Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE4302- Arquitectura de Computadores II
(Curse: CE4302 - Computers Architecture II)

Proyecto #3 - Clúster Beowulf para algoritmo de procesamiento de imágenes
(Project #3 - Beowulf Cluster for an image processing algorithm)

Realizado por:
(Made by:)
Arturo Salas Delgado – 201229804
César Solís Valverde - 201217648
Greivin Fallas Sánchez - 201036901

Profesor:
(Professor)
Jeferson González Gómez

Fecha: Cartago, 13 de Junio, 2017
(Date: Cartago, June 13, 2017)

Tabla de Contenido

Metodología de Diseño

A continuación se hará una explicación de la metodología utilizada en el proyecto, donde se involucra el análisis del problema, la investigación respectiva, algunas propuestas de diseño y la comparación y evaluación de tales propuestas.

Metodología Scrum

La metodología de diseño utilizada implica la división en módulos (compilación, ejecución e interfaz con el usuario), donde el comportamiento colectivo deseado de los sistemas emerge tras una serie de iteraciones de los componentes individuales. En esta, se pretende la reunión de diferentes sistemas (en este caso, módulos) para formar uno general (la aplicación deseada para el proyecto). Así, los módulos son especificados y separados según la conexión que conlleva cada una de las clases, donde cada paquete creado debe ser probado (es en cierta manera funcional) por separado para lograr un resultado deseado, el cual posteriormente será adjuntado al sistema final.

Por otro lado se establece no obtener todas las especificaciones desde un principio sino por medio de pequeños sprints (parecido a la metodología Scrum, véase la Figura 1, para lograr aportes pequeños y funcionales a ser transferidos a todo el sistema; es decir, cada vez se va trabajando de partes más pequeñas a partes más grandes. Así pues, se toma la decisión de usar la metodología mencionada aunque se sabe de sus desventajas como al trabajar en módulos más pequeños se pueden encontrar problemas al momento de unirlos al sistema general. Y también, buscar disminuir la documentación, obteniendo sólo lo necesario para desarrollar cada uno de los pequeños Sprints.

Figura 1. Proceso de metodología Scrum [6]

Algunas ventajas de las metodología Scrum son:

- Busca generar más y mejor software en menos tiempo.
- No genera tanta documentación.

- Usado no solo en software sino en proyectos que requieren flexibilidad y agilidad durante la construcción de los productos.
- Permite la construcción de proyectos de manera iterativa.
- Transparencia con respecto a los objetivos, avances y tiempos de entrega en el proyecto.

Análisis del Problema

En este proyecto se debe utilizar sistemas multiprocesadores interconectados para que trabajen como un solo computador, esta es la definición de un clúster. Los computadores nodos esclavo se interconectan con el nodo maestro a través de red lan. El nodo maestro se conecta con los demás nodos utilizando Secure Shell(SSH) para establecer un puente de comunicación. Sobre este puente se ejecuta una interfaz de pase de mensajes (MPI por sus siglas en inglés) que permite la ejecución concurrente de tareas y su respectiva sincronización entre los diferentes nodos.

