# ESTUDIO DEL DESPLIEGUE Y EJECUCIÓN DE APLICACIONES SOBRE ARQUITECTURAS POST-MOORE

**PABLO JOSUE ROJAS YEPES**

**UNIVERSIDAD INDUSTRIAL DE SANTANDER**
**FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS**
**ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**
**MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA**
**BUCARAMANGA**
**2022**

# ESTUDIO DEL DESPLIEGUE Y EJECUCIÓN DE APLICACIONES SOBRE ARQUITECTURAS POST-MOORE

**PABLO JOSUE ROJAS YEPES**

**Trabajo de grado para optar al título de Magíster en Ingeniería de Sistemas e Informática**

**Director**
**CARLOS JAIME BARRIOS HERNÁNDEZ**
**Doctor en Informática**

**Codirector**
**LUIZ ANGELO STEFFENEL**
**Doctor en Informática**

**UNIVERSIDAD INDUSTRIAL DE SANTANDER**
**FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS**
**ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA**
**MAESTRÍA EN INGENIERÍA DE SISTEMAS E INFORMÁTICA**
**BUCARAMANGA**
**2022**

## DEDICATORIA

Lleno de agradecimiento dedico este trabajo y cada uno de sus resultados a mis seres queridos, quienes han sido fundamentales para su desarrollo y culminación. Es para mí una gran satisfacción poder dedicárselo a ellos, que con mucho esfuerzo, esmero y trabajo se lo han ganado.

A mis padres Pablo y Consuelo, por ser un faro moral y ético al que he apuntado en mi vida. A mi hermana Paola, por ser un apoyo incondicional en todo este proceso. Y sin dejar de lado a todos los compañeros del grupo de investigación. Al profesor Carlos y Luiz por la oportunidad de investigar en temas que están a la vanguardia del desarrollo tecnológico del país.

**CONTENIDO**

**Pág.**

## LISTA DE FIGURAS

**Pág.**

## LISTA DE TABLAS

**Pág.**

## RESUMEN

**TÍTULO:** ESTUDIO DEL DESPLIEGUE Y EJECUCIÓN DE APLICACIONES SOBRE ARQUITECTURAS POST-MOORE[*]

**AUTOR:** PABLO JOSUE ROJAS YEPES, CARLOS JAIME BARRIOS HERNÁNDEZ, LUIZ ANGELO STEFFENEL[**]

**PALABRAS CLAVE:** POST-MOORE, DISPOSITIVOS EMBEBIDOS, BENCHMARK

**DESCRIPCIÓN:** la mejora en las capacidades computacionales durante las últimas décadas se ha basado en la ley de Moore, pero debido a varios factores ha perdido su validez. Para seguir avanzando, se han propuesto varias soluciones para abordar estas falencias, esto ha generado la aparición y rápida diversificación de dispositivos. Estos nuevos dispositivos usan las ventajas de la ley de Moore en nuevos enfoques para superar los límites de la ley, recibiendo el nombre de Post-Moore.

Dichos dispositivos Post-Moore ofrecen capacidades de rendimiento que merecen ser estudiadas mediante el despliegue y ejecución de aplicaciones. Además de los dispositivos, las necesidades de las aplicaciones limitan la elección de los dispositivos, ya que deben cubrir sus necesidades.

Este estudio identifica las características que tienen estos dispositivos para ser etiquetados como Post-Moore. Con las características propuestas se eligen dispositivos que cubren las necesidades de las aplicaciones. A estos dispositivos y aplicaciones se les establecen estrategias de despliegue que usan lineamientos que procuran minimizar el impacto en el rendimiento computacional. Además, se dispone de mecanismos de evaluación que permiten medir la escalabilidad, portabilidad, eficiencia e impacto en el rendimiento según el despliegue de las aplicaciones.

Al final, usando los dispositivos, aplicaciones, estrategias de despliegue, lineamientos y mecanismos de evaluación, se mide el impacto en el rendimiento computacional. La escalabilidad, eficiencia e impacto en el rendimiento se grafican como resultados comparativos según su estrategia de despliegue. Basado en los resultados se avalan los lineamientos propuestos para el despliegue de las aplicaciones. Esto permite una caracterización de la eficiencia computacional de estos dispositivos.


[*] Trabajo de grado

[**] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Maestría en Ingeniería de Sistemas e Informática. Director: Carlos Jaime Barrios Hernández, Ph.D. Codirector: Luiz Angelo Steffenel, Ph.D.

## ABSTRACT

**TITLE:** STUDY OF THE DEPLOYMENT AND EXECUTION OF APPLICATIONS ON POST-MOORE ARCHITECTURES*

**AUTHOR**: PABLO JOSUE ROJAS YEPES, CARLOS JAIME BARRIOS HERNÁNDEZ, LUIZ ANGELO STEFFENEL**

**KEY WORDS:** POST-MOORE, EMBEDDED DEVICES, BENCHMARK

**DESCRIPTION:** The improvement in computational capabilities during the last decades has been based on Moore's law, but due to various factors it has lost its validity. To continue advancing, several solutions have been proposed to address these shortcomings, this has led to the emergence and rapid diversification of devices. These new devices use the advantages of Moore's law in new approaches to overcome the limits of the law, receiving the name Post-Moore.

Such Post-Moore appliances offer performance capabilities that deserve to be explored when deploying and running applications. In addition to devices, the needs of applications limit the choice of devices, as they must meet your needs.

This study identifies the characteristics that these devices have to be labeled as Post-Moore. With the proposed characteristics devices are chosen that meet the needs of the applications. Deployment strategies are established for these devices and applications that use guidelines that seek to minimize the impact on computational performance. In addition, evaluation mechanisms are available to measure scalability, portability, efficiency and impact on performance according to the deployment of the applications.

Ultimately, using the devices, applications, deployment strategies, guidelines, and evaluation mechanisms, the impact on computational performance is measured. Scalability, efficiency, and performance impact are graphed as comparative results based on your deployment strategy. Based on the results, the proposed guidelines for the deployment of the applications are endorsed. This allows a characterization of the computational efficiency of these devices.


* Degree work

** Faculty of Physical-Mechanical Engineering. School of Systems and Computer Engineering. Master's degree in Systems and Computer Engineering. Director: Carlos Jaime Barrios Hernández, Ph.D. Co-director: Luiz Angelo Steffenel, Ph. D.

# INTRODUCCIÓN

Durante los últimos cuarenta años la mejora en las capacidades computacionales ha dependido en gran manera de la ley de Moore[1]. Dicha propuesta sugiere que cada dieciocho (18) meses las capacidades computacionales se duplicarán[2] mejorando el rendimiento computacional. Debido a problemas como la cantidad de transistores, el estancamiento en el incremento de las frecuencias de reloj o el consumo energético de los chips (W) se considera que ha perdido su efecto desde hace más de una década.

Esta pérdida plantea nuevos retos como el uso de: multinúcleos para aprovechar el paralelismo, sistemas computacionales heterogéneos, computación neuromórfica o computación cuántica para cubrir las necesidades específicas de las aplicaciones. A esta actualidad se le denomina como era Post-Moore[3]. La era Post-Moore ha provocado una "explosión cámbrica"[4], generando una mejora repentina de capacidades computacionales y rápida diversificación del hardware. De esta manera se plantean nuevos fundamentos en torno a la organización, diseño de tecnologías y abstracciones arquitecturales de lo que es un sistema computacional.

[1] «Foro Histórico de las Telecomunicaciones,» 2019. [En línea]. Available: https://forohistorico.coit.es/index.php/personajes/personajes-internacionales/item/moore-gordon-e. [Último acceso: 6 Febrero 2021]

[2] KUSZYK, A. y HAMMOUDEH, M. «Contemporary Alternatives to Traditional Processor Design in the Post Moore's Law Era,» *ICFNDS'18: International Conference on Future Networks and Distributed Systems,* pp. 1-5, 2018

[3] MATSUOKA, S.; AMANO, H.; NAKAJIMA, K.; INOUE, K.; KUDOH, T.; MARUYAMA, N.; TAURA, K.; IWASHITA, T.; KATAGIRI, T.; HANAWA, T. y ENDO, T. «From FLOPS to BYTES: disruptive change in high-performance computing towards the post-moore era,» *CF '16: Proceedings of the ACM International Conference on Computing Frontiers,* p. 274–281, 2016

[4] . MATSUOKA, «Cambrian explosion of computing and big data in the post-moore era,» *HPDC '18: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing,* p. 105, 2018

Estos nuevos dispositivos reciben el nombre de hardware Post-Moore. La utilización de este hardware permite un cambio en la escala, no solo física sino en términos de datos y procesos. El problema que surge es cómo se pueden desplegar aplicaciones en las que sus necesidades sean cubiertas por el hardware Post-Moore, procurando mantener cierta portabilidad, escalabilidad o eficiencia de las aplicaciones.

Como solución a este problema se realiza un estudio del despliegue y ejecución de aplicaciones sobre arquitecturas Post-Moore, el estudio evalúa el impacto en el rendimiento computacional según su despliegue durante la ejecución de aplicaciones sobre arquitecturas Post-Moore. Esto permite una caracterización de la eficiencia computacional según su método de despliegue. Además, se identifican las características especiales de estas arquitecturas Post-Moore y se seleccionan dispositivos que cubran estas peculiaridades. Igualmente se propone lineamientos para establecer estrategias de despliegue que procuren minimizar el impacto en el rendimiento computacional, la escalabilidad, portabilidad o eficiencia.

El trabajo se compone de las siguientes secciones:

- La primera sección se establecen los objetivos y alcance de este estudio.
- La segunda sección muestra el marco teórico.
- La tercera sección presenta el estado del arte necesario para el desarrollo del estudio.
- La cuarta sección exhibe un esquema de la metodología usada en el desarrollo del estudio.
- La quinta sección expone las características de los dispositivos Post-Moore.
- La sexta sección plantea los mecanismos, lineamientos y estrategia de despliegue sobre arquitecturas Post-Moore.
- La séptima sección esboza las pruebas, métricas y flujo de las pruebas para evaluar el despliegue de aplicaciones sobre arquitecturas Post-Moore.
- La octava sección enseña los resultados de las pruebas, siguiendo los

lineamientos y estrategia de despliegue.

Por último, se expone las conclusiones del estudio y posibles trabajos futuros.

# 1. OBJETIVOS

## 1.1 OBJETIVO GENERAL

Evaluar el impacto en el rendimiento computacional según su despliegue durante la ejecución de aplicaciones sobre arquitecturas Post-Moore que permitan una caracterización de la eficiencia computacional.

## 1.2 OBJETIVO ESPECIFICO

- Identificar las características especiales en arquitecturas Post-Moore para establecer estrategias de despliegue de aplicaciones.
- Establecer mecanismos de evaluación del impacto en aplicaciones, que permitan evaluar la escalabilidad, portabilidad, eficiencia entre otros factores asociados.
- Proponer lineamientos para la construcción de mecanismos de despliegue que minimicen el impacto sobre el rendimiento computacional en la ejecución de aplicaciones sobre arquitecturas Post-Moore.

## 1.3 ALCANCE

Este trabajo expone un estudio del despliegue y ejecución de aplicaciones sobre arquitecturas Post-Moore. Se planea comparar diferentes mecanismos de despliegue para medir su impacto sobre el rendimiento de una aplicación. Esto permite establecer lineamientos que procuren reducir la perdida de rendimiento en

la ejecución de aplicaciones. Dichos lineamientos se usarán en una estrategia de despliegue que será evaluada por medio de comparativas. Durante las comparativas se usarán las mismas aplicaciones variando la escala tanto en recursos como en datos para ver como esto afecta al rendimiento de las mismas aplicaciones según su método de despliegue. El trabajo busca establecer al menos un mecanismo, lineamiento o protocolo para el despliegue de una aplicación, el cual reduzca el impacto en el rendimiento computacional al momento de que una aplicación se despliegue sobre una arquitectura Post-Moore.

Con los objetivos planteados se procede a indagar en la teoría necesaria para realizar el estudio.

# 2. MARCO TEÓRICO

Las aplicaciones se enfrentan a desafíos como su adaptación a los avances del hardware, cambios en las librerías que usan o la enorme heterogeneidad de los sistemas computacionales. Con el tiempo, los sistemas han ido aumentando su complejidad, no solo en cuanto a hardware sino también a software. Esto aumenta el software necesario para administrar un sistema y sus aplicaciones, generando configuraciones únicas para cada sistema, ampliando los tiempos de despliegue por el ensayo y error, dificultando el trabajo de los administradores y la experiencia del usuario.

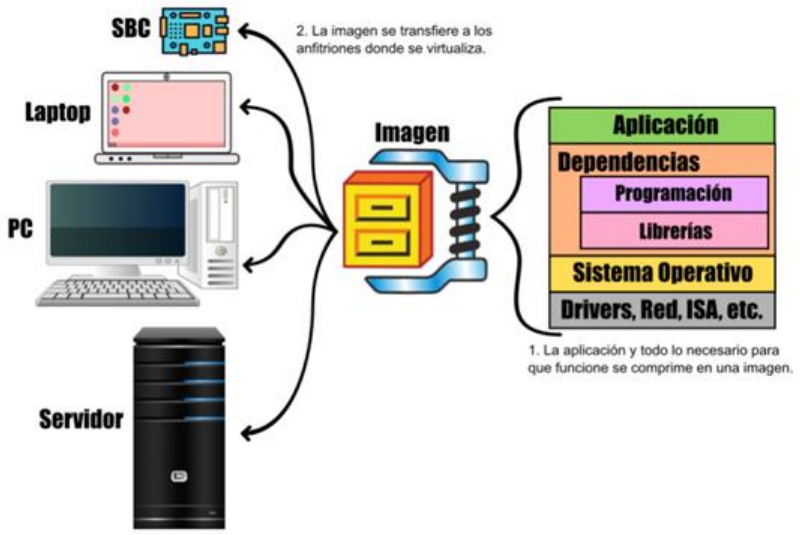
La portabilidad de las aplicaciones se ve afectada por las características y necesidades específicas de hardware y software. La mayoría de las aplicaciones está compuesta por miles de líneas de código, entornos de ejecución y otros softwares que hacen más compleja la instalación, administración y actualización. No existe garantía de que una aplicación pueda ser ejecutable en máquinas nuevas o diferentes si la arquitectura cambia, pero una posible solución es la virtualización.

