Implementación de un cluster HPC utilizando *single-board computers*

Jose Miguel Jaramillo Sánchez

Trabajo de grado para optar al título de Ingeniero Electrónico

Asesor

Sebastián Isaza Ramírez, PhD.

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería Electrónica

Medellín, Antioquia, Colombia

2025

Cita	(Jaramillo, J.M. 2025)
Referencia **Estilo APA 7 (2020)**	Jaramillo, J.M. (2025), *Implementación de un cluster HPC utilizando single-board computers*. Trabajo de grado profesional, Ingeniería Electrónica, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2025.

TABLA DE CONTENIDO

RESUMEN

Este trabajo presenta el diseño e implementación de un clúster de computación de alto rendimiento (HPC) utilizando como nodos tarjetas Jetson Nano, las cuales se caracterizan por su bajo consumo energético y la incorporación de una GPU integrada. El proyecto surge ante la necesidad de contar con infraestructuras HPC asequibles y escalables, especialmente en entornos educativos y de investigación, donde el presupuesto y el espacio suelen ser limitados.

La metodología abarcó cuatro fases principales: primero, se diseñó y prototipó un soporte en impresión 3D que facilita la fijación de las tarjetas en un rack estándar, optimizando el espacio. Luego, se desarrolló un sistema de distribución de energía modular capaz de alimentar múltiples Jetson Nano de manera estable. En la tercera etapa, se configuraron el sistema operativo, la red y los servicios esenciales (autenticación, almacenamiento compartido y gestor de colas SLURM) para coordinar la ejecución de tareas en paralelo a través de acceso remoto. Finalmente, se evaluó el rendimiento global del clúster mediante benchmarks especializados (como HPL y suites de la familia SPEC), analizando tanto el desempeño individual de cada nodo como la escalabilidad del sistema al aumentar la carga de trabajo.

Los resultados muestran que las Jetson Nano pueden sostener aplicaciones paralelas con un consumo reducido de energía, y que la arquitectura propuesta facilita la expansión futura del clúster. En conclusión, este trabajo sienta las bases para la adopción de soluciones HPC económicas y de bajo consumo, con un claro potencial para la formación académica y la investigación aplicada.

Palabras clave **— cluster HPC, Jetson Nano, programación paralela, computación distribuida, gestor de colas, eficiencia energética.**

I. INTRODUCCIÓN

En la última década, la necesidad de procesar grandes volúmenes de datos y resolver problemas computacionalmente intensivos ha impulsado el desarrollo de sistemas de computación de alto rendimiento (HPC, por sus siglas en inglés). Dichos sistemas resultan esenciales en áreas como la inteligencia artificial y la simulación para una amplia gama de aplicaciones, desde la investigación hasta su uso en la industria. Sin embargo, el costo y la complejidad de instalar y mantener clústeres HPC basados en hardware de servidor tradicional hacen que su implementación sea prohibitiva en muchos entornos académicos y empresas de menor escala. Por esta razón, han surgido propuestas alternativas que buscan ofrecer mayor accesibilidad y menor consumo energético, entre las cuales destaca el uso de computadoras de placa reducida (SBC), como la Jetson Nano.

El objetivo principal del presente trabajo es diseñar e implementar un clúster HPC a partir de tarjetas Jetson Nano, aprovechando su aceleración por GPU y bajo consumo energético para ejecutar y gestionar aplicaciones paralelas de manera eficiente. Con ello, se espera ofrecer una solución de bajo costo y alta eficiencia que facilite el acceso a la computación de alto rendimiento, a la vez que brinde la oportunidad de experimentar con infraestructura de clúster en entornos educativos o de investigación.

Desde la perspectiva del aporte a la ingeniería, esta propuesta contribuye al desarrollo de infraestructuras HPC más asequibles y sostenibles. Por un lado, se aprovecha la capacidad de cómputo en CPU y GPU de las Jetson Nano para abordar tareas relacionadas con la inteligencia artificial y el procesamiento científico. Por otro, la experiencia práctica de implementar un clúster real permite evaluar componentes clave como la distribución de energía, el montaje mecánico y la gestión de colas de trabajo, abarcando un espectro que va desde el diseño físico hasta la configuración de software especializado. Esta integración de aspectos técnicos ofrece un alto valor formativo, al facilitar que estudiantes e investigadores se familiaricen con el desarrollo completo de sistemas HPC.

Diversos trabajos previos han explorado la viabilidad de construir clústeres HPC utilizando computadoras de placa reducida, demostrando su aplicabilidad en tareas de paralelización y procesamiento distribuido. Estos proyectos evidencian que es posible emplear hardware de bajo costo para conformar clústeres funcionales en contextos educativos, experimentales e incluso productivos. Una revisión más detallada de estos antecedentes se presenta en el marco teórico.

Para llevar a cabo este trabajo, se adoptó una metodología dividida en cuatro grandes fases: (1) diseño y prototipado de un dispositivo de fijación para las Jetson Nano en un rack, (2) desarrollo de un sistema de distribución de energía eficiente y escalable, (3) configuración del software y un sistema de gestión de colas para ejecutar tareas en paralelo, y (4) evaluación del desempeño mediante benchmarks que explotan la aceleración por hardware y el cómputo en paralelo. Cada fase se describe detalladamente en los capítulos correspondientes, mostrando el flujo de trabajo seguido para el montaje físico, la instalación de software y la validación del rendimiento.

En términos de resultados generales, se logró la implementación de un clúster HPC conformado por varias tarjetas Jetson Nano, estableciendo un entorno de trabajo plenamente funcional para la ejecución de aplicaciones paralelas. Se verificó que el clúster es capaz de manejar cargas de trabajo que aprovechan la GPU integrada, y se obtuvo una caracterización de su comportamiento en cuanto a rendimiento, escalabilidad y consumo energético. Estas métricas, además de demostrar la viabilidad de la solución, servirán de base para trabajos futuros relacionados con la optimización de aplicaciones distribuidas, la integración de nodos adicionales y la exploración de otras áreas de computación paralela.

Con este proyecto, se sientan las bases para que, tanto en el ámbito académico como en el industrial, la experiencia de construir y operar un clúster HPC sea más accesible y sostenible. En los apartados siguientes, se describen con mayor detalle los conceptos teóricos, la metodología empleada y los resultados cuantitativos de la implementación.

II. OBJETIVOS

A. Objetivo general

Implementar una infraestructura tipo clúster, utilizando computadores de placa reducida aceleradas por GPU de referencia Jetson Nano, capaz de gestionar y ejecutar a través de acceso remoto, aplicaciones paralelizadas en todos sus nodos de manera eficiente y escalable.

B. Objetivos específicos

- Diseñar y prototipar un dispositivo de fijación para las tarjetas Jetson Nano utilizando software CAD e impresión 3D, que optimice el uso del espacio en el rack y facilite el montaje de las placas y los sistemas de conexión.

- Desarrollar y evaluar un sistema de distribución de energía modular y escalable, capaz de soportar múltiples tarjetas Jetson Nano, priorizando el alto rendimiento y la eficiencia energética para satisfacer la demanda eléctrica del clúster.

- Implementar un sistema de gestión de colas con el objetivo de facilitar la planificación y ejecución eficiente de tareas en paralelo, además de proporcionar mecanismos de acceso multiusuario al servidor.

- Evaluar el desempeño del clúster construido, usando programas de benchmarking que aprovechen al máximo las capacidades de cómputo en paralelo y aceleración por hardware disponibles en las tarjetas.

III. MARCO TEÓRICO

High-Performance Computing (HPC)

En los últimos años, el crecimiento acelerado de aplicaciones como la inteligencia artificial, el aprendizaje profundo, la analítica de grandes volúmenes de datos y la simulación científica ha generado una demanda sin precedentes de recursos computacionales. Estos dominios requieren ejecutar algoritmos cada vez más complejos y procesar conjuntos de datos masivos en tiempos reducidos, lo que ha impulsado una transformación en la manera en que se conciben las infraestructuras tecnológicas y ha aumentado significativamente la inversión en centros de datos, supercomputadores y plataformas HPC [1]. Esta evolución tecnológica también ha sido favorecida por la disponibilidad creciente de herramientas más eficientes y accesibles, que han facilitado su adopción por parte de organizaciones de todos los tamaños.

El concepto de High-Performance Computing (HPC) se refiere al uso de sistemas informáticos especializados capaces de ejecutar tareas altamente demandantes a velocidades considerablemente superiores a las de los equipos tradicionales. Estos sistemas están diseñados para procesar grandes volúmenes de datos y resolver problemas complejos mediante técnicas de paralelismo masivo y arquitecturas optimizadas que permiten dividir las tareas en múltiples subprocesos ejecutados de forma simultánea, reduciendo así los tiempos de cómputo de manera significativa [2].

Las plataformas HPC integran componentes de hardware de alto desempeño, tales como procesadores multinúcleo, memorias de alto ancho de banda y unidades de aceleración, fundamentales para ejecutar tareas altamente paralelizables. En este contexto, las arquitecturas heterogéneas, que combinan distintos tipos de procesadores como CPUs y GPUs, se han convertido en una piedra angular del diseño de sistemas HPC modernos. Una Unidad de Procesamiento Gráfico (GPU, por sus siglas en inglés) es un tipo de acelerador que incorpora una gran cantidad de pequeños núcleos de procesamiento, diseñados para ejecutar múltiples operaciones en paralelo de forma altamente eficiente. En sistemas HPC, las GPUs permiten acelerar tareas computacionalmente intensivas, mejorando significativamente el rendimiento y la eficiencia

energética en aplicaciones como simulaciones de dinámica molecular, física de partículas, predicción meteorológica o algoritmos de aprendizaje profundo [3] [4].

Además del procesamiento, los sistemas HPC se caracterizan por su capacidad para gestionar de forma eficiente el almacenamiento de datos, la comunicación entre procesos y la asignación dinámica de recursos en entornos distribuidos y complejos. Estas capacidades son clave para garantizar la escalabilidad, la estabilidad y el aprovechamiento óptimo de los recursos computacionales en escenarios de alta demanda.

Clústeres como infraestructura HPC

Una de las formas más comunes de implementar sistemas HPC es mediante clústeres de cómputo. Un clúster HPC es esencialmente un conjunto de múltiples computadoras interconectadas, denominadas nodos, que cooperan para ejecutar cargas de trabajo en paralelo, funcionando como un solo sistema lógico [2][5]. Esta configuración permite distribuir tareas entre varios nodos, lo que facilita el procesamiento simultáneo de grandes volúmenes de datos o la ejecución eficiente de algoritmos complejos. Debido a su escalabilidad y modularidad, los clústeres se han consolidado como una alternativa flexible y rentable frente a arquitecturas monolíticas tradicionales.

El propósito principal de un clúster es aumentar la capacidad de cómputo mediante la distribución de tareas entre varios nodos, lo que permite reducir considerablemente los tiempos de ejecución. Esta arquitectura facilita la resolución de problemas de gran escala que serían ineficientes o inviables de abordar en un solo equipo. Además, los clústeres presentan una alta escalabilidad, ya que su rendimiento puede incrementarse simplemente añadiendo más nodos al sistema, una ventaja clave frente a las arquitecturas monolíticas [6].

Desde el punto de vista estructural, un clúster HPC suele estar conformado por un nodo principal (también denominado nodo maestro o *head node*) y múltiples nodos de cómputo. El nodo principal se encarga de la administración central del sistema: alberga las herramientas de gestión de recursos, monitoreo, planificación de trabajos y las interfaces necesarias para la interacción con los

usuarios. En general, los usuarios acceden al clúster a través de este nodo para compilar, configurar y enviar sus tareas, las cuales son posteriormente distribuidas a los nodos de cómputo, encargados de ejecutar los cálculos intensivos en paralelo [7].

Clústeres: componentes de hardware

Los clústeres de computación de alto rendimiento (HPC) están compuestos por múltiples nodos, cada uno con su propio procesador (o procesadores), memoria RAM, almacenamiento y sistema operativo. Estos nodos se comunican entre sí mediante redes de alta velocidad, que proporcionan un alto ancho de banda y baja latencia para el intercambio eficiente de datos. De este modo, un clúster HPC combina la potencia de múltiples unidades de procesamiento y se comporta como un supercomputador distribuido capaz de resolver problemas complejos a gran escala [8] [9].

Físicamente, los nodos suelen organizarse en racks, lo que optimiza el espacio, facilita el cableado de red y energía, y mejora la ventilación [10]. Para garantizar un funcionamiento estable y continuo, especialmente en configuraciones de gran envergadura, se implementan soluciones de refrigeración activa como ventilación forzada, aire acondicionado de precisión o refrigeración líquida, según la densidad de cómputo. Asimismo, la infraestructura eléctrica integra fuentes de poder redundantes, sistemas de alimentación ininterrumpida (UPS) y protección contra sobrecarga. Estas medidas se vuelven críticas a medida que crece el número de nodos, ya que un clúster HPC puede consumir grandes cantidades de energía y generar un calor considerable. Por ello, resulta esencial evaluar tempranamente los requerimientos energéticos y térmicos del sistema para asegurar su eficiencia operativa y su sostenibilidad a largo plazo [9] [11].

Clústeres: componentes de software

A nivel de software, cada nodo de un clúster HPC ejecuta un sistema operativo, comúnmente basado en distribuciones Linux. Para coordinar el uso eficiente de los recursos disponibles, el clúster se apoya en un componente clave denominado gestor de colas (*job scheduler*). Este sistema organiza la ejecución de tareas, asignándolas automáticamente a los nodos

disponibles en función de criterios como la carga de trabajo actual, la disponibilidad de recursos y las prioridades definidas por el usuario o el administrador [12].

Además, los gestores de colas permiten gestionar dependencias entre trabajos, por ejemplo, asegurando que una tarea no se inicie hasta que otra haya finalizado, así como reprogramar tareas en caso de cambios en el entorno del sistema. Este tipo de planificación inteligente facilita la ejecución distribuida de aplicaciones complejas y garantiza un uso equilibrado de la infraestructura, lo que resulta fundamental para maximizar el rendimiento global del clúster.

Entre los gestores de colas más utilizados en entornos HPC se encuentran SLURM (Simple Linux Utility for Resource Management) y PBS (Portable Batch System). Ambos sistemas permiten a los usuarios enviar trabajos al clúster mediante scripts que describen los recursos necesarios, como el número de nodos, la duración estimada o la cantidad de memoria requerida. SLURM, de código abierto y altamente escalable, es especialmente popular en supercomputadores modernos debido a su flexibilidad, modularidad y amplia comunidad de desarrollo. Por su parte, PBS ha sido históricamente muy utilizado en instituciones académicas y centros de investigación, y cuenta con varias versiones, como PBS Professional y Torque, que extienden sus capacidades. Ambos gestores permiten definir políticas de uso, establecer prioridades y controlar el acceso a los recursos compartidos, lo que garantiza una operación ordenada y eficiente del clúster [12] [13] [14].

Además de planificar y administrar recursos, un clúster HPC requiere mecanismos que permitan la interacción entre procesos distribuidos. Dado que estos procesos suelen ejecutarse en nodos físicamente separados, se recurre a un *middleware* de comunicación: un software intermediario que facilita el intercambio de datos de forma estructurada y eficiente. En este contexto, la Message Passing Interface (MPI) se ha consolidado como el estándar de facto para este tipo de comunicación. MPI proporciona una interfaz escalable que permite dividir una tarea computacional en múltiples procesos independientes, los cuales se ejecutan en paralelo y se coordinan mediante el envío y recepción de mensajes. Implementaciones como OpenMPI y MPICH son ampliamente utilizadas en entornos HPC, ya que ofrecen mecanismos robustos y de bajo nivel que aseguran un rendimiento óptimo en la ejecución distribuida de aplicaciones [15].

En combinación con MPI, muchos clústeres también emplean modelos de paralelismo de memoria compartida como OpenMP, que permiten dividir tareas en múltiples hilos dentro de un mismo nodo. Esta estrategia híbrida, que combina paralelismo entre nodos (MPI) y dentro de los nodos (OpenMP), mejora el aprovechamiento de los recursos en arquitecturas multinúcleo y multiprocesador [16].

Para facilitar la administración de usuarios en sistemas con múltiples nodos, se integran soluciones de autenticación centralizadas como LDAP (Lightweight Directory Access Protocol), que permiten gestionar credenciales, permisos y políticas de acceso de manera unificada en todo el clúster. Esto resulta clave para mantener la seguridad y la consistencia operativa del entorno [17].

Asimismo, dado que muchas aplicaciones científicas generan y procesan grandes volúmenes de datos, los clústeres suelen incorporar sistemas de almacenamiento compartido como NAS (Network-Attached Storage). Este tipo de infraestructura permite a todos los nodos acceder a un repositorio común de archivos, lo cual es esencial para mantener la coherencia del entorno de ejecución, facilitar la colaboración y evitar redundancias [18].

En conjunto, estos componentes de software permiten que un clúster HPC funcione como un sistema coherente, capaz de ejecutar aplicaciones complejas de forma paralela, eficiente y escalable. La correcta configuración e integración de estos elementos resulta esencial para aprovechar al máximo la capacidad computacional disponible.

Uso de Single-Board Computers (SBC) en HPC

Debido al elevado costo de adquirir, configurar y mantener un clúster HPC con sistemas de cómputo tradicionales [19], una alternativa más económica que ha cobrado interés en el ámbito educativo es el uso de single-board computers (SBC) o computadoras de placa reducida como nodos de un clúster.

Una SBC es una computadora completa integrada en una sola placa de circuito, que incluye procesador, memoria, almacenamiento y puertos de entrada/salida básicos en un formato muy compacto [20]. Ejemplos comunes de SBC son la Raspberry Pi, BeagleBone, Khadas VIM3, NVIDIA Jetson, entre otros. Estas placas, originalmente diseñadas para educación, *makers* o aplicaciones embebidas, tienen varias ventajas: son muy económicas, de bajo consumo energético, pequeñas en tamaño y suficientemente versátiles al ejecutar sistemas operativos Linux completos. Sin embargo, también presentan limitaciones respecto a hardware de clase servidor, como menor potencia de cómputo, menos memoria, y menor robustez o confiabilidad a largo plazo. Aun así, su relación costo-beneficio las hace atractivas para construir clústeres experimentales a una fracción del costo de un clúster tradicional.

Emplear SBCs permite montar un clúster funcional con un presupuesto muy reducido, aprovechando la alta eficiencia energética de estos dispositivos. Por ejemplo, una Raspberry Pi modelo 4 consume del orden de 5-7 W bajo carga, comparado con los cientos de vatios de un servidor típico. Esto habilita la creación de clústeres “verdes” y portátiles. Un solo nodo SBC tiene capacidad limitada, pero al agrupar decenas de ellos es posible lograr un desempeño conjunto apreciable. Abrahamsson et al. reportan la construcción de un clúster de 300 Raspberry Pis denominado Bolzano Raspberry Pi Cloud [21], destacando que cada Pi es muy barata y de bajo consumo, lo que hace viable armar un clúster asequible y energéticamente eficiente; naturalmente, la contraparte es su baja potencia individual, que se compensa en parte con la cantidad de nodos trabajando en paralelo. Este experimento sirvió como banco de pruebas de bajo costo y bajo consumo para investigación en computación en la nube, demostrando que es posible un *testbed* HPC económico y “verde” útil para probar software y arquitecturas distribuidas [21].

De igual forma, Tso et al. desarrollaron el proyecto Glasgow Raspberry Pi Cloud (PiCloud) con 56 Raspberry Pis montadas en racks de LEGO, recreando a pequeña escala la infraestructura de un centro de datos en la nube. El PiCloud emula todas las capas de la pila de *cloud computing* (virtualización, redes, almacenamiento, aplicaciones) y provee un entorno completo para investigación y enseñanza en computación en nube y sistemas distribuidos [22]. Este proyecto evidenció que con SBC es posible construir una plataforma realista para experimentación

académica en escalamiento de centros de datos, por una fracción del costo y espacio requeridos por hardware empresarial.

Adicionalmente, un estudio desarrollado en el Departamento de Ingeniería Electrónica y Telecomunicaciones de la Universidad de Antioquia presentó una plataforma de clúster embebido para un laboratorio remoto de programación paralela, en la que se utilizaron placas Orange Pi 4 y Khadas VIM3 como nodos de cómputo. Este trabajo demostró la viabilidad de emplear SBCs como una solución de bajo costo y alta eficiencia energética en ambientes de aprendizaje en línea [23].

En el ámbito educativo, los clústeres de SBC se han vuelto populares como herramientas didácticas para introducir conceptos fundamentales de HPC y computación paralela. Estas plataformas permiten a estudiantes e investigadores aprender sobre planificación de trabajos, balanceo de carga, MPI y programación paralela, así como experimentar con la configuración completa de un clúster (sistemas operativos, redes, bibliotecas), todo ello sin necesidad de una infraestructura costosa.

Aunque el rendimiento de un clúster de SBC está lejos de igualar al de un supercomputador, por ejemplo, una Raspberry Pi 3 alcanza apenas entre 0.05 y 0.1 GFLOPS (giga-FLOPS, es decir, 10^9 operaciones en punto flotante por segundo) en pruebas Linpack [24], resulta suficiente para ejecutar aplicaciones paralelas de pequeña escala y demostrar principios fundamentales de HPC. Además, su bajo consumo energético en comparación con sistemas tradicionales las hace convenientes cuando el presupuesto y la eficiencia importan más que la velocidad. Por ello, estas plataformas se consolidan como entornos ideales para la enseñanza, el prototipado y la investigación exploratoria en computación paralela [25].

En resumen, los clústeres construidos con SBC ofrecen una alternativa accesible y funcional para explorar tecnologías HPC. Si bien no compiten en rendimiento con sistemas de producción, su bajo costo, eficiencia energética y facilidad de implementación los convierten en plataformas valiosas para la enseñanza, la investigación y el desarrollo de prototipos a pequeña escala.

La SBC Jetson Nano

La Jetson Nano, desarrollada por NVIDIA, es una SBC orientada a aplicaciones de inteligencia artificial, visión por computador y computación en el borde (*edge computing*). La arquitectura de su System-on-Chip principal Tegra X1, la compone una CPU ARM Cortex-A57 de cuatro núcleos y una GPU NVIDIA Maxwell con 128 núcleos CUDA. Esta configuración, junto con 4 GB de memoria LPDDR4 y soporte para almacenamiento microSD o eMMC, permite alcanzar hasta 472 GFLOPS en precisión simple (FP16), un rendimiento adecuado para cargas paralelas ligeras como inferencia de redes neuronales o procesamiento de imágenes [26].

Con modos de operación de 5 W y 10 W, y conectividad mediante Ethernet Gigabit, la Jetson Nano combina un bajo consumo energético con una velocidad de red adecuada para integrarse en configuraciones de clúster. Su compatibilidad con el entorno JetPack, que incluye Ubuntu para ARM, controladores GPU, CUDA Toolkit y bibliotecas como cuDNN y TensorRT, facilita significativamente la configuración de entornos HPC con aceleración por GPU.

A diferencia de otras SBC como la Raspberry Pi, la Jetson Nano incorpora una GPU CUDA totalmente funcional, lo que permite construir clústeres heterogéneos CPU+GPU para experimentos en paralelismo, balanceo de cargas y computación distribuida. Esta capacidad, sumada a su bajo costo y eficiencia energética, la convierte en una opción ideal para proyectos educativos, prototipado e investigación en HPC de bajo presupuesto [27].

Por sus capacidades previamente descritas y su disponibilidad en el Laboratorio de Electrónica Digital del Departamento de Ingeniería Electrónica, la Jetson Nano fue seleccionada como nodo base para el clúster experimental desarrollado en este trabajo.

Antecedentes Prácticos para la Construcción de Clústeres con Jetson Nano

Diversos proyectos recientes han demostrado la viabilidad técnica y educativa de construir clústeres HPC a pequeña escala utilizando placas NVIDIA Jetson Nano como nodos de cómputo. Estas iniciativas han puesto en práctica fundamentos de computación distribuida y paralelismo heterogéneo, y evidencian que, pese a sus limitaciones, las Jetson Nano pueden integrarse exitosamente en arquitecturas HPC funcionales.

Por ejemplo, Michael Riedl documenta la construcción de un clúster experimental basado exclusivamente en Jetson Nano, configurado con SLURM como gestor de colas, OpenMPI para comunicación entre procesos y el entorno CUDA habilitado para ejecutar tareas aceleradas por GPU. Este clúster fue utilizado para realizar pruebas de rendimiento y ejecutar modelos paralelos de aprendizaje automático, mostrando un desempeño coherente con su arquitectura y un consumo energético muy bajo [28].

De forma complementaria, Shahizat presenta un clúster heterogéneo conformado por dos Jetson Xavier NX, una Jetson Nano y un nodo maestro basado en Raspberry Pi 4, con almacenamiento compartido proporcionado por una Orange Pi 5 Plus. Esta configuración replica la arquitectura típica de un clúster HPC: nodo principal para planificación, nodos de cómputo distribuidos, red dedicada y sistema de archivos común mediante NFS. El software incluye Ubuntu 20.04, SLURM, OpenMPI y CUDA, con medidas de sincronización horaria y autenticación por claves SSH para una operación fluida. Este caso destaca la posibilidad de escalar y combinar diferentes SBCs para crear entornos HPC realistas con recursos limitados [29].

Ambos antecedentes respaldan el enfoque adoptado en este trabajo, al proporcionar referentes técnicos sobre el uso de Jetson Nano en configuraciones de clúster. Más allá de confirmar su viabilidad, el valor de este proyecto radica en analizar qué herramientas y configuraciones son necesarias para su implementación, qué limitaciones presenta y cómo se comporta el sistema frente a cargas de trabajo reales. Asimismo, estos proyectos, junto con la documentación oficial de las herramientas utilizadas, sirvieron como base técnica para orientar el desarrollo e integración del clúster propuesto.

Evaluar el rendimiento de un clúster

La evaluación del rendimiento de un clúster HPC es fundamental para determinar la efectividad de la configuración implementada y su capacidad para ejecutar cargas de trabajo paralelas [30]. Para ello, se aplican pruebas conocidas como benchmarks, que pueden clasificarse en dos grandes categorías: benchmarks sintéticos y benchmarks basados en aplicaciones reales.

Los benchmarks sintéticos, como el High Performance Linpack (HPL), se utilizan ampliamente para estimar el rendimiento computacional teórico de un clúster, midiendo cuántas operaciones en punto flotante por segundo (FLOPS) puede realizar. HPL resuelve un sistema de ecuaciones lineales mediante factorización LU con pivoteo parcial, y su rendimiento se expresa típicamente en GFLOPS. Este benchmark es el estándar utilizado para el ranking TOP500 de supercomputadoras, ya que permite comparar arquitecturas heterogéneas bajo una métrica homogénea de cálculo intensivo [31]. Además, HPL resulta particularmente útil para analizar la escalabilidad del sistema, ya que permite variar tanto el tamaño del problema como el número de nodos involucrados, observando el comportamiento del rendimiento conforme crece la complejidad computacional y la comunicación entre procesos.

Por otro lado, los benchmarks orientados a aplicaciones reales, como los desarrollados por la Standard Performance Evaluation Corporation (SPEC), ofrecen una visión más representativa del desempeño del sistema en contextos prácticos. En particular, suites como SPEC OMP2012 (para cargas paralelas con OpenMP), SPEC ACCEL (para cómputo acelerado con OpenCL y OpenACC) y SPEC MPI2007 (para aplicaciones paralelas distribuidas usando MPI) permiten evaluar el rendimiento del clúster bajo diferentes modelos de paralelismo. Estas pruebas no solo consideran la capacidad de cómputo, sino también el uso eficiente de la memoria, la latencia de comunicación, el acceso a datos y la interacción entre procesos. En conjunto, proporcionan una evaluación integral del sistema en condiciones cercanas al uso real en entornos HPC [32].

Entre las métricas más utilizadas para evaluar el rendimiento de un clúster destacan el *speedup*, la eficiencia y la escalabilidad. Estas permiten cuantificar el impacto del paralelismo al incrementar el número de nodos o procesadores.

