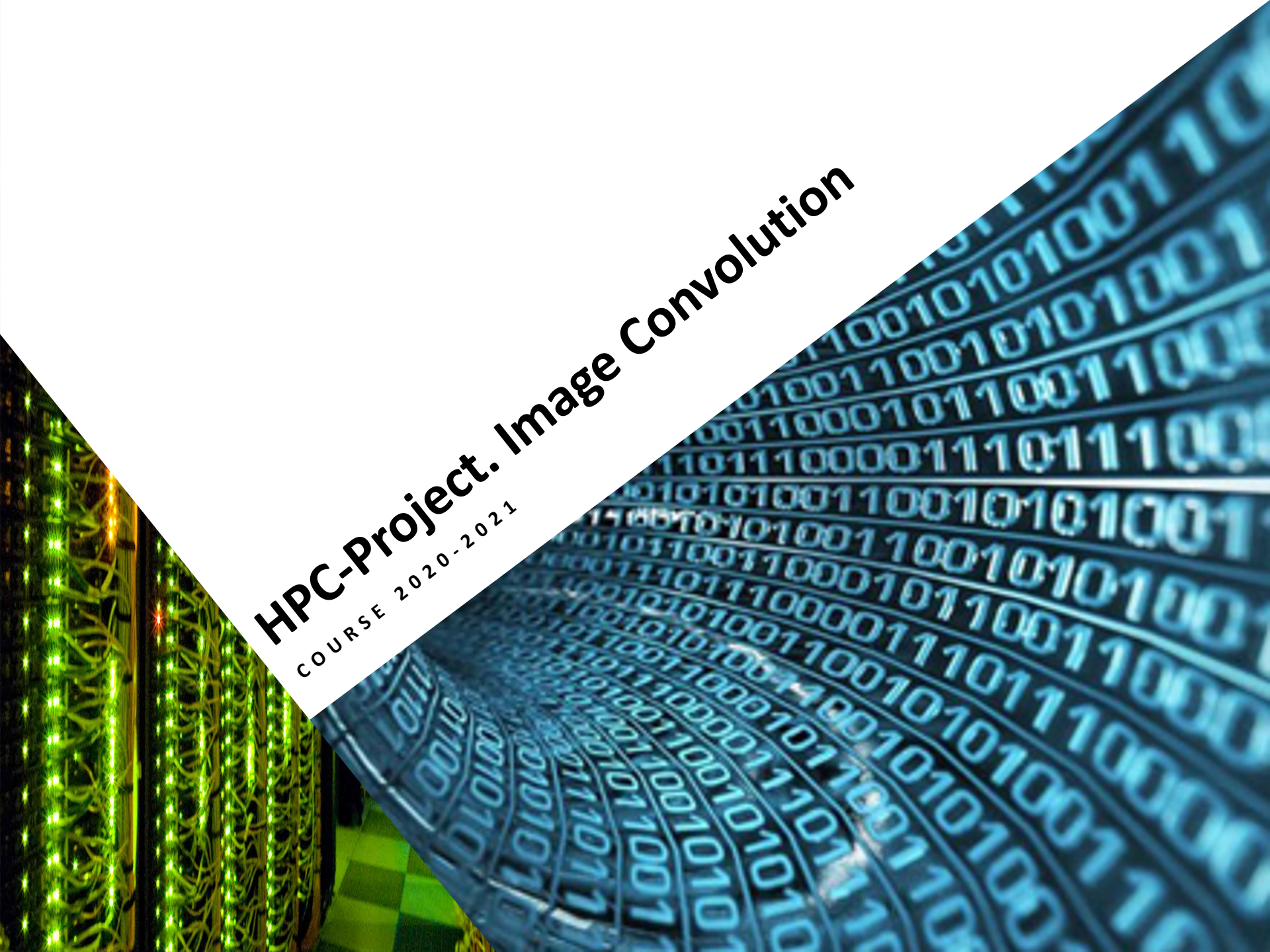
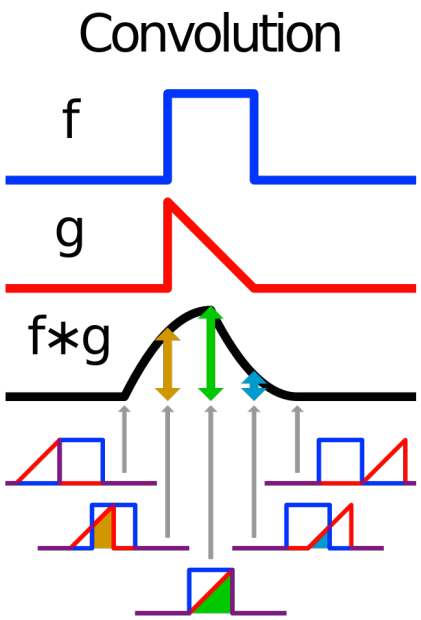
HPC-Project. Image Convolution

