

ANEXO A

El presente anexo corresponde a la ejecución del experimento de evaluación de rendimiento de la ejecución de 4 algoritmos de multiplicación de matrices. Los fuentes de los algoritmos se encuentran desarrollados en lenguaje C. Para el ejercicio, se utilizó la terminal de Linux para la ejecución de los comandos que se encuentran descritos a continuación.

Evaluación de desempeño OpenMP

OBJETIVO

Hacer un estudio de rendimiento de 4 algoritmos de multiplicación de matrices utilizando OpenMP.

Desarrollo del experimento:

El experimento fue realizado en un servidor con 20 núcleos de procesamiento y 64 Gb de RAM.

1. Localización de archivos en el directorio del repositorio:

Los fuentes utilizados para realización de este experimento se encuentran disponibles dentro del repositorio en la ruta que se muestra a continuación en la Figura 1:

/OpenMP/src

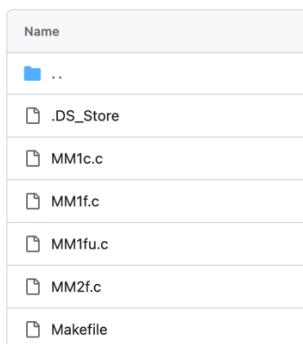


Figura 1: Directorio /OpenMP/src

Adicional a los archivos.c, se incluye el archivo makefile, utilizado para compilar los fuentes.

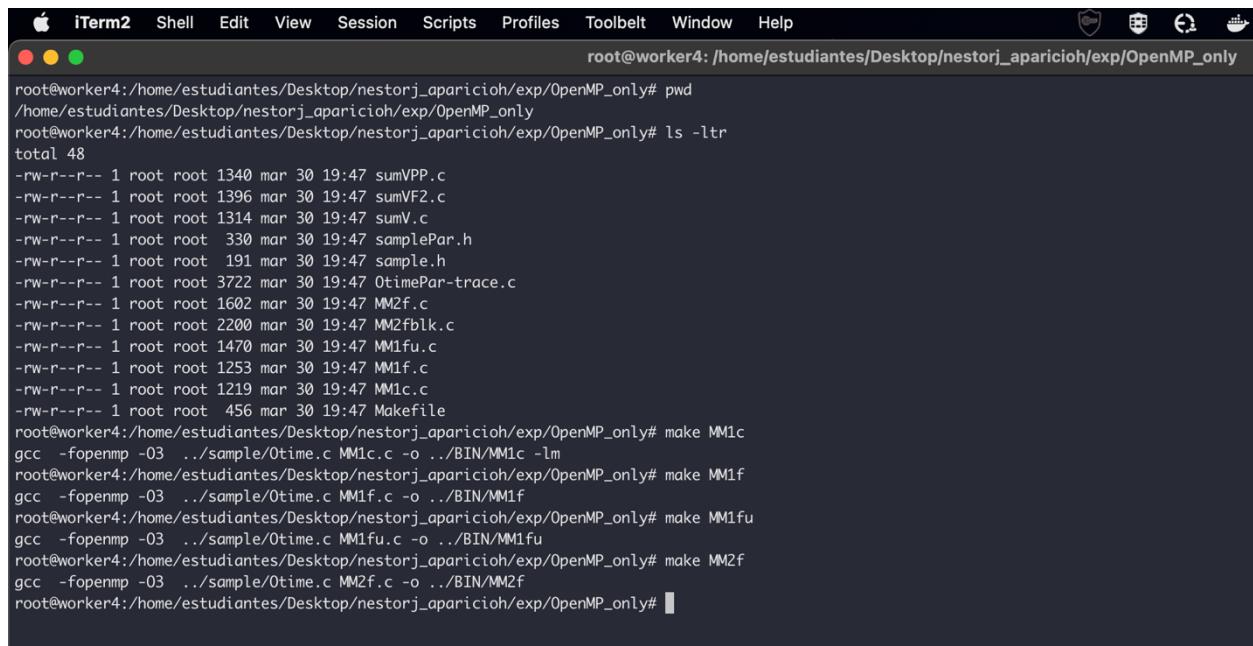
2. Compilar los archivos fuente:

Para el experimento vamos a compilar los archivos fuentes de cada uno de los algoritmos de multiplicación de matrices MM1c.c, MM1f.c, MM1fu.c y MM2f.c.

Desde el directorio `src` compilar los programas usando los siguientes comandos:

```
make MM1c  
make MM1f  
make MM1fu  
make MM2f
```

A modo de ejemplo, la Figura 2 muestra resultado de la ejecución de los comandos indicados:



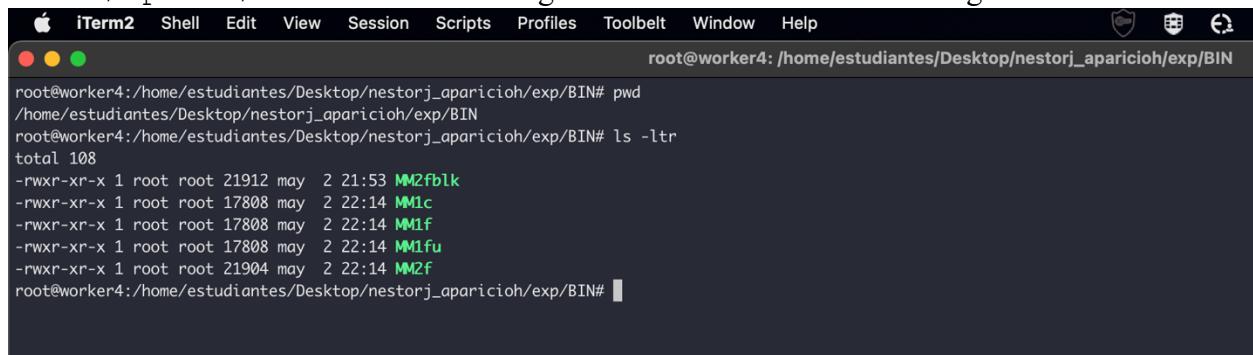
The screenshot shows a terminal window titled "root@worker4: /home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only". The user runs several commands to compile source files into executables. The output shows the creation of files like sumVPP.c, sumV.c, samplePar.h, sample.h, OtimePar-trace.c, MM2f.c, MM2blk.c, MM1c.c, MM1f.c, MM1fu.c, and Makefile. It also shows the compilation of these source files into executables named MM1c, MM1f, and MM1fu.

```
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# pwd  
/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# ls -ltr  
total 48  
-rw-r--r-- 1 root root 1340 mar 30 19:47 sumVPP.c  
-rw-r--r-- 1 root root 1396 mar 30 19:47 sumVF2.c  
-rw-r--r-- 1 root root 1314 mar 30 19:47 sumV.c  
-rw-r--r-- 1 root root 330 mar 30 19:47 samplePar.h  
-rw-r--r-- 1 root root 191 mar 30 19:47 sample.h  
-rw-r--r-- 1 root root 3722 mar 30 19:47 OtimePar-trace.c  
-rw-r--r-- 1 root root 1602 mar 30 19:47 MM2f.c  
-rw-r--r-- 1 root root 2200 mar 30 19:47 MM2blk.c  
-rw-r--r-- 1 root root 1470 mar 30 19:47 MM1c.c  
-rw-r--r-- 1 root root 1253 mar 30 19:47 MM1f.c  
-rw-r--r-- 1 root root 1219 mar 30 19:47 MM1c.c  
-rw-r--r-- 1 root root 456 mar 30 19:47 Makefile  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# make MM1c  
gcc -fopenmp -O3 ..\sample\Otime.c MM1c.c -o ..\BIN\MM1c -lm  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# make MM1f  
gcc -fopenmp -O3 ..\sample\Otime.c MM1f.c -o ..\BIN\MM1f  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# make MM1fu  
gcc -fopenmp -O3 ..\sample\Otime.c MM1fu.c -o ..\BIN\MM1fu  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only# make MM2f  
gcc -fopenmp -O3 ..\sample\Otime.c MM2f.c -o ..\BIN\MM2f  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/OpenMP_only#
```

Figura 2 Ejemplo compilación archivos fuente

3. Validar creación de archivos ejecutables:

Una vez se compilan los archivos fuentes, los ejecutables se almacenan en la ruta: `/OpenMP/bin`. Los archivos se generan como se muestra en la Figura 3:



The screenshot shows a terminal window titled "root@worker4: /home/estudiantes/Desktop/nestorj_aparicioh/exp/BIN". The user runs commands to check the contents of the `BIN` directory. The output lists several executable files: MM2blk, MM1c, MM1f, MM1fu, and MM2f, each with specific file sizes and modification dates.

```
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/BIN# pwd  
/home/estudiantes/Desktop/nestorj_aparicioh/exp/BIN  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/BIN# ls -ltr  
total 108  
-rwxr-xr-x 1 root root 21912 may 2 21:53 MM2blk  
-rwxr-xr-x 1 root root 17808 may 2 22:14 MM1c  
-rwxr-xr-x 1 root root 17808 may 2 22:14 MM1f  
-rwxr-xr-x 1 root root 17808 may 2 22:14 MM1fu  
-rwxr-xr-x 1 root root 21904 may 2 22:14 MM2f  
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/BIN#
```

Figura 3 Ejemplo archivos compilados

4. Archivos para lanzar el experimento:

En la carpeta `/OpenMP/tools` se incluyen los archivos utilizados para la ejecución del experimento, tal como lo ilustra la Figura 4:

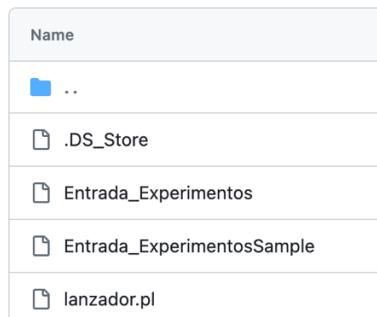


Figura 4 directorio `/OpenMP/tools`

Descripción de los archivos:

- a. **Entrada_Experimentos:** En este archivo se configuran los parametros para la ejecución para el experimento, así:

```
OpenMP      // Kind of Experiment – Tipo de experimento
MM1c,MM1f,MM1fu,MM2f // Binarys – Archivos ejecutables a utilizar en la prueba
2,4,6,8,10,12,14,16,18,20 // Threads – Cantidad de hilos que se utilizará en cada ciclo
30 // Number of Repetitions per experiment – Número de repeticiones
Adicional se configurará el tamaño de las matrices cuadradas de n*n así:
:3500
:4000
:4500
:5000
:5500
:6000
:6500
:7000
:7500
```