• **Speedup (aceleración):** representa la mejora en el tiempo de ejecución al ejecutar un programa en paralelo respecto a su ejecución secuencial. Se define como:

$$S(n)=\frac{T_1}{T_n} \qquad (1)$$

Donde:

T_1 es el tiempo de ejecución secuencial,

T_n es el tiempo usando n procesadores.

• **Eficiencia:** mide qué tan bien se utilizan los recursos disponibles. Se calcula como:

$$E(n)=\frac{S(n)}{n}=\frac{T_1}{(nT_n)} \qquad (2)$$

Donde:

T_1 es el tiempo de ejecución secuencial,

T_n es el tiempo usando n procesadores.

n es el número de procesadores

Una eficiencia del 100 % indica que todos los procesadores contribuyen completamente al rendimiento.

• **Ley de Amdahl: e**stablece que el rendimiento máximo (*speedup*) de un sistema paralelo está limitado por la parte del programa que no puede ser paralelizada. Se expresa mediante la fórmula:

$$S(n)=\frac{1}{\left((1-P)+\frac{P}{n}\right)} \qquad (3)$$

donde *P* es la fracción paralelizable, y *n* el número de procesadores disponibles.

Esta ley demuestra que, incluso con un número infinito de procesadores, el rendimiento está restringido por la porción secuencial del código. Así, proporciona una herramienta útil para estimar el potencial de mejora antes de implementar una solución paralela.

En clústeres construidos con plataformas como la Jetson Nano, donde los recursos son limitados y la arquitectura difiere de los sistemas HPC tradicionales, la evaluación del rendimiento cumple un papel crucial para determinar las capacidades reales del sistema. Las métricas como el *speedup* y la eficiencia permiten cuantificar el impacto del paralelismo y el comportamiento del clúster frente a distintas cargas de trabajo. La aplicación de benchmarks, ya sean sintéticos como HPL o suites basadas en aplicaciones reales como SPEC, facilita la identificación de limitaciones en la comunicación, el procesamiento o la distribución de tareas, proporcionando así una base técnica para validar el diseño y orientar futuras mejoras. En este tipo de entornos, donde se priorizan factores como la eficiencia energética, la modularidad y el bajo costo, contar con herramientas de evaluación bien definidas es esencial para caracterizar el sistema y explorar su potencial en contextos educativos o de investigación aplicada.

IV. METODOLOGÍA

Este capítulo describe la metodología adoptada para diseñar, construir y poner en marcha el clúster de Jetson Nano. Primero, se aborda la parte mecánica, centrada en el diseño y fabricación de un sistema de sujeción impreso en 3D que aprovecha eficazmente el espacio dentro del rack. Luego, se presenta la componente eléctrica, enfocada en el desarrollo de un sistema de distribución de energía modular y eficiente. A continuación, se explica la configuración del sistema operativo, la red y los mecanismos de acceso remoto, que permiten una gestión centralizada y segura. Posteriormente, se detalla la implementación del software de gestión de colas mediante SLURM, encargado de distribuir las tareas entre los nodos y coordinar su ejecución en paralelo. Finalmente, se describe el proceso de evaluación del rendimiento del clúster, empleando herramientas de benchmarking para medir su capacidad de cómputo y escalabilidad.

A. COMPONENTE MECÁNICA

La Jetson Nano es una SBC diseñada principalmente para aplicaciones de computación en el borde y el despliegue de modelos simples de inteligencia artificial, destacándose por su bajo consumo energético. No obstante, esta tarjeta no está orientada específicamente a integrarse en clústeres HPC, ni dispone de un sistema de montaje nativo compatible con racks comerciales de servidores. Por esta razón, se hizo necesario diseñar un mecanismo de fijación específico que facilitara la organización eficiente y segura de las tarjetas dentro de un rack estándar.

Uno de los primeros aspectos considerados fue el lugar donde se implementaría el clúster. El uso de un rack comercial facilita la logística del montaje, ya que proporciona una estructura sólida basada en medidas estandarizadas, lo que permite adaptar o diseñar un mecanismo de sujeción adecuado para las tarjetas. En este contexto, se decidió utilizar un rack disponible en el Departamento de Ingeniería Electrónica de la Facultad de Ingeniería, el cual ya había sido empleado en proyectos anteriores con clústeres basados en SBC. Esto requirió ajustar el diseño del soporte para las tarjetas Jetson Nano a las especificaciones y estructura ya existentes en dicho rack.

Si bien existen algunas opciones comerciales en línea para la sujeción de las tarjetas en racks estándar, estas son limitadas y no necesariamente compatibles con los requerimientos específicos del proyecto. Generalmente, estos mecanismos se presentan en dos configuraciones: una disposición horizontal que ocupa 1U de rack y una disposición vertical que ocupa 3U. En la primera, las tarjetas se alinean una al lado de la otra con el disipador orientado hacia arriba, lo que reduce el espacio vertical utilizado, pero limita la capacidad a un máximo de cuatro tarjetas por módulo. En la segunda, las tarjetas se montan verticalmente con el disipador orientado hacia un costado del rack, permitiendo alojar hasta 12 tarjetas en un espacio de 3U (ver **Fig. 1**). Esta última configuración permite concentrar en un solo módulo la misma cantidad de tarjetas que requeriría tres módulos de 1U, lo que reduce la cantidad de piezas necesarias para alcanzar una capacidad equivalente.

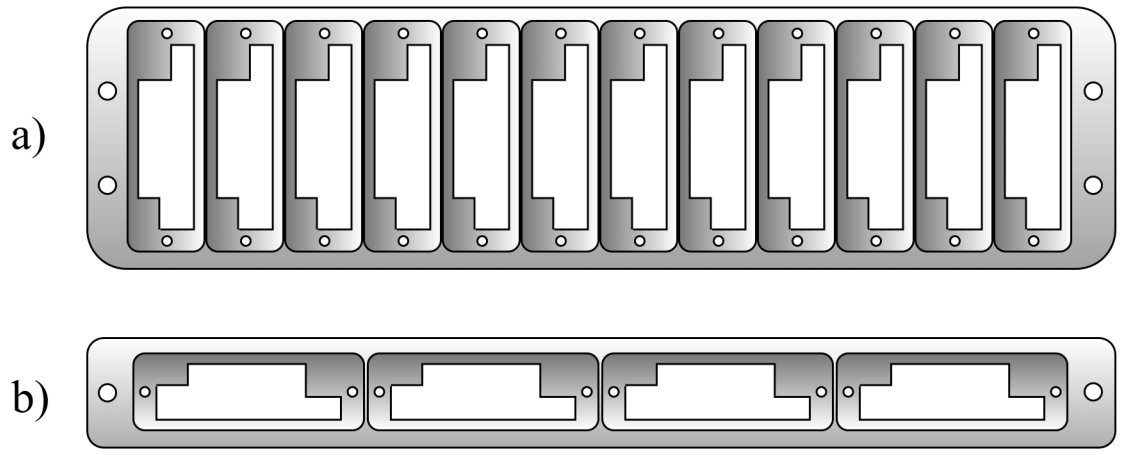


Fig. 1. Disposición de soporte en rack para Jetson Nano 3U (a) y 1U (b).

A pesar de que existen sistemas de soporte comerciales para el montaje de estas tarjetas en racks, su disponibilidad en el mercado local es limitada, por lo que se optó por diseñar y fabricar un sistema propio. En este proceso, la impresión 3D se consideró como una alternativa económica, que además ofrecía un mayor control y capacidad de personalización. Aunque existen algunos modelos 3D disponibles en internet, muchos no son compatibles con el sistema de rieles del rack, y replicar directamente los diseños comerciales de 1U y 3U mediante impresión presenta un desafío importante: por lo general, estos sistemas están compuestos por un panel principal que cubre todo el ancho del rack y por pequeños soportes que fijan las tarjetas a dicho panel (ver **Fig. 1**). Debido al

gran tamaño de esta pieza principal, su fabricación requeriría de una impresora 3D con unas dimensiones mayores a las disponibles, lo que llevó a explorar alternativas que simplificaran el diseño de este sistema de sujeción.

Una alternativa propuesta en proyectos anteriores fue un sistema de sujeción que no dependía completamente de un panel impreso en 3D. Esta solución consistió en utilizar un conjunto de tubos dispuestos a lo ancho del rack, fijados a los rieles mediante un par de piezas impresas aseguradas con tornillos. Dos de estos tubos, de color rojo (**Fig. 2**), sostendrían en disposición vertical los soportes impresos en 3D donde se acoplan las tarjetas, mientras que un tercer tubo, de color azul, serviría para organizar los cables de alimentación. Este enfoque, además de eliminar la necesidad de imprimir un panel principal, simplificó el proceso de diseño al reducir la complejidad general del sistema de sujeción.

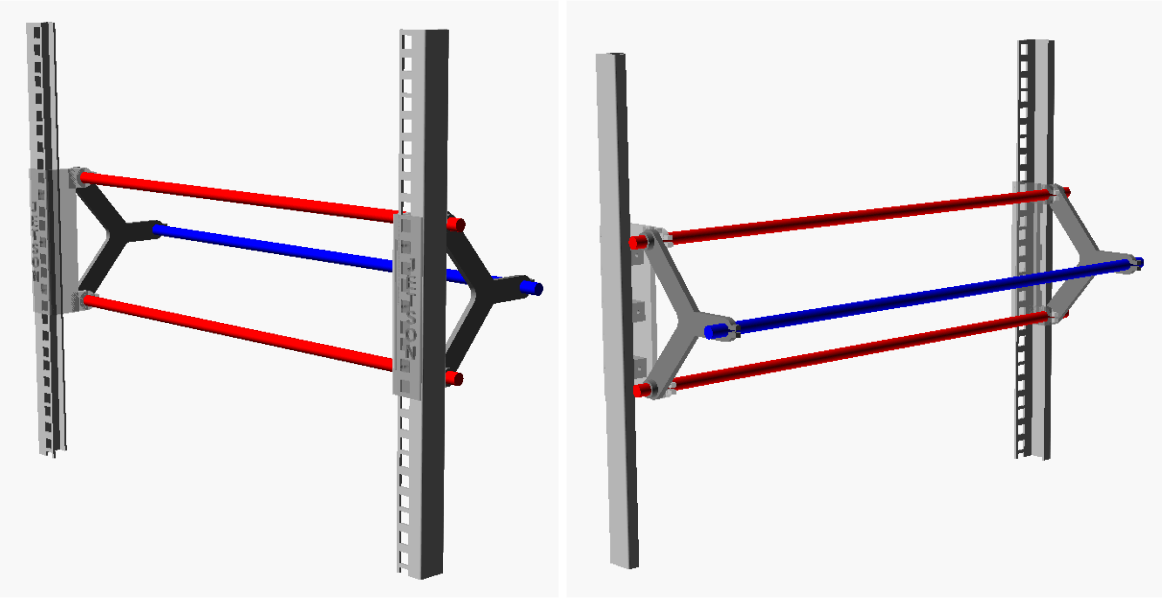


Fig. 2. Vista frontal y trasera del sistema de sujeción propuesto.

El sistema original fue diseñado para VIM3 y Orange Pi4, con una distancia entre tubos y soportes determinada por el tipo de placa. Por ello, fue necesario realizar ajustes para adaptarlo a la Jetson Nano, modificando tanto la separación entre los tubos como el diseño del soporte. Con estas modificaciones, el sistema de sujeción ocuparía 3U en el rack y permitiría alojar hasta 10 tarjetas a lo ancho.

Para modelar las piezas del soporte de la Jetson Nano, se evaluaron varias herramientas CAD de código abierto, buscando una solución adecuada para usuarios con poca experiencia en diseño mecánico. Las opciones consideradas fueron FreeCAD, Blender y OpenSCAD.

FreeCAD permite el modelado paramétrico con gran precisión y ofrece funciones avanzadas para diseño mecánico y ensamblaje; no obstante, su interfaz puede resultar compleja, lo que deriva en una curva de aprendizaje elevada. Blender, por su parte, es muy versátil y potente para modelado 3D, pero al estar orientado a animación y diseño gráfico, incluye numerosas herramientas innecesarias para el propósito principal de este proyecto. En contraste, OpenSCAD emplea un enfoque basado en código, haciendo uso de primitivas geométricas y operaciones booleanas. Su interfaz minimalista facilita el aprendizaje para quienes se inician en el diseño, ya que no presenta un exceso de menús o botones.

Tras comparar las tres opciones, se seleccionó OpenSCAD por ser una alternativa sencilla e intuitiva. Aunque el uso exclusivo de código es poco convencional en diseño mecánico, la experiencia previa en programación facilitó adoptar su lógica basada en funciones y operaciones geométricas. Además, esta aproximación estructurada y la facilidad de aprendizaje contribuyeron a agilizar el proceso de diseño, permitiendo realizar modificaciones rápidas y precisas incluso sin contar con amplia experiencia en el ámbito mecánico.

Una vez seleccionado el software de diseño, se tomaron las medidas de la Jetson Nano, se identificaron los orificios de montaje y se definió un bosquejo inicial del soporte y su método de fijación. Para asegurar precisión en el modelado y simplificar su desarrollo, se importó en OpenSCAD un modelo STL oficial de la tarjeta, disponible en el sitio web de NVIDIA, lo que permitió ajustar el diseño directamente a sus dimensiones reales.

Primero, se diseñó la parte frontal del soporte con un grosor de 1.5 mm, añadiendo aberturas para todos los puertos y dos adicionales para facilitar la extracción del soporte del sistema de sujeción (ver **Fig. 3**). Con el fin de mejorar la resistencia sin aumentar el espesor, se incorporaron nervios estructurales que refuerzan la pieza y reducen el consumo de material. Además, se ajustó la longitud a 3U y el ancho para garantizar una separación adecuada entre las tarjetas, evitando roces.

Posteriormente, se incluyeron dos pestañas cuadriláteras alineadas con los primeros dos orificios de la Jetson Nano para fijar la tarjeta al soporte. Debido a la orientación vertical del sistema de sujeción, la tensión mecánica se distribuye eficientemente a través de estos dos puntos, lo que permite prescindir de los cuatro orificios sin comprometer la estabilidad de la tarjeta.

Dado que la Jetson Nano posee soldaduras en su parte inferior, no fue posible montar la tarjeta directamente sobre las pestañas de sujeción. Para resolverlo, se diseñaron dos bujes que elevan la tarjeta, evitando así la interferencia con el soporte. Estos bujes se imprimieron por separado, ya que, en la orientación de impresión del soporte completo (con el panel frontal apoyado sobre la cama), su estructura resultaba frágil y ofrecía menor resistencia mecánica. La tarjeta se fijó al soporte mediante tornillos M2.5x10 mm y tuercas.

Con la tarjeta ya fijada, se modeló el mecanismo de acoplamiento a los tubos del sistema de sujeción del rack. Para ello, se diseñó una abrazadera tipo gancho, capaz de rodear el tubo y permitir su inserción o extracción mediante una ligera flexión. Inicialmente, se planeó imprimir todo el soporte en ASA, debido a su alta resistencia mecánica y estabilidad frente a condiciones ambientales. No obstante, la abrazadera requería un material más elástico para soportar flexiones repetidas sin deteriorarse, por lo que se optó por TPU exclusivamente para esta pieza, la cual fue diseñada e impresa por separado.

Finalmente, se incorporaron dos ranuras de acoplamiento en el soporte para fijar estas piezas flexibles, permitiendo su encaje a presión sin requerir elementos adicionales.

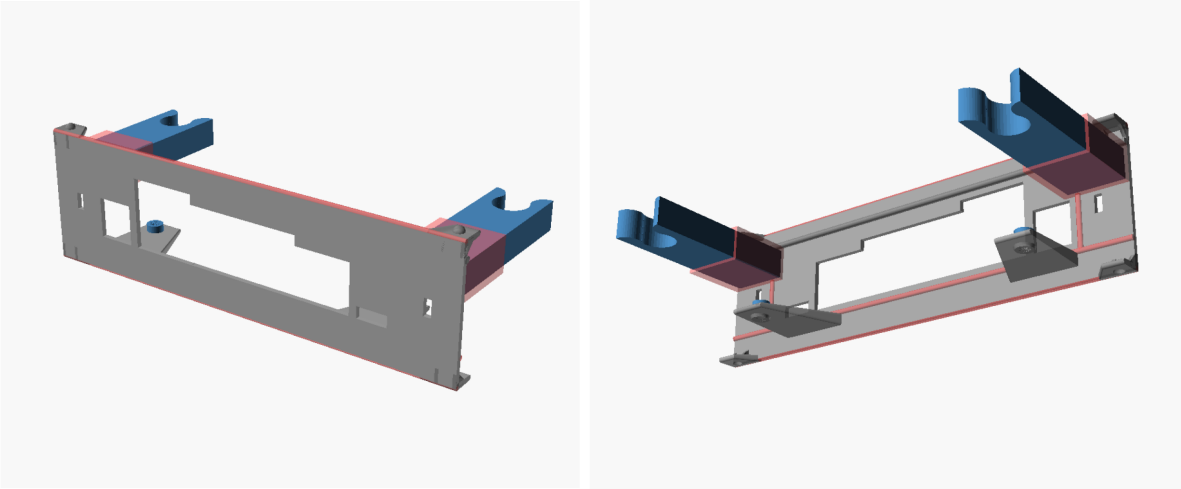


Fig. 3. Vista frontal y trasera del soporte 3D diseñado.

La **Fig. 4** muestra una simulación del montaje final del sistema, donde se observa la disposición completa de los soportes diseñados acoplados a los tubos del sistema de sujeción.

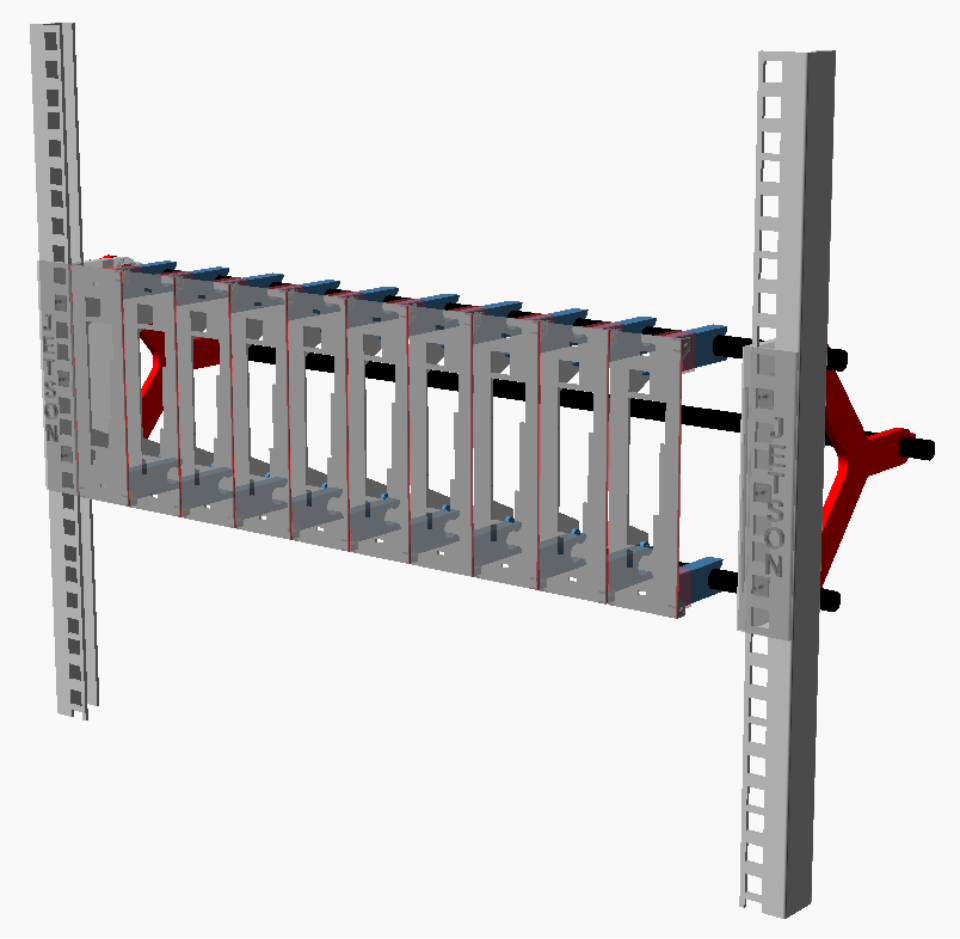


Fig. 4. Simulación 3D del montaje final con los soportes y el sistema de sujeción.

B. COMPONENTE ELÉCTRICA

La Jetson Nano es una tarjeta que opera principalmente a 5V de alimentación, admitiendo un rango de voltaje entre 4.75V y 5.25V. La corriente máxima que soporta depende del método utilizado para alimentarla, lo que influye directamente en su capacidad para operar en diferentes modos de energía.

En términos de consumo, la Jetson Nano cuenta con dos modos principales: uno de 5W y otro de 10W. El modo de 5W limita automáticamente el rendimiento para reducir el consumo de energía y mantener bajas temperaturas, siendo particularmente adecuado para aplicaciones que no requieren un elevado rendimiento computacional. En cambio, el modo de 10W permite un uso intensivo de los núcleos de CPU y GPU, incrementando significativamente la capacidad de procesamiento para tareas exigentes, aunque con un mayor consumo energético. Este último modo es especialmente útil para la ejecución de algoritmos complejos o benchmarks destinados a evaluar el rendimiento del clúster. Para alternar entre estos perfiles energéticos, basta con utilizar los comandos proporcionados por NVIDIA en la Jetson Nano o emplear herramientas nativas de monitorización de recursos como jtop (**Anexo I**).

Esta tarjeta puede ser alimentada mediante tres métodos principales: el conector micro-USB (J28), el jack barrel (J25) y los dos pines de 5V y GND en el cabezal J41 (ver **Fig. 5**). La opción del micro-USB es útil en etapas de prototipado, pero está limitada a aproximadamente 2.5A, lo que puede ser insuficiente para operar en modo de 10 W si consideramos el uso de periféricos externos. El conector barrel (J25), en cambio, ofrece mayor estabilidad en la entrega de corriente y permite alcanzar hasta 4 A, lo que posibilita el funcionamiento en modo de alto rendimiento. Sin embargo, su ubicación y el tipo de cableado necesario pueden hacerlo menos práctico en configuraciones de un clúster con múltiples tarjetas. Para habilitar la alimentación a través de J25, es necesario puentear los pines J48. Finalmente, la alimentación a través de los pines J41 permite alcanzar hasta 3A por pin (6A en total, si se usan ambos), con la ventaja de ofrecer un método de alimentación más personalizable y limpio que puede facilitar el diseño de una línea de distribución de energía.

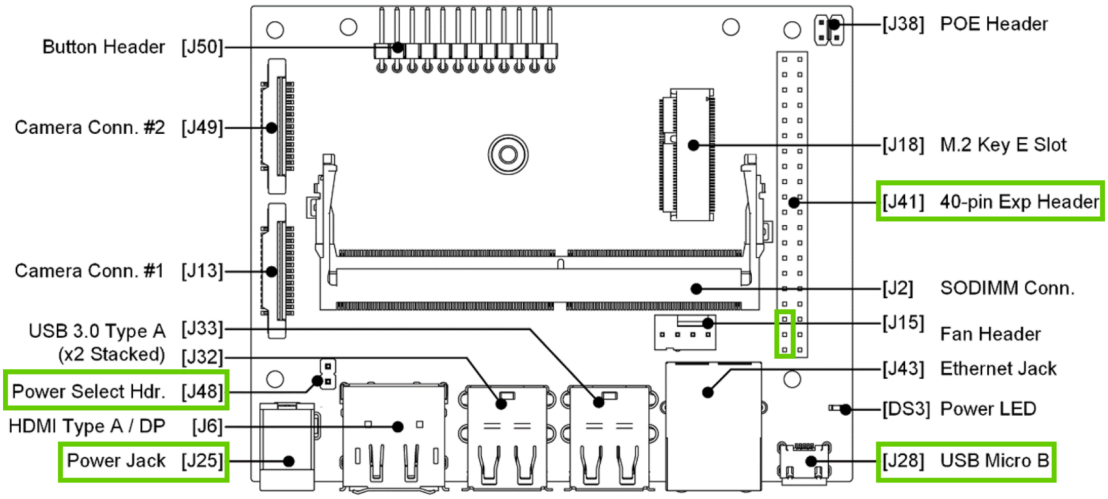


Fig. 5. Métodos de alimentación de la Jetson Nano.

Nota: fuente https://developer.nvidia.com/embedded/dlc/jetson_nano_developer_kit_user_guide

Durante la primera etapa de pruebas del clúster, cada Jetson Nano se alimentaba mediante el adaptador oficial de NVIDIA con conector jack barrel (J25), lo que requería el uso de múltiples regletas y adaptadores. Aunque esta solución fue funcional al inicio, pronto resultó ineficiente debido al excesivo uso de espacio en el rack, el desorden del cableado y las dificultades en el mantenimiento. Con el aumento en el número de tarjetas, se hizo evidente la falta de escalabilidad de este sistema, lo que llevó a la necesidad de diseñar una alternativa más eficiente y organizada.

Para abordar esta limitación, se propuso la implementación de una única línea de alimentación capaz de abastecer a todas las placas, permitiendo reducir el cableado individual, optimizar el uso del espacio y simplificar la instalación y el mantenimiento. Este sistema de distribución debía garantizar un suministro estable de potencia, evitando caídas de tensión que pudieran afectar el rendimiento de los dispositivos o comprometer la estabilidad de la red eléctrica.

Para llevar a cabo la implementación, se diseñó una línea principal de 5V desde la cual se realizaron derivaciones individuales hacia cada Jetson Nano. Esta línea debía estar alimentada por una o varias fuentes capaces de suministrar la potencia requerida, estimada en al menos 100W para diez tarjetas operando en modo de 10W, más un margen de seguridad del 15% al 20% para absorber picos de arranque y permitir futuras expansiones. Tras analizar los diferentes métodos de alimentación, se optó por utilizar los pines de 5V y GND del cabezal J41, principalmente por

razones de orden y eficiencia. A diferencia del jack barrel, que se encuentra en la parte frontal de la tarjeta y cuyo cableado podría invadir el espacio de los slots contiguos en el rack, los pines J41 permiten una distribución más compacta y organizada, facilitando el tendido de cables y proporcionando un montaje más limpio.

Una vez definidos los requerimientos de potencia y el método de alimentación, era necesario evaluar la fuente de energía. Se buscaba contar con un sistema centralizado de distribución, capaz de suministrar el voltaje adecuado y una potencia superior a la requerida, garantizando estabilidad y permitiendo futuras expansiones. También era fundamental que este sistema incluyera protecciones de seguridad contra sobrecorrientes, sobrevoltajes, cortocircuitos y sobrecargas, evitando daños en las tarjetas y otros componentes. Se optó por una fuente ATX debido a su compatibilidad con una amplia gama de voltajes, su variedad de conectores y la flexibilidad de sus puertos modulares, lo que facilita la integración con distintos dispositivos. Además, las fuentes ATX modernas incorporan mecanismos de protección avanzados, garantizando una alimentación segura y eficiente para el clúster.

En el rack se disponía de dos fuentes de alimentación ATX de alta potencia, previamente utilizadas en proyectos de clústeres con SBC. Estas fuentes, de referencia EVGA SuperNOVA 1200 P2, pueden suministrar hasta 1200W de potencia total, distribuida según se muestra en la **TABLA I**. Además, cuentan con certificación 80 PLUS Platinum, lo que garantiza una eficiencia de hasta 92 %, reduciendo pérdidas de energía y permitiendo un mejor aprovechamiento del consumo eléctrico. Para mayor seguridad y estabilidad, integran protecciones avanzadas contra sobrecorriente, sobrevoltaje, sobrecarga y cortocircuito, asegurando un suministro confiable para los dispositivos conectados.

TABLA I
ESPECIFICACIONES ELÉCTRICAS DE CADA FUENTE DE ALIMENTACIÓN EVGA 1200P.

