
 Universidad Pontificia Bolivariana	Facultad de Ingeniería de Sistemas e Informática. Sistemas Distribuidos MOOC - HPC - agosto 15 de 2020 Fecha de entrega: septiembre 7 de 2020
---	---

Realizado por:
Juan Camilo Restrepo Velez 000373886

MOOC1-HPC

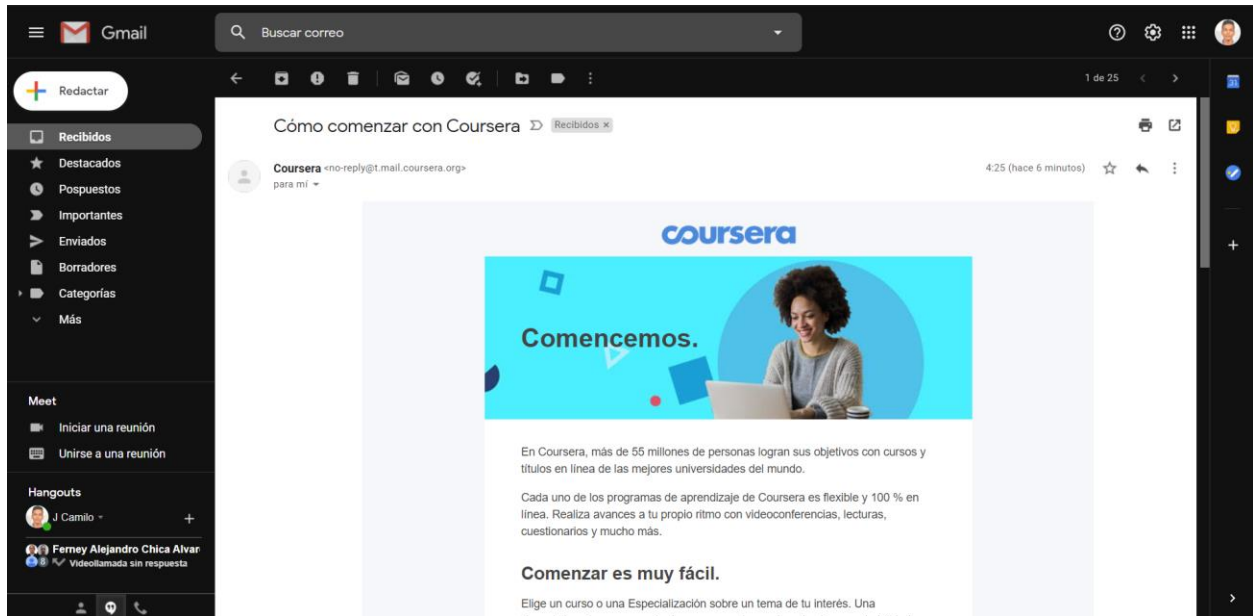
Tabla de contenido

1. La lectura de las notas y visualización de los videos.....	2
2. Presentación del Quiz de las semanas 1,2, 3 y 5 y adjuntar la evaluación.	8
Semana 1	8
Semana 2	9
Semana 3	10
Semana 5	12
3. Crear una cuenta en el clúster que Intel ofrece para el curso.	13
4. Descargar los códigos ejemplo de las unidades respectivas.	14
5. Compilarlos y correrlos en el clúster.	15
Semana 2	15
Stencil	15
Integral	16
Semana 3	16
Forks	16
Stencil	17
Semana 5	18
Stencil	18
Integral	19
6. REALIZAR el análisis de resultados.	20
Stencil.....	20
Integral.....	20
Forks	21

 Universidad Pontificia Bolivariana	<p>Facultad de Ingeniería de Sistemas e Informática. Sistemas Distribuidos MOOC - HPC - agosto 15 de 2020 Fecha de entrega: septiembre 7 de 2020</p>
---	--

