

Práctica 3 - Computación en la Nube

1. Desarrolla un algoritmo que desarrolle un algoritmo de tratamiento de imágenes/vídeo.

El algoritmo que se ha decidido desarrollar es el **Desenfoque Gaussiano** o también llamado **Alisamiento Gaussiano** es un algoritmo que hace uso de la [función Gaussiana](#) para emborronarla. Es un efecto que se usa ampliamente en software de manipulación de imágenes, usado típicamente para reducir el ruido y el detalle de una imagen.

A continuación se muestra la implementación secuencial del algoritmo:

```
Mat gaussBlur(Mat source, float radius) {
    int columnas = source.cols;
    int filas = source.rows;
    Mat result;
    result.create(filas, columnas, CV_8UC1);
    double rs = ceil(radius * 2.57);

    for (int i = 0; i < filas; i++) {
        for (int j = 0; j < columnas; j++) {
            float val = 0, sum = 0;
            for (int t = i - rs; t < i + rs + 1; t++) {
                for (int s = j - rs; s < j + rs + 1; s++) {
                    int x = min(columnas - 1, max(0, s));
                    int y = min(filas - 1, max(0, t));

                    float dsq = (s - j) * (s - j) + (t - i) * (t - i);
                    float weight = exp(-dsq / (2.0 * radius * radius)) /
(M_PI * 2.0 * radius * radius);

                    val += source.data[y * columnas + x] * weight;
                    sum += weight;
                }
                result.data[i * columnas + j] = round(val / sum);
            }
        }
    }
    return result;
}
```

2. Implementa una versión MPI para este algoritmo.

```
Mat *img;
Mat *result;
vector<uchar> *sub_result;
vector<uchar> *sub_img;
```

```

img = new Mat(imread(argv[1], IMREAD_GRAYSCALE));
if (!img->data)
    exit(-1);

result = new Mat(img->rows, img->cols, CV_8UC1);

int elements_per_proc = img->total() / size;

sub_img = new vector<uchar>(elements_per_proc);
sub_result = new vector<uchar>(elements_per_proc);

startwtime = MPI_Wtime();
MPI_Scatter(img->data, elements_per_proc, MPI_BYTE, sub_img->data(),
elements_per_proc, MPI_BYTE, 0, MPI_COMM_WORLD);

Mat *aux_img = new Mat(*sub_img);
Mat *aux_result = new Mat(*sub_result);

gaussBlur(aux_img, aux_result, atof(argv[2]));

MPI_Barrier(MPI_COMM_WORLD);

MPI_Gather(aux_result->data, elements_per_proc, MPI_BYTE, result->data,
elements_per_proc, MPI_BYTE, 0, MPI_COMM_WORLD);
endwtime = MPI_Wtime();

```

3. Desarrolla una versión OpenMP para este algoritmo.

```

Mat gaussBlur(Mat source, float radius)
{
    int columnas = source.cols;
    int filas = source.rows;
    Mat result;
    result.create(filas, columnas, CV_8UC1);
    double rs = ceil(radius * 2.57);

#pragma omp parallel for shared(source, result, columnas, filas, rs)
    for (int i = 0; i < filas; i++)
    {
        for (int j = 0; j < columnas; j++)
        {
            float val = 0, sum = 0;
            for (int t = i - rs; t < i + rs + 1; t++)
            {
                for (int s = j - rs; s < j + rs + 1; s++)
                {
                    int x = min(columnas - 1, max(0, s));
                    int y = min(filas - 1, max(0, t));

                    float dsq = (s - j) * (s - j) + (t - i) * (t - i);

```

```

        float weight = exp(-dsq / (2.0 * radius * radius)) /
(M_PI * 2.0 * radius * radius);

        val += source.data[y * columnas + x] * weight;
        sum += weight;
    }

    result.data[i * columnas + j] = round(val / sum);
}

}

return result;
}

