INFORME PRÁCTICA
LAB3
ARQUITECTURA DE COMPUTADORES

Ibai Zak Allan Aldanondo 1397750
Ramon Guimerà Ortuño 1400214
Grupo: 419
Fecha: 19/05/2016
Departamento: CAOS

Indice

Indice de tablas, códigos e imágenes

Objetivo de la práctica:

La práctica LAB3 de arquitectura de computadores tiene como objetivo evaluar distintas rutinas de ejecución sobre la misma CPU observando cómo se comportan ante el uso de threads, distinta cantidad de threads y el uso de la CPU combinado con hasta 2 GPUs a la vez.

Ejercicios:

G = 10^9
s = seconds
IPC = Instructions Per Cycle

Ejercicio 1A:

La complejidad del algoritmo es de log(N).
Por teoría dada en clase, binarySearch tiene una complejidad de log(N) en el peor de los casos.

Ejercicio 1B:

La instrucción 3 y 4 representan un if/else. La instrucción 3, si es cierto, L=M. La instrucción 4, si no es cierto, R=M.
Beneficios de cara a la ejecución:

- Se hace on variables de dato integer, lo que és más económico que hacerlo con una instrucción de tipo load.
- Se asigna el valor en la misma comparación.
- No se hacen jumps/brn
- Nos ahorramos los fallos de predicción de saltos

Ejercicio 1C:

No hay vectorización ya que no se sigue un orden de lectura del vector. Dependiendo del if, se escoge un valor u otro lo que hace imposible que se vectorice el vector.

Ejercicio 1D:

Resultados de calcular Instruction Count:

N	ICount (G)
100000	15,5
1000000	18,9
10000000	22,3
100000000	25

Tabla ejercicio 1D

Crece en una complejidad de log2(N), ya que binarySearch crece en esa complejidad.
ICount = log2(N) ≃ 3G.

Ejercicio 1E:

Resultados:

N	Time (s)	Cycles (G)	IPC
100000	8	2,4	6,46
1000000	16,07	4,8	3,94
10000000	37,25	11,2	1,99
100000000	61,9	18,6	1,34

Tabla ejercicio 1E

Decrece ya que los ciclos de ejecución crecen más rápidamente que la cantidad de instrucciones.

Ejercicio 1F:

Speedup obtenido:

N	Speedup
100000	1,06
1000000	1,4
10000000	1,5
100000000	1,467

Tabla ejercicio 1F

El speedup toma como valor máximo 1.5, aunque se puede observar como se estanca en un valor cercano a 1.5 tanto para el segundo como cuarto valor de N.

Ejercicio 1G:

Calculos de Speedup:

N	Instructions (G)	Time (s)	Speedup
100000	17709409644	5,39	1,48
1000000	20883235636	8,93	1,32
10000000	25387978662	23,14	1,1
100000000	45437684943	45,63	1,08

Tabla ejercicio 1G

Existe un mayor número de instrucciones, debido a que son las instrucciones de creación de threads. El hecho de que exista un mayor número de instrucciones, no significa que el speedup baje, ya que al usar más de un thread, la rutina de ejecución se realiza en un periodo de tiempo más corto.

Ejercicio 1H:

La mejora se ha hecho para BinSearchPref.

Tabla de mejoras:

	Original	Mejora
Time (s)	72,4	55,59
Instructions	45698706072	56341138069
Cycles	243946682427	187360984371
IPC	0,19	0,3
Speedup	-------	1,3

Tabla ejercicio 1H

Se incluye el código optimizado:

```
unsigned int BinSearchPref(const int *index, const int N, const int target)
{
   unsigned long L=-1, R=N, M, Ml, Mr;
   int value, v1, v2;
   M=(R-L)>>1;
   while (M >= N/1024)
   {
      M = M + L;
      value = index[M];
      L= (value < target) ? M : L; R= (value >= target) ? M : R;
      M = (R-L)>>1;
   }

   value = index[M+L];

   while (M>=4)
   {
      M = M + L;
      Ml = (L+M)>>1;
      Mr = (R+M)>>1;
      v1 = indx[Ml];
      v2 = index[Mr];
      R= (value >= target) ? M : R;
      L= (value < target) ? M : L;
      value = (value < target)? v2:v1;
      M = (R - L)>>1;
   }

   while (M)
   {
      m = M + L;
      value = index[M];
      L= (value < target) ? M : L; R= (value >= target) ? M : R;
      M = (R - L)>>1;
   }
   return R;
}
```

Ejercicio 2A:

Cálculos de Speedup:

N	Búsqueda (1 thread)	Búsqueda (4 threads)	Speedup
1000000	8,46	2,26 -> 2,46	3,439 -> 3,743
10000000	21,86	6,26 -> 6,36	3,437 -> 3,492
100000000	35,2	10,2 -> 10,4	3,384 -> 3,45

Tabla ejercicio 2A

Existe una mejora de speedup de un 3.4 de media respecto a la ejecución con 1 solo thread.
Con 4 threads existe un incremento de instrucciones, pero no superior a la ejecución de 1 solo thread. Estas instrucciones son de la creación de threads.