1. La lectura de las notas y visualización de los videos.

Inscripción al curso



Seguimiento del curso



**Universidad
Pontificia
Bolivariana**

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Welcome

- ✓ **Lectura:** Modern Code 10 min
- ✓ **Vídeo:** 1.0 Introduction 3 min
- ✓ **Vídeo:** 1.1 Why this course? 5 min

How computers get faster

- ✓ **Vídeo:** 1.2 How Computers Get Faster 6 min
- ✓ **Vídeo:** 1.3 Intel Architecture 6 min

Modern code

- ✓ **Vídeo:** 1.4 Modern Code 5 min
 - ✓ **Vídeo:** 1.5 What You Are Going To Learn 1 min
 - ✓ **Vídeo:** 1.6 Remote Access 11 min
 - ✓ **Herramienta externa sin calificación:** Register for cluster access 1 h
-



**Universidad
Pontificia
Bolivariana**

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Vector Arithmetics in Intel Architecture

- ✓ **Lectura:** Vectorization 10 min
 - ✓ **Vídeo:** 2.1 Vector Operations 5 min
 - ✓ **Vídeo:** 2.2 Vectorizing Your Code 5 min
-

Automatic Vectorization

- ✓ **Vídeo:** 2.3.1 Automatic Vectorization 5 min
 - ✓ **Vídeo:** 2.3.2 Will This Vectorize? 15 min
 - ✓ **Vídeo:** 2.4 Guided Automatic Vectorization 4 min
-

Demo: Stencil Kernel

- ✓ **Vídeo:** 2.8.1 Stencil Introduction 2 min
 - ✓ **Vídeo:** 2.8 Stencil 9 min
 - ✓ **Lectura:** Code Download 10 min
-



**Universidad
Pontificia
Bolivariana**

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Designing Vectorizable Code

- ✓ **Video:** 2.5 SIMD-Enabled Functions 3 min
- ✓ **Video:** 2.6 Vector Dependence 5 min
- ✓ **Video:** 2.7 Strip Mining 1 min

Demo: Numerical Integration

- ✓ **Video:** Numerical Integration Introduction 2 min
- ✓ **Video:** 2.9 Integral Vectorization 8 min
- ✓ **Lectura:** Code Download 10 min

Homework

- ✓ **Video:** 2.10 Learn More 1 min



**Universidad
Pontificia
Bolivariana**

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Multiple Cores in Intel Architecture

- ✓ **Lectura:** Multithreading with OpenMP 10 min
- ✓ **Vídeo:** 3.1 Cores and Threads 3 min
- ✓ **Vídeo:** Demo: Forks 9 min
- ✓ **Lectura:** Code Download 10 min

OpenMP

- ✓ **Vídeo:** 3.2 Creating Threads 4 min
- ✓ **Vídeo:** 3.3 Variable Sharing 1 min
- ✓ **Vídeo:** 3.4 Parallel Loops 5 min
- ✓ **Vídeo:** 3.5 Data Races Mutexes 4 min
- ✓ **Vídeo:** 3.7 Parallel Reduction 4 min

Demo

- ✓ **Vídeo:** Stencil Introduction 2 min
- ✓ **Vídeo:** Stencil Demonstration 3 min
- ✓ **Lectura:** Code Download 10 min



**Universidad
Pontificia
Bolivariana**

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Clusters with Intel Architecture

- ✓ **Lectura:** Clusters and MPI 10 min
- ✓ **Vídeo:** 5.1 Computing Clusters 2 min

Message Passing Interface, MPI

- ✓ **Vídeo:** 5.2 Message Passing Interface 5 min
- ✓ **Vídeo:** 5.3 Programming with MPI 2 min
- ✓ **Vídeo:** 5.4 Compiling and Running with MPI 4 min
- ✓ **Vídeo:** 5.5 Peer-to-Peer Messaging 3 min
- ✓ **Vídeo:** 5.6 Collective Communication 5 min

