Arquitecturas embebidas de Cómputo para el procesamiento masivamente paralelo con GPU's

Manuel Alfonso Utria Trujillo

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería de Sistemas e Informática Bucaramanga
2017

Arquitecturas embebidas de Cómputo para el procesamiento masivamente paralelo con GPU's

Manuel Alfonso Utria Trujillo

Tesis o trabajo de grado presentada(o) como requisito parcial para optar al título de:
Ingeniero de Sistemas

Director:
Ph.D. Carlos Jaime Barrios Hernández

Co-director:
MsC. Gilberto Javier Díaz Toro

Universidad Industrial de Santander
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingeniería de Sistemas e Informática Bucaramanga
2017

Dedicatoria

Si hay alguien que se merezca ésta dedicatoria eres tú mi vieja querida... Tú que viviste a mi lado toda esta etapa, madrugaste conmigo, también trasnochaste a mi lado, celebraste cuando hubo motivos pero núnca lloraste cuando me viste caído en cambio me diste ánimo, éste triunfo te lo dedico a ti, gran mujer ejemplo para todos de lucha y voluntad.

A mi padre y a mi madre que aún estando lejos de mí siempre me han apoyado y esa confianza depositada se ha mantenido intacta a pesar de las adversidades que han habido en el proceso.

No me puedo olvidar de mi cómplice y compañera de aventuras, para tí también va esta dedicación hermana mía.

A ti, Marta Olga, llegaste a mi vida en el momento indicado, y has sido ese apoyo que me ha ayudado a remar cuando el horizonte ha querido alejarse.

Gracias a mi familia, su granito de arena ha dado fruto y hoy puedo decir que no he caminado solo, conmigo han estado todos ustedes.

Agradecimientos

A mis directores de proyecto por la oportunidad. Al profesor Carlos Jaime Barrios Hernández, gracias por la confianza depositada en mí, la oportunidad de trabajar a su lado es algo que ha enriquecido mi vida académica y personal, también debo agradecer de manera especial al Ingeniero Gilberto Javier Díaz Toro, sin su ayuda y disposición para hacerme caer en cuenta de mis errores, no hubiese sido posible avanzar y mejorar, es algo que me llevo de esta etapa.

Al grupo de soporte del Centro de Super Computación y Cálculo Científico, Luis Alejandro Torres y Sergio Augusto Gélvez, muchachos uds. me sacaron de muchas penurias y me enseñaron el valor del apoyo académico.

Este proyecto se realizó con recursos del Centro de Super Computación y Cálculo Científico de la Universidad Industrial de Santander, muchas gracias por el apoyo.

Resumen

TITULO: ARQUITECTURAS EMBEBIDAS DE CÓMPUTO PARA EL PROCESAMIENTO MASIVAMENTE PARALELO CON GPU's[1]

AUTOR: Manuel Alfonso Utria Trujillo [2]

PALABRAS CLAVES: CLUSTER, EMBEBIDOS, SISTEMAS EMBEBIDOS, COMPUTACIÓN DE ALTO DESEMPEÑO, HPC, SCHEDULER, UNIDADES DE PROCESAMIENTO GRÁFICO GPU.

DESCRIPCION
La eficiencia y el consumo energético son los grandes desafíos que se presentan a futuro para la computación de alto rendimiento. La necesidad de realizar cálculos de manera más rápida ha llevado a diseñar procesadores con mayores prestaciones, donde cada vez se acercan al límite impuesto por la arquitectura del procesador e intentar seguir más allá supone un aumento en costos relacionado con el consumo y mantenimiento que ante un aumento desbordado haría inviable la supercomputación,.

Por tal motivo los centros de supercómputo investigan la forma de proponer nuevas arquitecturas que aprovechen mejor los recursos de hardware, buscando alternativas que provean por menos consumo el mejor rendimiento posible y supongan menos costes de compra y funcionamiento .

Este trabajo propone una alternativa para una necesidad de cómputo específica, donde haciendo uso de dispositivos con características de hardware similares a los utilizados día a día, se adecuan para recibir y procesar aplicaciones que bien pueden ejecutarse en máquinas comunes, pero con la ventaja de consumir una mínima parte y ocupar de igual forma poco espacio. Siendo una alternativa que pudiese ser llevada a diferentes sitios donde tener una infraestructura para el procesamiento sería de difícil acceso

Los resultados de esta investigación proponen lineamientos para que esta solución pueda ser escalada con otros dispositivos o complemento de futuros proyectos, que puedan servir de soporte a diferentes investigaciones

[1]Trabajo de grado modalidad de investigación

[2]Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas. Director: Carlos Jaime Barrios Hernádez, Doctor en Informática. Codirector: Gilberto Javier Díaz Toro, Magister en Informática

Abstract

TITLE: EMBEDDED CUSTOMIZED ARCHITECTURES FOR MASSIVE PARALLEL PROCESSING WITH GPU's[3]

AUTHOR: Manuel Alfonso Utria Trujillo [4]

KEYWORDS: CLUSTER, EMBEDDED, SYSTEMS SOAKED, HIGH PERFORMANCE COMPUTING, HPC, SCHEDULER, GPU GRAPHIC PROCESSING UNITS.

DESCRIPTION
Efficiency and power consumption are the big challenges ahead for high-performance computing. The need to perform calculations more quickly has led to the design of processors with higher performance, where they are approaching the limit imposed by the processor architecture and trying to go further entails an increase in costs related to consumption and maintenance than to a Overflow would make unfeasible supercomputing,.

For this reason supercomputing centers are investigating how to propose new architectures that take better advantage of the hardware resources, looking for alternatives that provide for less consumption the best possible performance and cost less to buy and operate.

This work proposes an alternative for a specific computational need, where using devices with hardware characteristics similar to those used every day, are suitable to receive and process applications that can be executed in common machines, but with the advantage of consuming A small part and occupy equally little space. Being an alternative that could be taken to different places where having an infrastructure for processing would be difficult to access

The results of this research propose guidelines so that this solution can be scaled with other devices or complement of future projects, that can support different investigations

[3]Bachelor's degree research work

[4]Department of Physical-Mechanical Engineering. School of Systems Engineering and Computer Science Advisor: Carlos Jaime Barrios Hernádez, Ph.D. in Computer Science. Co-advisor: Gilberto Javier Díaz Toro, Ms.C. in Computer Science.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Anexos

Introducción

Desde la creación del primer computador hasta el día de hoy se puede observar el avance en la fabricación de hardware, es así como el poder de cómputo en los procesadores de una generación a otra se incrementa exponencial-mente, los sistemas de almacenamiento de igual forma han visto como la capacidad disponible cada vez es mayor en concordancia con la capacidad requerida para almacenar grandes números de datos producidos por los usuarios y las redes han aumentado su capacidad para transmitir de forma más rápida y segura.

Este desarrollo ha traído consigo un aumento desmesurado en el consumo eléctrico, en consecuencia el costo de operación también se ve incrementado para muchas organizaciones, lo que puede resultar inviable si la relación costo vs beneficio está des-balanceada. De igual forma el cambio climático ha hecho que los gobiernos se esfuercen por estrategias que generen conciencia en el uso racional de los recursos, y la computación no es ajena a esta realidad.

Capítulo 1

Contexto de la Investigación

1.1. Descripción del problema

El alto gasto energético del hombre ha llevado a un deterioro de los recursos naturales, y como consecuencia la calidad de vida de las personas se ha visto afectada por éstos fenómenos. Razón por la cual se ha hecho necesario revisar el gasto energético que hacemos a diario, llevando a un replanteo en las políticas de consumo.

Gordon Moore en 1965, expresó que el número de transistores en microprocesador se duplicaría cada año, lo que se conoció como la ley de Moore, aunque años más tarde, en 1975 redefinió su ley y la amplió a dos años, no solo plantea un aumento exponencial en el poder de los procesadores también se habla de una disminución en el costo de estos. En consecuencia la ley de Moore está llegando a un punto de caducidad debido a que las velocidades de los procesadores se acercan al límite y aumentarlas trae consigo un mayor consumo energético que deriva también en mayores costos de refrigeración siendo inviable económicamente.[11]

La supercomputación considera la eficiencia energética una necesidad en el diseño de nuevos procesadores, a futuro con los sistemas Exascale (sistemas de computación capaces de realizar 1018 cálculos por segundo) se busca una reducción de energía de un 15 - 30 %. Desde el Centro de Supercomputación de Barcelona (BSC) se lidera el proyecto Mont-blanc, el cual busca establecer estándares globales de HPC, basándose en soluciones eficientes de energía utilizadas en dispositivos móviles y embebidos el cual plantea reducir el consumo a través del uso de procesadores con arquitectura ARM, los cuales se encuentran presentes en dispositivos de uso diario como celulares, tablets, smart TV, entre otros, los cuales a diferencia de los usados actualmente (Intel Xeon o AMD) presentan un bajo consumo y menor costo.[20, 6]

Este proyecto busca hacer uso de dispositivos embebidos con procesadores de 32 bits ARM y GPU's k20 (arquitectura kepler) fabricados por NVIDIA® acoplados en la Jetson TK1, con los cuales se desea aprovechar sus recursos de cómputo para realizar tareas de procesamiento de alto rendimiento mediante la construcción de un cluster con varios dispositivos, aprovechando el bajo consumo de éstos.

1.2. Objetivo General

- Diseño de lineamientos para el análisis e implementación de arquitecturas de cómputo masivamente paralelas basados en dispositivos de arquitecturas embebidas con GPU's

1.3. Objetivos específicos

- Caracterizar los recursos de las arquitecturas embebidas con GPU'S que soportan procesamiento de alto desempeño para obtener patrones que permitan comparar su rendimiento y viabilidad de la misma.
- Analizar y proponer una arquitectura software en dispositivos embebidos que soporte procesamiento masivamente paralelo que permita obtener una eficiencia computacional acorde al consumo energético
- Proponer una integración de dispositivos embebidos para una arquitectura hardware que soporte procesamiento masivamente paralelo, teniendo en cuenta densidad, eficiencia computacional y energética.
- Implementar un planificador de tareas para la arquitectura HW/SW propuesta que garantice escalabilidad, eficiencia computacional y energética.
- Construir un clúster basado en arquitecturas embebidas que soporte procesamiento masivamente paralelo que sirva de plataforma de prueba en investigaciones y desarrollos futuros,
- Evaluar el rendimiento de la planificación de tareas y la ejecución de aplicaciones sobre la arquitectura propuesta para dar lineamientos que orienten la construcción o modificación de dicha arquitectura pero que tengan en cuenta la eficiencia computacional y energética.

1.4. Alcance del proyecto

Este proyecto de investigación tiene como objetivo principal mostrar las bondades que se pueden encontrar en dispositivos embebidos y la arquitectura RISC (Reduced Instruction Set Computer) usada , la cual puede ser aprovechada para realizar tareas complejas presentes en el campo de la investigación. Aunque trabajos anteriores han demostrado que el uso dado a dispositivos embebidos está más allá de los televisores, móviles, tablets, microcontroladores, entre otros, los laboratorios de HPC ven en ellos una alternativa para reducir el gasto en consumo de energía y mantenimiento.

Una alternativa que se desea plantear en este trabajo es una solución al problema de aprovechamiento de los recursos. Si bien la disponibilidad está limitada por hardware, se busca darle eficiencia al proceso de ejecución de tareas mediante un calendarizador de recursos, con el cual se desea agendar por espacios de tiempo recursos de hardware para la ejecución de tareas atado a la disponibilidad de éstos, así se desea aprovechar al máximo el cluster y mantener una carga de trabajo constante. Lo cual en teoría sería una opción viable frente al consumo de máquinas más potentes pero que al tenerlas todo el tiempo encendidas ejecutando pequeñas tareas o algoritmos que se pueden ejecutar también en éste tipo de arquitecturas y que ayudarían a hacer computación eficiente y a menor costo.

La evaluación de los resultados ayudarán a dar lineamientos para futuros trabajos de grado, si bien ésta base permite el montaje de una arquitectura HW / SW para el manejo de un cluster con un dispositivo específico, permite que el cluster construido pueda ser escalado con más dispositivos de arquitectura similar.

