



Universidad
Nacional
de Córdoba

Cátedra de Sistemas Operativos II

Trabajo Práctico N° II

Salvatierra, Andrés
17 de mayo de 2019



Índice

Introducción	3
Propósito	3
Ámbito del Sistema	3
Definiciones, Acrónimos y Abreviaturas	3
Referencias	3
Descripción General del Documento	3
Descripción General	4
Perspectiva del Producto	4
Funciones del Producto	4
Características de los Usuarios	4
Restricciones	4
Suposiciones y Dependencias	5
Requisitos Específicos	5
Interfaces Externas	5
Funciones	5
Requisitos de Rendimiento	5
Restricciones de Diseño	6
Atributos del Sistema	6
Diseño de solución	7
Implementación y Resultados	9
Conclusiones	27

Introducción

En este trabajo se presenta la resolución de una convolución entre una imagen y un filtro de borde. Se lleva a cabo la implementación del mismo de forma procedural y de forma distribuida demostrando los beneficios de este último. Entre las técnicas y estándares más utilizados para sistemas de memoria compartida y memoria distribuida, se encuentra OpenMP y MPI, para la implementación de este práctico se utiliza la primera.

Propósito

El objetivo de este práctico es la resolución del problema utilizando el paradigma de memoria distribuida utilizando OpenMP. Primero se plantea una solución de manera procedural en la computadora de escritorio, para luego desplegar el mismo en el clusters aprovechando el paralelismo de la mejor manera posible.

Ámbito del Sistema

Es necesario instalar las librerías NetCDF4 sobre los sistemas en donde se va a ejecutar. Estas librerías permiten compartir información tecnológica, independiente de la arquitectura del sistema y también definen un formato de datos que se transformó en estándar abierto. La ejecución del sistema procedural se ejecuta en una computadora de escritorio, mientras para lograr el mejor rendimiento y aprovechamiento del paralelismo se ejecuta en el cluster proporcionado por la facultad.

Definiciones, Acrónimos y Abreviaturas

- Filtro de borde: es la matriz[3x3] con la que convolucionamos la imagen proporcionada por el satélite, cuyo objetivo es ponderar el valor de los pixeles de borde.

Referencias

[1] Filminas de SOII - Facultad de Ciencias Exactas Físicas y Naturales [Hugo Carrer]

[2] <https://www.unidata.ucar.edu/software/netcdf/examples/programs/>

[3] <http://supercomputingblog.com/openmp/tutorial-parallel-for-loops-with-openmp/>

[4] <https://stackoverflow.com/questions/10811439/malloc-an-array-of-struct-pointers>

Descripción General

Perspectiva del Producto

El práctico solicitado es un software que lea un archivo binario con información expresada en forma de una matriz [21696x21696] representa la Tierra, donde cada punto de la misma representa un pixel de la imagen de la Tierra.

La ejecución del software debe hacerse primero de una manera secuencial, ejecución del programa procedural, y luego en forma paralela, utilizando las librería de OpenMp.

Funciones del Producto

El producto ofrece las siguientes funciones:

- Lectura de un archivo binario con información sobre la imagen de la Tierra.
- Interpretación de los valores leídos.
- Convolución a partir del filtro de borde.
- Almacenamiento del resultado obtenido en otro archivo binario.

Características de los Usuarios

El usuario debe contar con el archivo binario donde se encuentra la información de la Tierra.

Restricciones

No se garantiza seguridad ni encriptación de los datos. No se especifica un número de hilos para el desarrollo de la paralelización, se espera un uso razonable de los mismos.

Suposiciones y Dependencias

- Se supone que la PC en la que correrá tiene instalada una distribución de Linux.
- El archivo binario de donde se obtiene la información no debe ser corrupto.
- Se deben tener las librerías instaladas de OpenMp.



Requisitos Específicos

Interfaces Externas

Para ejecutar el programa se utiliza la terminal, tanto en la ejecución del programa procedural como aquel que se corre en el cluster de la facultad.

Funciones

Las funciones que proporciona el software son:

- Lectura de un archivo binario, el cual contiene la imagen de la Tierra desplegada en una matriz donde cada una de sus posiciones es un pixel de la imagen.
- Algoritmo de filtro de borde utilizado en la convolución.
- Escritura de archivo binario con el resultado obtenido.
- El gráfico resultante a partir del archivo binario se realiza en un archivo aparte.

Requisitos de Rendimiento

Como requisito fundamental se espera un correcto uso de las librerías de OpenMP para la ejecución en paralelo y de esta manera demostrar la ganancia de tiempo entre la ejecución secuencial y paralela.

Restricciones de Diseño

- El programa procedural puede desarrollarse en C o Python, mientras el programa utilizado para la ejecución en el cluster debe ser desarrollado en lenguaje C.
- Se realiza una primera implementación secuencial(procedural) y una posterior utilizando las librerías OpenMp.
- Se debe implementar un algoritmo que aplique un filtro de borde.
- La imagen filtrada, es decir, el resultado de la convolución se debe guardar en un archivo binario.
- Debe incluirse un mecanismo de control y manejo de errores en todo el sistema.
- Documentación del código.
- Utilizar Cppcheck y compilar con el uso de las flags de -Werror, -Wall y -pedantic.
- Se debe realizar una recopilación de datos de varias ejecuciones del programa (30 estadísticamente suficiente) tanto para el programa procedural como en el que explotamos el uso de paralelismo.
- A partir de los datos recopilados, realizar gráficas comparando los resultados.
- Se debe utilizar alguna herramienta de profiling para el análisis de tiempos.

