MULTIPLICACION DE MATRICES CON PROCESAMIENTO PARALELO

Gonzalo Castillo López
Pontificia Universidad Javeriana, Bogotá D.C.
g.castillol@javeriana.edu.co

Resumen. Este articulo presenta la comparación del rendimiento en la resolución del producto de matrices, a través de algoritmos del producto (filas)*(columnas) y (filas)*(filas) con el uso de OpenMP, una interfaz de programación de aplicaciones diseñada para sistemas de memoria compartida. El objetivo es el de optimizar el desempeño en el procesamiento en esta operación algebraica.

Palabras claves. Procesamiento paralelo, OpenMP, algoritmos

I. INTRODUCCIÓN

La tarea de la multiplicación de matrices puede ser intensiva en términos computacionales, sin embargo, su rendimiento puede mejorarse mediante la introducción de paralelismo. En este caso, como herramienta de procesamiento se utilizará OpenMP, el cual proporciona el paralelismo permitiendo que varias secciones del programa se ejecuten simultáneamente en múltiples hilos.

El rendimiento en el procesamiento del producto de matrices se validará utilizando dos tipos de algoritmos desarrollados en C++, el primer caso realiza la operación multiplicando las filas de la primera matriz por las columnas de la segunda matriz y en el segundo el producto se realizará tomando las filas de ambas matrices.

Para la experimentación del caso de estudio, las ejecuciones de los procesos se realizaron con HTCONDOR herramienta para el soporte de la computación distribuida. HTCONDOR al ser un sistema de administración y monitoreo para trabajos de cómputo intensivos, ofrece las opciones como la de ejecutar uno o varios Jobs en varios Cores (una tarea en un Core es un Job), y permite definir cuando y donde ejecutar un Job, monitoriza el Job informando su finalización

II. METODOLOGIA

En el proceso de observación del procesamiento del producto de matrices, se definen dos pasos a seguir para obtener el resultado final:

1. Variación del algoritmo de cálculo, con un proceso de cálculo Columnas*Filas y Filas*Filas.
2. Utilizar herramientas de procesamiento en paralelo, para optimizar el rendimiento.
3. Realizar experimentos con variación de tamaño de matrices, cantidad de procesadores, hilos de procesos.
4. Observación de los resultados obtenidos en cada set de pruebas, e identificar cual seria la mejor opción para realizar este proceso.

III. EXPERIMENTACION

Para ejecución de trabajos paralelos, se utilizan los universos ofrecidos por la herramienta HTCONDOR, los cuales en el momento de la ejecución reclama una cantidad fija de máquinas para el trabajo del universo paralelo.

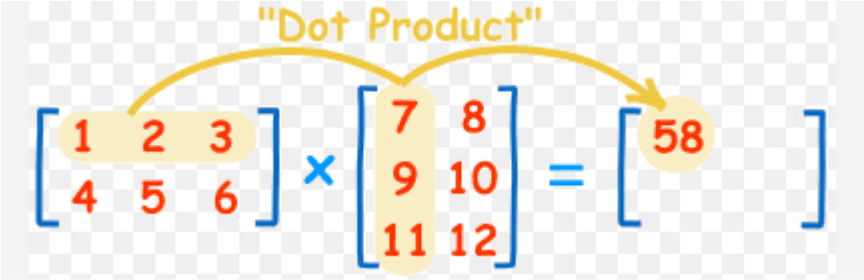
La versión utilizada es:

```
$CondorVersion: 8.6.11 Aug 10 2018 BuildID: RH-8.6.11-1.el6 $
$CondorPlatform: X86_64-RedHat_6.10 $
```

Antes de iniciar el proceso de experimentación de los casos se validan que los servicios de HTCONDOR estén arriba, para garantizar la correcta ejecución.

```
condor     476363  0.0  0.1  93448  3780 ?        Ss   Aug13   1:10 /usr/sbin/condor_master -pidfile /var/run/condor/cond
or.pid
root       476401  0.0  0.0  22724  2264 ?        S    Aug13  19:31 condor_procd -A /var/run/condor/procd_pipe -L /var/lo
g/condor/ProcLog -R 1000000 -S 60 -C 390
condor     476402  0.0  0.1  93124  4144 ?        Ss   Aug13   1:07 condor_shared_port -f
condor     476403  0.0  2.1 151332 63224 ?        Ss   Aug13 116:31 condor_schedd -f
condor     476404  0.0  0.1  93700  4656 ?        Ss   Aug13   0:59 condor_startd -f
```