Capítulo 2

Marco Teórico

2.1. Arquitecturas a nivel de hardware para procesamiento en paralelo

El tipo de arquitectura utilizada para la ejecución de aplicativos en paralelo varía dependiendo del soporte hardware para el procesamiento, entre las más utilizadas se encuentra:

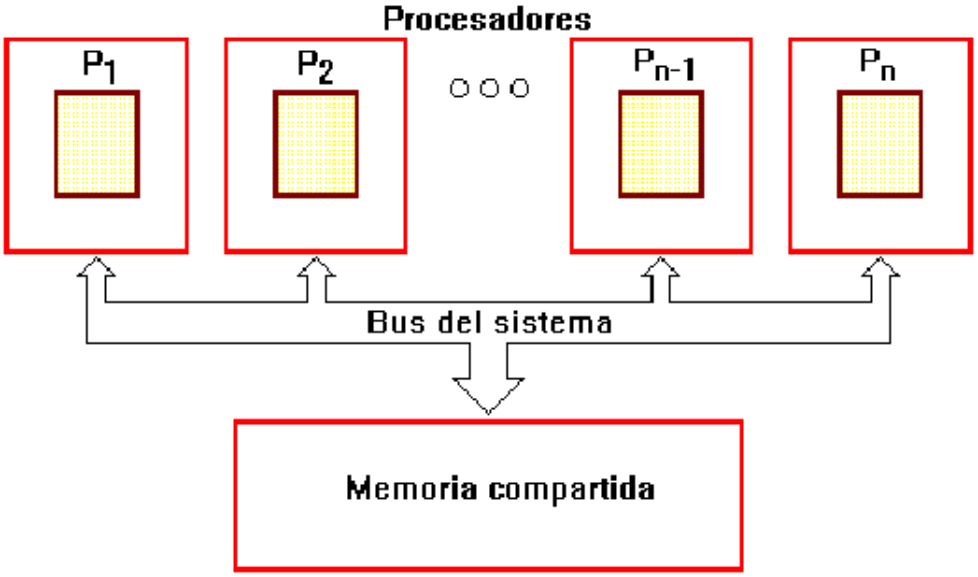
2.1.1. Multiprocesamiento simétrico (symmetric multiprocessing / SMP)

Arquitectura donde dos o más microprocesadores (hasta 32) de similares características son conectados en la misma placa base a la memoria, de manera que es la misma para todos los microprocesadores conectados.

La gestión de los recursos la realiza el kernel del sistema operativo, éste se encarga de la asignación de tareas entre los núcleos de procesamiento y así evitar cuellos de botella consecuencia del ancho de banda de la memoria. Este tipo de arquitectura se conoce como estrechamente acoplado (tightly coupled) o compartiendo todo (share everything).

La desventaja de este tipo de arquitecturas se presenta cuando todos los cores de procesamiento requieren de acceso a memoria, lo que puede congestionar el bus de transmisión de la memoria y disminuir el rendimiento del sistema. [15]

Figura 1: Arquitectura SMP[7]



2.1.2. Procesamiento masivamente paralelo (Massively parallel processing / MPP)

Este tipo de arquitectura cuenta con memoria distribuida entre los procesadores, si por alguna razón otro procesador necesita más capacidad en memoria, se puede utilizar un esquema de paso de mensajes similar al de una red de paquetes.

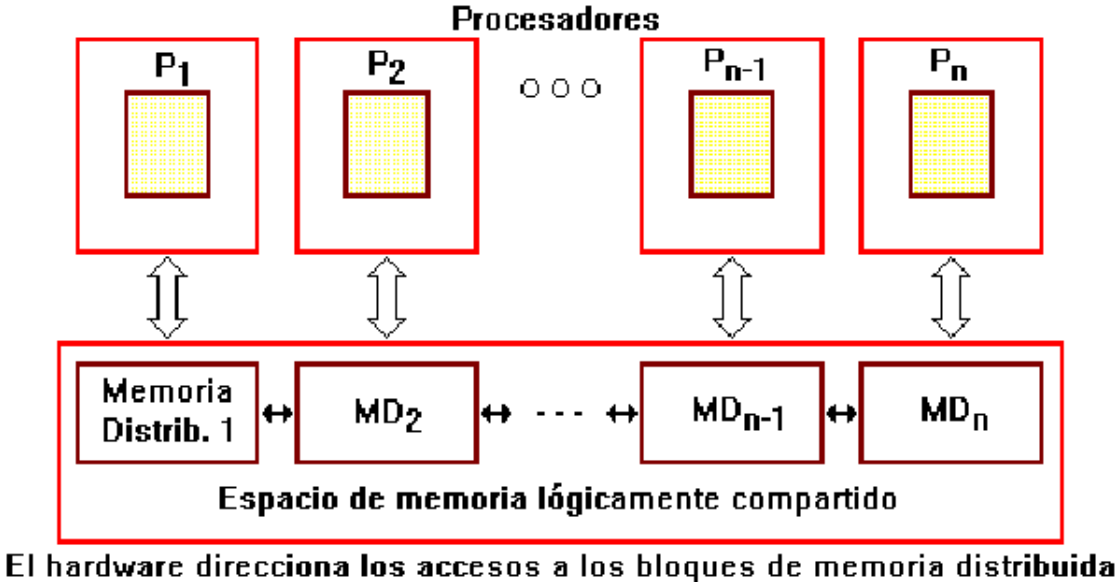
Este diseño se conoce como débilmente acoplado (loosely coupled) y como nada compartido (shared nothing)

La principal ventaja de este tipo de arquitectura radica en la posibilidad de tener conectados cientos de procesadores sin que se produzca un cuello de botella en el tráfico de la información.[9]

2.1.3. Procesamiento paralelo escalable (Scalable parallel processing / SPP)

Cuando se requiere hacer la instalación de más nodos (escalabilidad) al SPP, ésta se realiza en el segundo nivel de memoria, la cual hace su aparición ante los nodos como un espacio de memoria compartida, además en este segundo nivel se añade la posibilidad de operar como un bus, en el que el tráfico se reduce a analizar la coherencia de

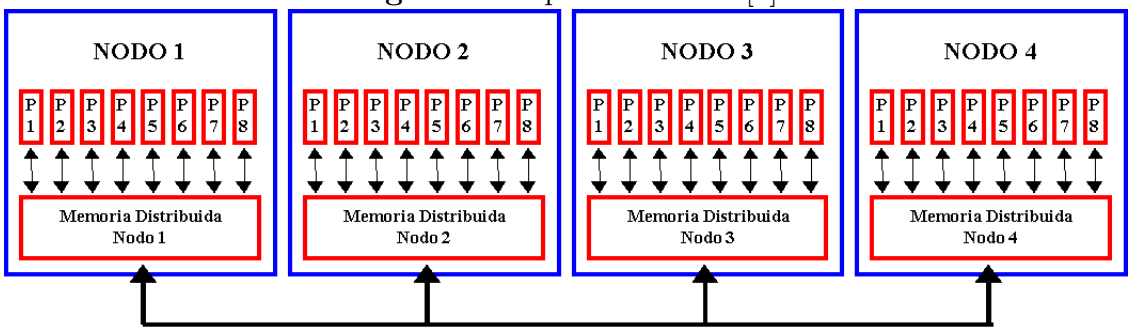
Figura 2: Arquitectura MPP[7]



memoria entre los nodos.

La principal ventaja de este tipo de arquitecturas es la fácil capacidad para programar las aplicaciones aunque el tema de la sincronización entre nodos resulta algo complejo.[9]

Figura 3: Arquitectura SPP[7]



2.2. Cluster informático

Se conoce como cluster, al conjunto de procesadores, de igual o diferente capacidad de cómputo, conectados entre sí por una red de alta velocidad con el fin de funcionar como una única computadora.[3, 19]

Se compone de: Nodos (ordenadores, procesadores, estaciones de trabajo, etc), almacenamiento, sistema operativo, middleware (encargado de la comunicación entre el sistema operativo y las aplicaciones) y ambientes de programación paralelos.

2.2.1. Nodo

Generalmente son ordenadores de escritorios o estaciones de trabajo, lo cuales se encargan de hacer el procesamiento de la información, pueden estar compuestos por uno o más procesadores, memoria ram, interfaces de red, dispositivos de entrada y salida, GPU' s, entre otros. Aunque no es regla que un cluster debe estar compuesto solamente por los equipos anteriormente mencionados, también se puede dar el caso que esté compuesto por microprocesadores, o que las arquitectura de los diferentes ordenadores sean diferentes, lo que se llamaría un cluster heterogéneo.

Las funciones de cada nodo dentro del cluster pueden ser dedicadas (solo está disponible para cumplir los requerimientos del cluster) o no dedicados (el cluster hace uso de los recurso libres para realizar las tareas del cluster). Cuando un nodo es dedicado no posee dispositivo alguno de entrada y salida, solo se dedica a cumplir con las tareas impuestas desde el nodo maestro, quien es el que posee los periféricos de entrada y salida y a su vez administra, controla y monitoriza las tareas del cluster.

2.2.2. Almacenamiento

El almacenamiento en un clúster se puede presentar de las siguientes formas:

- Almacenamiento conectado en red (Network attached storage, NAS)
- Red de área de almacenamiento (Storage Area Network, SAN)
- Almacenamiento interno en el servidor

Almacenamiento conectado en red (NAS)

Comparte la capacidad de almacenamiento de un ordenador (servidor) con otros equipos o servidores que son llamados clientes a través de red (usualmente TCP/IP). Para hacer uso de almacenamiento en red el sistema operativo debe sufrir modificaciones que permitan tener acceso con los protocolos CIFS, NFS, FTP O TFTP.[18]

Es común utilizar un almacenamiento conectado en red cuando se desea tener el almacenamiento de grandes cantidades de datos en forma centralizada para computadoras clientes, algunos usos de las NAS son:

- Copias de seguridad y respaldo
- Nube privadas
- Compartir archivos
- Servidor web
- Servidor de Impresión
- Servidor de VPN
- Virtualización

Red de área de almacenamiento (SAN)

Es una red de almacenamiento integral, concebida para conectar servidores, matrices (arrays) de discos y librerías de soporte. Basados en tecnología de canal de fibra y en iSCI.

Una SAN es una red dedicada al almacenamiento que está conectada a las redes de comunicación de una compañía, el rendimiento está relacionado con el tipo de red que se utiliza y la finalidad es conectar de forma rápida, segura y fiable los distintos elementos que conforman la red, su capacidad se puede extender casi ilimitadamente.

Almacenamiento interno en el servidor

Son dispositivos de almacenamiento conectados directamente a las máquinas, como lo son los discos duros internos, cabinas de almacenamiento (racks, etc), conectadas directamente a un servidor.

2.2.3. Sistema operativo

Es el encargado de gestionar el hardware de la máquina desde sus niveles más bajos, se compone de un conjunto de programas los cuales hacen la gestión de los recursos y proveen al usuario de una interfaz con la cual interactúan con la máquina.

Un sistema operativo debe tener la capacidad de proveer servicios y procesamiento a múltiples usuarios de forma simultánea lo que se conoce como multi-usuario. A su vez también debe ser Multiproceso, que es la capacidad de hacer uso de dos o más procesadores (CPU) para la ejecución de uno o varios procesos.

2.2.4. Conexión de red

Son las que permiten la conexión entre los nodos del cluster, puede ser una conexión de red sencilla como Ethernet, o especializadas con redes de alta velocidad como lo son Fast Ethernet, Gigabit Ethernet, infiniband, entre otras.

Comúnmente las conexiones de red más utilizadas son las Ethernet, debido a que su costo de instalación y operación son menores en relación a las redes de alta velocidad de tipo Infiniband.

2.2.5. Middleware

Es el software encargado de la administración del cluster, actúa entre el sistema operativo y las aplicaciones, permitiendo al usuario hacer uso de todas las máquinas como si fuesen una única de gran poder de cómputo.

El middleware se puede encargar de tareas como la optimización del sistema, migraciones, balanceo de cargas, tolerancia a fallos, entre otras tareas; también se encarga de la escalabilidad del cluster, detectando los nuevos nodos y su comunicación con otros.[17]

Entre los middleware de scheduling más utilizados se encuentra: Slurm, OAR y Torque.

Slurm (Simple Linux Utility for Resource Management)

administrador de recursos libre y Open Source desarrollado por el Lawrence Livermore National Laboratory, SchedMD, Linux NetworX, Hewlett-Packard, and Groupe Bull para diferentes sistemas operativos así como también para múltiples arquitecturas. Utilizado aproximadamente por el 60 % de supercomputadores del TOP500, se destaca por 3 elementos funcionales importantes[16]:

- Permite el acceso a nodos (exclusivos o no) a los usuarios por un tiempo determinado en el cual se hará la ejecución de trabajos
- El usuario tiene a disposición una herramienta que permite iniciar, ejecutar y monitorear trabajos en un conjunto localizado de nodos.
- Administra los trabajos pendientes en una fila, de tal forma que localiza y asigna recursos libres al trabajo que lo requiera.

Torque

Este manejador de recursos basado en el PBS project (Portable Batch System), provee característica similares a Slurm en especial en el algoritmo para la asignación de recursos, debido a que su sistema de colas es FIFO (first in, first out), de igual forma ofrece:

- Tolerancia a fallos
- Interface de administración
- Escalabilidad
- Usabilidad

2.2.6. Ambientes de programación paralelos

Permiten la implementación de algoritmos que hagan uso de recursos compartidos: CPU, GPU, memoria, datos, servicios, etc. Ejemplo: CUDA, OpenMP, OpenMPI, pyCuda, OpenACC.