## 2.1 VIRTUALIZACIÓN

La virtualización nos permite almacenar en archivo o imagen todo el software necesario para el funcionamiento de una aplicación, como se ve en la Figura 1 Esta imagen se puede desplegar virtualmente lo necesario para ejecutar la aplicación en un Anfitrión (Host) de manera portable, rápida y compatible. Herramientas como

VirtualBox,[5] VMware[6] o QEMU[7] han facilitado la accesibilidad, usabilidad y popularización de esta tecnología. Además, con la abundancia de hardware de bajo coste vigente es más factible ver a la virtualización como una plataforma sobre la cual desplegar aplicaciones.

**Figura 1. Ejemplo de virtualización**



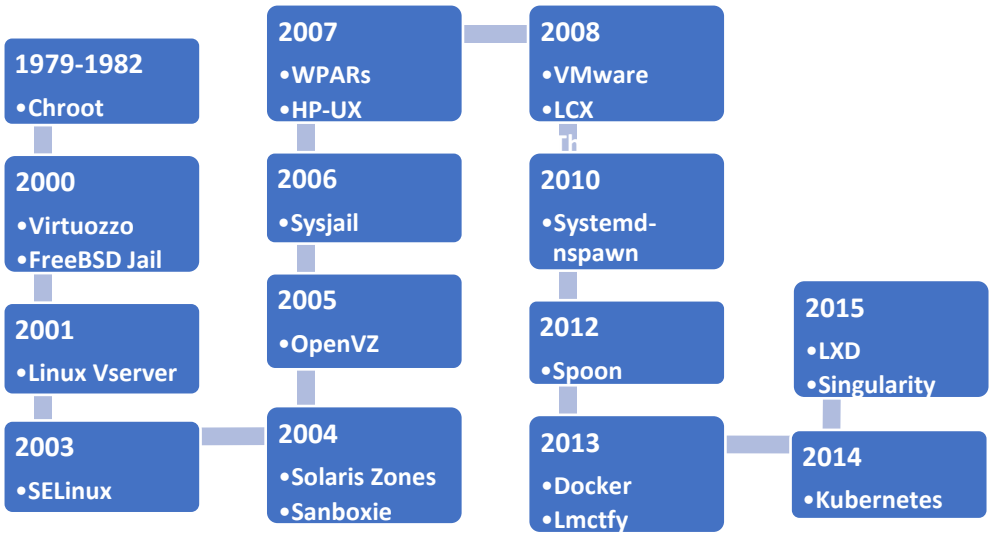

Fuente: https://www.redhat.com/es/topics/virtualization/what-is-virtualization

Una desventaja importante de esta tecnología es la adición de capas software que reducen el rendimiento de la aplicación virtualizada. Existen diferentes tipos de virtualización como virtualización asistida por hardware, virtualización de almacenamiento, particionamiento, hipervisor de almacenamiento o Virtualización de datos. La virtualización ha tenido un largo camino, como se puede ver en la Figura 2. Evolución de la virtualización Y se ha implementado desde antes de que se

[5] ORACLE, «VirtualBox,» Oracle, [En línea]. Available: https://www.virtualbox.org/wiki/Documentation. [Último acceso: 10 Febrero 2021]
[6] VMware, «VMware,» [En línea]. Available: https://www.vmware.com/company.html. [Último acceso: 11 Febrero 2021]
[7] QEMU, «QEMU,» [En línea]. Available: https://www.qemu.org/. [Último acceso: 12 Febrero 2021]

percibieran como tal.

**Figura 2. Evolución de la virtualización**




Los dos tipos de virtualización que más destacan son las máquinas virtuales (VM) y la virtualización a nivel de sistema operativo (contenedores). Las VM simulan un sistema de computación y puede ejecutar programas como una computadora física, una característica clave es que los procesos que se ejecutan en la VM no pueden escaparse de esta. Debido al alto consumo de recursos y la ralentización en la ejecución de aplicaciones. Las VM se descartan como una opción válida para el desarrollo de este trabajo, esto nos deja con la opción de contenedores.

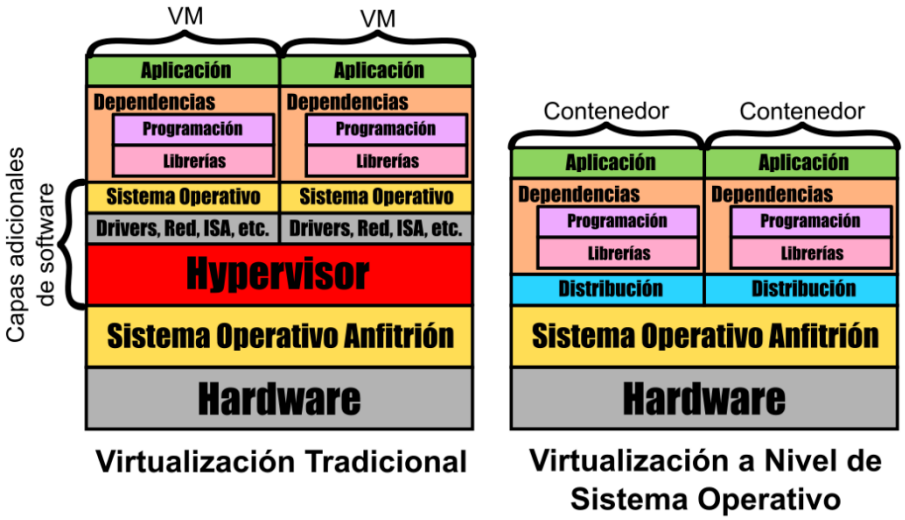
## 2.2 CONTENEDORES

Los contenedores[8] son un método de virtualización a nivel de sistema operativo para implementar y ejecutar aplicaciones sin lanzar una máquina virtual completa. En su

[8] LOPEZ, G. «Teldat Blog,» 3 Octubre 2018. [En línea]. Available: https://www.teldat.com/blog/es/contenedores-contenerizacion-virtualizacion-de-sistema-operativo-sd-wan/. [Último acceso: 11 Septiembre 2020

lugar, varios sistemas aislados se ejecutan en un anfitrión y acceden a un único kernel. Los contenedores albergan los componentes (como archivos, variables de entorno y bibliotecas, como se ve en la Figura 3) necesarios para ejecutar el software deseado. Debido a que los recursos se comparten de esta manera, se pueden generar contenedores que pongan menos presión a los recursos globales disponibles.

**Figura 3. Contenedores Vs Virtualización tradicional.**




Fuente: SFEIR, Christian. Máquinas Virtuales vs Contenedores, ¿Qué son y cómo elegir entre estas tecnologías? 29 de junio 2016 https://www.fayerwayer.com/2016/06/maquinas-virtuales-vs-contenedores-que-son-y-como-elegir-entre-estas-tecnologias/

En el fondo, los contenedores renuncian al hipervisor y otras capas de software utilizando las funcionalidades de aislamiento del kernel. Mediante la gestión del namespace se puede tener una jerarquía de procesos del sistema, acceso a los distintos dispositivos, memoria, etc. Todo esto se logra mediante la gestión de los 6 namespace[9]: identificador de procesos (PID), Red (NET), Comunicación entre

[9] QUIROGA, D. «NAMESPACES - AÍSLAR LOS PROCESOS GNU/LINUX EN SUS PROPIOS ENTORNOS DE SISTEMA,» [En línea]. Available: https://clibre.io/blog/por-secciones/codigo/item/384-

procesos (IPC), Mount (MNT), Usuario (USR), Unix Timesharing System (UTS).

Además del namespace, los contenedores pueden usar los *Cgroups* para administrar los recursos de hardware. *Cgroups* permite definir y monitorizar la cantidad de recursos que puede usar un proceso, ofreciendo las siguientes funcionalidades: Limitación de recursos, prioridad, contabilidad y control.

Los contenedores tienen algunas limitaciones. Solo están implementados a partir de la versión 3.8 del kernel de Linux, esto significa que puede haber problemas en versiones de kernel anteriores y otros sistemas operativos (OS) como MS Windows[10] o MacOS[11]. Además, los contenedores no ofrecen virtualización de hardware, por ello, no pueden virtualizar componentes de hardware de un sistema. Como resultado, están sujetos a restricciones y su uso no puede ser viable en todo tipo de situaciones. En definitiva, resulta en vano montar un contenedor con una aplicación que use GPUs, si el sistema donde se desplegara no las posee. También, no sirve de nada utilizar un contenedor con una aplicación compilada para una arquitectura de CPU si se va a desplegar en una arquitectura de CPU diferente.

Con la teoría presentada se procede a mostrar algo del estado del arte consultado para este trabajo.

namespaces-aislar-los-procesos-de-linux-en-sus-propios-entornos-de-sistema. [Último acceso: 15 Febrero 2021]

[10] Microsoft, «Windows,» [En línea]. Available: https://www.microsoft.com/es-co/windows. [Último acceso: 16 Febrero 2021]

[11] Apple, «MacOS,» [En línea]. Available: https://www.apple.com/co/macos/big-sur/. [Último acceso: 17 Febrero 2021]

# 3. ESTADO DEL ARTE

Para iniciar este trabajo se ha indagado en la literatura actual respecto al uso de contenedores, se debe de tener en cuenta que la portabilidad y el rendimiento son razones de peso para elegir los contenedores. Por ello, se debe dar un vistazo a las tecnologías de virtualización y sus diferentes variantes.

En la primera lectura, Matsuoka et. Al.,[12] presenta varios cambios disruptivos en el HPC de cara a la era Post-Moore. En este documento se enfatiza que, para seguir avanzando tecnológicamente, el hardware y software deben trabajar de manera integrada. Las necesidades de las aplicaciones deben dictar que hardware debe ser usado para su ejecución. Basado en las propuestas de este documento, se da una definición de la era Post-Moore, sus características y que dispositivos pueden ser considerados Post-Moore.

Los puntos más relevantes de la virtualización a nivel de sistema operativo son la portabilidad y flexibilidad. Por ello, Kovàcs[13] realiza una comparación entre diferentes contenedores Linux con ejecuciones nativas, usa herramientas de virtualización como Docker[14], Singularity[15], KVM[16] y LXC[17]. Los benchmark son algo básicos, pero logran demostrar que el rendimiento en los contenedores se iguala al rendimiento en nativo. Para este trabajo, se generan cargas más intensas para probar si el rendimiento puede ser superado.

---

[12] MATSUOKA, AMANO, NAKAJIMA, INOUE, KUDOH, MARUYAMA, TAURA, IWASHITA, KATAGIRI, HANAWA Y ENDO. Op. Cit.

[13] KOVÀCS, À. «Comparison of different Linux containers,» *40th International Conference on Telecommunications and Signal Processing (TSP),* pp. 47-51, 2017

[14] Más información sobre Docker: Véase 0

[15] Mas información sobre Singularity: Véase 0

[16] Red Hat, «KVM,» [En línea]. Available: http://www.linux-kvm.org/page/Guest_Support_Status. [Último acceso: 18 Febrero 2021]

[17] Canonical Ltd., «Linux Containers,» [En línea]. Available: https://linuxcontainers.org/. [Último acceso: 19 Febrero 2021]

Otras evaluaciones a los contenedores se realizan por parte de Benedicic et. al.[18], Ruiz et. al.[19] o Animesh Kuity y Sateesh Kumar Peddoju[20] realizan comparativas de rendimiento en sistemas HPC. También, Younge et. al.[21] y Hale et. al.[22], prueban a Singularity y Shifter[23] en HPC. Basado en estas investigaciones, el trabajo busca evaluar la implementación en Docker, Singularity y Nativo de diferentes cargas de trabajo sobre diferentes arquitecturas.

Uno de los participantes más activos en la experimentación con contenedores es Nvidia con NGC[24], este catálogo centraliza software optimizado para GPU enfocado en aprendizaje profundo (DL), aprendizaje automático (ML) y computación de alto rendimiento (HPC). NGC acelera la implementación de aplicaciones para científicos, desarrolladores o investigadores para que puedan centrarse en la creación de soluciones o productos.

