



ESCUELA DE INGENIERÍA
FACULTAD DE INGENIERÍA

DIPLOMADO BIGDATA

UNIDAD 7: COMPUTACIÓN DE ALTO RENDIMIENTO PARA BIG DATA.

INFORME DE ACTIVIDAD Nº3

28/Junio/2018

Alumnos:

1. Humberto Andrés Alvarez Vilches
2. Francisco Rodrigo Marín Küllmer



Tabla de contenido

Unidad 7: Computación de alto rendimiento para Big Data.....1

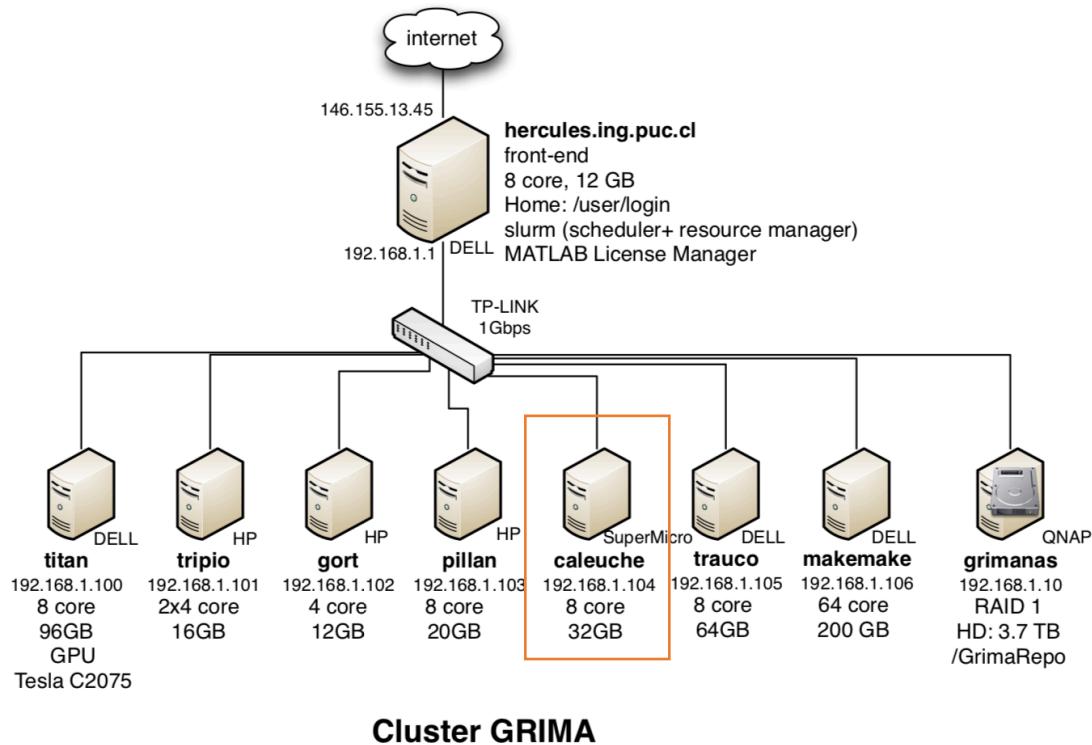
Informe de laboratorio Nº2	1
Ambientación de las pruebas.....	3
Selección de las imágenes	4
Resultados experimentales, Imagen GRANDE.....	5
Resultados experimentales, Imagen MEDIANA.....	7
Resultados experimentales, Imagen PEQUEÑA.....	9
Preguntas.....	11



Ambientación de las pruebas

La actividad se ejecuta en el Cluster Grima /Nodo Caleuche.