2.3. Aplicaciones de un Cluster

Su uso no solo está en el ámbito científico, empresarialmente es usual encontrar clusters en sistemas que proveen varios servicios al usuario, al separar cada uno de éstos entre varios nodos: la base de datos alojada en un nodo, el servidor web en otro, los componentes de comunicación entre el servidor y la base de datos en un tercer nodo, es un ejemplo de cluster.

2.4. Tipos de Cluster

Un cluster puede tener nodos con la misma configuración de hardware y software (cluster homogéneo), diferente rendimiento con arquitecturas y sistemas operativos similares (cluster semi-homogéneo) o tener hardware y sistema operativo diferente (cluster heterogéneo). Cualquiera que sea la configuración del cluster siempre buscará garantizar a un bajo costo: alto rendimiento, alta disponibilidad, alta eficiencia y escalabilidad.

2.5. Dispositivo embebido

Es un sistema de computación basado en microprocesadores diseñado para realizar una o varias funciones de forma dedicada en tiempo real, a diferencia de un ordenador

de propósito general que están diseñados para cubrir un amplio rango de necesidades específicas. [6] Los elementos que componen estos dispositivos (cpu, gpu, memoria, almacenamiento, etc) se encuentran incluidos en la misma placa base, debido a esto la modificación de hardware es más compleja que un ordenador común, en el apartado del sistema operativo hay la posibilidad de instalación de múltiples opciones también presentes para ordenadores de mesa (Linux, Windows, MS-DOS) con la diferencia que son diseñados exclusivamente para dispositivos con recursos limitados de hardware.

2.5.1. Arquitectura ARM

ARM es una arquitectura con conjunto de instrucciones RISC (Reduced Instruction ser Computing) ahora con soporte de 64 bits desarrollado por ARM Limited. Aunque su arquitectura es licenciable, por tal razón empresas como: Alcatel, Broadcom, Intel (a través de DEC), LG, NEC, NVIDIA, Qualcomm, Samsung, Sharp, Texas Instruments, entre otas, pueden producir procesadores bajo éste tipo de arquitecturas.[5]

Se destacan por utilizar una menor cantidad de transistores en referencia con los procesadores x86 CISC comunes en los equipos personales, lo que se traduce en un menor costo de producción, baja generación de calor, de ahí que no se necesite otros equipos para la disipación, en consecuencia su consumo de energía es mucho menor.

Su dominio en la electrónica móvil e integrada, lleva a que estén presentes en la mayoría de dispositivos móviles, microprocesadores y micro controladores.

Arquitectura RISC(Reduced Instruction Set Computer)

Es un tipo de arquitectura utilizada comúnmente en microprocesadores y micro-controladores, que se caracteriza por:

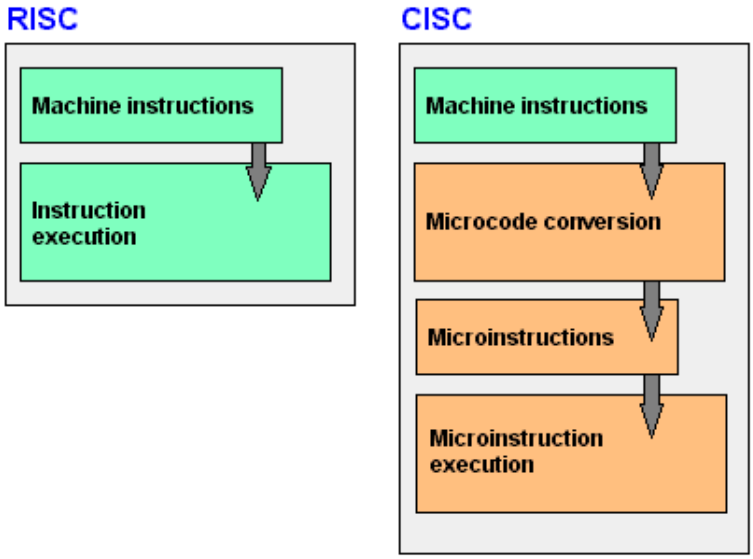
- Instrucciones de tamaño fijo y presentadas en un reducido número de formatos
- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos

Esta arquitectura está orientada hacia conjuntos de instrucciones pequeñas y simples que toman menor tiempo para ejecutarse, en consecuencia un chip RISC dedica menos transistores a la lógica principal, y pueden añadir funcionalidades como:

- Incrementar el tamaño del conjunto de registros.
- Mayor velocidad en la ejecución de instrucciones.
- Implementar medidas para aumentar el paralelismo interno.

- Añadir caches enormes.
- Añadir E/S y relojes para mini-controladores.

Figura 4: Conjunto de instrucciones RISC vs x86 CISC[1]



2.6. Software de monitorización

Este tipo de herramientas están concebidas para automatizar el proceso de monitoreo, independiente de la cantidad de nodos disponibles su trabajo es llevar el control en tiempo real del estado de la plataforma, ayudando que el intervalo de tiempo entre la detección de una falla y la corrección de esta sea menor; beneficiando la agilización de procesos y minimización de errores de tipo humano.

Se puede hablar de 2 tipos de monitorización: A nivel de red y de sistema.

2.6.1. Monitorización de red

Este software hace revisión de los componentes de red y envía alertas ante el fallo de alguno, ya sea que un servidor esté sobrecargado, caído, problemas con conexiones u otro dispositivo. Entre los más usados se encuentran: Nagios, MRTG Y Zabbix,

Nagios

Software de monitorización ampliamente utilizado, originalmente para ser ejecutado en GNU/Linux, pero también puede ser usado en variantes de Unix, licenciado bajo licencia GNU General Public Licence. Se destaca por su amplia personalización, permitiendo la instalación de múltiples plugins con los que se pueden agregar funcionalidades al sistema y tener un mayor control sobre los parámetros o servicios que se desean monitorear, su atractivo está en que no solo puede desarrollar tareas de monitoreo de uso de red también permite configurarse para hacer seguimiento a recursos de máquina[13]

MRTG: (Multi Router Traffic Grapher)

Es un software utilizado para la monitorización de dispositivos de red como switch, router, ... el cual basándose en el tráfico de red en estos dispositivos realiza gráficas estadísticas que muestran la información que se transmite por cada interface. Funciona en sistemas operativos GNU/Linux, Windows, AIX, bajo licencia GNU General Public Licence[14]

Zabbix

Tiene muchas similitud en las funciones que provee Nagios, debido a que puede realizar monitoreo de servicios de red, servidores y diferente hardware de red, y emitir alerta ante cualquier anomalía presentada. Licenciado bajo GNU General Public License version 2, operado en múltiple plataformas

2.6.2. Monitorización de sistema

Son herramientas que trabajan a nivel de máquina, concebidas principalmente para dar al usuario información acerca del uso de CPU, memoria, interfaces de red, GPU (si la hay), entre otros. Importante porque brinda información en tiempo real y permite conocer el estado del hardware, bastante útil cuando se hacen pruebas de software donde se busca tener una medición en el uso de recursos por parte del aplicativo, entre los cuales se pueden encontrar: Ganglia, Cacti, CollectD, entre otros.

Ganglia

Es un software de monitoreo open source (bajo licencia BSD), escalable y distribuido diseñado especialmente para sistemas de computación de alto desempeño como clusters y grids; gracias a su implementación robusta, ha sido probado en diferentes sistemas operativos y arquitecturas de procesadores, permitiendo ser escalado a sistemas de hasta

2000 nodos, donde los datos de mediciones son recogidos por cada nodo y enviado a un nodo de control que se encarga de mostrarlos.[8]

Capítulo 3

Lineamientos de un Cluster basado en dispositivos embebidos

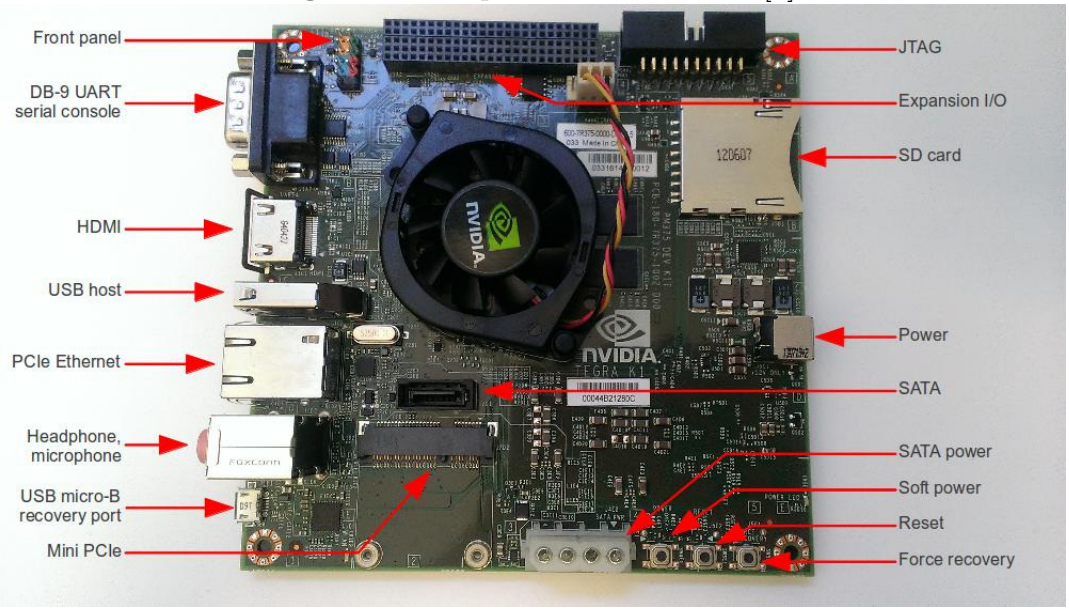
3.1. Cluster a utilizar

Para este proyecto se construirá un cluster Beowulf homogéneo basado en dispositivos embebidos fabricados por NVIDIA, de la familia Tegra K1, su elección se da con el fin de aprovechar la capacidad de cómputo brindada por sus GPU aceleradoras.

Vale aclarar que este cluster es Beowulf, porque si bien las Jetson no están diseñadas para supercomputación, si pueden clumpir tareas específicas como apoyo a procesos de investigación en el área de ciencias de la computación, y es homogéneo porque todos los nodos del cluster tendrán las mismas características a nivel de hardware.

Aunque el mercado ofrece distintas alternativas en cuanto a hardware (raspberry pi, cubieboard, etc) se busca aprovechar la capacidad de cómputo ofrecida por tarjetas aceleradoras fabricadas por NVIDIA para este tipo de dispositivos. Estas GPU de arquitectura Kepler se basan en la plataforma de cálculo paralelo NVIDIA CUDA y su modelo de programación para realizar tareas de alta computación, cálculo científico y aprendizaje automático con gran eficiencia energética y ofrecer una aceleración muy superior a la de los sistemas basados exclusivamente en la CPU para una gran variedad de aplicaciones científicas y comerciales.[10]

Figura 5: Componentes Jetson Tk1[2]



3.2. Arquitectura del sistema y diseño

3.2.1. Componentes y arquitectura del nodo

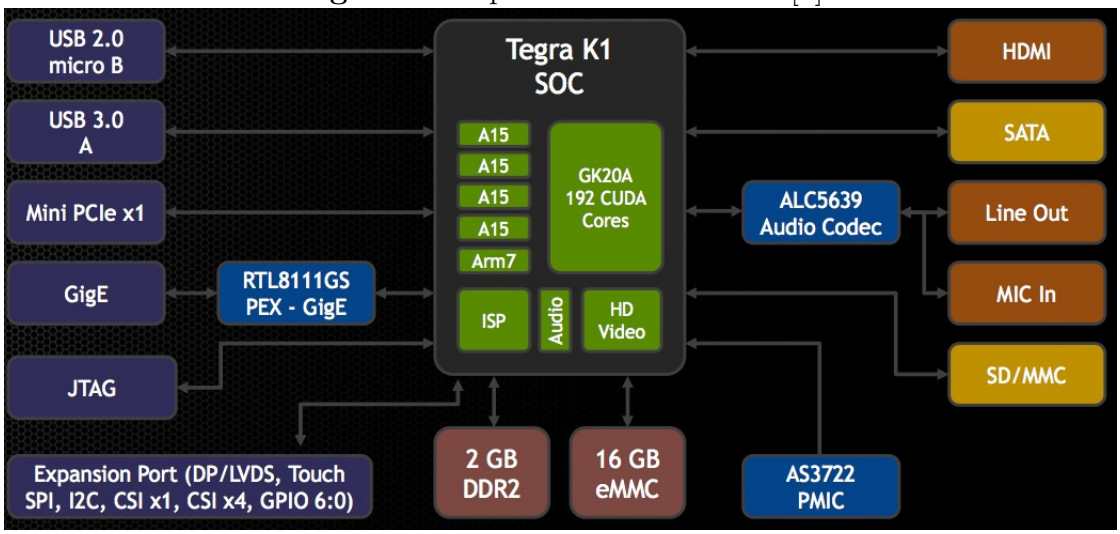
Cada nodo se compone de:

- Dimensión placa: 127mm x 127mm
- Tegra k1 SOC (CPU + GPU + ISP en un simple chip)

  CPU: ARM quad-core Cortex-A15 a 2.32Ghz

  GPU: NVIDIA Kepler GK20a GPU with 192 CUDA cores (upto 326 GFLOPS)
- DRAM: 2GB DDR3L 933 Mhz using 64 bit data width
- Storage: 16 GB fast eMMC (Embedded Multi-Media Card)
- Ranura SD, USB 3.0, USB 2.0, HDMI, puerto serial RS232, puerto SATA con soporte para discos de 2.5" y 3.5"

Figura 6: Arquitectura Jetson TK1[4]



3.2.2. Arquitectura y componentes del cluster embebido "coconuco"

El cluster coconuco se compone de 4 nodos de cómputo NVIDIA Jetson Tk1 conectados en una red ethernet de área local, gobernado desde uno de los nodos de cómputo utilizado para la ejecución de aplicaciones, aunque puede no ser muy recomendable utilizar el nodo maestro para realizar cómputo, se realiza esta asignación para evitar problemas en la compilación y ejecución de aplicaciones.