Diseño de solución

Se inicia el diseño de la solución implementando una lectura correcta del archivo binario. La lectura del archivo binario se realiza mediante la función `get_vara_float()`, indicando un inicio y un fin (estos son de dos dimensiones debido a que es una matriz).

Esta información obtenida es almacenada en un arreglo de punteros para posterior manipulación. A estos datos se los filtra en busca de valores del tipo “Nan” para que en el momento de graficar la imagen estos mejoran la visibilidad de la implementación del filtro de borde.

El filtro de borde está definido de la siguiente manera:

$$\omega = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

La imagen filtrada se obtiene a partir de la convolución:

$$g(y, z) = \omega * f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) f(x - s, y - t)$$

Donde $g(x,y)$ es la imagen filtrada, $f(x,y)$ es la imagen original y w es el filtro de borde.

Se realiza la convolución a partir de los datos almacenados en el arreglo y el filtro de borde. La convolución consiste en posicionarse en el punto donde deseamos ponderar, se recorren los puntos que rodean al mismo y se multiplica por el filtro de borde, estos resultados se suman y se almacenan en la posición donde me encontraba pero de una nuevo arreglo, es decir el arreglo original(la información original) no es alterado, y de esta forma se realiza con cada punto del arreglo original.

data_in										Resultante				
5	2	2	2	2						0	0	0	0	0
2	5	2	2	2		Filtro_borde				0	18			0
2	2	5	2	2		-1	-1	-1		0				0
2	2	2	5	2		-1	8	-1		0				0
2	2	2	2	5		-1	-1	-1		0				0
2	2	2	2	2						0	0	0	0	0

```

void conv(int x, int y, float *data_in, float *resultante, float mat_w [WX][WY])
{
    // struct timespec start, end;
    double start_time = omp_get_wtime();
    // if( clock_gettime( CLOCK_MONOTONIC_RAW, &start) == -1 ) {
    //     perror( "clock_gettime" );
    //     exit( EXIT_FAILURE );
    // }

    // #pragma omp parallel for num_threads(4)
    for(int i=x; i<NX_T-1; i=i+1)
    {
        for(int j=y; j<NY_T-1; j=j+1)
        {
            resultante[i*NX_T+j] = (data_in[(i-1)*NX_T + (j-1)]*mat_w[0][0] + data_in[(i-1)*NX_T + j]*mat_w[0][1] + data_in[(i-1)*NX_T + (j+1)]*mat_w[0][2] +
            data_in[i*NX_T + (j-1)]*mat_w[1][0] + data_in[i*NX_T + j]*mat_w[1][1] + data_in[i*NX_T + (j+1)]*mat_w[1][2] +
            data_in[(i+1)*NX_T + (j-1)]*mat_w[2][0] + data_in[(i+1)*NX_T + j]*mat_w[2][1] + data_in[(i+1)*NX_T + (j+1)]*mat_w[2][2])*0.00031746;
        }
    }
}

```

Ejemplo del punto [1,1] de la matriz data_in. Los ceros al borde de la matriz resultante implican que estos no pueden calcularse.

Una vez realizada toda la convolución, el arreglo resultante se escribe en un archivo binario. Este se grafica en python.

Para la implementación de forma paralela utilizando OpenMp, se pusieron secciones paralelas en todos los bucles for que tenían un número elevado de iteraciones, logrando dividir el trabajo entre varios hilos y logrando una mayor velocidad de ejecución.

Implementación y Resultados

Para la implementación como se mencionó anteriormente se utilizaron funciones de lectura y escritura de archivos binarios de las librerías de C, mientras que para el procesamiento y manipulación de los datos se llevó a cabo bucles de iteración for en distintas partes del programa para lograr procesar los datos.

Se verificó que las funciones de lectura y escritura de los datos en los archivos binarios se hicieran correctamente. Además se verificó que la implementación de la convolución se realice de forma adecuada a partir de matrices de prueba.