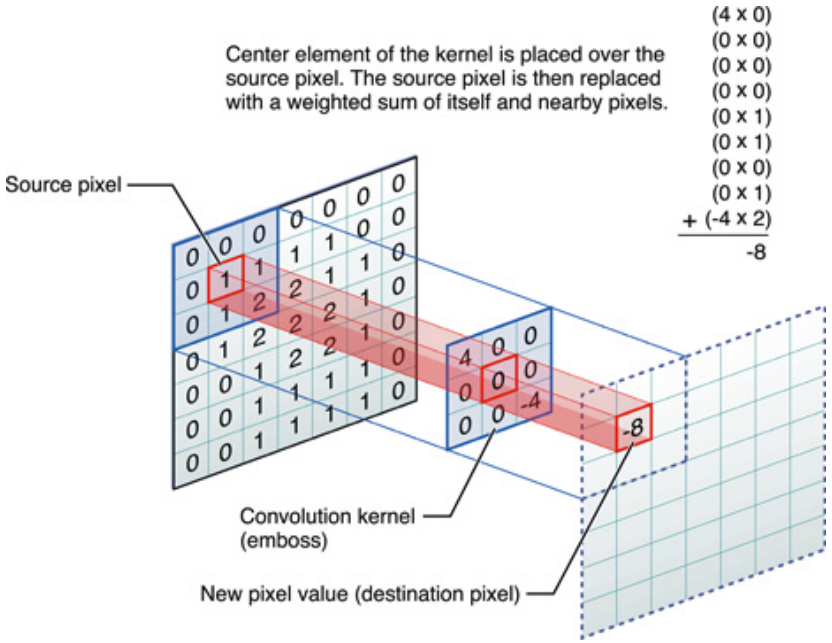
COURSE 2020-2021



HPC PROJECT – Image convolution (1)

We want to implement a parallel version for the image convolution algorithm. the convolution algorithm involves the product and subsequent sum up of a portion of the image values with a kernel matrix. The results of the convolution is a new image of the same size modified based on the kernel matrix values.





HPC PROJECT – Image convolution (2)

Varying the composition of the kernel, it is possible to obtain different results.

HPC PROJECT – Image convolution (3)

The problem size is directly related to the **Image and Kernel Sizes**. With the aim to be able to compare the results all groups should pass the following testbed:

Testbed composition - **Images Sizes**		
im01.ppm	1MB	tux
im02.ppm	2.2 MB	mandelbrot 6000x4000 pixels
im03.ppm	5.3 MB	shrek
im04.ppm	218.5 MB	mandelbrot 12000x8000
im05.ppm	873.9 MB	mandelbrot 24000x12000
im06.ppm	2.62GB	mandelbrot 48000x32000
Testbed composition – **kernel Sizes**		
Kernel_Edge.txt	3x3	Edge detection
kernel5x5_Sharpen.txt	5x5	Sharpening filter
Kernel25x25_random.txt	25x25	Randomly generated
Kernel49x49_random.txt	49x49	Randomly generated
Kernel99x99_random.txt	99x99	Randomly generated