Se utiliza el móvil para comparar el tamaño entre los dispositivos, mostrando que el cluster no necesita mayor espacio para su montaje.

Hardware de red

El Cluster hace uso de un router Dlink DES-1024R de 24 puertos, el cual puede operar a velocidades de 10 Mbps o 100 Mbps, dependiendo de la capacidad de cada nodo. En este caso particular solo se utilizan 4 puertos de todos los 24 disponibles por el switch.

El armado y arreglo del cableado se encuentra en el mismo stand de los nodos de procesamiento, a diferencia de los clusteres de alto desempeño profesionales que utilizan fibra óptica o red de alta velocidad infiniband, para este proyecto se utiliza cable Ethernet, si bien no es la mejor opción tiene una buena relación costo vs beneficio, que en últimas es lo que se busca en este proyecto.

Figura 7: Arquitectura Cluster Jetson TK1

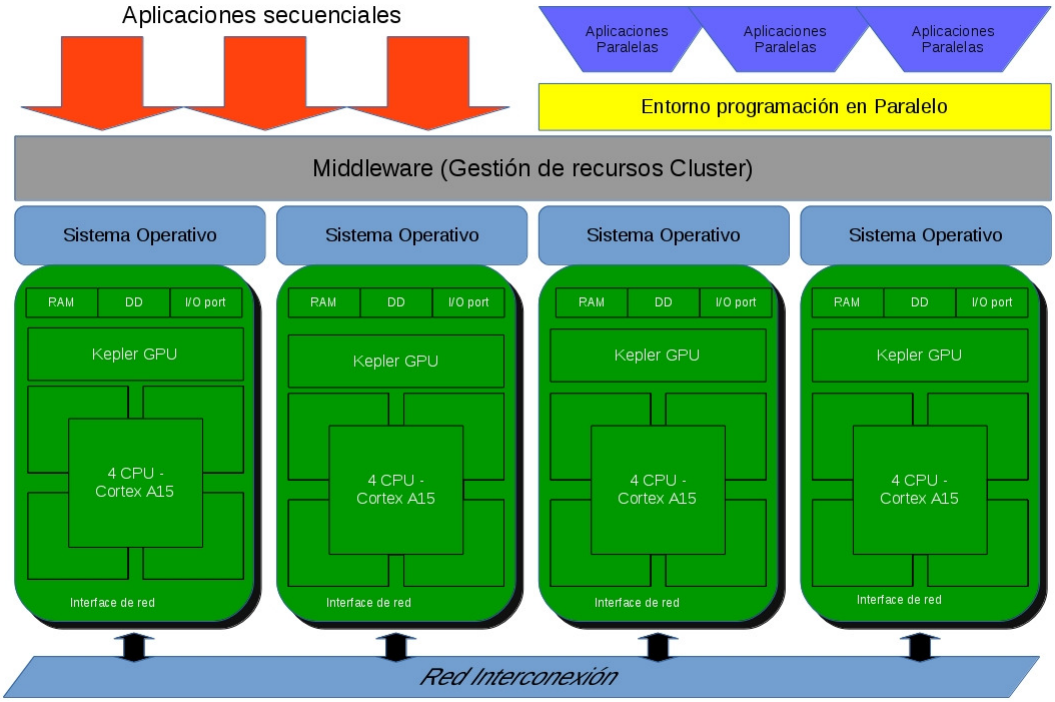


Figura 8: Montaje Cluster vista superior

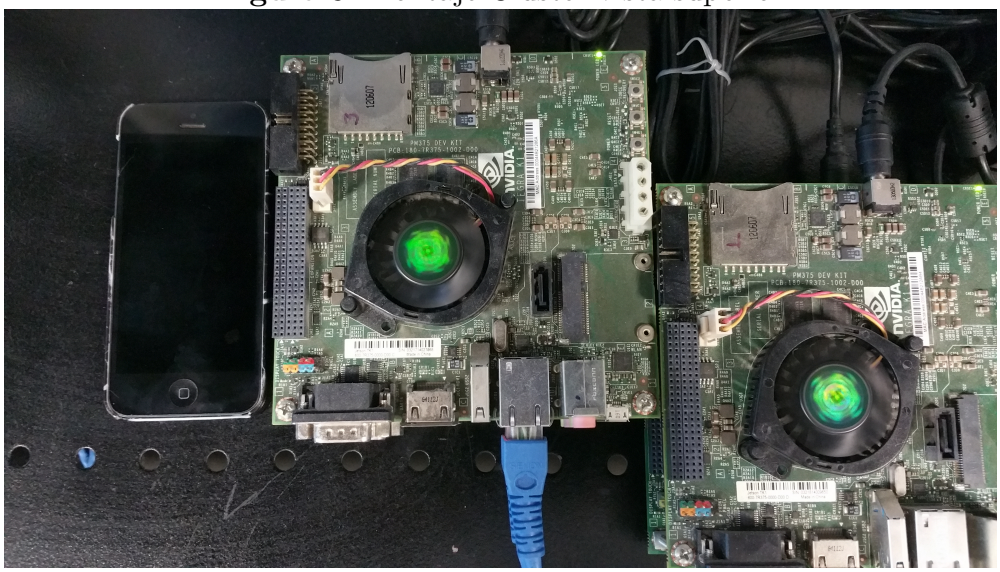
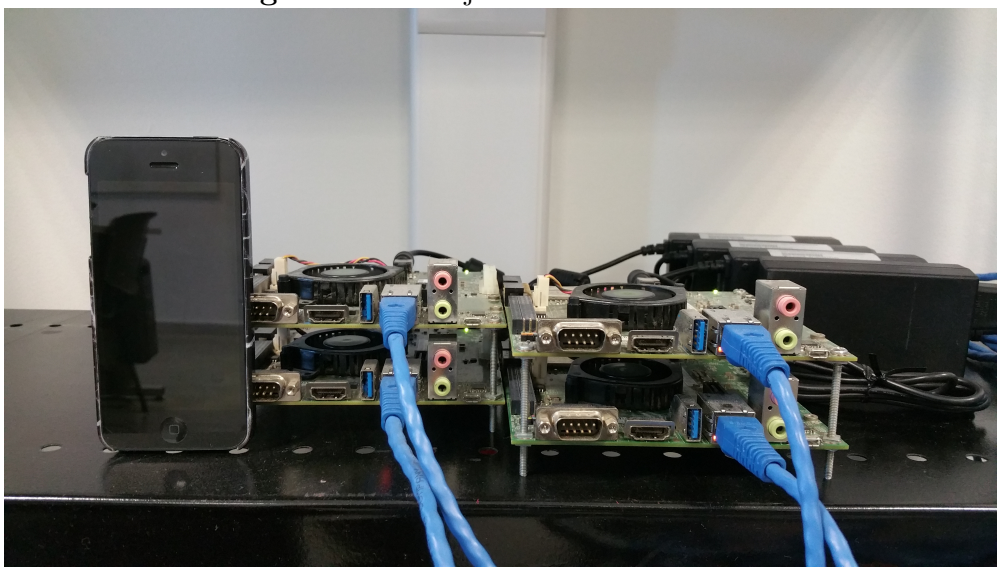


Figura 9: Montaje Cluster vista frontal



Server rack

En este caso debido a los recursos del grupo de investigación se realiza el montaje en uno de los rack dispuestos en la sala para otro proyecto, una ventaja del cluster es el poco espacio necesario para la ubicación de los dispositivos, cada uno posee medidas aproximadas de 12,5 x 12,5 [cm], permitiendo de algún modo portabilidad de los componentes. No así cada uno de los dispositivos cuenta con un adaptador de corriente, el cuál realiza las veces de regulador de voltaje, protegiendo cada dispositivo ante fallas eléctricas, pero penalizando el espacio de montaje.

Si se busca portabilidad en los componentes, lo ideal es utilizar fuentes de poder real para la alimentación de los dispositivos, la cual permitirá prescindir de los 4 adaptadores (1 por placa) y ahorrar espacio para el montaje, dejando la posibilidad de plantear un montaje que permita al cluster ser portado de un lugar a otro con la totalidad de sus componentes.

3.3. Arquitectura software del cluster

La elección del software se realiza pensando en el enfoque que se le desea dar a la máquina, en este caso se busca demostrar que con dispositivos embebidos se es posible construir un cluster y que el software utilizado en máquinas mucho más potentes, también se puede instalar y ejecutar en arreglos pequeños con recursos limitados.

Algunos paquetes y librerías se instalan desde las fuentes de la distribución, otras son descargadas desde los repositorios y compiladas con el fin de reducir el tamaño, y optimizar el rendimiento de acuerdo a la arquitectura de la máquina.

3.3.1. Sistema Operativo Cluster

En este proyecto se hace uso de una distribución Linux basada en Ubuntu optimizada por NVIDIA para éstos dispositivos llamada L4T OS.

Aunque la mayoría de distribuciones Linux (Fedora, OpenSuse, Arch Linux, entre otras) cuentan con soporte para arquitecturas ARM, para los nodos de cómputo la decisión de utilizar la instalada por el fabricante se tomó debido a la disposición del kernel optimizado para éste tipo específico de dispositivo, otro aspecto a tener en cuenta es la compatibilidad con los controladores de las GPU, si bien NVIDIA ofrece los binarios para descargarlos y compilarlos, para comodidad del usuario la versión instalada por ellos ya trae toda la paquetería compilada e instalada.

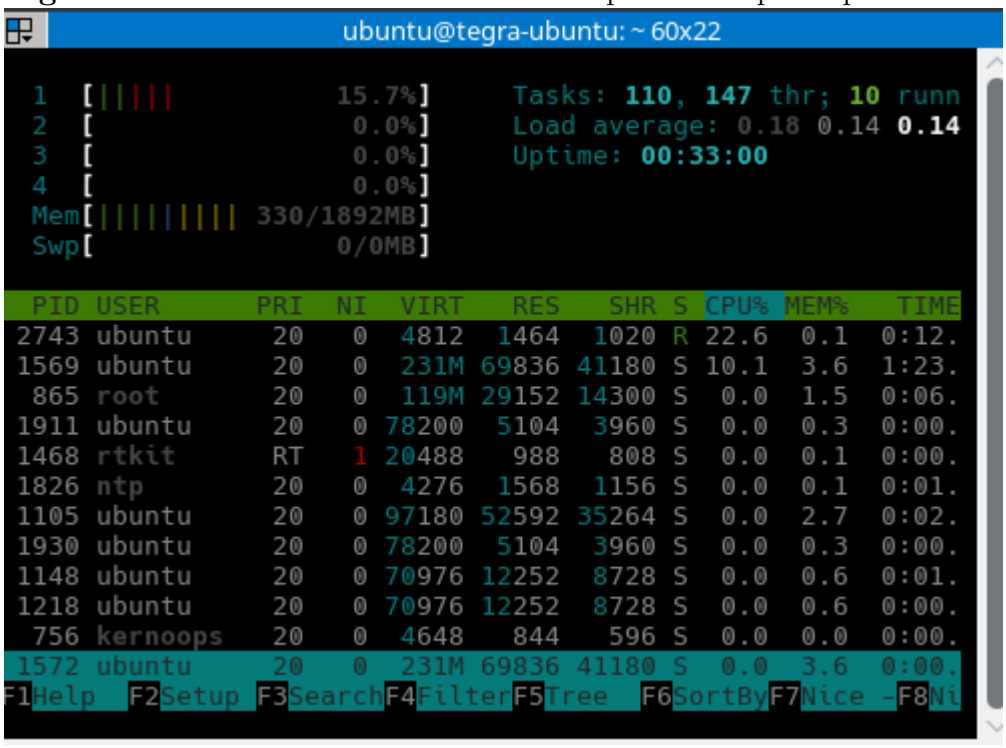
Resulta cómodo para el usuario final tener un sistema operativo listo usar, aunque no siempre esto será eficiente, debido a que así como se cuenta con paquetes necesarios para el usuario, también se tendrá otros que no van a ser necesarios, y un dispositivo con recursos restringidos como estos cualquier espacio o recurso que se deje libre influirá a la hora de ejecución de tareas; aunque si lo que se desea es obtener un rendimiento maximizado también está la opción de construir su propia distribución, pero eso conlleva tiempo y conocimientos específicos, aún así tampoco se garantiza que el resultado sea el esperado.

