Curso Computación de Alto Rendimiento

Taller Evaluación de Rendimiento

Sistemas Operativos

Daniel Felipe Castro Moreno

Eliana Katherine Cepeda González

María Paula Rodríguez Ruiz

Daniel Horacio González Orduz

Profesor:

John Corredor Franco

Dpto. de Ingeniería de Sistemas

Pontificia Universidad Javeriana

Bogotá D.C.

5 de noviembre del 2024

Contenido

Introducción

El presente informe tiene el propósito de comparar la ejecución de un algoritmo de multiplicación de matrices (método clásico y método transpuesto) tomándolo como benchmark con el fin de comparar diferentes sistemas de cómputo y analizar e interpretar los resultados para extraer recomendaciones y conclusiones.

Esto se realizará a través de distintas secciones, en las que primero se contrastarán los componentes de tres máquinas virtuales gracias a la ejecución de ciertos comandos, luego en otro se evidenciará el copiado de archivos de los algoritmos de multiplicación desde el host a ellas, para posteriormente ejecutarlos a gran escala de forma automática mediante un script en Pearl para tomar los diferentes tiempos de ejecución en una serie de casos de prueba descritos a continuación, resultando en datos que fueron recopilados en una hoja de cálculo para su posterior graficación y análisis mediante el uso de dos funciones principales de probabilidad como lo son el promedio y la desviación estándar de forma que su interpretación permitiera extraer las debidas conclusiones.

Comparación de máquinas virtuales

Para la elaboración del taller, se eligieron 3 máquinas virtuales asignadas por la universidad y se nombrarán máquina 1, 2 y 3, en orden de aparición. La máquina 1, cuya IP es 10.43.100.153, tiene las siguientes características:

- **Procesadores:** 4 núcleos, cada uno con un hilo y un socket.
- **Sistema operativo:** Ubuntu - Debian
- **Memoria caché:**
    - L1d: 4 instancias de 128 KiB cada una.
    - L1i: 4 instancias de 128 KiB cada una.
    - L2: 4 instancias de 1 MiB cada una.
    - L3: 4 instancias de 120 MiB cada una.

- **Memoria RAM:** 11914.2 MiB.
- **Arquitectura:** Soporte para 32 y 64 bits, con direccionamiento de 43 bits para la memoria física y 48 bits para la memoria virtual. El procesador utilizado es un Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, modelo 79.
- **Almacenamiento persistente:** 59 GB.

Por otro lado, la máquina 2 (10.43.103.160), tiene las siguientes características:

- **Procesadores:** 4 núcleos, cada uno con un hilo y un socket.
- **Sistema operativo:** Ubuntu – Noble Numbat.
- **Memoria caché:**
  - L1d: 4 instancias de 128 KiB cada una.
  - L1i: 4 instancias de 128 KiB cada una.
  - L2: 4 instancias de 1 MiB cada una.
  - L3: 4 instancias de 120 MiB cada una.

- **Memoria RAM:** 11914.2 MiB.
- **Arquitectura:** Soporte para 32 y 64 bits, con direccionamiento de 43 bits para la memoria física y 48 bits para la memoria virtual. El procesador utilizado es un Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz, modelo 79.
- **Almacenamiento persistente:** 59 GB.

Por último, la máquina 3 (10.43.103.218), tiene las siguientes características:

- **Procesadores:** 4 núcleos, cada uno con un hilo y un socket.
- **Sistema operativo:** Ubuntu - Noble Numbat
- **Memoria caché:**
  - L1d: 4 instancias de 128 KiB cada una.
  - L1i: 4 instancias de 128 KiB cada una.
  - L2: 4 instancias de 4 MiB cada una.
  - L3: 4 instancias de 66 MiB cada una.

- **Memoria RAM:** 11914.2 MiB.
- **Arquitectura:** Soporte para 32 y 64 bits, con direccionamiento de 43 bits para la memoria física y 48 bits para la memoria virtual. El procesador utilizado es un Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz, modelo 79.
- **Almacenamiento persistente:** 59 GB.

Análisis preliminar

Realizando la comparación entre las máquinas virtuales, las principales diferencias radican en sus configuraciones de memoria caché y procesador. La máquina 1 y la máquina 2 tienen configuraciones idénticas en cuanto a procesadores, memoria caché y almacenamiento, lo cual las hace igualmente adecuadas para tareas de procesamiento intensivo y almacenamiento de datos moderado. La máquina 3, aunque con una memoria RAM similar, presenta una capacidad menor en memoria caché L2 y L3 y, si bien su

procesador es el modelo más reciente, no representa una ventaja significativa para aplicaciones de alto rendimiento con el volumen de datos esperado en este taller.

Por lo anteriormente mencionado, para ejecutar eficientemente el algoritmo de multiplicación de matrices en distintas condiciones, se estima que la máquina 1 o 2 son las mejores opciones, ya que su configuración de caché superior permite un acceso rápido a los datos y una mayor velocidad en la realización de los cálculos. En contraste, la máquina 3 resulta menos ideal, ya que su menor caché L3 podría limitar su rendimiento en operaciones repetitivas de gran volumen de datos.

A continuación, se presentan los comandos empleados para las máquinas virtuales en su respectivo orden.