```

4. Compara las versiones.

Versión Secuencial

```

[ptondreau@ptondreau-1 p3]$ make run
bin/main.run sample3.png 2.5

1  [|||||] 9.9% 5 [|||||] 11.0%
2  [|||||] 100.0% 6 [|||||] 9.9%
3  [|||||] 11.2% 7 [|||||] 11.2%
4  [|||||] 11.2% 8 [|||||] 14.5%
Mem[|||||] 3.05G/7.73G Tasks: 102, 868 thr; 4 running
Swp[|||||] 0K/8.80G Load average: 3.92 2.91 2.10
Uptime: 01:07:35

PID USER PRI NI VIRT RES SHR S CPU MEM TIME+ Command
1433 ptondreau 20 0 320M 383M 179M S 30.4 0.5 0:05.42 bin/main.run sample3.png 2.5
1763 ptondreau 20 0 9323M 383M 179M S 30.4 0.5 0:05.15 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1445 ptondreau 20 0 4061M 581M 208M S 25.0 7.4 7:34.96 /usr/lib/firefox/firefox --sm-client-id 10149113105c
2484 ptondreau 20 0 9323M 383M 179M S 11.8 4.9 4:29.05 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1638 ptondreau 20 0 4061M 581M 208M S 10.5 7.4 2:18.89 /usr/lib/firefox/firefox --sm-client-id 10149113105c
1387 ptondreau 20 0 4025M 157M 106M S 5.5 2.0 1:49.26 /usr/bin/kwin.x11 --session 10149113105cf000158413998
827 root 20 0 1418M 137M 86448 S 5.3 1.7 1:55.80 /usr/lib/Xorg -nolisten tcp -auth /var/run/sddm/2c1
2453 ptondreau 20 0 9323M 383M 179M R 3.9 4.9 1:22.63 /usr/lib/firefox/firefox --contentproc -childID 1 -is
2452 ptondreau -11 0 9323M 383M 179M R 2.6 4.9 1:03.26 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1796 ptondreau 20 0 9323M 383M 179M S 2.6 4.9 0:54.22 /usr/lib/firefox/firefox --contentproc -childID 1 -is
4132 ptondreau 20 0 11432 4940 3416 R 2.6 0.1 0:07.11 htop
1976 ptondreau 20 0 3055M 194M 118M S 2.6 2.5 0:58.76 /usr/lib/firefox/firefox --contentproc -childID 4 -is
1425 ptondreau 9 -11 1806M 1496M 10328 S 2.0 0.2 1:02.89 /usr/bin/pulseaudio --daemonize=no
1788 ptondreau 20 0 9323M 383M 179M S 2.0 4.9 0:47.57 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1573 ptondreau 20 0 4061M 581M 208M S 2.0 7.4 0:51.72 /usr/lib/firefox/firefox --sm-client-id 10149113105c
1720 ptondreau 20 0 4061M 581M 208M S 2.0 7.4 0:39.93 /usr/lib/firefox/firefox --sm-client-id 10149113105c

```

Como se puede observar en la imagen anterior, la ejecución del programa sólo se realiza en un procesador, por lo tanto no hay paralelización.

```

[ptondreau@ptondreau-1 p3]$ make run
bin/main.run sample3.png 2.5
Tiempo de ejecución: 55 sec
Resultado escrito en ./result.png
[ptondreau@ptondreau-1 p3]$

```

El resultado que obtenemos es la imagen seleccionada difuminada y en blanco y negro. Con esta imagen de prueba (de 2816x2112), el programa secuencial dura unos 55 segundos.