AC Entrada	100 - 240 VAC, 15A, 50-60 Hz				
DC Salida	+3.3V	+5V	+12V	+5Vsb	-12V
MAX Salida	20A	20A	99.9A	2.5A	0.5A
	100W		1198.8W	12.5W	6W
Potencia de Salida	1200W @ +50C				

De acuerdo con las necesidades de potencia establecidas para las tarjetas, se evaluaron dos alternativas: utilizar la línea de 5V o la de 12V. Aunque la línea de 12V ofrece la mayor capacidad de potencia de la fuente, su uso requería convertidores DC-DC para reducir el voltaje a los 5V necesarios para las Jetson Nano, lo que implicaría costes adicionales. Por otro lado, la línea de 5V, aunque limitada a 100W por fuente (**TABLA I**, celdas sombreadas), permitía alimentar directamente las tarjetas sin necesidad de conversión, lo que simplificaba la distribución de energía y el diseño del sistema de alimentación, convirtiéndola en una buena opción para este proyecto. Para garantizar un suministro equilibrado y mantener un margen de seguridad, las tarjetas se dividieron en dos grupos de cinco, asignando cada grupo a una fuente distinta y evitando así ocupar por completo los 100 W disponibles en una sola fuente.

Debido a la modularidad de este tipo de fuentes ATX, existen múltiples opciones desde donde obtener la línea de 5 V, con varios conectores ofreciendo distintos voltajes simultáneamente (ver **Fig. 6**). Tras revisar los diferentes conectores disponibles, se determinó que el conector PERIF utiliza únicamente cuatro de sus seis pines, proporcionando líneas de 5V, 12V y dos puntos de tierra. Al ser un conector con menos cables, simplifica la elaboración de la línea de distribución. Con el objetivo de evitar la sobrecarga de un solo puerto, cada grupo de cinco Jetson Nano asignadas a cada fuente se dividieron entre los dos conectores PERIF disponibles: dos tarjetas conectadas a uno de ellos, y tres al otro. De esta manera, se garantiza una distribución equilibrada de la corriente, se disminuye el riesgo de sobrecalentamiento y se mejora la estabilidad general del sistema.

Fig. 6. Puertos disponibles en la fuente de alimentación EVGA 1200P.

Nota: fuente https://latam.evga.com/products/product.aspx?pn=220-P2-1200-X1

Establecidos todos los criterios de diseño, se realizó la construcción del sistema de distribución. Se definieron un par de líneas principales de 5V y GND para cada conector PERIF en cable de calibre 16 AWG, capaces de transportar la corriente requerida. A lo largo de ellas, se soldaron derivaciones en cable 22 AWG para cada Jetson Nano, colocando en el extremo de dichas derivaciones conectores JST-XH de 2.54 mm, compatibles con los pines J41 de la tarjeta (ver **Fig. 7**). Luego los cables se recubrieron con mangueras organizadoras para mantener el orden. Se realizó una línea de distribución por fuente, dejando un excedente de cable en cada una para futuras ampliaciones.

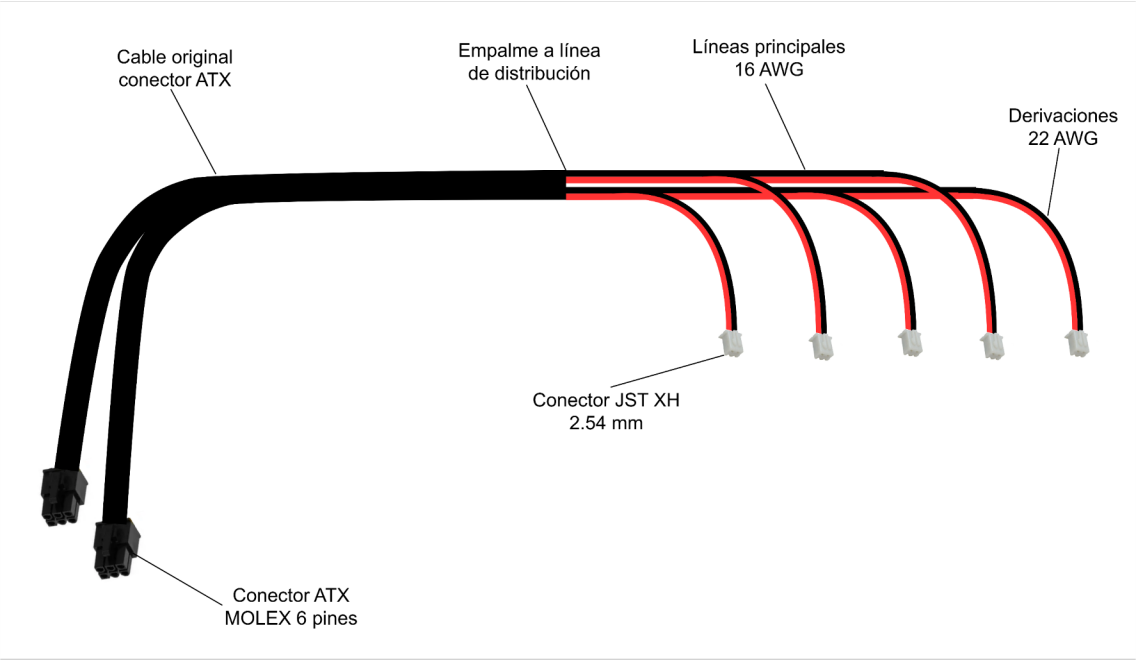


Fig. 7. Línea de distribución para una sola fuente.

Tras cablear las dos fuentes y sus correspondientes líneas, se midió el voltaje en cada derivación con un multímetro para verificar polaridad y continuidad. Primero se arrancaron una o dos Jetson Nano por línea para comprobar estabilidad, sin observar caídas de tensión significativas. Una vez verificado el correcto funcionamiento, se conectaron las restantes placas, alcanzando el total de diez sin presentar sobrecargas en los cables ni en los puertos de la fuente. El cableado se fijó con amarres plásticos en el rack, dejando suficiente holgura para eventuales ampliaciones y asegurando un montaje ordenado.

Finalmente, se realizó un análisis del presupuesto de potencia para garantizar que el consumo total de los dispositivos conectados al rack se mantuviera dentro de los límites de la infraestructura disponible. En la **Fig. 8** se presenta el diagrama eléctrico, que refleja la distribución de energía tras la incorporación de las Jetson Nano.

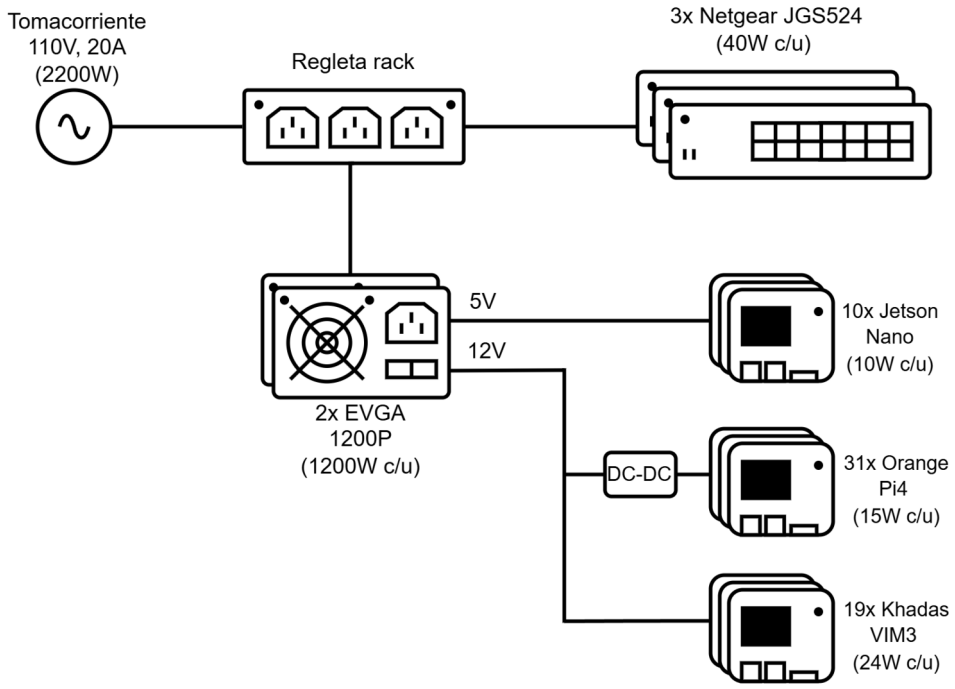


Fig. 8. Diagrama eléctrico de los dispositivos conectados en el rack.

El rack recibe alimentación a través de una regleta para rack, conectada al circuito eléctrico del laboratorio, el cual proporciona una corriente máxima de 20A. En esta regleta están conectadas las dos fuentes EVGA SuperNOVA 1200P2 y tres switches NetGear JGS524, cada uno con un consumo máximo de 40W. Además de la línea de 5V utilizada para alimentar las Jetson Nano, la línea de 12V de las fuentes se emplea para suministrar energía a 31 tarjetas Orange Pi 4 (15W cada una) mediante conversores DC-DC, y a 19 placas Khadas VIM3 (24W cada una), que se alimentan directamente.

Para cuantificar la demanda de energía del sistema, se realizó un cálculo del consumo máximo estimado de cada componente (ver **TABLA II**).

TABLA II
ESTIMACIÓN DE CONSUMO TOTAL EN EL RACK.

Dispositivo	**Cantidad**	**Consumo por unidad (W)**	**Consumo total (W)**
Switches NetGear JGS524	3	40	120
Orange Pi 4	31	15	465
Khadas VIM3	19	24	456
Jetson Nano	10	10	100
Total estimado	-	-	1141 W

Dado que el rack se conecta a un tomacorriente convencional de la red eléctrica del laboratorio cuya capacidad máxima es de 2200 W (considerando una tensión de 110 V y una corriente máxima de 20 A), se verificó que el consumo total del rack (1141 W) se encuentra dentro de los límites operativos. Esto deja un margen de seguridad de aproximadamente 1059 W, lo que permite la posibilidad de futuras expansiones sin comprometer la estabilidad del sistema.

Con base en estos cálculos, se determinó que el diseño del sistema eléctrico es viable y seguro, garantizando una distribución de energía estable y confiable.

C. COMPONENTE DE SOFTWARE

SISTEMA OPERATIVO

La configuración inicial de las tarjetas Jetson Nano se realizó siguiendo la guía oficial de NVIDIA, la cual detalla el proceso de instalación del sistema operativo y la preparación del entorno de desarrollo. NVIDIA proporciona una imagen de disco denominada JetPack, diseñada específicamente para estas tarjetas, cuya versión compatible más reciente es JetPack 4.6, basada en Ubuntu 18.04. Esta versión incluye herramientas fundamentales para el desarrollo de aplicaciones de inteligencia artificial, como CUDA, TensorFlow, cuDNN y TensorRT.

No obstante, con el objetivo de agilizar la configuración de múltiples tarjetas y contar con un sistema operativo más actualizado, se decidió utilizar una imagen personalizada previamente configurada y disponible en un repositorio público de GitHub. Esta imagen, basada en Ubuntu 20.04, ofrece compatibilidad con el software típico de JetPack y, además, incorpora herramientas adicionales para la monitorización de recursos y la gestión remota.

La imagen fue grabada en las tarjetas MicroSD utilizando herramientas como BalenaEtcher o Raspberry Pi Imager. Posteriormente, se realizaron algunas configuraciones, como expandir la partición del sistema operativo para maximizar el uso del almacenamiento disponible y la configuración de red necesaria para establecer una comunicación eficiente entre los nodos de cómputo y el nodo controlador.

Dado que el proyecto involucra la gestión de múltiples nodos en un entorno de clúster sin interfaz gráfica, la configuración de las Jetson Nano se realizó en modo *headless*, es decir, sin necesidad de conectar un monitor, teclado o mouse. Para ello, se estableció conexión con la tarjeta mediante protocolo SSH, inicialmente a través de la interfaz microUSB y, una vez completada la configuración básica, a través de la interfaz de red.

Los detalles técnicos, comandos utilizados y configuraciones avanzadas se describen en los anexos de este informe (**Anexo A**).

CONFIGURACIÓN DE RED

Una de las componentes más importantes en la implementación de cualquier clúster de cómputo es la infraestructura de red y la conectividad entre los dispositivos. Para garantizar un rendimiento óptimo, es fundamental disponer de una red eficiente que facilite la comunicación entre un nodo principal y los nodos de cómputo, permitiendo la distribución y ejecución de tareas de forma coordinada.

Una arquitectura de red básica para un clúster basado en SBCs debe incluir los siguientes elementos (ver **Fig. 9**):

• **Nodo principal**: se encarga de gestionar los recursos del clúster y coordinar la ejecución de los trabajos. Puede ser una Jetson Nano o un equipo de cómputo diferente.

• **Nodos de cómputo**: ejecutan las tareas enviadas desde el nodo principal y pueden comunicarse entre sí cuando es necesario, como en aplicaciones de procesamiento paralelo. En este caso, todos los nodos de cómputo son Jetson Nano.

• **Sistema de almacenamiento en red (NAS)**: centraliza el acceso a los datos para todos los nodos, evitando la duplicación de información y facilitando la ejecución de trabajos distribuidos. Puede implementarse mediante NFS o, en algunos casos, CIFS/SMB.

• **Dispositivo de interconexión**: un switch Ethernet que establece la subred del clúster, asegurando una comunicación rápida y estable entre todos los nodos. Un dispositivo con velocidades Gigabit es la opción más recomendable.

• **Conectividad a Internet:** permite mantener los sistemas actualizados y sincronizados, acceder a recursos en la web, así como habilitar el acceso remoto al clúster. Puede implementarse mediante un router que conecte la subred del clúster a Internet o mediante NAT y reglas de ruteo en el nodo principal, siempre que este cuente con múltiples interfaces de red y actúe como gateway.

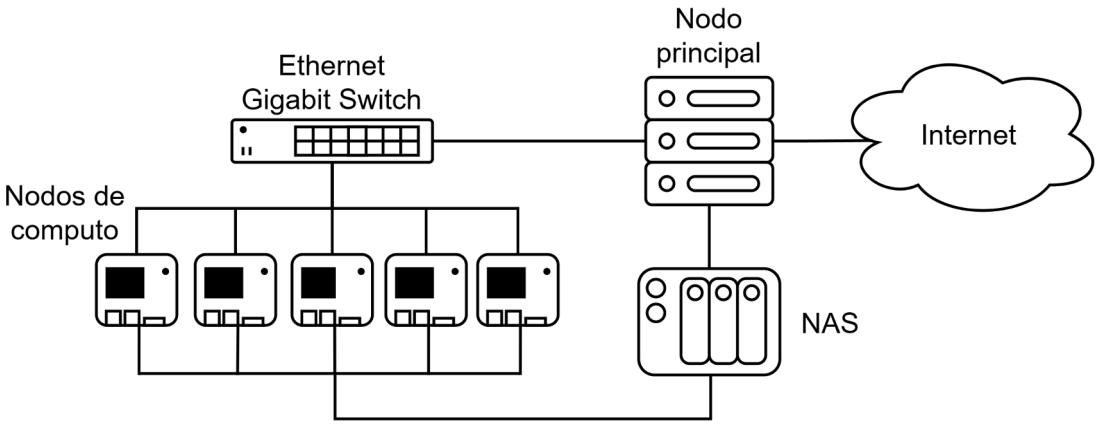


Fig. 9. Arquitectura de red básica para la implementación de clúster con SBCs.

El Departamento de Ingeniería Electrónica dispone de varios servidores y equipos de cómputo para investigación y docencia, algunos de los cuales pueden ser accedidos de forma remota a través de Internet. En este caso, se seleccionó un servidor del grupo GITA como nodo principal del clúster. Dicho servidor cumple la función de controlador, gestionando la ejecución de

trabajos, distribuyendo los recursos entre los nodos de cómputo y actuando como punto de acceso para los usuarios, desde donde pueden enviar tareas y monitorear remotamente su ejecución.

La gestión centralizada de la autenticación y los directorios de usuario se realiza mediante el protocolo LDAP, utilizando un servidor OpenLDAP previamente configurado. Para integrar las tarjetas Jetson Nano al sistema, solo fue necesario instalar los clientes LDAP, replicar la configuración del nodo principal y agregar los certificados SSL para garantizar una conexión segura. Esto permite un acceso unificado a los recursos del clúster, asegurando que los usuarios puedan iniciar sesión y acceder a sus archivos de forma consistente desde cualquier nodo conectado a la red.

Los nodos de cómputo (Jetson Nano) están conectados a este servidor a través de tres switches Netgear JGS524 de 24 puertos, los cuales están interconectados entre sí. Esta configuración posibilita una subred con hasta 69 conexiones, proporcionando una velocidad máxima de 1000 Mbps. Como se mencionó en la sección anterior, el rack incluye 50 tarjetas de proyectos anteriores, dejando 19 puertos libres, de los cuales se usaron 10 para las nuevas Jetson Nano y 9 permanecen disponibles para futuras ampliaciones.

El acceso a Internet de los nodos de cómputo se realiza a través del nodo principal, configurado como gateway mediante NAT para permitir la comunicación con redes externas.

Finalmente, se emplea un sistema NAS ya disponible, configurado como una partición exportada mediante NFS desde otra máquina. En este NAS se crean los directorios *home* de los usuarios, asegurando que sean accesibles desde todos los nodos del clúster. Cabe destacar que el servicio debe ser compatible con la versión NFSv3, ya que algunas implementaciones en las Jetson Nano presentan limitaciones al operar con versiones más recientes del protocolo.

La interfaz de red del servidor utilizada para conectarse con la subred del rack de SBCs opera con la subred 192.168.100.0/24. Las Jetson debieron configurarse en este rango de IP respetando las asignaciones existentes para las tarjetas anteriormente mencionadas, así como aquellas reservadas para otros dispositivos dentro de la red. De igual manera, para garantizar la

resolución de las direcciones de los nodos dentro del clúster, se asignó un *hostname* único e identificable a cada nodo, lo que simplifica su administración y configuración. Se configuró el acceso SSH sin contraseña entre el nodo principal y los nodos de cómputo, evitando así la necesidad de autenticación manual. Con estas consideraciones, se realizaron las siguientes asignaciones (ver **TABLA III**).

TABLA III
DIRECCIONAMIENTO DE DISPOSITIVOS EN LA RED.

Dispositivo	**Dirección IP**	**Hostnames**
Nodo principal	192.168.100.1	broly
NAS	192.168.100.2	serverhome
Nodos (Jetson Nano)	192.168.100.30-39	node1 - node9

Por políticas de seguridad, la universidad restringe el acceso a algunos servicios externos. El servidor NTP público de Ubuntu no es accesible dentro de su red, y el acceso a servidores DNS públicos es limitado. Para solucionar esto, se configuraron en las Jetson Nano los servidores internos proporcionados por la universidad para las funciones de NTP y DNS. La sincronización horaria es crucial para el clúster, ya que garantiza que todos los nodos operen en tiempos coordinados.

Se realizaron pruebas de conectividad entre los nodos para verificar el correcto funcionamiento de la infraestructura de red del clúster. Como resultado, se obtuvo una arquitectura heterogénea, en la que el nodo principal utiliza una arquitectura x86_64, mientras que los nodos de cómputo están basados en arquitectura ARM (aarch64).

Los detalles completos sobre la configuración de red y el sistema NAS en las Jetson Nano se encuentran documentados en los **Anexos B, C y D**.

GESTOR DE COLAS

Una vez establecida la conectividad entre los nodos del clúster, se procedió a evaluar el software necesario para la gestión y distribución de tareas de cómputo. Para ello, se requiere un gestor de colas, encargado de administrar los recursos disponibles y distribuir los trabajos de manera eficiente entre los nodos de procesamiento. En los servidores de la universidad se han utilizado previamente dos opciones principales: PBS y SLURM, ambos ampliamente adoptados en entornos de computación de alto rendimiento.

Si bien PBS ha sido utilizado tradicionalmente en los sistemas del departamento, en los últimos años se ha iniciado un proceso de migración hacia SLURM debido a su mayor flexibilidad, escalabilidad y mejor integración con tecnologías modernas como PMIx y OpenMPI. SLURM también proporciona una gestión de recursos más dinámica, con una comunidad de soporte activa. Dado que esta transición se está llevando a cabo en la infraestructura institucional, el trabajo se centró en la implementación y configuración de SLURM como gestor de colas en el clúster de SBCs, garantizando compatibilidad con los sistemas actuales y facilitando futuras integraciones.

Una de las ventajas clave de SLURM en la infraestructura del clúster es su capacidad para gestionar distintos tipos de hardware mediante el uso de particiones. Esto permite organizar los recursos disponibles de forma eficiente, separando, por ejemplo, nodos con GPUs tradicionales y nodos constituidos por Jetson Nano (ver **Fig. 10**). De este modo, los usuarios pueden especificar en qué partición desean ejecutar sus trabajos, asegurando que la carga de cómputo se asigne al hardware más adecuado según sus necesidades.

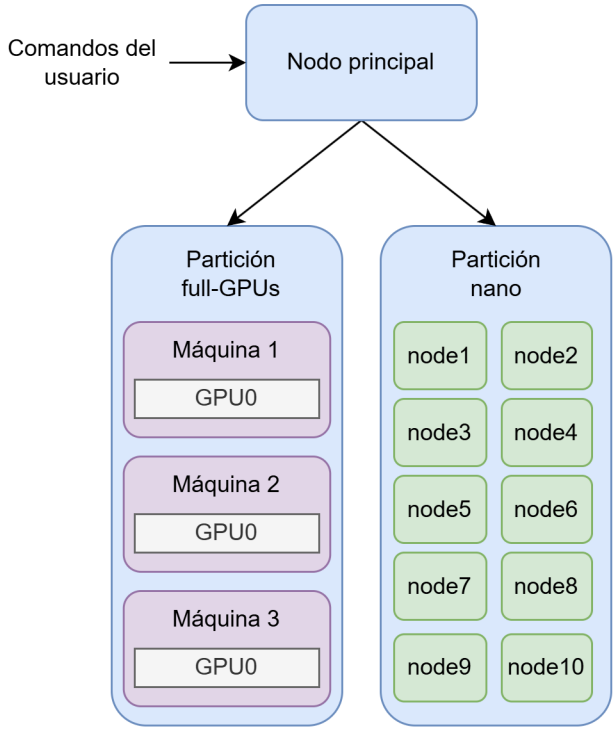


Fig. 10. Ejemplo de particiones definidas para separar las máquinas con GPU de los nodos Jetson Nano.

Para la implementación de SLURM en el clúster, fue necesario instalar previamente ciertas dependencias fundamentales. Entre ellas se encuentran Munge, para la autenticación entre nodos, y PMIx, que facilita la integración con OpenMPI.

Dado que SLURM requiere autenticación entre nodos para la gestión de tareas, se configuró Munge como mecanismo de seguridad. Este sistema permite la autenticación basada en claves sin necesidad de intercambiar contraseñas, lo cual simplifica la administración y mejora la seguridad. Su instalación se realizó desde el código fuente, compilándolo tanto en el nodo maestro como en los nodos de cómputo, asegurando compatibilidad con la arquitectura heterogénea del clúster. Para garantizar su correcto funcionamiento, se generó un archivo *munge.key* en el nodo maestro y se replicó en todos los nodos, de modo que cada uno pudiera validar sus conexiones sin inconsistencias. Cualquier diferencia en este archivo impediría la autenticación y ejecución de trabajos.

Para habilitar la ejecución eficiente de aplicaciones paralelas mediante MPI, se utilizó PMIx como plugin intermediario entre SLURM y OpenMPI. PMIx optimiza la comunicación entre procesos MPI, reduce la sobrecarga en la asignación de recursos y mejora la escalabilidad del sistema. Su instalación también se realizó desde la fuente, clonando una versión específica del repositorio oficial. Con el fin de agilizar el despliegue en todos los nodos, se creó un script automatizado, lo cual permitió una instalación recursiva y estandarizada en el clúster.

Una vez instalados Munge y PMIx, se procedió con la instalación de SLURM. Siguiendo un método similar al de las dependencias anteriores, se compiló el código fuente para garantizar compatibilidad con las distintas arquitecturas del clúster. La instalación se llevó a cabo en una carpeta específica, facilitando su empaquetado y distribución en los demás nodos Jetson Nano, lo que redujo significativamente el tiempo de despliegue.

Un elemento crítico en la configuración de SLURM es el archivo *slurm.conf*, donde se define la estructura del clúster, especificando los nodos, la asignación de recursos y las reglas de gestión de trabajos. Su correcta configuración asegura una sincronización eficiente entre todos los nodos y permite una distribución adecuada de la carga de trabajo. Este archivo puede generarse utilizando herramientas disponibles en la documentación de SLURM y en su sitio web. Para esta implementación, se realizaron ajustes menores, como la configuración del *hostname* del nodo controlador y la definición de una partición específica para los diez nodos Jetson Nano, denominada *nano*. Este archivo debe estar presente en todos los nodos y mantenerse idéntico en cada uno para garantizar la coherencia en la configuración del clúster.

En este punto, el sistema ya permitía lanzar trabajos en la partición *nano*, verificar y modificar el estado de los nodos, revisar la cola de tareas y realizar otras operaciones de administración. La interacción con SLURM se realizó principalmente de dos formas: mediante la línea de comandos, utilizando herramientas como *srun* para ejecutar trabajos en tiempo real, y a través de archivos *batch*, que permiten automatizar múltiples tareas y gestionar de manera más eficiente el uso de los recursos del clúster.

Un archivo *batch* es un script en el que se especifican los recursos necesarios para la ejecución de un trabajo, como la cantidad de nodos, el número de tareas o el tiempo de ejecución, así como los comandos requeridos para ejecutar uno o varios procesos en el clúster. Estos archivos se envían a SLURM mediante el comando *sbatch*, lo que posibilita la ejecución programada de tareas sin intervención manual. Este enfoque facilita la administración de cargas de trabajo y la ejecución eficiente de múltiples procesos en paralelo.

Para habilitar la ejecución de aplicaciones paralelas en el clúster, se instaló OpenMPI, una implementación de MPI ampliamente utilizada en entornos de computación distribuida. Su elección se debe a su compatibilidad con SLURM, su capacidad de adaptación a arquitecturas heterogéneas y su flexibilidad en el manejo de interfaces de red. Su integración con PMIx optimiza la gestión de procesos, reduciendo la latencia en la comunicación y mejorando la eficiencia en la ejecución de programas distribuidos. La instalación se realizó mediante compilación desde la fuente y fue empaquetada para facilitar su distribución.

Una vez finalizada la instalación y configuración del sistema, se realizaron pruebas básicas para verificar el correcto funcionamiento de SLURM, la ejecución de trabajos en la partición nano y la comunicación entre nodos mediante MPI. Estas pruebas incluyeron la ejecución de comandos simples en todos los nodos, trabajos de prueba con Python y un programa "Hola Mundo" utilizando MPI.

Para evaluar el desempeño del clúster en la ejecución de tareas paralelas, se llevó a cabo una prueba más elaborada, ejecutando un programa de cálculo mediante MPI cuyo objetivo era determinar la cantidad de primos hasta un cierto número. Este problema es paralelizable a nivel de datos, lo que significa que no requiere comunicación explícita entre nodos para obtener el resultado final, ya que cada uno se encarga de procesar una parte específica del cálculo. Se varió el número de nodos en diferentes iteraciones y se calcularon *speedup* y eficiencia para analizar la distribución de carga y el impacto en el rendimiento del clúster.

Durante la ejecución de trabajos con MPI, se presentó un problema en la identificación automática de la interfaz de red en los nodos, lo que impedía la correcta comunicación entre procesos distribuidos. Para solucionarlo, fue necesario especificarla manualmente en el script

batch, garantizando que OpenMPI gestionara correctamente la comunicación entre procesos y evitando errores en la ejecución de los trabajos.

Las instrucciones detalladas sobre la instalación y configuración de SLURM, Munge, PMIx y OpenMPI, así como los procedimientos para su despliegue en el clúster, la configuración del archivo slurm.conf, la gestión de nodos y la ejecución de trabajos, se encuentran documentadas en los **Anexos E, F, G, H y J**.