De manera paralela al desarrollo de los contenedores, los Sistemas Operativos desarrollaron herramientas como los manejadores de paquetes, estas herramientas simplifican el proceso de instalación, configuración, eliminación o actualización de las aplicaciones sobre diferentes arquitecturas de manera nativa, algunos referentes son APT, RPM o NIX. Un trabajo referente a estos manejadores fue realizado por

[18] BENEDICIC, L.; CRUZ, F. A.; MADONNA, A. y MARIOTTI, K. «Portable, high-performance containers for HPC,» 2017

[19] RUIZ, C.; JEANVOINE, E. y NUSSBAUM, L. «Performance evaluation of containers for HPC,» *European Conference on Parallel Processing,* pp. 813-824, 2015

[20] KUITY, A. y PEDDOJU, S. K. «Performance Evaluation of Container-Based High Performance Computing Ecosystem Using OpenPOWER,» *International Conference on High Performance Computing,* pp. 290-308, 2017

[21] YOUNGE, A. J.; PEDRETTI, K.; GRANT, R. E. y BRIGHTWELL, R. «A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds,» *Cloud Computing Technology and Science (CloudCom),* pp. 74-81, 2017

[22] HALE, J. S.; LI, L.; RICHARDSON, C. N. y WELLS, G. N.«Containers for Portable, Productive, and Performant Scientific Computing,» *Computing in Science & Engineering,* pp. 40 - 50, 2017

[23] NERSC, «Shifter - Containers for HPC,» [En línea]. Available: https://github.com/NERSC/shifter. [Último acceso: 20 Febrero 2021]

[24] Nvidia, «NGC Catalog,» [En línea]. Available: https://www.nvidia.com/en-us/gpu-cloud/. [Último acceso: 21 Febrero 2021]

Gómez et. al.[25]. Esta herramienta se puede invocar en el archivo de construcción para instalar y configurar la aplicación, simplificado la construcción de la imagen.

En consecuencia, este trabajo no solo compara el rendimiento de cada método de despliegue bajo diferentes cargas de trabajo. También compara la eficiencia energética del despliegue y como ciertos lineamientos pueden mejorar el rendimiento y facilitar la portabilidad de las aplicaciones o su escalabilidad.

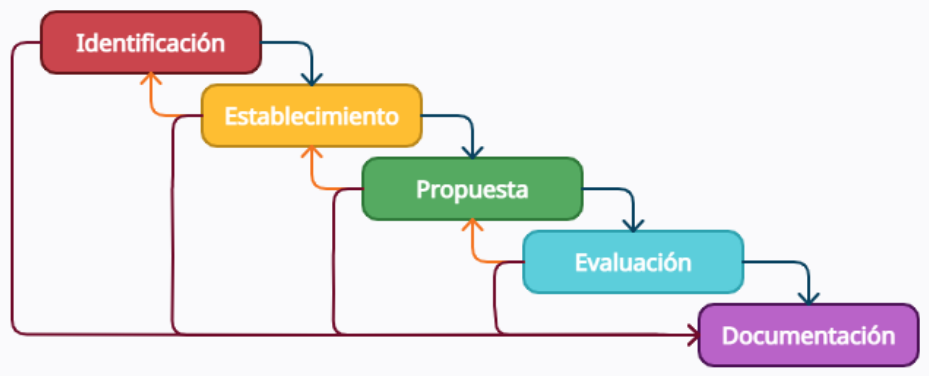
Con el estado del arte mostrado se presenta una metodología para el desarrollo del estudio.

[25] GÓMEZ HERNÁNDEZ, C. E. Análisis de Sistemas Embebidos HPC usando Manejadores de Paquetes, Bucaramanga: Universidad Industrial de Santander, 2020

# 4. **METODOLOGÍA**

Para el desarrollo de este trabajo, la metodología implementada consta de cuatro fases. Cada fase puede realimentar la fase anterior y al mismo tiempo genera su propia documentación. Puede verse como una metodología de cascada, pero al realimentarse, los resultados cubren los objetivos con mayor facilidad debido a las correcciones hechas por la realimentación y se facilita la documentación del trabajo como se muestra en la Figura 4.

**Figura 4. Metodología**




Cada fase en la metodología busca alcanzar cada uno de los objetivos planteados para el desarrollo del estudio, las fases se describen a continuación:

- **Identificación:** En esta fase se identifican las características especiales que poseen las arquitecturas Post-Moore, de esta manera se facilita la selección de dispositivos que se pueden considerar como arquitecturas Post-Moore y se plantea una estrategia para el despliegue de aplicaciones sobre la misma.
- **Establecimiento:** En esta fase se indaga sobre mecanismos de evaluación que se pueden implementar para medir el impacto (rendimiento, eficiencia,

escalabilidad o portabilidad) sobre las aplicaciones escogidas. Se usan los dispositivos seleccionados y la estrategia planteada en la fase de identificación para probar los diferentes mecanismos investigados y seleccionar al menos uno para cumplir con el objetivo general del trabajo.

- **Propuesta:** En esta fase se proponen lineamientos para la construcción de mecanismos de despliegue que reduzcan el impacto en las aplicaciones sobre las arquitecturas Post-Moore. Esta propuesta se evalúa con el mecanismo establecido en la fase anterior usando los dispositivos y estrategia identificados.
- **Evaluación:** En esta última fase se muestra la evaluación del impacto durante la ejecución de las aplicaciones sobre las arquitecturas Post-Moore y sus resultados. Se usan los lineamientos propuestos para el despliegue, el mecanismo de evaluación establecido y la estrategia y dispositivos identificados.
- **Documentación:** Esta fase se construye a partir del desarrollo de la investigación, el documento se revisa periódicamente para actualizar y agregar los avances y correcciones de cada fase de la investigación.

Con la metodología definida se procede a la documentación del desarrollo del estudio, presentada en la siguiente sección.

# 5. CARACTERIZACIÓN DE LOS DISPOSITIVOS POST-MOORE.

Para el desarrollo del estudio se plantea como primer objetivo específico identificar las características especiales de las arquitecturas Post-Moore. Antes de ello se debe definir en qué momento se pasa de Moore a Post-Moore.

## 5.1 EL TERMINO POST-MOORE

Existen diferentes exponentes sobre este término, pero una de las definiciones más completas se presenta en[26]. En este documento, S. Matsuoka et al., plantean el final de la ley de Moore para el periodo 2025-2030. En el documento se propone como Post-Moore a todo intento por superar las limitaciones presentadas por la ley de Moore. Entre las limitaciones presentadas por la ley de Moore se encuentra el límite físico de la reducción de la litografía, dificultades con la disipación de temperatura en los chips o la inestabilidad en el funcionamiento del chip cuando se aumentan las frecuencias de reloj. Los intentos se pueden dividir en dos categorías:

- La primera es la mejora en las tecnologías de los dispositivos que superen al CMOS.
- La segunda es la implementación de nuevos modelos computacionales asociados con dispositivos de computación cuántica y computación neuromórfica.

[26] MATSUOKA, AMANO, NAKAJIMA, INOUE, KUDOH, MARUYAMA, TAURA, IWASHITA, KATAGIRI, HANAWA y ENDO. Op. Cit.

Además, abarcan tecnologías como: Multinúcleo, Computación Heterogénea, 3D-Chips, Computación Cuántica, Computación Neuromórfica, etc. **Para el desarrollo del estudio no se tiene en cuenta tecnologías como los 3D-chips, computación cuántica o neuromórfica debido a la dificultad de acceso.** Las características especiales para este trabajo se enfocan en hardware tradicional o tecnologías de fabricación tradicional por su fácil disponibilidad.

## 5.2 CARACTERÍSTICAS DE LAS ARQUITECTURAS POST-MOORE

La era Post-Moore no se enfoca en chips individuales, su principal característica es la integración de múltiples chips de propósito específico que se adapten a las necesidades de las aplicaciones[27], como en la Figura 5 Para lograr esta integración, se debe tener en cuenta a la arquitectura de sistemas heterogéneos (HSA). Su objetivo es reducir la latencia de comunicación entre CPU, GPU y otros dispositivos haciéndolos más compatibles desde la perspectiva del programador[28]. Lenguajes como CUDA[29] u OpenCL[30] pueden usar HSA para aumentar el rendimiento de ejecución y el cálculo en paralelo[31].

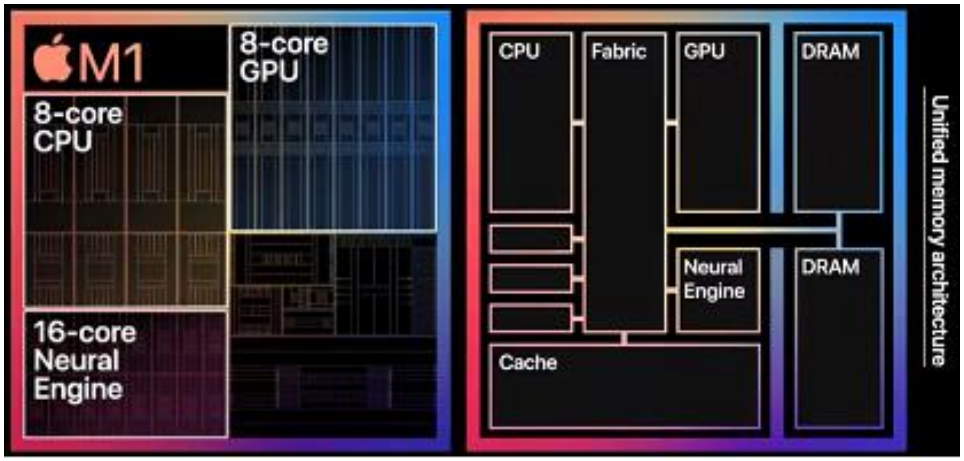
[27] Ibid
[28] KYRIAZIS, G. «Heterogeneous System Architecture: A Technical Review,» AMD, 2012
[29] «CUDA.,» [En línea]. Available: https://developer.nvidia.com/cuda-zone. [Último acceso: 1 Febrero 2020]
[30] «Khronos OpenCL Registry,» Khronos Group, 27 Abril 2020. [En línea]. Available: https://www.khronos.org/registry/OpenCL/. [Último acceso: 10 Julio 2020]
[31] ALTSCHULER, F. y GALLMEIER, J. «Heterogeneous system architecture: Multicore image processing using a mix of CPU and GPU elements.,» *Embedded Computing Design*, 2012

**Figura 5. ARM Cortex-A77, boceto de una arquitectura Post-Moore.**


Fuente: ARM keynote Computex 2019

El avance en las tecnologías de fabricación permite que los componentes se integren en un sistema en chip o System on Chip (SoC). Muchos procesadores incluyen la lógica para interactuar con otros dispositivos (controladores de memoria, USB, PCI, SATA, Ethernet, RFID, etc), así como FPGA, GPU, chips neurales y cerebrales, como se ve en la Figura 6

**Figura 6. Apple M1.**




Fuente: Muy Computer www.muycomputer.com

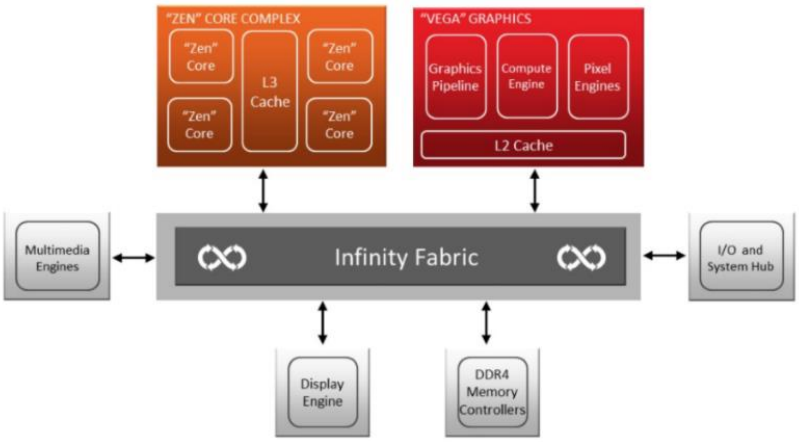
La agrupación de estos chips se dio originalmente a la largo y lo ancho del chip, pero recientemente se ha empezado a hacer hacia lo alto (3D-chips). En resumen, **La característica general de las arquitecturas Post-Moore es la integración de múltiples chips de propósito específico, esta integración se puede dar en un SoC. Por ello, el hardware Post-Moore basado en tecnologías de fabricación tradicional cuentan con dos características especifica: ser multinúcleo y altamente heterogéneo.** Como se habló antes, en esta investigación no se tomarán en cuenta los 3D-Chips, la Computación Cuántica o Neuromórfica.

Basado en las características expuestas antes, se pueden seleccionar algunos dispositivos como candidatos para el estudio. Para simplificar la elección, estos participantes deben tener al menos una CPU multinúcleo y una GPU, convirtiéndolos en arquitecturas heterogéneas. Si es posible, los dispositivos deben permitir el monitoreo de consumo energético.

## 5.3 DISPOSITIVOS POST-MOORE

Los dispositivos Post-Moore han sufrido una rápida expansión y diversificación. En estos dispositivos podemos encontrar a las CPU AMD Ryzen que integran una cantidad de núcleos (64, 32, 24 o 16 núcleos, 128, 64, 48 o 32 hilos) que hace años eran solo una ilusión en equipos de escritorio. Cabe resaltar las versiones G (8, 6, o 4 núcleos, 16, 12, o 8 hilos), que cuentan con GPU integrada como en la Figura 7.

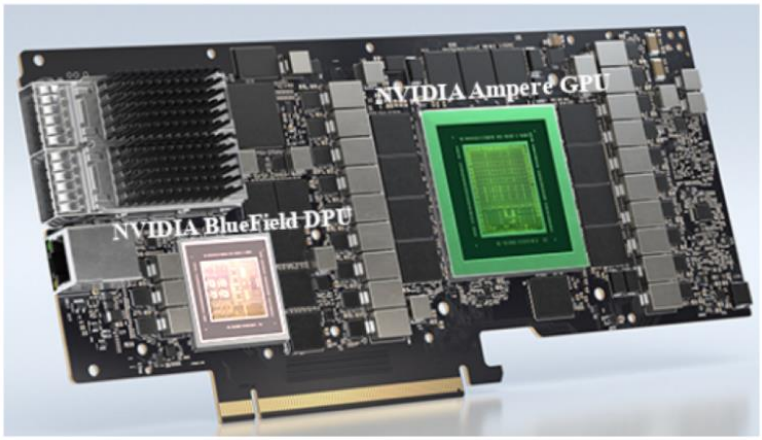
**Figura 7. Bosquejo de la microarquitectura APU de AMD.**




Fuente: AMD.