HPC PROJECT – Image convolution (4)

Considerations

- Source code for serial version is accessible through the virtual campus or at the cluster:
  - /share/apps/files/convolutional/code/convolution.c

- For images bigger than **500MB** and the **99x99** kernel matrix you should read the image in different sections. Execute the convolution program with the parameter partitions=2.

```
$convolution image_file kernel_file result_file partitions
```

- For images bigger than **2GB** some operating systems with 32bits didn't open the file. Using the complier option -D_FILE_OFFSET_BITS=64 you can open them.

```
$gcc convolution.c -D_FILE_OFFSET_BITS=64 -o convolution
```

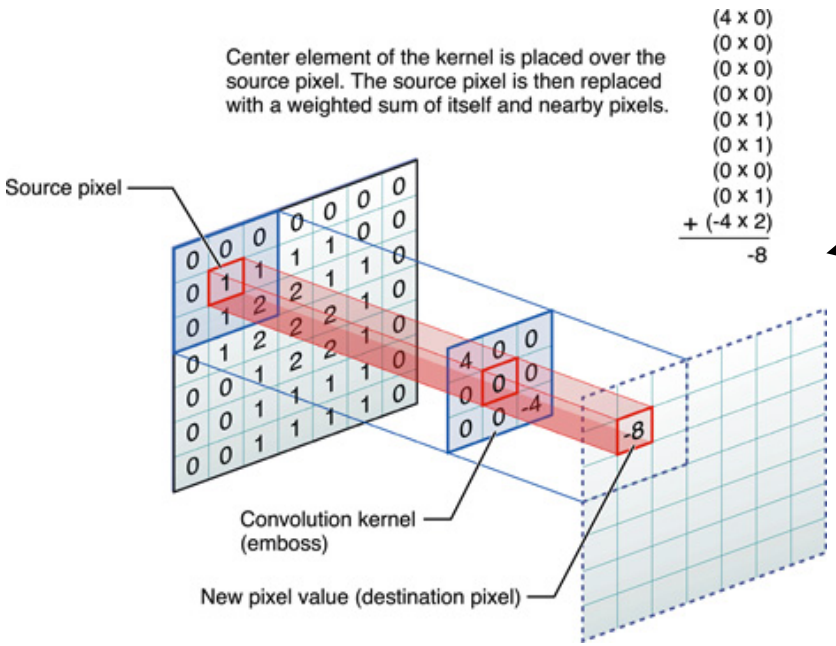
- Don't execute any program in the front-end of the cluster without using the system queue: **qsub** for parallel programs and **qlogin** for single machine programs.

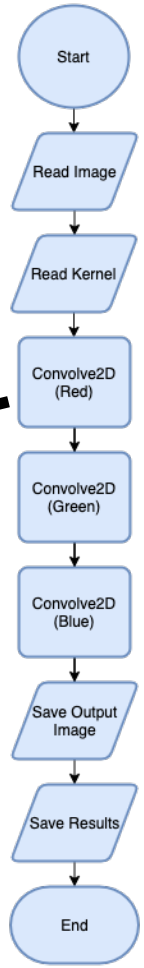
```
$qlogin -q high.q -pe smp 4
```

HPC PROJECT – Image convolution (5)

Considerations

- Serial Code Overview





HPC PROJECT – Image convolution (6)

Considerations

- Don't copy the original and/or resulting images to your home.