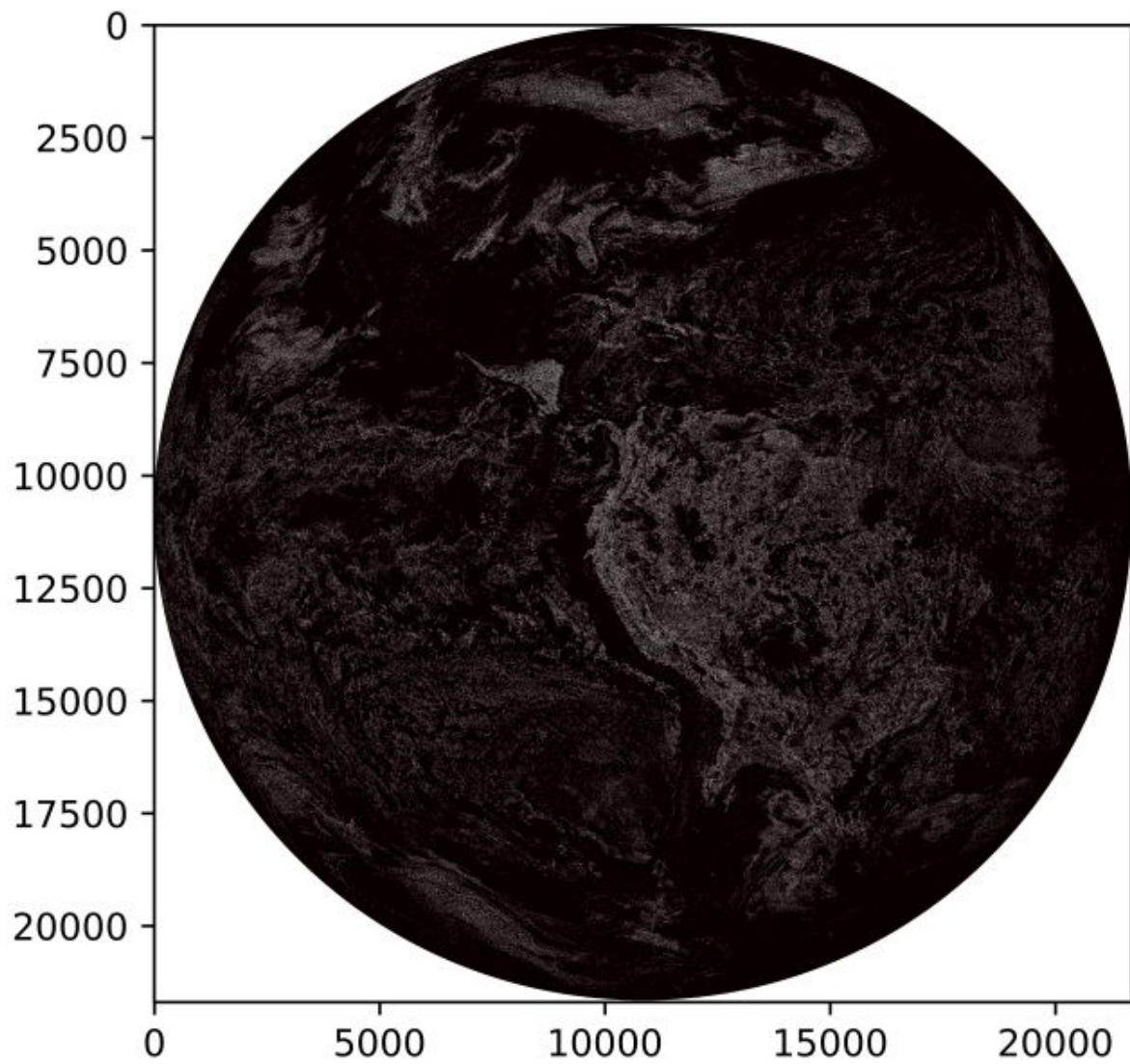
El práctico se dividió en dos mitades. La primera se centra en la ejecución de un programa procedural (sin explotar el paralelismo) con un solo hilo.


Primero se muestra la ejecución del programa procedural en la notebook:

```
andres@andres-X556UQK:~/Facultad/S011/Andres/Practico/SistemasOperativos2019/S02_TP2$ make
gcc -Wall -Werror -pedantic -O3 -o matriz convolucion.c -fopenmp `nc-config --cflags --libs`
andres@andres-X556UQK:~/Facultad/S011/Andres/Practico/SistemasOperativos2019/S02_TP2$ ./matriz
0.319110 Tiempo que el pasaje a nan
1.169918 Tiempo que demora la convolucion
```

Se puede observar que el archivo binario se creó correctamente

Se adjunta el resultado después de aplicar el filtro de borde.





Se llevó a cabo la paralelización de dos bloques de código, el primero donde se llevó a cabo la puesta de “NaN” en la matriz y el segundo bloque la convolución. Para poder realizar un análisis estadístico es necesario correr mínimo 30 veces el programa (estadístico suficiente); se agregó una medida de tiempo de ejecución en el mismo programa (librería time.h y funciones de openMP). Se utilizó compiladores de GCC e ICC para llevar a cabo este análisis estadístico.

Primera implementación:

- GCC

Se puede observar una ganancia de rendimiento de hasta 384,94 ms.

N.º de ejecución	GCC	
	1 hilo	64 hilos
1	445252	61962
2	447495	64547
3	449300	64496
4	449977	59992
5	448843	61755
6	446463	65336
7	448432	66264
8	445478	68422
9	444914	64014
10	445992	69114
11	444460	60749
12	446090	63256
13	447758	59459
14	446555	57872
15	447368	59801
16	447684	58571
17	447662	64681
18	446532	64045
19	446973	64565
20	446159	63976
21	448828	66256
22	445326	66470
23	446453	62825
24	452070	65089
25	494834	61013
26	446203	62898
27	444613	64154
28	448803	64108
29	447362	68667
30	443483	64698
[us]	448578,7333	63635,16667
[s]	0,448578733	0,063635167

Finalmente, se compararon distintas ejecuciones del programa paralelo con distinto número de hilos:

N.º de ejecución	GCC				
	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos
1	456307	265991	157532	87083	65592
2	406930	259273	150958	86364	78336
3	459023	262596	153830	88274	56080
4	452172	263326	154348	88738	59780
5	453591	258987	157252	91437	69304
6	455076	257721	167800	89815	55936
7	446276	257580	171417	91781	57097
8	451149	266085	152952	91661	59913
9	455234	259493	157992	90568	64183
10	522110	256116	157087	88096	64447
11	399744	264417	158906	86028	55755
12	456489	259765	170669	91573	58348
13	454026	259338	153740	87637	56970
14	447924	260220	151019	89572	55142
15	458020	264051	154652	90646	58150
16	455565	256776	152527	89072	69849
17	425212	257843	150903	90649	71017
18	450628	262208	154754	88924	81617
19	450526	260936	161859	89001	68824
20	448348	259826	157570	90322	72569
21	452055	254908	154705	90449	54973
22	454483	261329	152554	89439	59059
23	452670	263356	152759	86567	100133
24	415182	263863	167595	88493	58327
25	435578	258192	152832	89679	67624
26	409617	264489	153337	89338	59600
27	446479	255246	155654	89333	56431
28	448287	262664	153635	82955	83347
29	455215	256755	160750	87534	57104
30	449893	231660	154557	88189	96690
[us]	447460,3	259500,3333	156871,5	88973,9	65739,9
[s]	0,4474603	0,259500333	0,1568715	0,0889739	0,0657399

Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora.