D. EVALUACIÓN DE DESEMPEÑO

Para evaluar con precisión el rendimiento y la escalabilidad del clúster de tarjetas Jetson Nano, se llevaron a cabo pruebas utilizando distintas herramientas de benchmarking**.** Estas pruebas permitieron medir tanto la capacidad de cómputo individual de cada nodo como el desempeño global del clúster cuando todos los nodos trabajaban en conjunto. Además de verificar el correcto funcionamiento de la infraestructura, se obtuvieron métricas cuantitativas sobre el uso de los recursos de cómputo y los tiempos de ejecución en tareas computacionalmente intensivas.

El monitoreo detallado de estos recursos se realizó con *tegrastats*, una herramienta nativa de las plataformas Jetson de NVIDIA (ver **Anexo O**). Esta utilidad se ejecutó en segundo plano junto con cada benchmark, registrando en tiempo real el uso de CPU y GPU, la memoria utilizada, la temperatura de los componentes y el consumo energético. Los datos recolectados fueron almacenados en archivos de texto y posteriormente procesados mediante scripts en Python, convirtiéndolos a formato CSV para su posterior análisis y generación de gráficas comparativas.

Se seleccionaron cuatro benchmarks ampliamente utilizados en evaluaciones de rendimiento computacional y paralelismo: SPEC OMP2012, SPEC ACCEL, SPEC MPI2007 y HPL. Las dos primeras pruebas evaluaron de forma aislada el desempeño de la CPU y la GPU de un solo nodo. Las otras dos pruebas permitieron analizar el desempeño global del clúster, ejecutando aplicaciones paralelizadas con MPI, que simulan cargas de trabajo reales.

Las suites de SPEC fueron obtenidas tras una solicitud de uso no comercial a la organización a través de su sitio web oficial, mientras que el benchmark HPL está disponible para descarga libre. Para garantizar el máximo rendimiento posible en todas las pruebas, se configuraron los nodos en modo de 10W y fijando la CPU a su máxima frecuencia (ver **Anexo I**).

Todas las pruebas de SPEC se ejecutaron en modo base, sin optimizaciones de compilación adicionales y con una única iteración por test. Los resultados de estas suites se expresan en relación con una máquina de referencia definida por SPEC, utilizada para establecer una comparación estandarizada del rendimiento de cada sistema evaluado. HPL, en cambio, mide el desempeño del sistema en FLOPs, proporcionando un valor absoluto de rendimiento en operaciones de punto flotante.

Cada suite de SPEC incluye dos conjuntos de datos estándar para la ejecución de pruebas: *test* y *ref*. El dataset *test* es más ligero y se utiliza principalmente para verificar la compatibilidad y el correcto funcionamiento de la suite en el sistema. Por otro lado, el dataset *ref* es más exigente y está diseñado para evaluar el rendimiento real del sistema bajo cargas de trabajo computacionalmente intensivas. Dado que algunas pruebas podrían no ejecutarse correctamente, se siguió un proceso de depuración en varias fases:

1. **Ejecución preliminar**: se ejecutaron todas las pruebas con el dataset *test* para identificar cuáles compilaban y corrían correctamente.
2. **Selección de pruebas viables**: aquellas pruebas que se ejecutaron sin errores con *test*, se ejecutaron nuevamente con el dataset *ref*.
3. **Descartes finales**: las pruebas que no lograron ejecutarse exitosamente con *ref* fueron descartadas.
4. **Ejecución final**: se realizó una última ejecución de las pruebas exitosas, midiendo el desempeño del sistema con *tegrastats* y registrando la información en archivos de log.

BENCHMARKS PARA UN NODO INDIVIDUAL

SPEC OMP2012

El primer conjunto de pruebas incluyó la suite SPEC OMP2012, diseñada para evaluar el rendimiento de la CPU en cargas de trabajo que utilizan programación paralela a nivel de hilos mediante OpenMP. Estas pruebas permitieron analizar el desempeño de los cuatro núcleos del procesador de la Jetson Nano.

La máquina de referencia utilizada para la comparación es un Sun Fire X4140, equipado con dos procesadores AMD Opteron 2384 de cuatro núcleos a 2.7 GHz y 32 GB de RAM. Esta referencia permite contextualizar los resultados obtenidos por la Jetson Nano dentro de un marco de evaluación estandarizado. Sin embargo, el propósito de estas pruebas no es competir en rendimiento absoluto, sino analizar el comportamiento de la Jetson Nano en tareas paralelizables a nivel de nodo, considerando su arquitectura ARM y su orientación a sistemas de bajo consumo energético.

La instalación de SPEC OMP2012 en la Jetson Nano resultó sencilla, ya que el toolset proporcionado incluye soporte para esta arquitectura, eliminando la necesidad de compilar el código fuente (ver **Anexo K**).

Para ejecutar correctamente la suite, fue necesario habilitar memoria SWAP adicional, ya que algunos datasets superaban la capacidad de RAM integrada en la Jetson Nano. Según la documentación oficial de SPEC, el conjunto completo de pruebas requiere al menos 28 GB de memoria disponible, muy por encima de los 4 GB que ofrece la Jetson Nano. Para mitigar esta limitación y evitar errores por falta de memoria, incluso en las pruebas menos exigentes, se configuró un SWAP adicional de 10 GB. Sin embargo, debido a la alta demanda de ciertos datasets, es probable que algunas pruebas no se ejecutaran correctamente o presentaran una degradación significativa en el rendimiento.

La **TABLA IV** presenta las pruebas que conforman la suite en su totalidad. Cada una representa una simulación o modelo de aplicación científica intensiva, con distintos patrones de acceso a memoria y niveles de paralelización.

TABLA IV
PRUEBAS DISPONIBLES EN LA SUITE SPEC OMP2012.

Test	Lenguaje	Descripción
350.md	C	Simulación de dinámica molecular con interacción Lennard-Jones.
351.bwaves	Fortran	Simulación de propagación de ondas en fluidos.
352.nab	C	Simulación de plegamiento de proteínas y bioinformática.
357.bt331	Fortran	Simulación de flujos compresibles tridimensionales (Benchmark NASA BT).
358.botsalgn	C	Alineamiento de secuencias genéticas utilizando BWA.
359.botsspar	C	Factorización esparsa en matrices grandes.
360.ilbdc	Fortran	Modelo de transporte de contaminantes atmosféricos.
362.fma3d	Fortran	Análisis de elementos finitos en modelos estructurales.
363.swim	Fortran	Modelo de transporte de calor en fluidos.
367.imagick	C	Procesamiento de imágenes con el software ImageMagick.
370.mgrid	Fortran	Solución de ecuaciones de Laplace en múltiples niveles.
371.applu331	Fortran	Simulación de flujos aerodinámicos en 3D.
372.smithwa	C	Algoritmo de alineamiento de secuencias biológicas.
376.kdtree	C	Construcción y manipulación de árboles K-d para búsqueda espacial.

SPEC ACCEL

El segundo conjunto de pruebas incluyó la suite SPEC ACCEL, diseñada para evaluar el rendimiento de la GPU en cargas de trabajo aceleradas mediante OpenACC y OpenCL. Estas pruebas permitieron analizar la capacidad de cómputo de la GPU Maxwell de la Jetson Nano, que cuenta con 128 núcleos CUDA y 4 GB de memoria compartida.

La máquina de referencia definida por la suite SPEC corresponde a un sistema equipado con una NVIDIA Tesla C2075, una tarjeta basada en la arquitectura Fermi, con 448 núcleos CUDA y 6 GB de memoria GDDR5. Esta referencia sirve como estándar de comparación para contextualizar los resultados obtenidos en la Jetson Nano. El objetivo principal de estas pruebas fue evaluar la viabilidad de ejecutar aplicaciones OpenCL mediante PoCL en esta plataforma, y obtener una estimación de su rendimiento en cargas paralelas utilizando su GPU integrada.

La instalación de SPEC ACCEL presentó ciertos desafíos, debido a la ausencia de soporte nativo para OpenCL en la Jetson Nano. Las herramientas de OpenACC proporcionadas por NVIDIA están orientadas a arquitecturas ARM para servidores, por lo que se optó por utilizar

PoCL (*Portable Computing Language*), una implementación de OpenCL que permite ejecutar código en plataformas sin compatibilidad nativa. No obstante, esta solución introduce cierta sobrecarga, ya que la traducción de código resulta menos eficiente en comparación con entornos que disponen de soporte directo por hardware. Los detalles del proceso de instalación y ejecución pueden encontrarse en el **Anexo L**.

Para asegurar la correcta ejecución de la suite, se seleccionaron únicamente los benchmarks de OpenCL que resultaron viables en la Jetson Nano, verificando que todas las pruebas se ejecutaran sin errores. La **TABLA V** presenta los benchmarks seleccionados, incluyendo su nombre, lenguaje de implementación y una breve descripción. Estas pruebas abarcan distintas áreas, como dinámica molecular, procesamiento de imágenes, álgebra lineal y simulaciones físicas.

TABLA V
PRUEBAS OPENCL DISPONIBLES EN LA SUITE SPEC ACCEL.

Test	Lenguaje	Descripción
001.systest	C++	Rutina de soporte para pruebas (función de prueba interna).
101.tpacf	C++	Cálculo de correlación de pares de galaxias en astrofísica.
103.stencil	C++	Simulación de transferencia de calor mediante métodos de stencil.
104.lbm	C++	Simulación de dinámica de fluidos (método Lattice Boltzmann).
110.fft	C	Implementación de la Transformada Rápida de Fourier (FFT).
112.spmv	C++	Multiplicación de matrices dispersas por vectores (álgebra lineal).
114.mriq	C	Reconstrucción de imágenes por resonancia magnética.
116.histo	C	Cálculo de histogramas aplicado a verificación de obleas de silicio.
117.bfs	C	Implementación del algoritmo BFS para recorrido de grafos.
118.cutcp	C	Interacciones moleculares en simulaciones de dinámica molecular.
120.kmeans	C++	Algoritmo de agrupamiento K-Means (minería de datos).
121.lavamd	C	Simulación N-Body con fuerzas de Van der Waals (dinámica molecular).
122.cfd	C	Simulación de fluidos con mallas no estructuradas (CFD).
123.nw	C++	Alineamiento de secuencias biológicas (algoritmo Needleman-Wunsch).
124.hotspot	C	Simulación térmica de circuitos integrados.
125.lud	C	Factorización LU para resolución de sistemas de ecuaciones.
126.ge	C	Eliminación Gaussiana en álgebra lineal densa.
127.srad	C	Reducción anisotrópica de ruido en imágenes médicas.
128.heartwall	C	Detección de contornos en imágenes médicas de ultrasonido.
140.bplustree	C	Búsqueda en estructuras tipo B+Tree (recorrido estructurado).

BENCHMARKS A NIVEL DE CLUSTER

SPEC MPI 2007

Para evaluar el rendimiento del clúster en cargas de trabajo paralelizadas, se utilizó la suite SPEC MPI2007, diseñada para medir la eficiencia y escalabilidad de sistemas multiprocesador mediante MPI. A diferencia de SPEC OMP2012 y SPEC ACCEL, que evalúan el rendimiento de un nodo individual, SPEC MPI2007 analiza el comportamiento del clúster en su conjunto, midiendo la comunicación y distribución de tareas entre múltiples nodos, en pruebas que abarcan dinámica de fluidos, modelado climático, análisis estructural y procesamiento de datos biomédicos, todas representativas de cargas de trabajo en entornos HPC.

Para adaptarse a diferentes configuraciones de clústeres, SPEC MPI2007 proporciona dos datasets, cada uno asociado a una máquina de referencia:

- ***Medium*** **Dataset:** su referencia es un clúster de 8 nodos Celestica A2210, cada uno con un AMD Opteron 848 de un núcleo a 2.2 GHz y 4 GB de RAM por socket. La interconexión entre nodos utiliza Gigabit Ethernet y las pruebas en la referencia se ejecutaron con 16 procesos MPI.

- ***Large*** **Dataset:** diseñado para evaluar la escalabilidad del sistema, su referencia es un clúster de 64 nodos Intel SR1560SF, cada uno con dos procesadores Intel Xeon X5482 de cuatro núcleos a 3.2 GHz y 16 GB de RAM. La interconexión también es Gigabit Ethernet y las pruebas en la referencia se ejecutaron con 64 procesos MPI.

Cada dataset cuenta con dos versiones, *test* y *ref*, lo que da un total de cuatro combinaciones evaluadas en el clúster de Jetson Nano: mtest, mref, ltest y lref.

La instalación de SPEC MPI2007 en las tarjetas Jetson Nano presentó múltiples desafíos técnicos debido a que esta suite, relativamente antigua, no cuenta con soporte oficial para la arquitectura ARM64 utilizada por estos dispositivos. Por ello, fue necesario realizar ajustes

específicos y compilar desde cero tanto el código fuente como las herramientas auxiliares incluidas. Los detalles pueden encontrarse en el **Anexo M**.

Una vez adaptada la suite, se instaló en el NAS para proporcionar acceso centralizado a los archivos requeridos por todos los nodos. Sin embargo, debido a la configuración híbrida del clúster (con un controlador x86 y nodos ARM64), surgió una nueva limitación: los binarios compilados para ARM64 no podían ejecutarse en el controlador x86, impidiendo el lanzamiento del benchmark desde este. Para solucionar este inconveniente, se accedió a uno de los nodos Jetson, desde donde se reservaron todos los nodos del clúster utilizando el comando *salloc*. Esto permitió ejecutar el benchmark y lanzar el trabajo sin estar directamente en el nodo principal.

Para la ejecución de la suite, se aplicó el proceso de depuración previamente establecido para garantizar que solo se ejecutaran pruebas compatibles. Se utilizaron 40 procesos en paralelo para aprovechar al máximo los recursos del clúster.

La **TABLA VI** muestra las pruebas que conforman la suite, especificando cuáles pertenecen a los datasets *medium* y *large*, el lenguaje de programación utilizado y una breve descripción de este.

TABLA VI
PRUEBAS DISPONIBLES EN LA SUITE SPEC MPI2007.

Test	Dataset	Lenguaje	Descripción
104.milc	medium	C	Simulación de física de partículas subatómicas utilizando cromodinámica cuántica (QCD).
107.leslie3d	medium	Fortran	Simulación tridimensional de dinámica de fluidos compresibles utilizando métodos numéricos.
113.GemsFDTD	medium	Fortran	Resolución de ecuaciones de Maxwell mediante el método FDTD en simulaciones electromagnéticas.
115.fds4	medium	C/Fortran	Simulación computacional de incendios y dinámica de gases en escenarios complejos.
121.pop2	medium, large	C/Fortran	Modelo de circulación oceánica global utilizado para estudios climáticos y oceanográficos.
122.tachyon	medium, large	C	Renderizado de imágenes mediante trazado de rayos paralelo en entornos gráficos complejos.
125.RAxML	large	C	Cálculo de árboles filogenéticos mediante comparación de secuencias de ADN.
126.lammps	medium, large	C++	Simulación de dinámica molecular a gran escala para materiales y biomoléculas.
127.wrf2	medium	C/Fortran	Modelo de predicción meteorológica de alta resolución basado en dinámica atmosférica.
128.GAPgeofem	medium, large	C/Fortran	Simulación de transferencia de calor mediante métodos de elementos finitos (FEM).
129.tera_tf	medium, large	Fortran	Simulación de hidrodinámica Euleriana tridimensional en física de plasmas y fluidos.
130.socorro	medium	Fortran	Simulación de dinámica molecular basada en la teoría del funcional de la densidad (DFT).
132.zesump2	medium, large	C/Fortran	Simulación avanzada de dinámica de fluidos con múltiples regiones acopladas.
137.lu	medium, large	C/Fortran	Resolución numérica de flujos compresibles mediante métodos de volumen finito.
142.dmilc	large	C	Versión extendida del benchmark MILC para simulaciones de QCD en mallas más grandes.
143.dleslie	large	Fortran	Modelo de dinámica de fluidos tridimensional con turbulencia para simulaciones atmosféricas.
145.GlemsFDTD	large	Fortran	Simulación electromagnética usando el método FDTD aplicado a estructuras complejas.
147.l2wrf2	large	C/Fortran	Extensión del modelo WRF para simulaciones meteorológicas de gran escala.

HPL

El benchmark HPL es una prueba diseñada para medir el rendimiento máximo en operaciones de punto flotante de precisión doble (FLOPs) en un sistema distribuido. A diferencia de los benchmarks de SPEC, que evalúan diversas cargas de trabajo científicas y de ingeniería, HPL se enfoca en resolver sistemas de ecuaciones lineales densos utilizando el método de factorización LU con pivoteo parcial, lo que lo convierte en un referente estándar para evaluar el desempeño de clústeres de cómputo de alto rendimiento (HPC).

Dado que HPL no utiliza una máquina de referencia como los benchmarks de SPEC, su evaluación se basa en el rendimiento absoluto del clúster, expresado en GigaFLOPs (GFLOPs) o TeraFLOPs (TFLOPs). Para interpretar los resultados, es común compararlos con sistemas de referencia en la lista TOP500, donde los supercomputadores más potentes alcanzan PetaFLOPs (PFLOPs).

Para implementar HPL en el clúster de Jetson Nano, primero se instaló una biblioteca compatible con BLAS (Basic Linear Algebra Subprograms), optando por OpenBLAS debido a su compatibilidad con ARM64. Luego, se compiló HPL desde el código fuente, ajustando los archivos de compilación para adaptarlos a esta arquitectura.

El proceso de compilación genera un ejecutable junto con un archivo de configuración (HPL.dat), que permite modificar los parámetros del benchmark. La selección de estos parámetros se realiza mediante *tuning*, ajustándolos en función de la capacidad de cómputo, el uso de memoria y la eficiencia de comunicación entre nodos para maximizar el desempeño del clúster. En el caso de parámetros con múltiples valores, HPL ejecuta todas las combinaciones posibles, generando varias subpruebas que entregan como resultado el rendimiento en FLOPs.

HPL divide la matriz en bloques y la distribuye entre múltiples procesos en el clúster, los cuales trabajan en paralelo utilizando MPI para comunicarse. La ejecución del benchmark sigue las siguientes etapas:

1. **Inicialización y configuración:** se genera una matriz aleatoria de tamaño N × N y se divide en bloques de NB × NB, distribuidos entre los nodos del clúster. Los procesos MPI se organizan en una malla bidimensional P × Q, definiendo la distribución del trabajo (ver **Fig. 11**).
2. **Factorización LU distribuida:** cada nodo opera sobre sus bloques asignados, aplicando eliminación gaussiana con pivoteo parcial. Los procesos intercambian datos mediante MPI para actualizar la matriz y continuar con la factorización.
3. **Resolución del sistema y medición del rendimiento:** se resuelve el sistema triangular resultante mediante sustitución hacia adelante y hacia atrás. HPL mide el tiempo total de ejecución y calcula el rendimiento en FLOPs que representa la cantidad total de operaciones realizadas en la matriz.

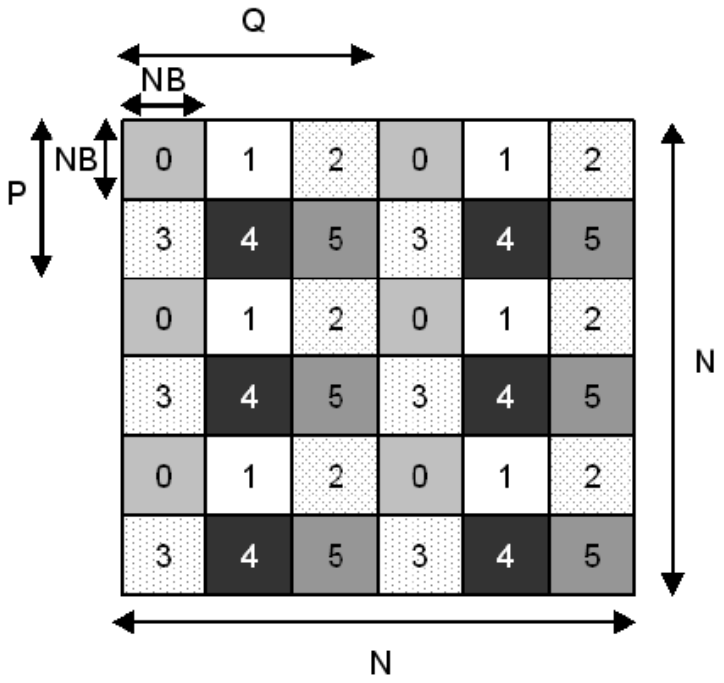


Fig. 11. Esquema de la matriz de dimensión N × N dividida en bloques de NB × NB y distribuida entre procesos MPI organizados en una malla P × Q.

Nota: fuente https://www.researchgate.net/figure/An-example-of-P-by-Q-partitioning-of-a-HPL-matrix-in-6-processes-2x3-decomposition_fig1_228524393

La ejecución del benchmark depende de tres parámetros clave que afectan la resolución del sistema:

Tamaño del problema (N): define la dimensión de la matriz cuadrada (N × N). Un valor mayor implica una mayor carga computacional y mejores mediciones de FLOPs, pero también un mayor uso de memoria.

Tamaño del bloque (NB): determina cómo se divide la matriz en bloques de NB × NB para optimizar la comunicación y el uso de la caché. Un NB demasiado pequeño puede hacer que la comunicación domine el cómputo, mientras que uno muy grande puede desbalancear la carga de trabajo.

Configuración de procesos (P × Q): especifica la distribución de procesos en una malla bidimensional. P × Q debe igualar el número total de procesos MPI utilizados. En general, se busca minimizar la latencia de comunicación y maximizar la paralelización. Típicamente se busca que P × Q sea igual al número de núcleos disponibles en el clúster.

Estos valores se ajustaron en función de la cantidad de nodos disponibles, la distribución de carga y el rendimiento de la memoria para obtener los mejores resultados en el benchmark.

Para la ejecución de HPL, se configuró un entorno MPI utilizando OpenMPI como *middleware* de comunicación entre nodos. Se realizaron múltiples ejecuciones con diferentes tamaños de problema (N) y (NB), ajustando los parámetros para optimizar el rendimiento del clúster. Las dimensiones de P y Q siempre se eligieron de forma que se utilizaran los 40 procesos disponibles en el clúster de Jetson Nano.

Los resultados se expresaron en GFLOPs, como medida del desempeño del clúster en cálculos de precisión doble. Paralelamente, se empleó *tegrastats* en cada nodo para monitorear el uso de CPU, GPU y memoria, lo que permitió evaluar el impacto del benchmark sobre los recursos del sistema.

La información detallada sobre la instalación y ejecución de HPL se encuentra en el **Anexo N**.

V. RESULTADOS

A. COMPONENTE MECÁNICA

Impresión del soporte de la Jetson Nano:

Como primer paso para alojar las tarjetas Jetson Nano en el rack, se diseñó un soporte impreso en 3D que permitiera fijar cada tarjeta de forma estable y aprovechar de manera eficiente el espacio vertical disponible. El criterio principal fue acomodar hasta 10 tarjetas en una altura de 3U, disponiéndolas de forma vertical para maximizar la densidad de integración en el rack (ver **Fig. 12**).

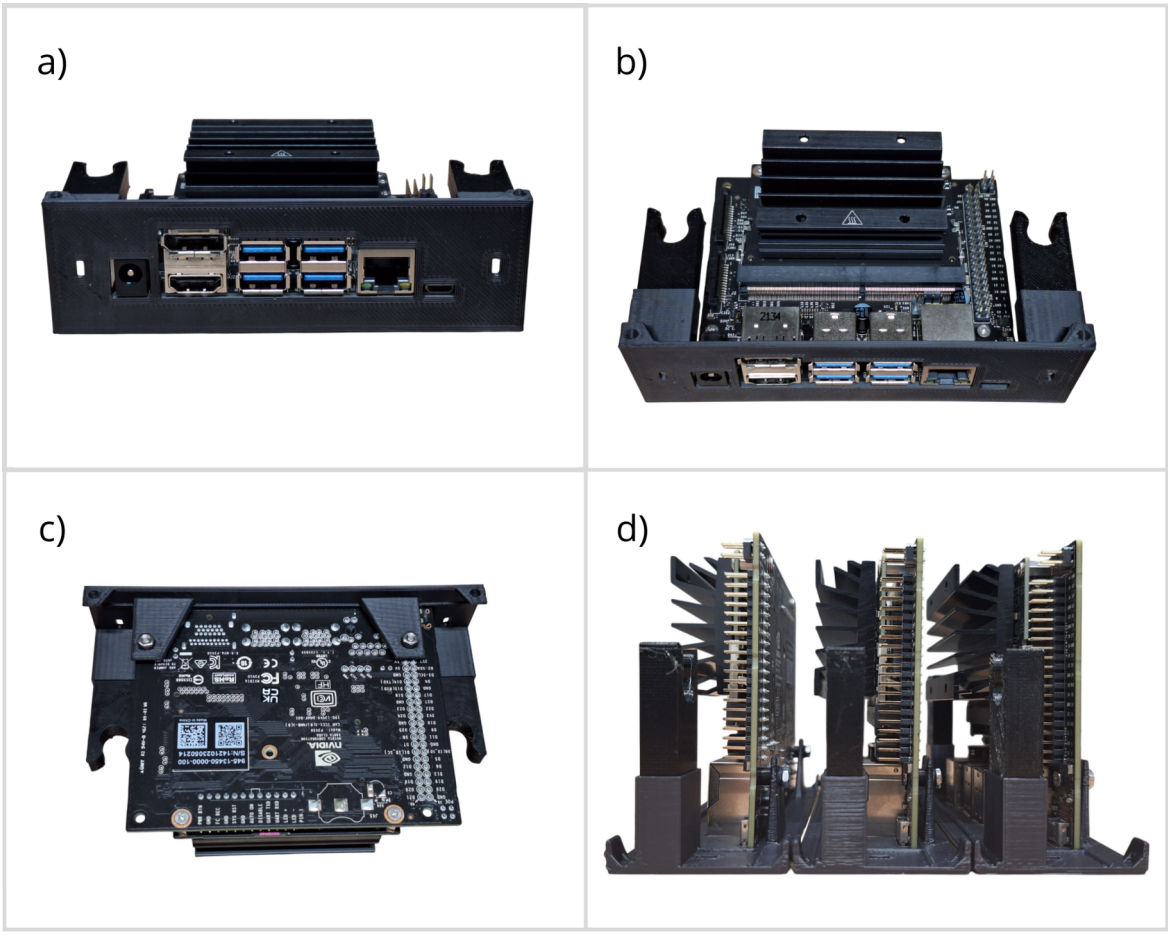


Fig. 12. Vista frontal (a), superior (b), inferior (c) y lateral (d) del soporte impreso en 3D para la Jetson Nano.

En la vista frontal (a), se observa la alineación precisa de las aberturas para HDMI, USB, Ethernet y otros conectores, garantizando un acceso cómodo a los puertos. Su dimensión corresponde con el estándar 3U.

La vista superior (b) muestra las pinzas impresas en material flexible, diseñadas para sujetar el soporte a los tubos del sistema de sujeción. Como se describió en la metodología, estas piezas se ensamblan a presión en las aberturas correspondientes. Además, se confirma que los pines GPIO de la tarjeta quedan completamente accesibles.

En la vista inferior (c), se aprecia la fijación de la tarjeta al soporte mediante tornillos y tuercas en las pestañas del diseño. Entre la tarjeta y el soporte, los bujes impresos aíslan las soldaduras del PCB, evitando posibles daños por contacto.

Por último, la vista lateral (d) resalta la modularidad del diseño, permitiendo la instalación ordenada de múltiples Jetson Nano en soportes idénticos. La distribución mantiene una separación adecuada entre tarjetas para facilitar la circulación de aire, contribuyendo a la ventilación y escalabilidad del sistema en el rack.

Sistema de sujeción:

Una vez diseñados los soportes individuales para las Jetson Nano, se integraron en el sistema de sujeción modular dentro del rack, como se muestra en la **Fig. 13**.

Fig. 13. Vista frontal (izquierda) y trasera (derecha) del sistema de sujeción.

En la vista frontal se observa cómo el sistema de sujeción se acopla a los rieles del rack mediante piezas impresas en 3D, ensambladas con tornillos y tuercas, lo que garantiza una instalación firme y segura. También se aprecia uno de los soportes montado verticalmente sobre los tubos mediante las pinzas flexibles.

