Clúster Beowulf para Algoritmo de Procesamiento de Imágenes

Greivin Fallas Sánchez
Escuela de Ingeniería en Computadores
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: greivinfas15@gmail.com

Arturo Salas Delgado
Escuela de Ingeniería en Computadores
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: artsaldel@gmail.com

César Solís Valverde
Escuela de Ingeniería en Computadores
Tecnológico de Costa Rica
Cartago, Costa Rica
Email: csolisv56@gmail.com

***Resumen*—En el siguiente documento se presentan los principales aspectos relacionados a la elaboración de un Clúster Beowulf, donde se incluye una breve descripción del diseño y solución implementada, además de una serie de resultados obtenidos a partir de algunas pruebas realizadas. Se deben aplicar técnicas de paralelismo y alto rendimiento para la ejecución, dado el esquema de memoria distribuida, en este caso se utiliza OpenMPI. Se desarrolló un programa para el procesamiento de imágenes que aplica un filtro denominado "Dilatación", el cual consiste en calcular e punto máximo entre los píxeles adyacentes de un kernel de 5x5, dicho algoritmo se ejecuta en la cantidad de nodos definida por el usuario, mínimo 1 y máximo 4. Se obtuvieron resultados importantes con respecto a los tiempos de duración de cada sistema de procesamiento, así como la deficiencia en el ancho de banda en transferencia de datos entre los nodos.**

Palabras clave: clúster beowulf, OpenMPI, píxeles, procesamiento de imágenes, dilatación.

1. Introducción

Un cluster puede ser definido cómo un conjunto de computadoras, ya sean servidores o de escritorio, interconectadas entre sí, que se comunican pero que son consideradas como un solo computador. Cada uno de estos computadores tiene su propio sistema operativo y puede mantenerse activo independientemente de los demas. La comunicación se da generalmente a través una red, ya sea LAN, MAN o WAN utilizando un protocolo de red. [1].
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. [1]

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.

1.1. Algoritmo de dilatación o máximos

El algoritmo de dilatación es un algoritmo que se basa en el vecindario de un pixel para determinar su nuevo valor. El tamaño del vecindario impacta el resultado de la imagen final. Para el vecindario de cada pixel 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.

Figura 1. Filtro de Máximos.

2. Sistema Desarrollado

El sistema desarrollado corresponde a un cluster Beowulf de hasta 4 computadores con procesadores de nucleo Intel i3, i5, i7 [12], de manera que el procesador

maestro llega a realizar su porcion de procesamiento con hasta 4 nucleos de forma paralela, y cada nodo esclavo proporciona 2 núcleos extra al sistema, llegando a un total de 10 núcleos de procesamiento. [2]

La manera en la que se distribuye el trabajo en cada computador del sistema corresponde a un calculo de distribución de píxeles, donde se envía una cierta cantidad de filas/columnas a cada nodo y las filas/columnas no paralelizables son computadas por el nodo central. El calculo de la cantidad de información que debe procesar cada nodo está determinado por la cantidad de procesos o Ranks que se definan, es decir, dependiendo la cantidad de nodos configurados y procesos se van a dividir las tareas, la imagen se divide en partes iguales.

Luego de distribuir el trabajo entre los nodos, se realiza un filtrado de filas con respecto al algoritmo espacial de dilatación o máxima, y se devuelve el resultado al nodo maestro. El nodo maestro construye la imagen obtenida, además se utiliza un NFS para compartir los archivos, es una herramienta la cual posee mejor desempeño que otros métodos, como lo podría ser enviar la imagen por medio de matrices.

3. Resultados

A continuación, se presentan algunos resultados de las pruebas experimentales. Los tiempos obtenidos para las diferentes configuraciones se realizan por medio de la biblioteca de C++, time.h, además se realiza una limpieza de la memoria caché antes de la ejecución de cada prueba.

La imagen utilizada para los resultados posee unas dimensiones de 1920x1080 píxeles, es relativamente grande. Además se realizan calculos del ancho de banda, tomando en cuenta el tamaño de la imagen, tiempo de ejecución y cantidad de procesos o nodos utilizadas.