Demo: Stencil

- ✓ **Vídeo:** Stencil Introduction 2 min
- ✓ **Vídeo:** Stencil Demonstration-MPI 9 min
- ✓ **Lectura:** Code Download 10 min

Demo: Numerical Integration

- ✓ **Vídeo:** Integral Introduction 2 min
- ✓ **Vídeo:** Integral Demonstration 4 min
- ✓ **Lectura:** Code Download 10 min



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

2. Presentación del Quiz de las semanas 1,2, 3 y 5 y adjuntar la evaluación.

Semana 1

1. Intel Xeon Phi processors are a...

1 punto

- ☐ ...specialized family of processors for applications requiring high clock speeds
- ☐ ...computing platform for legacy applications
- ☒ ...more efficient computing platform than traditional CPUs for applications that have good thread and vector parallelism

2. Multiple cores can share data in system memory

1 punto

- ☒ True
- ☐ False

3. In vector units, you can apply:

1 punto

- ☐ A single stream of instructions to a single data element in a vector
- ☐ Multiple streams of instructions to multiple data elements in a vector
- ☒ A single stream of instructions to multiple data elements in a short vector

4. There are no mainstream processors with clock speeds above 4 GHz. What is the reason for that? (pick all answers that apply)

1 punto

- ☒ Adding parallel processing capabilities to CPUs gives better performance per watt than increasing clock speeds
- ☒ Increasing the clock speed would increase the power requirement, which make the required cooling solutions expensive and impractical
- ☐ Clock speed above 4GHz would result in slower computational speed

5. Which of the following commands compiles a C++ application contained in a single file "code.cc" into an executable "myApp" using the Intel C++ compiler?

1 punto

- ☒ icpc code.cc -o myApp
- ☐ gcc code.cc -o myApp
- ☐ icpc code.cc myApp



Semana 2

1. Which of the two loops is safe to be vectorized with AVX-512 vector instructions acting on 512-bit vector registers? By "safe" we mean that the vectorized loop will produce the same results as the scalar loop. Here A is an array of type "double".

1 punto

Code A:

```
1 for (int i = 0; i < n-1; i++)  
2   A[i] += A[i+1];
```

Code B:

```
1 for (int i = 4; i < n; i++)  
2   A[i] += A[i-4];
```

- ☐ Both
- ☒ Code A
- ☐ Neither
- ☐ Code B
2. Which of the following cannot be automatically vectorized by the compiler?
- ☐ For loop
- ☒ While loop
3. What compiler argument do you have to use with the Intel C++ compiler to produce an optimization report in Linux?
- ☐ -opt-report
- ☒ -qopt-report
- ☐ /Qopt-report
4. In order to target multiple architectures, which compiler argument to use?
- ☐ -z[code]
- ☐ -ax[code]
- ☒ -x[code]
5. Which compiler argument should be used to target the architecture on which the code is compiled?
- ☒ -xAVX
- ☐ -xhost
- ☐ -axAVX
- ☐ -axMIC-AVX512



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Semana 3

1. OpenMP is a framework for...

- ☐ multiprocessing in distributed-memory systems (clusters)
- ☒ multithreading and vectorization in shared-memory systems

2. Which flag is required to compile your OpenMP application with Intel compiler?

- ☐ -fopenmp
- ☐ -xhost
- ☒ -qopenmp
- ☐ -qopt-report
- ☐ -openmp

3. Based on this snippet, which of the following is correct about the number of copies of a variable in memory at runtime if we are using 4 threads?

```
1 int A, B;  
2 #pragma omp parallel private(B)  
3 {  
4     int C;  
5     //code to be executed  
6 }
```

- ☒ There is a single copy of A and there are 4 copies of both B and C
- ☐ There is a single copy of A and B and there are 4 copies of C
- ☐ There are 4 copies of A, B and C

