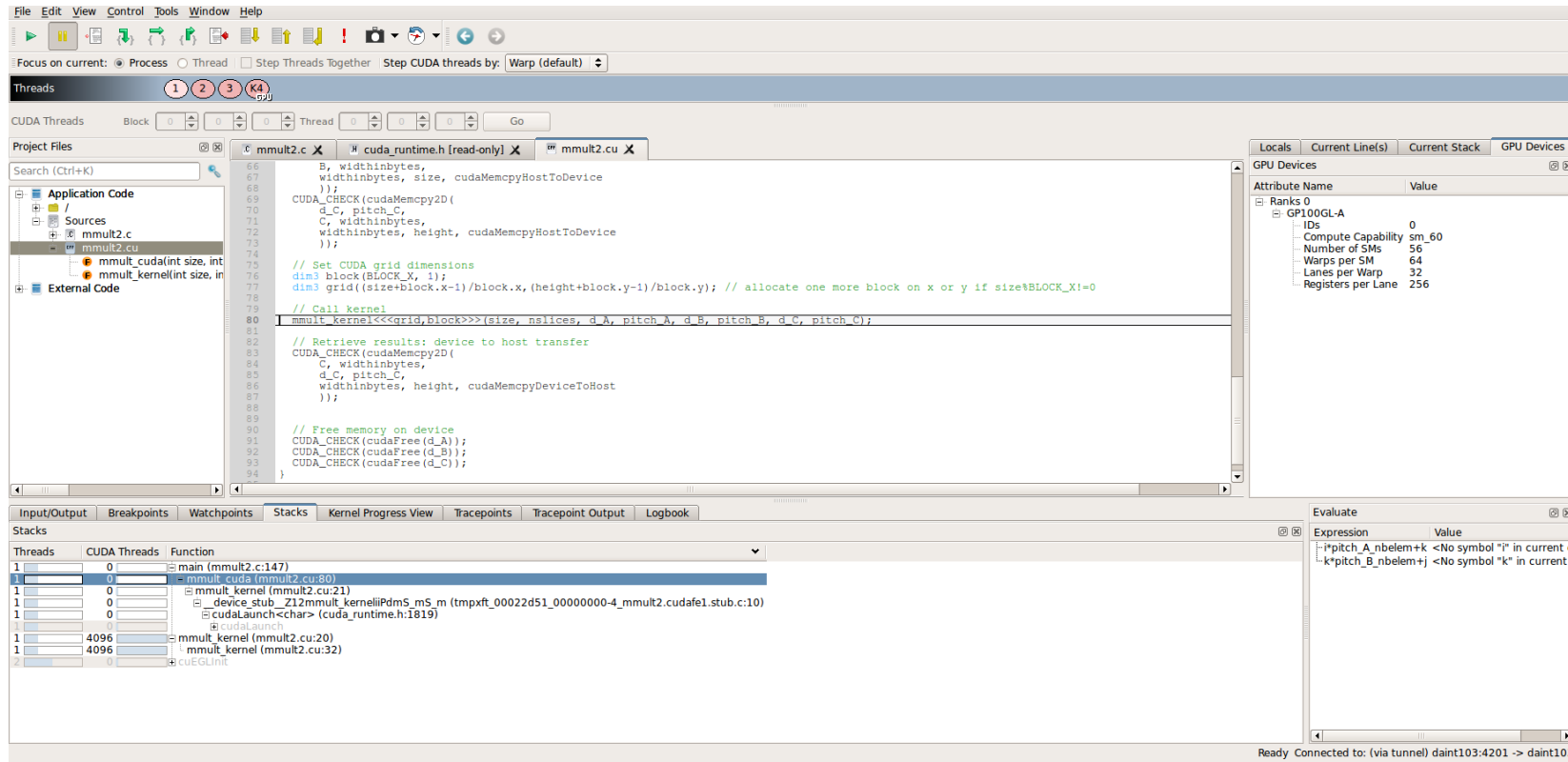


Line where the problem comes from is highlighted : 32

Error message : Lane illegal address -> Problem with offsets to access elements

Select the expressions in the source code viewer; Right click; Evaluate