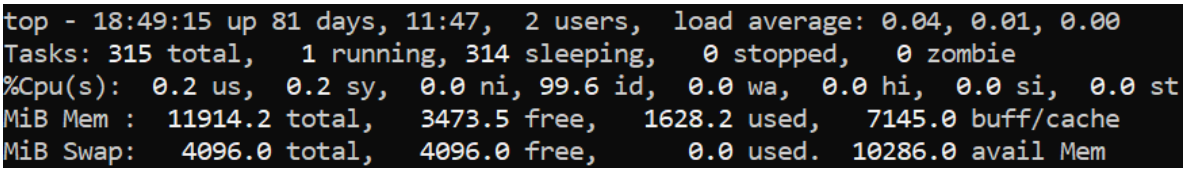

Figura 1. Máquina 1 - top.

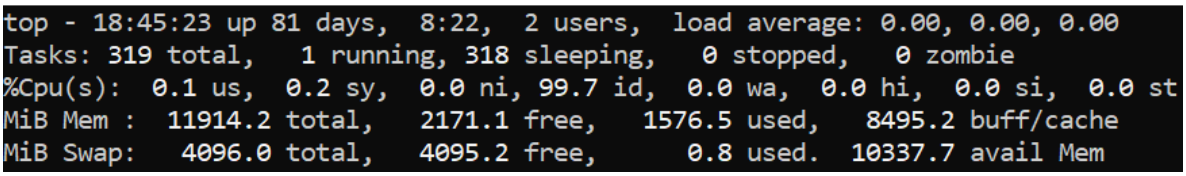

Figura 2. Máquina 2 - top.

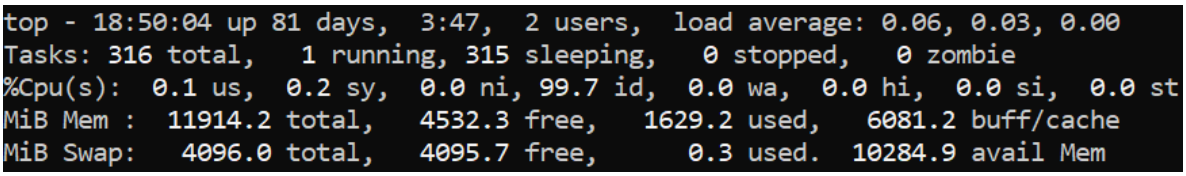

Figura 3. Máquina 3 - top.

```
estudiante@ing-gen64:~/Documents/taller_alto_rendimiento$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          43 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   4
  On-line CPU(s) list:    0-3
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
    CPU family:           6
    Model:                79
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            4
    Stepping:             0
    BogoMIPS:             4399.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic se
                          h_perfmon nopl xtopology tsc_reliable nons
                          dline_timer aes xsave avx f16c rdrand hype
                           bmi2 invpcid rdseed adx smap xsaveopt ara
Virtualization features:
  Hypervisor vendor:      VMware
  Virtualization type:    full
Caches (sum of all):
  L1d:                    128 KiB (4 instances)
  L1i:                    128 KiB (4 instances)
  L2:                     1 MiB (4 instances)
  L3:                     120 MiB (4 instances)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0-3
```

Figura 4. Máquina 1 - lscpu.

```
estudiante@ing-gen71:~$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          43 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   4
  On-line CPU(s) list:    0-3
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz
    CPU family:           6
    Model:                79
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            4
    Stepping:             0
    BogoMIPS:             4399.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic se
                          h_perfmon nopl xtopology tsc_reliable nons
                          dline_timer aes xsave avx f16c rdrand hype
                           bmi2 invpcid rdseed adx smap xsaveopt ara
Virtualization features:
  Hypervisor vendor:      VMware
  Virtualization type:    full
Caches (sum of all):
  L1d:                    128 KiB (4 instances)
  L1i:                    128 KiB (4 instances)
  L2:                     1 MiB (4 instances)
  L3:                     120 MiB (4 instances)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0-3
```

Figura 5. Máquina 2 - lscpu.

```
estudiante@ing-gen129:~$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          43 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   4
  On-line CPU(s) list:    0-3
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz
    CPU family:           6
    Model:                79
    Thread(s) per core:   1
    Core(s) per socket:   1
    Socket(s):            4
    Stepping:             0
    BogoMIPS:             4389.68
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic se
                          h_perfmon nopl xtopology tsc_reliable nons
                          dline_timer aes xsave avx f16c rdrand hype
                           bmi2 invpcid rdseed adx smap xsaveopt ara
Virtualization features:
  Hypervisor vendor:      VMware
  Virtualization type:    full
Caches (sum of all):
  L1d:                    128 KiB (4 instances)
  L1i:                    128 KiB (4 instances)
  L2:                     4 MiB (4 instances)
  L3:                     66 MiB (4 instances)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0-3
```

Figura 6. Máquina 3 - lscpu.

```
estudiante@ing-gen64:~/Documents/taller_alto_rendimiento$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           1.2G  2.0M  1.2G   1% /run
/dev/sda2        79G   16G   59G  21% /
tmpfs           5.9G     0  5.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           5.9G  220K  5.9G   1% /run/qemu
tmpfs           1.2G  100K  1.2G   1% /run/user/120
tmpfs           1.2G   84K  1.2G   1% /run/user/1000
```