  Shared Images: /share/apps/files/convolution/images/<<source_image.ppm>>

  Shared Kernels: /share/apps/files/convolution/kernel/<<kernel_file.txt>>

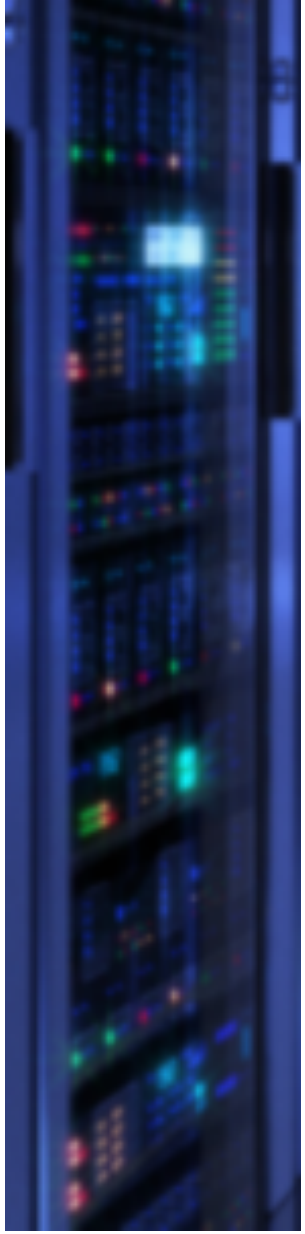
- Save resulting images locally to the execution node.

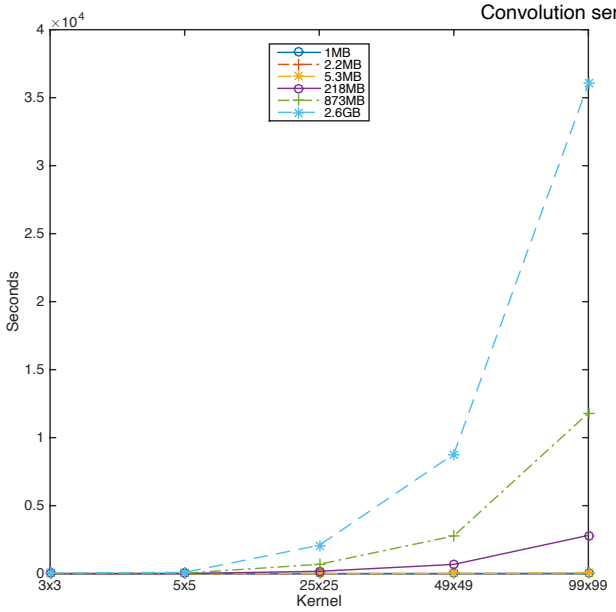
  Local partition: /state/partition1/<<resulting_image.ppm>>

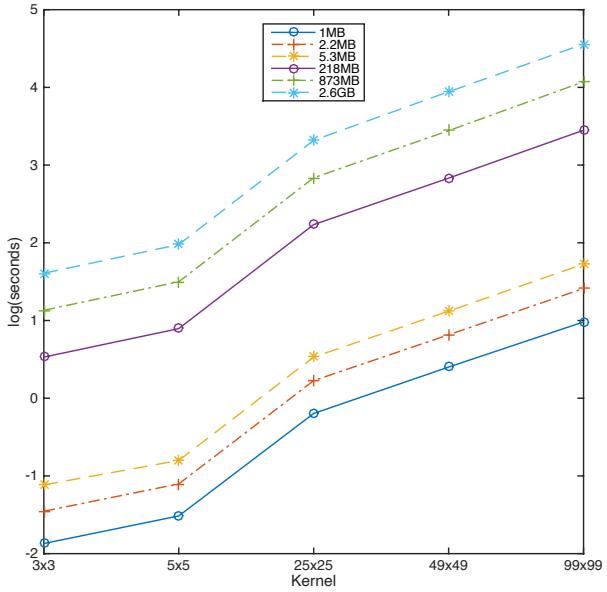
  IMPORTANT: **Delete** all generated images after all experimentation is done.