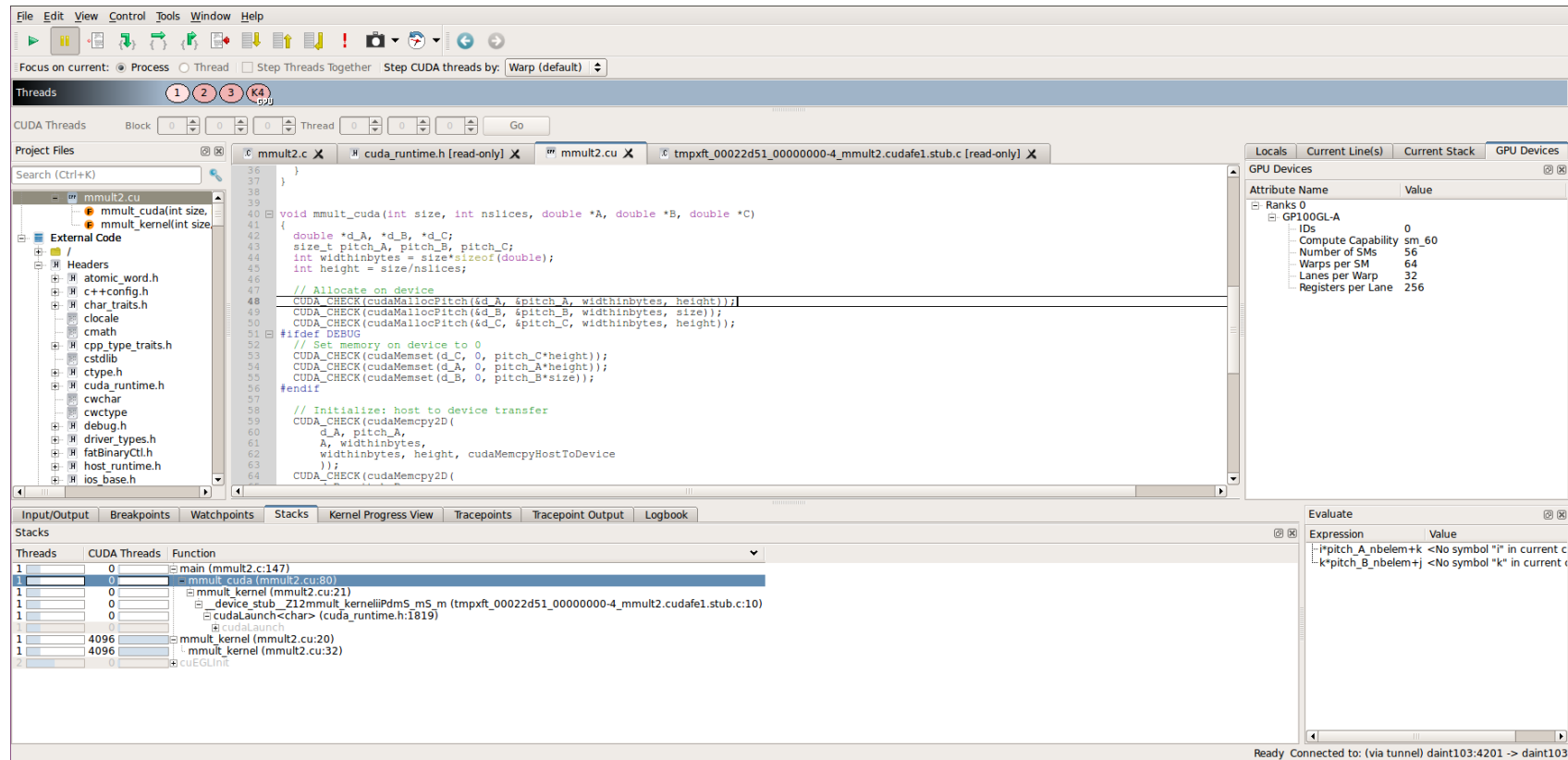
# Debug - Matrix Multiplication: $C = A \times B + C$



Use the stack to see which parameters have been given to the Cuda Kernel  
Next step : Identify where the arrays and parameters are coming from