Por otro lado, la vista trasera muestra la estructura completa del sistema de sujeción, conformada por tres tubos de aluminio dispuestos horizontalmente. Dos de ellos se encargan de sostener los soportes de las Jetson Nano, mientras que el tercero se utiliza para organizar el cableado de alimentación. Este último se fija al sistema mediante una pieza adicional en forma de Y impresa en 3D.

En conjunto, el sistema de montaje validó satisfactoriamente los criterios de estabilidad, modularidad y eficiencia espacial planteados en el diseño, facilitando la instalación ordenada de hasta 10 tarjetas en una altura de 3U, como se observa en el montaje completo mostrado en la **Fig. 14**.

Fig. 14. Montaje final de las diez tarjetas en el sistema de sujeción.

B. COMPONENTE ELÉCTRICA

La implementación del sistema de alimentación del clúster garantizó un suministro eléctrico estable y ordenado para las tarjetas Jetson Nano, optimizando la distribución de energía y minimizando el desorden del cableado en el rack.

La **Fig. 15** ilustra el sistema de distribución de energía implementado. A la izquierda, se muestra una de las líneas principales de 5V, con derivaciones individuales hacia cada tarjeta y conectores específicos para recibir alimentación desde una de las fuentes ATX de alta capacidad. Esta configuración, basada en la división de carga entre dos fuentes y líneas independientes por grupo, permitió reducir la posibilidad de sobrecarga, conservar un margen de seguridad adecuado y facilitar futuras expansiones. A la derecha, se observa el conector JST-XH utilizado para conectar cada derivación a su respectiva tarjeta Jetson Nano, mediante un pin de 5V y uno de GND, lo que permite suministrar hasta 3A por tarjeta.

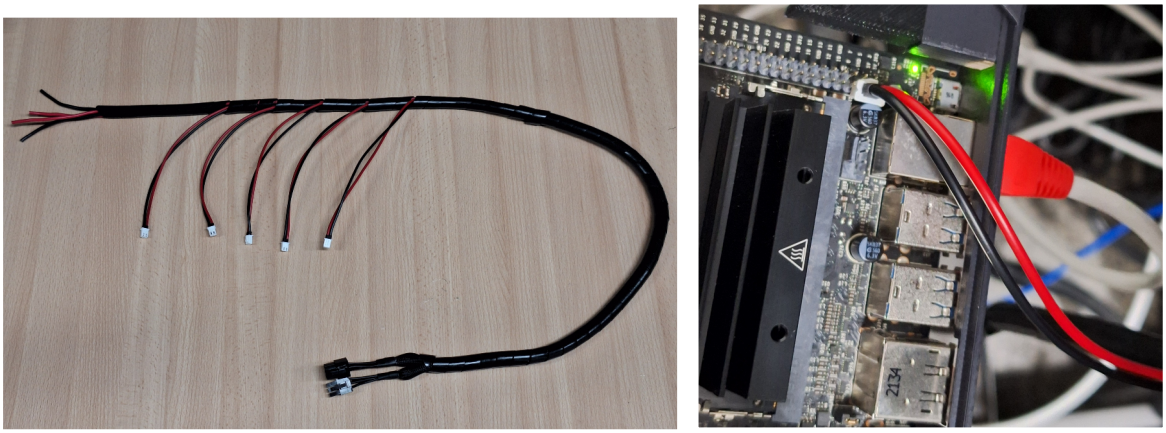


Fig. 15. Línea del sistema de distribución para una de las fuentes (izquierda) y conector utilizado para alimentar la tarjeta desde la derivación (derecha).

Ambas líneas fueron instaladas sobre el tubo del sistema de sujeción, desde donde se conectaron todas las tarjetas Jetson Nano (ver **Fig. 16**). Cada línea incluye un tramo de cable excedente, lo que permite realizar derivaciones adicionales en caso de futuras expansiones.

Fig. 16. Sistema de distribución eléctrica montado en el rack.

C. COMPONENTE DE SOFTWARE

Antes de abordar los aspectos específicos del software instalado, se verificó exitosamente la comunicación entre todos los nodos del clúster, garantizando una conexión estable para la gestión y transferencia de datos. La administración centralizada de usuarios en el servidor permite el acceso a la carpeta compartida desde cualquier nodo, facilitando la ejecución de archivos de manera uniforme. Asimismo, cada Jetson Nano puede ser accedida individualmente mediante SSH heredando los permisos asignados en el nodo principal, lo que habilita la ejecución de tareas específicas sin depender del gestor de colas ni del uso completo de la infraestructura.

Para evaluar el correcto funcionamiento del entorno de software instalado, se realizaron pruebas utilizando SLURM y OpenMPI, cuyos detalles se encuentran en el **Anexo J**. Las tres primeras pruebas validaron aspectos fundamentales del sistema: la operatividad del gestor de colas SLURM, el acceso y gestión de archivos a través del almacenamiento centralizado, y la correcta integración con OpenMPI para la ejecución de tareas distribuidas.

La cuarta prueba consistió en la ejecución de un programa más complejo, basado en MPI, diseñado para calcular la cantidad de números primos menores o iguales a un valor *n*. Esta tarea, altamente paralelizable a nivel de datos, fue distribuida entre los nodos del clúster utilizando distintas configuraciones en la cantidad de procesos MPI. Los resultados permitieron analizar el desempeño del sistema en términos de *speedup* y eficiencia, proporcionando información clave sobre su escalabilidad y capacidad de procesamiento distribuido.

Se evaluaron tres valores de entrada:

$n = 1\times10^6$: 78498 primos.

$n = 1\times10^7$: 664579 primos.

$n = 1\times10^8$: 5761455 primos.

En todos los casos, los resultados obtenidos coincidieron con los valores esperados. La **TABLA VII** presenta los tiempos de ejecución medidos al variar la cantidad de procesos MPI utilizados en cada caso.

TABLA VII
TIEMPOS OBTENIDOS PARA DIFERENTES VALORES DE *N* Y DIFERENTES NUMEROS DE PROCESOS.

Nodos	**Procesos**	**Tiempo (s)**		
		n= 1M	***n= 10M***	***n= 100M***
1	1	0.19233150	5.08651729	141.63363982
1	2	0.11821671	3.22072934	90.17847237
1	4	0.06264561	1.70935271	49.36826967
2	8	0.03209381	0.87967052	25.35791540
4	16	0.01699186	0.44476019	12.85474744
6	24	0.02006463	0.30726710	8.66692637
8	32	0.00892506	0.22371562	6.46551666
10	40	0.01602835	0.18946044	5.18848900

Los valores de *speedup* y eficiencia se calcularon a partir de las definiciones matemáticas presentadas en el marco teórico. Posteriormente, estos indicadores se graficaron en función del número de procesos MPI utilizados, con el fin de analizar el comportamiento del clúster y evaluar su escalabilidad (ver **Fig. 17** y **Fig. 18**).

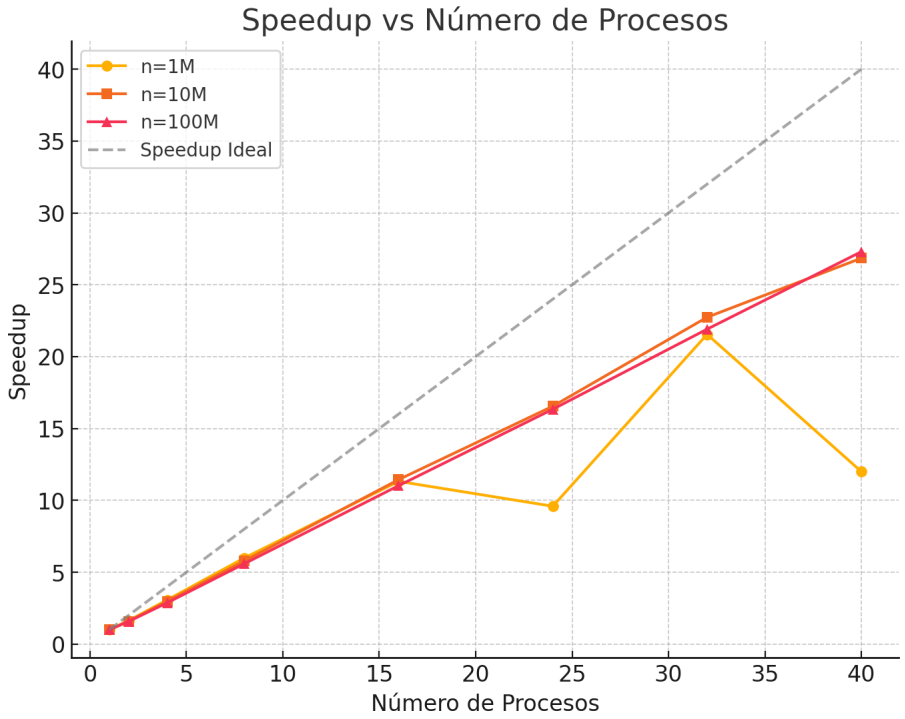


Fig. 17. Speedup en función del número de procesos.

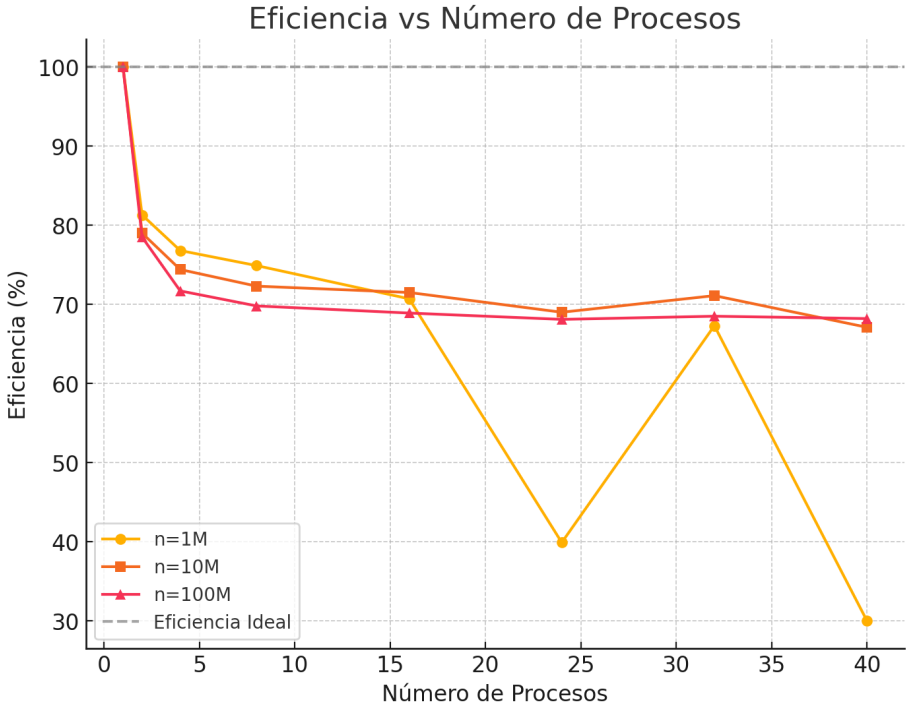


Fig. 18. Eficiencia en función del número de procesos.

La **Fig. 17** muestra el comportamiento del *speedup* en función del número de procesos para distintos tamaños de problema. En general, el *speedup* aumenta conforme se incrementa el número de procesos, lo que confirma la capacidad de paralelización de la tarea. Sin embargo, no se alcanza un crecimiento lineal ideal debido a los costos asociados con la creación, comunicación y sincronización de procesos, así como a la presencia inevitable de una fracción secuencial en el código.

En los problemas más grandes (n=10M y n=100M), el *speedup* se acerca más a la referencia ideal, lo que indica un mejor aprovechamiento del paralelismo. En contraste, para n=1M, se observa una degradación notoria en los puntos de 24 y 40 procesos, lo que sugiere que el *overhead* de comunicación supera las ganancias de la paralelización cuando el tamaño del problema es demasiado pequeño.

Este comportamiento coincide con la Ley de Amdahl, según la cual el rendimiento siempre estará limitado por la fracción secuencial del código $(1-p)$. Incluso al incrementar indefinidamente el número de procesadores, el tiempo total de ejecución no podrá ser menor que el tiempo requerido por esta fracción secuencial, estableciendo así un límite en la máxima mejora posible.

La **Fig. 18** muestra la eficiencia en el cálculo de números primos para distintos tamaños de problema según el número de procesos. Se aprecia que la eficiencia disminuye conforme aumenta la cantidad de procesos, lo que refleja la sobrecarga inherente a la paralelización. Sin embargo, en los casos de n=10M y n=100M, la eficiencia permanece relativamente alta y estable, evidenciando que la mayor carga de trabajo justifica la distribución entre múltiples procesos.

En problemas de menor escala (n=1M), la eficiencia cae abruptamente en ciertos puntos (24 y 40 procesos), lo que confirma la necesidad de encontrar un equilibrio entre el número de procesos y el tamaño del problema para optimizar el rendimiento.

Por último, la eficiencia nunca alcanza el 100% debido a factores como la fracción secuencial del código (Ley de Amdahl), el *overhead* en la creación y gestión de procesos, y el desbalance en la carga de trabajo. En consecuencia, el desafío no radica simplemente en añadir más procesos, sino en hallar una configuración óptima que maximice el rendimiento sin que la sobrecarga supere los beneficios del paralelismo.

D. EVALUACIÓN DE DESEMPEÑO:

BENCHMARKS PARA UN NODO INDIVIDUAL

SPEC OMP2012

La suite de benchmarks SPEC OMP2012 se utilizó para evaluar el rendimiento de un único nodo Jetson Nano bajo cargas de trabajo paralelizadas mediante OpenMP. Este conjunto de pruebas está orientado a medir la eficiencia del procesamiento paralelo a nivel de CPU, aprovechando múltiples hilos de ejecución que comparten un mismo espacio de memoria. Las pruebas se realizaron utilizando los cuatro núcleos de la Jetson, configurados para operar a su máxima frecuencia.

TABLA VIII
RESULTADOS DE LA SUITE SPEC OMP2012.

Benchmark	Tiempo de ejecución (s)	Base Ratio	Resultado	Error
350.md	31038	0.149	OK	-
351.bwaves	-	-	NR	Killed
352.nab	30445	0.128	OK	-
357.bt331	-	-	NR	Segmentation fault
358.botsalgn	68512	0.0635	OK	-
359.botsspar	-	-	NR	Killed
360.ilbdc	-	-	NR	Cannot allocate memory
362.fma3d	-	-	NR	Killed
363.swim	-	-	NR	Segmentation fault
367.imagick	61075	0.115	OK	-
370.mgrid331	-	-	NR	Cannot allocate memory
371.applu331	-	-	NR	Killed
372.smithwa	-	-	NR	Segmentation fault
376.kdtree	10958	0.411	OK	-

Como se muestra en la **TABLA VIII**, de los 14 benchmarks incluidos en la suite, únicamente cinco pruebas lograron completarse exitosamente con el conjunto de datos *ref*. Los resultados obtenidos muestran tiempos de ejecución entre 10.958 y 68.512 segundos, y valores de Base Ratio que oscilan entre 0.0635 y 0.411, siendo este último correspondiente a *376.kdtree*, el cual presentó el mejor desempeño relativo respecto a la máquina de referencia definida por SPEC.

El Base Ratio representa el cociente entre el tiempo de ejecución de la prueba en la Jetson Nano y el tiempo registrado en la máquina de referencia. Un valor más alto indica un mejor rendimiento relativo; por ejemplo, un ratio de 0.411 implica que la Jetson ejecutó esa prueba en aproximadamente el 41 % del tiempo de la máquina de referencia.

El resto de los benchmarks fallaron durante su ejecución y fueron clasificados como NR (Not Run). Las causas más comunes de fallo fueron:

- **Falta de memoria disponible**: reflejada en mensajes de error como "Cannot allocate memory" en pruebas como *360.ilbdc* y *370.mgrid331*. Aunque se habilitó un espacio de 10 GB de swap, este no fue suficiente para mitigar los altos requerimientos de memoria del conjunto *ref*.
- **Terminación inesperada del proceso (Killed)**: posiblemente causada por el sistema operativo al detectar consumo excesivo de recursos.
- **Errores de segmentación (Segmentation fault)**: observados en pruebas como *357.bt331* y *372.smithwa*, lo que sugiere posibles problemas de compatibilidad o estabilidad del código al ejecutarse en una arquitectura ARM64 con recursos limitados.

Estos resultados reflejan tanto las capacidades como las limitaciones de la Jetson Nano al ejecutar cargas de trabajo paralelas intensivas. A pesar de no ser una plataforma diseñada para HPC tradicional, logró ejecutar con éxito una parte significativa de la suite, mostrando un comportamiento aceptable en pruebas con requerimientos más moderados. Las pruebas fallidas, por otro lado, confirman que los recursos de memoria son un cuello de botella importante al trabajar con datasets grandes, lo cual debe ser tenido en cuenta para la planificación de tareas dentro del clúster.

Durante la ejecución de los benchmarks exitosos de SPEC OMP2012, se utilizó la herramienta tegrastats para monitorear en tiempo real el uso de recursos de la Jetson Nano. A partir de los datos recolectados, se generaron gráficas que ilustran el comportamiento del sistema a lo largo del tiempo.

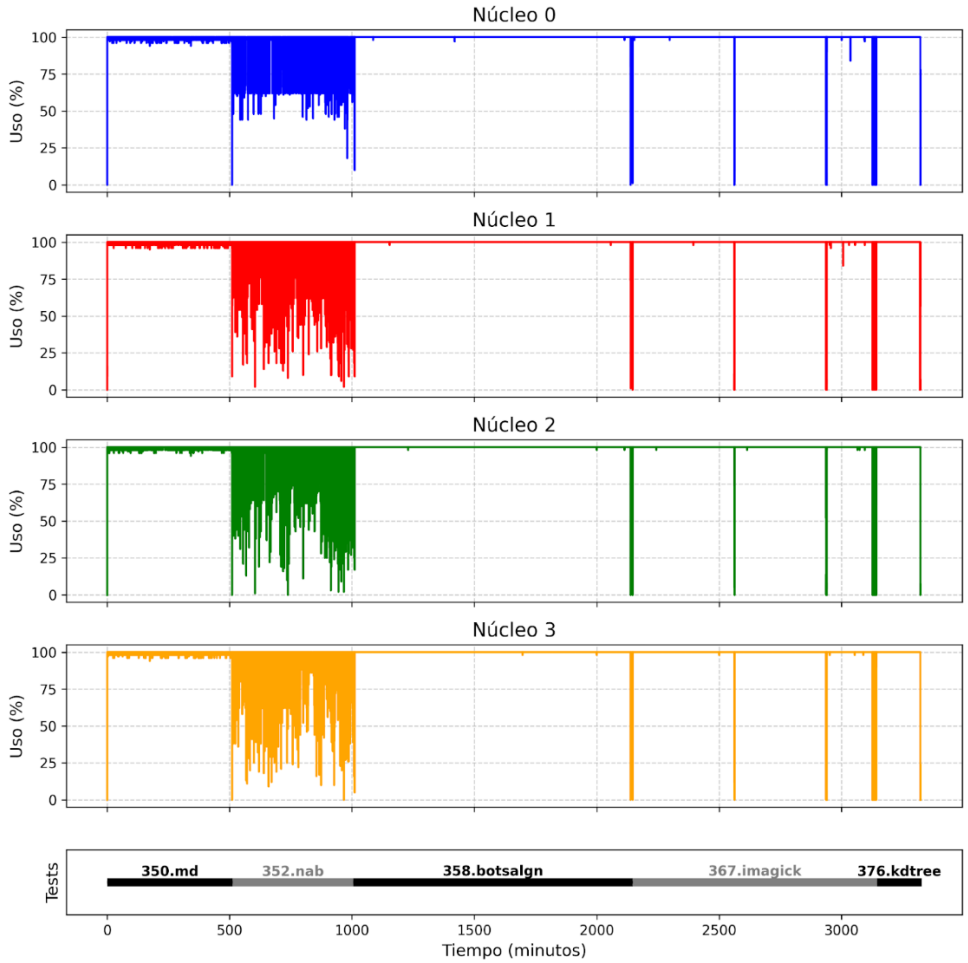


Fig. 19. Uso de los núcleos del procesador durante ejecución de SPEC OMP2012.

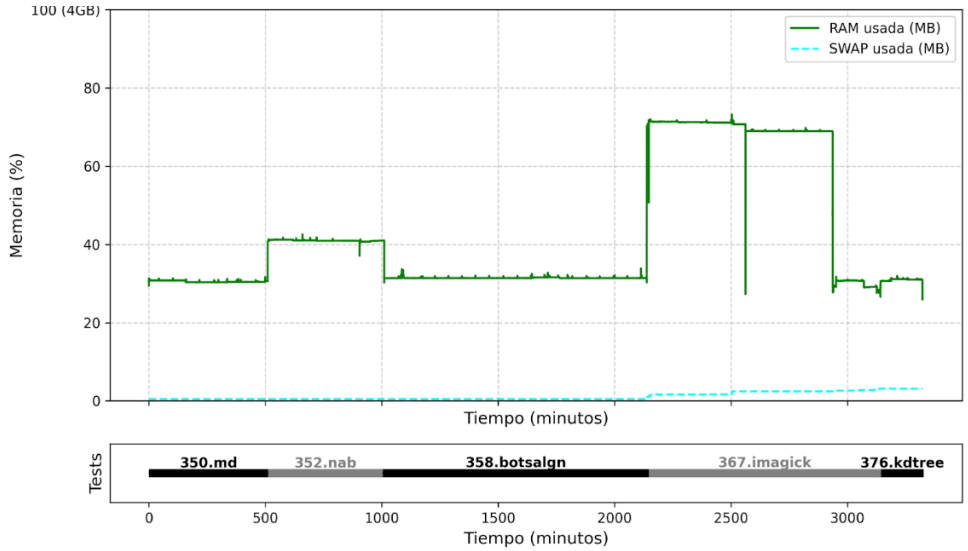


Fig. 20. Uso de memoria RAM y SWAP durante ejecución de SPEC OMP2012.

La **Fig. 19** desglosa el uso porcentual de los cuatro núcleos de CPU a lo largo de la ejecución de los benchmarks. Se aprecia una utilización efectiva del paralelismo en todas las pruebas exitosas, donde en general los núcleos operaron de forma sostenida cerca del 100 %, lo que valida el correcto funcionamiento del modelo multihilo implementado mediante OpenMP. No obstante, durante el benchmark *352.nab*, el uso de CPU mostró un comportamiento mucho más fluctuante, con variaciones abruptas en la carga de trabajo. En los casos de *367.imagick* y *376.kdtree*, se observaron varios periodos breves en los que los núcleos estuvieron completamente inactivos. Esto podría deberse a fases del algoritmo donde el acceso a memoria o disco predomina sobre el cómputo, o a particularidades propias de su ejecución.

La **Fig. 20** muestra el comportamiento del uso de memoria RAM y SWAP. El consumo de RAM se mantuvo, en general, por debajo del 50 %, con incrementos notables durante pruebas como *358.botsalgn*, que alcanzaron valores cercanos al límite de 4 GB. En el caso de *352.nab*, se evidenció un leve aumento en el uso de memoria, que coincide con su comportamiento irregular a nivel de CPU. Esto podría indicar que el benchmark realiza un mayor número de transacciones en memoria que operaciones de cómputo intensivo, lo que genera una carga menos constante sobre los núcleos.

Si bien se habilitó una partición de 10 GB de SWAP, su utilización fue prácticamente nula en las pruebas que finalizaron correctamente. Esto sugiere que los benchmarks exitosos no ejercieron una presión significativa sobre la memoria virtual, en contraste con aquellos que fallaron por errores de asignación, como *360.ilbdc* o *370.mgrid331*.

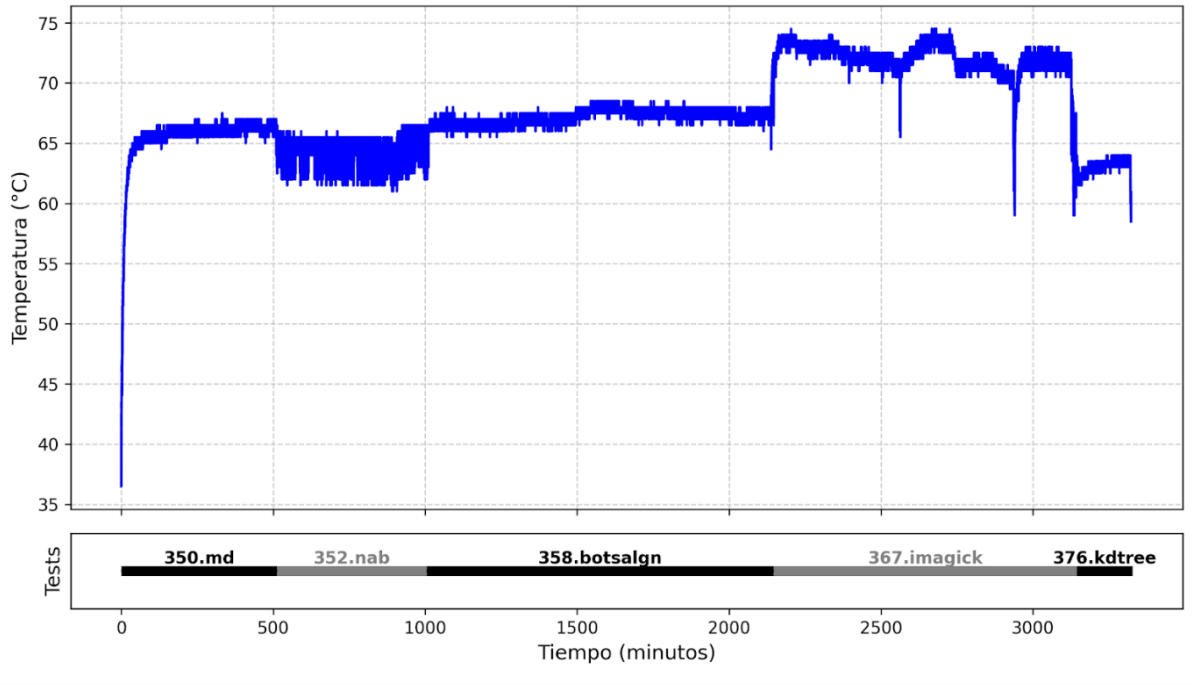


Fig. 21. Temperatura de la CPU durante ejecución de SPEC OMP2012.

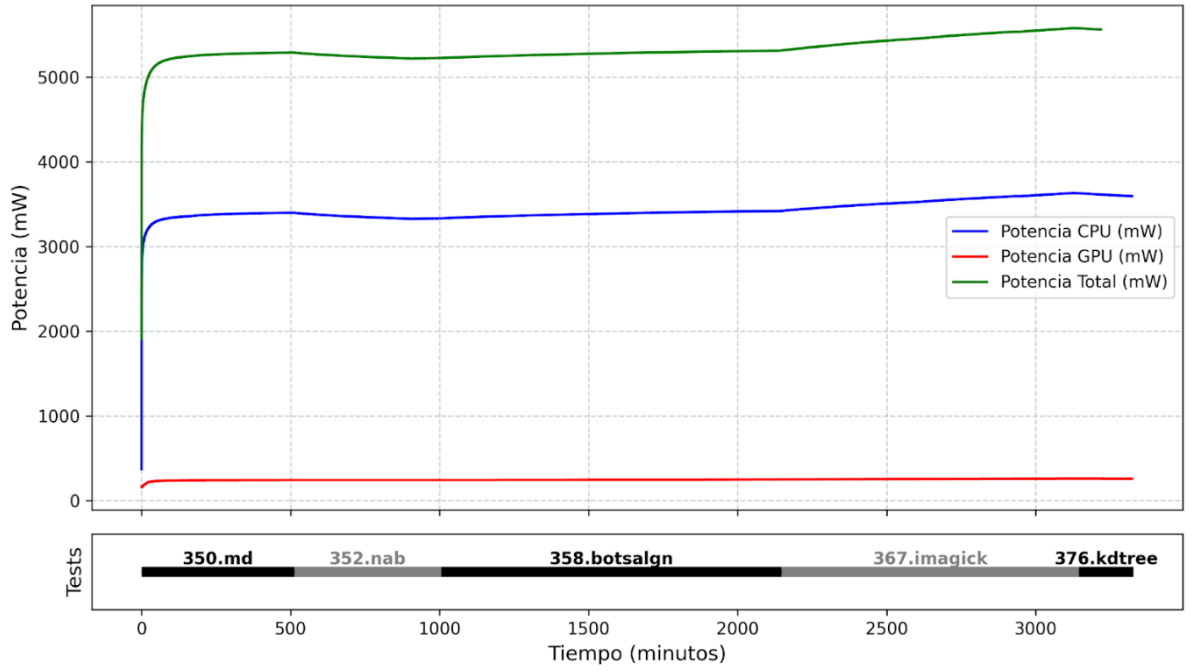


Fig. 22. Potencia promedio de la CPU, GPU y total durante ejecución de SPEC OMP2012.