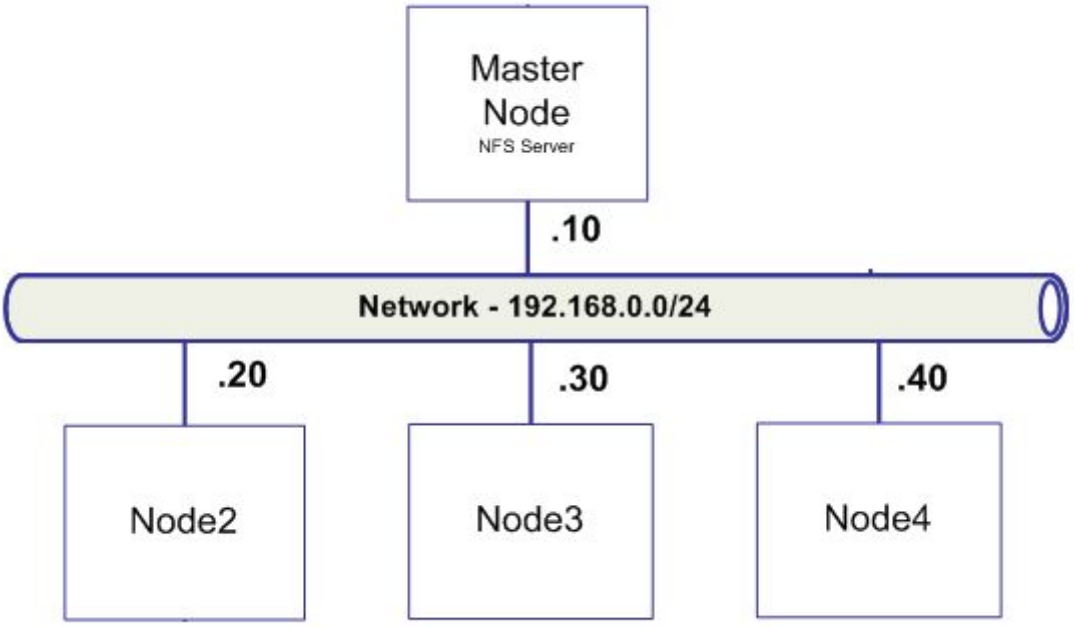

Figura 2. Esquema del clúster

La tarea principal y el objetivo del proyecto es aplicar un filtro a una imagen y mostrar la imagen resultante, pero la tarea se debe distribuir entre todos los nodos del clúster para obtener un menor tiempo de ejecución.

Los tiempos de respuesta van a variar conforme se cambien la cantidad de nodos utilizados por el clúster y esa es la meta, demostrar que la ejecución concurrente a través de MPI

permite mejores tiempos de ejecución que la misma tarea ejecutándose sobre un solo computador.
Para lograr el objetivo se debe de instalar un servidor de Sistemas de Archivos en Red (NFS por sus siglas en inglés) en la máquina maestra que permita compartir un directorio con los nodos esclavos con el fin de que los archivos en el directorio se puedan compartir y acceder entre todos los nodos sin necesidad de tener que copiar manualmente los archivos entre las diferentes computadoras cada vez que se edita el código fuente del proyecto.

Entre las tareas a realizar en el nodo maestro se encuentra generar la imagen a escala de grises de la imagen original, esta imagen será dividida en la cantidad de procesos totales que se van a utilizar, puede correr más de un proceso por nodo. El nodo maestro seguidamente se va a encargar de iniciar cada proceso y de dirá a cada uno de ellos cuál es la parte que le corresponde operar, finalmente cada parte resultante será unida por el nodo maestro para obtener la imagen final que se guardará en el mismo directorio compartido.

Investigación

Cluster Beowulf

Un cluster es una consolidación de software y hardware configurados para resolver tareas computacionales de forma paralela y en red. El más importante de todos los clusters fue uno humilde y legendario, el Cluster Beowulf creado en la NASA en 1994 con hardware de tercera clase, casi de desecho, sin embargo adecuado para crear una útil supercomputadora. Este cluster cambió la historia de la computación científica. [1]

Configuración de un Cluster Beowulf

Para la configuración de los 4 nodos necesarios en el cluster, se debe de realizar los pasos que se mostrarán a continuación, en el computador Master o principal y en los 3 computadores esclavos. Además, se debe de crear un directorio compartido en las 4 computadoras para la ejecución del programa.

En todos los computadores (nodos), se debe de instalar OpenMPI, OpenCV y NFS. Para el caso de OpenMPI se puede instalar ingresando el siguiente comando en una terminal de linux, en este caso, con el sistema operativo Ubuntu 16.04.1 y un versión de kernel 4.4.

```
apt-get install openmpi-bin openmpi-common libopenmpi1 libopenmpi-dev
```

Para la instalación de OpenCV se debe de descargar el comprimido desde https://sourceforge.net/projects/opencvlibrary/ y en todos los nodos se ejecutan los siguientes comandos:

```
sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

cd ~/opencv

mkdir release

cd release

cmake -DCMAKE_BUILD_TYPE=RELEASE -DCMAKE_INSTALL_PREFIX=/usr/local ..

make

sudo make install
```

Para la configuración del computador master se debe de ejecutar los siguientes comandos, considerando que la carpeta de trabajo puede cambiar al gusto, así como el nombre de esclavos y la cantidad de nodos esclavos

```
apt-get install openssh-client

ssh-keygen -t dsa

cp /home/mpiuser/.ssh/id_dsa.pub /home/mpiuser/.ssh/authorized_keys

scp /home/mpiuser/.ssh/id_dsa.pub mpiuser@slave1:.ssh/authorized_keys

chmod 700 /home/mpiuser/.ssh

chmod 600 /home/mpiuser/.ssh/authorized_keys
```