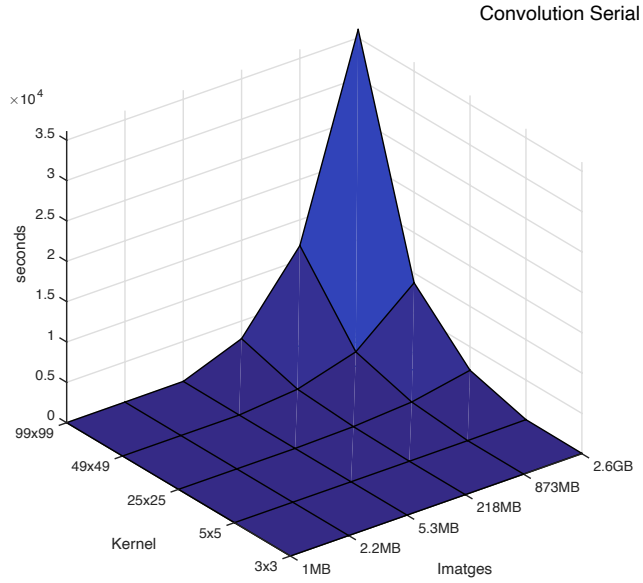
```
./c /share/apps/files/convolution/images/im05.ppm
/share/apps/files/convolution/kernel/kernel3x3_Edge.txt /state/partition1/prova.ppm
1 > ../results/im05kernel3x3_Edge.txt
```

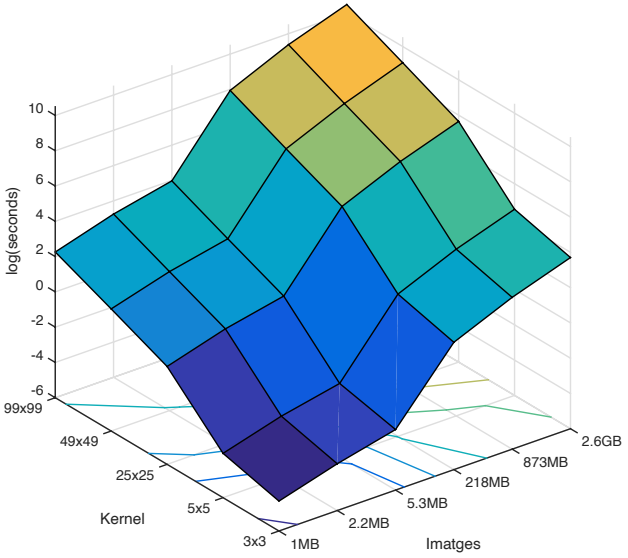
- The results must be graphically represented:
  - Graphics should be correctly formatted: title, axes tags, legend, etc.
  - The chart type should be correctly chosen (Line, bar, pie, etc.)
  - The chart should be represented correctly (Style, Color, Marker, etc)
  - The axes should be correctly scaled. (log, loglog, etc)
  - The graphics should be explained and the results justified.




Convolution serial time by kernel size
×10^4
Seconds
Kernel
3x3
5x5
25x25
49x49
99x99
1MB
2.2MB
5.3MB
218MB
873MB
2.6GB


log(seconds)
Kernel
3x3
5x5
25x25
49x49
99x99
1MB
2.2MB
5.3MB
218MB
873MB
2.6GB


Convolution Serial Elapsed time evolution
×10^4
seconds
Kernel
Imatges
3x3
5x5
25x25
49x49
99x99
1MB
2.2MB
5.3MB
218MB
873MB
2.6GB


log(seconds)
Kernel
Imatges
3x3
5x5
25x25
49x49
99x99
1MB
2.2MB
5.3MB
218MB
873MB
2.6GB