- ICC(xCORE_AVX512)

Se puede observar una ganancia de rendimiento de hasta 108 ms

N.º de ejecución	ICC	
	1 hilo	64 hilos
1	196912	110988
2	194155	124718
3	198525	142289
4	198801	124416
5	199700	121596
6	196189	123334
7	198259	151830
8	198441	115353
9	199145	121927
10	195035	122344
11	266498	134941
12	279841	111352
13	195393	150494
14	197460	139557
15	213396	137832
16	282259	112300
17	283673	127087
18	280441	117238
19	271020	121626
20	195174	126464
21	279334	128543
22	268591	128176
23	198474	144495
24	269558	114399
25	282880	132922
26	282482	116595
27	268863	125381
28	196510	119262
29	279328	121096
30	268200	123272
[us]	234484,5667	126394,2333
[ms]	234,4845667	126,3942333
[s]	0,234484567	0,126394233

N.º de ejecución	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos	64 hilos
1	145520	109921	102403	85375	106761	110988
2	170533	114944	97933	88970	88867	124718
3	169438	111117	106239	100442	115528	142289
4	157659	115408	95341	102911	112425	124416
5	144969	116211	94479	108500	108365	121596
6	145299	101608	105780	104736	112208	123334
7	154913	98435	98337	99305	114159	151830
8	145485	115635	111044	108262	118479	115353
9	145639	98943	107101	105588	117497	121927
10	145234	102072	106269	106173	114727	122344
11	146387	111444	99266	89369	112662	134941
12	145103	100203	129474	92277	117769	111352
13	146137	107299	101924	88808	115368	150494
14	143136	101223	106712	99761	115084	139557
15	140870	110138	92404	87701	113899	137832
16	145843	111797	102419	106314	118943	112300
17	145542	114664	112970	92366	100199	127087
18	117738	120681	106043	107802	119057	117238
19	144228	99389	98652	106508	113839	121626
20	168021	99014	105708	93996	118667	126464
21	170554	100705	96843	94258	118371	128543
22	171095	102053	104458	109368	119306	128176
23	144570	115920	110345	104913	109263	144495
24	144917	114166	106586	97797	110883	114399
25	169605	116076	91847	90427	114494	132922
26	168546	108412	105587	88630	107740	116595
27	144052	113976	104081	96953	109038	125381
28	170603	99308	93601	109399	112513	119262
29	168972	110564	106711	93218	111991	121096
30	144498	113922	105317	105416	119614	123272
[us]	152170,2	108508,2667	103529,1333	98851,43333	112923,8667	126394,2333
[ms]	152,1702	108,5082667	103,5291333	98,85143333	112,9238667	126,3942333
[s]	0,1521702	0,108508267	0,103529133	0,098851433	0,112923867	0,126394233

Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora hasta un cierto punto, si se aumenta el número de hilos a partir de este momento el rendimiento del sistema decae. Si se compara entonces con este punto se obtiene una ganancia de rendimiento de hasta 135,63 ms

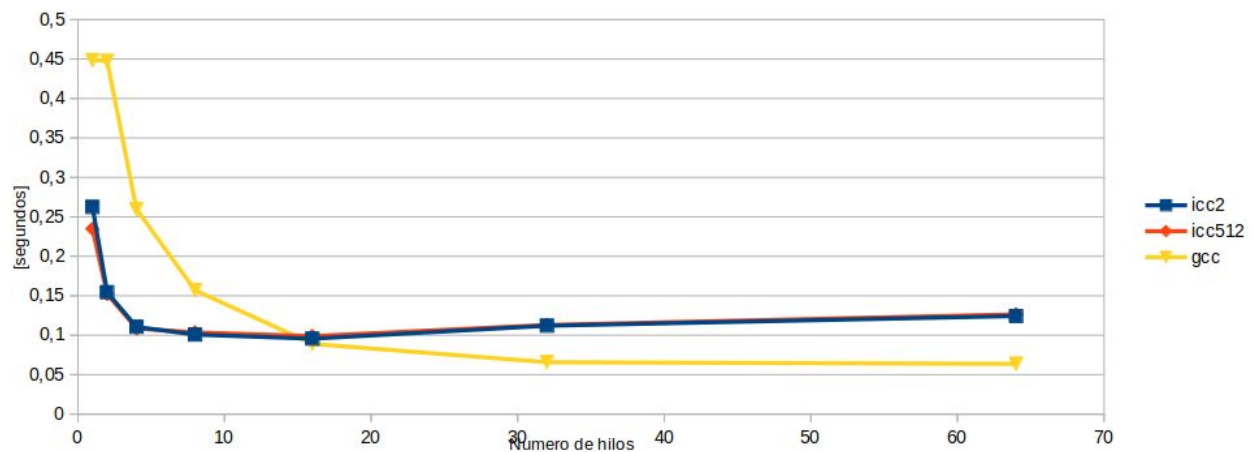
- ICC (xCORE_AVX2)