Ejercicio 2B:

Resultats Cilk+ amb 8 threads i OpenMP amb 4 threads:

	Cilk+ (8 threads)				
N	ICount (G)	Time (s)	Cycles (G)	IPC min	IPC max
1000000	27 +- 1	2 +- 0,1	6,33 -> 6,99	3,719	4,42
10000000	32 +- 1	5,2 +- 0,1	16,99 -> 17,66	1,755	1,94
100000000	57 +- 1	17,5 +- 0,1	57,99 -> 58,66	0,95	1

Tabla ejercicio 2B.1 cilk+

	OpenMP (4 threads)				
N	ICount (G)	Time (s)	Cycles (G)	IPC min	IPC max
1000000	27,2	2,9 +- 0,1	9,33 -> 9,99	2,722	2,915
10000000	32,5	7,65 +- 0,05	25,33 -> 25,66	1,266	1,283
100000000	57,4	21,1 +- 0,1	69,99 -> 70,65	0,81	0,82

Tabla ejercicio 2B.2 openmp

Con 8 threads, el número de instrucciones no cambia, pero si el tiempo y la cantidad de ciclos. A mayor cantidad de threads, mejor el IPC para los mismos valores de N.

Se observa en la siguiente tabla la mejora por speedup de ejecutar con Cilk+ u OpenMP:

	Cilk+		OpenMP			
N	Time max	Time min	Time max	Time min	Speedup max	Speedup min
1000000	2,1	1,9	3	2,8	1,578947368	1,333333333
10000000	5,3	5,1	7,7	7,6	1,509803922	1,433962264
100000000	17,6	17,4	21,2	21	1,218390805	1,193181818

Tabla ejercicio 2B.3

Ejercicio 2C:

Se observa en la siguiente tabla los datos obtenidos:

N	Inst Host (G)	Inst Device (G)	Cycles CPU (G)	Cycles GPU (G)	Time (s)
1000000	11	35	8,58	2,6156	2,6
10000000	12,2	41	10,23	3,1186	3,1
100000000	19,7	69,7	14,52	4,426	4,4

Tabla ejercicio 2C.1

Resultados de IPC (G):

	IPC		IPC Cilk+		IPC OpenMP	
N	CPU	GPU	max	min	max	min
1000000	12,28	13,36	4,42	3,719	2,915	2,722
10000000	1,19	13,14	1,94	1,755	1,283	1,266
100000000	1,35	15,73	1	0,95	0,82	0,81

Tabla ejercicio 2C.2

Visto que Cilk+ es más rápido que OpenMP, se realiza la comparación por Speedup con Cilk+ en vez de OpenMP (todos los tiempos son calculados en segundos):

N	GPU+CPU Time (s)	Cilk+ Time max	Cilk+ Time min	Speedup max	Speedup min
1000000	2,6	2,1	1,9	0,8076923077	0,7307692308
10000000	3,1	5,3	5,1	1,709677419	1,64516129
100000000	4,4	17,6	17,4	4	3,954545455

Tabla ejercicio 2C.3

GPU speed = 1,006 GHz
CPU speed = 3,333 GHz

El IPC de la GPU és casi constante, mientras que el de la CPU bajca de 12,28 a 1,19 quando N aumenta 10 veces. Su reducción es de casi 12 veces mientras que en Cilk+ u OpenMP su reducción es de 4 y 2 veces respectivamente.
La ejecución de CPU+GPU es mucho más rapida (hasta 4 veces más) que sólo CPU con 8 threads. Los datos de speedup entre CPU+GPU y Cilk+ lo demuestran.
Cuando N=1000000, sufre un speedup menor debido a la traída de datos de memoria a la GPU, cuando N es superior, la traida de datos es la misma, pero una vez esos datos se encuentran en la GPU, no es necesario volver a tener que leer de memoria, lo que no produce fallos y se consigue un speedup superior.

Ejercicio 2D:

Resultados obtenidos con 8 y 4 threads:

	8 threads		4 threads	
N	Instructions	Time	Instructions	Time
1000000	29667288645	2,21	27257793083	2,27
10000000	33474630648	5,25	32372705411	7,63
100000000	60986321817	17,78	57521587768	21,22