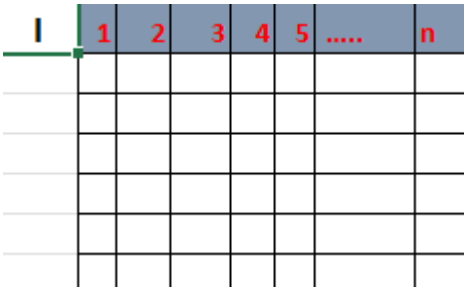
Código Desarrollado. La lógica del código utilizado en los programas dearrollados en C++, presentan dos variaciones para realizar la multiplicación de las matrices:



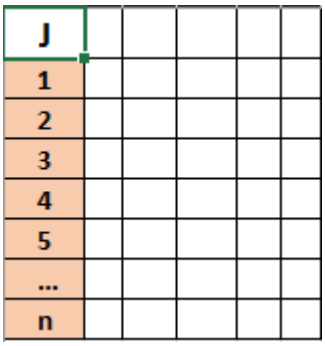
1. Calculo Columna * Fila

Para describir el proceso de una manera gráfica, diremos que la primera matriz tendrá una cantidad de columnas identificadas como **I** y una cantidad de filas a las que se identifican como **J**.

Columnas.



Filas.



En este primer caso la lógica del proceso toma 1 (primera columna) y multiplica por lo valores de todas las filas de esa columna, incrementa en uno la columna y multiplica por los valores de las filas de esa columna, y así hasta que finalice las columnas.

2. Calculo Fila * Fila

Para describir el proceso de una manera gráfica, diremos que la primera matriz tendrá una cantidad de columnas identificadas como **I** y una cantidad de filas a las que se identifican como **J**.

El proceso inicia con la fila 1, primera posición, de la primera matriz y la multiplica por todos los valores de la fila 1, luego incrementa a la segunda fila de la primera matriz, y multiplica por todos los valores de la fila 2 de la segunda matriz.

Ejecución del proceso en paralelo.

Para la ejecución de los procesos de utilizo el universo paralelo de HTCONDOR "**Vanilla**", en el cual se parametrizaron los dos programas desarrollados en C++, para la ejecución.

1. El primer programa de multiplicación de matrices Columna por filas, con el nombre MM1c.c, se ejecutó variando los argumentos.

```
Universe = vanilla
Executable = MM1c
Log = MM1-1.log
Output = MM1-1.out
Error = MM1-1.error
Notification = error
should_transfer_files = YES
Arguments = 10 1 0
Queue
Arguments = 100 1 0
Queue
Arguments = 1000 1 0
Queue
Arguments = 1000 1 0
Queue
[g.castillol@condor BIN]$ condor_sub MM1c-1.sub
-bash: condor_sub: command not found
[g.castillol@condor BIN]$ condor_q
```

El trabajo se somete para ejecución .

```
[g.castillol@condor BIN]$ condor_submit MM1c-1.sub
Submitting job(s)....
4 job(s) submitted to cluster 9681.
```

```
-- Schedd: condor.javeriana.edu.co : <10.34.1.2:9618?... @ 11/13/23 18:15:58
OWNER BATCH_NAME      SUBMITTED   DONE   RUN    IDLE   HOLD  TOTAL JOB_IDS

0 jobs; 0 completed, 0 removed, 0 idle, 0 running, 0 held, 0 suspended
[g.castillol@condor BIN]$ condor_submit MM1c-1.sub
Submitting job(s)....
4 job(s) submitted to cluster 9681.
```

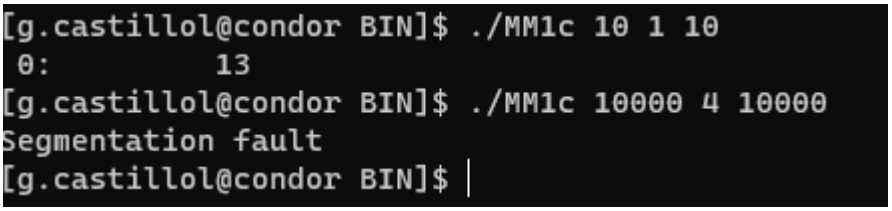
Validando con Condor se evidencia los procesos generan trabajos sometidos.

```
[g.castillol@condor BIN]$ cat MM1-3.log
000 (9684.000.000) 11/13 19:14:37 Job submitted from host: <10.34.1.2:9618?addrs=10.34.1.2-9618+[--1]-9618&noUDP&sock=476363_49d3_3>
...
000 (9684.001.000) 11/13 19:14:37 Job submitted from host: <10.34.1.2:9618?addrs=10.34.1.2-9618+[--1]-9618&noUDP&sock=476363_49d3_3>
...
000 (9684.002.000) 11/13 19:14:37 Job submitted from host: <10.34.1.2:9618?addrs=10.34.1.2-9618+[--1]-9618&noUDP&sock=476363_49d3_3>
...
000 (9684.003.000) 11/13 19:14:37 Job submitted from host: <10.34.1.2:9618?addrs=10.34.1.2-9618+[--1]-9618&noUDP&sock=476363_49d3_3>
```