Ejemplo de configuración del archivo `Entrada_Experimentos` para la ejecución del experimento:

```
#####
# PONTIFICIA UNIVERSIDAD JAVERIANA
# ZINE Centro de Alto Rendimiento Computacional Javeriano
#
#####

#####
# Inputs file experiments
#####

OpenMP          // Kind of Experiment
MM1c,MM1f,MM1fu,MM2f      // Binarys
2,4,6,8,10,12,14,16,18,20 // Threads
30                  // Number of Repetitions per experiment

#####

# Inputs file => Matrix Size NxN
#####
:3500
:4000
:4500
:5000
:5500
:6000
:6500
:7000
:7500
```

La Figura 5 muestra un ejemplo del archivo configurado para la ejecución del experimento:

```

root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# pwd
/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# ls -ltr
total 36
-rwxr-xr-x 1 root root 10240 mar 30 19:47 raiz.tar
drwxr-xr-x 2 root root 4096 mar 30 19:47 Old-TOOLS
drwxr-xr-x 2 root root 4096 mar 30 19:47 OLD_2023
-rwxr-xr-x 1 root root 2161 mar 30 19:47 lanzador.pl
-rwxr-xr-x 1 root root 859 may 2 22:01 Entrada_Experimentos_BK1
-rwxr-xr-x 1 root root 805 may 2 22:02 Entrada_Experimentos_bk
-rwxr-xr-x 1 root root 875 may 2 22:04 Entrada_Experimentos
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# cat Entrada_Experimentos
#####
# PONTIFICIA UNIVERSIDAD JAVERIANA
# ZINE Centro de Alto Rendimiento Computacional Javeriano
#
#####

#####
# Inputs file experiments
#####

OpenMP           // Kind of Experiment
MM1c,MM1f,MM1fu,MM2f // Binaries
2,4,6,8,10,12,14,16,18,20 // Threads
30               // Number of Repetitions per experiment

#####
# Inputs file => Matrix Size NxN
#####
:3500
:4000
:4500
:5000
:5500
:6000
:6500
:7000
:7500
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# 

```

Figura 5 Ejemplo archivo Entrada_Experimentos

- b. **lanzador.pl:** Este archivo se encarga de ejecutar dados los parámetros definidos en el archivo Entrada_Experimentos, cada una de las iteraciones propuestas en el experimento.

Este archivo fue desarrollado por el docente John Corredor y facilitado para la realización de este experimento:

```

#!/usr/bin/perl
# Created by: John Corredor UAB-CAOS
# john@aopcjc.uab.es
# Julio 2008
#
# Lanza el binary juntos con los argumentos desde argumentos.txt
# y los eventos, desde el archivo Entrada_Experimentos

```

```

if (@ARGV[0]) {
    $FInput = "$ARGV[0]";
} else {
    usage();
}

use Switch;

$path0 = `pwd`;
chomp($path0);
$T = index($path0,"T")-1;
$Path = substr($path0,0,$T);

@Labels = ("Experiment","Binarys","Threads","Repetitions");
&Search(\@Labels);
$Experiment = $Inputs[0];
@Binarys   = split(//,$Inputs[1]);
@Threads   = split(//,$Inputs[2]);
$Repet     = $Inputs[3];

@Separate = ("-",":");

#Capture SIZE MATRIX in vector @Size
$Size   = &Argument($Separate[1],1);  @Size   = split(/\n/, $arg);

#####
# Main program: WARNING Binary x Threads x Size x Repetitions  #
#####

foreach $executable (@Binarys) {
    $CARP = $executable."-".$Experiment."-";
    chop($CARP);
    system ("mkdir $Path/Soluciones/$CARP");
    foreach $thread (@Threads) {
        foreach $size (@Size) {
            $file = "$Path/Soluciones/$CARP/$executable-$size.-TH-.$thread.txt";
            print "$executable $size $thread ... on $file\n";
            for ($k = 0; $k < $Repet; $k++) {
                system("$Path/BIN/$executable $size $thread 0 0 >> $file");
                #print "$Path/BIN/$executable $size $thread 0 0 \n";
            }
            close($file);
        }
    }
}

```

```
#####
# Functions          #
#####

sub Argument {
    $lab = $_[0];
    $pos = $_[1]; # pos: 1 2 3 4 5
    $arg = `cat $FInput | grep $lab | cut -f2 -d$lab | cut -f$pos -d' ''`;
    chomp($arg);
}

sub Search {
    my($label) = @_;
    foreach (@{$label}) {
        $name = `grep $_ $FInput | cut -f1`;
        chomp($name);
        push(@Inputs, $name);
    }
}

sub Formato {
    $in = $_[0];
    if ($in < 1000) {
        $Id = $in;
    } elsif($in < 10000) {
        $ddk = $in/1000;
        $Id = $ddk."K";
    } elsif($in < 1000000){
        $ddk = $in/1000000;
        $Id = $ddk."M";
    }
}

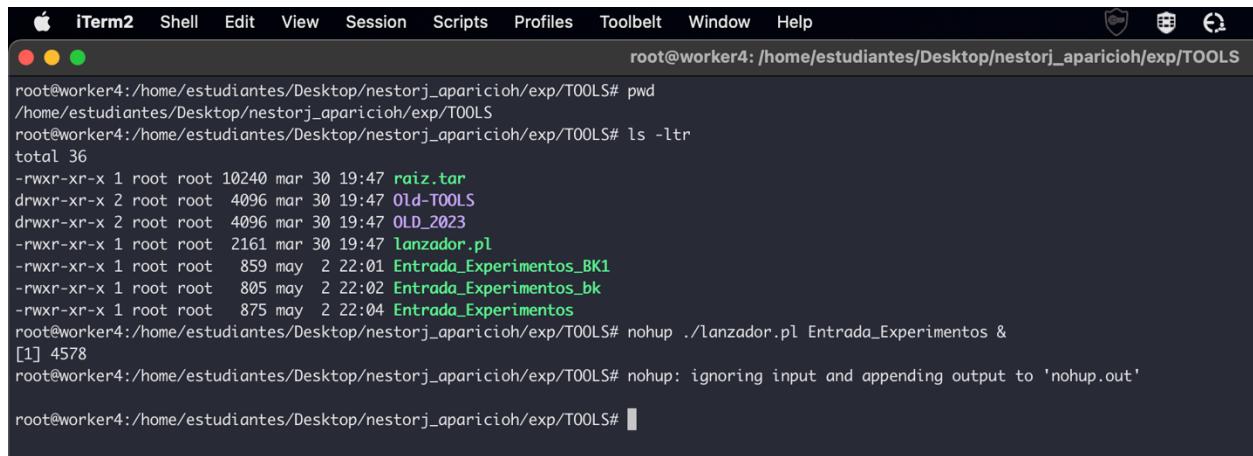
sub usage() {
    print "\n test.pl: Error incorrect usage \n\n";
    print "\t<directory> Is directory start point to check \n\n\n";
    exit(1);
}
```

5. Ejecución del experimento: Una vez revisados y configurados los archivos listados en el punto 4, se procede a hacer la ejecución del experimento, para ello, se debe ejecutar el archivo `lanzador.pl`, utilizando el siguiente comando:

```
./lanzador.pl <archivo_con_parametros>
./lanzador.pl Entrada_Experimentos
```

El comando anterior, ejecuta el proceso y mostrará su avance en la terminal de la sesión desde la que se ejecute, sin embargo, si la sesión se cierra, el experimento finalizará su ejecución. Para las ejecuciones de extensa duración como la propuesta en para este experimento, se sugiere utilizar el **nohup** para que la ejecución se haga en segundo plano y continue su tarea aún cuando la sesión desde la que se lanzó la instrucción haya finalizado (ejemplo en la Figura 6). De esta manera, el comando utilizado fue:

```
nohup ./lanzador.pl Entrada_Experimentos &
```



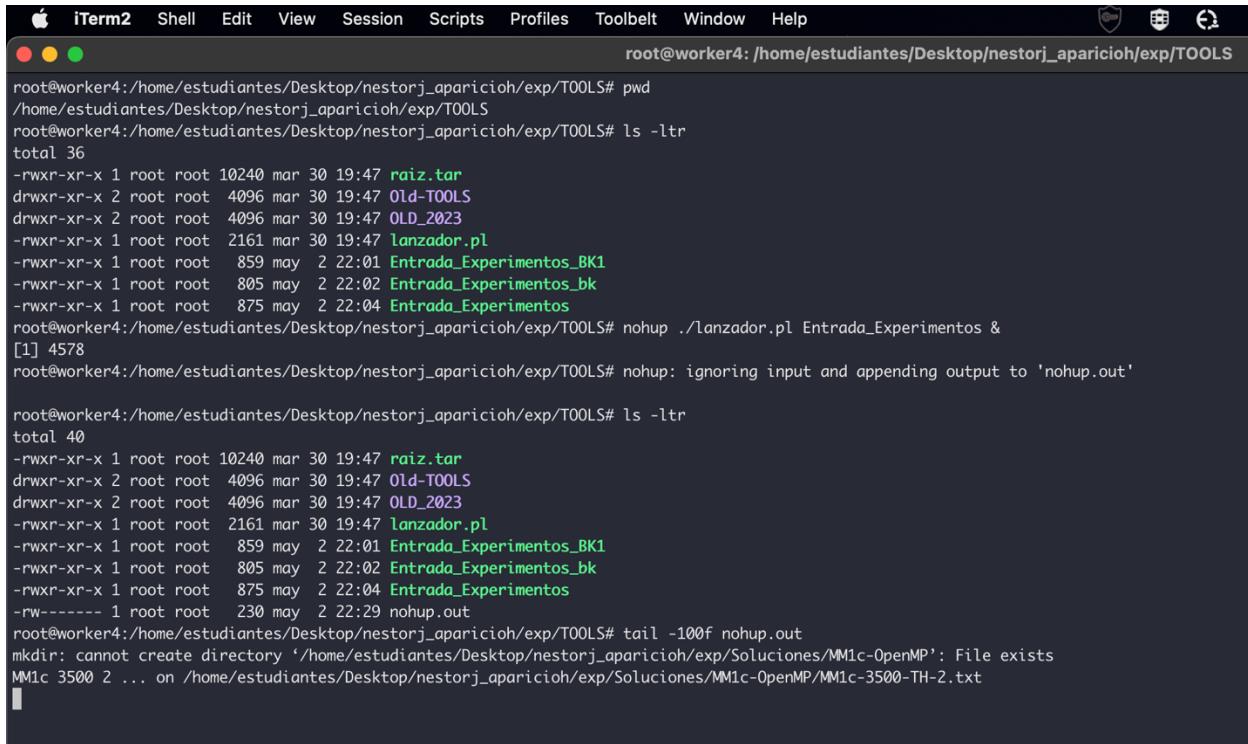
The screenshot shows an iTerm2 terminal window with the following session details:

- Terminal title: iTerm2
- Session: root@worker4: /home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
- Content:
 - pwd command output: /home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
 - ls -ltr command output:
 - total 36
 - rwxr-xr-x 1 root root 10240 mar 30 19:47 **raiz.tar**
 - drwxr-xr-x 2 root root 4096 mar 30 19:47 Old-TOOLS
 - drwxr-xr-x 2 root root 4096 mar 30 19:47 OLD_2023
 - rwxr-xr-x 1 root root 2161 mar 30 19:47 **lanzador.pl**
 - rwxr-xr-x 1 root root 859 may 2 22:01 **Entrada_Experimentos_BK1**
 - rwxr-xr-x 1 root root 805 may 2 22:02 **Entrada_Experimentos_bk**
 - rwxr-xr-x 1 root root 875 may 2 22:04 **Entrada_Experimentos**
 - nohup command execution:
 - nohup ./lanzador.pl Entrada_Experimentos &
 - [1] 4578
 - Final command: nohup: ignoring input and appending output to 'nohup.out'

Figura 6 Ejemplo ejecución lanzador.pl con nohup

Una vez lanzado el proceso con el comando nohup, generará un archivo nohup.out que contendrá el log de ejecución del comando ejecutado. De ser requerido, mientras se procesa el experimento, se puede hacer seguimiento del avance del proceso usando el comando tail para verificar el contenido del archivo nohup.out como se muestra a modo de ejemplo en la Figura 7.

```
tail -100f nohup.out
```



```
iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# pwd
/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# ls -ltr
total 36
-rwxr-xr-x 1 root root 10240 mar 30 19:47 raiz.tar
drwxr-xr-x 2 root root 4096 mar 30 19:47 Old-TOOLS
drwxr-xr-x 2 root root 4096 mar 30 19:47 OLD_2023
-rwxr-xr-x 1 root root 2161 mar 30 19:47 lanzador.pl
-rwxr-xr-x 1 root root 859 may 2 22:01 Entrada_Experimentos_BK1
-rwxr-xr-x 1 root root 805 may 2 22:02 Entrada_Experimentos_bk
-rwxr-xr-x 1 root root 875 may 2 22:04 Entrada_Experimentos
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# nohup ./lanzador.pl Entrada_Experimentos &
[1] 4578
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# nohup: ignoring input and appending output to 'nohup.out'

root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# ls -ltr
total 40
-rwxr-xr-x 1 root root 10240 mar 30 19:47 raiz.tar
drwxr-xr-x 2 root root 4096 mar 30 19:47 Old-TOOLS
drwxr-xr-x 2 root root 4096 mar 30 19:47 OLD_2023
-rwxr-xr-x 1 root root 2161 mar 30 19:47 lanzador.pl
-rwxr-xr-x 1 root root 859 may 2 22:01 Entrada_Experimentos_BK1
-rwxr-xr-x 1 root root 805 may 2 22:02 Entrada_Experimentos_bk
-rwxr-xr-x 1 root root 875 may 2 22:04 Entrada_Experimentos
-rw----- 1 root root 230 may 2 22:29 nohup.out
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# tail -100f nohup.out
mkdir: cannot create directory '/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP': File exists
MM1c 3500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-2.txt
```

Figura 7 Vista del contenido del archivo nohup.out

Finalizada la ejecución, el contenido del archivo nohup.out se muestra similar al que se muestra en la Figura 8. El archivo nohup.out, se crea en la ruta desde la cual se haya lanzado el comando, para el ejemplo de la Figura 8, el archivo se generó en la ruta:

/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS

```

root@worker4: /home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS
-rwxr-xr-x 1 root root 2161 mar 30 19:47 lanzador.pl
-rwxr-xr-x 1 root root 859 may  2 22:01 Entrada_Experimentos_BK1
-rwxr-xr-x 1 root root 805 may  2 22:02 Entrada_Experimentos_bk
-rw----- 1 root root 6489 may  7 17:57 nohup.out
-rwxr-xr-x 1 root root 875 may  8 21:53 Entrada_ExperimentosEjecutados
-rwxr-xr-x 1 root root 795 may  8 21:54 Entrada_Experimentos
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/TOOLS# cat nohup.out
mkdir: cannot create directory '/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP': File exists
MM1c 3500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-2.txt
MM1c 4000 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4000-TH-2.txt
MM1c 4500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4500-TH-2.txt
MM1c 5000 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5000-TH-2.txt
MM1c 5500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5500-TH-2.txt
MM1c 6000 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6000-TH-2.txt
MM1c 6500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6500-TH-2.txt
MM1c 7000 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7000-TH-2.txt
MM1c 7500 2 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7500-TH-2.txt
MM1c 3500 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-4.txt
MM1c 4000 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4000-TH-4.txt
MM1c 4500 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4500-TH-4.txt
MM1c 5000 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5000-TH-4.txt
MM1c 5500 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5500-TH-4.txt
MM1c 6000 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6000-TH-4.txt
MM1c 6500 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6500-TH-4.txt
MM1c 7000 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7000-TH-4.txt
MM1c 7500 4 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7500-TH-4.txt
MM1c 3500 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-6.txt
MM1c 4000 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4000-TH-6.txt
MM1c 4500 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4500-TH-6.txt
MM1c 5000 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5000-TH-6.txt
MM1c 5500 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5500-TH-6.txt
MM1c 6000 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6000-TH-6.txt
MM1c 6500 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6500-TH-6.txt
MM1c 7000 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7000-TH-6.txt
MM1c 7500 6 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7500-TH-6.txt
MM1c 3500 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-8.txt
MM1c 4000 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4000-TH-8.txt
MM1c 4500 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-4500-TH-8.txt
MM1c 5000 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5000-TH-8.txt
MM1c 5500 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-5500-TH-8.txt
MM1c 6000 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6000-TH-8.txt
MM1c 6500 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-6500-TH-8.txt
MM1c 7000 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7000-TH-8.txt
MM1c 7500 8 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-7500-TH-8.txt
MM1c 3500 10 ... on /home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP/MM1c-3500-TH-10.txt

```

Figura 8 Ejemplo contenido archivo nohup.out

6. Revisión de resultados:

De la ejecución anterior se creará el directorio **Soluciones**, al mismo nivel de directorio de la carpeta **tools**. Dentro de este directorio se almacenan las salidas producto de la ejecución del lanzador (**lanzador.pl**). Para cada algoritmo se crea una carpeta con el nombre de este. Así mismo, dentro de cada directorio de algoritmo, Por cada combinación de **Algoritmo-TamañoMatriz-CantidadHilos** que se haya configurado en el archivo **Entrada_Experimentos**, se crea un archivo con el resultado de cada una de las ejecuciones o cantidad de experimentos configurados en el mismo archivo **Entrada_Experimentos**. A modo de ejemplo, en la Figura 9 se muestra uno de estos directorios con los archivos generados para el experimento ejecutado:

```

● ● ●
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones#
BK MM1c-OpenMP MM1f-OpenMP MM1fu-OpenMP MM2F-OpenMP
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones# cd MM1c-OpenMP/
root@worker4:/home/estudiantes/Desktop/nestorj_aparicioh/exp/Soluciones/MM1c-OpenMP# ls -ltr
total 236
-rw-r--r-- 1 root root 300 may  2 23:18 MM1c-3500-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 00:33 MM1c-4000-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 02:32 MM1c-4500-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 05:20 MM1c-5000-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 09:11 MM1c-5500-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 14:23 MM1c-6000-TH-2.txt
-rw-r--r-- 1 root root 300 may  3 21:09 MM1c-6500-TH-2.txt
-rw-r--r-- 1 root root 320 may  4 05:43 MM1c-7000-TH-2.txt
-rw-r--r-- 1 root root 330 may  4 16:37 MM1c-7500-TH-2.txt
-rw-r--r-- 1 root root 300 may  4 17:03 MM1c-3500-TH-4.txt
-rw-r--r-- 1 root root 300 may  4 17:42 MM1c-4000-TH-4.txt
-rw-r--r-- 1 root root 300 may  4 18:45 MM1c-4500-TH-4.txt
-rw-r--r-- 1 root root 300 may  4 20:12 MM1c-5000-TH-4.txt
-rw-r--r-- 1 root root 300 may  4 22:13 MM1c-5500-TH-4.txt
-rw-r--r-- 1 root root 300 may  5 00:58 MM1c-6000-TH-4.txt
-rw-r--r-- 1 root root 300 may  5 04:31 MM1c-6500-TH-4.txt
-rw-r--r-- 1 root root 300 may  5 09:04 MM1c-7000-TH-4.txt
-rw-r--r-- 1 root root 300 may  5 14:51 MM1c-7500-TH-4.txt
-rw-r--r-- 1 root root 300 may  5 15:09 MM1c-3500-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 15:38 MM1c-4000-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 16:27 MM1c-4500-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 17:36 MM1c-5000-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 19:09 MM1c-5500-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 21:16 MM1c-6000-TH-6.txt
-rw-r--r-- 1 root root 300 may  5 23:59 MM1c-6500-TH-6.txt
-rw-r--r-- 1 root root 300 may  6 03:30 MM1c-7000-TH-6.txt
-rw-r--r-- 1 root root 300 may  6 07:53 MM1c-7500-TH-6.txt
-rw-r--r-- 1 root root 300 may  6 08:09 MM1c-3500-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 08:37 MM1c-4000-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 09:17 MM1c-4500-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 10:14 MM1c-5000-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 11:32 MM1c-5500-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 13:17 MM1c-6000-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 15:33 MM1c-6500-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 18:30 MM1c-7000-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 21:52 MM1c-7500-TH-8.txt
-rw-r--r-- 1 root root 300 may  6 22:04 MM1c-3500-TH-10.txt
-rw-r--r-- 1 root root 300 may  6 22:23 MM1c-4000-TH-10.txt

```

Figura 9 Ejemplo archivos generados. Algoritmo MM1c

7. Tabulación de los resultados:

A modo de referencia dentro del repositorio se incluye el directorio **salidas** y el archivo **ConsolidadoSalidasOpenMp.xlsx**, los cuales corresponden archivos generados como salida del experimento y la tabulación de esta información respectivamente. Se ilustra la ubicación de los archivos y el directorio en la Figura 10. La Figura 11 por su parte, muestra una de las gráficas obtenidas luego de graficar los resultados del algoritmo MM1c. Haciendo uso del archivo **ConsolidadoSalidasOpenMp.xlsx**, se puede validar la información para los demás algoritmos, así como generar las gráficas que el usuario desee en función a los datos recolectados en el experimento.