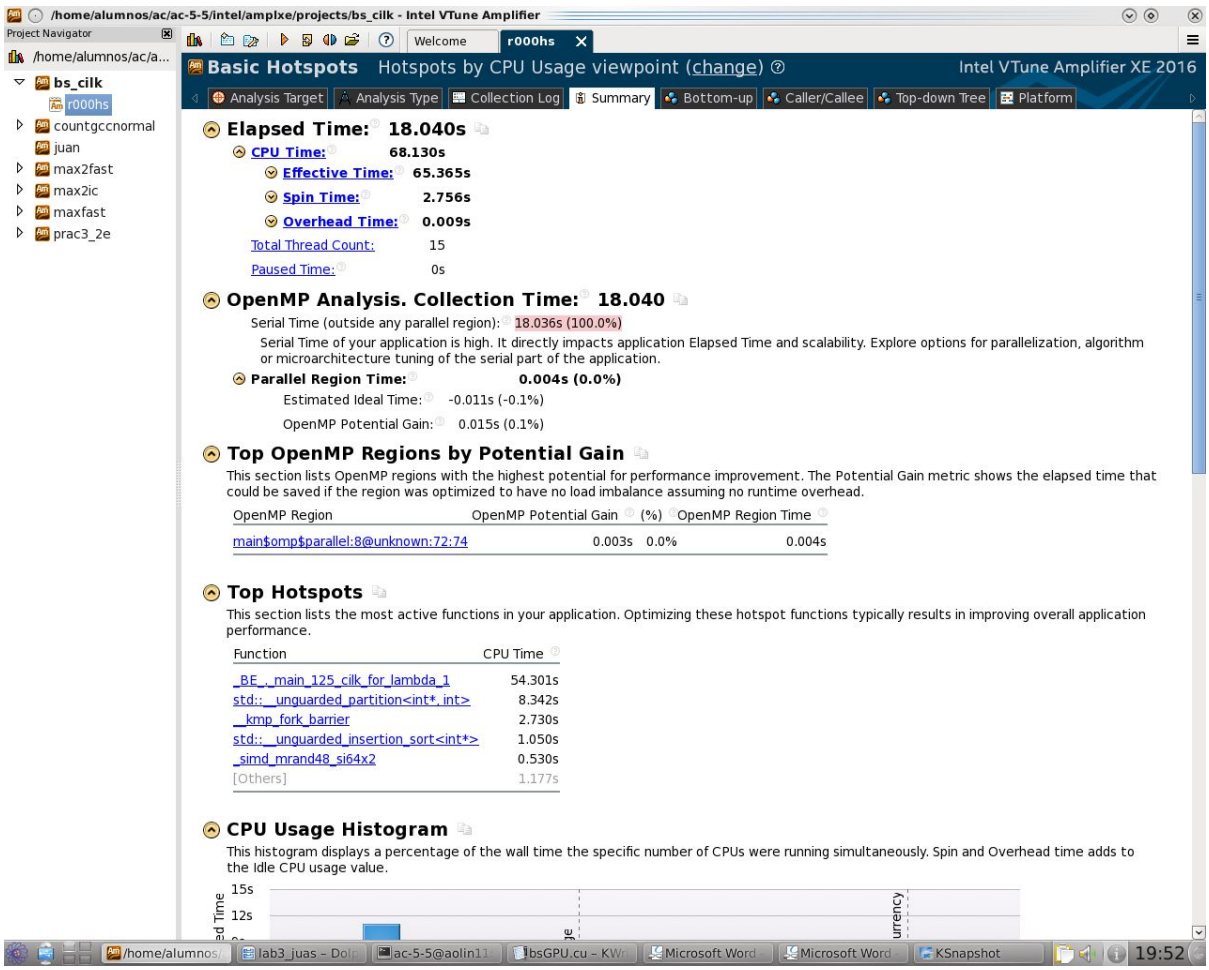
Tabla ejercicio 2D

El hecho de que con 8 threads exista un mayor número de instrucciones es devido a que son precisamente las instrucciones de creación de threads. Tal i como se ha dicho anteriormente, con 8 threads tardamos menos en realizar la rutina de ejecución.