Para este proyecto si bien no se ha construido una nueva distribución, se buscó optimizar la ya instalada, removiendo la paquetería no necesaria, así como apagando demonios del sistema(ver figura 10), esto contribuyó a mejorar el consumo de memoria por parte de aplicaciones abiertas en segundo plano, al pasar de 350 Mb a 50 -100 Mb en promedio, importante si lo que se desea es tener recursos disponibles para procesamiento que en dispositivos como éstos son limitados(ver figura 11).

3.3.2. Configuración de red

Todos los nodos del cluster tienen asignada una dirección IP en la subred 192.168.66.231/237, basado en lo siguiente el storage, que es una máquina virtual en GUANE tendrá

Figura 10: Consumo de recursos antes de aplicar cualquier optimización

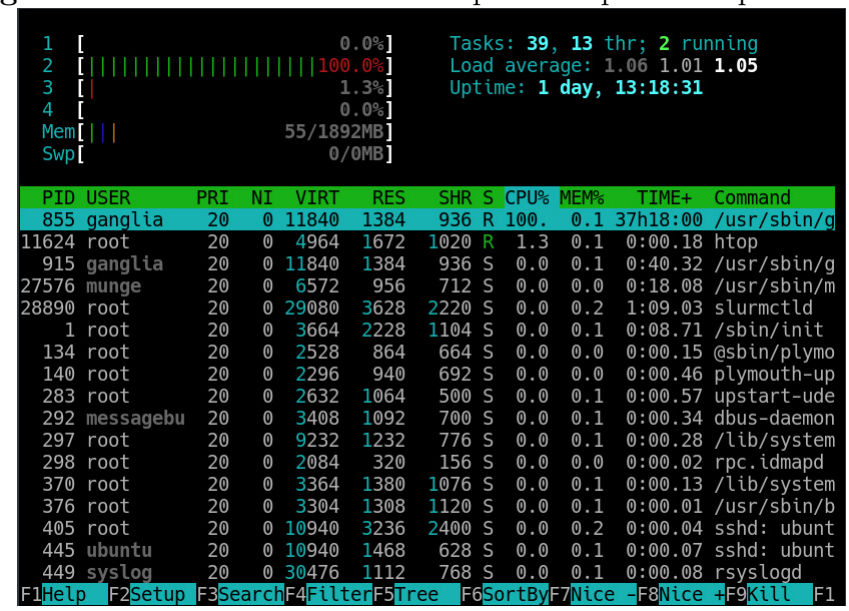


asignada la dirección IP 192.168.66.231, los demás nodos tendrán disponibles desde la 192.168.66.232/237, para ser consecuentes con el orden, el nodo 1 maestro será el 192.168.66.237 y el resto empiezan desde la 232, dejando libre dos espacios que pueden ser utilizados para escalar el cluster mas adelante.

Cada nodo se encuentra configurado de tal forma que la IP asignada se realiza de forma estática, previamente asignada por el administrador de red, esto permite una administración física de los nodos más fácil, aunque el escalamiento se ve limitado si no se obtienen más direcciones estáticas para los nodos.

El acceso a cada nodo de cómputo mediante un cliente ssh está restringido solamente al nodo maestro, esto se realiza con fines de obtener mayor seguridad, aunque los nodos comparten sus llaves ssh para el envío de paquetes MPI entre ellos sin necesidad de autenticación, de igual forma se utiliza munge para permitir la comunicación entre los nodos cuando se está haciendo ejecución de tareas por medio del Slurm.

Figura 11: Consumo de recursos después de aplicar la optimización



3.3.3. Sistema de archivos compartidos en red: NFS

El almacenamiento en disco es otra de las limitantes de éstos dispositivos, aunque cuentan con 16 GB y con capacidad para expandirla mediante el uso de SD externa o la conexión de un disco duro SATA, se propuso crear una carpeta compartida en red para todo el cluster, en donde se guardaran todos los archivos necesarios para procesamiento, de igual manera también los fuentes y los programas compilados, así solo se compila una vez y está disponible para todas los nodos.

Las carpetas compartidas se encuentran en:

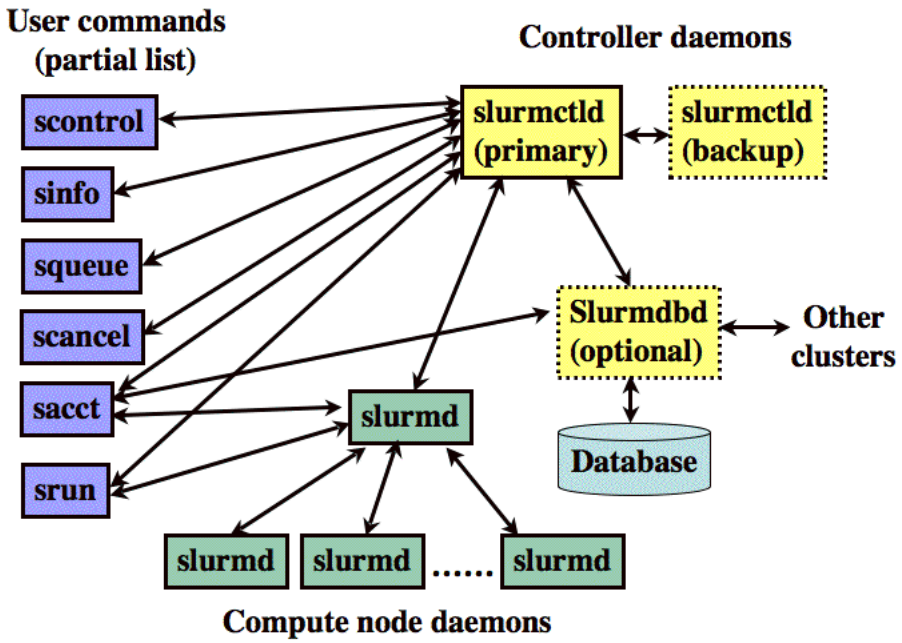
- /mnt/share. Utilizada para almacenar documentos para procesamiento y demás cosas ajenas a fuentes y programas compilados.
- /usr/local. Esta carpeta contiene todos los fuentes (/usr/local/src) y los programas compilados, quienes estarán disponibles para ser utilizados por todos los nodos simplemente haciendo un PATH hacia las librerías de cada aplicativo.

3.3.4. Manejador de recursos

Este software se encarga de la localización de recursos disponibles para realizar cómputo dentro del cluster y enviar la carga de trabajo a procesar, de igual manera puede realizar reservas para que estos recursos sean exclusivos de un usuario por la cantidad de tiempo que el así lo requiera.

Para éste proyecto se escoge Slurm entre otras alternativas debido a que es la más utilizada por cerca del 60 % de los supercomputadores que componen la lista del TOP500, muy popular por su capacidad de adaptamiento y modularidad, lo que permite integrar nuevas funcionalidades al software base.
Slurm como manejador de recurso necesita reconocer los nodos que tiene disponibles para la ejecución de tareas, por tal motivo se requiere de una herramienta que permita acceder a los nodos sin necesidad de autenticarse cada vez que requiera enviar una tarea, para tal fin se instala Munge, el cual se encarga de crear y validar credenciales.

Figura 12: Arquitectura de Slurm[16]



El proceso para la creación de las credenciales empieza instalando Munge en todos los nodos (maestro y nodos de cómputo), luego se crea la llave encriptada la cual se compartirá en todos los nodos y es la que permite al demonio ejecutado en segundo plano validar la identidad del host que requiere acceso, en este caso especial esta llave se comparte mediante un sistema compartido de archivos, como el proceso es el mismo en todos los nodos del cluster, se procede a montar un sistema de archivos compartidos NFS en el directorio /usr/local, de esta forma solo se hace una vez y se encuentra disponible en todos los dispositivos.(ver anexos)
Para la implantación del manejador de recursos con miras a hacer la instalación más eficiente y aprovechar el rendimiento de la arquitectura ARM se procedió a compilar

el código fuente en los nodos, esto nos asegura que la aplicación consume menos almacenamiento y su ejecución será mucho más rápida porque está optimizada. Realizar el path para que los comandos de slurm sean reconocidos en la consola (ver anexo: activar comandos slurm)
Lo siguiente después de instalar el calendarizador de recursos es la configuración de este, en el directorio /usr/local/slurm/etc/ modificar (o crear) el archivo slurm.conf es donde se configura el cluster, la información de cada uno de los nodos y los recursos de los que dispone. (ver anexo: archivo configuración slurm)

3.3.5. Paquetes instalados y librerías

Para la ejecución de benchmark en los dispositivos se hace necesario la instalación de algunos paquetes como: CUDA, MPI y la librería BLAS (basic linear algebra subroutines). Para CUDA, todos los nodos por defecto tienen instaladas las librerias de CUDA 6.5 que vienen en la distribución Linux pre-instalada de fábrica. Las librerías MPI, como son OpenMPI y MPICH, se descargaron las fuentes con las últimas versiones desde los repositorios de los programadores y se procedió a compilar en base a la arquitectura de cada nodo, con el fin de optimizar la instalación y buscar un mejor aprovechamiento de los recursos de máquina(ver anexo: instalación MPI). Y para la librería BLAS, se descargó la última versión de ATLAS (Automatically Tuned Linear Algebra Software) la cual trae consigo una versión optimizada de ATLAS(ver anexo: Instalación de ATLAS).

3.3.6. Configuración HPL Linpack

Haciendo uso de el Benchmark Linpack HPL modificado para aprovechar las GPU's se busca medir cuan rápido una máquina es capaz de resolver un sistema denso n x n de ecuaciones lineales Ax = B, el resultado de este análisis es la medición de rendimiento, la cual me dice que cantidad de operaciones de punto flotante puede realizar por segundo dicha máquina. Con esta prueba se desea conocer si hay beneficio o no al hacer uso de las GPU's para hacer procesamiento.

La versión elegida fue la hpl-2.0_FERMI_v15, pre-configurada para trabajar con GPU's NVIDIA pero de arquitectura FERMI, aunque con unas pequeñas modificaciones en el conjunto de instrucciones de compilación que se encuentran en el directorio principal en el archivo README, puede hacer que funcione para este tipo de arquitecturas de GPU. En los anexos se incluye un modelo del Make utilizado para la construcción del ejecutable.

3.3.7. Configuración OSU-Benchmark

OSU-Bechmark provee una serie de herramientas desarrolladas por MVAPICH para hacer pruebas de red y realizar diferentes mediciones a las librerías de paso de mensaje. Como tal ninguna de prueba necesita de un script de configuración específico, cuando se descarga el paquete de benchmarks se compilan dependiendo de la arquitectura de procesamiento de cada máquina.

La ejecución de las pruebas se realiza con las diferentes herramientas de paso de mensajes, bien sea OpenMPI o MPICH, en este caso se lanzan pruebas con ambas con el fin de comparar si hay diferencia alguna entre el uso de una u otra. (ver anexo: ejecución de OSU-Benchmark)

Capítulo 4

Ejecución de aplicaciones y análisis de rendimiento

El lograr el mayor rendimiento en la ejecución de aplicaciones depende de la forma como éstas se encuentran optimizadas para aprovechar los recursos del sistema.

Con la implementación de nuevas arquitecturas para procesamiento los fabricantes ponen a disposición del usuario soluciones que van desde sistemas con un solo procesador pero con varios cores hasta sistemas más específicos como utilizados en computación de alto rendimiento, donde se puede tener un procesador con hasta 24 cores y que un mismo nodo tenga varios procesadores.

Además nuevos fabricantes como NVIDIA o AMD, han incursionado en el mercado de las Unidades de Procesamiento Gráfico dedicadas para el procesamiento de alto rendimiento; lo que supone una alternativa para la construcción de soluciones más potentes debido a que la tarea de procesamiento no recae solo en la CPU si no que a través de la GPU (o varias GPUs en el mismo nodo) se puede realizar procesamiento también de forma paralela.

Para este caso en especial cada nodo cuenta con un procesador de 4 cores y una GPU NVIDIA de arquitectura Kepler que según el fabricante posee un rendimiento por encima de los 172 GFLOPS, la cual apoya el procesamiento de aplicaciones. Las mediciones llevadas a cabo se basan en 3 pruebas:

- Medición de aprovechamiento de red
- Medición de rendimiento de CPU
    - Rendimiento a nivel de nodo

- Medición de rendimiento del cluster propuesto

4.1. Análisis de rendimiento a nivel de aprovechamiento de red

Como la forma de comunicarse entre nodos del cluster se hace a través de red, las librerías de paso de mensajes como OpenMPI y MPICH son las que se encargan de hacer el paso de mensajes entre los nodos con el código de la aplicación (escrita en C, C++ o Fortran)

En este análisis se busca medir cual de las dos ofrece un mejor rendimiento al cluster, de tal manera que aproveche mejor los recursos de red y la arquitectura de los nodos. Para esto se descarga una versión desde los repositorios, luego se compilada según la arquitectura y se instala en una compartida entre los dispositivos.

