Clúster Beowulf para Algoritmo de Procesamiento de Imágenes

Fabián Astorga Cerdas
astorgafabian6@gmail.com

Javier Sancho Marín
jfsm1994@gmail.com

Óscar Ulate Alpízar
oscarjosue94@gmail.com

18 de mayo de 2018

Resumen

Este proyecto consiste en la implementación de un grupo Beowulf de tres nodos conectados bajo diferentes configuraciones. El clúster se utiliza para ejecutar un algoritmo de computación de alto rendimiento de procesamiento de imágenes. Finalmente, se presenta la comparación de rendimiento del clúster frente a una sola computadora que ejecuta el algoritmo.

Palabras clave: Clúster, nodos, paralelismo, MPICH, Gaussian Blur

Introducción

Este proyecto implementa el desarrollo de un clúster de tipo Beowulf entre tres nodos individuales conectados por medio de red local, bajo diferentes configuraciones y aprovechando su poder computacional al ejecutar de forma paralela una aplicacion de procesamiento de imagenes de alta demanda computacional. Este cluster trabaja por la tecnica maestro/esclavo, en donde el nodo maestro reparte el trabajo de calculos computacionales a los demas nodos (esclavos) [2]. Para la comunicación entre nodos se utilizó la interfaz de paso de mensajes MPICH. El propósito principal de este proyecto es comparar el desempeño de la aplicación bajo el sistema de clúster contra el desempeño de la aplicación ejecutada en una sola máquina. Este desempeño mide parámetros importantes como: tiempo de ejecución, ancho de banda, entre otros. Otro apartado relevante de este proyecto es el uso de diferen-

tes configuraciones del clúster para probar y encontrar el más adecuado a las necesidades de la descripción del proyecto.

Otro aspecto importante de este proyecto fue utilizar una aplicación de cálculos complejos computacionales. Estas aplicaciones son comúnmente usadas en ambientes de alto rendimiento (HPC). Se realizó una investigación para determinar una aplicación de procesamiento de imágenes que resultara idónea para el cluster Beowulf desarrollado. En secciones posteriores de este artículo se explica con detalle los resultados.

Sistema Desarrollado

El sistema desarrollado consiste en la implementación de un cluster Beowulf utilizando tres nodos.

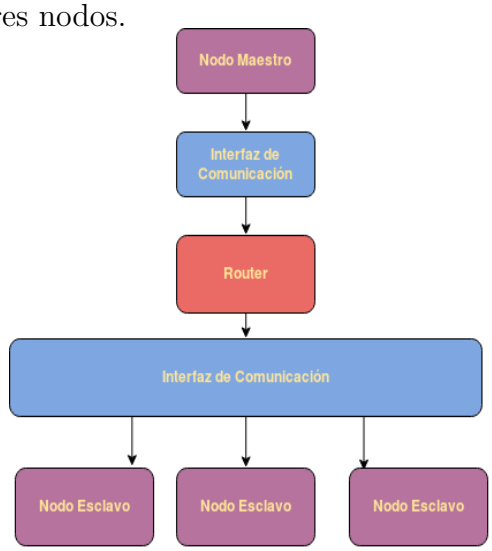

Figura 1. Diagrama del sistema desarrollado

Como se puede visualizar en la figura 1, se tiene dos tipos de nodos, un nodo maestro el cual es el encargado de distribuir el trabajo. Por otra parte se tiene el nodo esclavo los cuales son los encargados de procesar el trabajo asignado por el nodo maestro correspondiente. Asimismo, se cuenta con una interfaz de comunicación la cual es la responsable de permitir la comunicación entre los distintos nodos. Para dicha interfaz se utilizó MPICH, la cual es básicamente una implementación portable y de alto rendimiento del estándar MPI (Message Passing Interface). Como anteriormente se mencionó, sobre dicho clúster se implementó un filtro de procesamiento de imagenes, el cual se detalla en la sección **Gaussian Blur**.

Aplicación

La aplicación realizada para este proyecto es la de un filtro de difuminación de Gauss.

Gaussian Blur

Consiste en la creación de una máscara, llamada kernel, que al convolucionar con cada píxel de la imágen, genera el efecto de difuminado deseado. El kernel, también conocido como matriz de convolución, es calculado por medio de (1).