Name
..
Salidas
bin
src
tools
.DS_Store
ConsolidadoSalidasOpenMp.xlsx

Figura 10 Directorio /OpenMP

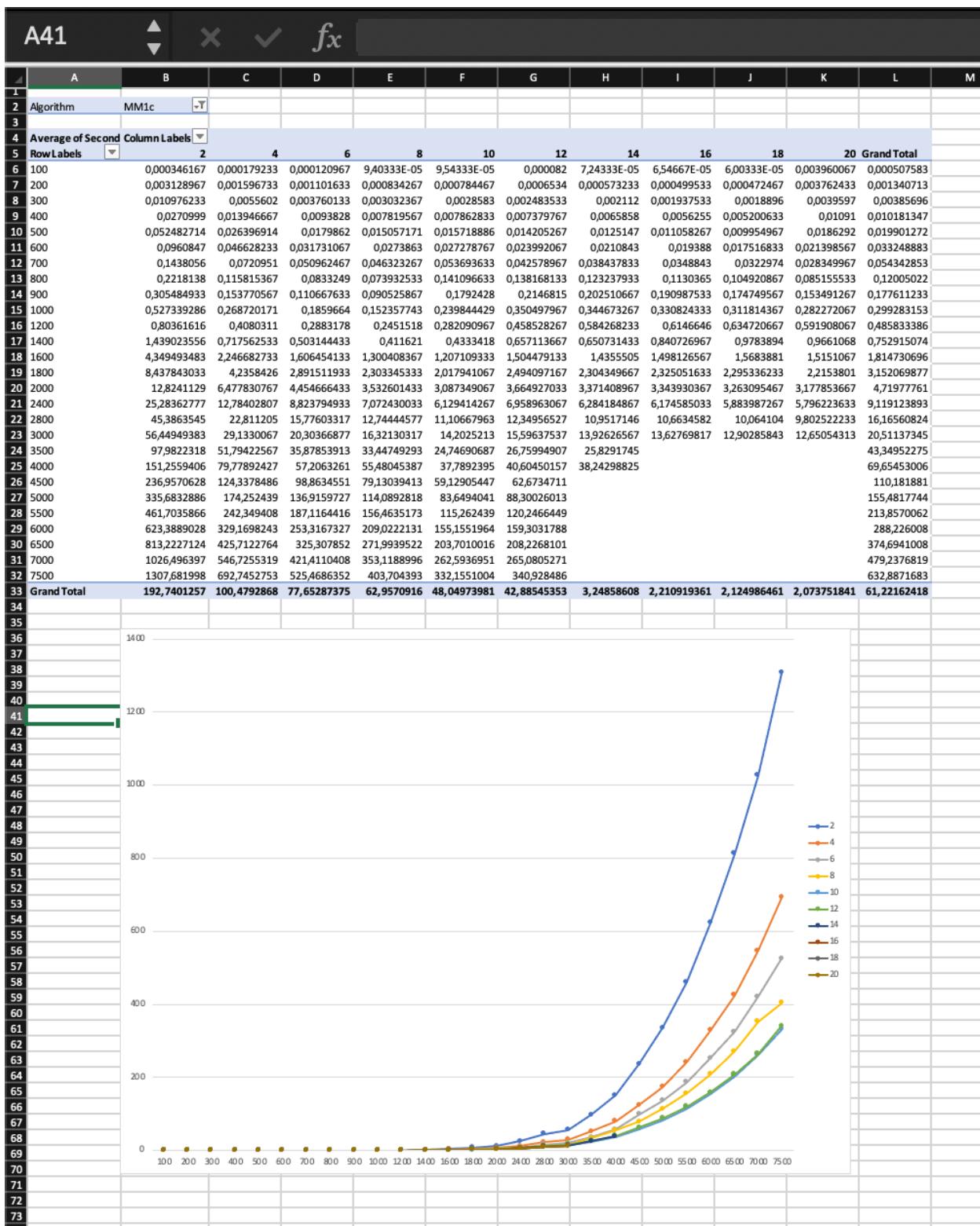


Figura 11 Ejemplo gráfica algoritmo MM1c

Evaluación de desempeño MPI

OBJETIVOS

- Hacer un estudio de rendimiento de 4 algoritmos de multiplicación de matrices utilizando MPI.
- Realizar una comparativa del estudio de rendimiento entre MPI vs OpenMP.

Desarrollo del experimento:

Para la ejecución del experimento se utilizó un cluster previamente configurado para la ejecución del experimento. Desde el nodo manager se realizó la ejecución de los comandos que se presentan a continuación:

Para la ejecución del experimento se utilizaron 2 nodos de tipo worker cada uno con 20 nucleos de procesamiento y 64 Gb de RAM.

Dentro del repositorio, se incluye el árbol de directorios que ilustra la Figura 12, los cuales se revisarán dentro del desarrollo del experimento:

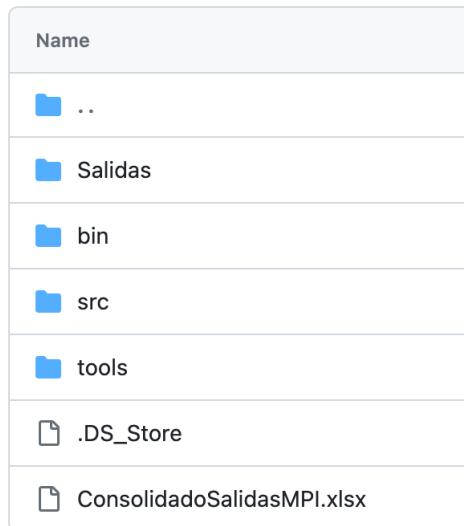
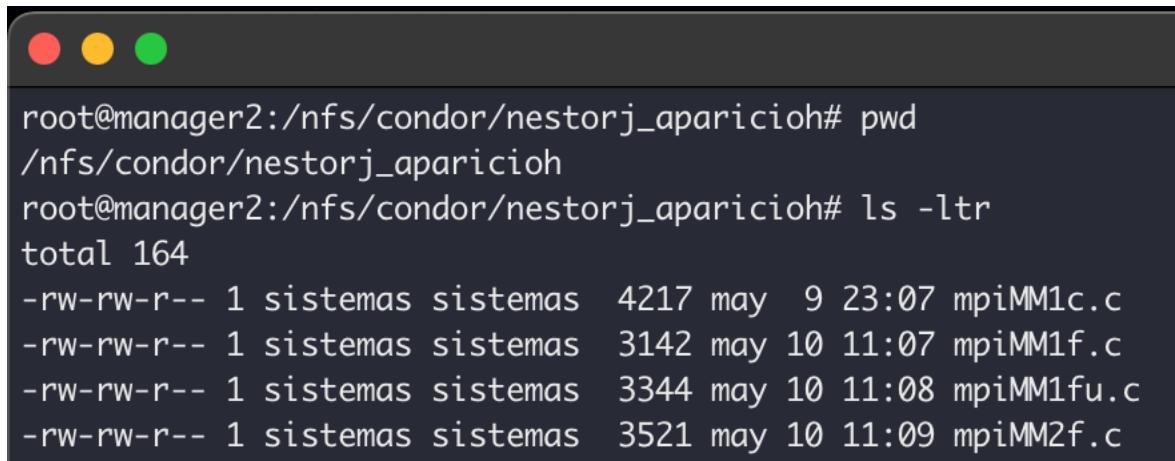


Figura 12 árbol del directorio /MPI

1. Localización de archivos en el directorio del repositorio:

Inicialmente se accede al nodo manager a través de una conexión ssh. Para el experimento previamente se configuró un directorio compartido en el cluster visible para todos los nodos del mismo, el cual se mostrará en el presente anexo como el directorio /nfs. Dentro de la ruta /nfs/condor/ se crea la carpeta con el nombre de usuario de uno de los autores del experimento. Dentro de esta carpeta se copian los fuentes de los algoritmos de multiplicación de matrices MM1c.c, MM1f.c, MM1fu.c y MM2f.c en lenguaje C y con

estructura de **MPI** como se ilustra en la Figura 12. Dentro del repositorio se incluyen los fuentes descritos en el directorio `/MPI/src` como se ilustra en la Figura 13.



```
root@manager2:/nfs/condor/nestorj_aparicioh# pwd
/nfs/condor/nestorj_aparicioh
root@manager2:/nfs/condor/nestorj_aparicioh# ls -ltr
total 164
-rw-rw-r-- 1 sistemas sistemas 4217 may  9 23:07 mpiMM1c.c
-rw-rw-r-- 1 sistemas sistemas 3142 may 10 11:07 mpiMM1f.c
-rw-rw-r-- 1 sistemas sistemas 3344 may 10 11:08 mpiMM1fu.c
-rw-rw-r-- 1 sistemas sistemas 3521 may 10 11:09 mpiMM2f.c
```