Figura 7. Máquina 1 - df - h.

```
estudiante@ing-gen71:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           1.2G  2.0M  1.2G   1% /run
/dev/sda2        79G   16G   59G  22% /
tmpfs           5.9G     0  5.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           5.9G  220K  5.9G   1% /run/qemu
tmpfs           1.2G  120K  1.2G   1% /run/user/120
tmpfs           1.2G   84K  1.2G   1% /run/user/1000
```

Figura 8. Máquina 2 - df - h.

```
estudiante@ing-gen129:~/Documents/evaluacion$ df -h
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           1.2G  2.0M  1.2G   1% /run
/dev/sda2        79G   16G   59G  21% /
tmpfs           5.9G     0  5.9G   0% /dev/shm
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           5.9G  220K  5.9G   1% /run/qemu
tmpfs           1.2G  104K  1.2G   1% /run/user/120
tmpfs           1.2G   84K  1.2G   1% /run/user/1000
```

Figura 9. Máquina 3 - df - h.

```
estudiante@ing-gen64:~/Documents/taller_alto_rendimiento$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
```

Figura 10. Máquina 1 – cat /etc/os-release.

```
estudiante@ing-gen71:~/Documents/Cepeda$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04.1 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04.1 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
```

Figura 11. Máquina 2 – cat /etc/os-release.

```
estudiante@ing-gen129:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 24.04 LTS"
NAME="Ubuntu"
VERSION_ID="24.04"
VERSION="24.04 LTS (Noble Numbat)"
VERSION_CODENAME=noble
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=noble
LOGO=ubuntu-logo
```

Figura 12. Máquina 3 – cat /etc/os-release.

Copiado de archivos y ejecución

Pasos para seguir:

1. Descomprimir y abrir carpeta del taller.
2. Separar el fichero fuente en biblioteca de funciones, interfaz y principal.
3. Copiar ficheros fuente del taller a las máquinas virtuales.
4. Compilación de ficheros para multiplicación de matrices de manera clásica y de manera transpuesta.
5. Modificación y ejecución del script en Pearl para la extracción de datos.

La separación del fichero fuente se realizó en 6 ficheros diferentes, los cuáles son:

- mm_clasico.c: Programa en C que implementa el algoritmo clásico de multiplicación de matrices cuadradas usando hilos. Utiliza las funciones definidas en mm_clasicoBiblioteca.c para realizar la multiplicación.
- mm_transpuesta.c: Programa en C que implementa el algoritmo de multiplicación de matrices cuadradas transpuestas usando hilos. Utiliza las funciones definidas en mm_transpuestaBiblioteca.c.
- mm_clasicoBiblioteca.c y mm_clasicoBiblioteca.h: Biblioteca de funciones y variables globales para el programa mm_clasico.c.
- mm_transpuestaBiblioteca.c y mm_transpuestaBiblioteca.h: Biblioteca de funciones y variables globales para el programa mm_transpuesta.c.

Nota: Todos los ficheros y su documentación se encuentran en el enlace al repositorio de GitHub en las referencias.

En las siguientes imágenes se presenta la copia de los ficheros empleando scp (Secure Copy Protocol), la recepción y ejecución para la máquina virtual 2.

Nota: El mismo procedimiento se siguió para las demás usando pwd (Print name of current/Working Directory) para obtener la ruta.

```
PS C:\Users\elice\OneDrive\Documentos\Javeriana\Sexto semestre\Sistemas Operativos\Taller Rendimiento>
 scp *.c *.pl *.h estudiante@10.43.100.160:/home/estudiante/Documents/Cepeda/evaluacion_
estudiante@10.43.100.160's password:
mm_clasico.c                                  100% 3664    68.8KB/s   00:00
mm_clasicoBiblioteca.c                        100% 4122   251.6KB/s   00:00
mm_transpuesta.c                              100% 4222   257.7KB/s   00:00
mm_transpuestaBiblioteca.c                    100% 4350    90.4KB/s   00:00
lanza.pl                                      100% 2576   157.2KB/s   00:00
lanzaTranspuesta.pl                           100% 2582   157.6KB/s   00:00
mm_clasicoBiblioteca.h                        100% 2620    80.0KB/s   00:00
mm_transpuestaBiblioteca.h                    100% 2841    75.0KB/s   00:00
PS C:\Users\elice\OneDrive\Documentos\Javeriana\Sexto semestre\Sistemas Operativos\Taller Rendimiento>
```

Figura 13. scp desde el host para la máquina virtual 2.

```
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ ls
clasico   lanzaTranspuesta.pl     mm_clasicoBiblioteca.h  mm_transpuestaBiblioteca.c  mm_transpuesta.c
lanza.pl  mm_clasicoBiblioteca.c  mm_clasico.c            mm_transpuestaBiblioteca.h
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ rm clasico
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ ls
lanza.pl            mm_clasicoBiblioteca.c  mm_clasico.c              mm_transpuestaBiblioteca.h
lanzaTranspuesta.pl  mm_clasicoBiblioteca.h  mm_transpuestaBiblioteca.c  mm_transpuesta.c
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ gcc mm_clasico.c mm_clasicoBiblioteca.c -o clasico
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ gcc mm_transpuesta.c mm_transpuestaBiblioteca.c -o transpuesta
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$ ls
clasico   lanzaTranspuesta.pl     mm_clasicoBiblioteca.h  mm_transpuestaBiblioteca.c  mm_transpuesta.c
lanza.pl  mm_clasicoBiblioteca.c  mm_clasico.c            mm_transpuestaBiblioteca.h  transpuesta
estudiante@ing-gen71:~/Documents/Cepeda/evaluacion_$
```

Figura 14. Recepción y ejecución de los ficheros en la máquina virtual 2.