La **Fig. 21** presenta la evolución térmica de la CPU durante la ejecución de los benchmarks. La temperatura se estabilizó rápidamente tras el inicio de cada prueba, oscilando entre 65 °C y 72 °C, sin alcanzar niveles críticos. Esto indica que el sistema de refrigeración pasiva incorporado en la Jetson Nano fue suficiente para mantener un funcionamiento estable incluso bajo cargas prolongadas. La curva térmica muestra una variabilidad notable a lo largo del tiempo, principalmente debido al intervalo de muestreo utilizado por la herramienta tegrastats, que genera un perfil ligeramente irregular en los registros. Aun así, se observa una correlación clara entre los picos de temperatura y aquellas pruebas que también presentaron un mayor uso de memoria, como *367.imagick*, lo que sugiere que el acceso intensivo a memoria también contribuye significativamente al calentamiento del procesador.

En la **Fig. 22** se aprecia cómo la potencia total del sistema (línea verde) asciende rápidamente al comienzo de la ejecución de los benchmarks, alcanzando alrededor de 5000 mW y, posteriormente, se mantiene estable durante casi todo el periodo de pruebas. Este comportamiento sugiere que la Jetson Nano entra rápidamente en su modo de máximo rendimiento y no presenta fluctuaciones bruscas de potencia, lo cual se traduce en un consumo contenido y predecible.

Por su parte, la potencia consumida por la CPU (línea azul) se sitúa cerca de los 3000 mW, mientras que la GPU (línea roja) permanece prácticamente constante y en valores bajos (~300mW). Este resultado es coherente con la naturaleza de los benchmarks de SPEC OMP2012, que se centran en la ejecución paralela sobre CPU en lugar de aprovechar la aceleración por GPU. En conjunto, los valores observados respaldan la eficiencia energética de la Jetson Nano, ya que el consumo global del sistema nunca supera los 6000 mW, incluso bajo cargas de trabajo prolongadas.

SPEC ACCEL

La suite SPEC ACCEL fue utilizada para evaluar el rendimiento de la Jetson Nano bajo cargas de trabajo paralelizadas con aceleración por GPU, utilizando el modelo de programación OpenCL. Esta suite está diseñada para medir la eficiencia computacional en arquitecturas heterogéneas, donde se combinan CPU y GPU. Las pruebas se ejecutaron utilizando PoCL, una implementación de OpenCL que permite correr código compatible en plataformas sin soporte nativo. En este caso, se empleó el backend CUDA de PoCL para aprovechar la GPU integrada de la Jetson Nano, basada en la arquitectura Maxwell con 128 núcleos CUDA.

TABLA IX
RESULTADOS DE LA SUITE SPEC ACCEL.

Benchmark	Tiempo de ejecución (s)	Base Ratio	Resultado	Error
101.tpacf	–	–	NR	pb_GetDevice + exit code 1
103.stencil	5143	0.0243	OK	–
104.lbm	325	0.344	OK	–
110.fft	–	–	NR	CL_INVALID_BUFFER_SIZE
112.spmv	661	0.222	OK	–
114.mriq	–	–	NR	pb_GetDevice + exit code 1
116.histo	–	–	NR	Not enough local memory
117.bfs	–	–	NR	Segmentation fault
118.cutcp	586	0.169	OK	–
120.kmeans	–	–	NR	Bus error (core dumped)
121.lavamd	–	–	NR	pb_GetDevice + exit code 195
122.cfd	553	0.228	OK	–
123.nw	549	0.210	OK	–
124.hotspot	524	0.218	OK	–
125.lud	–	–	NR	pb_GetDevice + exit code 255
126.ge	3091	0.0501	OK	–
127.srad	761	0.150	OK	–
128.heartwall	–	–	NR	CUDA_ERROR_LAUNCH_TIMEOUT
140.bplustree	–	–	NR	Miscompare of output

Como se muestra en la **TABLA IX**, de los 19 benchmarks que conforman la suite SPEC ACCEL OpenCL, únicamente 10 pruebas lograron completarse exitosamente utilizando la GPU integrada de la Jetson Nano. Los tiempos de ejecución registrados para estas pruebas varían entre 325 y 5143 segundos, mientras que los valores de Base Ratio oscilan entre 0.0243 y 0.344, siendo

este último correspondiente al benchmark *104.lbm*, que presentó el mejor desempeño relativo frente a la máquina de referencia definida por SPEC. Estos resultados reflejan la capacidad de la Jetson Nano para ejecutar ciertas cargas de trabajo paralelas aceleradas por hardware, aunque limitada por factores como la compatibilidad de drivers OpenCL, el uso de memoria y la frecuencia de operación de la GPU.

Del total de benchmarks, los 9 restantes fallaron por diversos motivos, representando aproximadamente un 47% de fallos. A continuación, se detallan los principales tipos de errores identificados durante las ejecuciones fallidas:

- **Inicialización fallida del dispositivo OpenCL**: el mensaje de error "pb_GetDevice + exit code" fue reportado en varios casos (*101.tpacf, 114.mriq, 121.lavaMD, 125.lud*), indicando que la plataforma OpenCL no logró detectar o inicializar correctamente el dispositivo GPU.
- **Errores de memoria**: el benchmark *116.histo* falló por falta de memoria local compartida suficiente para el kernel, mientras que *110.fft* presentó el error CL_INVALID_BUFFER_SIZE, relacionado con la asignación de búferes de tamaño excesivo.
- **Fallos por acceso indebido a memoria**: se observaron errores como "Segmentation fault (core dumped)" y "Bus error" en pruebas como *117.bfs* y *120.kmeans*, posiblemente asociados a incompatibilidades en la arquitectura de memoria o errores en el manejo de punteros.
- **Errores específicos de CUDA**: en el caso de *128.heartwall*, se presentó el error CUDA_ERROR_LAUNCH_TIMEOUT, lo que sugiere que el kernel superó el tiempo máximo de ejecución permitido.
- **Resultado incorrecto**: finalmente, el benchmark *140.bplustree* fue clasificado como "Miscompare of output", al detectarse una discrepancia entre la salida esperada y la generada.

Estos resultados evidencian que, si bien la Jetson Nano cuenta con capacidades de cómputo acelerado por GPU mediante OpenCL, su compatibilidad con ciertos benchmarks de la suite SPEC ACCEL es limitada. Esto era previsible, dada la ausencia de soporte nativo para OpenCL y las limitaciones inherentes al empleo de PoCL como plataforma de ejecución. Las fallas recurrentes

relacionadas con la detección del dispositivo, la administración de memoria y el tiempo de ejecución reflejan las restricciones tanto del hardware como del soporte parcial ofrecido por PoCL en esta arquitectura. En consecuencia, es fundamental tener en cuenta estas limitaciones al diseñar y escalar un clúster basado en tarjetas Jetson Nano, priorizando aplicaciones que se ajusten a los recursos disponibles y que presenten una mayor compatibilidad con su hardware nativo.

Para complementar el análisis de desempeño, a continuación, se presentan las gráficas generadas con tegrastats, las cuales permiten visualizar el comportamiento de los recursos del sistema durante la ejecución de los benchmarks exitosos de SPEC ACCEL.

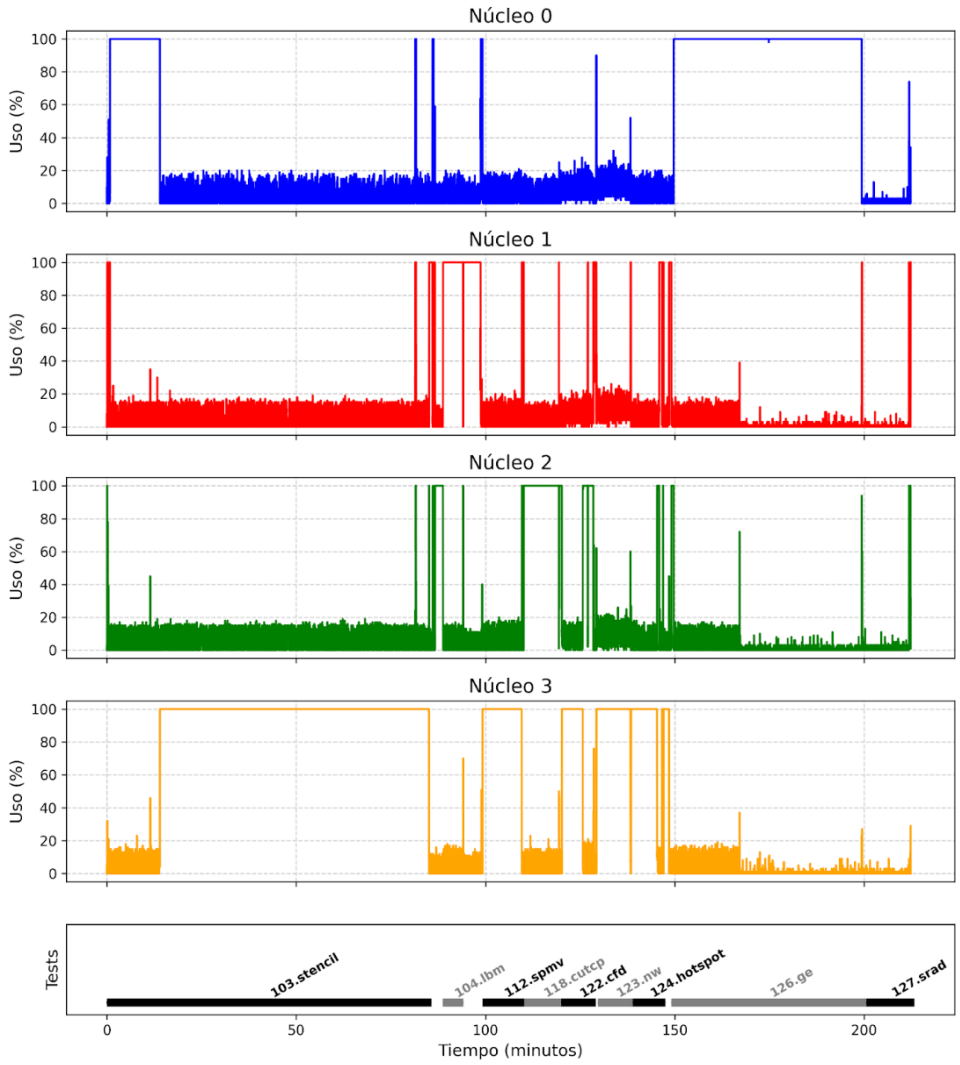


Fig. 23. Uso de los núcleos del procesador durante ejecución de SPEC ACCEL.

En la **Fig. 23**, se analiza el uso individual de cada uno de los cuatro núcleos del CPU durante la ejecución de los benchmarks. Se aprecian patrones de actividad dispares entre los núcleos: algunos, como el 0 y el 3, presentan tramos con alta utilización sostenida, mientras que otros muestran actividad más puntual o intermitente. A pesar de esta variabilidad, se observa que el procesador está en uso de forma continua, aunque alternando cuál núcleo asume la carga en cada momento. Este comportamiento refleja que la CPU mantiene una participación activa a lo largo de toda la ejecución, desempeñando principalmente funciones de coordinación y gestión de los kernels en la GPU, en lugar de ejecutar directamente las tareas de cómputo intensivo.

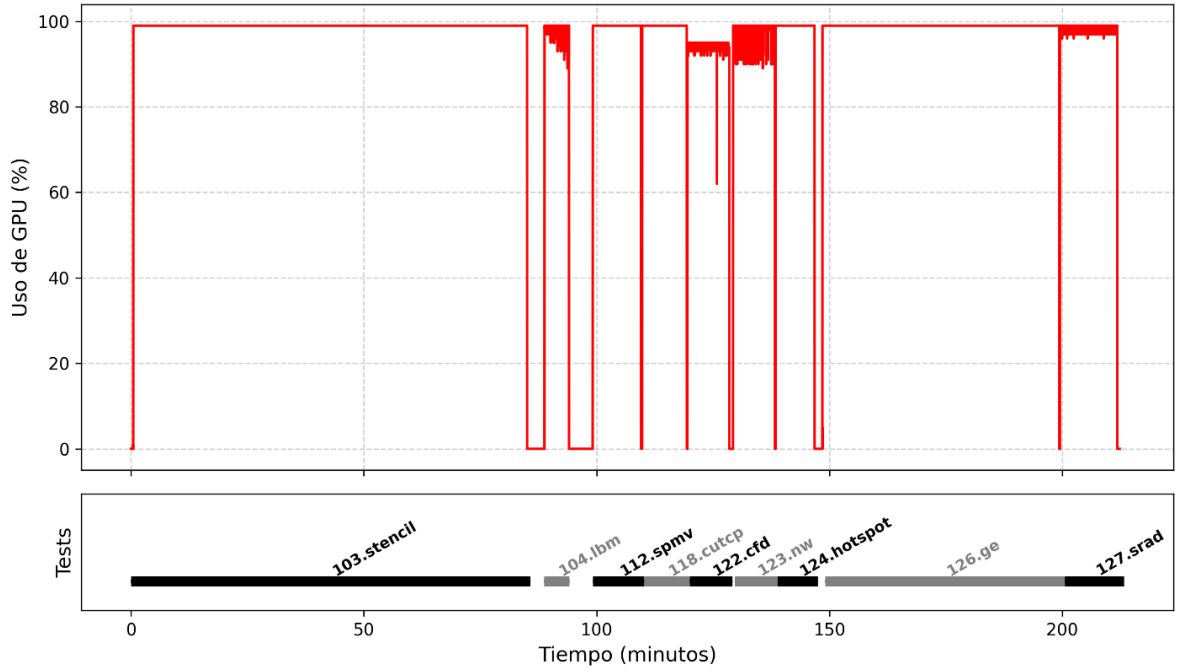


Fig. 24. Uso de la GPU durante ejecución de SPEC ACCEL.

En la **Fig. 24**, se muestra el uso de la GPU a lo largo del tiempo, alcanzando el 100 % en múltiples tramos que coinciden con la ejecución de los benchmarks exitosos. Se evidencian periodos de máxima ocupación separados por breves lapsos de inactividad, atribuibles a procesos de inicialización o finalización. Este patrón confirma que los kernels OpenCL fueron ejecutados de forma efectiva y que la GPU fue utilizada al máximo durante las fases de cómputo intensivo, consolidando su papel como componente central del rendimiento en esta suite.

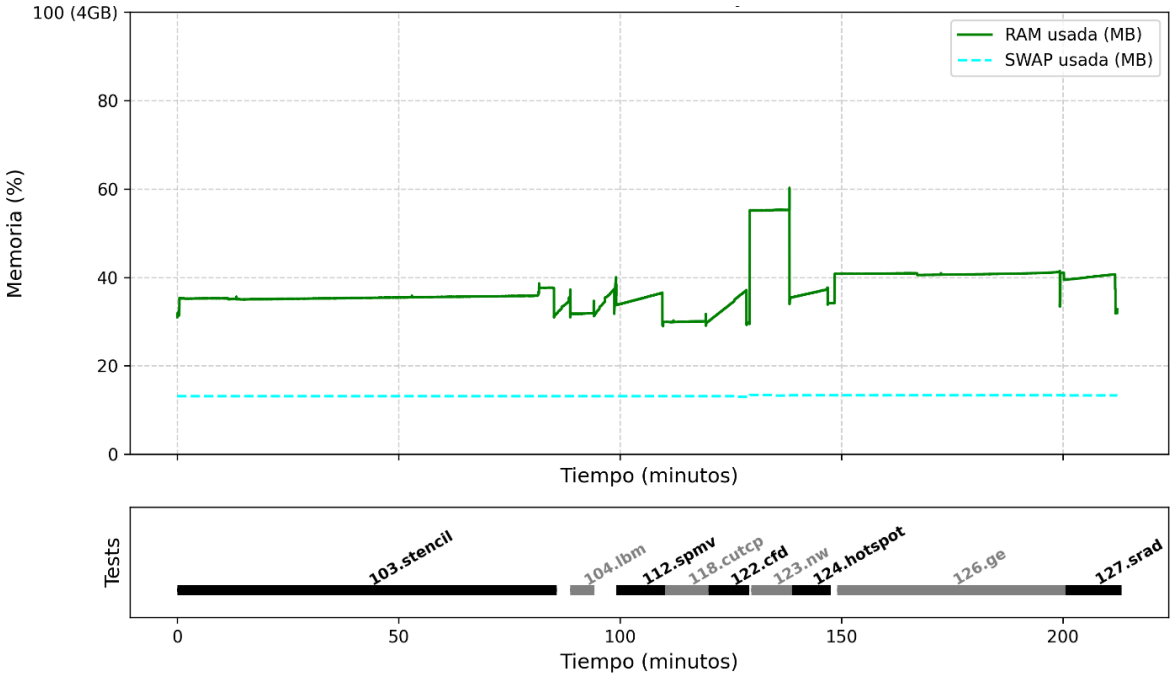


Fig. 25. Uso de memoria RAM y SWAP durante ejecución de SPEC ACCEL.

Complementariamente, la **Fig. 25** presenta el comportamiento del sistema en términos de uso de memoria RAM y SWAP. La utilización de RAM se mantiene relativamente estable, con valores que oscilan entre el 30 % y el 60 % de los 4 GB disponibles, y con picos puntuales durante pruebas como *123.nw*, lo que sugiere una mayor demanda de almacenamiento temporal en momentos específicos. Por otro lado, la memoria SWAP se mantiene constante alrededor del 13 %, lo que evidencia que no hubo presión significativa sobre la RAM que obligara al sistema a recurrir de forma intensiva al área de intercambio. En conjunto, estos resultados reflejan un uso eficiente de los recursos de memoria, sin síntomas de saturación que pudieran comprometer el rendimiento general.

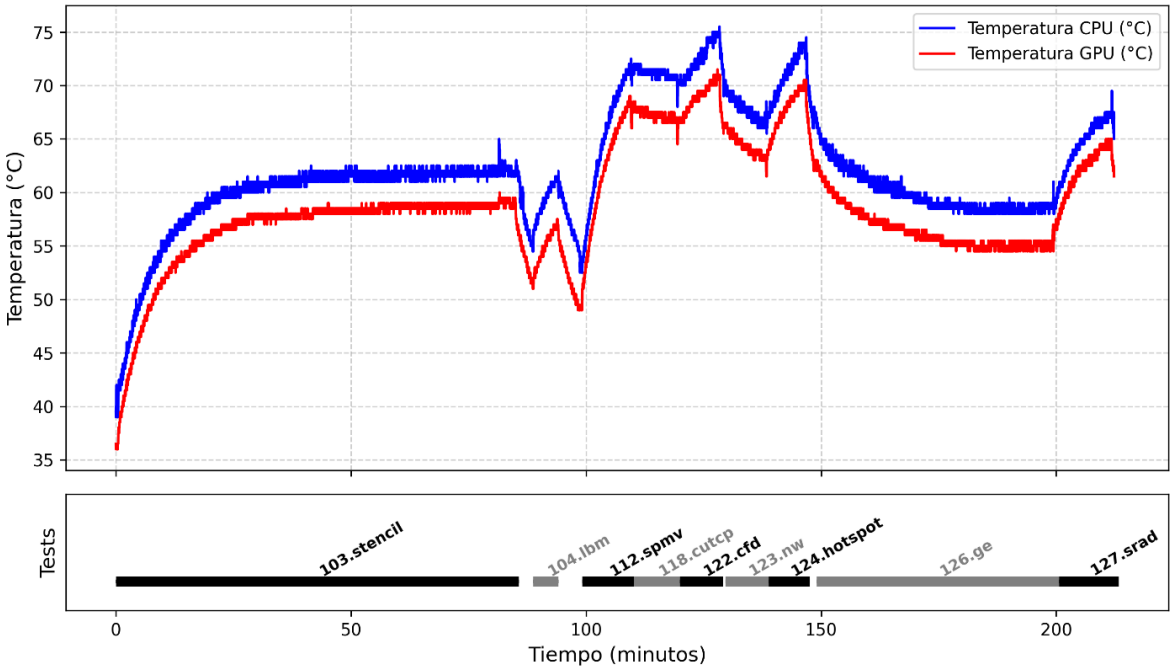


Fig. 26. Temperatura de la CPU durante ejecución de SPEC ACCEL.

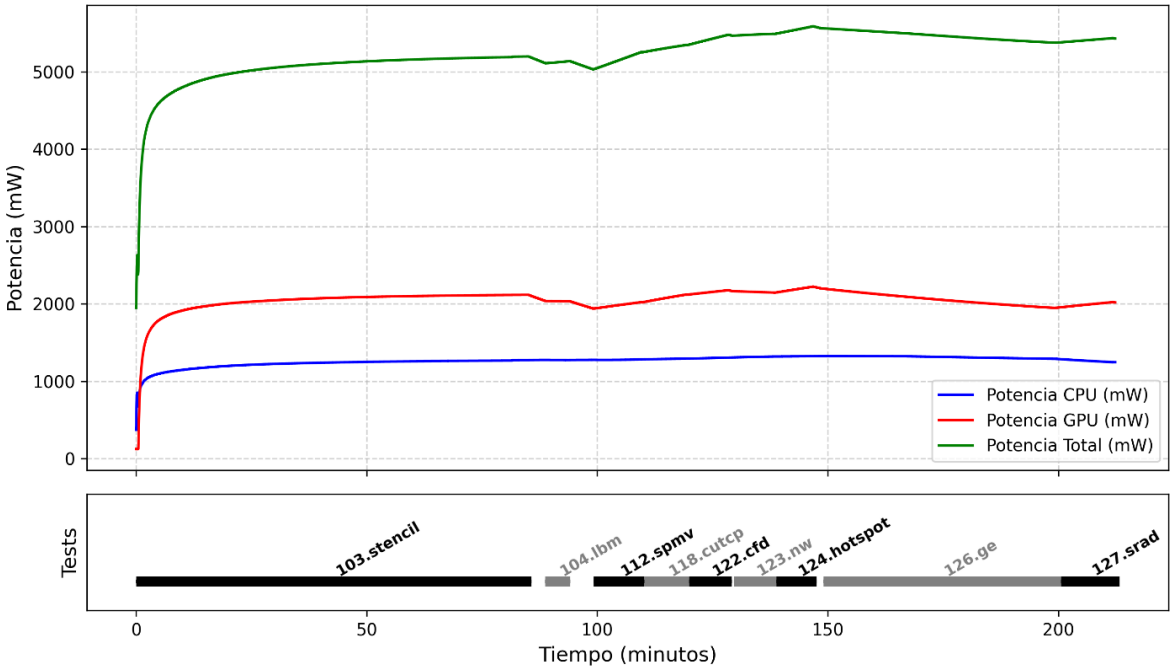


Fig. 27. Potencia promedio de la CPU, GPU y total durante ejecución de SPEC ACCEL.

En la **Fig. 26**, se muestra la evolución de la temperatura de la CPU y la GPU durante la ejecución de los benchmarks exitosos. Ambas alcanzan valores máximos cercanos a los 75 °C en los momentos de mayor carga, aunque se mantienen en general por debajo de este umbral. El perfil térmico evidencia una relación directa entre el uso intensivo de la GPU y los picos de temperatura observados. Algunas caídas abruptas en ambas curvas coinciden con los intervalos entre pruebas, reflejando breves periodos de inactividad o transición. En conjunto, este comportamiento sugiere una adecuada disipación térmica del sistema, sin indicios de sobrecalentamiento que comprometan su estabilidad.

Por su parte, la **Fig. 27** presenta el consumo promedio de potencia asociado a la CPU, la GPU y el sistema en su conjunto. Se observa que la GPU mantiene un consumo relativamente constante alrededor de los 2000 mW, mientras que la CPU oscila entre 1100 y 1300 mW. La potencia total alcanza valores entre los 5000 y 5500 mW, con pequeñas fluctuaciones a lo largo del tiempo. Esta distribución energética confirma que la GPU fue el componente dominante en términos de carga computacional, lo cual es coherente con el enfoque acelerado por hardware de la suite SPEC ACCEL OpenCL.

El análisis conjunto de las gráficas de uso de recursos permite concluir que la Jetson Nano respondió de forma eficiente ante las cargas de trabajo paralelas impuestas por los benchmarks de SPEC ACCEL. La GPU fue utilizada de manera intensiva durante los periodos de cómputo, mientras que la CPU mantuvo una actividad constante como coordinador de las tareas, con un consumo energético equilibrado entre ambos componentes. Las temperaturas se mantuvieron dentro de rangos operativos seguros, sin evidencias de reducción del rendimiento por sobrecalentamiento, y no se detectaron cuellos de botella significativos en el uso de memoria. Estos resultados confirman que, dentro de sus limitaciones, la Jetson Nano es capaz de ejecutar tareas aceleradas por hardware de forma estable y eficiente, siempre que las aplicaciones se encuentren bien adaptadas a su arquitectura y capacidad de procesamiento.

BENCHMARKS A NIVEL DE CLUSTER

SPEC MPI2007

Para evaluar el rendimiento del clúster de Jetson Nano en cargas de trabajo paralelizadas, se utilizó la suite SPEC MPI2007, un conjunto de pruebas diseñado para medir la eficiencia de sistemas distribuidos mediante el estándar MPI, donde los procesos se ejecutan en memoria separada y requieren mecanismos explícitos de comunicación. Esta suite está compuesta por benchmarks representativos de aplicaciones científicas reales, tales como dinámica de fluidos, modelado climático, simulaciones físicas y análisis estructural. Las pruebas se ejecutaron en modo base, utilizando 40 procesos MPI distribuidos entre los nodos del clúster. Inicialmente se consideraron los conjuntos de datos de referencia *medium* y *large* con el objetivo de analizar el comportamiento del sistema frente a distintos tamaños de problema. Sin embargo, debido a limitaciones técnicas propias de los recursos de cómputo disponibles en la Jetson Nano, no fue posible ejecutar satisfactoriamente el conjunto *large*, por lo que los resultados presentados corresponden únicamente al conjunto *medium*.

TABLA X
RESULTADOS DE LA SUITE SPEC MPI2007 CON DATASET *MEDIUM.*

Benchmark	Tiempo de ejecución (s)	Base Ratio	Resultado	Error
104.milc	656	2.38	OK	–
107.leslie3d	2345	2.23	OK	–
113.GemsFDTD	2154	2.93	OK	–
115.fds4	1102	1.77	OK	–
121.pop2	7637	0.541	OK	–
122.tachyon	1482	1.89	OK	–
126.lammps	1755	1.66	OK	–
127.wrf2	—	—	NR	Build error
128.GAPgeofem	961	2.15	OK	–
129.tera_tf	—	—	NR	Miscompare of output
130.socorro	—	—	NR	Segmentation fault
132.zeusmp2	1251	2.48	OK	–
137.lu	1448	2.54	OK	–

Como se muestra en la **TABLA X**, de los 13 benchmarks que conforman el dataset *medium* de la suite SPEC MPI2007, un total de 10 pruebas se completaron exitosamente en el clúster de Jetson Nano. Los tiempos de ejecución variaron entre 656 y 7637 segundos, mientras que los

valores de *Base Ratio* se ubicaron en un rango de 0.541 a 2.93. El mejor desempeño relativo se observó en *113.GemsFDTD* (2.93), seguido por *137.lu* (2.54) y *132.zeusmp2* (2.48), lo cual indica una buena adaptación del clúster a cargas de trabajo con alto grado de paralelismo.