Figura 13 Ejemplo copia de fuentes nodo master del cluster

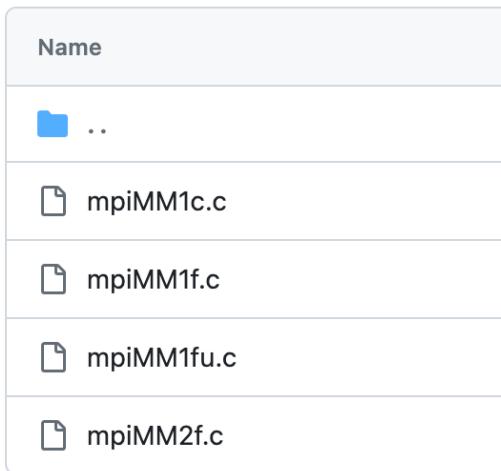


Figura 14 Repositorio directorio `/MPI/src`

2. Compilar los archivos fuente:

Se procedió a compilar estos fuentes utilizando **openMPI**, para ello se utilizaron los siguientes comandos:

```
mpicc mpiMM1c.c -o mpiMM1c
mpicc mpiMM1f.c -o mpiMM1f
mpicc mpiMM1fu.c -o mpiMM1fu
mpicc mpiMM2f.c -o mpiMM2f
```

La Figura 15 muestra el ejemplo de la generación de los ejecutables con los comandos escritos anteriormente. Dentro del repositorio, se incluyen estos archivos compilados en la ruta `/MPI/bin`, como lo ilustra la Figura 16.

```

sistemas@manager2:/nfs/condor/nestorj_aparicioh$ pwd
/nfs/condor/nestorj_aparicioh
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ ls -ltr
total 164
-rw-rw-r-- 1 sistemas sistemas 4217 may  9 23:07 mpiMM1c.c
-rw-rw-r-- 1 sistemas sistemas 3142 may 10 11:07 mpiMM1f.c
-rw-rw-r-- 1 sistemas sistemas 3344 may 10 11:08 mpiMM1fu.c
-rw-rw-r-- 1 sistemas sistemas 3521 may 10 11:09 mpiMM2f.c
drwxrwxr-x 2 sistemas sistemas 4096 may 10 11:14 ref_OpenMP
-rwxrwxr-x 1 sistemas sistemas 17480 may 10 11:16 mpiMM1c
-rwxrwxr-x 1 sistemas sistemas 17448 may 10 11:16 mpiMM1f
-rwxrwxr-x 1 sistemas sistemas 17448 may 10 11:16 mpiMM1fu
-rwxrwxr-x 1 sistemas sistemas 17448 may 10 11:17 mpiMM2f

```

Figura 15 Ejemplo compilación fuentes algoritmos con MPI

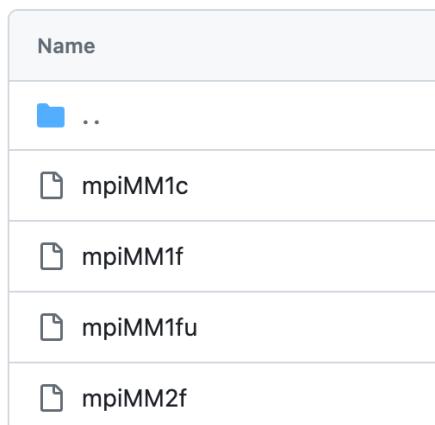


Figura 16 Repositorio. Ruta /MPI/bin

3. Pruebas locales de los ejecutables: Se realizan pruebas locales desde el nodo master para cada uno de los archivos recién compilados:

Prueba1: Ejecución local de cada algoritmo con 2 hilos y una matriz de 300*300. Ver Figura 17:

```

mpirun -np 2 mpiMM1c 300
mpirun -np 2 mpiMM1f 300
mpirun -np 2 mpiMM1fu 300
mpirun -np 2 mpiMM2f 300

```

```

sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun -np 2 mpiMM1c 300
approx 2-process time Tp: 0.029229 sec.
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun -np 2 mpiMM1f 300
total execution time: 0.034036
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun -np 2 mpiMM1fu 300
total execution time: 0.012194
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun -np 2 mpiMM2f 300
total execution time: 0.023012
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ █

```

Figura 17 Ejemplo ejecución Prueba1

Prueba2: Ejecución desde el manager2 a un nodo externo. 2 hilos, matriz de 300*300. Ver Figura 18:

```

mpirun --host worker12:2 ./mpiMM1c 300
mpirun --host worker12:2 ./mpiMM1f 300
mpirun --host worker12:2 ./mpiMM1fu 300
mpirun --host worker12:2 ./mpiMM2f 300

```

```

sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun --host worker12:2 ./mpiMM1c 300
approx 2-process time Tp: 0.045601 sec.
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun --host worker12:2 ./mpiMM1f 300
total execution time: 0.026489
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun --host worker12:2 ./mpiMM1fu 300
total execution time: 0.012539
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun --host worker12:2 ./mpiMM2f 300
total execution time: 0.023280
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ █

```

Figura 18 Ejemplo ejecución Prueba2

Prueba3: Ejecución desde el manager2 a **dos nodos externos**. 2 hilos, matriz de 300*300. Para los 4 ejecutables falló esta prueba. Ver Figura 19.

```

sistemas@manager2:/nfs/condor/nestorj_aparicioh$ mpirun --host worker5:2,worker4:2 ./mpiMM1c 300
-----
WARNING: Open MPI failed to TCP connect to a peer MPI process. This
should not happen.

```

Figura 19 Error Prueba3

Luego de realizar la revisión correspondiente se evidencia que los nodos tienen instalado Docker. Dada la virtualización de las tarjetas de red, el comando no logra establecer las conexiones TCP a los nodos del cluster.

Para solventar la situación, se incluyó el parámetro [1][2]

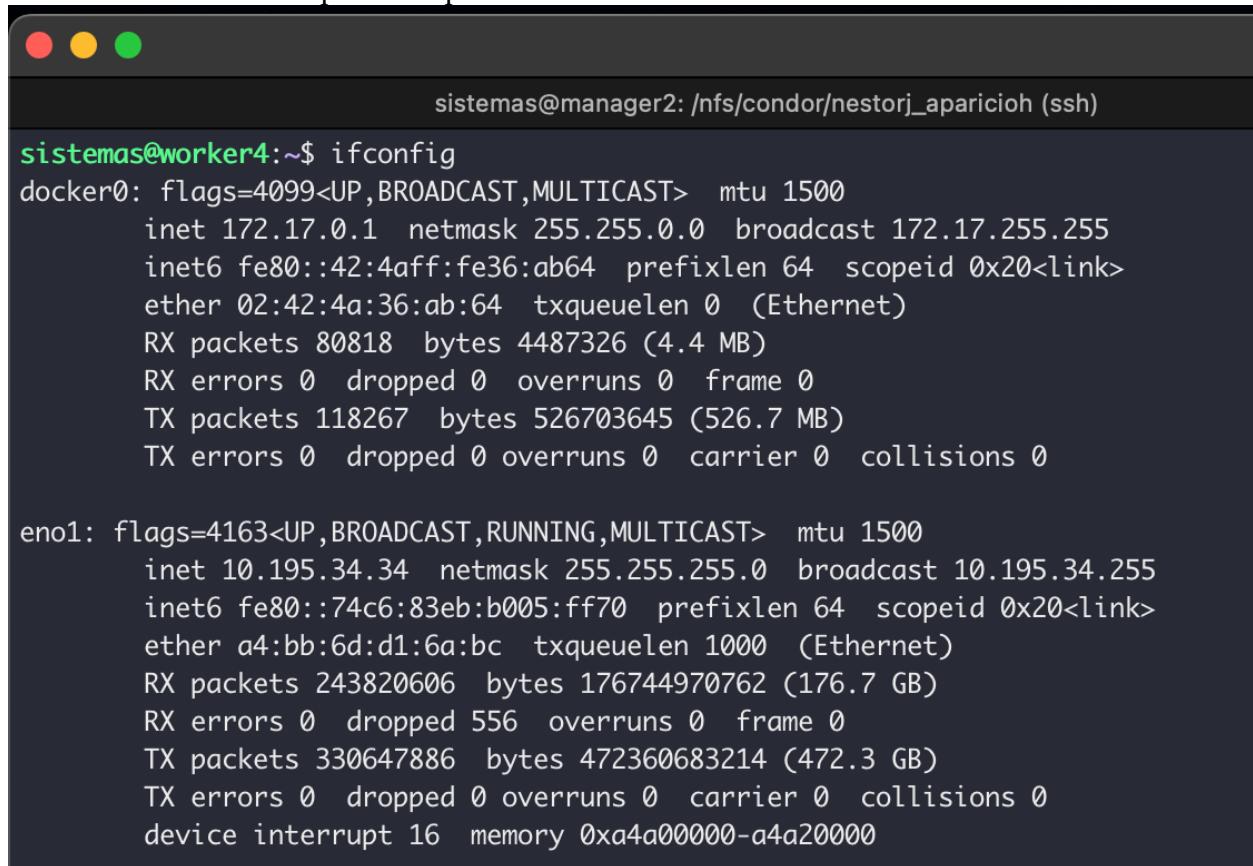
```
--mca btl_tcp_if_include eth0
```

Donde “eth0” corresponde a la interfaz de red que se va a utilizar en ambos nodos externos.

De esta manera el comando para poder hacer la ejecución de MPI en 2 nodos externos, a 2 hilos, para una matriz de 300*300 es el siguiente:

```
mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM1c 300
mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM1f 300
mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM1fu 300
mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM2f 300
```

Donde “eno1” corresponde a la interfaz de red a utilizar en los nodos “worker4” y “worker5”. Las Figuras 20 y 21 muestran la configuración de las tarjetas de red de los nodos utilizados para el experimento.