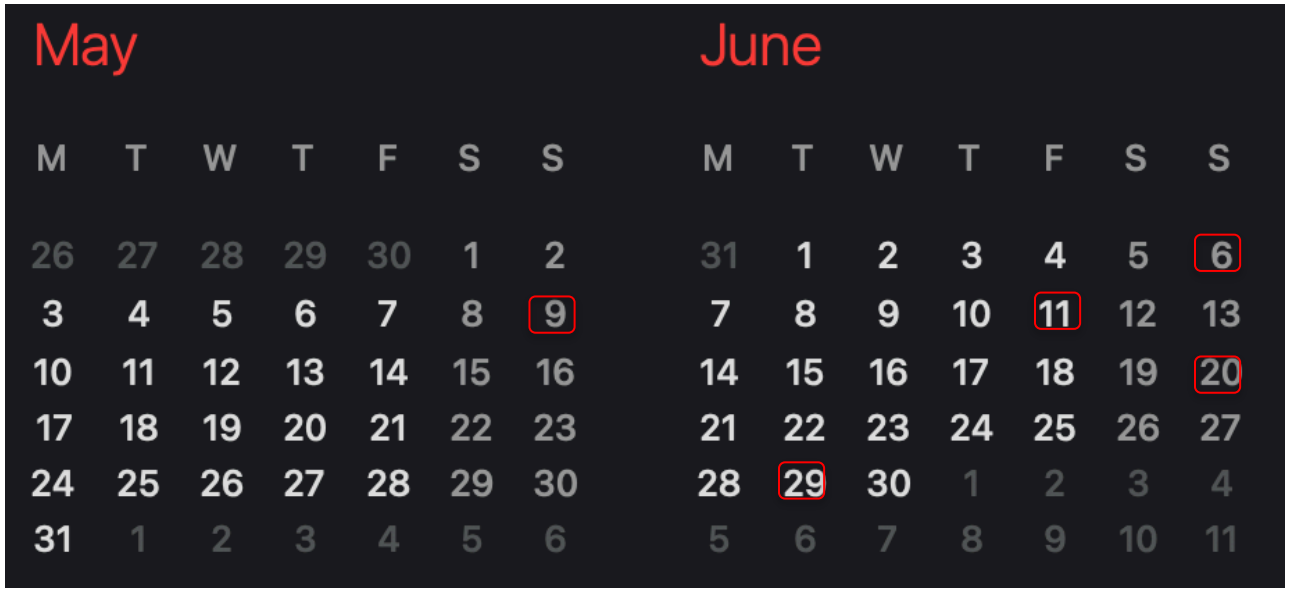
HPC PROJECT – Image convolution (7)

Considerations

- You can use any program for creating the graphical representation: Excel, R, gnuplot, MATLAB, etc. You don’t need sophisticated tools, you only need to configure it correctly.
- It is possible to use Libreoffice for opening PPM images. If you want you can use other image processing tools such us gimp, etc. However, some visualization tool don't accept negative values on the image. Take a look to the convolution source code for configure the output accordingly to the visualization tool.

HPC PROJECT – Deliveries

Delivery	Deadline
OpenMP solution (25%)	**09th of May**
MPI solution (25%)	**06th of June**
Hybrid solution (15%)	**11th of June**
Comparison of the results (15%)	**20th of June**
Oral Presentation	**29th of June**



HPC PROJECT – OpenMP Solution

Use OpenMP directives to parallelize the serial code according to the hardware and work decomposition model at node level.

- **Justify your decisions** according to the selected partitioning pattern, size of partitioning, data location, loops parallelization configuration, and so on.
- **Check the performance** obtained using different **thread scheduling policies** (static, dynamic, guided) and discuss about the results.
- Analyze **scalability** and **speedup** in relation to the number of threads and size problem. Consider on your discussion the effects of the communications in your solution.

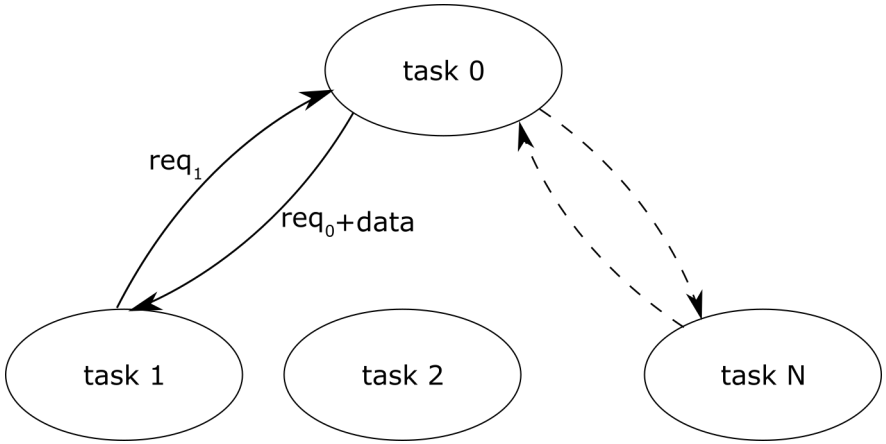
deadline: **9th of May** (OpenMP Program)