Ejecución:

El script en Perl se modificó de tal forma que realizara pruebas de rendimiento para un programa de multiplicación de matrices, ejecutándolo con diferentes tamaños de matrices y distintas configuraciones de hilos. El objetivo es analizar el tiempo de ejecución para cada combinación y almacenar estos resultados en un archivo de salida.

Para comenzar, el script define varias configuraciones iniciales, incluyendo la ruta de trabajo actual ($path) y el nombre del programa a evaluar, en este caso, "clasico". También se especifican los tamaños de las matrices (@Num_Matriz) y el número de hilos (@Num_Hilos) que se usarán en cada prueba. La variable $Repeticiones indica el número de veces que se repetirá cada combinación de prueba para obtener un promedio o una muestra más representativa. Finalmente, se especifica el archivo de salida ($output_file), donde se guardarán todos los resultados.

Una vez configuradas las variables iniciales, el script abre el archivo de salida en modo de escritura. Si ocurre un error al intentar abrir el archivo, el script muestra un mensaje y termina la ejecución. A continuación, escribe un encabezado en el archivo de salida para estructurar los datos: la primera columna se reserva para los tamaños de las matrices, mientras que las siguientes columnas representan las distintas combinaciones de hilos utilizados en cada prueba.

El proceso de ejecución de pruebas comienza con un bucle que itera sobre cada tamaño de matriz en @Size_Matriz. Para cada tamaño, el script agrega una nueva fila en el archivo de salida, comenzando con el tamaño de la matriz en la primera columna. Luego, itera sobre cada configuración de hilos en @Num_Hilos, ejecutando el programa con la combinación actual de tamaño de matriz y número de hilos.

Cada vez que el programa se ejecuta, el script intenta capturar el tiempo de ejecución en milisegundos mediante una expresión regular, asegurándose de que se obtenga un único valor numérico. Si se captura correctamente el tiempo de ejecución, este se escribe en la fila correspondiente del archivo de salida. En caso contrario, el script registra un mensaje de error que indica qué combinación de programa, tamaño de matriz y número de hilos falló.

Finalmente, tras procesar todas las combinaciones de tamaños de matrices y configuraciones de hilos, el archivo de salida se cierra y el script imprime un mensaje indicando que los resultados se han guardado exitosamente. A continuación, se detalla el contenido del script.

```perl
#!/usr/bin/perl
#*****************************************************************************************
# Fecha: 31/10/2024
# Autor: Daniel Castro-Eliana Cepeda-Maria Paula Rodriguez- Daniel Gonzalez
# Materia: Sistemas Operativos
# Tema: Taller 6 – Evaluación de rendimiento
# Descripción:
#     Este programa realiza la multiplicación de matrices cuadradas N×N empleando el modelo
#     de hilos POSIX (Pthreads) y evalúa el tiempo de ejecución del algoritmo clásico de
#     multiplicación de matrices distribuyendo el trabajo en múltiples hilos
#*****************************************************************************************/

use strict;
use warnings;

# Ruta del directorio de trabajo actual
my $Path = `pwd`;
chomp($Path);

# Nombres de los ejecutables que se van a ejecutar
my @Nombres_Ejecutables = ("clasico");

# Tamaños de las matrices a evaluar
my @Size_Matriz = (600, 800, 1000, 1800);

# Número de hilos a utilizar en la ejecución
my @Num_Hilos = (1,2,4);

# Número de repeticiones para cada combinación de tamaño de matriz y número de hilos
my $Repeticiones = 30;

# Nombre del archivo de salida
my $output_file = "$Path/resultados_Clasico.dat";

# Abre el archivo de salida para escribir los resultados
open(my $out_fh, '>', $output_file) or die "No se pudo abrir el archivo $output_file: $!";

# Escribe el encabezado de la tabla en el archivo de salida
print $out_fh "\t\t";  # Espacio para la primera columna
foreach my $hilo (@Num_Hilos) {
    print $out_fh "Hilos: $hilo\t\t";
}
print $out_fh "\n";

# Itera sobre cada tamaño de matriz
foreach my $size (@Size_Matriz) {
    for (my $i = 0; $i < $Repeticiones; $i++) {
        print $out_fh "$size\t\t";  # Escribe el tamaño de la matriz en la primera columna

        # Itera sobre el número de hilos
        foreach my $hilo (@Num_Hilos) {
            my $output = `$Path/$Nombres_Ejecutables[0] $size $hilo`;

            # Verifica si el valor de tiempo está en la salida (asumimos un solo valor de tiempo en ms)
            if ($output =~ /\b(\d+(\.\d+)?)\b/) {
                printf $out_fh "%.d   \t\t", $1;
            } else {
                die "No se encontró un valor de tiempo en la salida de $Nombres_Ejecutables[0] con tamaño $size y $hilo hilos.";
            }
        }
        print $out_fh "\n";  # Nueva línea para la siguiente ejecución
    }
        print $out_fh "\n";  # inserta un espacio entre los datos de un tamaño y otro
}

# Cierra el archivo de salida
close($out_fh);
print "Resultados guardados en $output_file\n";
```

Figura 15. Script en Perl para clásico.