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

donde x y y hacen referencia a la posición del píxel en la imagen y la letra griega sigma () es la desviación estándar de la distribución de Gauss [1]. Una vez calculado el kernel, se convoluciona el mismo con cada píxel de la imagen. Mientras más grande sea la matriz de convolución, más preciso es el filtro, pero requiere un mayor procesamiento para ser completado.

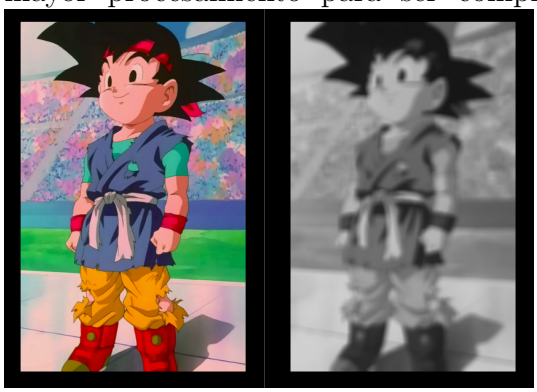

Figura 2. Filtro *Gaussian Blur* con kernel 15x15, sigma de 10 y con efecto

reflect".

El efecto *reflect"* hace referencia a que los bordes de la imagen, utilizan su pixel opuesto (espejo) para realizar el algoritmo. El efecto *çircular"* utilizado en el segundo ejemplo es otra manera de utilizar los píxeles para manejar los bordes.

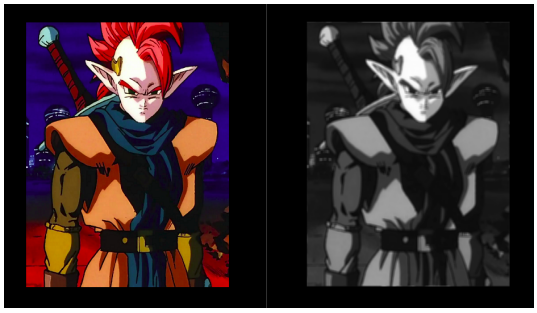

Figura 3. Filtro *Gaussian Blur* con kernel 5x5, sigma de 8 y con efecto "circular".

Clúster Beowulf

Como anteriormente se explicó, el clúster Beowulf está diseñado sobre tres nodos, el cual uno actúa como maestro y los demás como nodos esclavos. Los nodos implementados poseen distintos procesadores Intel de la gama core i5 e i7. Asimismo, como se mostró en la figura 1, se cuenta con un router, el cual permite la localización a nivel de red de los distintos nodos. El clúster Beowulf permite la ejecución del filtro Gaussian Blur de manera paralela entre los distintos nodos como si fuera una única computadora [2]. Para el desarrollo de este cluster Beowulf, se utilizó el sistema operativo Ubuntu Desktop 16.04 LTS, el cual es una distribución de Linux [4]. Para este sistema operativo se agregaron las bibliotecas y APIs (Application Protocol Interface) para la comunicación por paso de mensajes entre nodos utilizado en el proyecto.

Interfaz de Paso de Mensajes

Con respecto al protocolo de paso de mensajes, se utilizó MPICH. Esta es una implementación portable y de alto rendimiento de la interfaz MPI (Message Passing Interface) estándar. Dicha implementación es utilizada en nueve de las diez supercomputadoras con mejor desempeno a nivel mundial. Según sus desarrolladores, los objetivos de MPICH son: proporcionar una implementación de MPI que soporte eficientemente diferentes plataformas de computación y comunicación incluyendo clusters (sistemas de escritorio, sistemas de memoria compartida, arquitecturas multinúcleo), redes de alta velocidad (10 Gigabit Ethernet, InfiniBand, Myrinet , Quadrics) y sistemas de computación de gama alta patentados(como Blue Gene o Cray). Así como permitir la investigación de vanguardia en MPI a través de un marco modular fácil de extender para otras implementaciones derivadas [3].