En todos los computadores, se debe crear un archivo en /home/mpiuser/.mpi_hostfile que contendrá la siguiente información. Se debe de tener en cuenta que el nombre de los nodos esclavos y la cantidad de slots puede variar.

```
localhost slots=2

slave1
slave2
slave3
```

En este paso es importante mencionar que los nodos slave1, slave2 y slave3 corresponden a host de la misma red y para que se puedan acceder se deben de tener estos host registrados en el archivo /etc/hosts, el siguiente cuadro es un ejemplo del archivo.

```
127.0.0.1       localhost master
192.168.0.100       master
192.168.0.101         slave1
192.168.0.102         slave2
192.168.0.103         slave3
```

Para la configuración de los computadores esclavos, se debe de ejecutar el siguiente comando.

```
apt-get install openssh-server
```

Para la creación de un directorio compartido, se debe de configurar un servidor remoto NFS. En el nodo maestro se ejecutarán los siguientes comandos:

```
sudo apt-get install nfs-kernel-server nfs-common portmap

mkdir directorio

sudo nano /etc/exports
```

Se debe de agregar la siguiente línea al archivo abierto. Si se tiene más de un nodo, agregar esa linea nueva con la ip correspondiente al nodo:

```
/home/mpiuser/directorio 192.168.0.0/24(rw,no_subtree_check,async,no_root_squash)
```

```
etc/init.d/nfs-kernel-server restart
```

La configuración del servidor NFS en los nodos esclavos se realiza ejecutando los siguientes comandos en cada nodo:

```
sudo apt-get install nfs-common portmap

mkdir directorio
```

La ip que se muestra a continuación es la ip correspondiente al nodo master:

```
sudo mount -t nfs 192.168.0.100:/home/mpiuser/directorio /home/mpiuser/directorio

mount

sudo nano /etc/fstab
```

Se agrega la siguiente línea al archivo abierto:

```
192.168.0.100:/home/mpiuser/directorio /home/mpiuser/directorio nfs      rw,sync,hard,intr
0 0

umount /home/mpiuser/directorio/

mount -a
```

Para la compilación del programa desde el nodo master se debe de ejecutar los siguientes comandos:

```
eval `ssh-agent`

ssh-add ~/.ssh/id_dsa

mpirun -np 2 --hostfile ../.mpi_hostfile ./app
```

Propuestas de Diseño

Algoritmos para Procesamiento de Imágenes

A continuación, se dispone de tres algoritmos para procesamiento de imágenes que podría ejecutar el cluster Beowulf, mediante la división de trabajo por nodos.

1. **Algoritmo de dilatación o máximos**

El algoritmo de dilatación es un algoritmo que se basa en el vecindario de un píxel para determinar su nuevo valor. El tamaño del vecindario impacta el resultado de la imagen final. Para el vecindario de cada píxel se utiliza una matriz alrededor del pixel a editar, se ordena y se selecciona el valor mayor. El resultado de este algoritmo es una imagen más clara que la original con las zonas claras levemente expandidas. Este algoritmo permite eliminar el ruido de sal y pimienta de pixeles negros al ser de valores bajos.[4]

Figura 3. Comparación entre diferentes tamaños de matriz para el filtro de máximos.[4]

2. **Filtro Gaussiano**

El filtro Gaussiano realiza un suavizado de imágenes con ruido utilizando una máscara que posee valores almacenados según una distribución de Gauss. Para este tipo de filtro se puede ajustar tanto el sigma como las dimensiones de la máscara. Estos ajustes

determinan que tanto afecta el valor de los pixeles del vecindario sobre el valor del píxel central. Para determinar en una imagen el valor que se agrega a cada pixel del vecindario se utiliza la siguiente fórmula de distribución de Gauss en 2 dimensiones.[4]

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Figura 4. Valores de una matriz de 5x5 con sigma=1