El proceso para multiplicar matrices por el método de transposición es idéntico y solamente se cambió el nombre del ejecutable, el archivo de escritura y la información de las salidas para que concuerde con la tarea asignada. A continuación, se detalla el contenido del script.

```perl
#!/usr/bin/perl
#*********************************************************************************************
# Fecha: 31/10/2024
# Autor: Daniel Castro-ELiana Cepeda-Maria Paula Rodriguez- Daniel Gonzalez
# Materia: Sistemas Operativos
# Tema: Taller 6 – Evaluación de rendimiento
# Descripción:
#     Este programa realiza la multiplicación de matrices cuadradas N×N empleando el modelo
#     de hilos POSIX (Pthreads) y evalúa el tiempo de ejecución del algoritmo clásico de
#     multiplicación de matrices distribuyendo el trabajo en múltiples hilos
#*********************************************************************************************/

use strict;
use warnings;

# Ruta del directorio de trabajo actual
my $Path = `pwd`;
chomp($Path);

# Nombres de los ejecutables que se van a ejecutar
my @Nombres_Ejecutables = ("transpuesta");

# Tamaños de las matrices a evaluar
my @Size_Matriz = (600, 800, 1000, 1800);

# Número de hilos a utilizar en la ejecución
my @Num_Hilos = (1,2,4);

# Número de repeticiones para cada combinación de tamaño de matriz y número de hilos
my $Repeticiones = 30;

# Nombre del archivo de salida
my $output_file = "$Path/resultados_Transpuesta.dat";

# Abre el archivo de salida para escribir los resultados
open(my $out_fh, '>', $output_file) or die "No se pudo abrir el archivo $output_file: $!";

# Escribe el encabezado de la tabla en el archivo de salida
print $out_fh "\t\t";  # Espacio para la primera columna
foreach my $hilo (@Num_Hilos) {
    print $out_fh "Hilos: $hilo\t\t";
}
print $out_fh "\n";

# Itera sobre cada tamaño de matriz
foreach my $size (@Size_Matriz) {
    for (my $i = 0; $i < $Repeticiones; $i++) {
        print $out_fh "$size\t\t";  # Escribe el tamaño de la matriz en la primera columna

        # Itera sobre el número de hilos
        foreach my $hilo (@Num_Hilos) {
            my $output = `$Path/$Nombres_Ejecutables[0] $size $hilo`;

            # Verifica si el valor de tiempo está en la salida (asumimos un solo valor de tiempo en ms)
            if ($output =~ /\b(\d+(\.\d+)?)\b/) {
                printf $out_fh "%.d   \t\t", $1;
            } else {
                die "No se encontró un valor de tiempo en la salida de $Nombres_Ejecutables[0] con tamaño $size y $hilo hilos.";
            }
        }
        print $out_fh "\n";  # Nueva línea para la siguiente ejecución
    }
        print $out_fh "\n";  # inserta un espacio entre los datos de un tamaño y otro
}

# Cierra el archivo de salida
close($out_fh);
print "Resultados guardados en $output_file\n";
```

Figura 16. Script en Perl para transpuesta.

Batería de Experimentación

Primeramente, la ley de los grandes números (LGN) es un principio fundamental en la teoría de la probabilidad que establece que, a medida que se realizan un gran número de ensayos o experimentos, la media de los resultados de estos ensayos tenderá a acercarse a la media esperada o al valor verdadero de la población.

Así, al aplicar la LGN, se pueden realizar múltiples ejecuciones del mismo experimento (al menos 30 repeticiones, en este caso). Esto permite calcular el promedio y la desviación estándar de los tiempos de ejecución que son datos más representativos del comportamiento real del algoritmo. Cuantos más ensayos se realicen, mayor será la certeza de que el promedio calculado se asemeje a la media esperada y menor será la desviación estándar al reducir el ruido gaussiano.

Los siguientes son los parámetros tenidos en cuenta para la ejecución de la batería:

1. **Selección de Sistemas de Cómputo**:
    - Se seleccionan las 3 máquinas virtuales asignadas por la universidad, dado que, en el taller anterior, se demostró su superioridad frente a las asignadas por el profesor.
2. **Selección de Tamaños de Matrices (@Num_Matriz):**
    - Los valores seleccionados fueron 600, 800, 1000 y 1800, ya que estos tamaños ofrecen una variedad suficiente que permite observar el comportamiento del algoritmo tanto en matrices pequeñas (donde se espera que el tiempo de ejecución sea menor) como en matrices grandes, donde la complejidad y el tiempo de ejecución aumentan significativamente. Al elegir matrices de diferentes tamaños, se pueden observar patrones de rendimiento y escalabilidad. Cabe resaltar que se intentó con 2000, pero el sistema no fue capaz de procesarlo.
3. **Selección de Valores de Hilos de Ejecución (@Num_Hilos)**:
    - La elección de hilos de 1, 2, y 4 de ejecución permite evaluar el rendimiento tanto en un enfoque secuencial (1 hilo) como en un enfoque paralelo (usando 2 y 4). Esto permite determinar la eficiencia del algoritmo en la utilización de múltiples núcleos del procesador, los cuáles se determinó previamente que son 4 para todas las máquinas, y su capacidad de escalabilidad.
4. **Ejecución de Cada Valor Seleccionado (@Repeticiones)**:
    - Se ejecutaron 30 repeticiones para cada combinación de tamaño de matriz y número de hilos, asegurando una muestra suficiente para calcular un promedio confiable del tiempo de ejecución, debido a las variaciones en el tiempo de

ejecución que pueden ser causadas por la carga del sistema, el uso de recursos y otros factores externos (ruido gaussiano).