Se puede observar una ganancia de rendimiento de hasta 138.17 ms

N.º de ejecución	ICC	
	1 hilo	64 hilos
1	274922	123106
2	283974	120680
3	271259	136708
4	271254	119763
5	196118	125813
6	273834	114510
7	285624	139394
8	276055	114965
9	196795	113517
10	198762	127009
11	198477	118450
12	201199	131571
13	270131	119360
14	285979	117741
15	206966	135651
16	285839	119263
17	272631	134721
18	275412	118170
19	276905	116686
20	273542	112064
21	285422	138459
22	274152	128821
23	271313	128497
24	292482	117008
25	272474	123692
26	273848	116589
27	285554	124486
28	285042	126795
29	282176	120425
30	275740	144836
[us]	262462,7	124291,6667
[ms]	262,4627	124,2916667
[s]	0,2624627	0,124291667

N.º de ejecución	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos	64 hilos
1	149481	103317	85797	100462	85014	123106
2	138461	118028	103620	105247	105504	120680
3	143112	117505	100664	100319	108383	136708
4	146538	112250	95177	97961	115664	119763
5	149978	116849	103091	101410	113680	125813
6	162279	108965	72011	85810	113851	114510
7	171523	120328	101544	87294	118936	139394
8	159465	105326	94797	88963	115829	114965
9	162091	111358	106621	107651	107980	113517
10	165805	101293	106154	92692	114597	127009
11	173898	99714	102949	87224	117035	118450
12	169170	112016	105748	98163	111967	131571
13	172347	117571	112041	101915	109103	119360
14	171856	99303	103803	84107	111216	117741
15	167304	111995	94304	92991	110414	135651
16	159162	99867	106946	107589	118467	119263
17	129391	98983	102725	89128	114500	134721
18	126189	109313	95286	86453	121653	118170
19	136324	118158	102808	87573	109089	116686
20	172917	110724	103305	108834	104357	112064
21	149101	103243	95083	91782	117568	138459
22	147865	118388	105499	84093	111299	128821
23	159116	117853	121531	97658	116326	128497
24	159314	110478	93031	88667	118503	117008
25	148838	111482	110474	85488	106582	123692
26	156700	103571	106949	102911	114260	116589
27	145472	117538	92377	105382	112115	124486
28	139634	110524	104850	99638	114662	126795
29	147854	116918	105901	107778	112407	120425
30	149336	111780	87905	91847	108257	144836
[us]	154350,7	110487,9333	100766,3667	95567,66667	111973,9333	124291,6667
[ms]	154,3507	110,4879333	100,7663667	95,56766667	111,9739333	124,2916667
[s]	0,1543507	0,110487933	0,100766367	0,095567667	0,111973933	0,124291667

Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora hasta un cierto punto, si se aumenta el número de hilos a partir de este momento el rendimiento del sistema decae. Si se compara entonces con este punto se obtiene una ganancia de rendimiento de hasta 166,89 ms



Comparando los promedios de cada una de las ejecuciones mencionadas anteriormente con respecto al número de hilos se pueden sacar las siguientes conclusiones, cuando el número de hilos es pequeño se presenta un mejor rendimiento del compilador icc(entre ambos icc tienen un comportamiento similar, al inicio el icc512 presenta mejor rendimiento, mientras que a medida que aumentamos el número de hilos el icc2 presenta mejor rendimiento), pero a medida que aumentamos el número de hilos el compilador gcc otorga mejores rendimientos.

Segunda implementación: (convolución)

- GCC

Se puede observar una ganancia de rendimiento de hasta 2,11 s.

N.º de ejecución	GCC						
	1 hilo	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos	64 hilos
1	2265209	1368316	956281	487274	255658	155825	174929
2	2264000	1178453	880143	491413	253825	146708	176232
3	2267938	1371142	978134	484040	255776	151346	171566
4	2266870	1834158	834290	481797	256017	153437	167598
5	2265699	1353321	836169	481020	258562	153271	153974
6	2266086	1355430	900359	484304	259881	154125	164667
7	2266511	1339773	903647	479947	260638	155890	167510
8	2265415	1395725	937266	484090	258659	152768	165556
9	2269172	1349689	835341	484682	256487	150264	172125
10	2265029	1502575	949631	482712	257906	153145	172141
11	2263703	1857929	902595	485952	255702	162572	162703
12	2265231	1363842	899154	479450	262003	149888	165723
13	2264625	1345521	895213	479941	255611	158294	152574
14	2276544	1343798	835863	481030	260256	152104	159058
15	2263331	1363888	895829	483090	259112	154637	159883
16	2264699	1368938	824107	479828	256759	154667	158026
17	2262948	1261547	820796	480465	261184	155792	161531
18	2288151	1330086	964855	480985	258436	148157	171626
19	2269474	1346214	1126223	480274	257748	147869	167726
20	2265535	1567500	943949	486411	260246	149543	174073
21	2266097	1343339	818882	480080	257089	154471	169617
22	2262687	1526068	834597	483072	258318	154425	170111
23	2264703	1542016	829689	485483	255405	165468	166631
24	2295963	1233913	933629	482358	252234	158954	164296
25	2286063	1336425	940965	476905	256489	156270	149137
26	2266095	1214759	833385	495409	251699	151289	147696
27	2264406	1793235	964708	485271	257208	154791	148916
28	2264410	1344263	836998	481909	256407	153760	163465
29	2264390	1354194	995740	485831	253053	148703	165545
30	2263797	1364373	761890	479864	253889	158960	158708
[us]	2268159,366667	1408347,667	895677,6	483162,9	257075,2333	153913,1	164111,4333
[ms]	2268,159366667	1408,347667	895,6776	483,1629	257,0752333	153,9131	164,1114333
[s]	2,268159366667	1,408347667	0,8956776	0,4831629	0,2570752333	0,1539131	0,1641114333

Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora hasta un cierto punto, si se aumenta el número de hilos a partir de este momento el rendimiento del sistema decae.