4.1.1. Evaluación OpenMPI vs MPICH

Para la evaluación de las 2 instalaciones se hace uso del benchmark Osu-micro-benchmark desarrollado por MVAPICH basado en MPICH.[12]

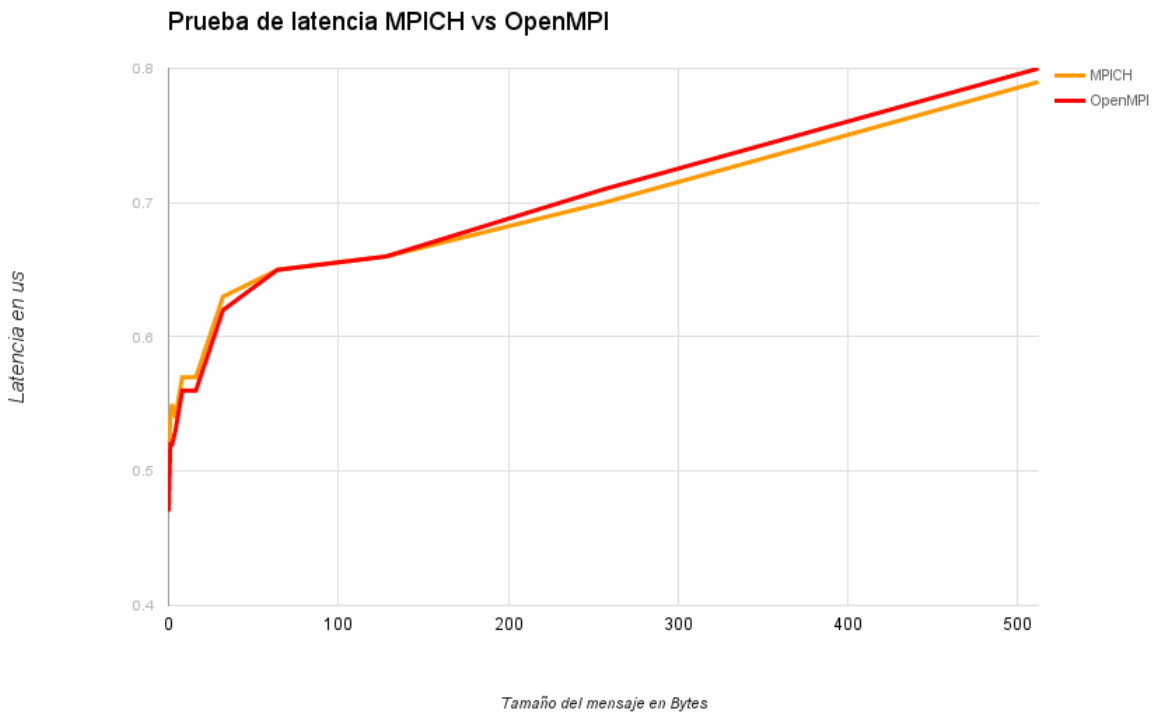
Con ellos se busca medir métricas de latencia y ancho de banda de la librería MPI

Si bien el uso de cualquiera de las dos librerías no supone un cambio abrupto, MPICH tiene una menor latencia, lo que en términos prácticos quiere decir que la transmisión de un paquete se hará de forma más rápida, aunque su comportamiento nos dice que para paquetes pequeños resulta menos eficiente que OpenMP.

Llevado a la práctica el uso de cualquiera de las dos librerías no supone una gran diferencia porque las velocidades de latencia están dadas en micro-segundo, en la figura 13 se puede observar las diferencias entre las transacciones de paquetes hechos con cada una de las librerías. Entre menor sea la latencia entre paquetes, mayor será el rendimiento de los nodos en conjunto.

En la figura 14 se observa que de la totalidad de los 6 MB/s que posee la red en la que se interconectan los nodos, ambas librerías tienen un aprovechamiento similar del ancho de banda, destacando que MPICH tiene una leve ventaja frente a OpenMP aunque no muy significativa, lo que si vale la pena destacar es la diferencia que se encuentra entre

Figura 13: Latencia entre MPICH vs OpenMPI



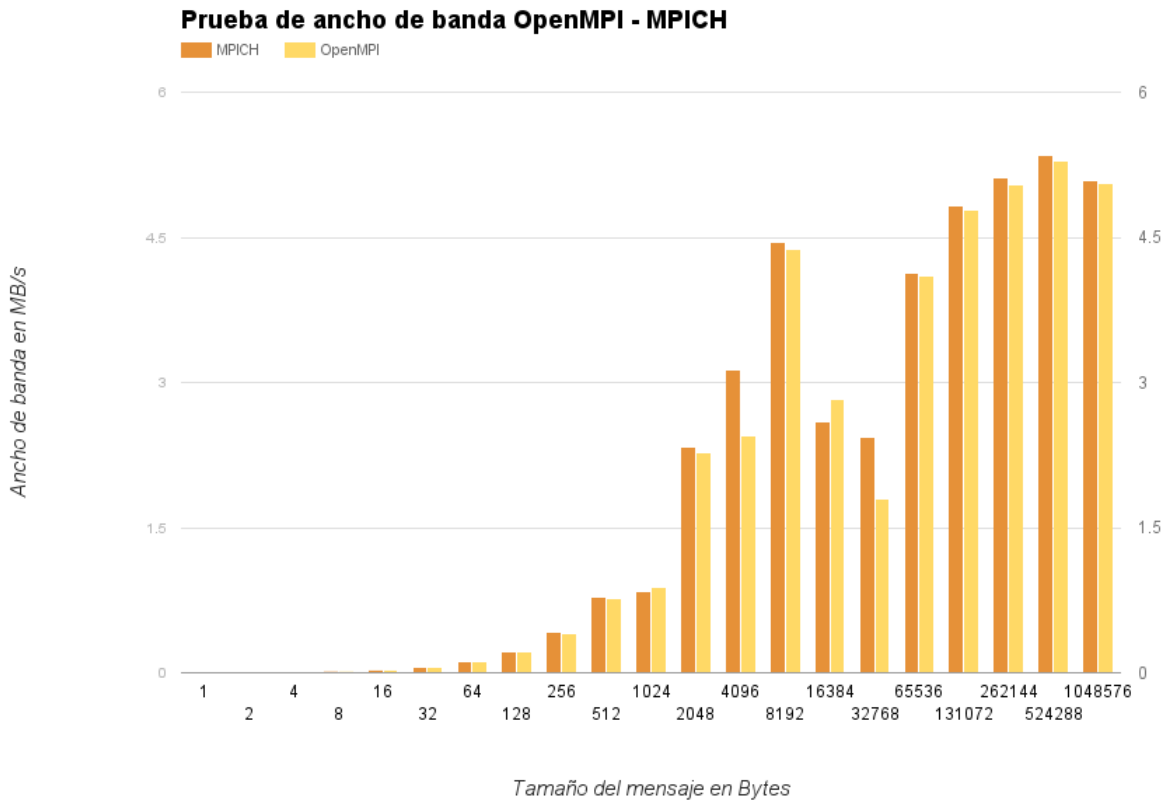
paquetes de 8 - 130 MB en donde se presenta un descenso acrecentado en el ancho de red que utilizan para la transmisión, a esto se le puede atribuir un problema de ruido en la red.

4.2. Análisis de rendimiento a nivel de CPU

Para realizar éste análisis se utiliza ATLAS (Automatically Tuned Linear Algebra Software), cuyo propósito es proveer software de álgebra lineal óptimo y portátil, el cual sirve de apoyo para otros aplicativos como:

- MAPLE
- MATLAB
- Mathematica

Figura 14: Prueba de ancho de banda entre OpenMPI - MPICH



- Octave

También es utilizado como apoyo para múltiples benchmark, entre ellos el más importante: linpack HPL, con el que se realizan las mediciones para el escalafón del TOP500 de supercomputadoras; de ahí el interés de instalarlo y medir el rendimiento a nivel de cpu para los nodos. Valga la aclaración que al tener nodos de arquitectura homogénea tanto hardware como software la medición se realiza una sola vez y se toma como teórica para los demás.

Este proceso consta de dos partes, en ellas se busca establecer diferencias entre construir una versión optimizada para la arquitectura del hardware y la otra hacer la misma instalación de forma genérica y observar si hay o no beneficio.

Si bien ATLAS no es un benchmark, contiene herramientas que ayudan a medir el rendimiento del procesador ejecutando rutinas de Algebra Lineal.

Tabla 1: Tiempo para precisión simple de ATLAS optimizado

	Single precision			
	Real		**Complex**	
Benchmark	**Reference**	**Present**	**Reference**	**Present**
kSelMM	177.6	173.0	179.0	175.9
kGenMM	151.2	152.6	128.1	155.2
kMM_NT	119.6	148.1	131.8	133.2
kMM_TN	124.8	141.6	134.3	137.8
BIG_MM	175.1	173.1	179.1	176.8
kMV_N	61.2	57.5	102.1	105.8
kNV_T	76.9	86.2	107.1	99.2
kGER	49.0	51.7	87.7	88.1

4.2.1. Medición con ATLAS optimizado para la arquitectura

Las tablas anteriores muestra en la columna reference los valores estimados para la arquitectura dados por los programadores de ATLAS y en present los calculados a partir de los flags declarados de acuerdo a las características de la arquitectura que se posee para compilar.

Notablemente se puede observar que algunos valores se ajustan en referencia a los esperados y otros difieren, lo que hace intuir que la optimización no es del todo buena o las banderas no son las acertadas. Otro aspecto a resaltar es el rendimiento a la hora de hacer operaciones de doble precisión, debido a que éste tipo de arquitectura no tiene soporte NEON de doble precisión, en consecuencia el procesador debe ser exigido más a la hora de hacer este tipo de procesamiento y la demora puede significar pérdida de poder de computo a la hora de ser evaluado.

Aún así el rendimiento dado por el aplicativo de ATLAS muestra que el procesador tiene un rendimiento de 3.73 Gflops, es decir, que puede procesar $3.73*10^9$ operaciones de coma flotante por segundo

Tabla 2: Tiempo para precisión doble de ATLAS optimizado

	Double precision			
	Real		**Complex**	
Benchmark	**Reference**	**Present**	**Reference**	**Present**
kSelMM	172.5	161.3	169.7	164.1
kGenMM	139.7	137.7	141.6	133.1
kMM_NT	127.6	120.8	122.4	105.7
kMM_TN	141.5	128.7	131.4	128.7
BIG_MM	165.9	155.1	167.7	160.8
kMV_N	39.4	39.7	70.2	72.2
kNV_T	47.6	44.1	72.3	72.2
kGER	28.3	28.3	53.4	52.9

4.2.2. Medición con ATLAS no optimizado

En el caso de la versión no optimizada (ver tabla 3) se da cuenta de que los resultados obtenidos son acordes a los esperados, razón por la que se puede concluir que las banderas dadas para compilación no son suficientes para obtener un mejor rendimiento, a continuación se presenta una comparativa del rendimiento general de la cpu, en donde se confirma que la opción más acertada en este caso es la instalada desde los repositorios.

Aún así el rendimiento dado por el aplicativo de ATLAS muestra que el procesador tiene un rendimiento de 3.91 Gflops, es decir, que puede procesar $3.91*10^9$ operaciones de coma flotante por segundo

De igual forma se quiere aclarar que el proceso de compilar esta herramienta toma alrededor de 5 horas, mientras que desde repositorios no lleva más de una hora. Por tal motivo aplicando banderas de compilación más específicas, los tiempos podrían verse afectados y tenderán a subir.

Tabla 3: Tiempo para precisión simple de ATLAS no optimizado

	Double precision			
	Real		**Complex**	
Benchmark	**Reference**	**Present**	**Reference**	**Present**
kSelMM	177.6	182.0	179.0	181.3
kGenMM	151.2	155.7	128.1	143.4
kMM_NT	119.6	130.3	131.8	147.2
kMM_TN	124.8	124.5	134.3	122.9
BIG_MM	175.1	176.4	179.1	180.2
kMV_N	61.2	64.7	102.1	99.2
kNV_T	76.9	86.2	107.1	105.8
kGER	49.0	47.0	87.7	88.1

Tabla 4: Tiempo para precisión doble de ATLAS no optimizado

	Double precision			
	Real		**Complex**	
Benchmark	**Reference**	**Present**	**Reference**	**Present**
kSelMM	172.5	178.9	169.7	173.3
kGenMM	139.7	139.8	141.6	141.5
kMM_NT	127.6	128.8	122.4	130.0
kMM_TN	141.5	142.9	131.4	130.0
BIG_MM	165.9	168.1	167.7	168.6
kMV_N	39.4	39.7	70.2	72.2
kNV_T	47.6	44.1	72.3	72.2
kGER	28.3	28.3	53.4	52.9

4.3. Analisis de rendimiento a nivel de cluster

La finalidad de esta prueba es conocer el rendimiento de todo el conjunto del cluster, allí se comprueba la integración que se pueden dar entre los distintos componentes de cada nodo y la forma como éstos se unen para resolver un problema específico.