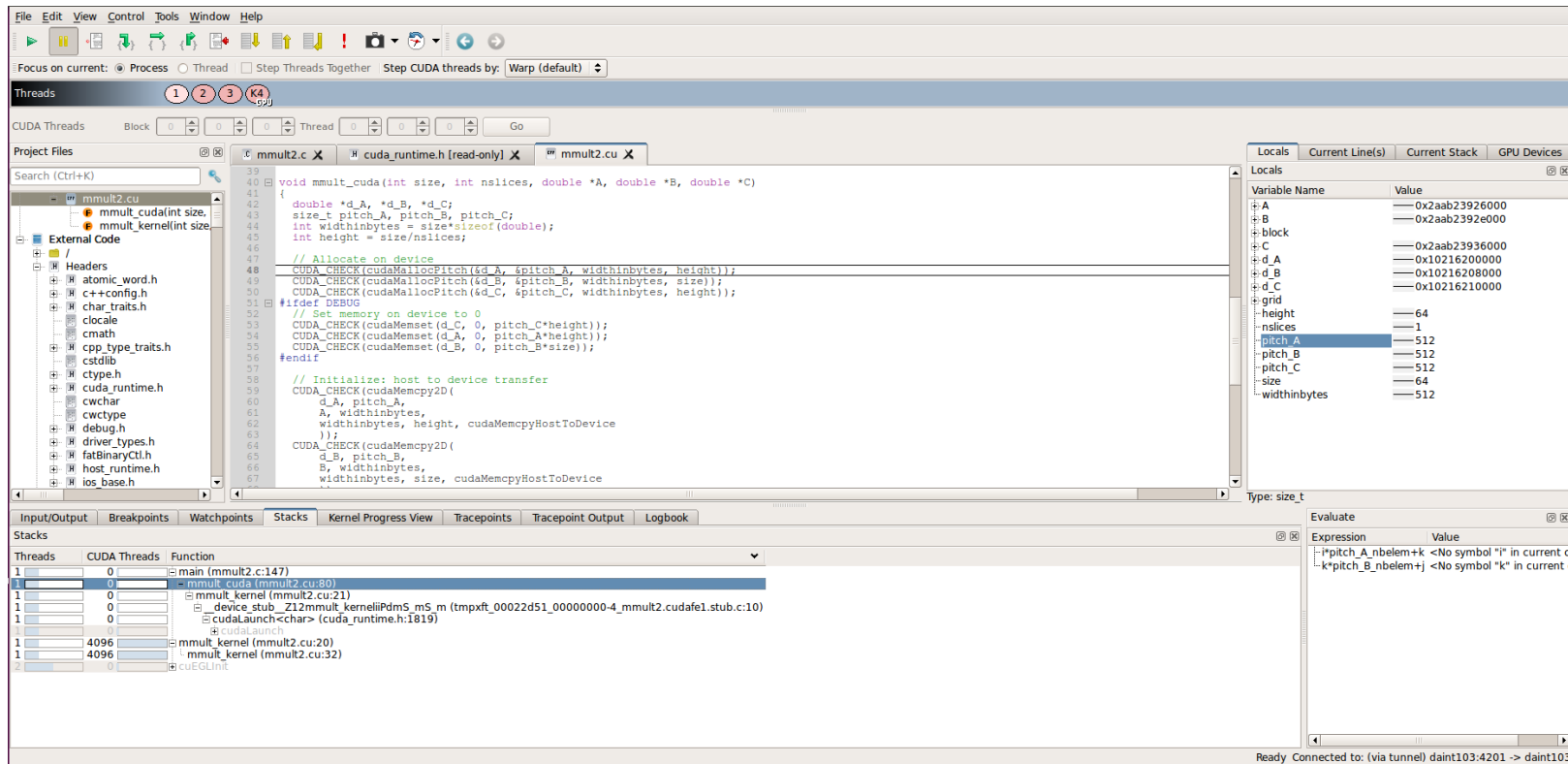
# Debug - Matrix Multiplication: $C = A \times B + C$



Pitch allocation -> pitch returned value in bytes not in number of elements

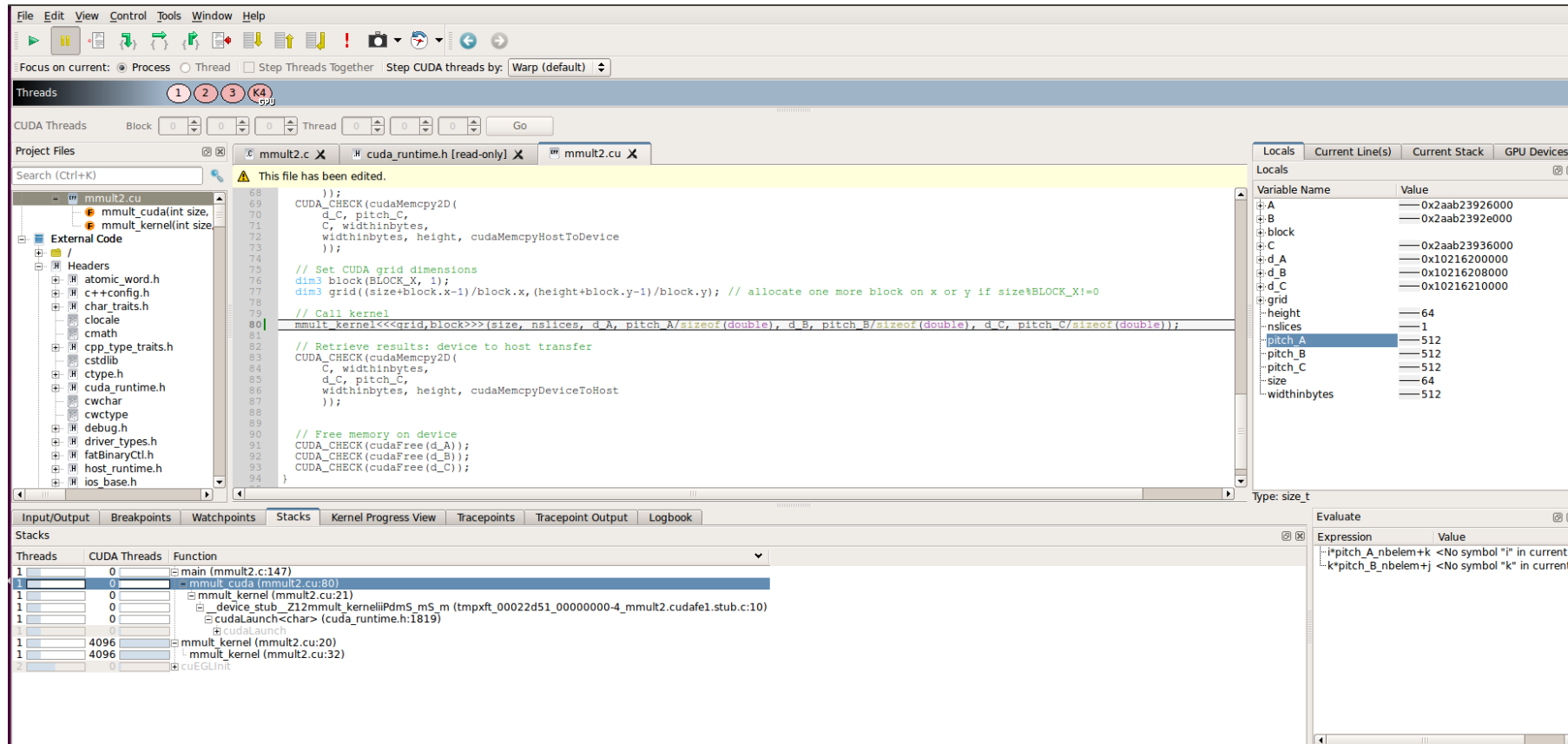
([Link to cudaMallocPitch documentation](#))