Otros dispositivos que cuentan con características similares son las GPU de Nvidia que mezcla núcleos CUDA, núcleos Tensores, núcleos RT, unidades de ROP, geometría y textura. Estos chips no solo ofrecen excelentes gráficos, sino que permite el entrenamiento y ejecución de modelos de IA, ML y DL. En los últimos modelos no solo se da manejo a el big data, también, se encargan de la ciberseguridad y la administración con las Unidades de Procesamiento de Datos (DPU), como en la Figura 8.

**Figura 8. Tarjetas NVIDIA BlueField-2X AI-Powered DPU.**




Fuente: NVIDIA.

Con todo lo dicho antes, debe quedar claro que, **los dispositivos Post-Moore son altamente heterogéneos lo que les permite cubrir un amplio rango de necesidades de las aplicaciones**, teniendo esto en cuenta se seleccionan un par de dispositivos.

## 5.4 DISPOSITIVOS SELECCIONADOS

Para el desarrollo de la investigación, los dispositivos seleccionados cuentan con características adicionales a las planteadas anteriormente, estas son: bajo costo económico, popularidad y facilidad de acceso o adquisición. Esto reduce los costos de la investigación, haciéndola asequible para futuros proyectos. Para cumplir con las características adicionales, los dispositivos seleccionados pertenecen a dos ramas diferentes.

En la primera rama se encuentran las Computadoras de Tarjeta Simple o Single Board Computer (SBC), son computadoras en las que CPU, GPU, RAM, etc., se encuentra en un solo circuito o tarjeta. Sus aplicaciones abarcan desde entornos industriales hasta sistemas domésticos de internet de las cosas (IoT). Debido a la alta integración de componentes y tamaño reducido, estos dispositivos cuentan con una mayor confiabilidad, mejor manejo de consumo y menor peso. Como desventaja, si se requiere una actualización o la tarjeta se daña, debe ser reemplazada por otra SBC. A pesar de lo mencionado anteriormente, las SBC se puede producir en masa para reducir los costos y, según su OS, su propósito puede ser desde general a muy específico. La segunda rama son las Computadoras Personales (PC), la PC ha sido heterogénea desde hace bastante tiempo. Además, el costo de los cálculos por segundo ha ido disminuyendo, mejorando el acceso al nuevo hardware con mayor potencia.

Después de una búsqueda, hay varios candidatos, entre los que se destacan: Raspberry Pi (RPi)[32], Orange pi[33], asus tinkerboard[34], Odroid[35] y Nvidia Jetson Nano[36]. Para este proyecto, se elige a la Jetson Nano de primera generación, esto se debe a que cumple con cada una de las características propuestas anteriormente. Cuenta con una CPU, una GPU y una RAM superior a la RPi, orange pi, tinkerboard o odroid. El costo económico para los recursos que ofrece, popularidad y fácil acceso también supera a las otras SBC. El otro dispositivo seleccionado es algo más tradicional, una PC armada por partes, enfocada a propósito general y gaming. Los dispositivos se describen en la Tabla 1.

[32] Raspberry Foundation, «Raspberry pi,» [En línea]. Available: https://www.raspberrypi.org/products/. [Último acceso: 22 Febrero 2021]
[33] Xunlong Software CO, «Orange pi,» [En línea]. Available: http://www.orangepi.org/Docs/mainpage.html. [Último acceso: 23 Febrero 2021]
[34] Asus, «Asus Tinker Board,» [En línea]. Available: https://tinker-board.asus.com/product/tinker-board.html. [Último acceso: 23 Febrero 2021]
[35] Hardkernel, «Odroid,» [En línea]. Available: https://wiki.odroid.com/. [Último acceso: 23 Febrero 2021]
[36] Nvidia Developer, «Jetson Nano,» [En línea]. Available: https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/. [Último acceso: 24 Febrero 2021]

**Tabla 1. Características de los Dispositivos Elegidos.**

| ***Dispositivo*** | ***Jetson Nano*** | ***PC*** |
|---|---|---|
| *CPU* | *ARM Cortex-A57 Quad-Core de 64 bits a 1,43 GHz* | *Ryzen 5 3600 Hexa-Core con 64 bits a 3.6~4,2GHz* |
| *GPU* | *Nvidia Maxwell de 128 núcleos a 921 MHz* | *Nvidia Pascal GTX 1050Ti a 1350Mhz* |
| *RAM* | *4 GB de RAM LPDDR4 a 1600 MHz* | *16GB RAM DDR4 a 3200MHz* |
| *Almacenamiento* | *32 GB SD card* | *240 GB SSD* |

En conclusión, **Post-Moore es todo intento por superar las limitaciones presentadas por la ley de Moore, las arquitecturas Post-Moore cuenta con las características de ser multinúcleo y heterogéneas, esto permite cubrir un amplio rango de necesidades de las aplicaciones y aprovechar el paralelismo**, para medir estas capacidades, se generan estrategias y mecanismos de evaluación, esto se trata en las próximas secciones.

# 6. ESTRATEGIA DE DESPLIEGUE SOBRE ARQUITECTURAS POST-MOORE

La segunda parte del primer objetivo específico es establecer una estrategia de despliegue. La estrategia propuesta consta de dos etapas:

- Etapa 1: se presentan los mecanismos de despliegue que se indagaron en el marco teórico.
- Etapa 2: se exponen los lineamientos para la construcción de los mecanismos mencionados en la primera etapa.

De esta manera se aborda el tercer objetivo específico. Finalizada estas dos etapas, se formula una estrategia para el despliegue de aplicaciones sobre arquitecturas Post-Moore y se termina de abordar el primer objetivo específico.

## 6.1 MECANISMOS DE DESPLIEGUE

Los mecanismos de despliegue que se indagaron para este estudio son: Despliegue en nativo y Despliegue en contenedores. La opción de máquinas virtuales se descarta por la cantidad de capas de virtualización adicionales en comparación a los contenedores, esta diferencia se muestra en la Figura 3 y se explica en el Marco Teórico.

**6.1.1 Despliegue en Nativo**. Consiste en instalar una herramienta o aplicación con todas sus dependencias sobre una máquina y su sistema operativo (OS). Es un despliegue sin capas de virtualización. Es en teoría, la manera que menos genera impacto sobre el rendimiento de una aplicación. Para este trabajo se toma el despliegue en nativo como la base para medir la diferencia respecto al otro método

de despliegue.

**6.1.2 Despliegue en Contenedores.** Consiste en utilizar una herramienta que ofrece una capa de virtualización sobre el OS de la máquina para el despliegue de una herramienta o aplicación con todas sus dependencias. El contenedor aísla la aplicación del resto del sistema, en teoría, mejora su portabilidad, fluidez y flexibilidad. Existen varios exponentes de esta tecnología, aun así, para este trabajo se considera que los dos más representativos son Docker y Singularity.

**6.1.2.1 Docker:** Es una herramienta que empaqueta una aplicación y sus dependencias en un contenedor. Se puede ejecutar en multiplataforma, esto genera flexibilidad y portabilidad para la ejecución de la aplicación, ya sea en las instalaciones físicas, nube pública o privada. No requiere incluir un OS independiente, utiliza el aislamiento de recursos, accede a la virtualización del kernel por medio de bibliotecas como libcontainer, libvirt, LXC o systemd-nspawn. Los recursos pueden ser aislados, los servicios restringidos y múltiples contenedores comparten el mismo kernel. Cada contenedor puede restringirse a utilizar solo una cantidad definida de recursos (CPU, RAM, I/O, etc) simplificado la creación de sistemas altamente distribuidos. Esto permite que múltiples aplicaciones, tareas y otros procesos funcionen en la misma maquina física o múltiples máquinas virtuales. De esta manera se habilita que el despliegue de nodos se realice a medida que se dispone de recursos o cuando se necesitan más nodos, también simplifica la creación y funcionamiento de cargas de trabajo y colas.

**6.1.2.2 Singularity:** Es un programa libre, multiplataforma y de código abierto que realiza contenerización. Uno de sus usos principales es llevar la contenerización y reproducibilidad a la computación científica y el HPC. Los desarrolladores pueden trabajar en entornos de su elección y diseño, que al ser reproducibles se pueden copiar y ejecutar con facilidad en otras plataformas. Esto aumenta su utilidad en áreas como Big data, ML y DL. También se integra a la perfección con muchos

administradores de recursos (SLURM, torque, SGE, etc).

Las características principales de Docker y Singularity se presentan en la Tabla 2.

**Tabla 2. Características principales de Docker y Singularity.**

| ***Características*** | ***Docker*** | ***Singularity*** |
|---|---|---|
| *Instalación Ligera* | ✓ | ✓ |
| *Diseñado para HPC* | ✓ | ✓ |
| *Compatible con gestores de Colas* | × | ✓ |
| *Uso de Cgroups* | ✓ | × |
| *Namespace* | PID, NET, IPC, MNT, UTC | PID, MNT, USR |
| *Seguridad* | Root deamon | SUID/UserNS |
| *Soporte MPI* | ✓ | ✓ |
| *Soporte GPU* | ✓ | ✓ |

## 6.2 LINEAMIENTOS PARA CONSTRUIR MECANISMOS DE DESPLIEGUE

Durante la fase de evaluación se prueba la efectividad de los lineamientos, gracias a esta realimentación, los lineamientos que mejor resultado mostraron son presentados. Antes de desplegar cualquier aplicación se debe realizar una caracterización de esta, así se obtiene un conocimiento detallado de su funcionamiento y necesidades. Con el funcionamiento y necesidades identificadas se procede a formular los lineamientos.

**6.2.1 Despliegue en Nativo.** Este despliegue se realiza directamente sobre la máquina y OS. Los lineamientos para construir un despliegue nativo son:

1. Confirmar si la aplicación se encuentra disponible para el manejador de paquetes provisto por el OS y si la versión coincide.
   a. Si la aplicación está disponible se procede a la instalación según el manejador de paquetes.
   b. Si no, se deben seguir los pasos del 2 en adelante.
2. Confirmar que la aplicación esta o puede ser compilada para la arquitectura de la maquina en la que se va a desplegar.
3. Confirmar que la maquina cuenta con el hardware necesario para el funcionamiento de la aplicación.
4. Verificar las dependencias de la aplicación para su correcto funcionamiento, en caso de algún faltante, proceder a instalarlas.
5. Verificar la receta de instalación provista por los desarrolladores antes de instalar la aplicación.
6. Implementar la receta de instalación de la aplicación, estas recetas son genéricas y casi siempre falta información para la instalación. Es muy beneficio generar una receta de instalación donde se cubran estas áreas faltantes y se haga pública.
7. Comprobar el funcionamiento de la aplicación por medio de la ejecución con parámetros predeterminados o con benchmark internos de la aplicación.

**6.2.2 Despliegue en Contenedores.** Este despliegue es similar al despliegue nativo, se deben tener en cuenta algunos pasos adicionales. Los lineamientos son los siguientes:

1. Comprobar que el contenedor tenga los permisos suficientes para la ejecución de la aplicación y uso del hardware de la máquina.
2. Comprobar si existe una imagen de la aplicación para la arquitectura en la que se va a desplegar.
   a. Si existe puede proceder con el despliegue.
   b. Si no existe o se requiere una configuración específica siga al paso 3.
3. Realizar el paso 1, 2, 3, 4 y 5 del despliegue nativo.

4. Generar un directorio en el que se albergue el archivo de construcción y su contexto (librerías, ejecutables o dependencias necesarias para la construcción).
5. Verificar que la aplicación tiene solo un proceso padre para que al momento de detener el contendor todos los procesos finalicen. Utilice hipervisores o scripts para generar un solo proceso del cual dependa la aplicación, no es recomendable que un contenedor ejecute más de una aplicación o proceso.
6. Escribir un archivo de construcción de imagen (dockerfile, def, etc.), la información recopilada en el paso 2 facilita la configuración del archivo[37], el archivo debe especificar la instalación de la aplicación.
   a. Si la aplicación está disponible en un manejador de paquetes, se procede configurar la instalación según el manejador de paquetes en el archivo de construcción.
   b. Caso contrario, se realiza el paso 6 del despliegue nativo, configurándolo en el archivo de construcción.
7. Simplifique las capas de construcción del archivo, cada instrucción dentro de un archivo crea una capa, agrupar instrucciones permite reutilizar o mantener el mismo contexto en una capa, reduciendo la cantidad de capas. Si se necesitan herramientas como unzip o wget y no son necesarias para la ejecución de la aplicación, es mejor removerlas de la imagen una vez esté construida.
8. Comprobar que las herramientas adicionales dentro del contenedor no generen problemas de seguridad.
9. Evitar la ejecución de comandos como root, cree usuarios para realizar la construcción de imágenes e inicie el contenedor en modo de solo lectura.
10. Ejecutar la construcción de la imagen y comprobar su funcionamiento de manera similar al paso 7 del despliegue en nativo.

[37] RODRIGUEZ, R. «Recetas Docker,» 2017. [En línea]. Available: https://recetas-docker.readthedocs.io/es/latest/index.html. [Último acceso: 25 Febrero 2021]

## 6.3 PLANTEAMIENTO DE LA ESTRATEGIA DE DESPLIEGUE

Con los lineamientos planteados se formula una estrategia para el desarrollo del estudio. La estrategia usada en los diferentes despliegues se explica a continuación:

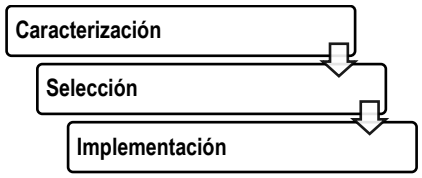
1. **Caracterización de la aplicación seleccionada,** se debe realizar una caracterización de las necesidades de hardware y software, los contenedores no virtualizan hardware. Si la aplicación necesita de GPU, FPGA, etc., no podrán ser desplegadas. Lo mismos sucede si esta compilada para una arquitectura específica, una aplicación compilada para x86 no se pueden desplegar en otras arquitecturas.