5. **Tabla de Resultados**:
    - Los resultados de la ejecución en cada máquina se guardaron en los ficheros "resultados_X.dat", donde X es el método de multiplicación. Con estos datos, se empleó una hoja de cálculo en Excel para elaborar las tablas, promedios y gráficas. El enlace se encuentra en las referencias.

Gráficas y análisis

Para cada grupo de datos (es decir las 30 ejecuciones) se calculó el promedio y la desviación estándar, y con el fin de poder analizar los datos, estos se agruparon en diferentes tablas que se pueden ver a detalle en la hoja de cálculo adjunta.

En cuanto a las pruebas realizadas, sus resultados se presentan en la hoja de cálculo adyacente y, para una mejor visualización, se anexan además las gráficas realizadas con su respectivo análisis.

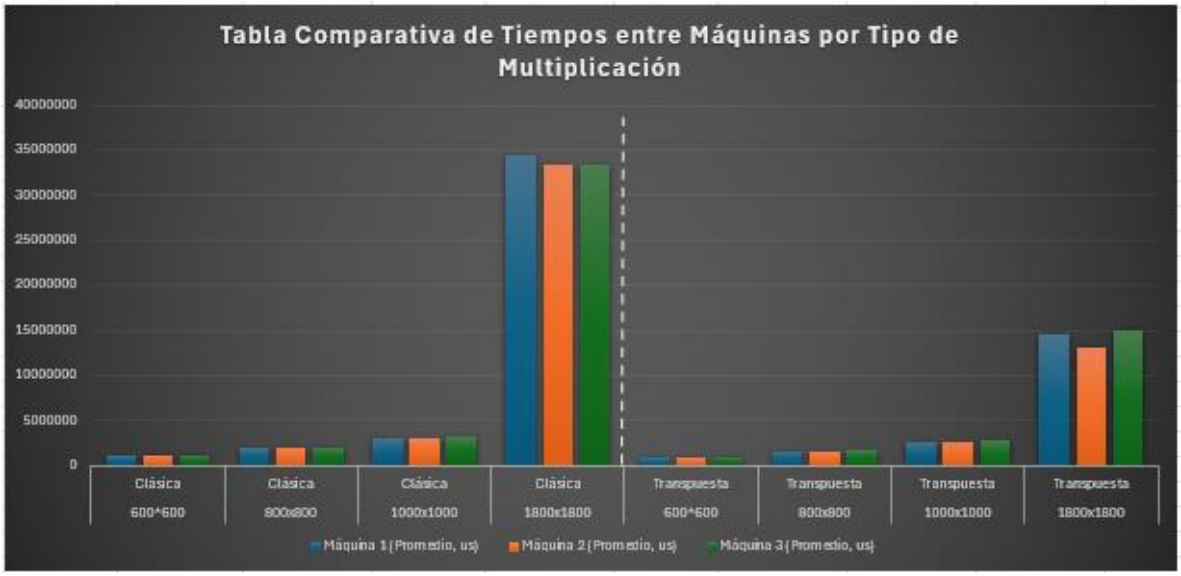

Figura 17. Gráfica comparativa de tiempos entre maquinas.

En la gráfica se puede ver la comparativa de rendimiento entre maquinas, y aunque todas tienen un comportamiento muy similar se ve una diferencia significativa en los tiempos de

la última prueba (Transpuesta con tamaño de 1800) donde se evidencia un mejor resultado con la Maquina 2. Esto puede deberse a lo que se mencionó anteriormente en el análisis de recursos ya que se evidencia que la Máquina 2 tiene una mejor distribución de la memoria. Además, parece oportuno concluir que efectivamente el método de multiplicación de matrices transpuesto reduce significativamente el tiempo necesario de ejecución, no solo en matrices pequeñas sino significativamente en matrices grandes por lo que se podría afirmar que ese algoritmo es mucho mejor.

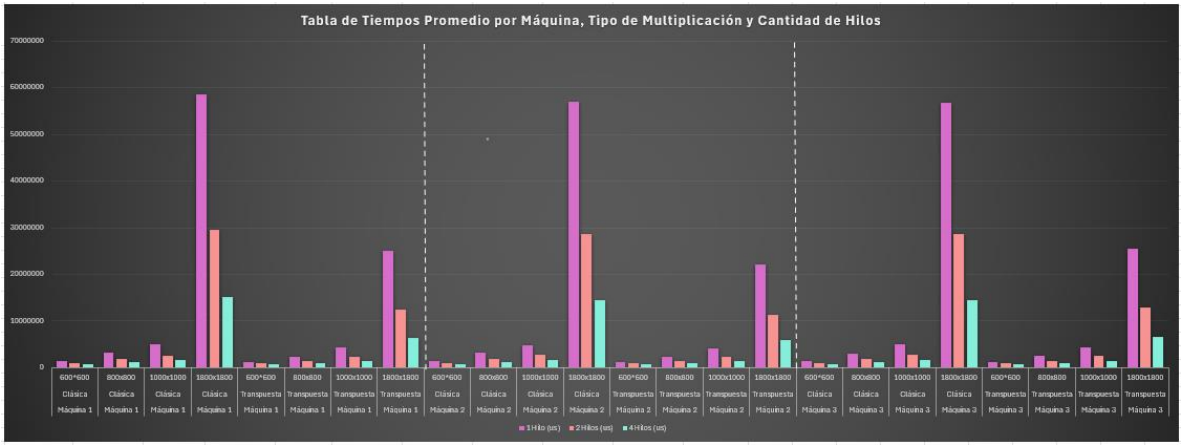

Figura 18. Gráfica comparativa de tiempos entre tipo de multiplicación y cantidad de hilos.