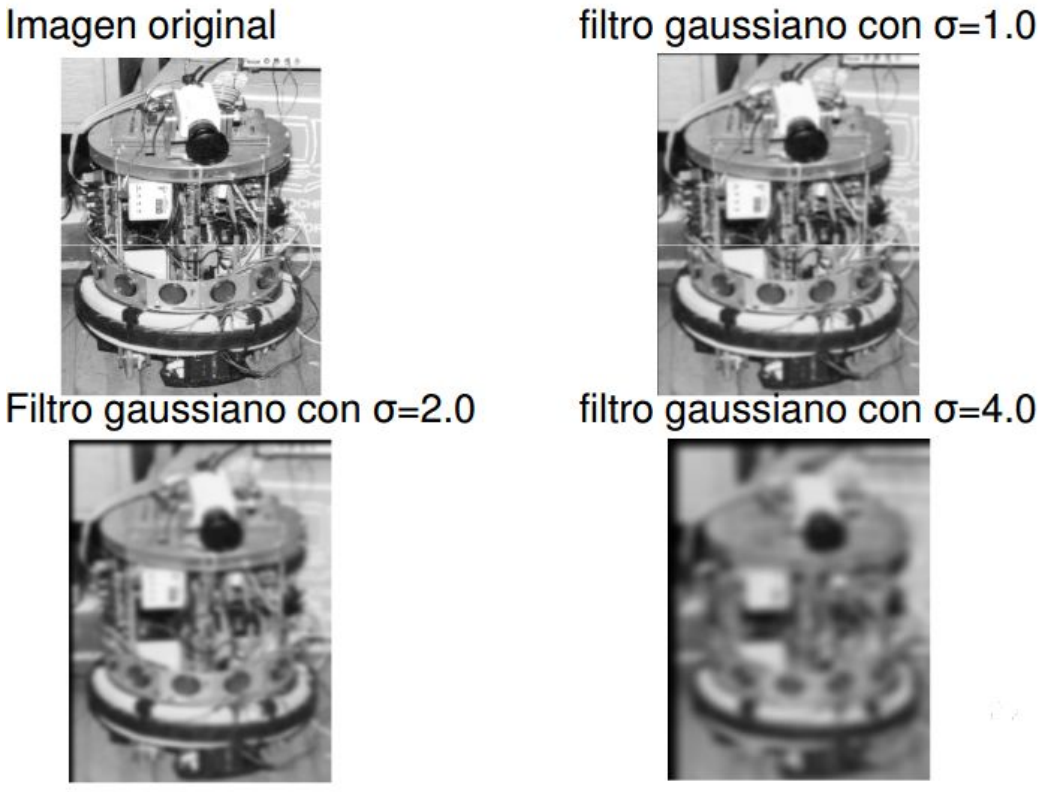

Figura 5. Ejemplos de diferentes sigma[4]

3. Filtro de punto medio de entorno

El filtro de punto medio de entorno es similar al filtro de máximos, sobre un píxel se extrae una ventana de pixeles correspondientes a los pixeles vecinos. El valor a asignar al píxel corresponde a la suma del píxel de mayor valor y al pixel de menor valor entre 2, acorde a la siguiente fórmula.[4]

$$g(x,y) = \frac{f_{\max}(i,j) + f_{\min}(i,j)}{2} \quad (i,j) \in S$$

Este algoritmo permite eliminar el ruido uniforme que se presente en las imágenes pero reduce la nitidez de la misma.

Figura 6. Ejemplo de filtro de imagen original contra filtro de punto medio[5]

Elección de propuesta

La elección se orientó hacia el objetivo del proyecto el cuál es demostrar que un computador ordinario puede formar un clúster y reducir sus tiempos de ejecución mediante la división de tareas. Por esto se ha elegido utilizar el algoritmo de Dilatación debido a que ya se tenía una implementación de dicho algoritmo en C. Todos los filtros permiten eliminar de cierta forma el ruido presente en la imagen pero el filtro Gaussiano da como resultado una imagen más distorsionada que los demás filtros a pesar de usar más cálculos para determinar el píxel resultante de una matriz, en este caso es más sencillo utilizar el algoritmo de dilatación.

Para el filtro de punto medio se tiene que es bueno para ruidos uniformes en una imagen pero este tipo de ruido se puede tratar de otras formas y no necesariamente a través de un

cálculo entre el punto máximo y mínimo de una matriz, además en imágenes oscuras no se puede apreciar los cambios. Para que los cambios tengan una notable presencia en la imagen resultante se decidió utilizar el filtro de dilatación.

Además, debido a la división de trabajo, este filtro se puede trabajar por arreglos en vez de un Kernel, por ejemplo, si se tiene un kernel de 5x5 píxeles, en vez de analizar todo el kernel, se puede analizar primero las filas y luego las columnas en arreglos de 5 píxeles, buscando el valor máximo de ese arreglo y sustituyéndolo en la posición media del arreglo. De esta manera, se obtendrían los mismos resultados a si se trabajase con el kernel, pero de manera más fácil al separar las responsabilidades de trabajo de cada nodo y los núcleos utilizados en cada nodo.