2. **Selección del método de despliegue**, solo se deben seguir los lineamientos planteados para cada mecanismo de despliegue

3. **Implementación del método de despliegue**, en el despliegue nativo, cuando se finalizar la instalación de la aplicación se puede ejecutar casi de inmediato la aplicación, al finalizar la ejecución los recursos son liberados. En el caso de los contenedores, se inicia el contenedor, se ejecuta la aplicación y se finaliza el contenedor para que los recursos sean liberados.

De esta manera se establece una estrategia de despliegue como se ve en la Figura 9.

**Figura 9. Planteamiento de la Estrategia.**




Con la estrategia y los lineamientos definidos se pasa a la elección de los mecanismos de evaluación que permitan cumplir con el segundo objetivo específico y alcanzar el objetivo general del estudio.

# 7. MECANISMOS DE EVALUACIÓN

Los lineamientos, estrategia y dispositivos identificados se evalúan por medio de métricas y pruebas. El objetivo de estas pruebas es evidenciar el impacto que sufren las aplicaciones cuando se realizan diferentes despliegues sobre las arquitecturas Post-Moore. Estas pruebas se ejecutan siguiendo un flujo que será descrito en esta sección.

## 7.1 PRUEBAS

Al principio del estudio se plantearon una serie de pruebas para medir el impacto en las aplicaciones, a medida que se desarrolló el estudio no fue posible realizarlas todas por problemas con los contenedores en el dispositivo embebido. Las pruebas que si se implementaron son aplicaciones sencillas que estresan la CPU, realizan cargas de trabajo como el manejo de números flotantes, enteros, primos u operaciones matriciales, las cargas de trabajo son las siguientes:

**7.1.1 Cfloat.** Es una prueba en la que la carga de trabajo se enfoca en la solución de operaciones con números de tipo flotante. Estas operaciones se realizan seleccionando de manera aleatoria las operaciones y los números. Cada vez que la prueba inicia, se genera un nuevo paquete de números aleatorios, se resuelven 1000 operaciones y la prueba finaliza.

**7.1.2 Correlate.** Es una prueba en la que la carga de trabajo correlaciona 16384×1024 números aleatorios de tipo double. Cada vez que la prueba inicia, se genera un nuevo grupo de números.

**7.1.3 Unión**. Es una prueba en la que la carga de trabajo realiza aritmética de enteros en una combinación de campos de bits en una unión C. Esta prueba muestra que tan bien se puede realizar la carga y almacenamiento de campos de bit enteros.

**7.1.4 Hyperbolic.** Es una prueba en la que la carga de trabajo consiste en calcular la solución de la ecuación $\sinh(\theta)\times\cosh(\theta)+\sinh(2\theta)+\cosh(3\theta)$ usando primero números de tipo flotante, seguido de números de tipo doble y por ultimo números de tipo doble largo. Además, $\theta=0$ a $2\pi$ en 1500 pasos para cada iteración de números.

**7.1.5 Prime**. Es una prueba en la que la carga de trabajo consiste en encontrar los números primos de 1 a 1,000,000 usando un algoritmo de fuerza bruta. Cuando termina, lleva un conteo de los primos encontrados, el contador se resta con una variable que tiene almacenada la cantidad de primos de 1 a 1,000,000, si la diferencia es igual a cero la prueba se considera valida.

**7.4.6 Matrixprod.** Es una prueba en la que la carga de trabajo calcula el producto matricial de dos matrices de 128×128 que usan números de tipo doble. Es una prueba similar a HPL pero no utiliza librerías u otras características de HPL, hace la multiplicación en el mismo código de la aplicación.

## 7.5 MÉTRICAS

Las métricas dependen de un único factor, la aplicación. Cada aplicación presenta al menos un resultado cuando finaliza su ejecución. En la mayoría de los casos se toma su tiempo de ejecución como medida para determinar su rendimiento.

En este estudio el tiempo de ejecución de las aplicaciones es mínimo, así que se establece un tiempo general durante el cual las aplicaciones se ejecutan de manera repetitiva. Los resultados presentados usan la cantidad de operaciones que se realizan y la energía consumida durante el tiempo general. Las pruebas se realizan múltiples veces para poder promediar sus resultados.

Para este estudio se establece como métrica general la operación o Ops, **una Ops es la finalización correcta de la ejecución de la aplicación**. La prueba se repite durante el tiempo general T, de esta manera se obtiene la cantidad de Ops/s.

$$\frac{O}{T} = Ops/s$$

Para este estudio la cantidad de Ops/s se mantiene constante cuando la cantidad de tiempo general supera los 60 s. Al mismo tiempo que la prueba se realiza, se capturan los datos de consumo. La medición de la energía consumida será tomada con un medidor externo el cual cuenta con las siguientes características:

**Tabla 3. Especificaciones del medidor de consumo de energía.**

| | |
|---|---|
| **Nombre** | Enchufe Inteligente VTA |
| **Referencia** | VTA-84630 |
| **Wi-Fi** | 2.4 GHz |
| **Voltaje de Operación** | 125 VAC 60Hz |
| **Corriente Máxima** | 10 A |
| **Potencia Máxima** | 1250 W |

El enchufe inteligente[38] cuenta con una aplicación móvil, que lleva un registro mensual del consumo, pero se puede tomar nota del consumo segundo a segundo cuando se realiza la prueba.

Para determinar la variación en el consumo electico se debe dar seguimiento al consumo cuando el dispositivo está en reposo. La diferencia entre el consumo eléctrico durante la prueba y el consumo en reposo nos muestra el consumo de la prueba.

$$C_t - C_r = C_p \Rightarrow \frac{C_p}{T} = W/s$$

Donde, $C_t$ es el consumo promedio total del dispositivo durante la prueba, $C_r$ es el consumo promedio total en reposo del dispositivo y $C_p$ es el consumo durante la prueba. W es solo la unidad de medida del consumo. Cuando se han obtenido las Ops y W, como el tiempo es el mismo para ambas métricas se puede obtener Ops/W.

$$\frac{Ops/s}{W/s} = Ops/W$$

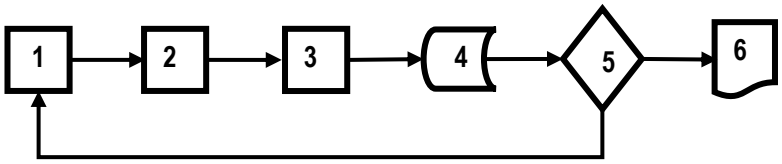
Una vez determinadas las Ops/W se puede comparar los resultados para cada una de las configuraciones.

a. **Flujo de las Pruebas**

Para explicar mejor el desarrollo de las pruebas, se presenta un diagrama general del flujo de trabajo de la prueba, este diagrama presenta seis pasos y se describe a continuación:

[38] VTA, «Tomacorriente Inteligente VTA-84630,» [En línea]. Available: https://www.vta.co/wp-content/uploads/2021/08/VTA-84630_manual_web.pdf. [Último acceso: 24 Febrero 2021]

**Figura 10. Diagrama General del Flujo de Trabajo de las Pruebas.**




i. **Configurar los diferentes requisitos de prueba** Cada prueba es diferente, se pueden modificar valores como el tiempo de duración o el porcentaje de carga de la prueba. Además, se pueden modificar los archivos de configuración del OS para incrementar los MHz o los modelos de consumo de los dispositivos. Por este motivo es necesario configurar la prueba.

ii. **Configurar el monitor de consumo de energía** El monitor de consumo está siempre activo y mostrando datos, por lo tanto, luego de configurar la prueba debemos buscar la forma de capturar los datos de consumo. Si es posible debemos buscar monitores que mantengan un registro de sus métricas, si no es posible, podemos capturar en video o manualmente los datos del monitor durante la prueba.

iii. **Iniciar el monitoreo de consumo de energía y lanzar la prueba.** Una vez configurada la prueba y lista la captura de los datos de consumo, procedemos a ejecutar la prueba, debemos estar muy pendientes del inicio y final de la prueba ya que en este momento es cuando la probabilidad de falla es mayor.

iv. **Almacenar y etiquetar los resultados de las pruebas.** Este paso es tan importante como el primer paso. Esto se debe al etiquetado, casi siempre cuando se hace mal, genera confusión al comparar los resultados y se debe repetir la prueba. Si es posible, la etiqueta debe llevar el nombre del dispositivo, la prueba realizada, los recursos utilizados, su duración o el modelo de consumo.

v. **Repetición de la prueba o inicio de nueva prueba.** Este paso es una bifurcación, en ciertos casos es necesario tener mayor certeza en los resultados. Por ello, es bueno repetir la prueba varias veces y variar su configuración, esto

nos permite ver patrones de comportamiento en los dispositivos. La prueba también puede cambiarse, estas modificaciones pueden mostrar problemas como cuellos de botella, saturación de memoria, bajo rendimiento al cargar o almacenar tipos de datos, etc.

vi. **Agrupar los resultados y generar los gráficos de las pruebas.** Basado en las etiquetas, los datos se procesan para generar los gráficos.

Las pruebas, métricas y flujo tienen como fin plantear un método general de testeo. Las pruebas pueden ser cambiadas por otras aplicaciones y mantener las métricas propuestas. Una métrica muy utilizada son los FLOPS, pero con este método se pueden manejar operaciones con enteros, FPS, primos, asignación de memoria o cualquier otra operación que necesite realizar otro estudio. Los resultados obtenidos en cada una de las pruebas realizadas para diferentes configuraciones se presentan en la siguiente sección.

# 8. RESULTADOS

Las aplicaciones deben estar compiladas para el hardware donde serán desplegadas, esto reduce la portabilidad de las aplicaciones. En muchos trabajos se realiza una virtualización del hardware para desplegar las aplicaciones, los contenedores no pueden emular el hardware para el que fue compilada la aplicación, por ello se tomó como solución, aplicaciones que tenga una compilación para múltiples arquitecturas. Esta solución permite un despliegue en nativo o en contenedores, facilitando el desarrollo del estudio.

Se usa el mismo archivo de configuración para Docker y Singularity, durante el desarrollo se presentaron problemas de ejecución con imágenes que fueron descargadas. Este archivo se crea usando la estrategia y lineamientos de desplegué.

En el apartado anterior se explicaron las pruebas realizadas para el análisis de los dos dispositivos seleccionados. Se debe aclarar que en las etiquetas de las figuras, “N”, “D”, “S” hace referencia al método de despliegue de las pruebas (Nativo, Docker o Singularity) y “Nano” o “PC” hace referencia al dispositivo.

En esta sección se presentan cada una de pruebas realizadas para medir los diferentes impactos, cada prueba se realiza un total de cinco veces usando el flujo de pruebas presentado en 7.3. Los resultados presentados a continuación son los promedios en cada prueba y se han dividido en cinco asuntos:

- ✓ La comparación de los métodos de despliegue.
- ✓ El impacto en el rendimiento según su despliegue.
- ✓ El impacto en los tiempos de ejecución.
- ✓ El impacto en el consumo energético.

✓ El impacto en la escalabilidad.

Mientras se desarrollaba el trabajo se realizaron pruebas en CPU-GPU de manera nativa y se presentaron[39], pero al momento de ser ejecutada las pruebas con los contenedores, solo fue posible ejecutar las pruebas de CPU, esto se debe a que los contenedores no se encuentran completamente funcionales en las arquitecturas ARM y tienen altas probabilidades de falla cuando usan GPU, con el tiempo se corregirán estos errores pero estarían fuera del alcance de este trabajo.

Para las pruebas en CPU la limitante es la cantidad de Cores que tiene el dispositivo, como resultado, solo se usan cuatro Cores por la Jetson Nano. La prueba se repite cinco veces para un total de veinte repeticiones por prueba y un total de ciento veinte repeticiones por dispositivo. El promedio del coeficiente de variación estándar para las pruebas en la Nano fue de 0,0026 en Nativo, 0,0070 en Docker y 0,0074 en Singularity. En el caso de las pruebas en la PC se registra un promedio de coeficiente de variación estándar en Nativo de 0,0024, en Docker de 0,0038, y en Singularity de 0,0027.