Con esta gráfica podemos evidenciar dos propiedades importantes. Primero, como afecta el tipo de multiplicación que se realiza, y segundo, como afecta la cantidad de hilos que se usan para realizar la tarea.

Se puede ver que el algoritmo que usa la matriz transpuesta reduce el tiempo de ejecución, esto se debe al principio de proximidad espacial, ya que al usar la matriz transpuesta las posiciones de los datos que se van multiplicando coinciden y al estar más cerca se requieren menos pasos para conseguir los datos. Como contexto, se tiene que el principio de proximidad espacial nos indica que las cosas que están más cerca son las que tienen más probabilidades de ser usadas. Que es en parte lo que busca el algoritmo al usar la matriz transpuesta para multiplicar las matrices.

También se puede evidenciar que en las gráficas a medida que se usan más hilos se reduce significativamente el tiempo de ejecución de la tarea, esto se debe a que los hilos permiten hacer procesos en paralelo y por tanto se reduce el tiempo. En teoría la reducción del tiempo debería ser proporcional a la cantidad de hilos que se use, sin embargo, hay que tener en consideración el overhead que se genera al tener que integrar los datos recibidos de los diferentes procesadores

Por último se presentan las siguientes tres gráficas para verificar la veracidad y confiabilidad de los datos obtenidos para cada una de las maquinas.

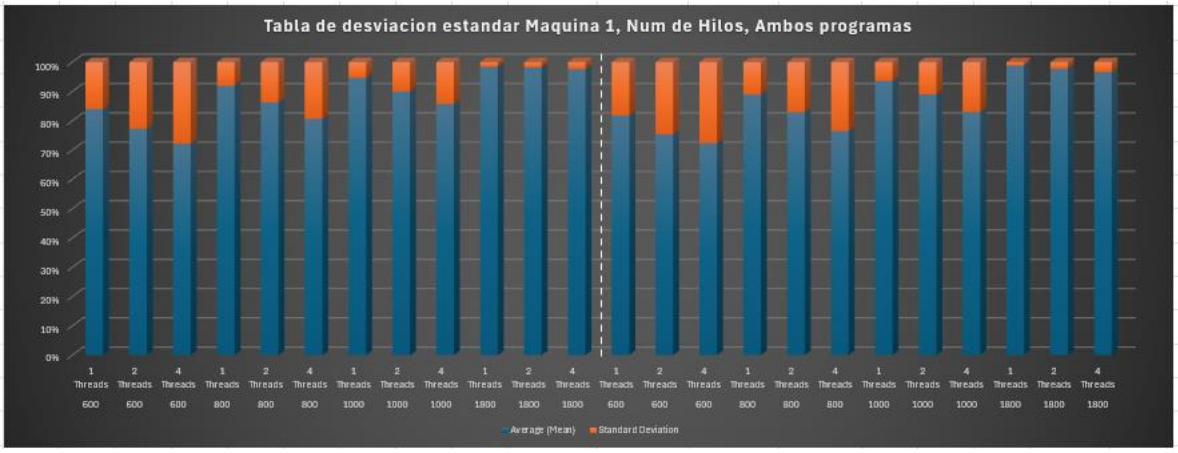

Figura 19. Desviación estándar Maquina 1.

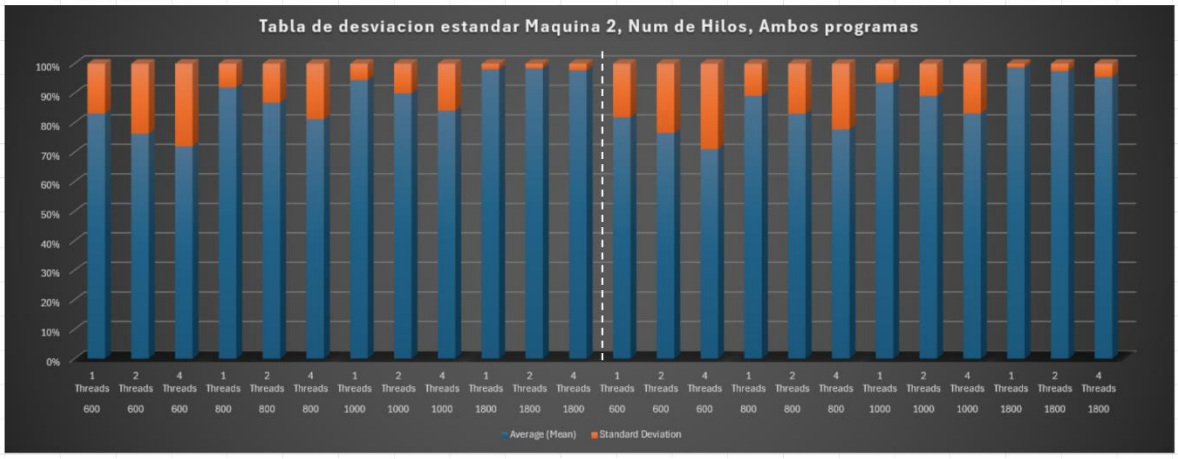

Figura 20. Desviación estándar Maquina 2.