- ICC (xCORE_AVX2)

Se puede observar una ganancia de rendimiento de hasta 0.896 s.

N.º de ejecución	ICC(2)						
	1 hilo	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos	64 hilos
1	1003832	614150	311998	139011	95827	105187	167378
2	982492	528188	265445	140018	90577	104537	140734
3	1021829	609795	259392	137636	106590	108739	181939
4	994620	619800	248003	139393	101921	108171	131891
5	967751	596776	264244	138584	95445	96531	156798
6	1015689	469505	250614	140716	92625	108613	152841
7	939935	601767	264127	137287	102615	107162	128426
8	1014720	469464	272390	142821	100346	109172	123867
9	966735	468497	245448	139275	108362	106242	139496
10	976227	593741	286864	140509	90548	109473	145833
11	973916	598098	266603	140966	110282	105178	131317
12	980603	591763	251369	138519	112336	110869	131216
13	1031301	474590	260071	135754	97116	111410	132453
14	989889	592400	253363	139453	116039	109815	140191
15	988976	441793	253865	152962	106781	108362	126955
16	988770	476438	252060	139202	92938	104933	180577
17	997165	641812	255302	130710	90988	106753	144361
18	1023972	508810	253730	153778	108546	110909	167684
19	1032231	462570	265067	138651	96526	105329	130245
20	1033234	601529	248024	136190	91913	109321	188386
21	989950	488803	312449	161488	101230	106215	173254
22	1016484	597805	263583	139152	98248	110219	141087
23	990808	480753	246392	138436	108899	107541	139335
24	980844	471390	255082	140451	95158	110412	131539
25	1013893	470162	265897	155957	105646	106665	129424
26	1009861	480249	279999	138210	103641	110818	135103
27	988797	470272	263127	139558	101876	108304	133889
28	977198	539901	249821	138495	100022	107658	143816
29	1049331	488265	300769	140470	101237	107626	152379
30	991462	489800	247572	140339	116544	105582	140968
[µs]	997750,5	531296,2	263755,6667	141133,0333	101360,7333	107591,5333	145446,0667
[ms]	997,7505	531,2962	263,7556667	141,1330333	101,3607333	107,5915333	145,4460667
[s]	0,9977505	0,5312962	0,263755667	0,141133033	0,101360733	0,107591533	0,145446067

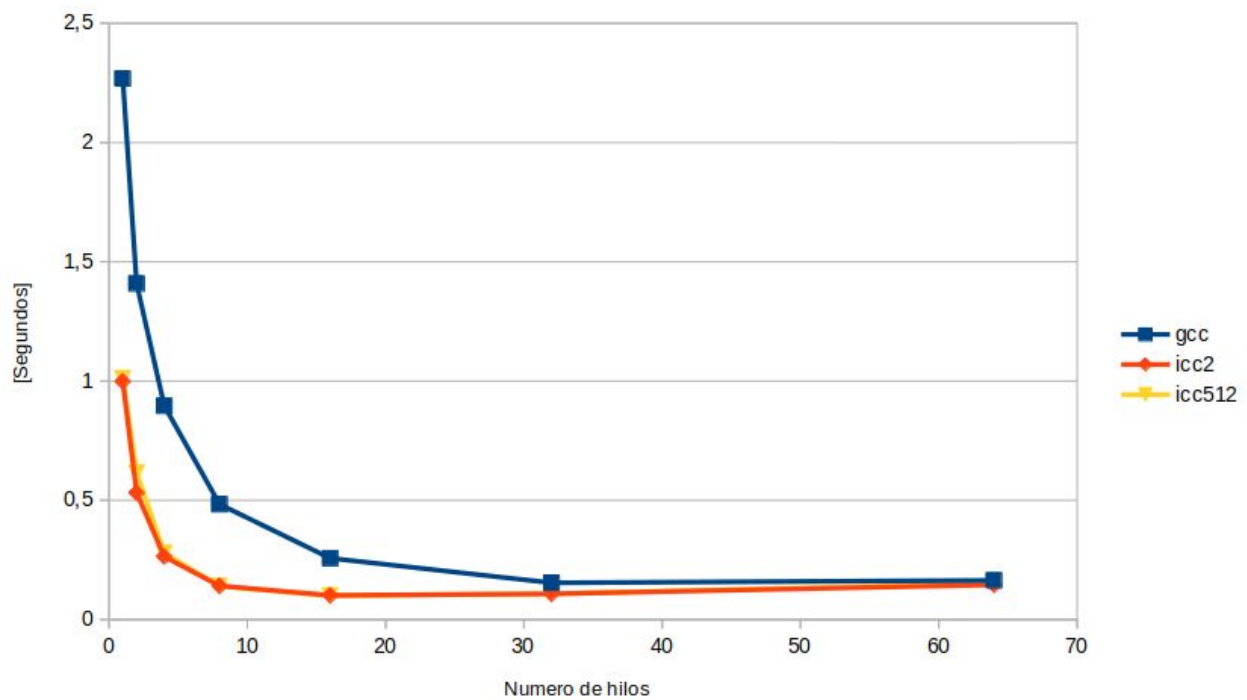
Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora hasta un cierto punto, si se aumenta el número de hilos a partir de este momento el rendimiento del sistema decae.

- ICC (xCORE_AVX512)

Se puede observar una ganancia de rendimiento de hasta 0.911169 s.