Prueba 1: Cuatro ranks ejecutados por el nodo maestro
El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, los resultados obtenidos se observan en la Figura 2

```
cesar@cesar:~/Resultado$ mpirun -np 4 --hostfile ../.mpi_hostfile ./app
Master: cesar
El proceso  #1 corresponde a cesar
El proceso  #2 corresponde a cesar
El proceso  #3 corresponde a cesar

Tiempo de duracion = 0.219809 seg
Ancho de banda =  4.184358 MB/seg
```

Figura 2. Resultados para 4 procesos.

Prueba 2: Seis ranks ejecutados por el nodo maestro y un esclavo

El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, además los esclavos cuentan con 2, al ejecutar 6 procesos solamente el maestro y un esclavo realizan el balanceo de cargas, los resultados obtenidos se observan en la Figura 3

```
cesar@cesar:~/Resultado$ mpirun -np 6 --hostfile ../.mpi_hostfile ./app
El proceso  #1 corresponde a cesar
El proceso  #2 corresponde a cesar
El proceso  #3 corresponde a cesar
Master: cesar
El proceso  #5 corresponde a cesarn5
El proceso  #4 corresponde a cesarn5

Tiempo de duracion = 0.427977 seg
Ancho de banda =  2.387874 MB/seg
```

Figura 3. Resultados para 6 procesos.

Prueba 3: Ocho ranks ejecutados por el nodo maestro y dos esclavo
El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, además los esclavos cuentan con 2, al ejecutar 8 procesos solamente el maestro y dos esclavos realizan el balanceo de cargas, los resultados obtenidos se observan en la Figura 4

```
cesar@cesar:~/Resultado$ mpirun -np 8 --hostfile ../.mpi_hostfile ./app
El proceso  #3 corresponde a cesar
Master: cesar
El proceso  #1 corresponde a cesar
El proceso  #2 corresponde a cesar
El proceso  #6 corresponde a arturo
El proceso  #7 corresponde a arturo
El proceso  #5 corresponde a cesarn5
El proceso  #4 corresponde a cesarn5

Tiempo de duracion = 0.351296 seg
Ancho de banda =  3.054554 MB/seg
```

Figura 4. Resultados para 8 procesos.

Prueba 4: Diez ranks ejecutados por el nodo maestro y tres esclavo
El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, además los esclavos cuentan con 2, al ejecutar 10 procesos solamente el maestro y tres esclavos realizan el balanceo de cargas, los resultados obtenidos se observan en la Figura 5

```
cesar@cesar:~/Resultado$ mpirun -np 10 --hostfile ../.mpi_hostfile ./app
El proceso  #1 corresponde a cesar
El proceso  #2 corresponde a cesar
El proceso  #3 corresponde a cesar
Master: cesar
El proceso  #6 corresponde a arturo
El proceso  #7 corresponde a arturo
El proceso  #8 corresponde a cesart1
El proceso  #4 corresponde a cesarn5
El proceso  #9 corresponde a cesart1
El proceso  #5 corresponde a cesarn5

Tiempo de duracion = 0.299768 seg
Ancho de banda =  3.681885 MB/seg
```

Figura 5. Resultados para 10 procesos.

Prueba 5: Siete ranks ejecutados por el nodo maestro y dos esclavo
El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, además los esclavos cuentan con 2, al ejecutar 7 procesos solamente el maestro y dos esclavos realizan el balanceo de cargas, para este caso en específico no se utiliza todo el procesamiento de un nodo, ya que solamente

se utiliza un slot del esclavo, los resultados obtenidos se observan en la Figura 5

```
cesar@cesar:~/Resultado$ mpirun -np 7 --hostfile ../.mpi_hostfile ./app
El proceso  #1 corresponde a cesar
El proceso  #2 corresponde a cesar
El proceso  #3 corresponde a cesar
Master: cesar
El proceso  #6 corresponde a arturo
El proceso  #5 corresponde a cesarn5
El proceso  #4 corresponde a cesarn5

Tiempo de duracion = 0.377953 seg
Ancho de banda =  2.781176 MB/seg
```

Figura 6. Resultados para 7 procesos.