## 8.1 COMPARATIVA DE LOS MÉTODOS DE DESPLIEGUE

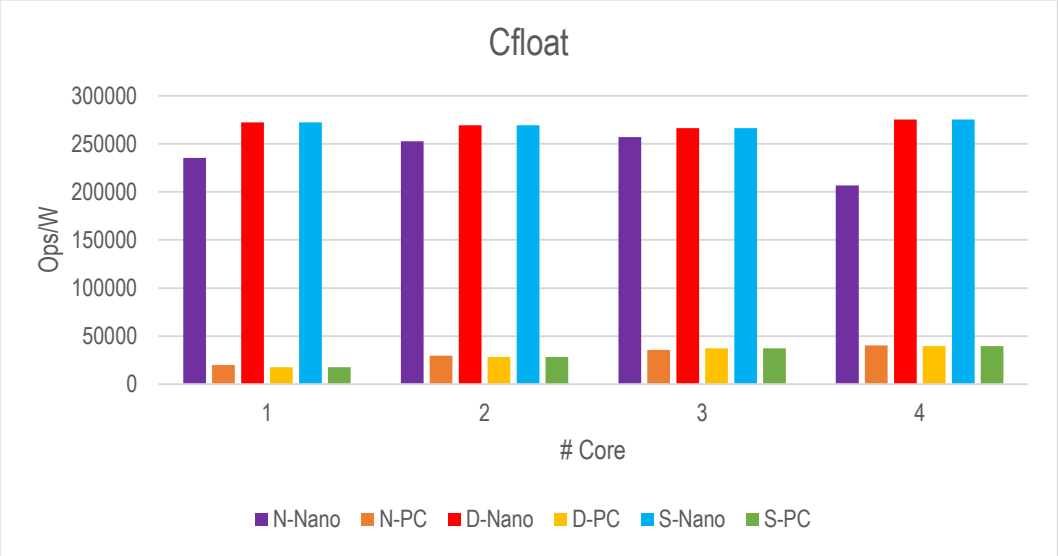
Para no ser repetitivos, todas las pruebas cuando inician, lo primero que hacen es generar el paquete de datos con el que van a trabajar, lo segundo es resolver la prueba seleccionada, esto se contabiliza como una Ops y por último la prueba se repite durante el tiempo que se le asigne a la prueba. Al finalizar la prueba se presenta el acumulado de Ops realizadas, estas pruebas pueden realizarse durante segundos, minutos, horas, días o años, así que no se miden picos de rendimiento

[39] YEPES, P. J. R.; HERNANDEZ, C. J. B. y STEFFENEL, L. A. «Low Energy Consumption on Post-Moore Platforms for HPC Research,» *ACI Avances en Ciencias e Ingenierías,* 2021

sino el desempeño del hardware sometido a estrés.

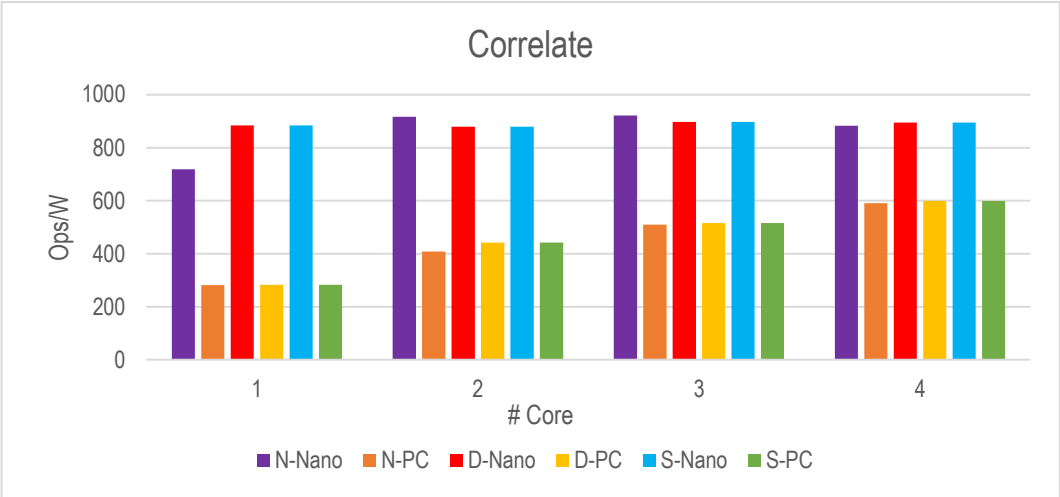
La primera comparativa que se realiza es con la prueba de Cfloat. La prueba de Cfloat realiza una carga de trabajo que se enfoca en la solución de operaciones con números de tipo flotante, estas operaciones se realizan seleccionando de manera aleatoria las operaciones y los números.

**Figura 11. Eficiencia Energética de los Dispositivos en la Prueba de Cfloat.**




En la Figura 11, se aprecia la eficiencia de cada dispositivo durante la prueba de Cfloat. La grafica muestra una mejor eficiencia por parte de la Nano. El que la cantidad de operaciones no varié en la Nano se debe a que el consumo de cada Core es homogéneo. La segunda comparación se realiza con la prueba de Correlate. Es una prueba en la que la carga de trabajo correlaciona números aleatorios de tipo double.

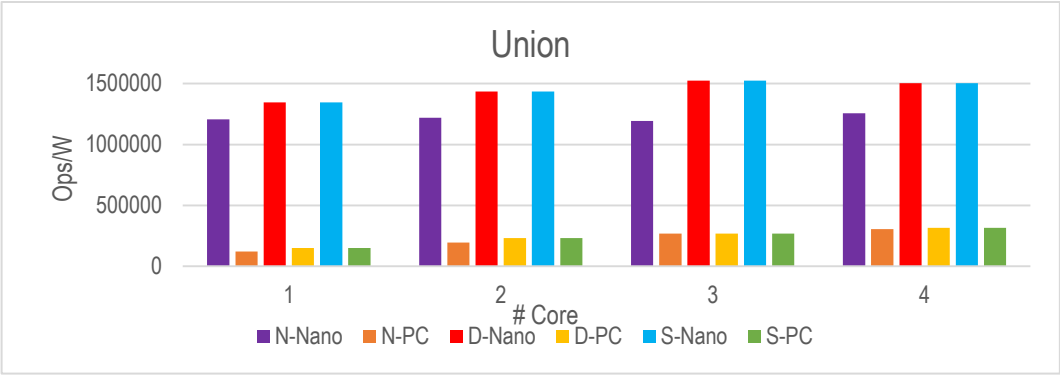
**Figura 12. Eficiencia Energética de los Dispositivos en la Prueba de Correlate.**




La Figura 12 presenta un comportamiento similar al mostrado en la Figura 11, pero la eficiencia de la PC en esta prueba aumenta. El aumento escalonado de la PC se debe a que a medida que se aumenta la cantidad de Cores usados, el consumo por Core de la CPU disminuye. Otro resultado que sale a la luz es que la escalabilidad de la prueba tiene un crecimiento lineal. Solo cuando se usan todos los núcleos de la Nano tenemos una perdida, dicha disminución se da por la competencia de recursos entre el OS y la aplicación. Se debe recordar que las capacidades computacionales de la Nano son de menos de la mitad que la PC.

La siguiente prueba es Unión, esta es una prueba en la que la carga de trabajo realiza aritmética de enteros en una combinación de campos de bits en una unión C. Esta prueba muestra que tan bien se puede realizar la carga y almacenamiento de campos de bit enteros.

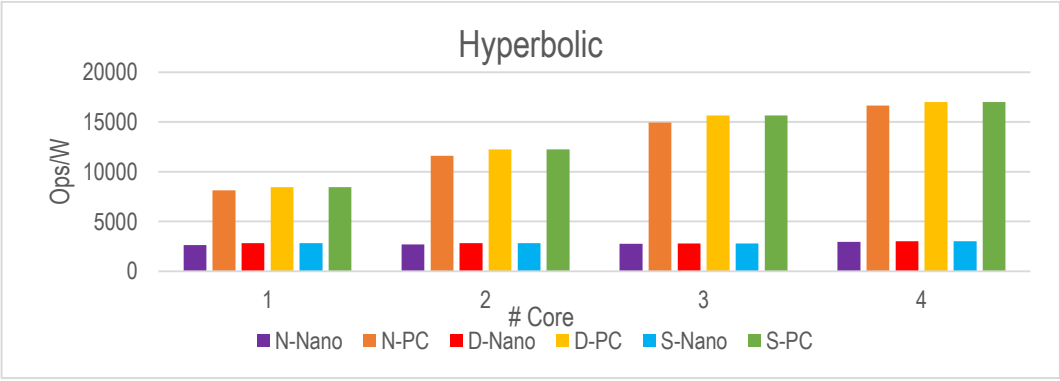
**Figura 13. Eficiencia Energética de los Dispositivos en la Prueba de Union.**




El comportamiento de los dispositivos en la Figura 13 es similar a los mostrados en la Figura 11, resaltando la eficiencia de los dispositivos embebidos. A pesar de que el tipo de dato que se maneja es diferente, el comportamiento se mantiene. También se puede afirmar que los procesadores ARM son buenos en el manejo de enteros, manteniendo un consumo homogéneo por núcleo.

La siguiente prueba es Hyperbolic, en esta prueba la carga de trabajo consiste en calcular la solución de una ecuación usando números de tipo flotante, seguido de números de tipo doble y por último números de tipo doble largo.

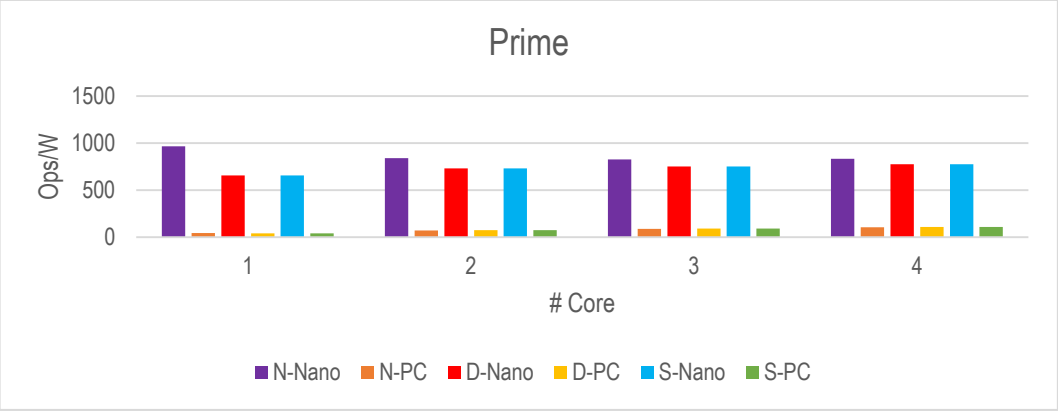
**Figura 14. Eficiencia Energética de los Dispositivos en la Prueba de Hyperbolic.**



El resultado mostrado en la Figura 14 se diferencia de la Figura 11 en que es la solución de una sola ecuación con operaciones complejas y no la solución de operaciones aleatorias. Esto implica que la CPU ARM soluciona operaciones sencillas (seno hiperbólico, coseno, etc) con datos flotantes más fácilmente, que la concatenación (suma de coseno hiperbólico, seno, etc) de dichas operaciones. También, es probable que la arquitectura x86 posea un mejor manejo de datos de tipo doble y doble largo.

La siguiente prueba es Prime, en esta prueba, la carga de trabajo consiste en encontrar números primos usando un algoritmo de fuerza bruta. Cuando termina, lleva un conteo de los primos encontrados. El contador se resta con una variable que tiene almacenada la cantidad de primos de 1 a 1,000,000, si la diferencia es igual a cero la prueba se considera valida.

**Figura 15. Eficiencia Energética de los Dispositivos en la Prueba de Prime.**

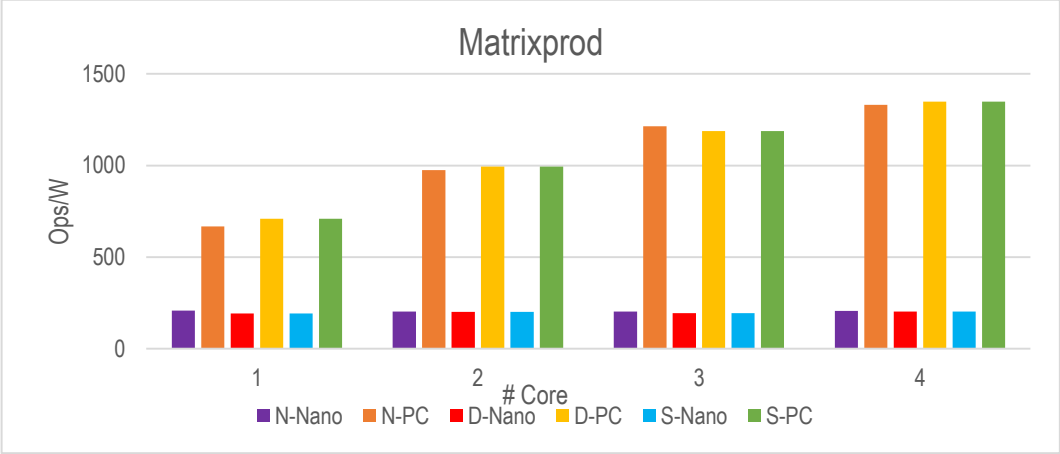



La prueba Prime en la Figura 16 tiene un comportamiento similar a la Figura 11. Esto demuestra que tan eficientes son los dispositivos embebidos si se usan en aplicaciones que requieran trabajar con números primos, flotantes, enteros u operaciones aleatorias. Además, pueden ser usados en aplicaciones en las que el

consumo energético sea una prioridad.

La última prueba es Matrixprod, la prueba genera una carga de trabajo que calcula el producto matricial de dos matrices que usan números de tipo doble. Es una prueba similar a HPL, pero realiza la multiplicación en el mismo código de la aplicación.

**Figura 16. Eficiencia Energética de los Dispositivos en la Prueba de Matrixprod.**




En la Figura 16 se muestra un comportamiento similar a la Figura 14, se ve una eficiencia abrumadora por parte del PC, duplicando o triplicando el resultado. De la misma manera se ve un aumento escalonado de la eficiencia a medida que se aumenta los recursos para la prueba, esto se debe a que entre más Cores use el PC menor es el consumo de cada Core.

Por otro lado, influye que el procesador ARM funciona a menos de la mitad de la frecuencia del x86, posea una litografía de mayor tamaño (16 nm vs 7 nm), tiene una menor cantidad y área de transistores. Sería interesante comparar un

procesador ARM con frecuencias, litografía, área y cantidad de transistores similar a el procesador x86.