# Debug - Matrix Multiplication: $C = A \times B + C$



Pitch allocation -> pitch returned value in bytes not in number of elements  
From the Locals panel we can see that pitch\_A value is 512 bytes  
Because it is double, the number of elements for a “pitch” is  $512/8 = 64$   
Next action : Pass the number of elements to the Cuda kernel

# Debug - Matrix Multiplication: $C = A \times B + C$

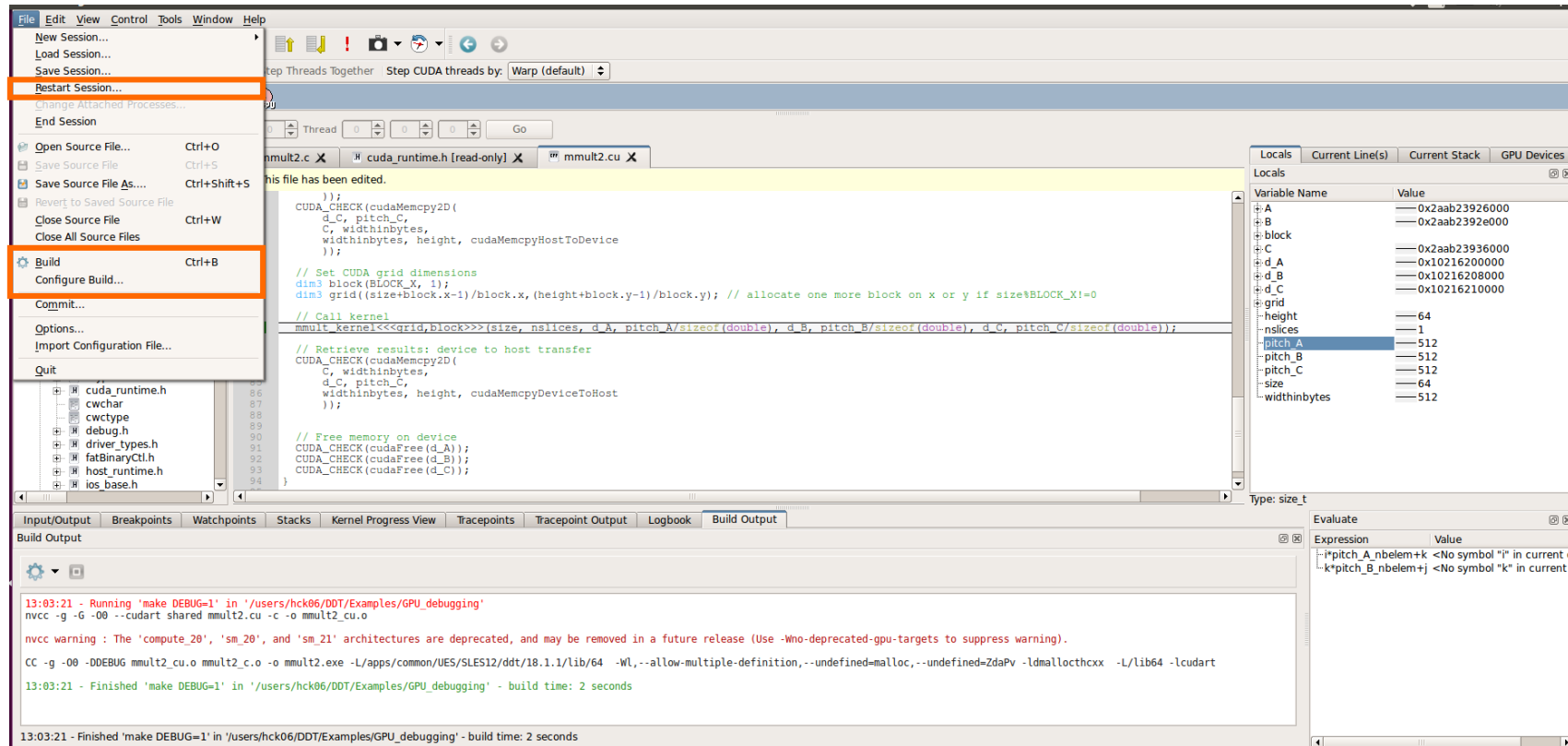


Modify source code

Save the changes (CTRL + s)