Versión MPI

```

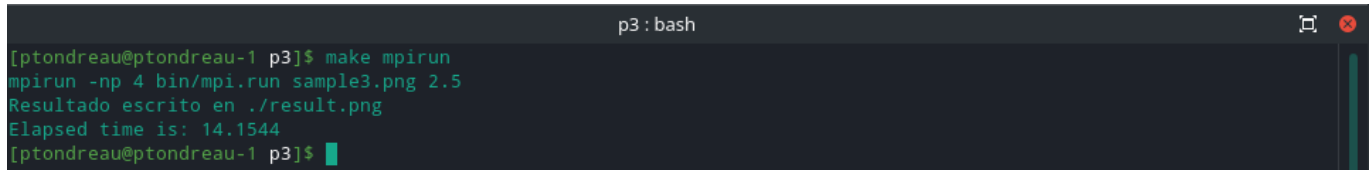
[ptondreau@ptondreau-1 p3]$ make mpirun
mpirun -np 4 bin/mpi.run sample3.png 2.5

1  [|||||] 100.0% 5 [|||||] 16.6%
2  [|||||] 17.0% 6 [|||||] 100.0%
3  [|||||] 10.7% 7 [|||||] 100.0%
4  [|||||] 11.2% 8 [|||||] 100.0%
Mem[|||||] 3.04G/7.73G Tasks: 102, 862 thr; 9 running
Swp[|||||] 0K/8.80G Load average: 1.91 2.12 1.67
Uptime: 01:02:27

PID USER PRI NI VIRT RES SHR S CPU MEM TIME+ Command
4140 ptondreau 20 0 374M 4564 3056 R 100 0.6 0:03:52 bin/mpi.run sample3.png 2.5
4142 ptondreau 20 0 374M 4594 3056 R 100 0.6 0:03:50 bin/mpi.run sample3.png 2.5
4141 ptondreau 20 0 374M 46184 3056 R 100 0.6 0:03:52 bin/mpi.run sample3.png 2.5
4143 ptondreau 20 0 374M 46176 3056 R 99.6 0.6 0:03:48 bin/mpi.run sample3.png 2.5
1763 ptondreau 20 0 9310M 365M 167M S 41.5 4.6 12:39.74 /usr/lib/firefox/firefox --contentproc -childID 1 -is
2484 ptondreau 20 0 9310M 365M 167M S 17.0 4.6 3:31.07 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1445 ptondreau 20 0 4075M 553M 202M S 7.9 7.0 6:17.56 /usr/lib/firefox/firefox --sm-client-id 10149113105c
2453 ptondreau 20 0 9310M 365M 167M S 3.9 4.6 1:04.79 /usr/lib/firefox/firefox --contentproc -childID 1 -is
2452 ptondreau -11 0 9310M 365M 167M S 3.2 4.6 0:49.51 /usr/lib/firefox/firefox --contentproc -childID 1 -is
2455 ptondreau 20 0 9310M 365M 167M S 2.6 4.6 0:44.37 /usr/lib/firefox/firefox --contentproc -childID 1 -is
4132 ptondreau 20 0 11432 4940 3416 R 2.6 0.1 0:00.28 htop
1425 ptondreau 9 -11 1806M 1496M 10328 S 2.0 0.2 0:49.56 /usr/bin/pulseaudio --daemonize=no
1796 ptondreau 20 0 9310M 365M 167M S 2.0 4.6 0:41.48 /usr/lib/firefox/firefox --contentproc -childID 1 -is
1573 ptondreau 20 0 4075M 553M 202M S 2.0 7.0 0:41.63 /usr/lib/firefox/firefox --sm-client-id 10149113105c
827 root 20 0 1408M 129M 86128 S 1.3 1.6 1:30.22 /usr/lib/Xorg -nolisten tcp -auth /var/run/sddm/2c1
1387 ptondreau 20 0 4025M 156M 106M S 1.3 2.0 1:23.22 /usr/bin/kwin.x11 --session 10149113105cf000158413998

```

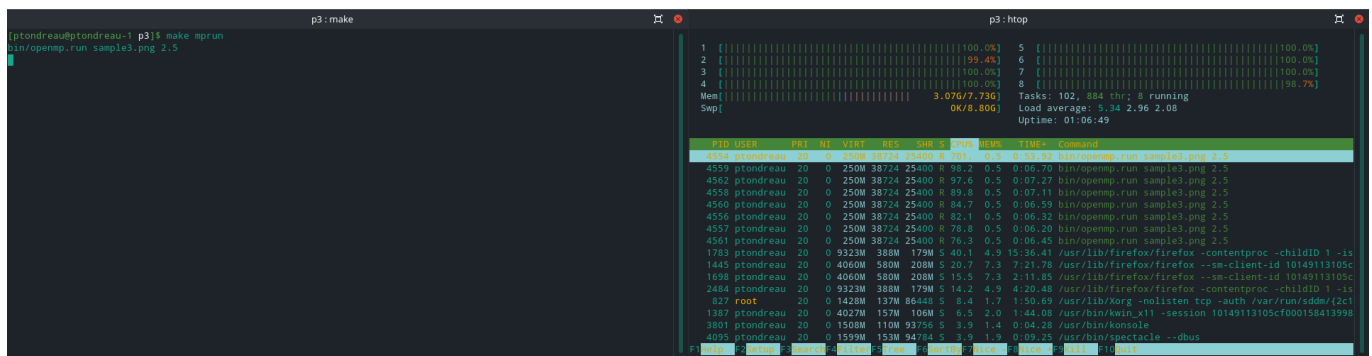
Como se puede observar en la imagen anterior, la ejecución del programa se realiza sobre cuatro procesadores, dado que al ejecutar el programa con `mpirun` se aplica la opción `-np 4`.



```
p3: bash
[ptondreau@ptondreau-1 p3]$ make mpirun
mpirun -np 4 bin/mipi.run sample3.png 2.5
Resultado escrito en ./result.png
Elapsed time is: 14.1544
[ptondreau@ptondreau-1 p3]$
```

El resultado que obtenemos es la imagen seleccionada difuminada y en blanco y negro. Con esta imagen de prueba (de 2816x2112), el programa con paso de mensajes (MPI) dura unos 14 segundos.