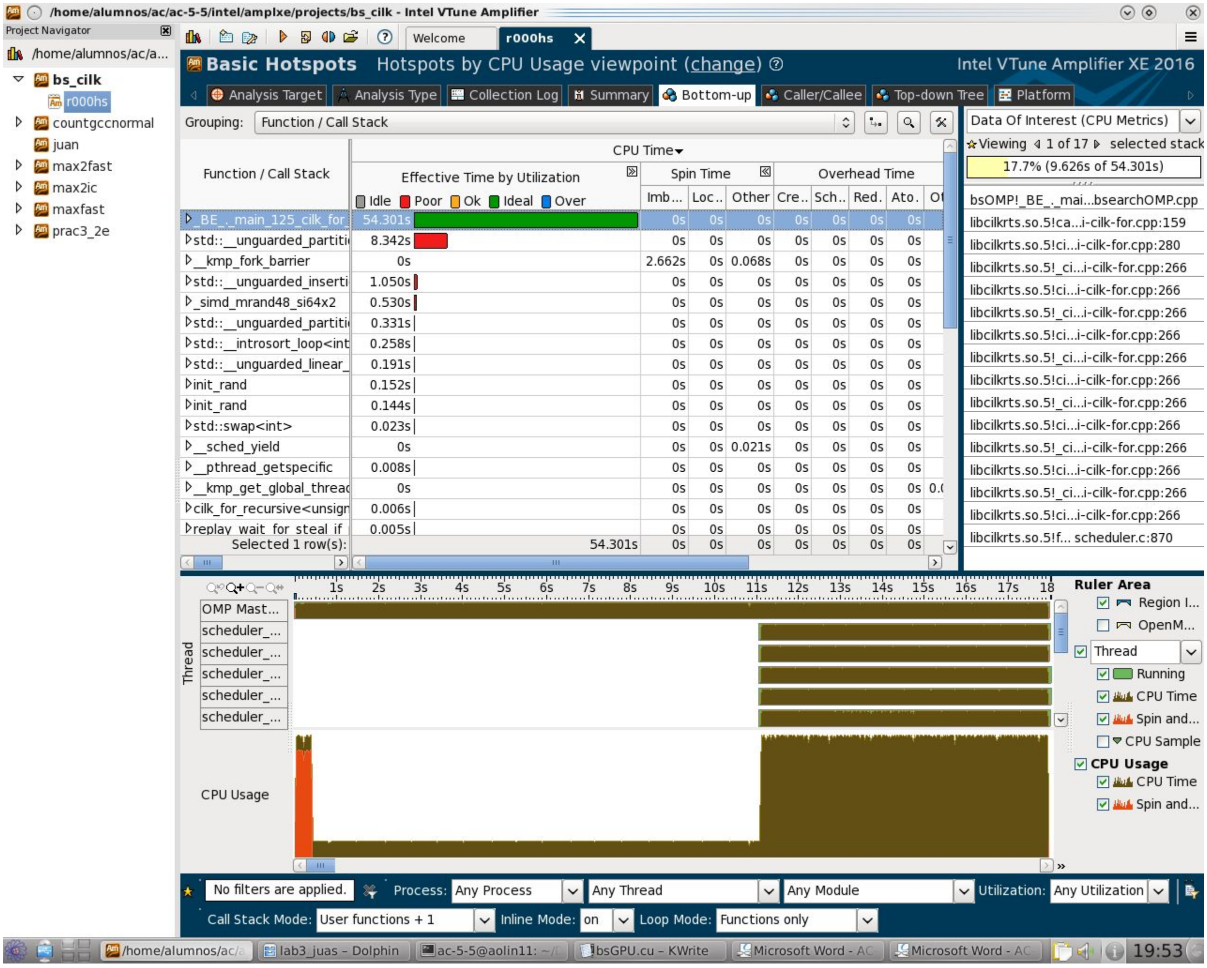
Ejercicio 2E:

Se adjuntan las imágenes extraídas de ejecutar VTune:

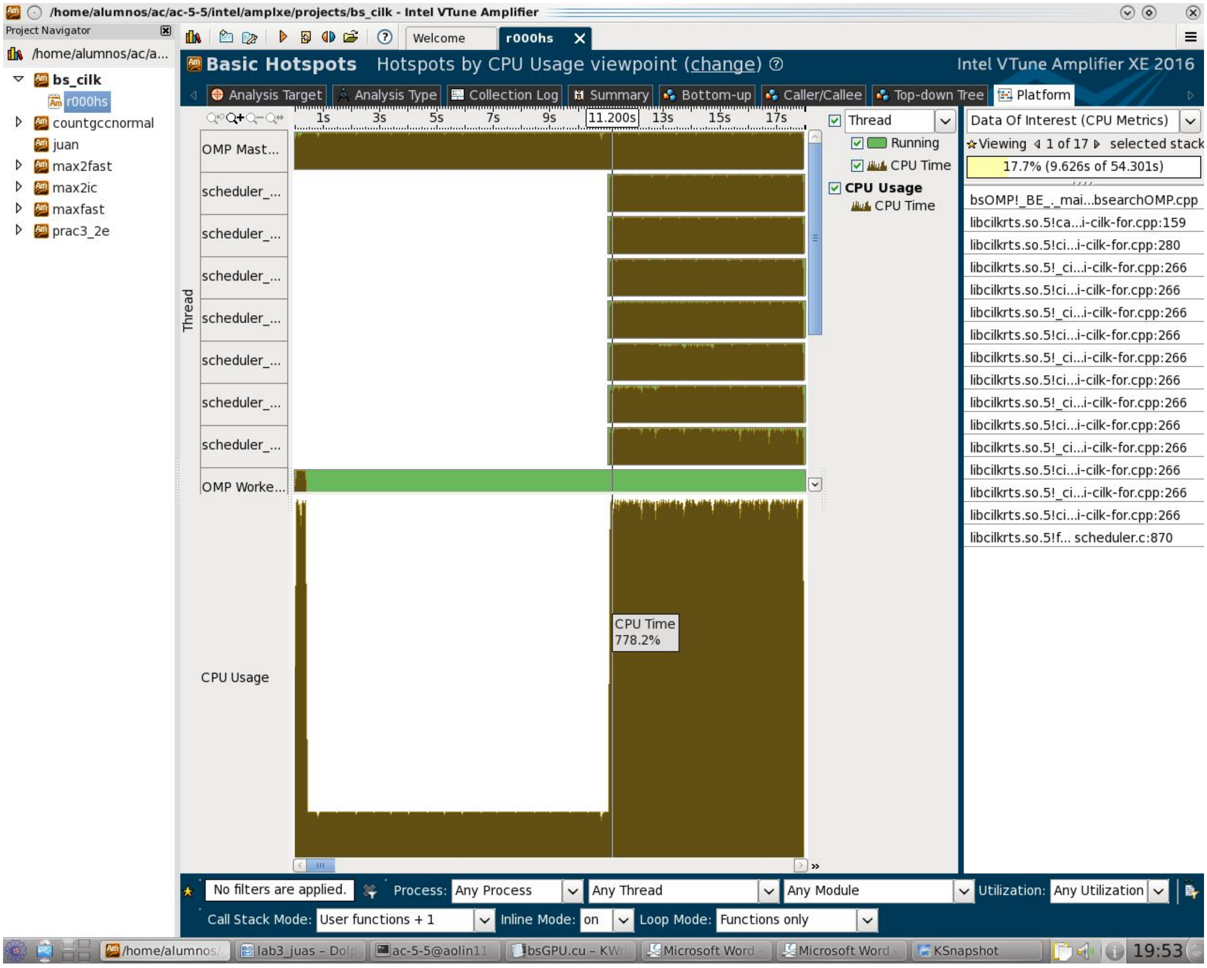
Cilk+



Summary Cilk+

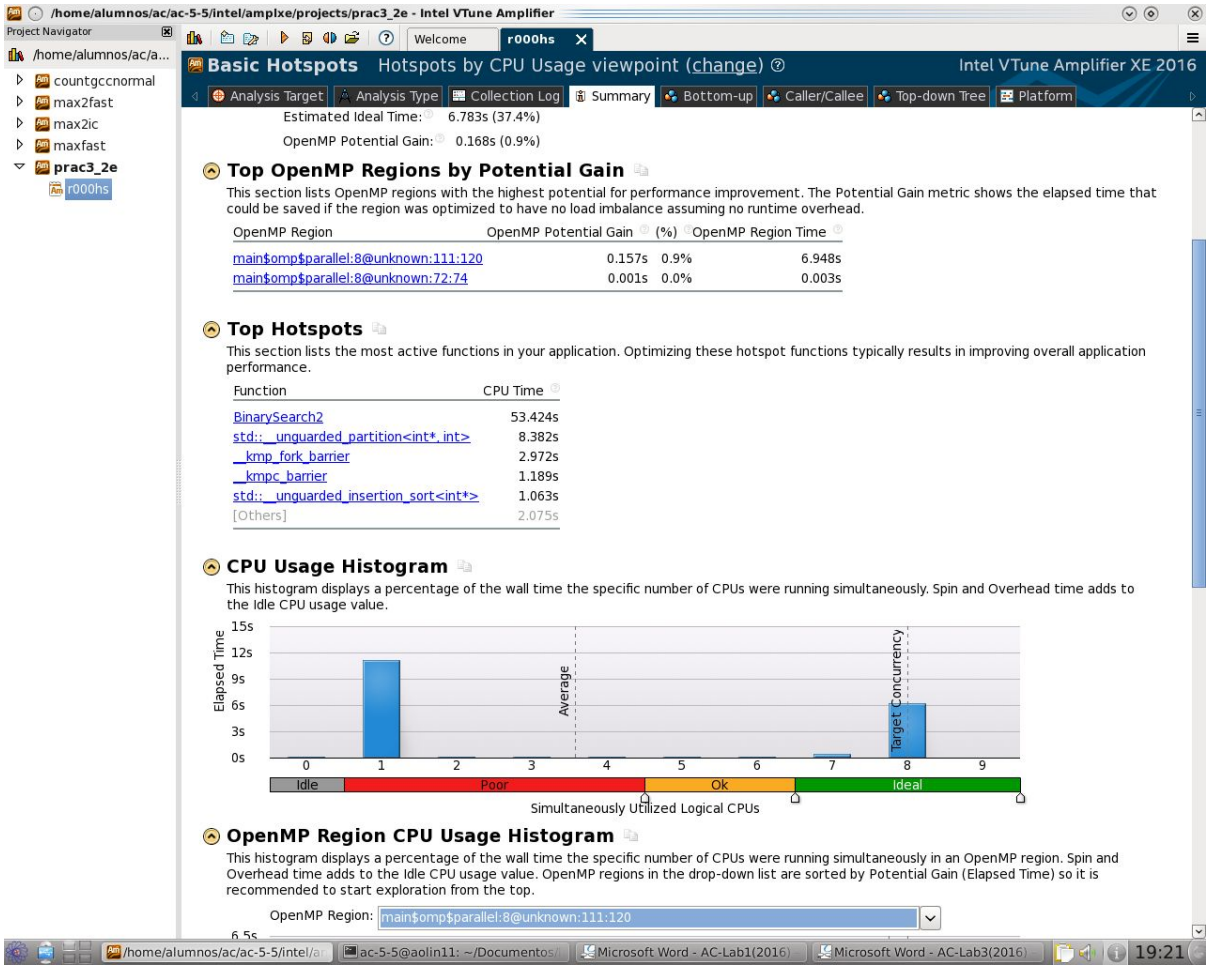


Bottom-Up Cilk+

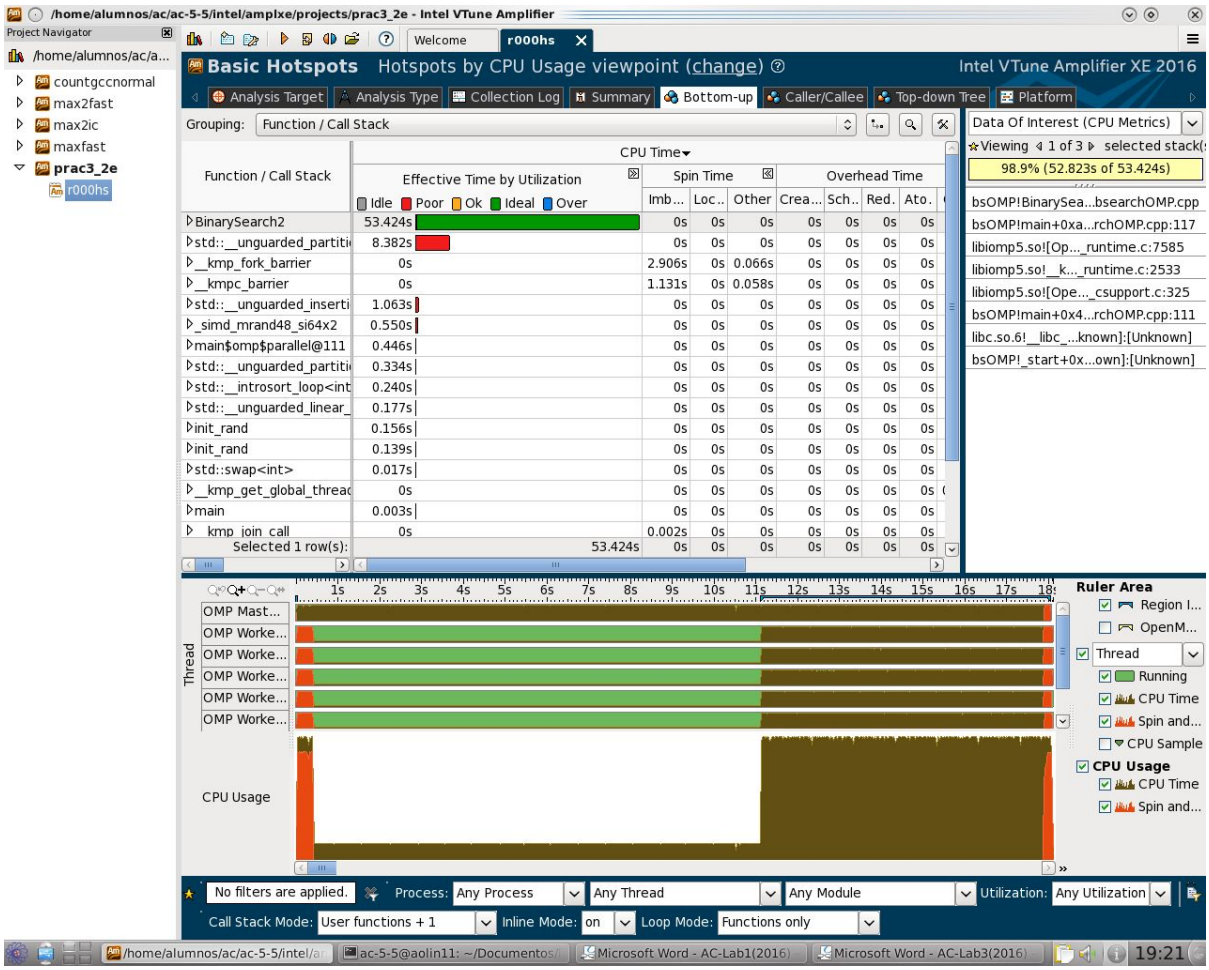


Platform Cilk+

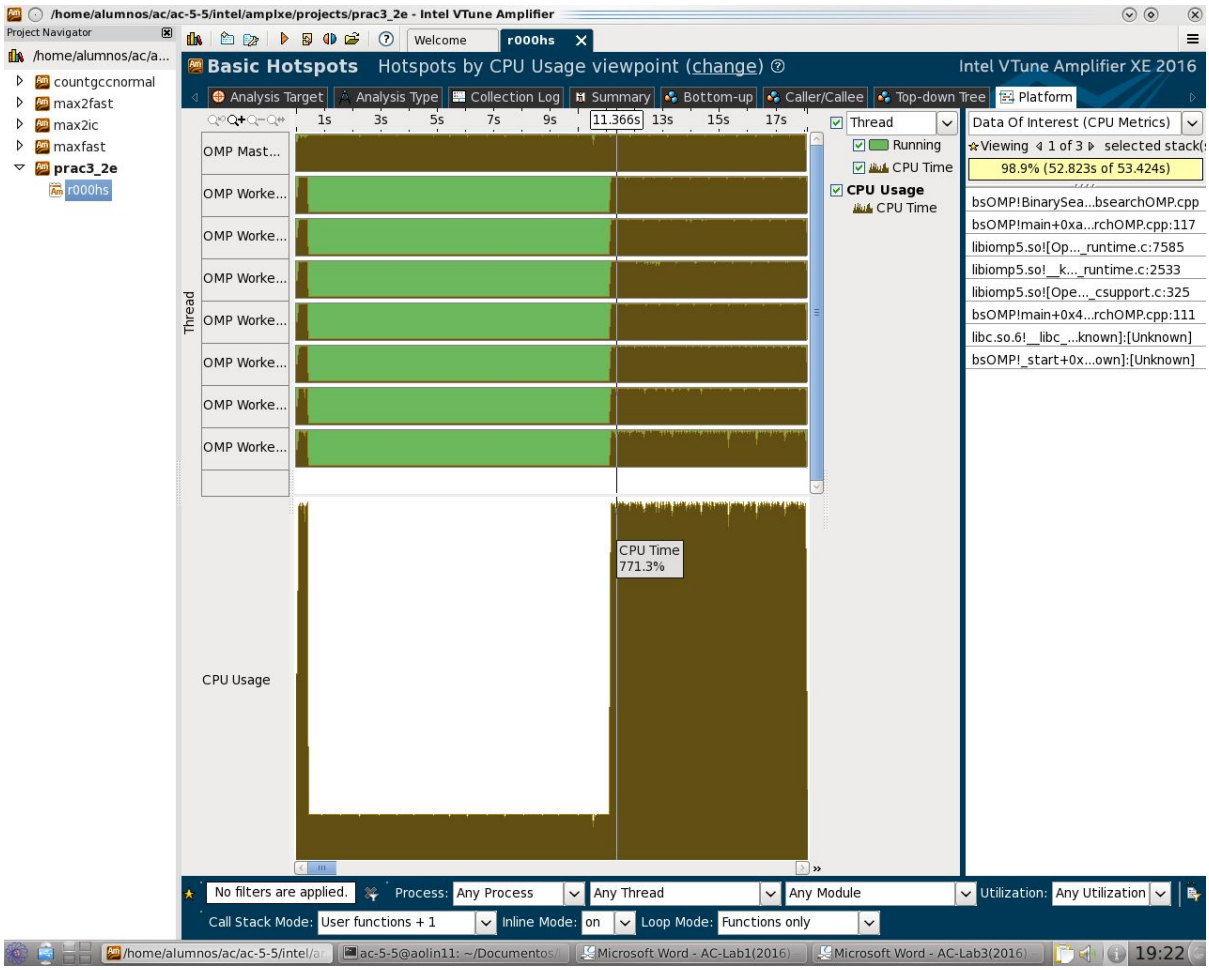
OpenMP



Summary OpenMP



Bottom-Up OpenMP



Platform OpenMP

Se pueden observar en las imágenes y sobretodo en las imágenes de Platform (tanto de una ejecución como de la otra) el número de threads en ejecución. También, de la imagen de Summary, podemos obtener el tiempo total del proceso en CPU.

Ejercicio 2F:

Datos de las GPUs del sistema:

	GTX 480	GT 430
GPU Clock Rate	1401 MHz	1400 MHz
Memory Clock Rate	1848 MHz	800 MHz
Memory Bus Width	384 bits	128 bits
Global Memory Size	1610285056	10702648414
L2 Cache Size	786432 bytes	131 022 bytes
Nº Multiprocesadores	15	2
Nº Nucleos de Ejecución	480	64
Ancho de Banda memoria(GB/s)	88,704	12,8
Núcleos de Cómputo		

Tabla ejercicio 2F

Ejercicio 2G:

Resultados:

	GTX 480			GT 430		
N	Inst	Time	IPC	Inst	Time	IPC
1000000	9933854010	1,91	1,55	36683287333	8,6	1,31
10000000	11139684686	2,32	1,43	ERROR		
100000000	12189777383	1,89	1,93	ERROR		

Tabla ejercicio 2G

Se ve claramente como la GPU GTX480 es mucho superior que GT430 tanto para la ejecución con N=1000000 como para con valores de N superiores, ya que en esos casos, la GPU GT430 da error de ejecución.
El error en la ejecución en la GPU GT430 es debido a que no es capaz de traer todos los datos de memoria.