A terminal window titled "sistemas@manager2: /nfs/condor/nestorj_aparicioh (ssh)". The window shows the output of the "ifconfig" command for the "worker4" node. It lists two interfaces: "docker0" and "eno1". The "docker0" interface has an IP of 172.17.0.1 and is connected to a Docker container. The "eno1" interface has an IP of 10.195.34.34 and is connected to the external network. Both interfaces show high traffic volumes, with RX/TX counts in the thousands and gigabytes.

```
sistemas@worker4:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
              inet6 fe80::42:4aff:fe36:ab64 prefixlen 64 scopeid 0x20<link>
                ether 02:42:4a:36:ab:64 txqueuelen 0 (Ethernet)
                  RX packets 80818 bytes 4487326 (4.4 MB)
                  RX errors 0 dropped 0 overruns 0 frame 0
                  TX packets 118267 bytes 526703645 (526.7 MB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.195.34.34 netmask 255.255.255.0 broadcast 10.195.34.255
              inet6 fe80::74c6:83eb:b005:ff70 prefixlen 64 scopeid 0x20<link>
                ether a4:bb:6d:d1:6a:bc txqueuelen 1000 (Ethernet)
                  RX packets 243820606 bytes 176744970762 (176.7 GB)
                  RX errors 0 dropped 556 overruns 0 frame 0
                  TX packets 330647886 bytes 472360683214 (472.3 GB)
                  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
                device interrupt 16 memory 0xa4a00000-a4a20000
```

Figura 20 Configuración tarjeta de red nodo Worker4

```
sistemas@worker5:~$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
        inet 172.17.0.1 netmask 255.255.0.0 broadcast 172.17.255.255
        inet6 fe80::42:dcff:fe4c:c6c7 prefixlen 64 scopeid 0x20<link>
          ether 02:42:dc:4c:c6:c7 txqueuelen 0 (Ethernet)
            RX packets 79716 bytes 4427977 (4.4 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 116270 bytes 526582955 (526.5 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.195.34.33 netmask 255.255.255.0 broadcast 10.195.34.255
        inet6 fe80::f2d6:8b36:9db1:843 prefixlen 64 scopeid 0x20<link>
          ether a4:bb:6d:d1:4e:8c txqueuelen 1000 (Ethernet)
            RX packets 415288304 bytes 489042339012 (489.0 GB)
            RX errors 0 dropped 540 overruns 0 frame 0
            TX packets 159192012 bytes 159626501568 (159.6 GB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
            device interrupt 16 memory 0xa4a00000-a4a20000
```

Figura 21 Configuración tarjeta de red nodo Worker5

Se realizó nuevamente la ejecución de los comandos con el parámetro `--mca btl_tcp_if_include eno1` y la salida se generó sin errores.

4 . Archivos para lanzar el experimento:

parametrosMPI.txt:

Se procede a crear el lanzador para la ejecución de los ejecutables en MPI. El programa, de manera similar al experimento con OpenMP, estará dividido en 2 partes. Inicialmente un archivo `.txt` que contendrá los parámetros de ejecución del experimento a nivel de hilos, tamaño de matriz y algoritmo a ejecutar:

Para la ejecución del experimento se utilizó la configuración que se ilustra en la Figura 22. El archivo debe ser nombrado como `parametrosMPI.txt`:

```
drwxrwxr-x 2 sistemas sistemas 12288 may 10 21:29 MM1c
sistemas@manager2:/nfs/condor/nestorj_aparicioh$ cat parametrosMPI.txt
threads=(2 4 6 8 10 12 14)
matrixes=(100 200 300 400 500 600 700 800 900 1000 1200 1400 1600 1800 2000 2400 2800 3000 3500 4000 4500 5000 5500 6000 6500 7000 7500)
programs=("MM1c")
sistemas@manager2:/nfs/condor/nestorj_aparicioh$
```

Figura 22 Ejemplo archivo `parametrosMPI.txt`

El archivo recibe 3 parámetros:

- Threads:** Corresponde a la cantidad de hilos a utilizar, se cargan como un arreglo, separados por espacios.

- b. **Matrixes:** Se ingresa la lista del tamaño de matrices que se desea procesar.
- c. **Programs:** Lista el o los programas que se correrán.

```
threads=(2 4 6 8 10 12 14)
matrixes=(100 200 300 400 500 600 700 800 900 1000 1200 1400 1600 1800 2000 2400 2800
3000 3500 4000 4500 5000 5500 6000 6500 7000 7500)
programs=("MM1c")
```

lanzadorMPI.sh

El archivo lanzador será el encargado de ejecutar las iteraciones configuradas en el archivo `parametrosMPI.txt` para llevar a cabo el experimento. En la Figura 23 se muestra el contenido del archivo `lanzadorMPI.sh`:

```
1#!/bin/bash
2
3 # Execution Parameters
4 source parametrosMPI.txt
5
6 # Iteration executions
7 for ((i=0; i<10; i++));
8 do
9 echo "Main FOR - Execute i = ${i}"
10    for thread in "${threads[@]}";
11    do
12      echo "Thread - Execute Thread = ${thread}"
13        for matrix in "${matrixes[@]}";
14        do
15          echo "Matrix - Execute matrix = ${matrix} thread = ${thread}"
16            for program in "${programs[@]}";
17            do
18              # Commands
19              filename="MPI-${program}-${matrix}-TH-${thread}.txt"
20              echo "Execute MPI-${program}-${matrix}-TH-${thread}"
21              mpirun --mca btl_tcp_if_include eno1 --host worker4:${thread},worker5:${thread} mpirun ${program} ${matrix} >> ${program}/${filename}
22              #Comando de referencia:
23              #Se debe incluir el parametro --mca btl_tcp_if_include eno1 para
24              #restringir el socket de red que se puede usar, pues el proceso
25              #estaba evaluando todos, incluido el de docker instalado
26              #recientemente.
27              #mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM1c 300
28
29              #echo "Ejecutando $cmd"
30              #$cmd > "salida_${program}_${thread}_${matrix}.txt"
31            done
32          done
33      done
34 done
35 echo "Todas las ejecuciones han finalizado."
```

Figura 23 Archivo `lanzadorMPI.sh`

```
#!/bin/bash

# Execution Parameters
source parametrosMPI.txt

# Iteration executions
for ((i=0; i<10; i++));
do
```

```

echo "Main FOR - Execute i = ${i}"
for thread in "${threads[@]}";
do
echo "Thread - Execute Thread = ${thread}"
for matrix in "${matrixes[@]}";
do
echo "Matrix - Execute matrix = ${matrix} thread = ${thread}"
for program in "${programs[@]}";
do
# Commands
filename="MPI-${program}-${matrix}-TH-${thread}.txt"
echo "Execute MPI-${program}-${matrix}-TH-${thread}"
mpirun --mca btl_tcp_if_include eno1 --host
worker4:${thread},worker5:${thread} mpi${program} ${matrix} >> ${program}/${filename}
#Comando de referencia:
#Se debe incluir el parametro --mca btl_tcp_if_include eno1 para
#restringir el socket de red que se puede usar, pues el proceso
#estaba evaluando todos, incluido el de docker instalado
#recientemente.
#mpirun --mca btl_tcp_if_include eno1 --host worker4:2,worker5:2 ./mpiMM1c
300

#echo "Ejecutando $cmd"
#${cmd} > "salida_${program}_${thread}_${matrix}.txt"
done
done
done
done
echo "Todas las ejecuciones han finalizado."

```

El archivo sh funciona de la siguiente manera:

```

1#!/bin/bash
2
3# Execution Parameters

--Se carga el archivo parametrosMPI.txt con las variables que utilizará la
ejecución:
4 source parametrosMPI.txt
5
6# Iteration executions

--Se inicializa un for para contar las iteraciones del experimento. Para este
ejemplo con 10 iteraciones:
7 for ((i=0; i<10; i++));
8 do
9 echo "Main FOR - Execute i = ${i}"

--Se realiza lectura del arreglo de hilos a utilizar

```

```

10      for thread in "${threads[@]}";
11      do
12          echo "Thread - Execute Thread = ${thread}"

--Se realiza lectura del arreglo de tamaños de matrices a utilizar
13          for matrix in "${matrixes[@]}";
14          do
15              echo "Matrix - Execute matrix = ${matrix} thread = ${thread}"

--Se realiza lectura del arreglo de programas a utilizar
16          for program in "${programs[@]}";
17          do
18              # Commands
19              filename="MPI-${program}-${matrix}-TH-${thread}.txt"
20              echo "Execute MPI-${program}-${matrix}-TH-${thread}"

--Se ejecuta el comando mpi:
21          mpirun --mca btl_tcp_if_include eno1 --host
worker4:${thread},worker5:${thread} mpi${program} ${matrix} >>
${program}/${filename}
22          #Comando de referencia:
23          #Se debe incluir el parametro --mca btl_tcp_if_include eno1
para
24          #restringir el socket de red que se puede usar, pues el
proceso
25          #estaba evaluando todos, incluido el de docker instalado
26          #recientemente.
27          #mpirun --mca btl_tcp_if_include eno1 --host
worker4:2,worker5:2 ./mpiMM1c 300
28
29          #echo "Ejecutando $cmd"
30          #$cmd > "salida_${program}_${thread}_${matrix}.txt"
31          done
32      done
33  done
34 done
--Finalizado el ciclo de ejecución del comando mpi por programa * tamaño de
matrices * cantidad de hilos * iteraciones del experimento, se imprime este
mensaje al log
35 echo "Todas las ejecuciones han finalizado."

```

Nota: Para la ejecución del experimento con MPI utilizaremos recursos distribuidos entre los nodos del cluster worker 4 y worker5, como lo indica el comando del archivo lanzadorMPI.sh visto anteriormente:

```

mpirun      --mca      btl_tcp_if_include      eno1      --host
worker4:${thread},worker5:${thread}      mpi${program}      ${matrix}      >>
${program}/${filename}

```

Los archivos parametrosMPI.txt y lanzadorMPI.sh utilizados para el experimento, se incluyen dentro del repositorio en la ruta /MPI/tolos

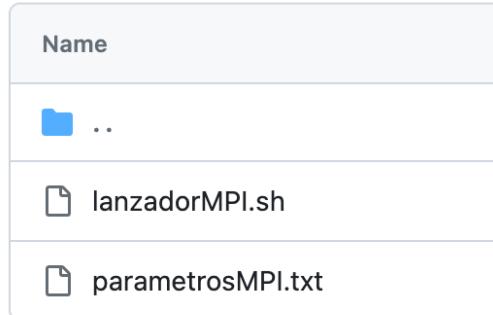


Figura 24 Repositorio /MPI/tools

5. Ejecución del experimento:

Previo a la ejecución del experimento, se crea una carpeta para cada algoritmo. Dentro de cada carpeta se copia, el ejecutable del algoritmo correspondiente, el archivo de parámetros y el archivo del lanzador, como lo ilustra la Figura 25:

```
sistemas@manager2:/nfs/condor/nestorj_aparicioh/MM1c$ ls -ltr
total 568
-rwxrwxr-x 1 sistemas sistemas 17480 may 10 20:48 mpiMM1c
-rwxrwxr-x 1 sistemas sistemas 1281 may 10 20:54 lanzadorMPI.sh
-rw-rw-r-- 1 sistemas sistemas 137 may 10 21:05 parametrosMPI.txt
```