CPU. 8 CORE
RAM. 32 Gb





Selección de las imágenes

Ubicados en la carpeta /user/cruz/DipBD/2017-2/AC03/images, listamos las imágenes con el comando: ls -lsh

```
[grupo8d6@hercules:~/AC03$ ls -lsh /user/cruz/DipBD/2017-2/AC03/images/
total 231M
324K -rw-r--r-- 1 cruz grima 321K Jul 10 2017 Bird.png
3.6M -rw-r--r-- 1 cruz grima 3.6M Jul 10 2017 Rose.png
444K -rw-r--r-- 1 cruz grima 444K Jul 10 2017 Yoshi.png
480K -rwxr-xr-x 1 cruz grima 479K Jul 10 2017 ash.png
7.6M -rw-r--r-- 1 cruz grima 7.6M Jul 10 2017 big.png
3.5M -rw-r--r-- 1 cruz grima 3.5M Jul 10 2017 cat.png
9.7M -rw-r--r-- 1 cruz grima 9.7M Jul 10 2017 chrono.png
1.5M -rw-r--r-- 1 cruz grima 1.5M Jul 10 2017 distribuidos1s.png
8.1M -rw-r--r-- 1 cruz grima 8.1M Jul 10 2017 dog.png
136K -rw-r--r-- 1 cruz grima 136K Jul 10 2017 example.png
4.7M -rw-r--r-- 1 cruz grima 4.7M Jul 10 2017 lake.png
552K -rw-r--r-- 1 cruz grima 552K Jul 10 2017 lena.png
136K -rw-r--r-- 1 cruz grima 136K Jul 10 2017 original.png
120K -rwxr-xr-x 1 cruz grima 119K Jul 10 2017 primarina.png
188M -rw-r--r-- 1 cruz grima 188M Jul 10 2017 really_big.png
20K -rw-r--r-- 1 cruz grima 17K Jul 10 2017 test.jpg
1.2M -rw-r--r-- 1 cruz grima 1.2M Jul 10 2017 test1.png
44K -rwxr-xr-x 1 cruz grima 41K Jul 10 2017 test2.png
1.1M -rw-r--r-- 1 cruz grima 1.1M Jul 10 2017 titania.png
56K -rw-r--r-- 1 cruz grima 56K Jul 10 2017 world.png
grupo8d6@hercules:~/AC03$ ]
```

Usaremos las siguientes

Grande → lake.png – 4,7 MB

Mediana → Yoshi.png – 444 KB

Pequeña → world.png – 56 KB



Resultados experimentales, Imagen GRANDE.

Se ejecuta el programa ./masker para la imagen GRANDE.

Imagen original: lake.png – 4,7 MB	Imagen modificada: Img_lake.png – 3,7MB
lake.png – 4,7 MB	Img_lake.png – 3,7MB
<p><i>Justificación de la elección.</i> Elegimos la imagen lake.png, considerando que es la de mayor peso en Bytes. (4,7 Mb). Se eligió considerando el formato PNG .</p>	

Con esto se obtiene los siguientes tiempos de ejecución:

	Iteraciones		
Threads	10	50	100
1	9,645843	47,827253	98,142908
2	5,138992	24,608324	52,230964
4	2,634882	13,282095	30,231417
8	2,243051	9,5614050	27,126446
16	2,013196	9,0094940	18,849647

Rendimiento:

Threads	Iteraciones					
	10		50		100	
Threads	Speedup	Eficiencia	Speedup	Eficiencia	Speedup	Eficiencia
1	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000
2	1,87699125	0,93849562	1,94353963	0,97176982	1,87901774	0,93950887
4	3,66082542	0,91520635	3,60088171	0,90022043	3,24638795	0,81159699
8	4,30032264	0,53754033	5,00211559	0,62526445	3,61797885	0,45224736
16	4,79130845	0,29945678	5,30853930	0,33178371	5,20661782	0,32541361