Ejercicio 2H:

Anchos de banda:

	Ancho de Banda	
	Calculado	Obtenido (GB/s)
GTX 480		117,5
GT 430		25,6

Tabla ejercicio 2H

Visto que en el enunciado ya aparecen las acciones que se ejecutan en GT430, se adjunta la imagen para observar lo que sucede en GTX480:

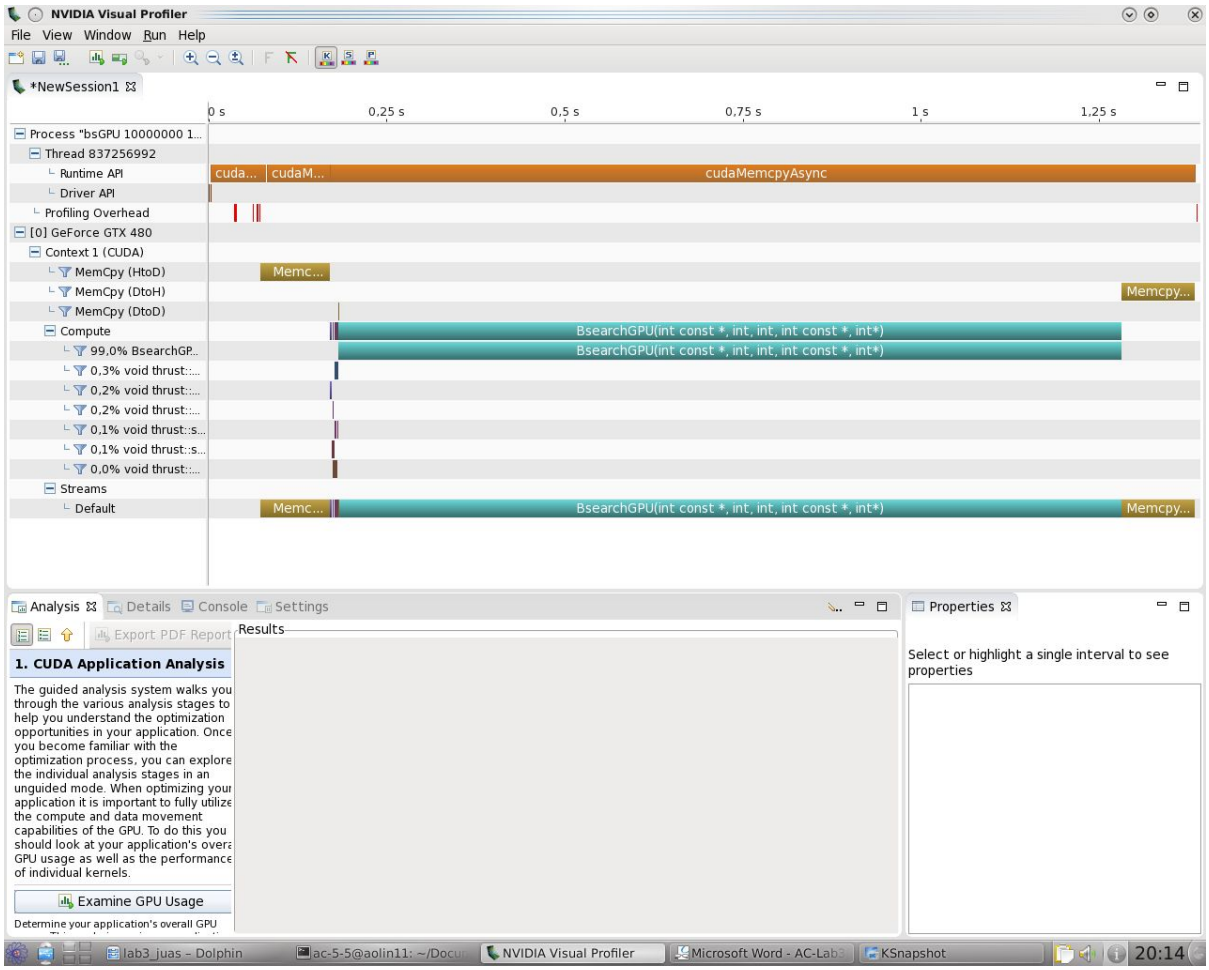


Imagen NVIDIA

Barra de acciones:

CudaMalloc	CudaMemcpy Async	CudaMemcpy Async	CudaFree

Ejercicio 2I:

Resultados obtenidos:

1) 1,099 s
2) 150,164 GB/s
3) 27267061872

Conlcusiones:

Con los resultados obtenidos en el transcurso de ambas sesiones de trabajo de laboratorio, sumado al trabajo previo, podemos observar y por ende, concluir que, el uso de threads en un programa bajo la misma CPU consiguen como beneficio que la ejecución de la rutina sea mucho más rápida que sin el uso de threads.
Además, dependiendo de que opción se use, 8 threads no significan una mejora muy considerable ante 4 threads.

Por otro lado, en el caso del uso de CPU+GPU, la ejecución de la misma rutina es considerablemente más provechosa que sin el uso de GPU.