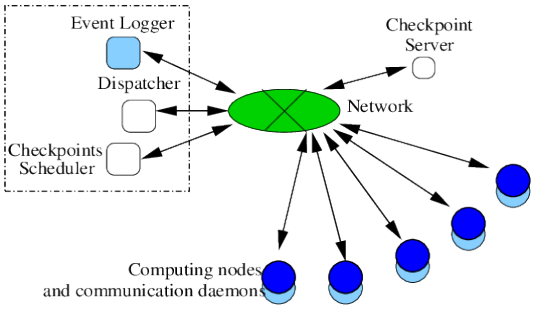

Figura 4. Diagrama de alto nivel de la interfaz de paso de mensajes MPICH

Como se puede observar en la figura 4, se tiene una serie de nodos la cuales se realizan el calculo computacional de la aplicacion del filtrado de la imagen. Asimismo, la interfaz de comunicacion MPICH se basa en la componente central la cual es la red. Esta permite el direccionamiento de mensajes entre el nodo maestro y los demas nodos esclavos. El protocolo MPICH utiliza una serie de procesos, los cuales son los que, básicamente, realizan los cálculos computacionales de cada nodo.

Resultados

Aplicación

En esta sección se detallan los resultados obtenidos a nivel de aplicación realizados por el cluster Beowulf. En la figura 5 se muestra el resultado del filtro realizado por el cluster utilizando tres diferentes nodos, 10 núcleos y un kernel de 21x21.

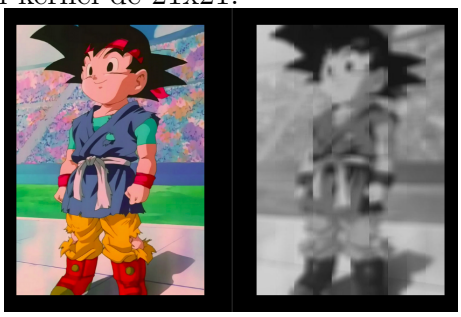

Figura 5. Imagen filtrada configurando el cluster con tres nodos

Se puede afirmar que el sistema esperado es totalmente funcional. Los resultados están acorde a los requirimientos, por lo que se procede a realizar un análisis de rendimiento en la subsección **Rendimiento**.

Rendimiento

Con respecto al rendimiento, se realizaron mediciones sobre dos tipos de variables: tiempo de ejecución y ancho de banda bajo una configuración establecida. Primeramente, se muestran las mediciones del tiempo de ejecución utilizando tres nodos y distintas configuraciones a nivel de núcleos (véase la tabla 1).

Tabla 1. Mediciones de tiempo de ejecución

Nodos	Núcleos	Tiempo ejecución(s)
3	1	0.265652
3	2	0.461224
3	3	0.518404
3	4	0.606437
3	5	0.687703
3	6	0.372756

Como ya se mencionó, dicha tabla mostrada anteriormente ilustra las mediciones del tiempo de ejecución según distintos núcleos utilizados con tres nodos. Como se puede observar, el factor de comunicación a través de una red de alta velocidad y del equipo utilizado en ambientes de computación de alto rendimiento es sumamente importante considerarlo. Por consiguiente, se puede observar la figura 6, la cual representa una gráfica sobre las distintas mediciones realizadas.

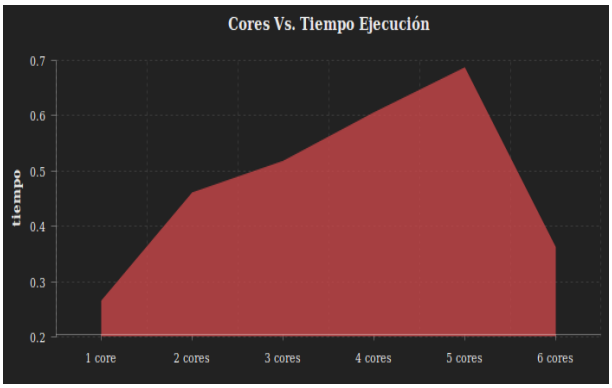

Figura 6. Gráfica de tiempos de ejecución vs. cantidad de núcleos utilizados con una configuración de tres nodos

Se puede apreciar que el tiempo de ejecución varía según la cantidad de núcleos utilizados, como ya anteriormente se mencionó, el factor de la velocidad de la red, así como el equipo utilizado y su capacidad de ancho de banda, afecta de manera notable el rendimiento de la aplicación general. Como se puede observar en la figura 6, tanto el uso de un núcleo, como de 6 núcleos (cada uno de los nodos con dos núcleos) son las que presentan menor tiempo de ejecución. Esto debido a que, para varios cores se distribuye mejor el trabajo y para la ejecución de un solo núcleo no se realiza la comunicación entre nodos, ya que solo se requiere de un solo core para procesar la imagen, por lo que se ahorra en gran parte este componente que afecta el rendimiento en general.