El conjunto de set de instrucciones de punto flotante del Linpack son de 64 bits, esto no significa que son incompatible con el tipo de arquitectura de los nodos, éstos podrán hacer el procesamiento siempre y cuando el tamaño del problema no desborde la capacidad en memoria, en consecuencia el problema se puede resolver pero tardará más tiempo. Así la medición hecha por el benchmark será muy diferente a la específica de fabrica, conocida también como capacidad teórica.

Tabla 5: Tamaños de problema para diferente cantidad de nodos

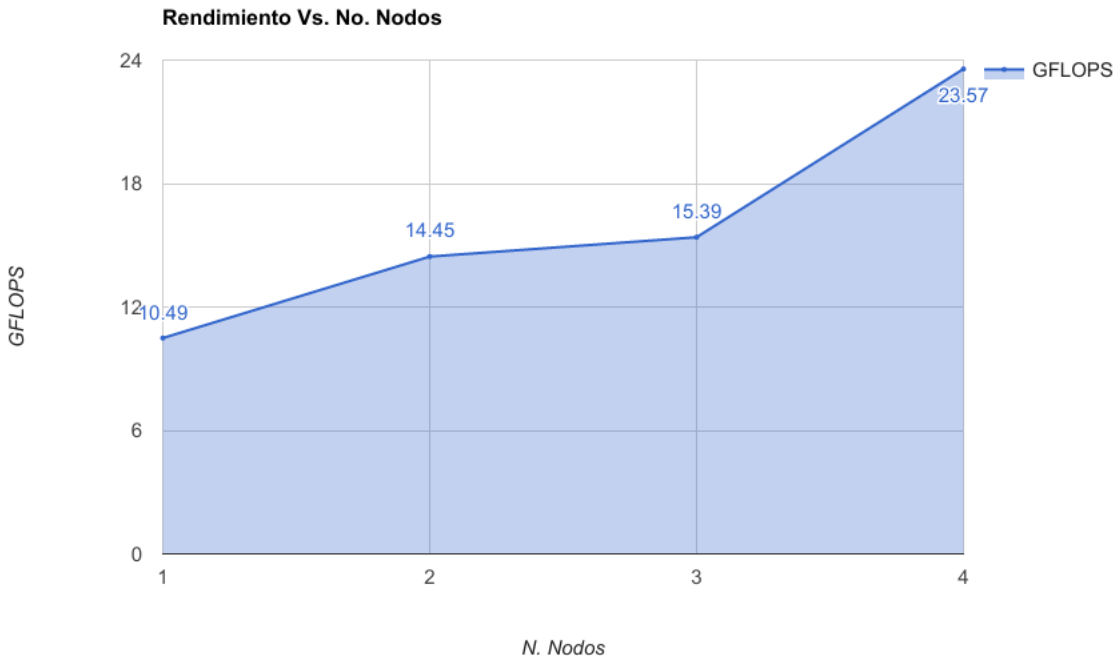
No. Nodos	N	P	Q	NB	GFLOPS	TIEMPO
1	14080	2	2	192	10.49	177.45
2	19584	2	4	192	14.45	346.58
3	24192	3	4	192	15.39	613.42
4	28192	4	4	192	23.57	634.43

El hpl linpack hace la medición de rendimiento basándose en el tiempo que se demora en resolver un sistema de ecuaciones de tamaño N, en este caso las pruebas realizadas en la figura 15 buscan hallar el tamaño máximo de problema a realizar con la cantidad de nodos disponibles, maximizando el rendimiento, cabe aclarar que el tamaño del problema se ve limitado por la capacidad de memoria en los nodos.

Que si bien muestra un aumento de rendimiento al aumentar el número de nodos para el procesamiento, nos muestra también que la comunicación penaliza el rendimiento que puede ofrecer el cluster, entre más nodos se utilicen mayor será el costo de enviar paquetes, viéndose reflejado por ejemplo en el aumento entre tener 1 nodo y 2, se esperaba un aumento del doble en el rendimiento, pero a cambio solo llegó a la mitad, algo sospechoso si se analiza el caso de tener 3 nodos para procesamiento, donde no se nota un cambio significativo.

Reflejado en tiempos cuando se utilizan 3 nodos para procesamiento se obtiene un aumento del noble al utilizado para obtener el máximo rendimiento con 2 nodos, razón

Figura 15: Gráfica de Número de nodos vs rendimiento



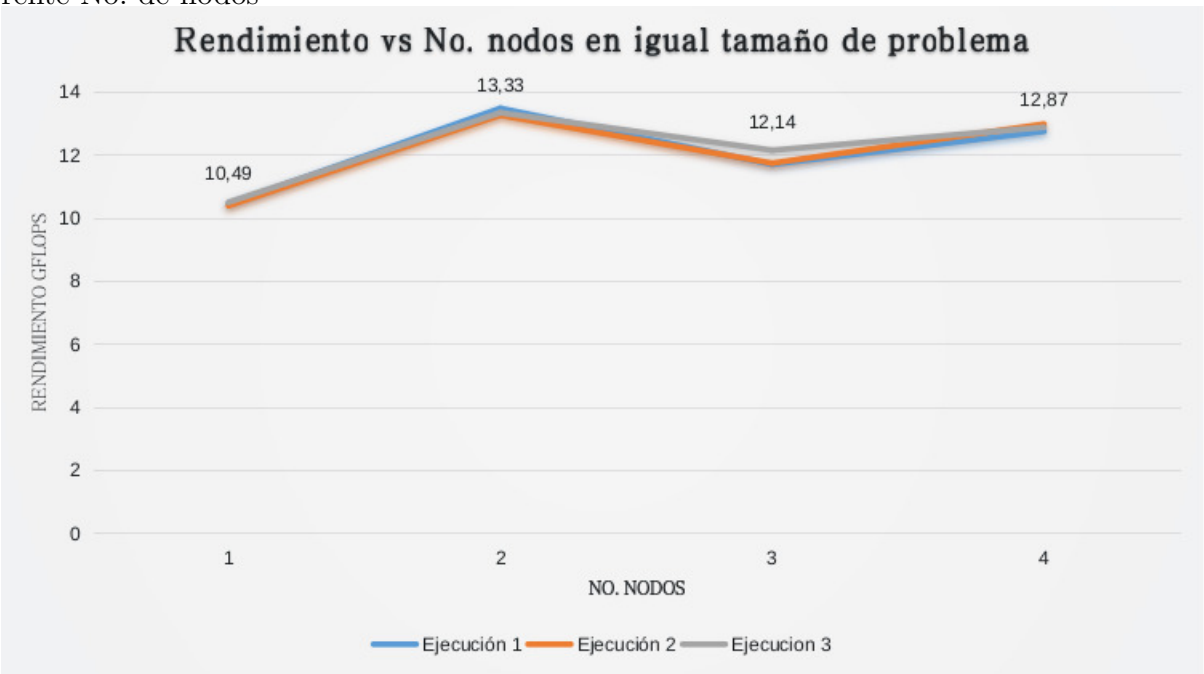
para seguir pensando en que la comunicación o la configuración no están permitiendo obtener un óptimo rendimiento.

En el caso del mayor rendimiento para todo el cluster se observa un aumento casi igual al rendimiento dado por un solo nodo, lo que puede intuirse, es que las configuraciones de entrada no están permitiendo un óptimo aprovechamiento de recursos con miras a obtener el mayor rendimiento con diferentes configuraciones para el cluster.

Otro de los problemas tratados en las mediciones está dado cuando se posee el mismo tamaño para la matriz de entrada, escoger una cantidad de óptima de nodos para el procesamiento no es una tarea fácil, en este caso puntual se basa en intentos de prueba y error llegando a la conclusión que no siempre el uso de más nodos para procesar es lo óptimo, cada transacción tiene un costo y eso influye en el procesamiento, por esto la tarea está en buscar la cantidad óptima de nodos para realizar el procesamiento. Ver figura 16

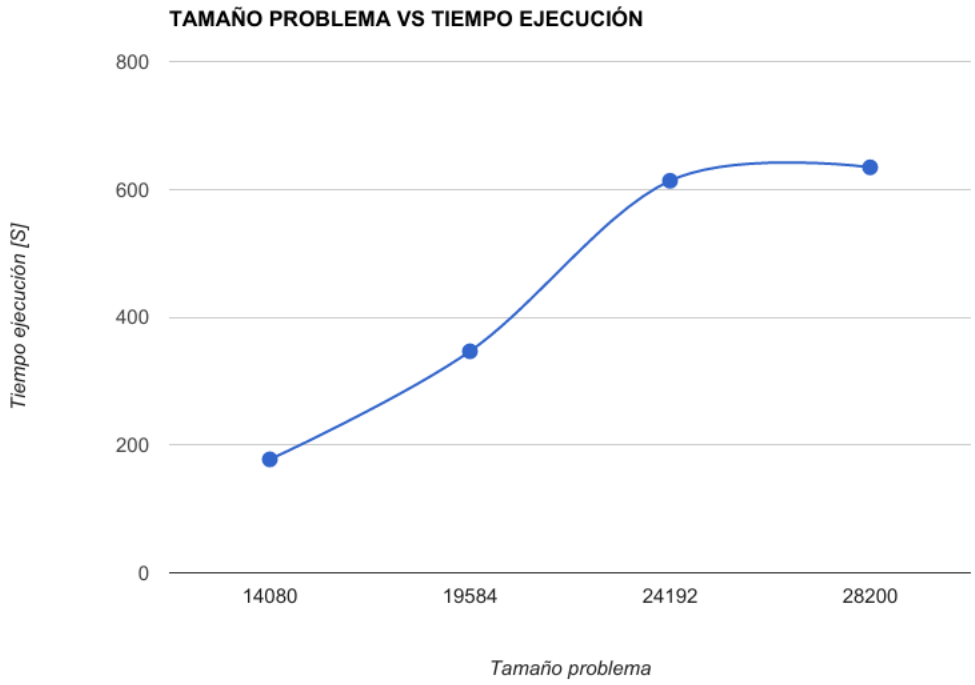
Vale la pena señalar que la gráfica de tiempo vs tamaño de problema nos muestra más

Figura 16: Gráfica de ejecución rendimiento en problema de igual tamañan con diferente No. de nodos



gráficamente la forma como va subiendo el tiempo que gastan los nodos para ejecutar el tamaño máximo de tareas posibles, y como se puede observar entre la ejecución hecha por 2 y 3 nodos existe una diferencia muy acentuada, diferente a lo que sucede cuando se utiliza el cluster completo, mostrando que hay un punto donde el tiempo de ejecución no aumenta si no que permanece constante hasta llegar al límite.

Figura 17: Gráfica de Tamaño del problema vs Tiempo de ejecución



Capítulo 5

Conclusiones

5.1. Conclusiones

- Se comprobó que la utilización de dispositivos embebidos con arquitectura RISC para el cómputo de alto rendimiento es una alternativa eficiente respecto al uso de arquitecturas comunes.

- Se ha comprobado que el scheduler diseñado para administrar clusters de gran tamaño y diferentes arquitecturas, puede funcionar en dispositivos embebidos sin perder funcionalidad alguna.

- La construcción de un cluster con este tipo de dispositivo permite abordar problemas de mayor tamaño que bien pudiesen ser resueltos en dispositivos con mayor poder de cómputo pero con menor eficiencia energética.

5.2. Trabajo futuro

- Si bien de fábrica cada dispositivo trae su propia distribución de sistema operativo y su paquetera instalada, sera algo muy interesante la construcción de unos lineamientos que conduzcan a una optimización mas profunda en cuanto a kernel y paquetera pre-instalada, con interés en el cómputo de alto rendimiento ya sea enfocado a la visualización o al cálculo de problemas complejos.

- Realizar el escalamiento del cluster con cintos de dispositivos, conservando la misma arquitectura.

- Experimentar con dispositivos de tipo arm64 bits, sea la NVIDIA TX1, u otro disponibles en el mercado, los cuales en teoría deben brindar un mejor rendimiento, manteniendo un bajo consumo.