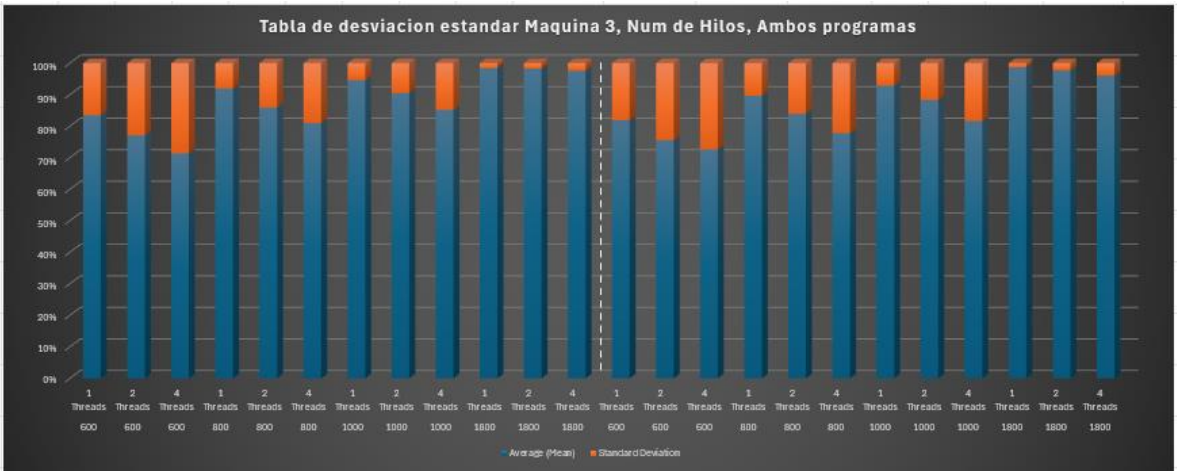

Figura 21. Desviación estándar Maquina 3.

En las tres maquinas se observa un comportamiento similar, el porcentaje de la desviación estándar va aumentando con el aumento de la cantidad de hilos y va disminuyendo con el aumento en el tamaño de la matriz. El primer fenómeno sucede debido a que entre más hilos hay más se tienen que desplazar los datos y por tanto más probabilidad de ruido, y es segundo fenómeno se da porque al haber más datos el porcentaje de ruido es mucho menor comparado con el tiempo real estimado de ejecución.

Es decir, entre más grande sea la matriz que hay que procesar, encontraremos menos porcentaje de ruido, y por el contrario entre más se tengan que desplazar los datos (relacionado con la cantidad de hilos) habrá un mayor porcentaje de ruido.

Sin embargo, ninguno de los porcentajes da mayor al 30% y por tanto podemos considerar que en los términos prácticos de este taller los datos son congruentes y podemos sacar conclusiones verídicas del análisis de estos.

Conclusiones y recomendaciones

La Máquina 2 demostró ser la más adecuada para realizar tareas de multiplicación de matrices, especialmente en configuraciones que requieren mayor rendimiento. Esto se debe a su distribución de memoria, la cual favorece un acceso más rápido a los datos, como se evidencia en los tiempos de ejecución de las pruebas, especialmente con el tamaño de matriz más grande y con el algoritmo más eficiente (Transpuesta 1800). La Máquina 2 aprovechó mejor el uso de múltiples hilos y mostró menores tiempos de ejecución, lo que sugiere una mayor eficiencia en la gestión de procesos paralelos.

Por otro lado, aunque la Máquina 1 mostró un rendimiento aceptable, similar al de la Máquina 2 en configuraciones de matrices pequeñas, en aplicaciones de mayor demanda y con un mayor número de hilos, la Máquina 2 sobresale. En contraste, la Máquina 3, a pesar de tener un procesador más reciente, tiene una menor capacidad de caché L3, lo que puede

haber limitado su rendimiento en operaciones intensivas y en configuraciones de matrices grandes.

En conclusión, para las condiciones y el tipo de procesamiento requerido en este taller, la Máquina 2 es la más recomendable debido a su balance entre capacidad de procesamiento y distribución de memoria caché, optimizando así el rendimiento en tareas de alto consumo de recursos, siendo congruente con el análisis preliminar hecho teniendo en cuenta el hardware de cada máquina.

Referencias

Repositorio en GitHub de cada integrante:

https://github.com/Dani2044/Sistemas-Operativos/tree/main/Taller%206%20-%20Rendimiento

https://github.com/MapaRuiz/OperatingSystems/tree/main/Cap03/Taller_de_Evaluacion

https://github.com/Gonzalez-Daniel-H/SISTEMAS-OPERATIVOS/tree/main/TERCER%20CORTE/TALLER_DE_EVALUACION

https://github.com/ElianaCepeda/Sistemas-Operativos/tree/main/Cap03/Taller_Rendimiento_Grupo

Hoja de cálculo en Excel:

https://livejaverianaedu-my.sharepoint.com/:x:/g/personal/elianacepeda_javeriana_edu_co/EYeqaDZsVyVAsRgi_xJItrgBUo9rqFswPi3uVsDep96GMw?e=9Xcz5j