Resultado gráfico	Conclusión del experimento																								
<p style="text-align: center;">Speedup Imagen Grande</p> <table border="1"><caption>Data for Speedup Imagen Grande</caption><thead><tr><th>Cores</th><th>Speedup 10</th><th>Speedup 50</th><th>Speedup 100</th></tr></thead><tbody><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>2</td><td>2.00</td><td>2.00</td><td>2.00</td></tr><tr><td>4</td><td>3.50</td><td>3.50</td><td>3.50</td></tr><tr><td>8</td><td>4.50</td><td>4.50</td><td>4.50</td></tr><tr><td>16</td><td>5.00</td><td>5.00</td><td>5.00</td></tr></tbody></table>	Cores	Speedup 10	Speedup 50	Speedup 100	1	1.00	1.00	1.00	2	2.00	2.00	2.00	4	3.50	3.50	3.50	8	4.50	4.50	4.50	16	5.00	5.00	5.00	<p>Observamos que durante la ejecución realizada por 4 a 8 core, el speedup se reduce sustancialmente. Posterior a ellos, la curva es decreciente. La aplicación es que el cluster responde con un buen Speedup mientras hayan CORE disponibles para parallelizar.</p>
Cores	Speedup 10	Speedup 50	Speedup 100																						
1	1.00	1.00	1.00																						
2	2.00	2.00	2.00																						
4	3.50	3.50	3.50																						
8	4.50	4.50	4.50																						
16	5.00	5.00	5.00																						
<p style="text-align: center;">Eficiencia Imagen Grande</p> <table border="1"><caption>Data for Eficiencia Imagen Grande</caption><thead><tr><th>Cores</th><th>Eficiencia 10</th><th>Eficiencia 50</th><th>Eficiencia 100</th></tr></thead><tbody><tr><td>1</td><td>1.00</td><td>1.00</td><td>1.00</td></tr><tr><td>2</td><td>0.95</td><td>0.95</td><td>0.95</td></tr><tr><td>4</td><td>0.85</td><td>0.85</td><td>0.85</td></tr><tr><td>8</td><td>0.60</td><td>0.60</td><td>0.60</td></tr><tr><td>16</td><td>0.30</td><td>0.30</td><td>0.30</td></tr></tbody></table>	Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100	1	1.00	1.00	1.00	2	0.95	0.95	0.95	4	0.85	0.85	0.85	8	0.60	0.60	0.60	16	0.30	0.30	0.30	<p>La eficiencia posee buen indicador usando hasta 4 CORE. La eficiencia entre 1 y 0.8, poseen un valor aceptable, mientras que la eficiencia cae pasando el umbral de 4 core. La diferencia entre la iteración=50 (amarillo) , iteración=10 (naranja) versus el iteración=100 (verde), la cual decae, creemos que es debido a un mayor porcentaje de código secuencial (no paralelizable) que impacta cuando se trabaja con mayor cantidad de Core.</p>
Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100																						
1	1.00	1.00	1.00																						
2	0.95	0.95	0.95																						
4	0.85	0.85	0.85																						
8	0.60	0.60	0.60																						
16	0.30	0.30	0.30																						



Resultados experimentales, Imagen MEDIANA.

Se ejecuta el programa ./masker para la imagen MEDIANA.

Imagen original: Yoshi.png – 444 KB	Imagen modificada: Img_Yoshi.png – 532 KB
	
<p><i>Justificación de la elección.</i> Elegimos la imagen lake.png, considerando que es la de mayor peso en Bytes. (444 Kb). Se consideró la elección de imagen en formato PNG.</p>	

Con esto se obtiene los siguientes tiempos de ejecución:

Threads	Iteraciones		
	10	50	100
1	3,342240	16,716926	32,623741
2	1,685830	9,701681	17,187236
4	0,908381	4,977571	9,3971940
8	0,822689	2,742120	8,1800870
16	0,683779	3,099466	7,3915100

Rendimiento:

Threads	Iteraciones					
	10		50		100	
Speedup	Eficiencia	Speedup	Eficiencia	Speedup	Eficiencia	
1	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000
2	1,98254866	0,99127433	1,72309582	0,86154791	1,89813772	0,94906886
4	3,67933719	0,91983430	3,35845054	0,83961263	3,47164707	0,86791177
8	4,06258015	0,50782252	6,09635100	0,76204387	3,98818998	0,49852375
16	4,88789507	0,30549344	5,39348585	0,33709287	4,41367745	0,27585484

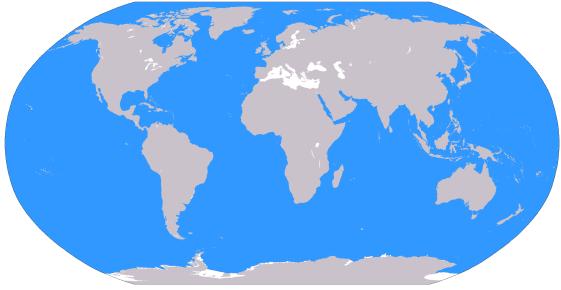
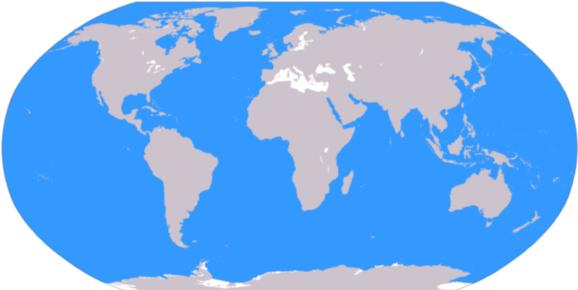