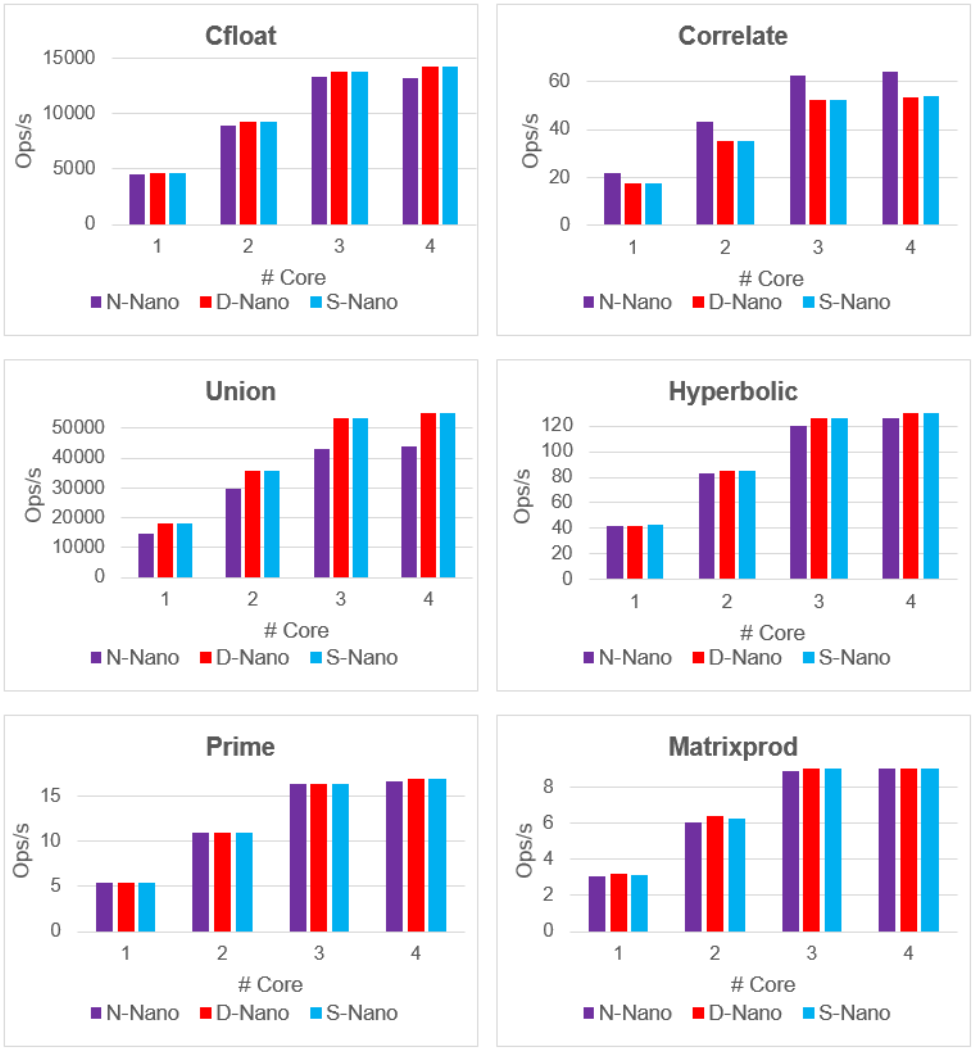
Los resultados mostrados por las pruebas dejan claro que no existe una arquitectura que cubra todas las necesidades de todas las aplicaciones. Cada dispositivo cubre ciertas necesidades y tiene una mejor eficiencia en ciertas cargas de trabajo. Por ello, se debe hacer una buena caracterización de la aplicación para conocer a detalle cuáles son sus requisitos de hardware-software y poder seleccionar un dispositivo adecuado.

Cada una de las Figuras comparativas evidencia que el método de despliegue no influye en la eficiencia de las aplicaciones. En la mayoría de las pruebas no se ve una gran variación en los resultados, por ello, se debe verificar el impacto en el rendimiento de los métodos de despliegue para confirmar si se presenta algún impacto importante.

## 8.2 IMPACTO EN EL RENDIMIENTO SEGÚN SU DESPLIEGUE

Las gráficas mostradas a continuación presentan la cantidad de Ops/s de cada dispositivo en las diferentes pruebas según su método de despliegue.

**Figura 17. Rendimiento de la Nano en las diferentes pruebas.**



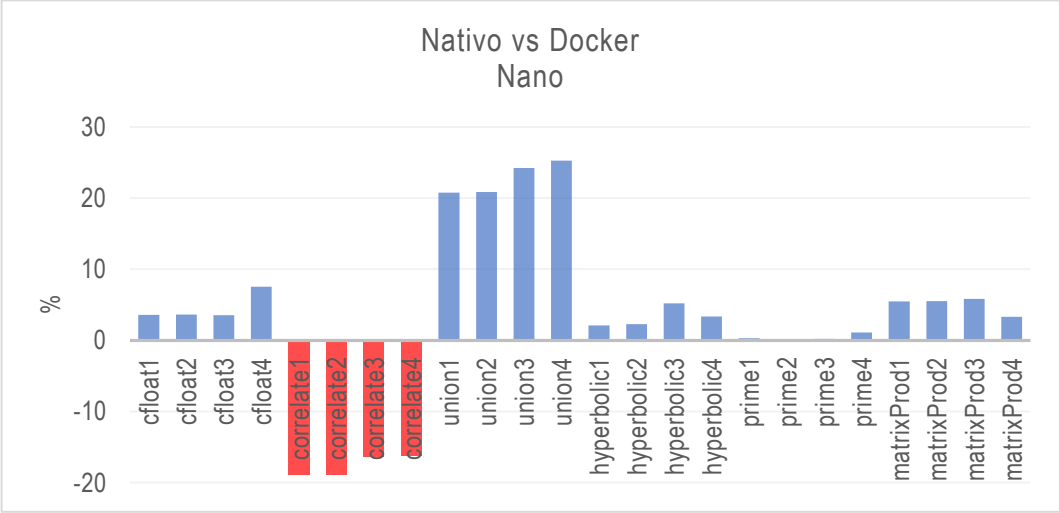

Debe quedar claro que la Figura 17 no es una comparacion entre las pruebas, cada grafica muestra los resultados de cada prueba. Lo que se compara son los diferentes metodos de despliegue que se muetran en cada grafica.

Se puede ver como escala el rendimiento a medida que se varia la configuracion de recursos, en el caso de la Nano se aprecia un aplanamiento a medida que los recursos aumentan, esto se debe a la pelea por los resursos entre el OS y la

aplicacion que se ejecuta. Una manera de evitar este problema es configurar la carga de trabajo de tal manera que un Core quede libre y se encarge de las necesidades del OS o reducir la carga de trabajo a un punto en el que se equilibre la ejecucion de la aplicación y el OS cuando se usan todos los Cores.

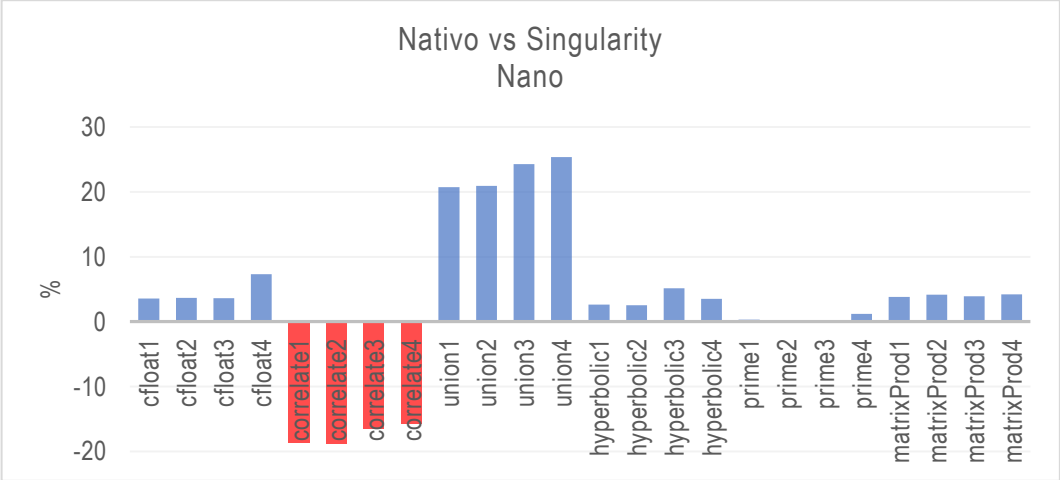
Con la Figura 17 se procede a comparar los resultados obtenidos en los despliegues en Nativo y Docker. Dicha comparación de presenta en la Figura 18, esta comparación mide le impacto en el rendimiento cuando se realiza un despliegue sobre la Nano. Para medir el impacto en el rendimiento se toma como base el despliegue en Nativo y se mide la diferencia con Docker.

**Figura 18. Impacto en el rendimiento de la Nano, despliegue Nativo Vs Docker.**



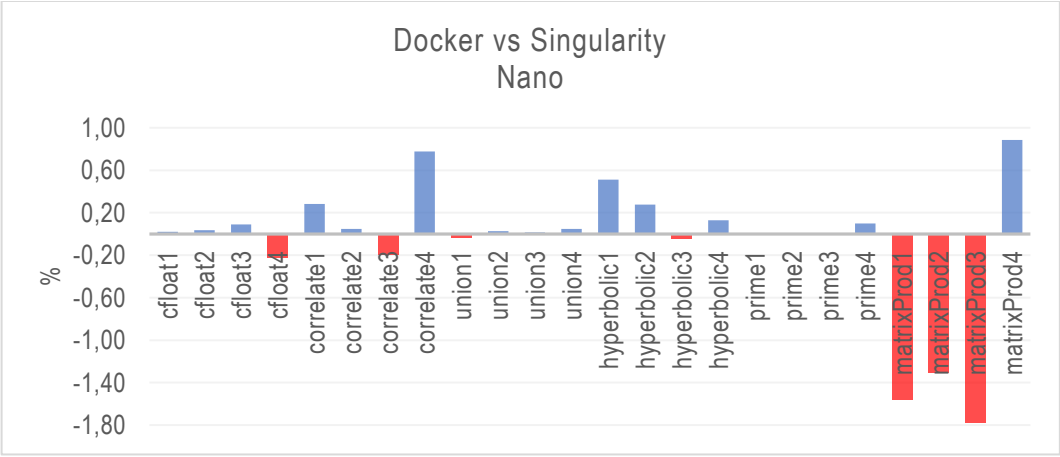

En general, el despliegue en Docker es beneficioso, mejora en hasta un 25% el rendimiento de las pruebas. Solo existe una pérdida de un 18% en la prueba de Correlate, pero no opaca los beneficios que presenta la contenerización.

**Figura 19. Impacto en el rendimiento de la Nano, despliegue Nativo Vs Singularity.**



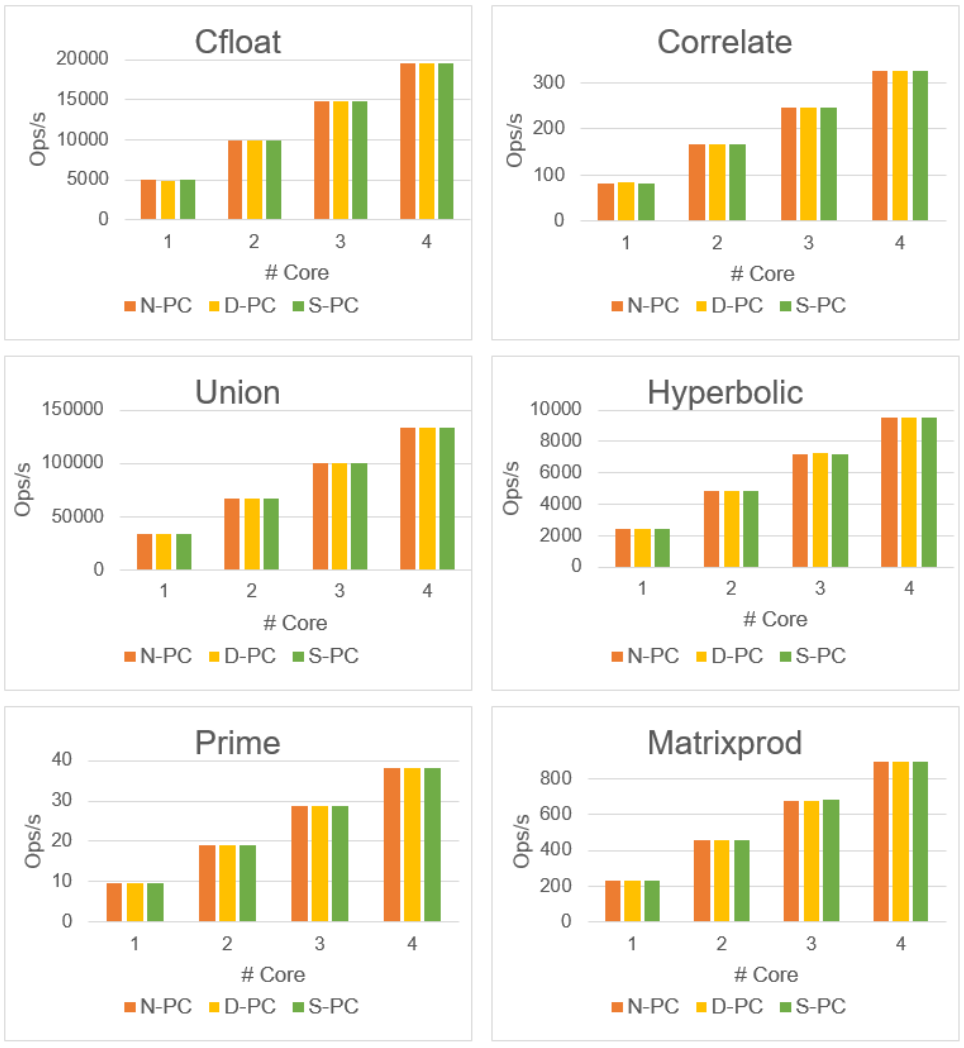

La Figura 19 compara el rendimiento del despliegue en Nativo y Singularity. El comportamiento es similar al mostrado en la Figura 18. Esto reafirma que la contenerizacion mejora el rendimiento de las aplicaciones en la Nano, pero surge la incognita de cual contenedor da un mejor rendimiento.