	ICC(512)						
N.º de ejecución	1 hilo	2 hilos	4 hilos	8 hilos	16 hilos	32 hilos	64 hilos
1	1008022	608468	255597	135562	107199	107416	137528
2	993516	624264	275100	152176	101373	97744	180986
3	1001644	620695	254254	138267	103817	106085	144282
4	1013084	484860	299226	154706	108940	107695	114594
5	1010091	635204	285306	137309	96954	109866	142495
6	1000763	637811	319531	139861	99409	104911	159890
7	1000272	602781	280364	134179	107679	110284	124103
8	1004237	633463	256459	136849	106671	111843	163268
9	1012672	634276	268918	139128	101436	112958	142750
10	997840	633485	321026	140865	95983	107933	128178
11	1003141	609406	258764	138792	90755	108638	142096
12	1004868	635653	298197	133231	90918	113634	162055
13	1001374	635307	258474	135728	105545	108053	145686
14	999171	607535	292285	137789	92009	105727	116995
15	1010849	606809	256671	138668	108219	110886	205248
16	1021583	633725	284365	134427	92122	105584	126669
17	1017871	636669	274061	140251	92369	111058	137346
18	1008213	644415	256119	136489	107361	111436	138471
19	1043225	629467	309719	151974	102965	106728	182500
20	989810	606557	284261	136176	95537	107955	135315
21	997348	605936	313612	148924	92832	112905	183816
22	1029870	527196	266578	139309	109705	105729	176911
23	1004872	619661	275624	136533	98010	111441	171732
24	1022780	633857	258505	140818	101113	107898	156319
25	1023299	615828	297529	136525	96590	105881	190564
26	1004093	617359	256227	140848	104307	108113	157891
27	1033921	627111	276361	137530	92589	107884	164877
28	998315	614240	260193	137065	97115	105478	129978
29	1010352	599808	261899	138003	90975	108622	139987
30	1051719	637721	298041	136225	93246	109246	138947
[us]	1010627,167	615318,9	278442,2	139473,5667	99458,1	108321,0333	151382,5667
[ms]	1010,627167	615,3189	278,4422	139,4735667	99,4581	108,3210333	151,3825667
[s]	1,010627167	0,6153189	0,2784422	0,139473567	0,0994581	0,108321033	0,151382567

Se observa que a medida que aumenta el número de hilos, el rendimiento del sistema mejora hasta un cierto punto, si se aumenta el número de hilos a partir de este momento el rendimiento del sistema decae.