Next action : Recompile and launch new debugging session to see if we fixed the bug

# Debug - Matrix Multiplication: $C = A \times B + C$

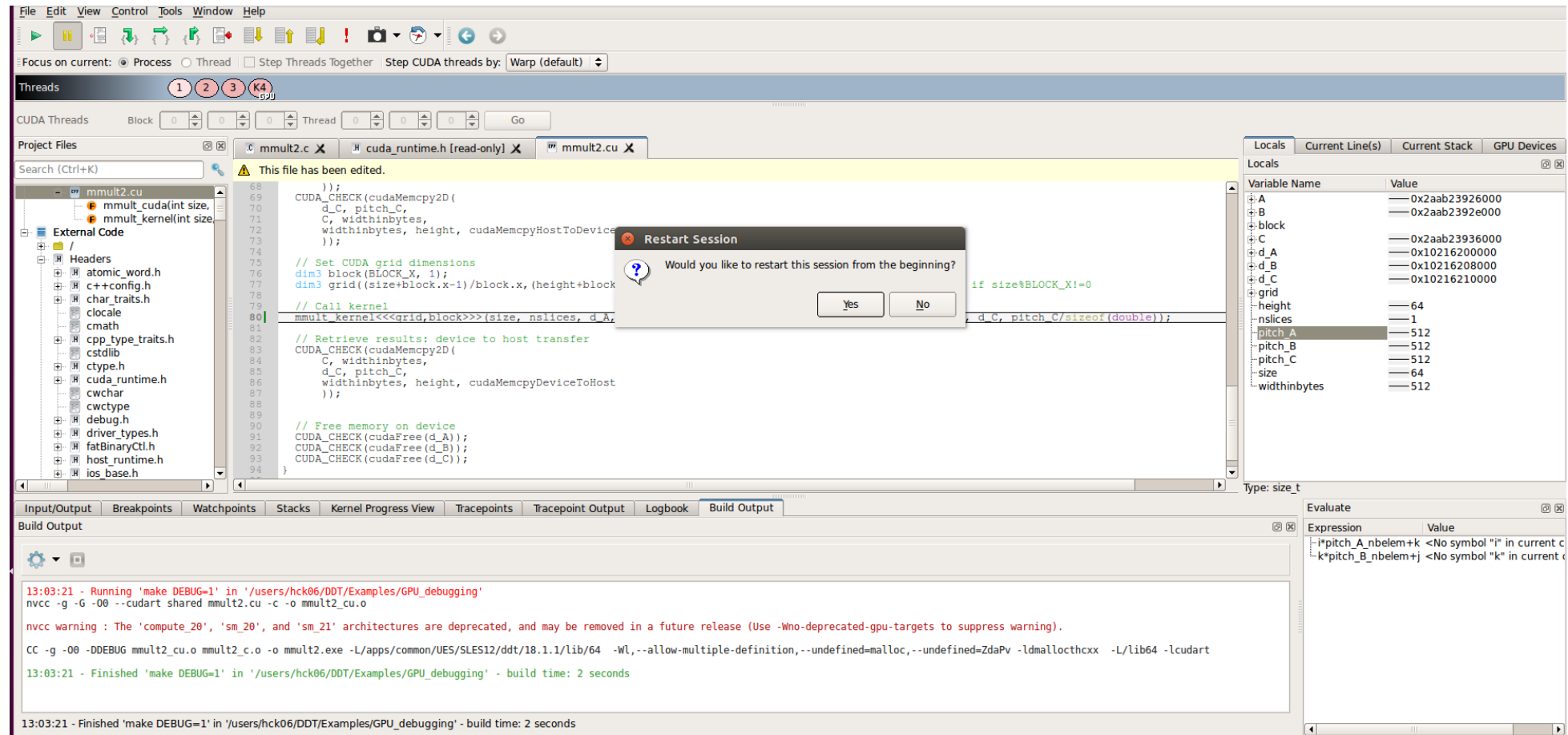


File / Configure Build ... [make DEBUG=1]