Resultado gráfico	Conclusión del experimento																								
<p style="text-align: center;">Speedup Imagen Mediana</p> <table border="1"><caption>Data for Speedup Imagen Mediana</caption><thead><tr><th>Cores</th><th>Speedup 10</th><th>Speedup 50</th><th>Speedup 100</th></tr></thead><tbody><tr><td>1</td><td>1,00</td><td>1,00</td><td>1,00</td></tr><tr><td>2</td><td>1,80</td><td>1,80</td><td>1,80</td></tr><tr><td>4</td><td>3,50</td><td>3,50</td><td>3,50</td></tr><tr><td>8</td><td>4,00</td><td>6,00</td><td>4,00</td></tr><tr><td>16</td><td>4,50</td><td>5,50</td><td>4,50</td></tr></tbody></table>	Cores	Speedup 10	Speedup 50	Speedup 100	1	1,00	1,00	1,00	2	1,80	1,80	1,80	4	3,50	3,50	3,50	8	4,00	6,00	4,00	16	4,50	5,50	4,50	<p>Se destaca el mejor Speedup de la iteración 50. Creemos que posiblemente existe desbalance al dividir la imagen grupos de potencias Base 2. Por lo que mejora el speedup con múltiplos de 5 iteraciones. Pese a que se ejecutó repetidas veces se logró obtener siempre el mismo resultado cercano a 6 Speedup.</p> <p>Observamos que el aumento de CORE en las 3 iteraciones (10, 50 y 100), se mantienen bastante similares en sus Speedup.</p> <p>Cuando se alcanza CORE 4 y 8, se genera una diferencia notable entre las iteraciones 50 y las 10,100.</p>
Cores	Speedup 10	Speedup 50	Speedup 100																						
1	1,00	1,00	1,00																						
2	1,80	1,80	1,80																						
4	3,50	3,50	3,50																						
8	4,00	6,00	4,00																						
16	4,50	5,50	4,50																						
<p style="text-align: center;">Eficiencia Imagen Mediana</p> <table border="1"><caption>Data for Eficiencia Imagen Mediana</caption><thead><tr><th>Cores</th><th>Eficiencia 10</th><th>Eficiencia 50</th><th>Eficiencia 100</th></tr></thead><tbody><tr><td>1</td><td>1,00</td><td>1,00</td><td>1,00</td></tr><tr><td>2</td><td>0,95</td><td>0,85</td><td>0,90</td></tr><tr><td>4</td><td>0,90</td><td>0,80</td><td>0,85</td></tr><tr><td>8</td><td>0,50</td><td>0,75</td><td>0,55</td></tr><tr><td>16</td><td>0,30</td><td>0,30</td><td>0,25</td></tr></tbody></table>	Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100	1	1,00	1,00	1,00	2	0,95	0,85	0,90	4	0,90	0,80	0,85	8	0,50	0,75	0,55	16	0,30	0,30	0,25	<p>Lo más destacable de este análisis, es la Eficiencia en las 3 iteraciones, cuando son utilizados 4 CORE para la actividad.</p> <p>La eficiencia se reduce de 1 a 0.8, lo cual es un muy buen indicador todavía. Mientras que la eficiencia cae pasando el umbral de 4 a 8 CORE.</p>
Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100																						
1	1,00	1,00	1,00																						
2	0,95	0,85	0,90																						
4	0,90	0,80	0,85																						
8	0,50	0,75	0,55																						
16	0,30	0,30	0,25																						



Resultados experimentales, Imagen PEQUEÑA.

Se ejecuta el programa ./masker para la imagen PEQUEÑA.

Imagen original: world.png – 56 KB	Imagen modificada: Img_world.png – 990 KB
	
<p><i>Justificación de la elección.</i> Elegimos la imagen world.png, considerando que es una de las imágenes con menor peso en Bytes. (56 Kb). El formato considerado es PNG.</p>	

Con esto se obtiene los siguientes tiempos de ejecución:

Threads	Iteraciones		
	10	50	100
1	13,139919	65,714986	131,531881
2	6,997945	36,033334	71,035224
4	4,084180	23,400934	41,650713
8	2,657337	12,784658	26,266155
16	2,613382	12,061834	23,731415

Rendimiento:

Threads	Iteraciones					
	10		50		100	
Threads	Speedup	Eficiencia	Speedup	Eficiencia	Speedup	Eficiencia
1	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000	1,00000000
2	1,87768252	0,93884126	1,82372761	0,91186380	1,85164308	0,92582154
4	3,21727226	0,80431806	2,80822065	0,70205516	3,15797429	0,78949357
8	4,94476952	0,61809619	5,14014423	0,64251803	5,00765647	0,62595706
16	5,02793660	0,31424604	5,44817529	0,34051096	5,54252163	0,34640760