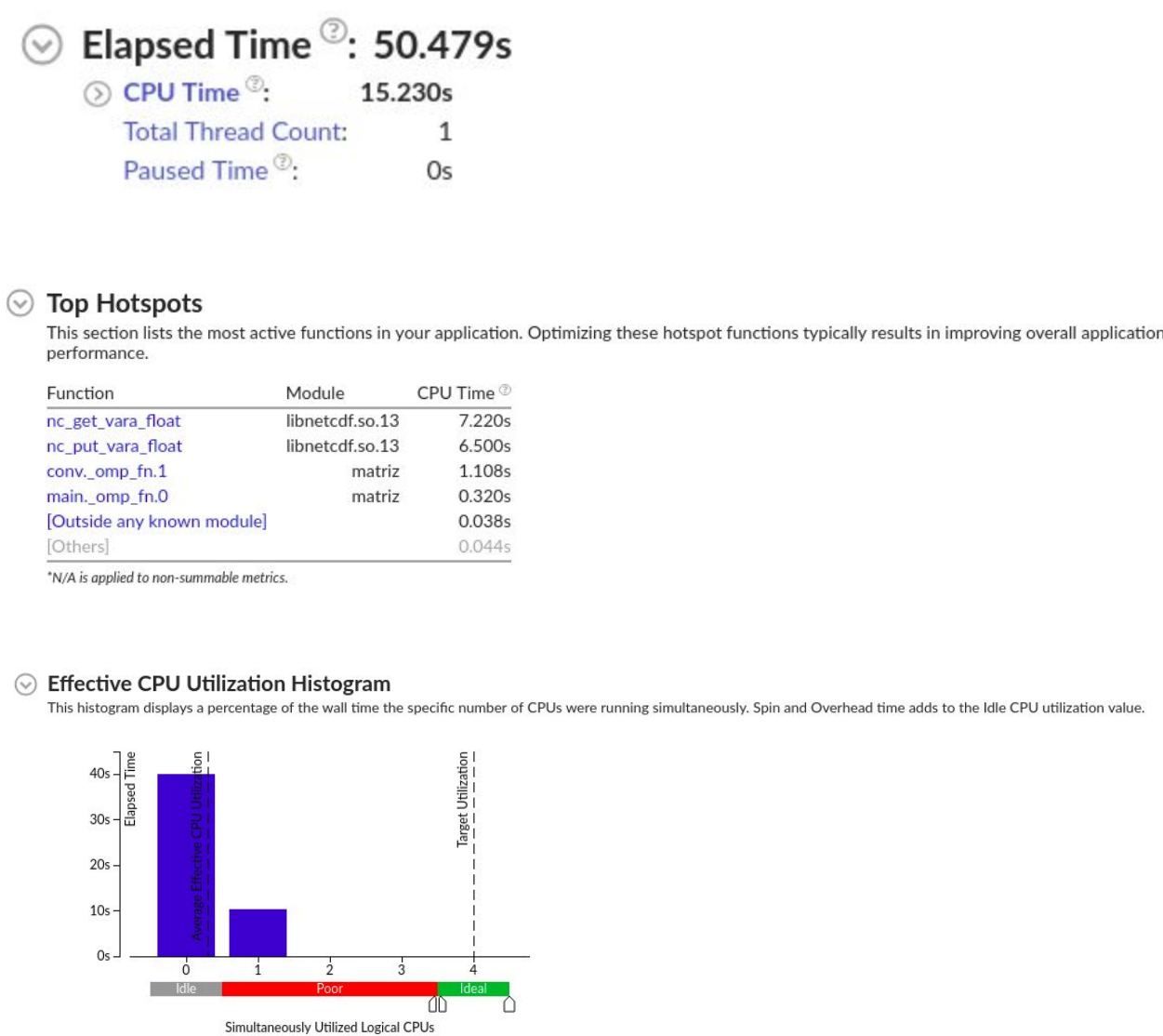


Comparando los promedios de cada una de las ejecuciones mencionadas anteriormente con respecto al número de hilos se pueden sacar las siguientes conclusiones, cuando el número de hilos es pequeño se presenta un mucho mejor rendimiento del compilador icc (entre ambos icc tienen un comportamiento similar, al inicio el icc2 presenta mejor rendimiento, mientras que a medida que aumentamos el número de hilos presentan un rendimiento similar), pero a medida que aumentamos el número de hilos el compilador gcc se acerca al de icc.

Uso de profiling

La herramienta utilizada para hacer el profiling viene en el paquete System Studio de Intel y es el vTune Amplifier.

Primero se muestra la ejecución del programa de manera secuencial con un solo hilo:



Function	CPU Time: Total ▼	CPU Time: Self
_start	100.0%	0s
__libc_start_main	100.0%	0s
main	99.9%	0s
nc_get_vara_float	47.5%	7.220s
nc_put_vara_float	42.7%	6.500s
GOMP_parallel	9.4%	0s
conv	7.3%	0s
conv_omp_fn.1	7.3%	1.108s
main_omp_fn.0	2.1%	0.320s
[Unknown stack frame(s)]	0.3%	0s
[Outside any known module]	0.3%	0.038s
nc_open	0.2%	0.030s
func@0x1ce60	0.1%	0.010s
nc_close	0.0%	0.004s

Se observa que las funciones para leer los datos del archivo binario como la función para generar el archivo binario una vez realizada la convolución son las que más tiempo de CPU consume, mientras que la función de convolución es una de las de menor tiempo y consumo de CPU. El histograma muestra una poca eficiencia ya que no se aprovecha la capacidad multi-core de la PC.

Ahora se ejecuta el programa de forma paralela con 4 hilos y se analiza nuevamente con la herramienta:

⌵

Elapsed Time [?]

56.498s

📄

📄

⌵

CPU Time [?]

18.730s

Total Thread Count:

4

Paused Time [?]

0s

⌵

Top Hotspots

This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

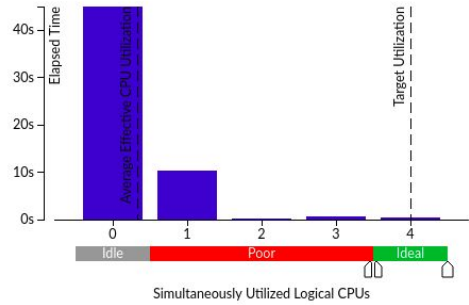
Function	Module	CPU Time [?]
nc_put_vara_float	libnetcdf.so.13	7.370s
nc_get_vara_float	libnetcdf.so.13	7.028s
conv_omp_fn.1	matriz	3.523s
main_omp_fn.0	matriz	0.592s
func@0x18ea0	libgomp.so.1	0.060s
[Others]		0.157s

*N/A is applied to non-summable metrics.

⌵

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



Function	CPU Time: Total ▾ ⓘ	CPU Time: Self ⓘ	Module
__libc_start_main	82.9%	0s	libc.so.6
_start	82.9%	0s	matriz
main	82.8%	0s	matriz
nc_put_vara_float	39.3%	7.370s	libnetcdf.so.13
nc_get_vara_float	37.6%	7.028s	libnetcdf.so.13
conv_omp_fn.1	18.8%	3.523s	matriz
start_thread	17.1%	0s	libpthread.so.0
clone	17.1%	0s	libc.so.6
func@0x16850	17.1%	0s	libgomp.so.1
GOMP_parallel	5.5%	0.010s	libgomp.so.1
conv	4.6%	0s	matriz
main_omp_fn.0	3.2%	0.592s	matriz
func@0x18ea0	0.3%	0.060s	libgomp.so.1
func@0x19030	0.3%	0.048s	libgomp.so.1
[Outside any known module]	0.2%	0.040s	
[Unknown stack frame(s)]	0.2%	0s	
nc_open	0.2%	0.032s	libnetcdf.so.13
func@0x17d80	0.1%	0.027s	libgomp.so.1
nc_close	0.0%	0.000s	libnetcdf.so.13

Se puede observar que al ejecutar el programa con 4 hilos el tiempo del mismo aumento, es decir, su comportamiento fue opuesto al esperado.



Conclusiones

Al finalizar este trabajo práctico se pudo consolidar lo aprendido en las clases teóricas sobre los conceptos de OpenMp y la eficiencia obtenida con el uso correcto del paralelismo. Se pudieron llevar a la práctica estos conceptos logrando cumplir los objetivos planteados en la consigna.

Se obtuvo experiencia en el manejo de la librería OpenMp y de las ventajas y desventajas en el manejo de un número determinado de hilos.

Se puede observar a partir de los gráficos y tablas incorporadas en este informe, que el uso correcto de hilos en el cluster logra una resolución del problema de una manera mucho más eficiente que el procedural.