4. Which of these mutexes has the highest performance penalty?

- ☐ #pragma omp parallel
- ☐ #pragma omp atomic
- ☒ #pragma omp critical



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

5. What is the default number of OpenMP threads in an application running on an Intel Xeon Phi processor 7210 with 4-way hyper-threading? Feel free to look up the technical specifications of this processor model online

- ☐ 128
☐ 192
☒ 64
☐ 1
☐ 256

6. Which of the following code snippets has a data race ("race condition"):

Code A:

```
1 #pragma omp parallel for
2
3 for (int i = 0; i < n-1; i++)
4
5 A[i] += A[i+1];
```

Code B:

```
1 #pragma omp parallel for
2
3 for (int i = 1; i < n; i++)
4
5 A[i] += A[i-1];
```

- ☒ Both
☐ Code A
☐ Code B

7. Consider the following piece of code for computing (incorrectly?) the mean and standard deviation of values in an array:

```
1 void ComputeStats(int n, double* x, double & mean, double & stdev) {
2     mean = 0.0;
3     stdev = 0.0;
4
5     #pragma omp parallel for
6
7     for (int i = 0; i < n; i++) {
8         mean += x[i];
9         stdev += x[i]*x[i];
10    }
11
12    mean /= double(n);
13    stdev = sqrt(stdev/double(n) - mean*mean);
14
15 }
```

What should you change in this code to make it correct and efficient?

- ☐ Add a reduction clause in the parallel pragma
☒ Add a #pragma omp critical clause
☐ Add a #pragma omp atomic clause

Semana 5

1. In distributed-memory systems, the _____ framework can be used for communicating data between processes.

1 punto

- ☐ OpenMP
- ☐ Vectorization
- ☐ Ethernet
- ☒ MPI

2. The function MPI_Comm_size returns the _____

1 punto

- ☒ number of processes in a given communicator
- ☐ number of communicators in the applications
- ☐ number of cores in the processor
- ☐ size of the last communicated message

3. The MPI function that sends data from a group of processes to one process is _____

1 punto

- ☒ MPI_Gather
- ☐ MPI_Scatter
- ☐ MPI_Bcast
- ☐ MPI_Send

4. In a hybrid OpenMP+MPI application for a modern multicore processor, how do you manage the code of processes and threads? (choose the best answer)

1 punto

- ☐ MPI processes replace multithreading with one process running per core
- ☐ MPI processes send and receive the code of their threads by way of passing message
- ☒ The code of MPI processes has multithreading
- ☐ Threads launch multiple MPI processes

5. In this snippet, 2 processes are communicating, one is sending a message and the other is receiving it, in order to prevent this application from blocking, what are the correct values of DESTINATION and SOURCE ?

1 punto

```
1  if (myid == 1)
2  {
3      buffer="Hello world";
4      MPI_Send(&buffer, count, MPI_INT, DESTINATION, tag, MPI_COMM_WORLD);
5      printf("processor %d sent %d integers\n",myid,count);
6  }
7  if (myid == 2)
8  {
9      MPI_Recv(&buffer, count, MPI_INT, SOURCE, tag, MPI_COMM_WORLD, &status);
10     printf("processor %d received %d integers\n",myid,count);
11 }
```

- ☐ DESTINATION=2 and SOURCE=1 or 2
- ☐ DESTINATION=1 or 2 and SOURCE=1
- ☒ DESTINATION=2 and SOURCE=1
- ☐ DESTINATION=1 and SOURCE=2




Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020


3. Crear una cuenta en el clúster que Intel ofrece para el curso.

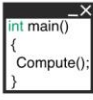
User ID: u47664@c008


Colfax Cluster for Coursera [Home](#) [Learn](#) [Connect](#) [Program](#) [Compute](#) [Log out](#)


Welcome to the Colfax Cluster for Coursera!