Figura 25 Ejemplo directorio para ejecución MPI MM1c

La figura 26 ilustra la definición del archivo `parametrosMPI.txt`. Se hizo una ejecución llamando varios algoritmos desde el mismo proceso, pero para aprovechar los recursos de los nodos worker, para cada algoritmo, el archivo de parámetros se creó de manera individual.

```
1 threads=(2 4 6 8 10 12 14)
2 matrixes=(100 200 300 400 500 600 700 800 900 1000 1200 1400 1600 1800 2000 2400 2800 3000)
3 programs=("MM1c")
```

Figura 26 Ejemplo `parametrosMPI.txt` para algoritmo MM1c

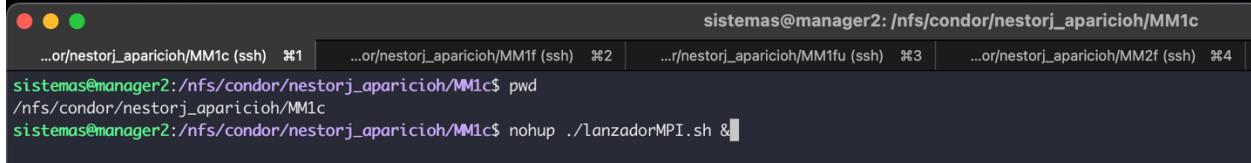
Se realiza la prueba de cada algoritmo ejecutando el siguiente comando:

```
./lanzadorMPI.sh
```

El comando anterior, ejecuta el proceso y mostrará su avance en la terminal de la sesión desde la que se ejecute, sin embargo, si la sesión se cierra, el experimento finalizará su ejecución. Para las ejecuciones de extensa duración como la propuesta en para este

experimento, se sugiere al igual que para OpenMP, utilizar el nohup para que la ejecución se haga en segundo plano y continue su tarea aún cuando la sesión desde la que se lanzó la instrucción haya finalizado (ejemplo en la Figura 27). De esta manera, el comando utilizado fue:

```
nohup ./lanzadorMPI.sh &
```



A terminal window titled 'sistemas@manager2: /nfs/condor/nestorj_aparicioh/MM1c'. It shows a command history with four entries. The first three entries are standard shell commands: '...or/... (ssh) %1', '...or/... (ssh) %2', and '...r/... (ssh) %3'. The fourth entry is '...or/... (ssh) %4' which contains the command 'nohup ./lanzadorMPI.sh &'. The prompt 'sistemas@manager2:' is visible at the bottom.

Figura 27 Ejecución lanzadorMPI.sh

Una vez lanzado el proceso con el comando nohup, generará un archivo nohup.out que contendrá el log de ejecución del comando ejecutado. De ser requerido, mientras se procesa el experimento, se puede hacer seguimiento del avance del proceso usando el comando tail para verificar el contenido del archivo nohup.out.

```
tail -100f nohup.out
```

El proceso de MPI al utilizar memoria compartida, nos permite ver en los nodos configurados para realizar la ejecución del proceso (en nuestro experimento el **worker4** y **worker5**) el consumo de CPU's durante la ejecución del proceso. La figura 28 muestra al **worker4**, ejecutando el proceso MPI lanzado desde el nodo master del cluster:

```
...manager2: /nfs/condor/nestorj_aparicio/MM2f (ssh) ● ❶ sistemas@manager2: ~ (ssh) ❷ sistemas@worker4: ~ (ssh)
top - 08:49:16 up 8 days, 14:10, 2 users, load average: 10,52, 5,07, 3,36
Tasks: 398 total, 17 running, 381 sleeping, 0 stopped, 0 zombie
%CPU0 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU1 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU2 : 6,3 us, 0,3 sy, 0,0 ni, 93,4 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU3 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU4 : 9,6 us, 1,0 sy, 0,0 ni, 89,4 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU5 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU6 : 69,3 us, 0,3 sy, 0,0 ni, 30,3 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU7 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU8 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU9 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU10 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU11 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU12 : 95,0 us, 0,0 sy, 0,0 ni, 5,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU13 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU14 : 94,7 us, 0,0 sy, 0,0 ni, 5,0 id, 0,0 wa, 0,0 hi, 0,3 si, 0,0 st
%CPU15 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU16 : 31,1 us, 0,3 sy, 0,0 ni, 68,5 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU17 : 99,7 us, 0,3 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU18 : 0,7 us, 0,3 sy, 0,0 ni, 98,3 id, 0,7 wa, 0,0 hi, 0,0 si, 0,0 st
%CPU19 :100,0 us, 0,0 sy, 0,0 ni, 0,0 id, 0,0 wa, 0,0 hi, 0,0 si, 0,0 st
MiB Mem : 64008,7 total, 37741,4 free, 4297,5 used, 21969,8 buff/cache
MiB Swap: 3816,0 total, 3816,0 free, 0,0 used. 58856,9 avail Mem


```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1220724	sistemas	20	0	1829264	94352	10852	R	100,3	0,1	0:05.66	mpiMM1fu
1220733	sistemas	20	0	1829264	95508	11076	R	100,3	0,1	0:05.65	mpiMM1fu
1220735	sistemas	20	0	1829264	93556	10716	R	100,3	0,1	0:05.66	mpiMM1fu
1220815	sistemas	20	0	1758980	98560	7792	R	100,3	0,2	0:03.66	mpiMM1c
1220719	sistemas	20	0	1829408	224740	10868	R	100,0	0,3	0:05.65	mpiMM1fu
1220720	sistemas	20	0	1829264	95524	11092	R	100,0	0,1	0:05.65	mpiMM1fu
1220721	sistemas	20	0	1829264	94228	10852	R	100,0	0,1	0:05.65	mpiMM1fu
1220722	sistemas	20	0	1829264	94240	11004	R	100,0	0,1	0:05.65	mpiMM1fu
1220723	sistemas	20	0	1829264	94296	10928	R	100,0	0,1	0:05.65	mpiMM1fu
1220726	sistemas	20	0	1829264	94236	10864	R	100,0	0,1	0:05.65	mpiMM1fu
1220730	sistemas	20	0	1829264	95620	10928	R	100,0	0,1	0:05.65	mpiMM1fu
1220737	sistemas	20	0	1829264	95424	11000	R	100,0	0,1	0:05.65	mpiMM1fu
1220739	sistemas	20	0	1829264	94148	11028	R	100,0	0,1	0:05.65	mpiMM1fu
1220812	sistemas	20	0	1758988	220972	7700	R	100,0	0,3	0:03.66	mpiMM1c
1220813	sistemas	20	0	1758980	98532	7760	R	100,0	0,2	0:03.66	mpiMM1c
1220814	sistemas	20	0	1758980	98524	7764	R	100,0	0,2	0:03.65	mpiMM1c
1152	avahi	20	0	13992	9504	3428	S	4,7	0,0	223:15.93	avahi-daemon
1220734	sistemas	20	0	2073244	181596	62424	S	1,2	0,2	29:26.92	avahiddaemon

Figura 28 Ejecución de proceso MPI desde el worker4

Finalizada la ejecución, el contenido del archivo `nohup.out` contendrá el listado de archivos generados. El archivo `nohup.out`, se crea en la ruta desde la cual se haya lanzado el comando. La figura 29 ilustra el contenido de la carpeta del algoritmo MM1c del experimento ejecutado:

```
sistemas@manager2:/nfs/condor/nestorj_aparicioh/MM1c ls
MPI-MM1c-1200-TH-10.txt MPI-MM1c-1200-TH-2.txt MPI-MM1c-2000-TH-14.txt MPI-MM1c-2800-TH-6.txt MPI-MM1c-400-TH-12.txt MPI-MM1c-600-TH-4.txt MPI-MM1c-900-TH-10.txt
MPI-MM1c-100-TH-12.txt MPI-MM1c-100-TH-4.txt MPI-MM1c-1800-TH-10.txt MPI-MM1c-2000-TH-2.txt MPI-MM1c-2800-TH-8.txt MPI-MM1c-400-TH-14.txt MPI-MM1c-600-TH-6.txt MPI-MM1c-900-TH-12.txt
MPI-MM1c-100-TH-14.txt MPI-MM1c-1200-TH-6.txt MPI-MM1c-1800-TH-12.txt MPI-MM1c-2000-TH-4.txt MPI-MM1c-300-TH-10.txt MPI-MM1c-400-TH-2.txt MPI-MM1c-600-TH-8.txt MPI-MM1c-900-TH-14.txt
MPI-MM1c-100-TH-2.txt MPI-MM1c-1200-TH-8.txt MPI-MM1c-1800-TH-14.txt MPI-MM1c-2000-TH-6.txt MPI-MM1c-300-TH-12.txt MPI-MM1c-400-TH-4.txt MPI-MM1c-700-TH-10.txt MPI-MM1c-900-TH-2.txt
MPI-MM1c-100-TH-4.txt MPI-MM1c-1400-TH-10.txt MPI-MM1c-1800-TH-2.txt MPI-MM1c-2000-TH-8.txt MPI-MM1c-300-TH-14.txt MPI-MM1c-400-TH-6.txt MPI-MM1c-700-TH-12.txt MPI-MM1c-900-TH-4.txt
MPI-MM1c-100-TH-6.txt MPI-MM1c-1400-TH-12.txt MPI-MM1c-1800-TH-4.txt MPI-MM1c-2400-TH-10.txt MPI-MM1c-300-TH-2.txt MPI-MM1c-400-TH-8.txt MPI-MM1c-700-TH-14.txt MPI-MM1c-900-TH-6.txt
MPI-MM1c-100-TH-8.txt MPI-MM1c-1400-TH-14.txt MPI-MM1c-1800-TH-6.txt MPI-MM1c-2000-TH-12.txt MPI-MM1c-300-TH-4.txt MPI-MM1c-500-TH-10.txt MPI-MM1c-700-TH-2.txt MPI-MM1c-900-TH-8.txt
MPI-MM1c-1000-TH-10.txt MPI-MM1c-1400-TH-2.txt MPI-MM1c-1800-TH-8.txt MPI-MM1c-2000-TH-14.txt MPI-MM1c-300-TH-6.txt MPI-MM1c-500-TH-12.txt MPI-MM1c-700-TH-4.txt MPI-MM1c-900-TH-10.txt
MPI-MM1c-1000-TH-12.txt MPI-MM1c-1400-TH-4.txt MPI-MM1c-200-TH-10.txt MPI-MM1c-2400-TH-2.txt MPI-MM1c-300-TH-8.txt MPI-MM1c-500-TH-14.txt MPI-MM1c-700-TH-6.txt MPI-MM1c-900-TH-12.txt
MPI-MM1c-1000-TH-14.txt MPI-MM1c-1400-TH-6.txt MPI-MM1c-200-TH-12.txt MPI-MM1c-2400-TH-4.txt MPI-MM1c-300-TH-10.txt MPI-MM1c-500-TH-2.txt MPI-MM1c-500-TH-8.txt MPI-MM1c-700-TH-10.txt
MPI-MM1c-1000-TH-2.txt MPI-MM1c-1400-TH-8.txt MPI-MM1c-200-TH-14.txt MPI-MM1c-2400-TH-6.txt MPI-MM1c-300-TH-12.txt MPI-MM1c-500-TH-4.txt MPI-MM1c-800-TH-10.txt MPI-MM1c-900-TH-12.txt
MPI-MM1c-1000-TH-4.txt MPI-MM1c-1600-TH-10.txt MPI-MM1c-200-TH-2.txt MPI-MM1c-2400-TH-8.txt MPI-MM1c-300-TH-14.txt MPI-MM1c-500-TH-6.txt MPI-MM1c-800-TH-12.txt MPI-MM1c-900-TH-14.txt
MPI-MM1c-1000-TH-6.txt MPI-MM1c-1600-TH-12.txt MPI-MM1c-200-TH-4.txt MPI-MM1c-2800-TH-10.txt MPI-MM1c-300-TH-2.txt MPI-MM1c-500-TH-8.txt MPI-MM1c-800-TH-14.txt MPI-MM1c-900-TH-16.txt
MPI-MM1c-1000-TH-8.txt MPI-MM1c-1600-TH-14.txt MPI-MM1c-200-TH-6.txt MPI-MM1c-2800-TH-12.txt MPI-MM1c-3000-TH-4.txt MPI-MM1c-600-TH-10.txt MPI-MM1c-800-TH-2.txt MPI-MM1c-900-TH-18.txt
MPI-MM1c-1200-TH-10.txt MPI-MM1c-1600-TH-2.txt MPI-MM1c-200-TH-8.txt MPI-MM1c-2800-TH-14.txt MPI-MM1c-3000-TH-6.txt MPI-MM1c-600-TH-12.txt MPI-MM1c-800-TH-4.txt MPI-MM1c-900-TH-20.txt
MPI-MM1c-1200-TH-12.txt MPI-MM1c-1600-TH-4.txt MPI-MM1c-2000-TH-10.txt MPI-MM1c-2800-TH-2.txt MPI-MM1c-3000-TH-8.txt MPI-MM1c-600-TH-14.txt MPI-MM1c-800-TH-6.txt MPI-MM1c-900-TH-22.txt
MPI-MM1c-1200-TH-14.txt MPI-MM1c-1600-TH-6.txt MPI-MM1c-2000-TH-12.txt MPI-MM1c-2800-TH-4.txt MPI-MM1c-400-TH-10.txt MPI-MM1c-600-TH-2.txt MPI-MM1c-800-TH-8.txt MPI-MM1c-900-TH-24.txt
sistemas@manager2:/nfs/condor/nestorj_aparicioh/MM1c [
```