HPC PROJECT – MPI Solution (1)

Two different versions must be implemented:

a. ***Static mapping of tasks*** The region is divided into a fixed number of fragments. the allocation is done at the beginning of the execution of the program and cannot be modified.

b. ***Dynamic mapping of tasks*** In this case, the fragments will be mapped to different processors / cores as soon as they finish with the previous calculations.



HPC PROJECT – MPI Solution (2)

For both strategies, you must to analyse the performance of each implementation based on the following points:

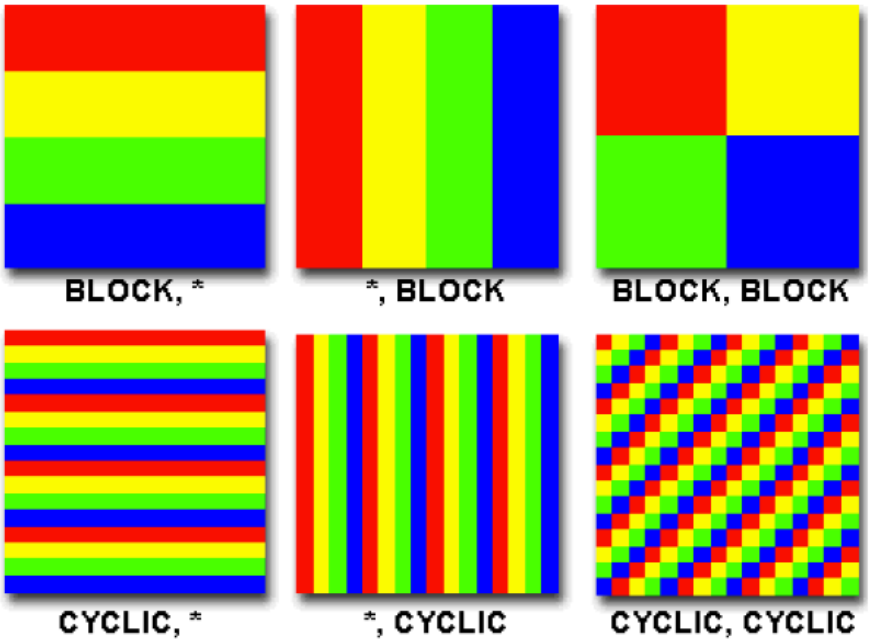
- **Justify your decisions** according to the selected partitioning pattern, size of partitioning, data location, communication directives, and so on.
- **Evaluate the Speedup** and **Efficiency** compared to the number of cores.
- Identify if there is **imbalanced**.
- Quantify the additional cost (**overhead**) of parallelization.

Explain the strengths and weaknesses of each implementation and discuss some scenarios where you think it would be more convenient to use and why. Consider on discussion the effects of heterogeneity in the execution nodes.

deadline: **06th of June** (MPI Program)

HPC PROJECT – MPI Solution (3)

Data Decomposition



HPC PROJECT – MPI Solution (4)