Versión OpenMP

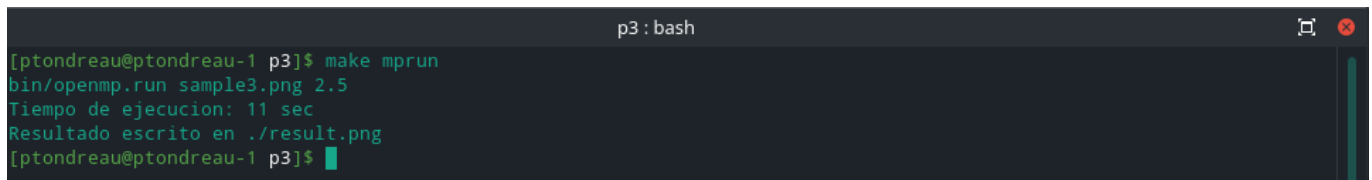


```
p3: make
[ptondreau@ptondreau-1 p3]$ make mprun
bin/openmp.run sample3.png 2.5
Tiempo de ejecucion: 11 sec
Resultado escrito en ./result.png
[ptondreau@ptondreau-1 p3]$
```

```
p3: http
1 [|||||] 100.0%
2 [|||||] 99.4%
3 [|||||] 100.0%
4 [|||||] 100.0%
Mem[|||||] 3.07G/7.73G
Swp[|||||] 0K/8.80G
Tasks: 102, 884 thr: 8 running
Load average: 5.34 2.96 2.08
Uptime: 01:06:49
```

PID	USER	PR	NI	VIRT	RES	SHR	%C	%MEM	TIME+	Command
4554	ptondreau	20	0	250M	38724	25400	R	78.1	0.5	0:53.52 bin/openmp.run sample3.png 2.5
4559	ptondreau	20	0	250M	38724	25400	R	98.2	0.5	0:06.70 bin/openmp.run sample3.png 2.5
4562	ptondreau	20	0	250M	38724	25400	R	97.6	0.5	0:07.27 bin/openmp.run sample3.png 2.5
4558	ptondreau	20	0	250M	38724	25400	R	85.8	0.5	0:07.11 bin/openmp.run sample3.png 2.5
4560	ptondreau	20	0	250M	38724	25400	R	84.7	0.5	0:06.59 bin/openmp.run sample3.png 2.5
4556	ptondreau	20	0	250M	38724	25400	R	82.1	0.5	0:06.32 bin/openmp.run sample3.png 2.5
4557	ptondreau	20	0	250M	38724	25400	R	78.8	0.5	0:06.20 bin/openmp.run sample3.png 2.5
4561	ptondreau	20	0	250M	38724	25400	R	76.3	0.5	0:06.45 bin/openmp.run sample3.png 2.5
1783	ptondreau	20	0	9323M	388M	179M	S	40.1	4.9	15:36.41 /usr/lib/firefox/firefox -contentproc -childID 1 -is
1445	ptondreau	20	0	4066M	580M	208M	S	20.7	7.3	7:21.78 /usr/lib/firefox/firefox --sm-client-id 10149113105c
1696	ptondreau	20	0	4066M	580M	208M	S	15.5	7.3	2:11.85 /usr/lib/firefox/firefox --sm-client-id 10149113105c
2484	ptondreau	20	0	9323M	388M	179M	S	14.2	4.9	4:20.48 /usr/lib/firefox/firefox -contentproc -childID 1 -is
827	root	20	0	1428M	137M	86448	S	8.4	1.7	1:50.69 /usr/lib/xorg -nolisten tcp -auth /var/run/sdssd/(2c1
1387	ptondreau	20	0	4027M	157M	106M	S	6.5	2.0	1:44.08 /usr/bin/kwin_x11 -session 10149113105cf000158413998
3801	ptondreau	20	0	1508M	110M	93758	S	3.9	1.4	0:04.28 /usr/bin/konsole
4895	ptondreau	20	0	1599M	153M	94754	S	3.9	1.3	0:09.25 /usr/bin/spectacle --dbus

Como se puede observar en la imagen anterior, la ejecución del programa sólo se distribuye a cada procesado.



```
p3: bash
[ptondreau@ptondreau-1 p3]$ make mprun
bin/openmp.run sample3.png 2.5
Tiempo de ejecucion: 11 sec
Resultado escrito en ./result.png
[ptondreau@ptondreau-1 p3]$
```

El resultado que obtenemos es la imagen seleccionada difuminada y en blanco y negro. Con esta imagen de prueba (de 2816x2112), el programa con OpenMP dura unos 11 segundos.