Bibliografía

[1] *Cómputo Integrado*. http://rcmcomputointegrado.blogspot.com.co/2012/03/arquitectura-risc-y-cisc.html

[2] *Nvidia Jetson TK1*. http://elinux.org/File:Nvidia-tegra124-jetson-tk1-labelled.jpg

[3] Barbosa, Jorge G. ; Moreira, Belmiro

[4] BDti. *NVIDIA's Jetson TK1 Evaluation Board Gets Embedded Vision*. http://www.bdti.com/InsideDSP/2014/04/29/NVIDIA

[5] Bravo Navarro, J. A. ; Cortiguera Herrera, Héctor ; Bravo Navarro, J. A. ; Cortiguera Herrera, Héctor and Quintás Rodríguez, Jorge: Minix 3 sobre arquitectura ARM. 2009. – Informe de Investigación

[6] Center, Barcelona S. *Introduction to Mont Blanc*. http://www.montblanc-project.eu/project/introduction

[7] ESPE. *SMP Architecture*. http://publiespe.espe.edu.ec/articulos/sistemas/arquitectura/arqu

[8] Ganglia. *Ganglia Monitoring System*. http://ganglia.info/

[9] Guglieri, J.A.C.: *Reingeniería y seguridad en el ciberespacio*. Díaz de Santos, 1996. – 124 p.. – ISBN 9788479782733

[10] Januario, Sebastien: La arquitectura ARM64 entra en la alta computación gracias a las GPU NVIDIA. 2014. – Informe de Investigación

[11] Moore, Gordon. *Ley de Moore*

[12] MPICH. *Osu Benchmark*. http://mvapich.cse.ohio-state.edu/benchmarks/

[13] Nagios Enterprises, LLC. *What is Nagios?* https://www.nagios.org/about/overview/

[14] Oetiker, Tobi. *The Multi Router Traffic Grapher*. http://oss.oetiker.ch/mrtg/

[15] Qingbo, Y. ; Yungang, B. ; Mingyu, C. ; Ninghui, S.: A Scalability Analysis of the Symmetric Multiprocessing Architecture in Multi-Core System. (2009), July, p. 231–234

[16] SchedMD. *Slurm Workload Manager - Overview.* https://slurm.schedmd.com/overview.html

[17] solingest. *Cluster de servidores, ¿Qué es y como funciona?* http://www.solingest.com/blog/cluster-de-servidores-que-es-y-como-funciona

[18] Wikipedia. *Almacenamiento conectado en red.* https://es.wikipedia.org/wiki/Almacenamiento_conectado_en_red

[19] Wikipedia. *Cluster Informático.* https://es.wikipedia.org/wiki/Cl %C3 %BAster_(inform %C

[20] Wikipedia. *Computación Exaescale.* https://es.wikipedia.org/wiki/Computaci %C3 %B3n_E

•

Anexo: Script para la habilitación de cores de CPU

Su ejecución se puede dar en cualquier momento que se desee levantar las restricciones impuestas por el governor del sistema, por tal motivo se guarda en un archivo .sh y se ejecuta una sola vez

```
#!/bin/bash

echo '0' ¿/sys/devices/system/cpu/cpuquiet/tegra_cpuquiet/enable

g_ONLINE="$(cat /sys/devices/system/cpu/cpu0/online)";
if [ "$g_ONLINEi= '1' ]; then
echo '1' ¿/sys/devices/system/cpu/cpu0/online
fi

g_ONLINE="$(cat /sys/devices/system/cpu/cpu1/online)";
if [ "$g_ONLINEi= '1' ]; then
echo '1' ¿/sys/devices/system/cpu/cpu1/online
fi
g_ONLINE="$(cat /sys/devices/system/cpu/cpu2/online)";
if [ "$g_ONLINEi= '1' ]; then
echo '1' ¿/sys/devices/system/cpu/cpu2/online
fi
g_ONLINE="$(cat /sys/devices/system/cpu/cpu3/online)";
if [ "$g_ONLINEi= '1' ]; then
echo '1' ¿/sys/devices/system/cpu/cpu3/online
fi

echo 'performance' ¿/sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
```

Anexo: Comandos para la optimización de sistema operativo

```
//Desinstalar el unity
$sudo apt-get purge unity
//Desinstalar el xorg-server-control
$sudo apt-get purge xserver-xorg-core
//Desinstalar el libx11.* y libqt*
$sudo apt-get purge libx11.* libqt.*
//Luego hacer un update de los paquetes
$sudo apt-get update
```

Anexo: Instalación Munge

```
//Instalar munge y sus dependencias.
$apt-get install libmunge-dev libmunge2 munge

// Creación llave , cambio de permisos
$dd if=/dev/urandom bs=1 count=1024 ¿/etc/munge/munge.key
$chown munge:munge /etc/munge/munge.key
$chmod 400 /etc/munge/munge.key

//Los demás nodos del clúster deben tener la misma llave del nodo maestro
//para esto se realiza
$scp /etc/munge/munge.key root@192.168.66.237:/etc/munge/

//Repetir estos pasos en todos los nodos
//cambiar el pass en /etc/passwd :
$munge:x:501:501::/var/run/munge;/sbin/nologin

//Forzar el arranque en /etc/default/munge
OPTIONS=-force"

//Revisar Propietarios:
$ls -ld /etc/munge/ /var/*/munge

//Para cambiar ejecutar los siguientes comandos:
$chown munge:munge /etc/munge/
$chown munge:munge /var/lib/munge/
$chown munge:munge /var/log/munge/
$chown munge:root /var/run/munge/

//La salida esperada será:
//drwxr-xr-x 2 munge munge 4096 Nov 21 06:28 /etc/munge/
```

```
//drwxr-xr-x 2 munge munge 4096 Nov 27 07:49 /var/lib/munge
//drwxr-xr-x 2 munge munge 4096 Nov 27 07:49 /var/log/munge
//drwxr-xr-x 2 munge root 100 Nov 27 07:49 /var/run/munge

//Cambio de permisos en la carpeta que contiene los logs: $chmod g+w
/var/log
$chmod 775 /var/log

//Creación del archivo y cambio de permiso de los logs del sistema:
$touch /var/log/munge/munged.log
$chmod 640 /var/log/munge/munged.log
$chown munge:munge /etc/munge/munge.key
$chmod 400 /etc/munge/munge.key

//Inicio del demonio munge
$/etc/init.d/munge start
```

Anexo: Compilación Slurm

```
$cd /usr/local/src
$wget http://www.schedmd.com/download/latest/slurm-16.05.4.tar.bz2
$tar xvjf slurm-16.05.4.tar.bz2
$cd /usr/local/src/slurm-16.05.4
$./configure –prefix=/usr/local/slurm
$make -j4
$make install
```

Pasos para realizar el PATH de la ruta Slurm hacia las librerías

```
$nano /etc/profile.d/
$nano /etc/profile.d/slurm.sh
$source /etc/profile
```

Anexo: Archivo de configuración Slurm

Este archivo se encuentra en /usr/local/slurm/etc, su nombre es slurm.conf. Con este archivo de configuración Slurm reconoce cada uno de los nodos del cluster, asi como los recursos disponibles y el enlace a las librerías necesarias para la ejecución de aplicativos.

```
# slurm.conf file generated by configurator.html.
# Put this file on all nodes of your cluster.
# See the slurm.conf man page for more information.
#
ControlMachine=coconuco01
ControlAddr=192.168.66.237
#
AuthType=auth/munge
CryptoType=crypto/munge
ReturnToService=2
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=root
SlurmdUser=root
StateSaveLocation=/var/spool/slurmctld
SwitchType=switch/none
#
# TIMERS
InactiveLimit=0
KillWait=30
MinJobAge=300
SlurmctldTimeout=120
```

```
SlurmdTimeout=10
Waittime=0
#
# SCHEDULING
FastSchedule=1
SchedulerType=sched/backfill
SchedulerPort=7321
SelectType=select/cons_res
SelectTypeParameters=CR_Core
#
# LOGGING AND ACCOUNTING
AccountingStoreJobComment=YES
ClusterName=coconuco
JobCompType=jobcomp/none
JobAcctGatherFrequency=30 JobAcctGatherType=jobacct_gather/linux
SlurmctldDebug=3SlurmdLogFile=/var/log/slurm/slurmd.log
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdDebug=3
#
# COMPUTE NODES
PartitionName=debug Nodes=coconuco[01-04] Default=YES MaxTime=INFINITE State=UP
NodeName=coconuco01 CPUs=4 RealMemory=1892 Sockets=4 CoresPerSocket=1 ThreadsPerCore=1 State=UNKNOWN
NodeName=coconuco02 CPUs=4 RealMemory=1888 Sockets=4 CoresPerSocket=1 ThreadsPerCore=1 State=UNKNOWN
NodeName=coconuco03 CPUs=4 RealMemory=1894 Sockets=4 CoresPerSocket=1 ThreadsPerCore=1 State=UNKNOWN
NodeName=coconuco04 CPUs=4 RealMemory=1894 Sockets=4 CoresPerSocket=1 ThreadsPerCore=1 State=UNKNOWN
```

Anexo: Instalación OpenMPI y MPICH

```
$wget https://www.open-mpi.org/software/ompi/v1.8/downloads/openmpi-1.8.3.tar.bz2
$tar xvjf openmpi-1.8.3.tar.bz2
$cd openmpi-1.8.3
$CFLAGS=O3 -march=armv7-a -mcpu=cortex-a15 -mtune=cortex-a15 -mfloat-abi=hard"
$./configure –prefix=/usr/local/openmpi1.8.3
$make -j4
$make install
```

```
$wget http://www.mpich.org/static/downloads/3.1.3/mpich-3.1.3.tar.gz
$tar -zxvf mpich-3.1.3.tar.gz
$cd mpich-3.1.3
$CFLAGS=O3 -march=armv7-a -mcpu=cortex-a15 -mtune=cortex-a15 -mfloat-abi=hard"
$./configure –prefix=/usr/local/mpich-3.1.3
$make -j4
$make install
```

Anexo: Instalación y prueba de rendimiento con Osu-Benchmark

Pasos para instalación de Osu-micro-benchmark para OpenMPI

```
$wgethttp://mvapich.cse.ohio-state.edu/download/mvapich/
osu-micro-benchmarks-5.3.2.tar.gz
$tar -xzvf osu-micro-benchmarks-5.3.2.tar.gz
$cd osu-micro-benchmarks-5.3.2
$.configure –prefix=/usr/local/osu-benchmark_openmpi
$CC=/usr/local/openmpi1.8.3/bin/mpicc
$make
$make install
```

Pasos para la instalación de Osu-micro-benchmark para MPICH

```
$wgethttp://mvapich.cse.ohio-state.edu/download/mvapich/
osu-micro-benchmarks-5.3.2.tar.gz
$tar -xzvf osu-micro-benchmarks-5.3.2.tar.gz
$cd osu-micro-benchmarks-5.3.2
$./configure –prefix=/usr/local/osu-benchmarks_mpich
$CC='/usr/local/mpich-3.1.3/bin/mpicc'
$make
$make install
```

Pasos para realizar pruebas de latencia en OpenMPI y MPICH

```
$cd /usr/local/osu_benchmark_openmpi/libexec/osu-micro-
benchmarks/mpi/pt2pt/
$/usr/local/openmpi1.8.3/bin/mpicc -f  /Documents/ -np 2 ./osu_latency
//Ejecución de prueba de ancho de banda
$/usr/local/openmpi1.8.3/bin/mpicc -f  /Documents/ -np 2 ./osu_bw
```

```
$cd /usr/local/osu_benchmark_mpichi/libexec/osu-micro-benchmarks/mpi/pt2pt/
$/usr/local/mpich-3.1.3/bin/mpicc -f  /Documents/ -np 2 ./osu_latency
//Ejecución de prueba de ancho de banda
$/usr/local/mpich-3.1.3/bin/mpicc -f  /Documents/ -np 2 ./osu_bw
```

Anexo: Instalación de ATLAS optimizado y sin optimizar

Instalación para ATLAS optimizado a la arquitectura

```
$Wgethttps://sourceforge.net/projects/math-atlas/files/Stable/
3.10.3/atlas3.10.3.tar.bz2/download
$tar -xvjf atlas3.10.3.tar.bz2
$cd atlas3.10.3.tar.bz2
$mkdir arm-a15
$CFLAGS=O3 -march=armv7-a -mcpu=cortex-a15 -mtune=cortex-a15 -
mfloat-abi=hard mfpu=vfpv3-d16"
$./configure -m 2320 -t 4 -Fa alg -mcpu=cortex-a15 -mtune=cortex-a15 -
mfloat-abi=hard mfpu=vfpv3-d16 –prefix=/usr/local/atlas
$make build
$make check
$Make ptcheck
```

Instalación para ATLAS no optimizado

```
$Wgethttps://sourceforge.net/projects/math-atlas/files/Stable/
3.10.3/atlas3.10.3.tar.bz2/download
$tar -xvjf atlas3.10.3.tar.bz2
$cd atlas3.10.3.tar.bz2
$mkdir arm-no-flag
$cd arm-no -flasg
$./configure -m –prefix=/usr/local/atlas
$make build
$make check
$make ptcheck
$make install
```