En el log del sistema se evidencia que la ejecución de los trabajos sometidos

```
001 (9684.000.000) 11/13 19:14:43 Job executing on host: <10.34.1.15:9618?addrs=10.34.1.15-9618+[--1]-9618&noUDP&sock=1367668_1198_3>
...
001 (9684.002.000) 11/13 19:14:43 Job executing on host: <10.34.1.15:9618?addrs=10.34.1.15-9618+[--1]-9618&noUDP&sock=1367668_1198_3>
...
001 (9684.001.000) 11/13 19:14:43 Job executing on host: <10.34.1.15:9618?addrs=10.34.1.15-9618+[--1]-9618&noUDP&sock=1367668_1198_3>
```

Se evidencia que para los valores elevados en los argumentos, el proceso no se ejecuta.



Como resultado de esta ejecución del programa MM1c, se toman los resultados de tiempo, para validación con el segundo programa.

2. El segundo programa de multiplicación de matrices Columna por filas, con el nombre **MMFF1c.c**, se ejecutó variando los argumentos.

```
[g.castillol@condor BIN]$ cat MMFF1c-A.sub
Universe = vanilla
Executable = MMFF1c
Log = MMFF1c.$(Process).log
Output = MMFF1c.$(Process).out
Error = MMFF1c.$(Process).error
Notification = error
should_transfer_files = YES
when_to_transfer_output = ON_EXIT

Arguments = 10 1 10
Queue
Arguments = 100 2 100
Queue
Arguments = 1000 3 1000
Queue
Arguments = 10000 4 10000
Queue
[g.castillol@condor BIN]$ condor_submit MMFF1c-A.sub
Submitting job(s)....
4 job(s) submitted to cluster 9685.
[g.castillol@condor BIN]$
```

Las pruebas y comportamiento en la parametrización con universos paralelos de HTCONDOR, es el mismo que se describió para las pruebas del programa MM1c.c (Columnas por filas).

IV. RESULTADOS.

Los resultados de la experimentación permitieron evaluar el tiempo de ejecución al utilizar OpenMP para la multiplicación de matrices. La variación observada fue proporcional al número utilizado en los argumentos modificados al momento de la ejecución.

La capacidad de procesamiento paralelo se hizo evidente, sin embargo al no tener otra herramienta de procesamiento no permitió realizar validaciones con los parámetros inicialmente planteados.

TRABAJO	Job submitted	Job executing	Job terminated.
	COLUMNAS * FILAS		
MM1c.0.log	000 (9683.000.000) 11/13 18:36:43	001 (9683.000.000) 11/13 18:36:59	005 (9683.000.000) 11/13 18:37:00
MM1c.1.log	000 (9683.001.000) 11/13 18:36:43	001 (9683.001.000) 11/13 18:36:59	006 (9683.001.000) 11/13 18:37:00
MM1c.2.log	000 (9683.002.000) 11/13 18:36:43	001 (9683.002.000) 11/13 18:36:59	006 (9683.002.000) 11/13 18:37:02
	FILAS * FILAS		
MMFF1c.1.lo	000 (9685.000.000) 11/13 21:28:09	001 (9685.000.000) 11/13 21:28:21	005 (9685.000.000) 11/13 21:28:22
MMFF1c.2.lo	000 (9685.001.000) 11/13 21:28:09	001 (9685.001.000) 11/13 21:28:21	005 (9685.001.000) 11/13 21:28:21
MMFF1c.2.lo	000 (9685.002.000) 11/13 21:28:09	001 (9685.002.000) 11/13 21:28:21	005 (9685.002.000) 11/13 21:28:24

V. CONCLUSIONES.

El resultado de la experimentación evidencia la eficacia de OpenMP en la optimización de la multiplicación de matrices mediante la introducción de paralelismo, permitiendo a través de una implementación sencilla y eficiente, tener un aumento notable en el rendimiento de los programas.