Asimismo, a nivel de ancho de banda utilizado en la aplicación, se realizaron distintas mediciones según varias configuraciones en los nodos. A continuación, se presenta la medición en el ancho de banda utilizando la configuración de tres nodos (véase la figura 7).

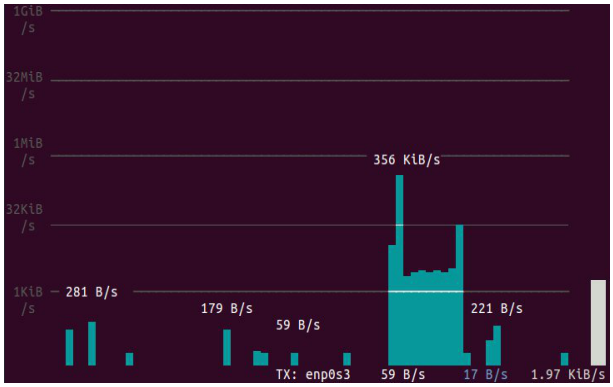

Figura 7. Medición de ancho de banda utilizando la configuración de tres nodos

Como se puede visualizar en la figura 7, para dicha configuración se tienen picos en el ancho de banda. Esto es debido a que el router se encuentra en un estado en el cual el ancho de banda no es sumamente utilizado hasta que se ejecuta la distribución de trabajo entre nodos por medio del cluster. En ese momento el ancho de banda del router empieza a crecer de manera considerable, hasta alcanzar el pico expuesto en dicha figura (356 KiB/s). El ancho de banda máximo dado por el router es un aspecto de gran relevancia a la hora de implementar clusters, debido a que la comunicación a nivel de red puede ser uno de los factores que afectan de mayormente el procesamiento, y por ende rendimiento general del algoritmo computacional.

Conclusiones

En el presente proyecto, se logró obtener el comportamiento general de un un cluster tipo Beowulf, del cual cabe resaltar que, cuando se requiere hacer procesamiento de grandes cantidades de datos, es una opción apta para aprovechar el paralelismo a nivel de datos. Sin embargo, existen limitantes en este tipo de configuraciones (redes de computadores) tales como el ancho de banda dispuesto por los dispositivos de red, cantidad óptima de nodos según el flujo de datos de entrada y el tipo de problema a resolver.

Además, se debe considerar la capacidad de procesamiento que posee cada nodo y los recursos disponibles para ejecutar determinada tarea. Cabe destacar que, el implementar un cluster tipo Beowulf, se hace más sencillo en un entorno UNIX, como Ubuntu Desktop, utilizado en este caso, ya que proveen herramientas útiles como SSH y MPICH que facilitan el diseño y la comunicación entre los nodos existentes.

En el trabajo desarrollado, se obtuvo un comportamiento lineal relativo para un valor óptimo de seis nucleos totales en el cluster, en el cual se vió la reducción del cuello de botella en el ancho de banda del router utilizado. Además, se vio como al utilizar solo un núcleo en un cluster con n nodos, no es necesario utilizar un protocolo de comunicación ya que con solo un nodo es suficiente para completar la ejecución del algoritmo computacional.

Referencias

[1] sonic0002. (2012). Gaussian Blur Algorithm. 17 May 2018, de PixelsTech.net Sitio web: https://www.pixelstech.net/article/1353768112-Gaussian-Blur-Algorithm

[2] Personals.ac.upc.edu. (n.d.). Creacion de un Beowulf. [online] Available at: http://personals.ac.upc.edu/enric/PFC/Beowulf/bec [Accessed 17 May. 2018].

[3] Mpich.org. (2017). MPICH — High-Performance Portable MPI. [online] Available at: https://www.mpich.org/ [Accessed 17 May. 2018].

[4] Ubuntu.com. (2017). Ubuntu PC operating system — Ubuntu. [online] Available at: https://www.ubuntu.com/desktop [Acces- sed 17 May. 2018].