File / Build

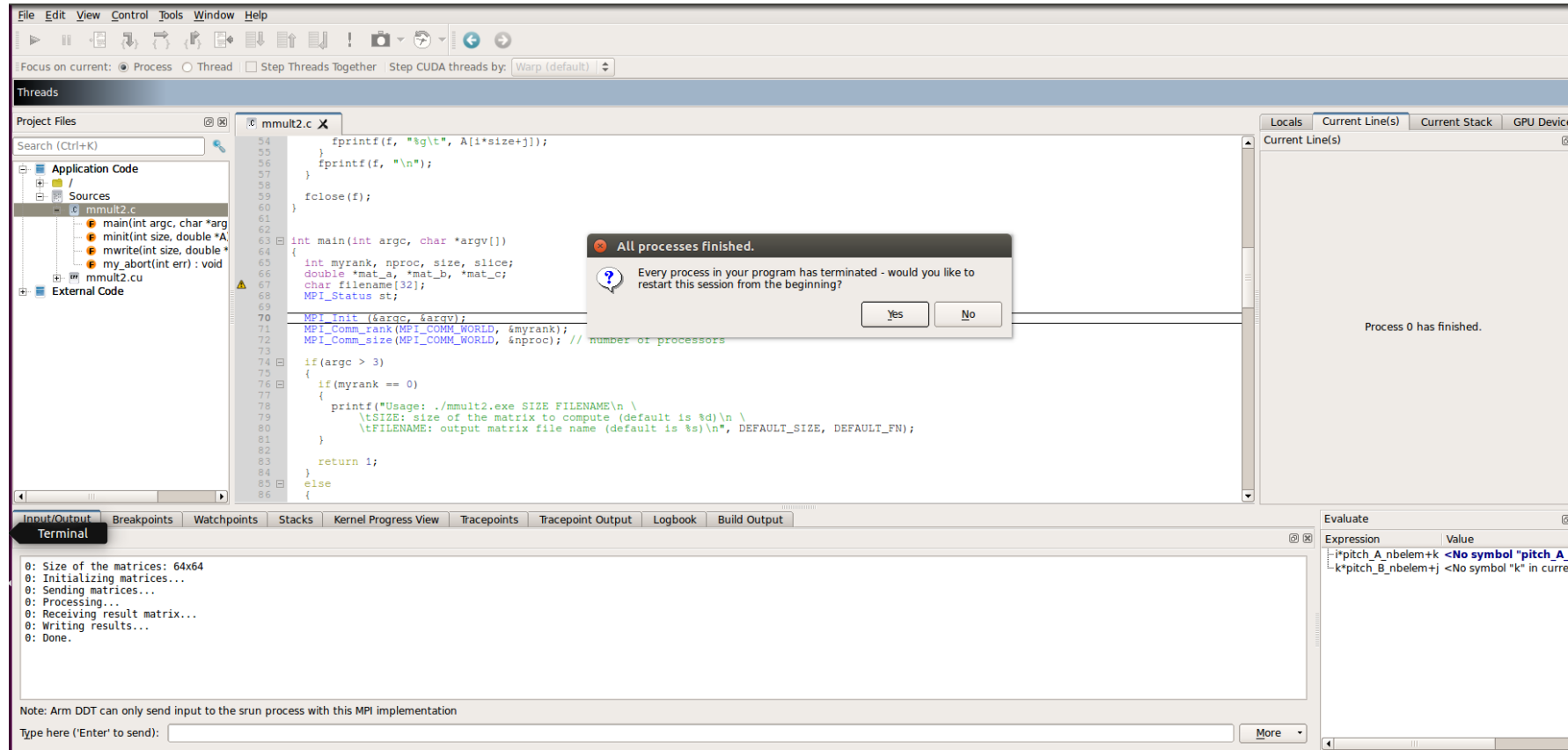
Have a look to the Build Output to make sure that compilation ends successfully

# Debug - Matrix Multiplication: $C = A \times B + C$



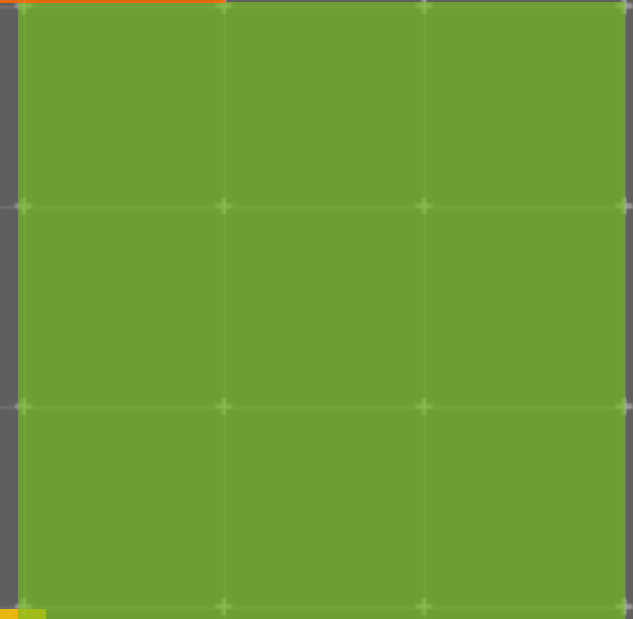
Restart debugging session [File / ...]

# Debug - Matrix Multiplication: $C = A \times B + C$



Play / Run -> program runs to completion -> Congratulation !

# PGI OpenACC



# PGI OpenACC

Add `-g` everywhere

`-O0` and Switch off optimization flags advised (at least for Host)

Add debug to the `-ta` flag

Example `-ta=nvidia, debug`



# Memory debugging

# Heap debugging options available

## Fast

### basic

- Detect invalid pointers passed to memory functions (e.g. malloc, free, ALLOCATE, DEALLOCATE,...)

### check-fence

- Check the end of an allocation has not been overwritten when it is freed.

### free-protect

- Protect freed memory (using hardware memory protection) so subsequent read/writes cause a fatal error.