Descripción de Bibliotecas

A continuación, se detallan las bibliotecas más importantes utilizadas para el correcto funcionamiento del cluster Beowulf.

OpenMPI

Es una biblioteca que se ejecuta con programas estándar de C o Fortran, utilizando servicios de sistema operativo comúnmente disponibles para crear procesos paralelos e intercambiar información entre estos procesos [2]. Esta biblioteca está diseñada para permitir a los usuarios crear programas que pueden ejecutarse de manera eficiente en la mayoría de las arquitecturas paralelas y también permite soportar la ejecución de programas distribuidos en hardware heterogéneo con el fin de distribuir tareas entre los centros de trabajo que estén disponibles.

OpenCV

OpenCV (Open Source Computer Vision Library) es liberado bajo una licencia BSD y por lo tanto está libre tanto para uso académico y comercial. Tiene interfaces de C ++, C, Python y Java y es compatible con Windows, Linux, Mac OS, iOS y Android. OpenCV fue diseñado para la eficiencia computacional y con un fuerte enfoque en aplicaciones en tiempo real. Escrito en optimizado C / C ++, la biblioteca puede tomar ventaja de procesamiento multi-núcleo. Se activa con OpenCL, se puede aprovechar la aceleración de hardware de la plataforma de computación heterogénea subyacente. [3]

Descripción de Métodos

A continuación, se detalla todos los métodos creados en software para el buen funcionamiento del cluster Beowulf.

- **int main(int argc, char* argv[])** Este se encarga de inicializar la conexión entre nodos del cluster, así como de la división de trabajo entre todos los computadores enlazados.

- **int AbrirImagen(char *direccion).** Este método se encarga de abrir una imagen que se encuentra en cierta dirección o path. Se denota que recibe un argumento llamado *direccion*; este será la dirección de la imagen que se pretende abrir.

- **void DivideImagen(int procs).** Este método pretende dividir la imagen a procesar entre la cantidad de procesos totales que ejecutan los nodos conectados al cluster. Se denota que recibe un armentos llamado *procs* que corresponde a la cantidad de procesos paralelos que ejecutará el algoritmo sobre la imagen original.

- **void EjecutaDilatacion(int proc).** Este método se encarga de ejecutar el algoritmo de dilatación explicado anteriormente sobre el slice de imagen correspondiente a cada proceso. Se denota que recibe un argumento llamado *proc* que corresponde al número de proceso que ejecutará el algoritmo de dilatación. Dependiendo del número de proceso se abrirá la imagen correspondiente y se escribirá el resultado en una versión nueva de la imagen.

- **void CombinaImagen(int nodos).** Una vez que los slices de imagen de cada proceso han finalizado la ejecución del algoritmo de dilatación de toda la imagen original en escala de grises, se pretende unir todos esos slices. Este método se encarga de eso.

- **double AnchoBanda(int nodos, char *path, double time).** Este método se encarga de calcular el ancho de banda de datos de todo el proceso de ejecución de la imagen mediante el algoritmo de dilatación.Recibe tres argumentos: el primero es *procs*, que corresponde a la cantidad de procesos elegidos para la ejecución del algoritmo sobre la imagen, el segundo es *path* que corresponde a la dirección de la imagen original y el tercero es *time* que corresponde al tiempo de ejecución total del algoritmo. El ancho de banda se calcula como el total de bytes ejecutado por cada proceso el tiempo de ejecución. El resultado tendrá como unidad MB/s (Megabytes por segundo.)

Diagrama UML

El diagrama UML del sistema desarrollado es el que se muestra a continuación:

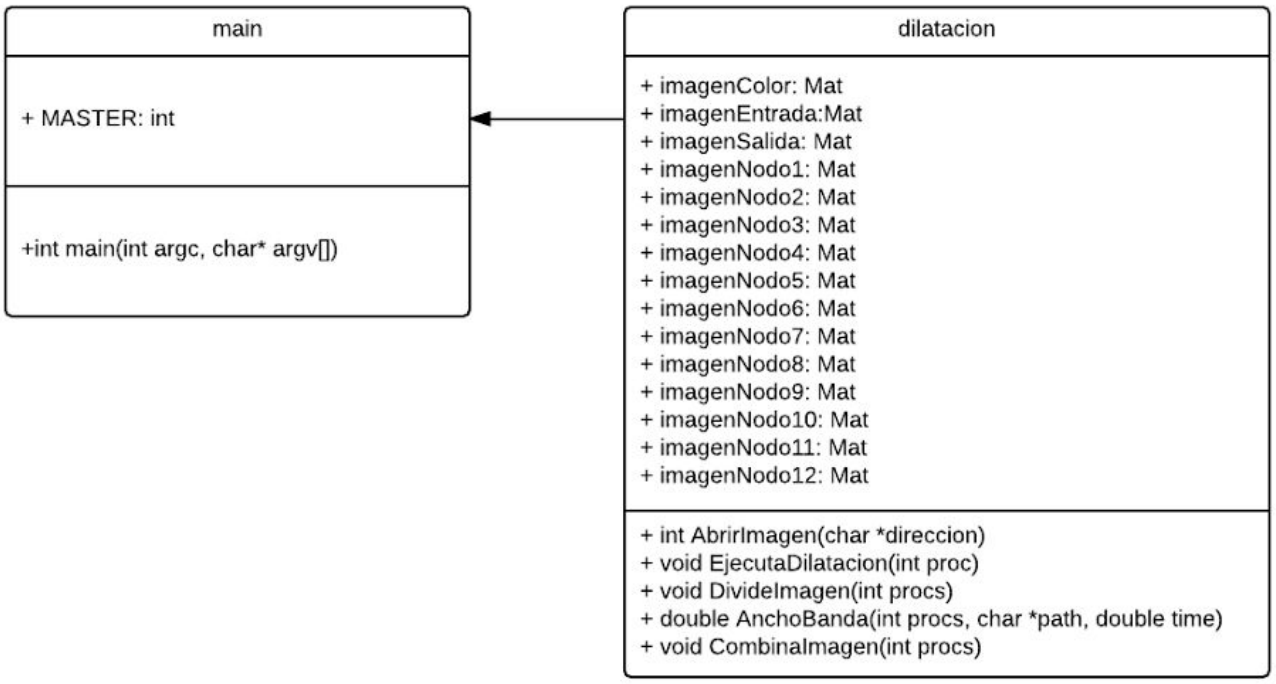

Figura 7. Diagrama UML del sistema

Requisitos de Software del Sistema

A continuación, se muestra una lista con la totalidad de los requisitos para completar el cluster de manera correcta.

Identificador	Requerimiento	Descripción	Estado (0 a 5)
RQ-001	Implementar algoritmo de procesamiento de imagen	Se debe de implementar un algoritmo lineal para el procesamiento de imágenes, en este caso, el algoritmo de dilatación	5
RQ-002	Implementar cluster Beowulf	Se debe de implementar un cluster Beowulf para al menos 4 nodos, donde se realice división de tareas paralelas	5
RQ-003	División de imagen	Se debe dividir una imagen entre la cantidad de procesos distribuidos entre 4 nodos.	5
RQ-004	Resultados de tiempo de ejecución	Se debe de mostrar el tiempo de ejecución una vez finalizado la ejecución del programa	5
RQ-005	Ancho de banda	Se debe de mostrar el ancho de banda utilizado una vez finalizado la ejecución del programa	5

Referencias

[1] A. Lazande. Historia de la Tecnología: Clúster Beowulf, la supercomputadora de los pobres. [Online]. Disponible: https://hipertextual.com/2011/11/historia-de-la-tecnologia-cluster-beowulf-la-supercomputadora-de-los-pobres [Accesado: 05 - Junio - 2017]

[2]} D. Thomasset, M. Grobe, Ïntroduction to the Message Passing Interface (MPI) using C", Condor.cc.ku.edu. [Online]. Disponible: http://condor.cc.ku.edu/grobe/docs/intro-MPIC.shtml. [Accesado: 06 - Junio - 2017].

[3] OpenCV Team.OpenCV. [Online]. Disponible: http://opencv.org/ [Accesado: 06 - Junio - 2017]

[4] Filtros. Grupo de Topología Computacional y Matemática Aplicada. [Online]. Disponible: http://alojamientos.us.es/gtocoma/pid/tema3-1.pdf [Accesado: 07-Junio-2017].

[5] O. Sánchez. Filtros Espaciales. [Online]. Disponible: https://es.slideshare.net/omarspp/imagen-filtrado-espacial [Accesado: 07-Junio-2017]

[6] Lenguajedeprogramacion21.blogspot.com. (2013). "LENGUAJE DE PROGRAMACIÓN C++". [Online]. Disponible: http://lenguajedeprogramacion21.blogspot.com/. [Accesado: 05 - Junio- 2017].