Learn

what to expect on the Colfax Cluster for Coursera

Connect

from your home computer to the cloud

Program

using modern code practices

Compute

with cluster job management tools

**To:** Coursera Learner 25764
From: Colfax Cluster for Coursera

Dear Coursera Learner 25764,

Please browse this portal for connection and usage instructions. If you have any problems connecting to or using the Colfax Cluster for Coursera, please search or post on the [Forum](#). Include your user ID in your post: u47664@c008.

Your account is active until **Sep 22 2020 11:34:53 UTC**. Your account and data will be deleted at expiration, so transfer out any data you wish to preserve before this date.

By using the Colfax Cluster for Coursera, you agree to abide by the following terms:

- [Colfax Terms of Service](#)

Sincerely,
Colfax Cluster for Coursera

© Colfax International 2016-2019



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

4. Descargar los códigos ejemplo de las unidades respectivas.

```
[u47664@c008 ~]$ ls -l && ls -l Semana\ 2 && ls -l Semana\ 3 && ls -l Semana\ 5
total 12
drwxrwxr-x. 4 u47664 u47664 4096 Sep  2 02:25 Semana 2
drwxrwxr-x. 4 u47664 u47664 4096 Sep  2 02:27 Semana 3
drwxrwxr-x. 4 u47664 u47664 4096 Sep  2 02:28 Semana 5
total 8
drwxrwxr-x. 3 u47664 u47664 4096 Apr 19 2017 integral
drwxr-xr-x. 3 u47664 u47664 4096 Apr 12 2017 stencil
total 8
drwxrwxr-x. 2 u47664 u47664 4096 Apr 11 2017 forks
drwxr-xr-x. 3 u47664 u47664 4096 Apr 12 2017 stencil
total 8
drwxrwxr-x. 3 u47664 u47664 4096 Apr 19 2017 integral
drwxr-xr-x. 3 u47664 u47664 4096 Apr 12 2017 stencil
[u47664@c008 ~]$
```



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

5. Compilarlos y correrlos en el clúster.

Semana 2

Stencil

```
u47664@c008:~/Semana 2/stencil/solutions/1-simd
[ u47664@c008 2-openmp]$ cd ~/Semana\ 2/stencil/solutions/
1-simd/      3-mcdram/      5-nontemporal/ 7-mpi/
2-threads/   4-memkind/      6-pngbyte/
[ u47664@c008 2-openmp]$ cd ~/Semana\ 2/stencil/solutions/1-simd/
[ u47664@c008 1-simd]$ make -B
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o main.o main.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output locati
on
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o image.o image.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output locati
on
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o stencil.o stencil.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output locati
on
icpc -qopenmp -lpng -o stencil main.o image.o stencil.o
[ u47664@c008 1-simd]$ make queue
echo 'cd $PBS_O_WORKDIR ; ./stencil test-image.png' | qsub -l nodes=1:flat -N edgedetection
126451.c008
[ u47664@c008 1-simd]$ qstat
[ u47664@c008 1-simd]$ cat edgedetection.o126451

#####
# Colfax Cluster - https://colfaxresearch.com/
#   Date:           Wed Sep  2 03:15:23 PDT 2020
#   Job ID:         126451.c008
#   User:           u47664
# Resources:        neednodes=1:flat,nodes=1:flat,walltime=00:02:00
#####

#####
# Colfax Cluster
# End of output for job 126451.c008
# Date: Wed Sep  2 03:15:24 PDT 2020
#####
```



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Integral

```
u47664@c008:~/Semana 2/integral/solutions/1-simd
[u47664@c008 1-simd]$ cd ~/Semana\ 2/integral/solutions/1-simd/
[u47664@c008 1-simd]$ make -B
icc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o main.o main.cc
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
icc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o library.o library.cc
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
icc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o worker.o worker.cc
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
icc -qopenmp -o integral main.o library.o worker.o
[u47664@c008 1-simd]$ make queue
echo 'cd $PBS_O_WORKDIR ; ./integral' | qsub -l nodes=1:flat -N numintegr
126452.c008
[u47664@c008 1-simd]$
[u47664@c008 1-simd]$ cat numintegr.o126452

#####
# Colfax Cluster - https://colfaxresearch.com/
#   Date:           Wed Sep  2 03:17:07 PDT 2020
#   Job ID:         126452.c008
#   User:           u47664
#   Resources:      neednodes=1:flat,nodes=1:flat,walltime=00:02:00
#####

#####
# Colfax Cluster
# End of output for job 126452.c008
# Date: Wed Sep  2 03:17:08 PDT 2020
#####
```