Figura 29 Ejemplo salidas experimento MPI MM1c

6. Revisión de resultados:

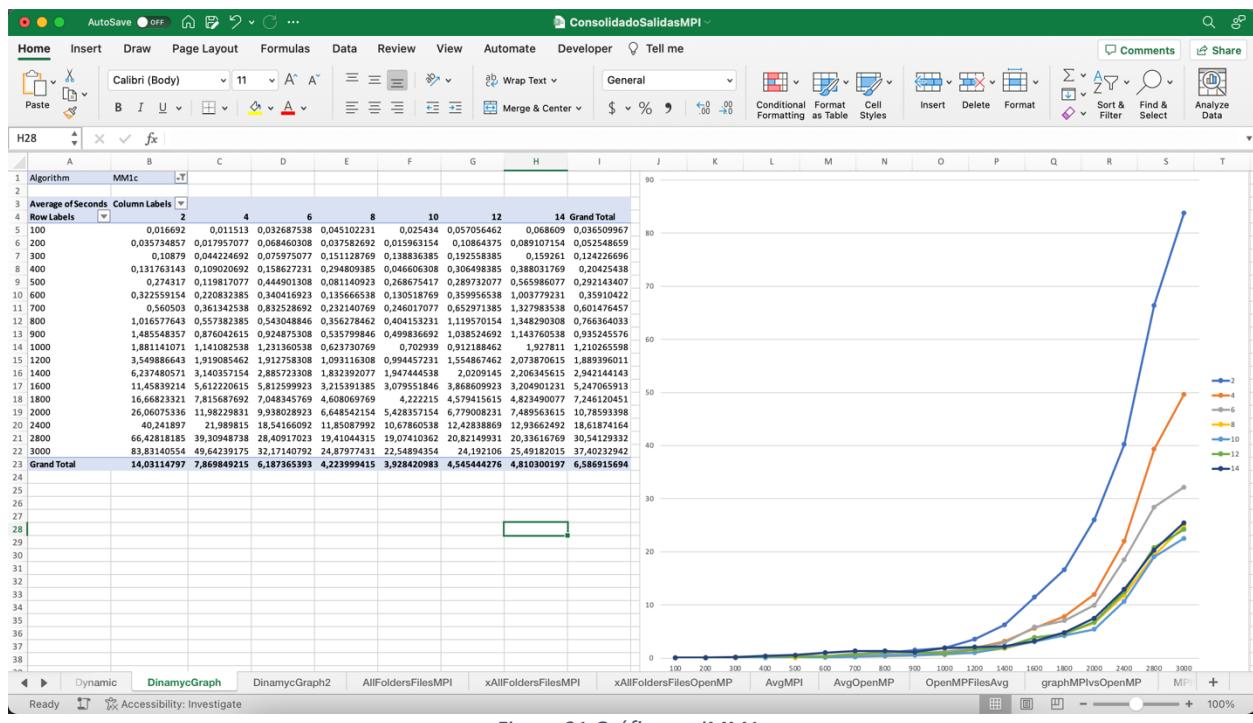
Dentro de cada directorio de algoritmo, Por cada combinación de Algoritmo-TamañoMatriz-CantidadHilos que se haya configurado en el archivo parametrosMPI.txt, se crea un archivo con el resultado de cada una de las ejecuciones o cantidad de experimentos configurados en el mismo archivo parametrosMPI.txt tal como se pudo apreciar en la figura 29.

7. Tabulación de los resultados:

A modo de referencia dentro del repositorio se incluye el directorio /MPI/salidas y el archivo ConsolidadoSalidasMPI.xlsx, los cuales corresponden archivos generados como salida del experimento y la tabulación de esta información respectivamente. Se ilustra la ubicación de los archivos y el directorio en la Figura 30. La Figura 31 por su parte, muestra una de las gráficas obtenidas luego de graficar los resultados del algoritmo mpiMM1c. Haciendo uso del archivo ConsolidadoSalidasMPI.xlsx, se puede validar la información para los demás algoritmos, así como generar las gráficas que el usuario desee en función a los datos recolectados en el experimento.

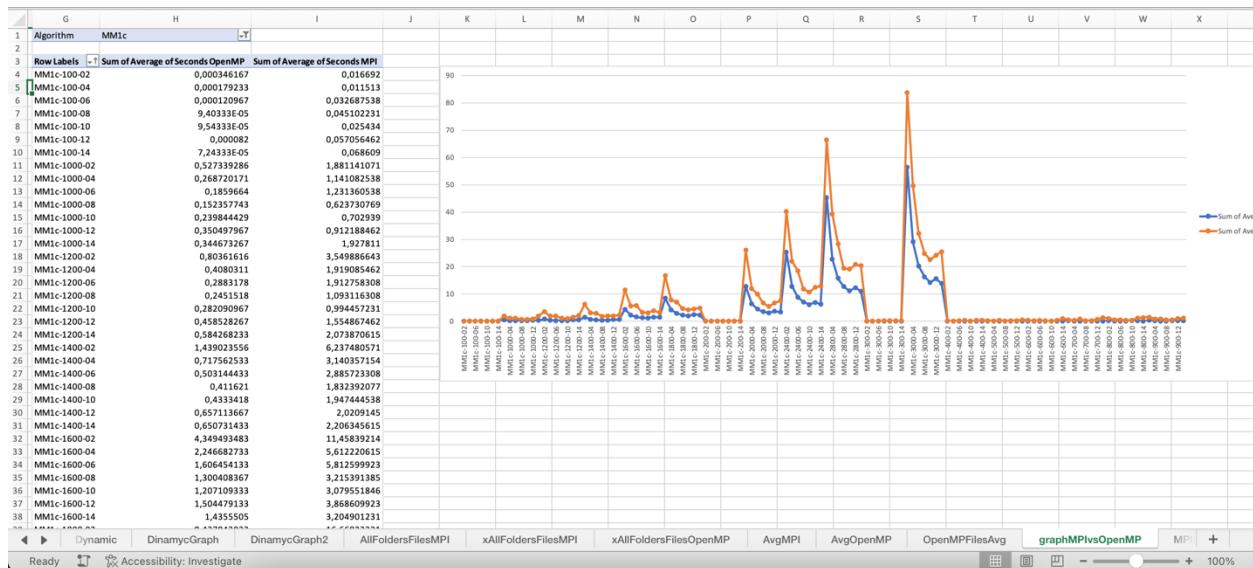
Name
..
Salidas
bin
src
tools
.DS_Store
ConsolidadoSalidasMPI.xlsx

Figura 30 Repositorio Directorio /MPI



Comparativa MPI vs OpenMP

El detalle de la comparativa, los conceptos y el análisis y las conclusiones referenciadas en el presente anexo los podrá encontrar en el archivo **Evaluación Rendimiento Informe.pdf** y **Evaluación Rendimiento Presentación.pdf** incluidos en el repositorio en la ruta /Files, a continuación la Figura 32 muestra a modo de ejemplo la comparativa obtenida entre OpenMP y MPI para el algoritmo MM1c:



Referencias:

1. <https://github.com/open-mpi/ompi/blob/main/opal/mca/btl/tcp/help-mpi-btl-tcp.txt#L116>
2. <https://www.mail-archive.com/users@lists.open-mpi.org/msg34182.html>