Este rendimiento favorable se debe en parte a la diferencia en la cantidad de procesos MPI utilizados respecto a la máquina de referencia definida por SPEC. Mientras que dicha máquina ejecuta las pruebas con 16 procesos distribuidos en 8 nodos, en este caso se emplearon 40 procesos en 10 nodos. Esta mayor disponibilidad de procesos permitió compensar las limitaciones individuales de cada Jetson Nano, facilitando una distribución de carga más eficiente y mejorando el rendimiento en aquellas aplicaciones que logran beneficiarse del paralelismo distribuido.

A pesar de los resultados positivos en la mayoría de las pruebas, tres benchmarks no pudieron completarse satisfactoriamente, representando aproximadamente un 23 % del total. En el caso de *127.wrf2*, la prueba falló durante la compilación, probablemente debido a incompatibilidades con bibliotecas o configuraciones específicas del entorno. El benchmark *129.tera_tf* fue marcado como “Miscompare of output” al detectarse una discrepancia entre la salida esperada y la generada, posiblemente por diferencias en la arquitectura o errores de precisión numérica. Finalmente, *130.socorro* presentó un fallo de ejecución por “segmentation fault”, lo que podría estar asociado a accesos inválidos a memoria o falta de compatibilidad con la arquitectura ARM64.

En conjunto, los resultados obtenidos demuestran que, a pesar de tratarse de un clúster de bajo consumo energético y construido con hardware modesto, el sistema es capaz de ejecutar satisfactoriamente una parte significativa de la suite SPEC MPI2007, alcanzando incluso rendimientos superiores a los de la máquina de referencia en varios casos. Esto valida la viabilidad de utilizar plataformas basadas en SBC como entorno de experimentación para cargas de trabajo paralelas, siempre que se tenga en cuenta la naturaleza de las aplicaciones y las limitaciones propias de este tipo de arquitectura.

Durante la ejecución de la suite, la herramienta tegrastats corrió localmente en cada uno de los nodos del clúster, generando un total de 40 gráficas (4 por nodo). Con el fin de simplificar el

análisis, se observó que los nodos del 1 al 9 presentaron un comportamiento muy similar, por lo que se calculó el promedio de cada métrica entre estos nueve nodos, comparándolo luego con el nodo 10, el cual mostró diferencias significativas respecto al resto del clúster.

De esta forma, se presentan cuatro figuras que reúnen las métricas promedio de los nodos [1–9] y las del nodo 10, acompañadas de la línea de tiempo que indica los intervalos de ejecución de las pruebas. Estas comparaciones permiten evaluar el rendimiento del sistema desde una perspectiva global e individual, destacando aquellos puntos donde el nodo 10 se desvía del comportamiento uniforme esperado en el clúster.

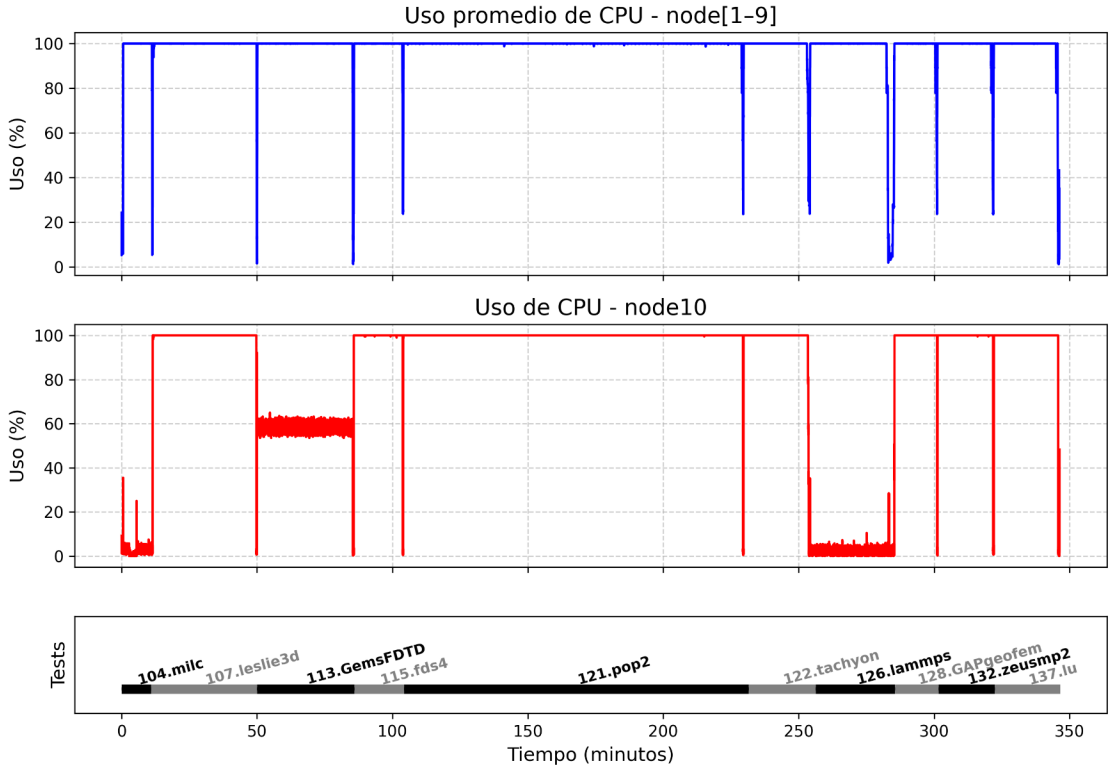


Fig. 28. Comparativa de uso de CPU entre nodos [1-9] y nodo 10 durante ejecución de SPEC MPI2007.

La **Fig. 28** muestra el uso promedio de CPU para los nodos del 1 al 9 y el uso individual del nodo 10; se promediaron los cuatro núcleos para simplificar la visualización. Se observa que, en promedio, los nodos del 1 al 9 mantuvieron un uso cercano al 100 % durante la mayor parte de la ejecución de las pruebas, presentando solo descensos puntuales coincidentes con los cambios de test. Por otro lado, el nodo 10 presentó un comportamiento claramente diferenciado, mostrando períodos prolongados de uso parcial del CPU: alrededor del 60 % durante la prueba

113.GemsFDTD y cercano al 0 % durante las pruebas *104.milc* y *126.lammps*. Este desempeño indica que el nodo 10 podría haber tenido una configuración ligeramente distinta o haber asumido un rol diferente durante la ejecución, lo cual explicaría la menor utilización de sus núcleos. Es posible, por ejemplo, que haya ejecutado el proceso maestro encargado de coordinar a los demás, el cual realiza menos trabajo computacional directo que los procesos esclavos, sin que esto represente necesariamente un cuello de botella significativo para el clúster.

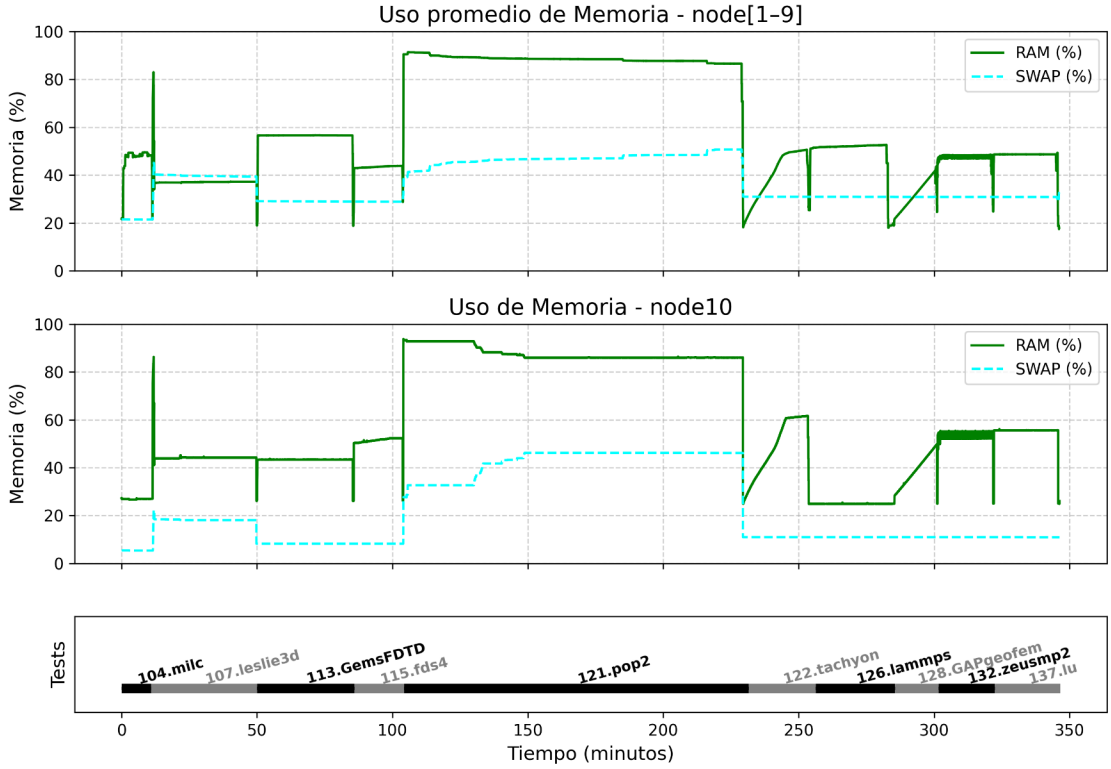


Fig. 29. Comparativa de uso de memoria entre nodos [1-9] y nodo 10 durante ejecución de SPEC MPI2007.

La **Fig. 29** del uso de memoria muestra una tendencia muy similar entre el promedio de los nodos [1–9] y el nodo 10, alcanzando valores máximos alrededor del 80–90 % para la RAM durante ciertos intervalos, especialmente durante la ejecución del test *121.pop2*. El uso de SWAP fue moderado en general, con incrementos puntuales en los momentos de máxima carga, indicando que algunos tests demandan recursos cercanos al límite disponible de RAM (4 GB). Cabe resaltar que, durante las pruebas *104.milc* y *126.lammps*, se observa una carga notablemente baja de memoria en el nodo 10, coincidiendo con los períodos de inactividad en la CPU para dicho nodo reportados previamente en la **Fig. 28**.

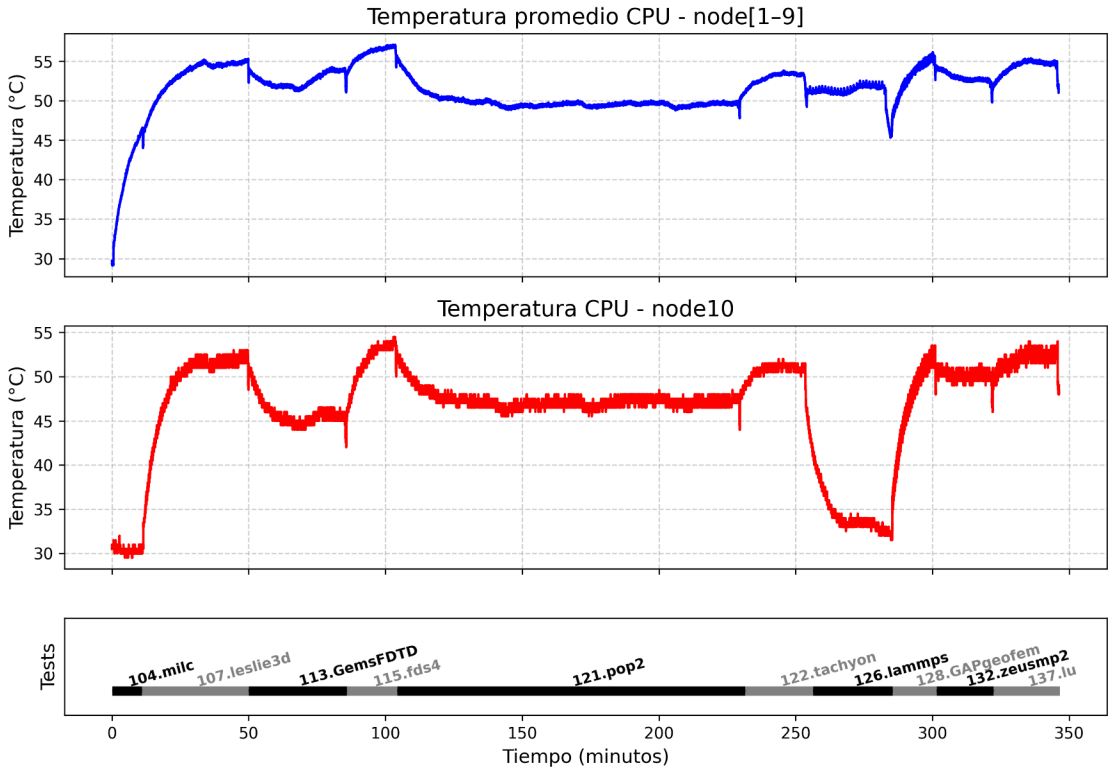


Fig. 30. Comparativa de temperatura de CPU entre nodos [1-9] y nodo 10 durante ejecución de SPEC MPI2007.

La **Fig. 30** muestra la temperatura promedio del CPU para los nodos [1–9] y el nodo 10, presentando una curva de calentamiento y estabilización bastante similar entre ambos casos, con temperaturas máximas cercanas a los 50–55 °C durante los períodos de mayor carga computacional. Sin embargo, es importante destacar que el nodo 10 presenta disminuciones notables en la temperatura del CPU durante los períodos de inactividad registrados previamente, particularmente en las pruebas *104.milc* y *126.lammps*. Esto es coherente con el bajo uso de CPU y memoria observado en estas pruebas específicas. En términos generales, el comportamiento térmico no revela problemas críticos ni situaciones de sobrecalentamiento en ninguno de los nodos, manteniéndose siempre dentro de los rangos esperados y seguros de operación para este tipo de hardware.

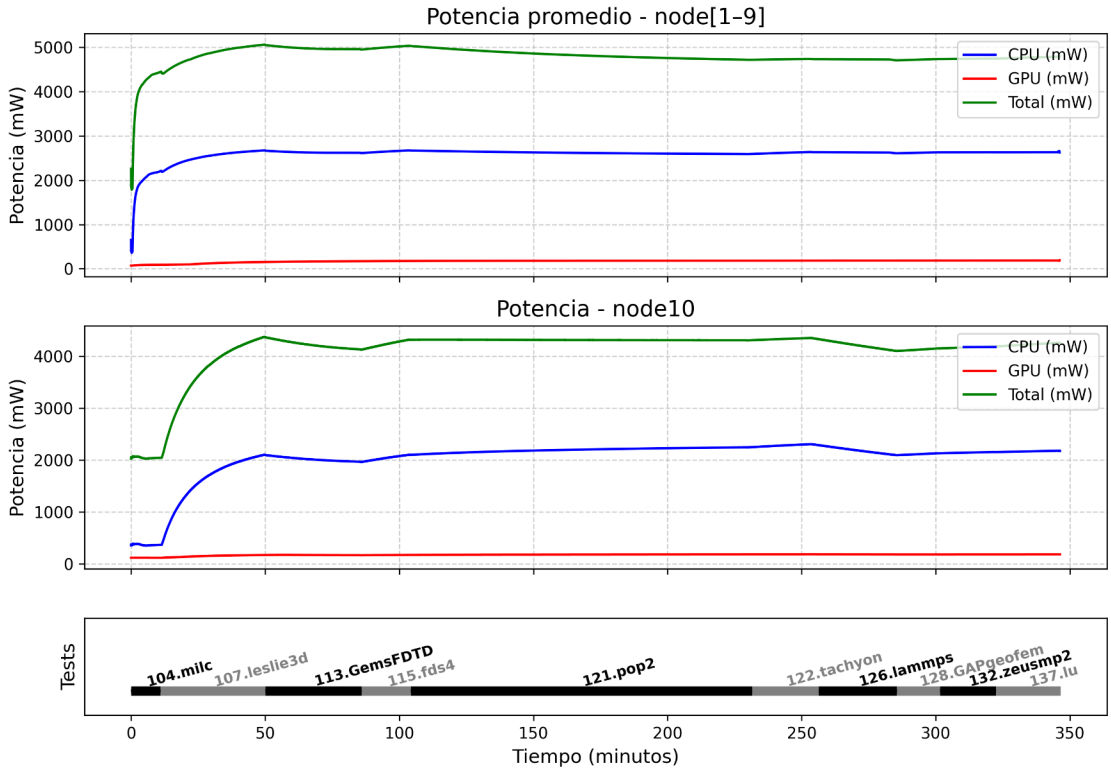


Fig. 31. Comparativa de potencia entre nodos [1-9] y nodo 10 durante ejecución de SPEC MPI2007.

Finalmente, en la **Fig. 31** se aprecia que el consumo de potencia total es relativamente similar entre el promedio de los nodos [1–9] y el nodo 10, estabilizándose en torno a los 4500–5000 mW. No obstante, es destacable que tanto la potencia total como la potencia consumida por el CPU fueron consistentemente mayores en el promedio de los nodos [1–9], con diferencias alrededor de 500–600 mW respecto al nodo 10. El consumo de GPU fue bajo y constante en ambos casos, puesto que estas pruebas no hacen uso del hardware gráfico disponible. El menor consumo del CPU observado en el nodo 10 correlaciona directamente con los períodos de bajo uso de CPU identificados previamente, reafirmando que la principal diferencia del nodo 10 frente al resto del clúster está asociada a la gestión y aprovechamiento del CPU durante la ejecución paralela de las pruebas.

HPL

Para evaluar el rendimiento máximo del clúster en operaciones de punto flotante, se utilizó el benchmark HPL, el cual permite cuantificar la capacidad de cómputo distribuido mediante la resolución de sistemas lineales densos. En esta sección se presentan los resultados obtenidos al modificar tres parámetros clave del benchmark: el tamaño de la matriz (N), el tamaño de bloque (NB) y la configuración de procesos (P×Q), con el fin de identificar la combinación que maximiza el rendimiento en términos de GFLOPs.

Se exploraron diversas combinaciones de estos valores, seleccionadas en función de las restricciones de memoria por nodo y del objetivo de equilibrar la carga computacional con la sobrecarga de comunicación entre procesos.

La **TABLA XI** resume los parámetros empleados y el tiempo requerido para ejecutar las pruebas del benchmark. Se llevaron a cabo dos pruebas: en la primera se utilizaron valores moderados de los parámetros NB y N. Tras verificar que el uso de memoria permitía incrementar el tamaño del problema, se realizó una segunda prueba con valores más exigentes. En ambos casos se mantuvo la misma configuración de procesos (P×Q).

HPL genera automáticamente todas las combinaciones posibles entre los parámetros definidos y ejecuta cada una 18 veces. En este caso, el número de valores configurados dio lugar a 48 combinaciones por prueba, resultando en un total de 864 ejecuciones por prueba.

TABLA XI
VALORES DE PARAMETROS PARA PRUEBAS HPL

Prueba	PxQ	NB	N	Duración prueba
1	[4x10] [10x4] [5x8] [8x5]	64 128 256	16000 20000 25000 32000	~32.86 hr.
2		128 256 384	32000 40000 48000 56000	~154.77 hr.

La selección de los valores de N respondió a la necesidad de aprovechar eficientemente la memoria RAM disponible. En la primera prueba se exploró un rango entre 16.000 y 32.000, mientras que en la segunda se amplió hasta 56.000. El incremento de N supone una mayor carga

computacional que, en general, mejora el rendimiento en GFLOPs, aunque también incrementa el uso de memoria y el tiempo de ejecución.

El tamaño de bloque (NB) se evaluó con valores representativos (64, 128, 256 y 384), lo que permitió analizar el equilibrio entre el cómputo local y la carga de comunicación entre procesos. Por su parte, se definieron cuatro configuraciones para la malla de procesos (P×Q) que satisfacen el uso total de los 40 procesos disponibles, priorizando combinaciones equilibradas (por ejemplo, 4×10 y 5×8) para favorecer una distribución homogénea de la carga y minimizar la latencia de comunicación.

La **TABLA XII** y **TABLA XIII** presentan las cinco combinaciones con mayor rendimiento promedio obtenidas en cada una de las pruebas realizadas. En ambos casos se observa que las mejores configuraciones corresponden a valores altos de N y combinaciones equilibradas de procesos (P×Q), lo que confirma la influencia positiva del tamaño del problema y la distribución eficiente de la carga sobre el desempeño del clúster.

TABLA XII
TOP 5 DE RESULTADOS EN PRUEBA 1 DE HPL

P	Q	NB	N	GFLOPs promedio
4	10	128	32000	102.45
4	10	256	32000	98.12
4	10	64	32000	97.08
5	8	128	32000	94.54
5	8	256	32000	90.95

TABLA XIII
TOP 5 DE RESULTADOS EN PRUEBA 2 DE HPL

P	Q	NB	N	GFLOPs promedio
4	10	128	56000	122.44
4	10	256	56000	119.87
4	10	128	48000	116.86
4	10	384	56000	115.92
5	8	128	56000	115.56

Al comparar los resultados de ambas pruebas, se evidencia una mejora significativa en el rendimiento al incrementar el tamaño del problema (N). La mejor configuración de la Prueba 1 alcanzó un promedio de 102.45 GFLOPs, mientras que en la Prueba 2 se obtuvo un máximo de 122.44 GFLOPs, lo que representa un incremento aproximado del 19.5% en el desempeño.

Este aumento se atribuye principalmente al uso de valores más altos de N en la segunda prueba (hasta 56.000), que permite una mayor explotación de la capacidad de cómputo del clúster. Además, se mantiene la tendencia observada en la Prueba 1, donde las combinaciones más equilibradas de procesos (P=4, Q=10 y P=5, Q=8) y valores intermedios o altos de NB (128 y 256) ofrecen un mejor rendimiento.

Estos resultados confirman que el dimensionamiento adecuado del problema, junto con una distribución eficiente de procesos, son factores determinantes para maximizar el rendimiento de HPL en arquitecturas limitadas como la utilizada.

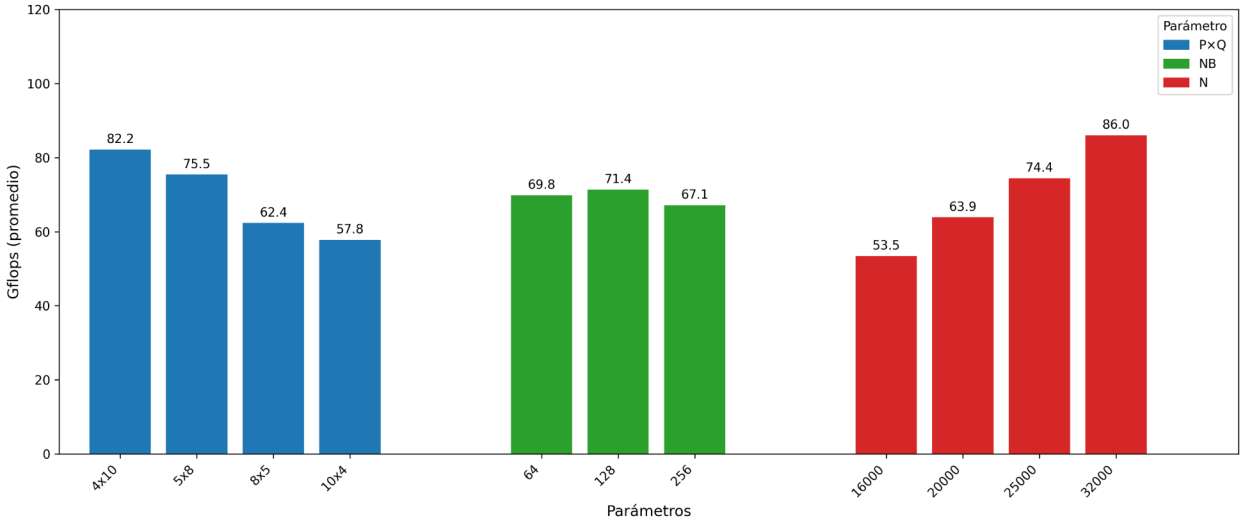


Fig. 32. Rendimiento promedio por parámetro en la Prueba 1 de HPL.

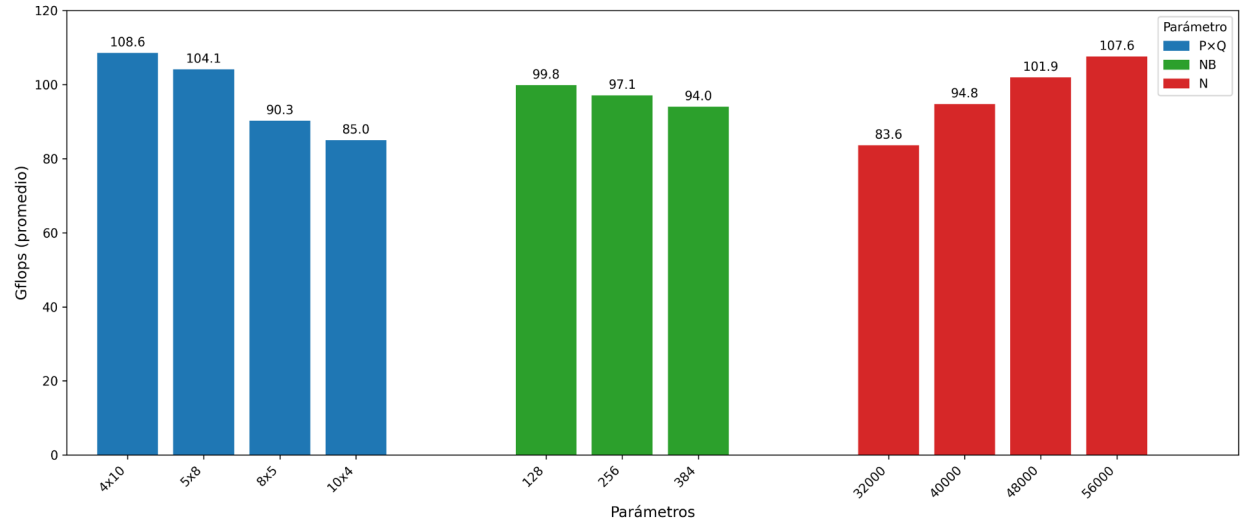


Fig. 33. Rendimiento promedio por parámetro en la Prueba 2 de HPL.

La **Fig. 32** y **Fig. 33** muestran el rendimiento promedio en GFLOPs obtenido al mantener fijo cada parámetro (P×Q, NB y N) y promediar los resultados correspondientes a todas sus combinaciones con los otros dos parámetros. Este enfoque permite aislar el efecto de cada parámetro sobre el rendimiento y facilita la identificación de tendencias generales en el comportamiento del sistema. En términos generales, se observa una mejora significativa en la segunda prueba, atribuible principalmente al aumento del tamaño del problema (N) y al ajuste de los valores de NB.

En ambas pruebas, las configuraciones más equilibradas, como [4×10] y [5×8], presentaron los mejores resultados. En la Prueba 1, [4×10] alcanzó un rendimiento promedio de 82.2 GFLOPs, mientras que en la Prueba 2 superó los 108 GFLOPs, lo que representa una mejora superior al 30 %. En contraste, combinaciones como [10×4] y [8×5], aunque equivalentes en número de procesos, mostraron un rendimiento considerablemente menor. Esta diferencia se explica por la forma en que los procesos se asignan a los nodos físicos del clúster: según la orientación de la malla, algunas configuraciones generan mayor tráfico de comunicación entre nodos distintos, lo que incrementa la latencia y reduce el rendimiento.

El parámetro NB mostró un comportamiento más estable en la Prueba 2, con resultados en un rango estrecho entre 94 y 100 GFLOPs para los valores evaluados (128, 256 y 384). Esto indica que, una vez alcanzado un tamaño de problema suficientemente grande, la sensibilidad al valor de NB disminuye. Por el contrario, en la Prueba 1 se observó una mayor dispersión, alcanzando un rendimiento máximo de 71.4 GFLOPs con NB=128 y bajando a 67.1 GFLOPs con NB=256, sugiriendo una penalización asociada a la sobrecarga de comunicación en problemas pequeños.