Semana 3

Forks

```
u47664@c008:~/Semana 3/forks
[u47664@c008 forks]$ cd ~/Semana\ 3/forks/
[u47664@c008 forks]$ make
icc -o multithreading fork-thread.c -pthread
icc -o multiprocessing fork-process.c
[u47664@c008 forks]$ # Ejecutar con hilos
[u47664@c008 forks]$ ./multithreading
In child thread at 0x6040e0: 'uninitialized':
In parent thread at 0x6040e0: 'I'm a little teapot, short and stout':
In child thread at 0x6040e0: 'I'm a little teapot, short and stout':
[u47664@c008 forks]$ # Ejecutar con procesos
[u47664@c008 forks]$ ./multiprocessing
In child process at 0x6040e0: 'uninitialized'
In parent process at 0x6040e0: 'I'm a little teapot, short and stout'
In child process at 0x6040e0: 'uninitialized'
[u47664@c008 forks]$
```




Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Stencil

```
u47664@c008:~/Semana 3/stencil/solutions/2-threads
[ u47664@c008 forks ]$ cd ~/Semana\ 3/stencil/solutions/2-threads/
[ u47664@c008 2-threads ]$ make -B
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o main.o main.cc
icpc: remark #10397: optimization reports are generated in *.optprt files in the output location
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o image.o image.cc
icpc: remark #10397: optimization reports are generated in *.optprt files in the output location
icpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o stencil.o stencil.cc
icpc: remark #10397: optimization reports are generated in *.optprt files in the output location
icpc -qopenmp -lpng -o stencil main.o image.o stencil.o
[ u47664@c008 2-threads ]$ make queue
echo 'cd $PBS_O_WORKDIR ; ./stencil test-image.png' | qsub -l nodes=1:flat -N edgedetection
126453.c008
[ u47664@c008 2-threads ]$ qstat
[ u47664@c008 2-threads ]$ cat edge
edgedetection.e126453 edgedetection.o126453
[ u47664@c008 2-threads ]$ cat edge
edgedetection.e126453 edgedetection.o126453
[ u47664@c008 2-threads ]$ cat edgedetection.o126453

#####
# Colfax Cluster - https://colfaxresearch.com/
#   Date:      Wed Sep  2 03:19:38 PDT 2020
#   Job ID:    126453.c008
#   User:      u47664
# Resources:   neednodes=1:flat,nodes=1:flat,walltime=00:02:00
#####

#####
# Colfax Cluster
# End of output for job 126453.c008
# Date: Wed Sep  2 03:19:39 PDT 2020
#####
```



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Semana 5

Stencil

```
u47664@c008:~/Semana 5/stencil/solutions/7-mpi
[ u47664@c008 2-threads]$ cd ~/Semana\ 5/stencil/solutions/7-mpi/
[ u47664@c008 7-mpi]$ make -B
mpicc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o main.o main.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpicc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o image.o image.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpicc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o stencil.o stencil.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpicc -qopenmp -lpng -lmemkind -o stencil main.o image.o stencil.o
[ u47664@c008 7-mpi]$ make queue
echo 'cd $PBS_O_WORKDIR ; mpirun -machinefile $PBS_NODEFILE ./stencil test-image.png' | qsub
-l nodes=4:flat -N edgedetection
126454.c008
[ u47664@c008 7-mpi]$ ls
image.cc image.o main.cc main.optrpt stencil stencil.h stencil.optrpt
image.h image.optrpt main.o Makefile stencil.cc stencil.o test-image.png
[ u47664@c008 7-mpi]$ cat edgedetection.o126454

#####
# Colfax Cluster - https://colfaxresearch.com/
# Date: Wed Sep 2 03:21:27 PDT 2020
# Job ID: 126454.c008
# User: u47664
# Resources: neednodes=4:flat,nodes=4:flat,walltime=00:02:00
#####

#####
# Colfax Cluster
# End of output for job 126454.c008
# Date: Wed Sep 2 03:21:29 PDT 2020
#####
```