Resultado gráfico	Conclusión del experimento																								
<p style="text-align: center;">Speedup Imagen Pequeña</p> <table border="1"><caption>Data for Speedup Imagen Pequeña</caption><thead><tr><th>Cores</th><th>Speedup 10</th><th>Speedup 50</th><th>Speedup 100</th></tr></thead><tbody><tr><td>1</td><td>1,00</td><td>-</td><td>-</td></tr><tr><td>2</td><td>2,00</td><td>-</td><td>-</td></tr><tr><td>4</td><td>3,00</td><td>2,80</td><td>-</td></tr><tr><td>8</td><td>5,00</td><td>5,00</td><td>5,00</td></tr><tr><td>16</td><td>5,50</td><td>5,20</td><td>5,20</td></tr></tbody></table>	Cores	Speedup 10	Speedup 50	Speedup 100	1	1,00	-	-	2	2,00	-	-	4	3,00	2,80	-	8	5,00	5,00	5,00	16	5,50	5,20	5,20	<p>Observamos que el Speedup se mantiene con un buen rendimiento hasta que se ocupan los 4 CORE.</p> <p>Al comparar los Speedups de las imágenes anterior, el indicador es inferior a todas ellas Aproximadamente por 4 Core, obtenemos 3,1 S. A diferencia de las anterior imágenes que por los mismos 4 CORE, se obtienen 3,2 y 3,4.</p> <p>El costo de ejecutar un proceso de subdividir una imagen pequeña, genera un mayor costo de tiempo, el solo lanzar el programa en cada uno de los Core.</p>
Cores	Speedup 10	Speedup 50	Speedup 100																						
1	1,00	-	-																						
2	2,00	-	-																						
4	3,00	2,80	-																						
8	5,00	5,00	5,00																						
16	5,50	5,20	5,20																						
<p style="text-align: center;">Eficiencia Imagen Pequeña</p> <table border="1"><caption>Data for Eficiencia Imagen Pequeña</caption><thead><tr><th>Cores</th><th>Eficiencia 10</th><th>Eficiencia 50</th><th>Eficiencia 100</th></tr></thead><tbody><tr><td>1</td><td>1,00</td><td>-</td><td>-</td></tr><tr><td>2</td><td>0,95</td><td>0,90</td><td>0,90</td></tr><tr><td>4</td><td>0,70</td><td>0,65</td><td>0,65</td></tr><tr><td>8</td><td>0,60</td><td>0,55</td><td>0,55</td></tr><tr><td>16</td><td>0,30</td><td>0,25</td><td>0,25</td></tr></tbody></table>	Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100	1	1,00	-	-	2	0,95	0,90	0,90	4	0,70	0,65	0,65	8	0,60	0,55	0,55	16	0,30	0,25	0,25	<p>En este análisis el rendimiento se destaca la iteración=50, que decae al ser ejecutado desde 4 Core en adelante.</p>
Cores	Eficiencia 10	Eficiencia 50	Eficiencia 100																						
1	1,00	-	-																						
2	0,95	0,90	0,90																						
4	0,70	0,65	0,65																						
8	0,60	0,55	0,55																						
16	0,30	0,25	0,25																						



Preguntas

En esta sección se debe dar una respuesta fundamentada a las siguientes preguntas:

- ¿Hasta cuánto aumenta el speedup en el nodo escogido? ¿Por qué?
 - Hemos visualizado que los procesos son eficientes considerando hasta los 8 Cores que posee el Nodo (Caleuche).
 - Probablemente, si aumentamos los CORE del proceso, estos quedarán encoladas hasta que algunos de CORE se haya liberado.
- ¿Es eficiente la forma de paralelizar del programa?
 - Según lo evaluado podemos, Si. Es eficiente pues confirmamos que llegando al total de Core del nodo, se pudo obtener buenos indicadores de Speedup y Eficiencia.
- ¿Cómo se puede relacionar OpenMP al ecosistema Hadoop?
 - OpenMP y Hadoop, ambos paralelizan sus tareas/procesos en cada uno de sus CORE. En el caso del uso de memoria, OpenMP comparte la misma memoria entre sus CORE, mientras que Hadoop, no lo hace.
- ¿Qué hace la instrucción collapse(2)? ¿Qué sucede si la elimina del código de masker?
 - Segùn nuestra interpretación, la clausula en el código permite reducir un exceso de la granularidad de tareas paralelas.
 - Entendemos que a menor cantidad de trabajos por threads mayor eficiencia con bloques de memoria grandes, que aquel que posee un mayor número mayor de trabajos on bloques mas pequeños.