El impacto más evidente sobre el rendimiento lo tuvo el tamaño del problema (N). En la Prueba 1, el rendimiento aumentó progresivamente desde 53.5 GFLOPs (N=16000) hasta 86.0 GFLOPs (N=32000). Esta tendencia se acentuó en la Prueba 2, donde el aumento de N hasta 56000 elevó el rendimiento promedio hasta 107.6 GFLOPs. Estos resultados confirman que un mayor tamaño del sistema lineal permite una utilización más eficiente de los recursos de cómputo, al incrementar la proporción de operaciones útiles frente al costo asociado a la comunicación y sincronización entre procesos.

Se monitoreó el comportamiento de los nodos con la herramienta tegrastats durante la ejecución del benchmark. Aunque se capturaron múltiples métricas, aquí se muestran únicamente la **Fig. 34** y **Fig. 35**, relativas al uso de memoria y consumo de potencia durante la prueba 2. Además, se observó que la carga de la CPU se mantuvo al 100 % de manera continua, mientras que la temperatura osciló entre 60 °C y 75 °C, dentro de los rangos esperados para la Jetson Nano.

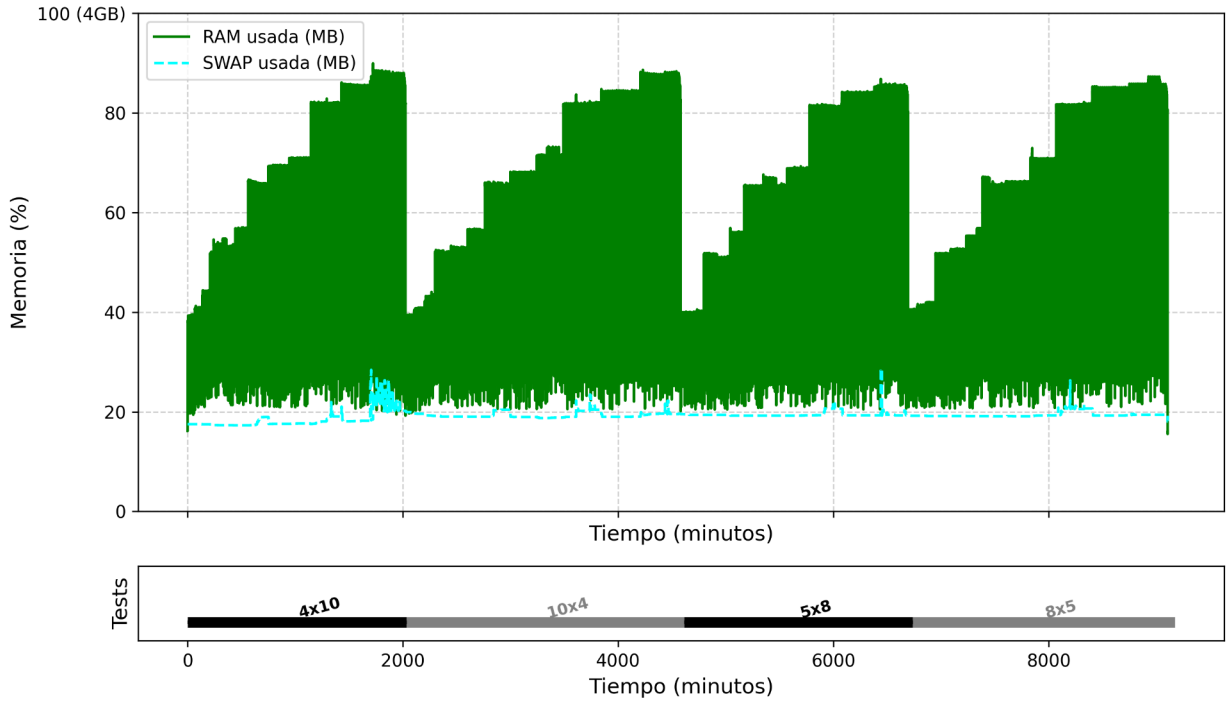


Fig. 34. Uso de memoria durante la ejecución de la prueba 2 del benchmark HPL.

En cuanto al uso de memoria (**Fig. 34**), se registró un consumo progresivo de la RAM, con ocupaciones superiores al 80 % en los tramos de mayor carga. Este incremento está ligado al aumento del parámetro N en las combinaciones evaluadas: problemas más grandes requieren más memoria. La memoria SWAP se utilizó solo marginalmente, lo que indica una gestión adecuada sin llegar a la saturación, confirmando que el dimensionamiento de las pruebas fue acertado para la capacidad de los nodos.

En lo referente al consumo de potencia (**Fig. 35**), la CPU mantuvo un nivel constante de uso energético, cercano a los 4500–5000 mW durante la mayor parte del tiempo, lo que refleja un uso sostenido del procesador sin interrupciones ni caídas significativas.

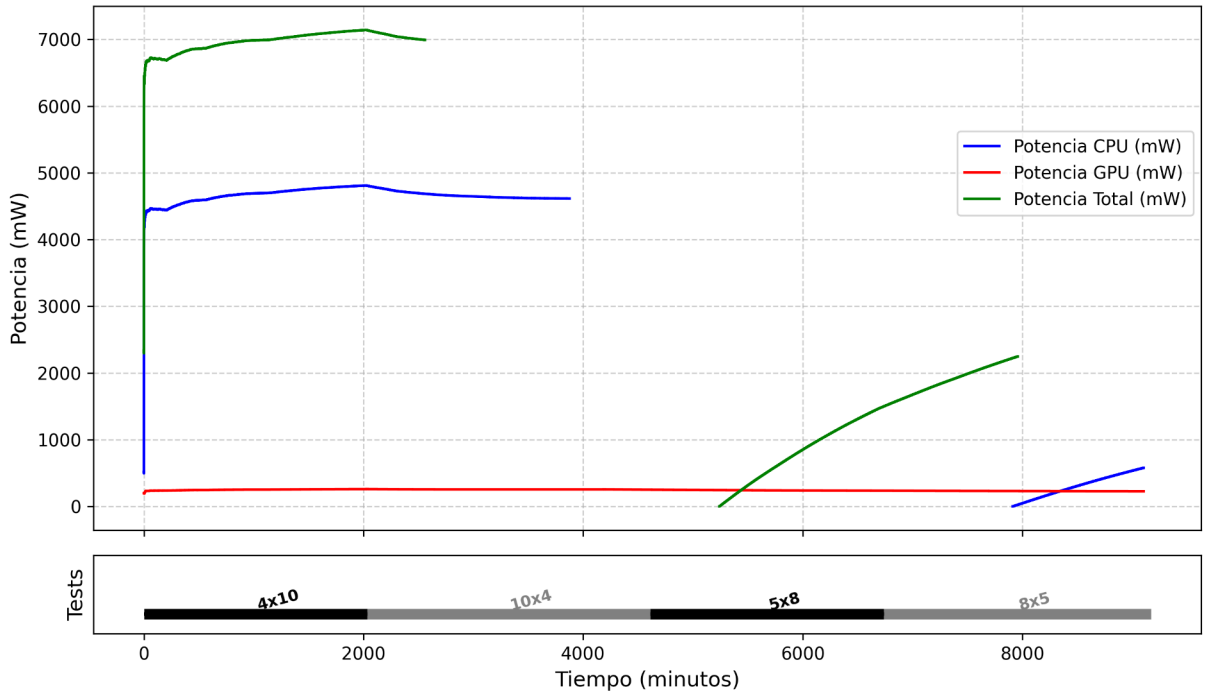


Fig. 35. Consumo de potencia durante la ejecución de la prueba 2 del benchmark HPL.

El consumo por parte de la GPU fue prácticamente nulo, como era esperado, ya que HPL no realiza operaciones aceleradas por GPU. Sin embargo, durante la ejecución del benchmark se detectó una anomalía en los valores de potencia reportados por la herramienta tegrastats: los registros comenzaron a mostrar valores negativos que aumentaban artificialmente. Esta irregularidad, visible en la gráfica a través de las líneas verde y azul, parece originarse en un error interno de lectura de la herramienta. Cabe resaltar que dicha anomalía no afectó el desarrollo de las pruebas ni comprometió los resultados obtenidos. Su impacto se limitó exclusivamente a la visualización de los datos de potencia, sin repercusiones sobre la validez de las métricas de rendimiento.

Las pruebas con HPL permitieron caracterizar el rendimiento del clúster y evidenciar el impacto de los parámetros de configuración sobre los GFLOPs obtenidos. El tamaño del problema (N) resultó ser el factor más determinante, seguido por la distribución de procesos (P×Q). Además, el monitoreo del sistema mostró un uso eficiente y estable de los recursos, sin incidencias relevantes. Estos resultados confirman la viabilidad del clúster Jetson Nano para tareas de cómputo intensivo, dentro de las limitaciones propias de su arquitectura.

VI. CONCLUSIONES

La implementación del clúster de Jetson Nano permitió construir una infraestructura HPC económica, modular y energéticamente eficiente, capaz de reproducir, a pequeña escala, prestaciones típicas de sistemas profesionales. El diseño y fabricación mediante impresión 3D del sistema de fijación modular posibilitó montar hasta diez tarjetas, gestionando eficientemente el uso del espacio en el rack y facilitando las conexiones de red y energía. La distribución eléctrica, basada en fuentes ATX y líneas centralizadas, garantizó una alimentación estable en modo 10 W sin fluctuaciones significativas de voltaje durante las pruebas realizadas.

La pila de software SLURM–Munge–PMIx–OpenMPI permitió la gestión de recursos y la ejecución paralela de tareas bajo un esquema multiusuario en un entorno distribuido. Las pruebas de desempeño mediante SPEC OMP2012, SPEC ACCEL, SPEC MPI2007 y HPL demostraron que, pese a las limitaciones inherentes a la memoria RAM y la CPU del Jetson Nano, el sistema alcanza rendimientos notables en tareas con alto grado de paralelismo, alcanzando hasta ≈120 GFLOPs en la prueba HPL. Estos resultados validan la viabilidad práctica del clúster para ejecutar aplicaciones paralelizadas, siempre que se respeten las restricciones técnicas propias de la plataforma empleada, como la capacidad limitada de memoria y la velocidad de conexión Ethernet.

En comparación con sistemas HPC tradicionales, el clúster implementado destaca por su bajo costo inicial y su alta eficiencia energética, lo que permite construir infraestructuras de cómputo paralelo más accesibles para entornos educativos y de investigación. La integración de una pila de software equivalente a la utilizada en sistemas de supercomputación y la exposición directa a herramientas reales de HPC, como SLURM, MPI y sistemas de archivos distribuidos, convierte esta plataforma en un recurso didáctico de gran valor para la enseñanza de programación paralela, administración de sistemas y clústeres, permitiendo familiarizarse con la gestión de colas de trabajos y la ejecución distribuida de aplicaciones paralelas en un entorno operativo realista.

Adicionalmente, la arquitectura modular tanto del sistema de distribución eléctrica como de los soportes 3D facilita la gestión de las tarjetas, permitiendo, por ejemplo, extraer nodos individuales para su integración en proyectos independientes orientados a *edge computing* o robótica, sin necesidad de rediseñar la estructura principal.

Sin embargo, también se identificaron algunas limitaciones relevantes. La necesidad de diseñar y fabricar componentes adicionales, como el sistema de fijación y la distribución eléctrica, implica un esfuerzo adicional de desarrollo que no está presente en soluciones comerciales preensambladas. Asimismo, el uso de SBCs como nodos introduce desafíos específicos en la instalación y configuración de software: las herramientas tradicionales de HPC no siempre ofrecen soporte directo para arquitecturas ARM, lo que puede requerir adaptaciones o compilaciones manuales.

Desde el punto de vista del hardware, las Jetson Nano presentan limitaciones en memoria y en frecuencia de reloj frente a procesadores utilizados en clústeres comerciales, afectando el desempeño en aplicaciones más exigentes. Además, la conexión mediante Ethernet Gigabit, aunque adecuada para cargas moderadas, introduce latencias y cuellos de botella en aplicaciones que requieren una comunicación intensiva, donde serían preferibles interconexiones de mayor velocidad como 10 GbE o Infiniband, comunes en clústeres HPC tradicionales.

De manera general, el clúster de Jetson Nano desarrollado constituye una infraestructura HPC de bajo costo, modular y energéticamente eficiente, capaz de reproducir entornos de cómputo paralelo representativos a pequeña escala. Su equilibrio entre accesibilidad económica, eficiencia operativa y realismo en la gestión de tareas distribuidas lo convierte en una alternativa valiosa para la enseñanza práctica y para proyectos de investigación aplicada. En conjunto, el proceso técnico descrito y los resultados alcanzados aportan un significativo valor académico, lo que consolida la relevancia del presente trabajo como referencia para proyectos posteriores que busquen explorar soluciones similares utilizando SBCs.

REFERENCIAS

[1] M. Nossokoff, T. Sorensen, J. Ludema, M. Riddle, B. Sorensen, and E. Joseph, *Top Predictions for the Global HPC Community in 2024*, Hyperion Research, Special Analysis Report #HR126.0456.01.12.2024, Jan. 2024. [Online]. Available: https://hyperionresearch.com/wp-content/uploads/2024/02/Hyperion-Research-Special-Analysis-2024-Top-HPC-Predictions-January-2024-1.pdf

[2] IBM, "High performance computing (HPC)," IBM, [Online]. Available: https://www.ibm.com/topics/hpc. [Accessed: Apr. 2, 2025].

[3] Dell EMC HPC Engineering, *Design Principles for HPC*, Dell EMC, Technical White Paper, Version 1.0, Jan. 2018. [Online]. Available: https://dl.dell.com/manuals/all-products/esuprt_solutions_int/esuprt_solutions_int_solutions_resources/high-computing-solution-resources_white-papers48_en-us.pdf

[4] NVIDIA, "¿Qué es la computación con GPU?", NVIDIA, [Online]. Available: https://www.nvidia.com/es-la/drivers/what-is-gpu-computing/. [Accessed: Apr. 2, 2025].

[5] Pure Storage, "What is an HPC cluster?", Pure Storage, [Online]. Available: https://www.purestorage.com/knowledge/what-is-an-hpc-cluster.html. [Accessed: Apr. 2, 2025].

[6] Supermicro, "What is a High-Performance Computing Cluster (HPC)?," Supermicro, [Online]. Available: https://www.supermicro.com/en/glossary/hpc-clusters. [Accessed: Apr. 2, 2025].

[7] Donostia International Physics Center, "HPC system overview," DIPC Technical Documentation, [Online]. Available: https://scc.dipc.org/docs/general/overview/. [Accessed: Apr. 2, 2025].

[8] T. Sterling, M. Anderson, and M. Brodowicz, *High Performance Computing: Modern Systems and Practices*. San Francisco, CA, USA: Morgan Kaufmann, 2018.

[9] P. Vicat-Blanc, S. Soudan, R. Guillier, and B. Goglin, *Computing Networks: From Cluster to Cloud Computing*. Hoboken, NJ, USA: Wiley-ISTE, 2011. [Online]. Available: https://doi.org/10.1002/9781118602003

[10] G. F. Pfister, *In Search of Clusters*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 1998.

[11] Hewlett Packard Enterprise, "What is an HPC Cluster?", HPE Glossary, [Online]. Available: https://www.hpe.com/emea_europe/en/what-is/hpc-clusters.html. [Accessed: Apr. 2, 2025].

[12] D. H. Ahn, J. E. Garlick, M. A. Grondona, D. A. Lipari, and R. R. Springmeyer, *A High Performance Computing Scheduling and Resource Management Primer*, Lawrence Livermore National Laboratory, Livermore, CA, USA, Rep. LLNL-TR-652476, Mar. 31, 2014.

[13] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux Utility for Resource Management," in *Job Scheduling Strategies for Parallel Processing*, vol. 2862, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer, 2003, pp. 44–60. [Online]. Available: https://doi.org/10.1007/10968987_3

[14] Altair Engineering, "PBS Professional Documentation," 2023. [Online]. Available: https://www.altair.com/pbs-professional/. [Accessed: Apr. 2, 2025].

[15] Open MPI Project, "Open MPI: Open Source High Performance Computing," [Online]. Available: https://www.open-mpi.org/. [Accessed: Apr. 2, 2025].

[16] OpenMP Architecture Review Board, "OpenMP Application Programming Interface," Version 5.1, Nov. 2020. [Online]. Available: https://www.openmp.org/specifications/. [Accessed: Apr. 2, 2025].

[17] OpenLDAP Project, "LDAP System Architecture," [Online]. Available: https://www.openldap.org/. [Accessed: Apr. 2, 2025].

[18] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed., Morgan Kaufmann, 2013.

[19] L. Alvarez, E. Ayguadé, and F. Mantovani, "Teaching HPC Systems and Parallel Programming with Small-Scale Clusters," in *Proc. IEEE/ACM Workshop on Education for High-Performance Computing (EduHPC)*, Dallas, TX, USA, 2018, pp. 1–10. [Online]. Available: https://doi.org/10.1109/EduHPC.2018.00004

[20] Techopedia, "Single Board Computer (SBC)," *Techopedia*, [Online]. Available: https://www.techopedia.com/definition/9266/single-board-computer-sbc. [Accessed: Apr. 2, 2025].

[21] P. Abrahamsson *et al.*, "Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment," in *Proc. 2013 IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Bristol, UK, 2013, pp. 170–175. [Online]. Available: https://doi.org/10.1109/CloudCom.2013.121

[22] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures," in *Proc. 2013 IEEE 33rd Int. Conf. Distributed Comput. Syst. Workshops (ICDCSW)*, Philadelphia, PA, USA, 2013, pp. 108–112. [Online]. Available: https://doi.org/10.1109/ICDCSW.2013.25

[23] R. A. Velásquez, S. Isaza, E. Montoya, L. G. García, and J. Gómez, "Embedded cluster platform for a remote parallel programming lab," in *Proc. 2020 IEEE Global Eng. Educ. Conf. (EDUCON)*, Porto, Portugal, 2020, pp. 763–772. [Online]. Available: https://doi.org/10.1109/EDUCON45650.2020.9125270

[24] J. Layton, "Small'Board Computers," *ADMIN Magazine*, no. 25, 2015. [Online]. Available: https://www.admin-magazine.com/Archive/2015/25/Small-board-computers. [Accessed: Apr. 2, 2025].

[25] S. Bourhnane, M. R. Abid, K. Zine-Dine, N. Elkamoun, and D. Benhaddou, "Cluster of Single-Board Computers at the Edge for Smart Grids Applications," *Applied Sciences*, vol. 11, no. 22, p. 10981, Nov. 2021. [Online]. Available: https://doi.org/10.3390/app112210981

[26] NVIDIA Corporation, "Jetson Nano Developer Kit," [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit. [Accessed: Apr. 2, 2025].

[27] Seeed Studio, "NVIDIA Jetson Nano vs Raspberry Pi: Which One is Better for Your Project?," [Online]. Available: https://www.seeedstudio.com/blog/2020/01/16/nvidia-jetson-nano-vs-raspberry-pi/. [Accessed: Apr. 2, 2025].

[28] M. Riedl, "Building a Jetson Nano GPU Cluster," *michaelriedl.com*, Apr. 27, 2023. [Online]. Available: https://michaelriedl.com/2023/04/27/tp2-gpu-cluster.html

[29] S. Shahizat, "How to Build NVIDIA Jetson HPC Cluster Using SLURM," *Hackster.io*, [Online]. Available: https://www.hackster.io/shahizat/how-to-build-nvidia-jetson-hpc-cluster-using-slurm-ed61a7. [Accessed: Apr. 2, 2025].

[30] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Waltham, MA, USA: Morgan Kaufmann, 2012.

[31] J. Dongarra, P. Luszczek, and A. Petitet, "The LINPACK Benchmark: Past, Present and Future," *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, 2003. [Online]. Available: https://doi.org/10.1002/cpe.728

[32] SPEC – Standard Performance Evaluation Corporation, "Benchmark Suites," [Online]. Available: https://www.spec.org/. [Accessed: Apr. 2, 2025].

[33] J. L. Aguilar and E. Leiss, *Introducción a la Computación Paralela*, 1ra ed., Mérida, Venezuela: CEMISID, 2004. [Online]. Available: https://gc.scalahed.com/recursos/files/r161r/w25041w/introduccionalacomputacionparalela_S5.pdf. [Accessed: Apr. 2, 2025].

ANEXOS

Los siguientes anexos técnicos complementan la información presentada en este informe y se encuentran organizados en el siguiente repositorio público:

https://github.com/JoseMJaramilloS/informe-anexos

Cada anexo corresponde a un archivo individual en formato Markdown (.md) y está identificado por la letra con la que se hace referencia a lo largo del documento. La estructura es la siguiente:

Anexo A. Sistema operativo y modo *headless*
Anexo B. Configuración de red
Anexo C. Sistema de archivos y almacenamiento compartido
Anexo D. Configuración de cliente LDAP
Anexo E. Instalación de Munge
Anexo F. Instalación de PMIx
Anexo G. Instalación de SLURM
Anexo H. Instalación de OpenMPI
Anexo I. Herramientas de rendimiento
Anexo J. Pruebas SLURM y OpenMPI
Anexo K. Benchmark SPEC OMP2012
Anexo L. Benchmark SPEC ACCEL
Anexo M. Benchmark SPEC MPI2007
Anexo N. Benchmark HPL
Anexo O. Herramientas de monitoreo

Departamento de Ingeniería Electrónica y de Telecomunicaciones

Implementación de un cluster HPC utilizando single-board computers

UNIVERSIDAD DE ANTIOQUIA
1803
Facultad de Ingeniería

PRACTICANTE: Jose Miguel Jaramillo Sánchez
ASESORES: Sebastián Isaza Ramírez
PROGRAMA: Ingeniería Electrónica
Semestre de la práctica: 2024-2

Objetivos

- Diseñar y prototipar un soporte para las Jetson Nano mediante impresión 3D, logrando un uso eficiente del espacio en el rack y facilitando su montaje y conexión.
- Desarrollar un sistema de distribución de energía modular y eficiente, capaz de alimentar múltiples Jetson Nano y cubrir la demanda eléctrica del clúster.
- Implementar un sistema de gestión de colas que permita la ejecución de tareas en paralelo, junto con el software necesario para el acceso multiusuario al clúster.
- Evaluar el desempeño del clúster mediante programas de benchmarking que aprovechen sus capacidades de cómputo paralelo y procesamiento acelerado.

Introducción

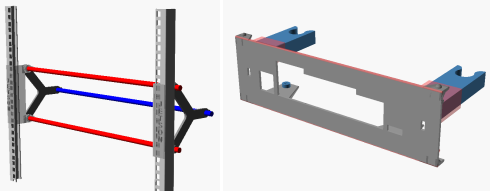
El aumento en el uso de sistemas HPC en investigación e industria ha impulsado también la demanda de expertos en estas infraestructuras. Dado su alto costo, se propone una alternativa asequible basada en SBC, que facilita el acceso al cómputo paralelo en contextos educativos y profesionales.

- **HPC (High Performance Computing):** técnicas y arquitecturas para procesar grandes volúmenes de datos o tareas complejas a alta velocidad, usando recursos paralelos o distribuidos.
- **Cluster:** conjunto de nodos interconectados que cooperan para ejecutar tareas en paralelo, funcionando como un solo sistema lógico y aumentando la capacidad de procesamiento.
- **Gestor de colas:** software que administra y distribuye tareas en un cluster (p. ej. SLURM), asignando recursos según prioridades y disponibilidad.
- **Computación paralela:** divide un problema en subprocesos simultáneos en varios núcleos o nodos, coordinados con herramientas como **MPI (Message Passing Interface).**
- **SBC (Single Board Computer):** computadoras completas en placa única (CPU, memoria, puertos). Destacan por bajo costo y consumo energético.

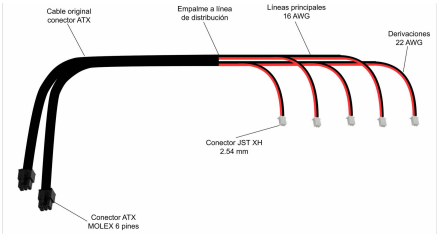
- **CPU:** Quad-core ARM Cortex-A57 (1.43 GHz).
- **GPU:** NVIDIA Maxwell, 128 núcleos CUDA.
- **RAM:** 4 GB LPDDR4
- **Consumo:** 5W -10W
- **Interfaz Red:** Ethernet Gigabit

Metodología

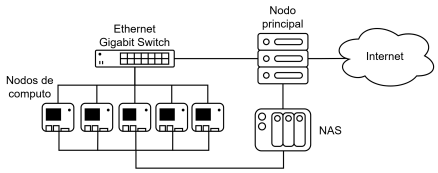
Diseño de soporte 3D



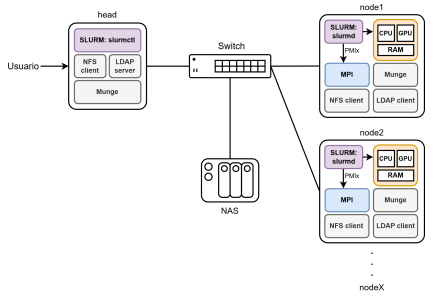
Sistema de distribución eléctrico



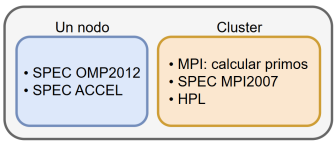
Arquitectura de red



Arquitectura de software

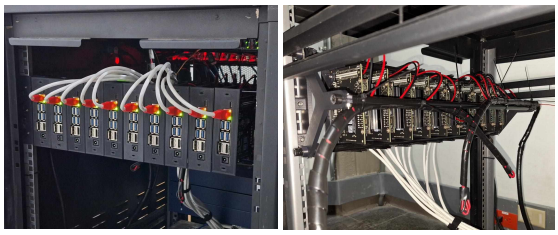


Evaluación de desempeño



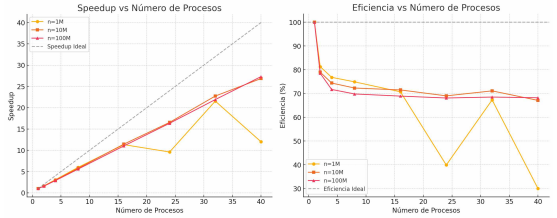
Resultados

Montaje final



Escalabilidad del cluster

Cálculo de números primos con MPI



Benchmarks SPEC y HPL

Suite	Test Exitosos	Base Ratio Promedio	Mejor Base Ratio
SPEC OMP2012	5/14	0.173	0.411
SPEC ACCEL	9/19	0.179	0.344
SPEC MPI2007	10/13	2.057	2.93

En HPL, el clúster alcanzó un rendimiento máximo de 122 GFLOPs utilizando 40 procesos y únicamente las CPUs de las Jetson Nano.

Conclusiones

- Se implementó un clúster HPC de bajo costo y alta eficiencia energética, utilizando diez Jetson Nano, soportes impresos en 3D y un sistema de distribución eléctrica centralizada.
- La pila de software permitió reproducir, a pequeña escala, prestaciones típicas de sistemas HPC profesionales, habilitando la ejecución paralela de aplicaciones y alcanzando un desempeño notable pese a las limitaciones de memoria, CPU y comunicación Ethernet propias de las Jetson Nano.
- La implementación del clúster, basada en un bajo costo inicial y en un entorno operativo real de HPC, constituye una estrategia didáctica valiosa para la enseñanza de programación paralela, administración de clústeres y operación de sistemas distribuidos.
- En comparación con sistemas HPC tradicionales, un clúster basado en SBC requiere el diseño y fabricación de componentes mecánicos y eléctricos adicionales, lo que implica un esfuerzo de implementación extra, además de no alcanzar los niveles de rendimiento característicos de plataformas de alto desempeño.

DATOS DE CONTACTO DEL AUTOR:

+57 317 2140224 josem.jaramillo@udea.edu.co https://www.linkedin.com/in/josem-jaramillos/