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

Integral

```
u47664@c008:~/Semana 5/integral/solutions/4-mpi
[u47664@c008 4-mpi]$ cd ~/Semana\ 5/integral/solutions/4-mpi/
[u47664@c008 4-mpi]$ make -B
mpiicpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o main.o main.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpiicpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o library.o library.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpiicpc -c -qopenmp -qopt-report=5 -xMIC-AVX512 -c -o worker.o worker.cc
icpc: remark #10397: optimization reports are generated in *.optrpt files in the output location
mpiicpc -qopenmp -o integral main.o library.o worker.o
[u47664@c008 4-mpi]$ make queue
echo 'cd $PBS_O_WORKDIR ; mpirun -machinefile $PBS_NODEFILE ./integral' | qsub -l nodes=4:flat -N numintegr
126456.c008
[u47664@c008 4-mpi]$ qstat
Job ID              Name              User              Time Use S Queue
-----
126456.c008         numintegr         u47664            0 R batch
[u47664@c008 4-mpi]$ cat numintegr.o126456

#####
# Colfax Cluster - https://colfaxresearch.com/
#   Date:           Wed Sep  2 03:25:16 PDT 2020
#   Job ID:         126456.c008
#   User:           u47664
# Resources:        neednodes=4:flat,nodes=4:flat,walltime=00:02:00
#####

#####
# Colfax Cluster
# End of output for job 126456.c008
# Date: Wed Sep  2 03:25:18 PDT 2020
#####
```



Universidad
Pontificia
Bolivariana

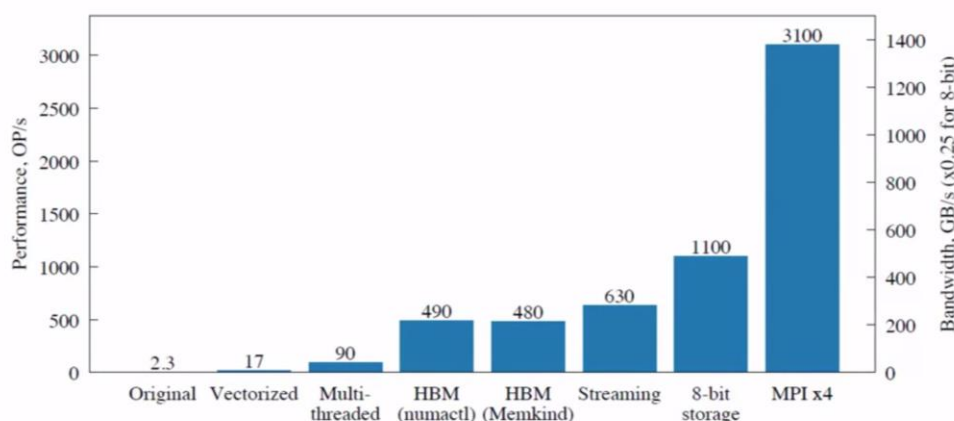
Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020

6. REALIZAR el análisis de resultados.

Stencil

En cuanto a los resultados arrojados por la ejecución se ve la diferencia entre la ejecución de MPI (Semana 5) contra Vectorización (Semana 2) e Hilos (Semana 3), donde los nodos de trabajos pasan de 1 a 4, ya que la vectorización y los hilos se realizan en la misma máquina, mientras el uso de MPI se realiza para la ejecución distribuida, es decir, en diferentes máquinas.