### Added goodness

- Memory usage, statistics, etc.

## Balanced

### free-blank

- Overwrite the bytes of freed memory with a known value.

### alloc-blank

- Initialise the bytes of new allocations with a known value.

### check-heap

- Check for heap corruption (e.g. due to writes to invalid memory addresses).

### realloc-copy

- Always copy data to a new pointer when re-allocating a memory allocation (e.g. due to realloc)

## Thorough

### check-blank

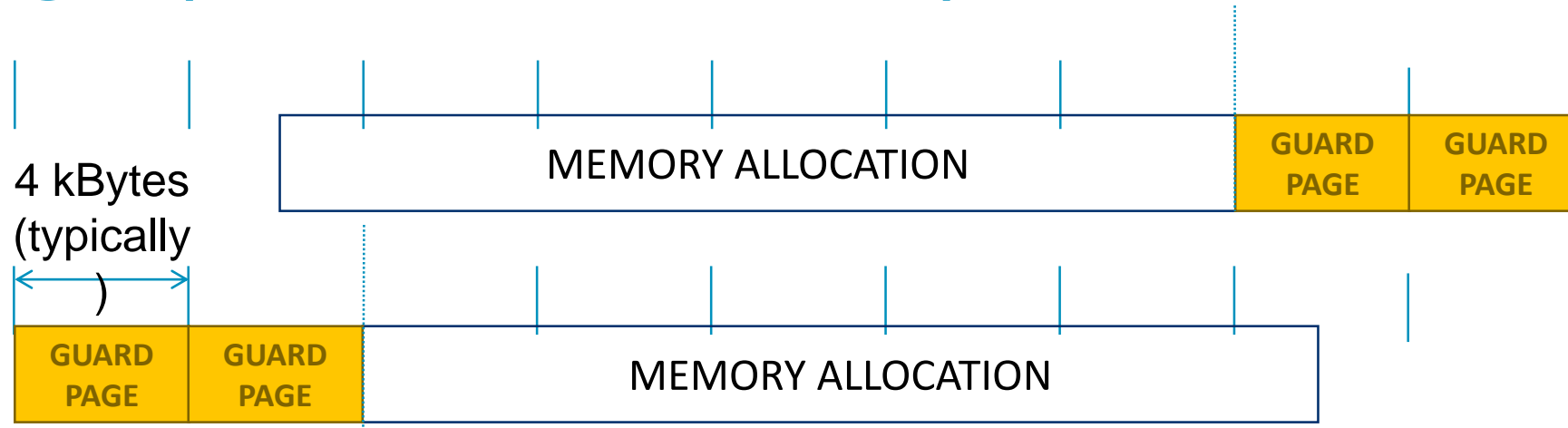
- Check to see if space that was blanked when a pointer was allocated/freed has been overwritten.

### check-funcs

- Check the arguments of addition functions (mostly string operations) for invalid pointers.

*See user-guide:  
Chapter 12.3.2*

# Guard pages (aka “Electric Fences”)



- **A powerful feature...:**
  - Forbids read/write on guard pages throughout the whole execution  
*(because it overrides C Standard Memory Management library)*
- **... to be used carefully:**
  - Kernel limitation: up to 32k guard pages max ( “mprotect fails” error)
  - Beware the additional memory usage cost

# Compilation flags for debugging

```
make clean; make DEBUG=1
```

```
Compiler : -O0 -g  
nvcc : -g -G -O0 --cudart shared
```

Optional Flags  
Necessary for Memory  
Debugging only

```
Linking : -L/apps/common/UES/SLES12/ddt/18.1.1/lib/64 -Wl,--  
allow-multiple-definition,--undefined=malloc,--  
undefined=_ZdaPv -ldmallocthcxx
```