Prueba 6: Once ranks

El nodo maestro cuenta con 4 núcleos o slots para la ejecución del filtro, además los esclavos cuentan con 2, al ejecutar 11 procesos se notifica que la cantidad de slots es insuficiente, esto se puede solucionar asignando más procesos a cada nodo. El mensaje se observa en la Figura 7

```
cesar@cesar:~/Resultado$ mpirun -np 11 --hostfile ../.mpi_hostfile ./app
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 11 slots
that were requested by the application:
  ./app

Either request fewer slots for your application, or make more slots available
for use.
--------------------------------------------------------------------------
```

Figura 7. Resultados para 11 procesos.

Para visualizar la información anteriormente citada, en la Figura 8 y Figura 9 se observa el comportamiento de las distintas pruebas de manera gráfica.

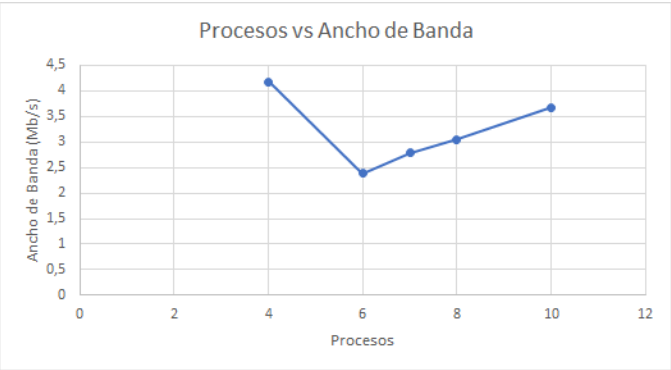

Figura 8. Procesos vs Ancho de Banda.

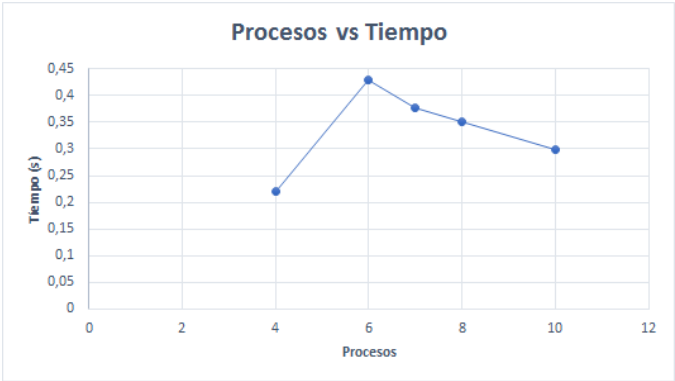

Figura 9. Procesos vs Tiempo.

El ancho de banda del sistema disminuye respecto al aumento en cantidad de computadores que reciben/transmiten informacion en el clúster. Esto debido a que existe una menor cantidad de datos a procesar por cada computador al repartir la carga equitativamente, donde la comunicacion a otros procesadores presenta una latencia significativa respecto al tiempo de procesamiento de datos.

4. Conclusiones

En base a los resultados obtenidos se puede concluir que el uso de clusters para procesamiento de imágenes se encuentra ligado fuertemente al tiempo de comunicación y al tamaño de datos a procesar. A menor tamaño de imagen, el procesamiento secuencial es más rápido que el paralelo, debido a la poca cantidad de datos. La eficiencia disminuye conforme se incrementa la cantidad de procesadores.

Conforme se aumenta el tamano de datos a procesar, aumenta la mejora en velocidad del procesamiento paralelo. A una cantidad alta de procesamiento (muchos píxeles) en filtros con capacidad de procesamiento paralelo.

Referencias

[1] J. Gonzalez. (2017). Introducción A HPC. Tecnológico de Costa Rica, Cartago.

[2] Finnamore, C. (2016). Best Intel Processor: Core i3, i5 and i7 explained. Recurso obtenido de: http://www.trustedreviews.com/opinions/best-intel-processor-core-i3-i5-i7