En cuanto a las medidas de rendimiento (Op/s) de las implementaciones se hace evidente en las gráficas compartidas durante el curso, la importancia y el impacto que tiene la programación distribuida sobre la demás, donde se alcanza a optimizar una aplicación en una gran medida.



Integral

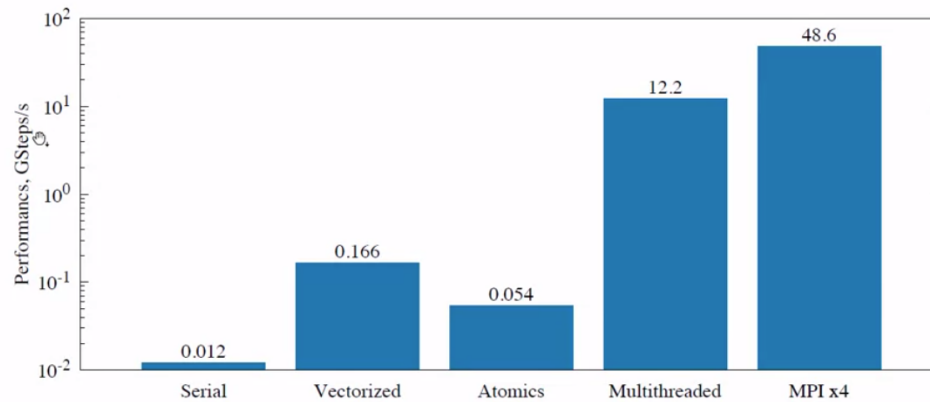
En el ejemplo de la integral se logra evidenciar lo explicado con el programa de Stencil; para la ejecución con MPI se asignan más de un equipo de trabajo (4 en este caso) mientras que para la aplicación de vectorización e hilos solo es requerido un equipo ya que estos se realizan internamente en la misma máquina.

En cuanto al rendimiento pasa lo mismo; la ejecución por medio de la programación distribuida logra alcanzar una mejor optimización del programa que por medio de la vectorización e hilos, que se evidencia en la gráfica compartida por el curso.



Universidad
Pontificia
Bolivariana

Facultad de Ingeniería de Sistemas e
Informática.
Sistemas Distribuidos
MOOC - HPC - agosto 15 de 2020
Fecha de entrega: septiembre 7 de 2020



Forks

Cuando se ejecuta el código con la implementación de hilos lo que pasa es que ambos flujos de instrucciones comparten el mismo espacio de memoria, por lo que están apuntando a la misma variable y cuando un hilo, en este caso el padre, cambia la variable el otro hilo también se ve afectado por dicho cambio, es decir que es la misma variable para ambos hilos.

```
void *Child_Thread(void *tid) {
    printf("In child thread at %p: '%s'\n", &msg, msg);
    sleep(2);
    printf("In child thread at %p: '%s'\n", &msg, msg);
}

void Parent_Thread() {
    sleep(1);
    strcpy(msg, "I'm a little teapot, short and stout");
    printf("In parent thread at %p: '%s'\n", &msg, msg);
}
```

Por otro lado, cuando la implementación se realiza creando otro proceso, la ejecución es diferente debido a que cuando se crea el otro proceso, este tiene la misma imagen que el padre y mapea las variables a su propio espacio de memoria, por lo que no comparten un espacio en común, es decir que si un proceso cambia el valor de una variable no afecta al otro proceso porque son variables diferentes.

```
void Child_Process() {
    printf("In child process at %p: '%s'\n", &msg, msg);
    sleep(2);
    printf("In child process at %p: '%s'\n", &msg, msg);
}

void Parent_Process() {
    sleep(1);
    strcpy(msg, "I'm a little teapot, short and stout");
    printf("In parent process at %p: '%s'\n", &msg, msg);
}
```