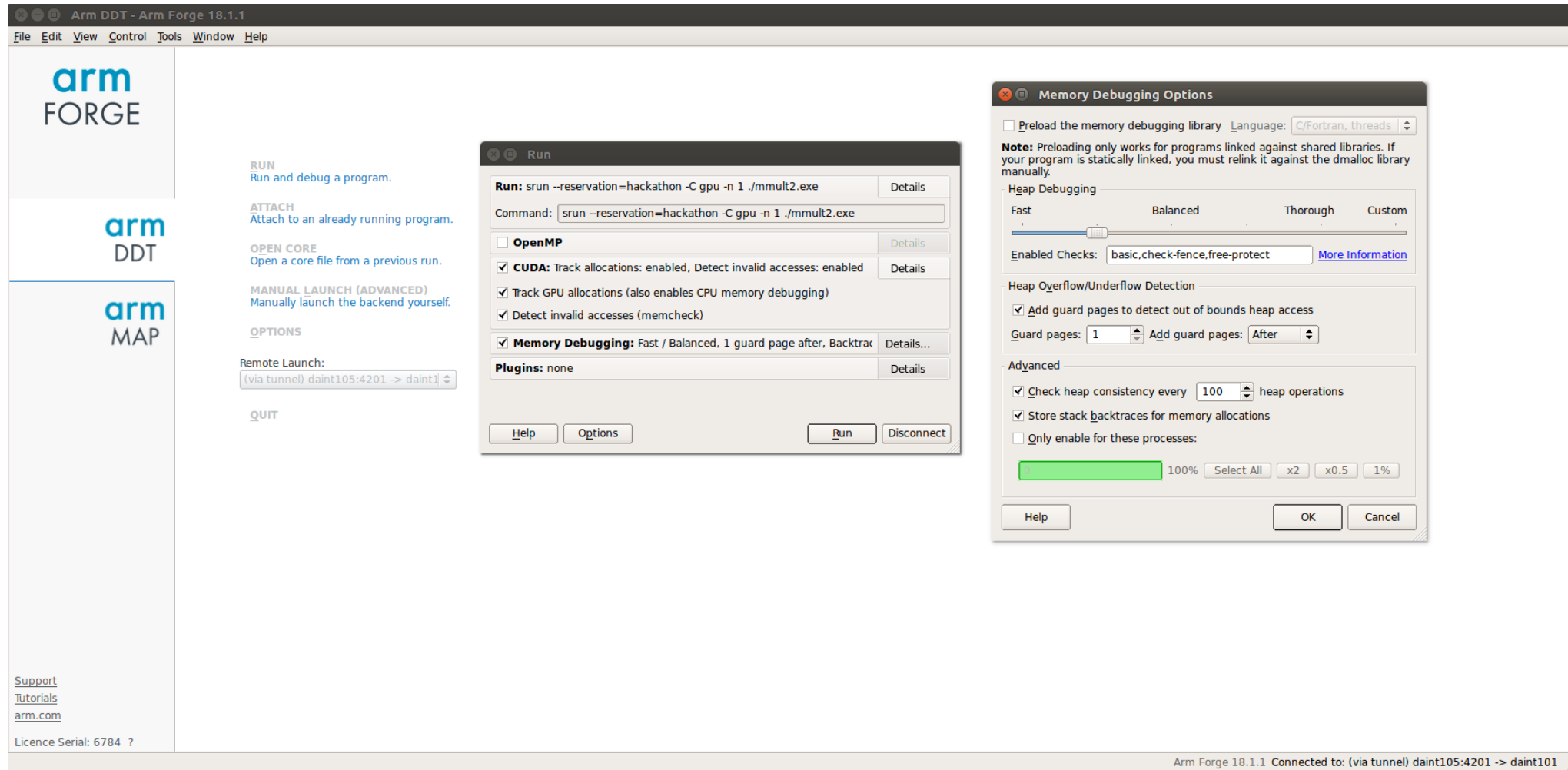
For DDT 18.2.2, use `-L/users/hck06/DDT/18.2.2/lib/64`

For DDT 19.0, use `-L/users/hck06/DDT/19.0/lib/64`

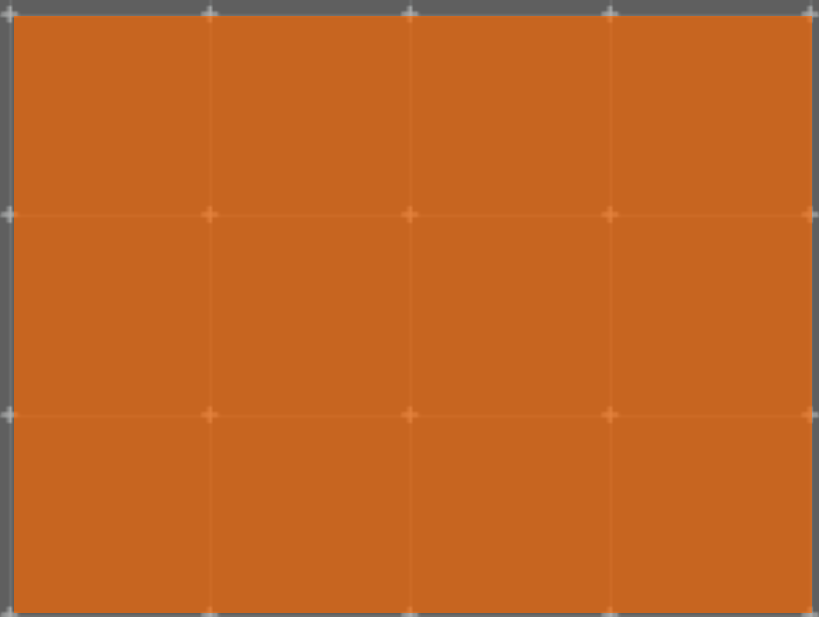
```
[Allocate resources]
```

```
ddt --connect srun -n 1 ./mmult2.exe
```

# Memory debugging



# Contact Support



# Issues with DDT ? Our support team is here to help !

For any question during the Hackathon :

[conrad.hillairet@arm.com](mailto:conrad.hillairet@arm.com)

or Conrad Hillairet in the EuroHack18 slack

For any question after the Hackathon :

[Support-hpc-sw@arm.com](mailto:Support-hpc-sw@arm.com)

CC : [conrad.hillairet@arm.com](mailto:conrad.hillairet@arm.com)

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

**arm**