Dynamic mapping MPI structure

Master (Processor 0) sends **chunks** of data on demand, meanwhile there exist data to process, or receives the calculated data by the worker. (the worker is identified by Processor **status.MPI_SOURCE)**

```
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &nproc);
MPI_Comm_rank(MPI_COMM_WORLD, &iproc);

. . .

if (iproc == 0) {           // master code
    while (nchunks < total_chunks) {
        MPI_Recv(myRecvArr,1,MPI_INT,MPI_ANY_SOURCE,0,MPI_COMM_WORLD,&status);
              if (myRecvArr[0] == -1) { // worker wants more work
                  MPI_Send(mySendArr,1,MPI_INT,status.MPI_SOURCE,0,MPI_COMM_WORLD);
                  nchunks++;
                  . . .
        }
               else {    // tell worker there isn't any more chunks
                          mySendArr[0] = -1;

             MPI_Send(mySendArr,1,MPI_INT,status.MPI_SOURCE,0,MPI_COMM_WORLD);
               }
             }
             else if (myRecvArr[0] == -2) {          // worker wants to send finished work

MPI_Recv(myRecvArr,WIDTH*HEIGHT+1,MPI_INT,status.MPI_SOURCE,0,MPI_COMM_WORLD,&status);
                  . . .
       }
}
```

HPC PROJECT – MPI Solution (5)

Dynamic mapping MPI structure

Slave processes (Processor i!=0) sends ask for a chunk of data on demand, and check the response. When the Slave receives a new chunk compute it and send back the results to the master, otherwise the Slave finish his work.

```
. . .

else {        // worker code
              while (loop) {
             // ask master for work
             mySendArr[0] = -1;
             MPI_Send(mySendArr,1,MPI_INT,0,0,MPI_COMM_WORLD);
             // recv response (starting number or -1)
             MPI_Recv(myRecvArr,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);
             if (myRecvArr[0] == -1) {    // -1 means no more
                           break;                                          // break
the loop
             }
             else {
                           myJobStart = myRecvArr[0];
              // do computation
                                        . . .
                           // tell master work is done and ready to send
              mySendArr[0] = -2;
              MPI_Send(mySendArr,1,MPI_INT,0,0,MPI_COMM_WORLD);
              // send work
              mySendArr[0] = myJobStart;
              for (i = 1; i < BLOCK_WIDTH * BLOCK_HEIGHT + 1; i++) {
                     mySendArr[i] = pixels[i-1];
                 }
              MPI_Send(mySendArr,WIDTH*HEIGHT+1,MPI_INT,0,0,MPI_COMM_WORLD);
                   } // end conditional
       } // end while
} // end conditional
```

HPC PROJECT – MPI Solution (6)

Time stamp

In order to measure the time in the MPI programs, we recommend to use the ***MPI_Wtime*** **function**, as the flowing example shows:

```
float x, y, z;
double start, elapsed;

start = MPI_Wtime();

. . .

elapsed = MPI_Wtime() - start;
```

HPC PROJECT – Hybrid Solution

The Hybrid solution is the combination of the OpenMP optimization at node level and the MPI solution for the distributed memory model. Based on your Hybrid proposal arguee the following points:

- **Justify your decisions** according to the selected partitioning pattern, size of partitioning, communication directives, data location, loops parallelization configurationthread scheduling policies, and so on.
- **Evaluate Speedup and Efficiency**.
- **Compare the results** with the previously obtained.

deadline: **11th of June** (Hybrid Program)

HPC PROJECT – Hybrid Solution

Compile

In order to compile, you must continue the same commands of gcc openmp ”fopenmp” but using the mpicc compiler:

```
mpicc -fopenmp hybrid.c -o hello
```

HPC PROJECT – Hybrid Solution

Execution

Follow the same structure of mpi batch-file but now you must define the thread number :

```
#!/bin/bash

...

## Parallel programming environment (mpich) to instantiate and
number of computing slots.
#$ -pe mpich 32

..

##Passes an environment variable to the job
#$ -v  OMP_NUM_THREADS=4

## In this line you have to write the command that will execute
your application.
mpiexec -hosts compute-0-7.local,compute-0-2.local,compute-0-
1.local -n 3 ./hello
```

HPC PROJECT – Comparison of the results

Compare the quality of your proposal with respect a reference solution provided in the virtual campus. Identify the best solution. Indicate and justify possible improvements.

The aim of this activity is to promote critical thinking and help to identify the strengths and weaknesses of your own work. This is an optional activity with a qualification of an additional score over the global mark.

deadline: **20th of June** (Comparative study)