**Figura 20. Impacto en el rendimiento de la Nano, despliegue Docker Vs Singularity.**



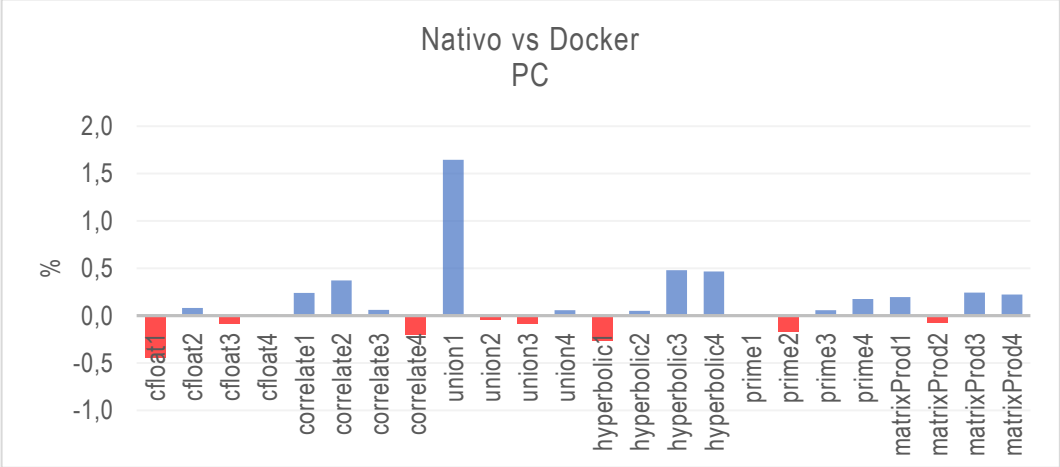
La Figura 20 muestra la comparación entre Docker y Singularity, la base de la diferencia es Docker. Esta comparación muestra un comportamiento parejo en la mayoría de las pruebas, solo en la prueba de Matrixprod se ve una diferencia, pero no supera el 2%. En conclusión, la contenerización en la Nano aporta un mejor rendimiento en la mayoría de las pruebas.

**Figura 21. Rendimiento de la PC en las diferentes pruebas.**



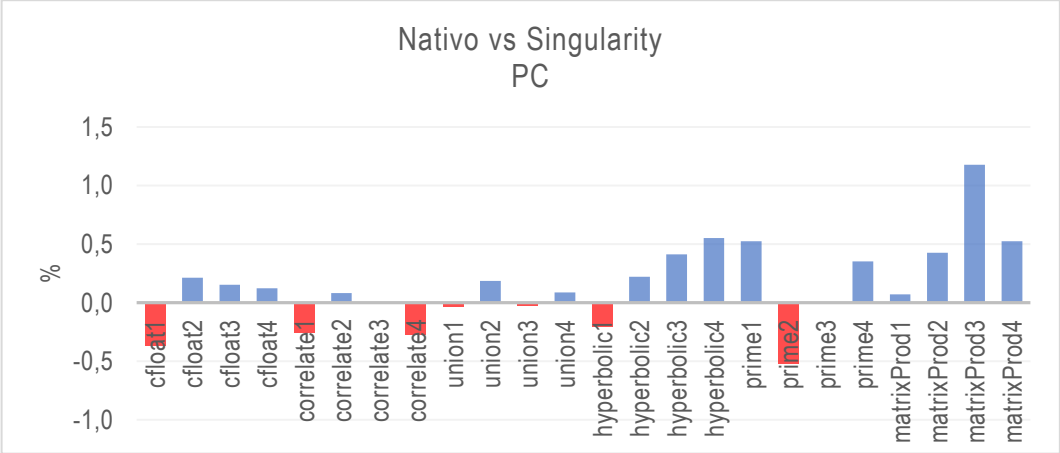
En la Figura 21 se presenta el rendimiento de la PC en cada prueba. Al igual que en la Figura 17 se tiene un comportamiento escalonado a medida que se aumentan los recursos para las pruebas. También se debe aclarar que, al tener mayor cantidad de núcleos, no se percibe el aplanamiento en el rendimiento que si se presenta en la Nano.

**Figura 22. Impacto en el rendimiento de la PC, despliegue Nativo Vs Docker.**



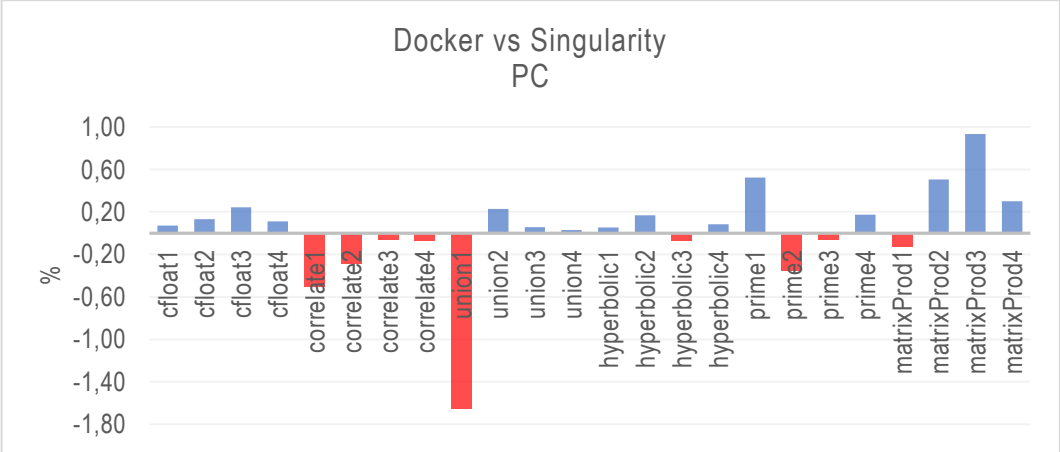

La Figura 22 presenta el impacto en el rendimiento de la PC en los despliegues de Nativo y Docker, la base que se usa es el despliegue en Nativo. En este caso las pruebas tienen un mejor rendimiento en Docker pero en promedio no supera el 0,7%.

**Figura 23. Impacto en el rendimiento de la PC, despliegue Nativo Vs Singularity.**




La Figura 23 presenta la comparación entre despliegue Nativo y Singularity, la base es de nuevo el despliegue Nativo. En esta comparación se aprecia una mejora en el rendimiento cuando se despliega con Singularity, pero no es tan significativa. Así que cualquiera de los métodos puede ser elegido para realizar el despliegue de una aplicación sobre la PC.

**Figura 24. Impacto en el rendimiento de la PC, despliegue Docker Vs Singularity.**



La Figura 24 compara el rendimiento de la PC en Docker y Singularity, se toma como base a Docker. Tanto Docker como Singularity tienen un rendimiento similar, aunque Singularity logra sacar un poco más de rendimiento en la mayoría de las pruebas. En conclusión, la contenerización aporta un mejor rendimiento en la mayoría de las pruebas, pero queda a preferencia del usuario decidir por una u otra opción de contenerización.

En conclusión, el impacto sobre el rendimiento según su método de despliegue es mínimo para arquitecturas grandes y se evidencia una mejora en el rendimiento en arquitecturas pequeñas. Por ello no se debe considerar la pérdida de rendimiento como un factor para descartar la contenerización y se conservan todas sus ventajas. En cuanto a los dos contenedores, quien presenta mayor flexibilidad y simplicidad es Singularity, debido a su enfoque para entornos HPC, no necesitar permisos de root y la simplicidad al traducir archivos de construcción de Docker (dockerfile).

En la mayoría de las pruebas no se ve una gran variación en los resultados, por ello, se verifica el impacto en los tiempos de ejecución de los métodos de despliegue para confirmar si se presenta algún impacto importante.

## 8.3 IMPACTO EN LOS TIEMPOS DE EJECUCIÓN SEGÚN SU DESPLIEGUE

Un punto importante al momento de desplegar una aplicación es su tiempo de ejecución, este punto ha sido una métrica importante al momento de decidir el método de despliegue de una aplicación, convirtiéndose en una desventaja importante de la virtualización.

Se debe aclarar que el tiempo de la prueba se elimina por dos razones:

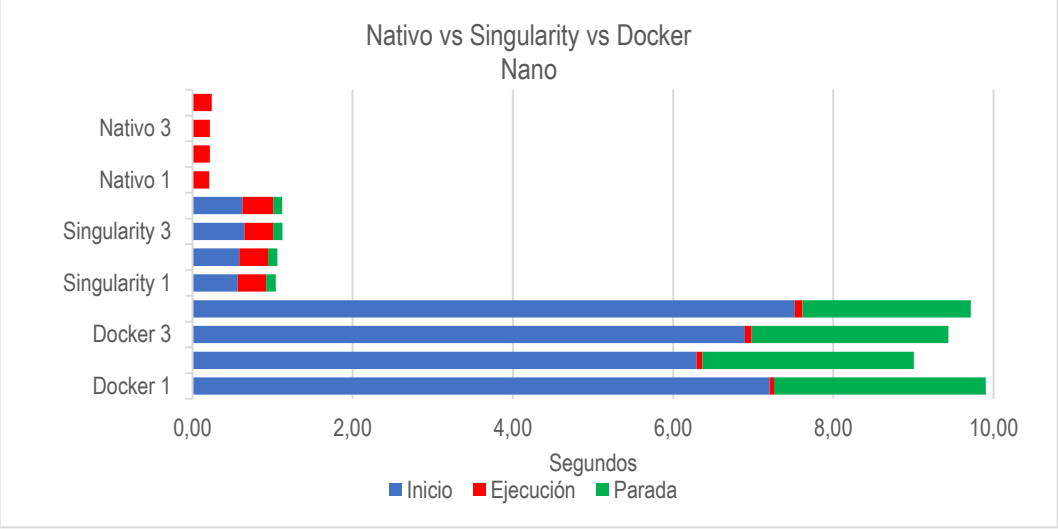
1. Todas las pruebas se realizan durante la misma cantidad de tiempo eliminando su importancia como variable de medida.
2. Se da una mejor visibilidad de los tiempos que se agregan al tiempo de la prueba.

Las pruebas en Nativo son las que menor tiempo agregado presentan, el tiempo mostrado es el tiempo que tarda el OS en asignar los recursos para ejecutar la prueba, a este tiempo se le denomina Ejecución. Los despliegues en la Nano de Docker y Singularity tienen dos tiempos más que son:

1. Inicio es el tiempo que demora en ser iniciado un contenedor.
2. Parada es el tiempo que toma en detener el contenedor y liberar los recursos que fueron asignados durante la ejecución.

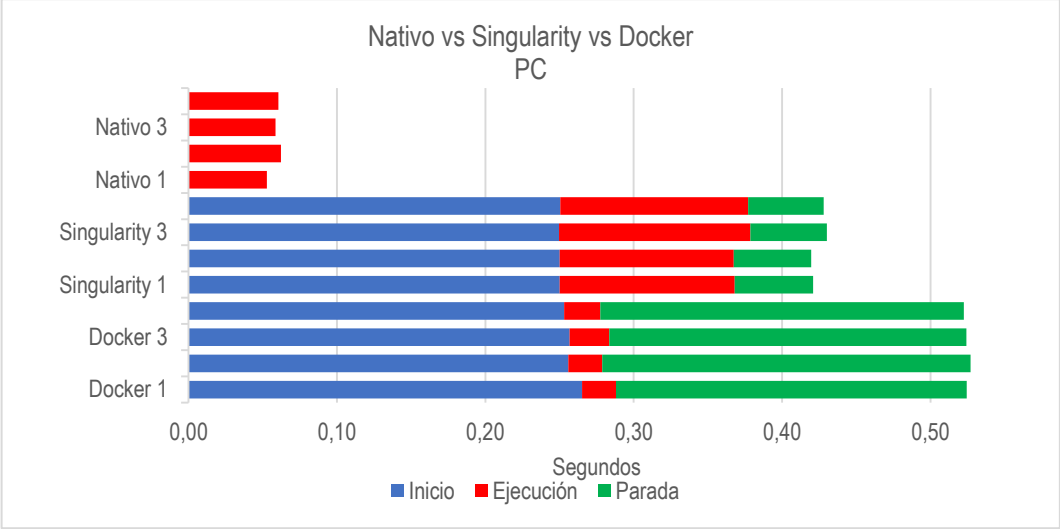
La variación de los tiempos en cada una de las pruebas no era muy diferente, por este motivo se promedian los tiempos y se presentan en las siguientes figuras.

**Figura 25. Impacto en el Tiempo de Ejecución de las aplicaciones en la Nano.**



La Figura 25 evidencia que Singularity logra tiempos muy reducidos, poco más de 1 s, no supera al despliegue en Nativo, pero deja atrás a Docker y sus casi 10 s. Esto mejora la posición de Singularity como opción de contenerización para el despliegue de aplicaciones.

**Figura 26. Impacto en el Tiempo de Ejecución de las aplicaciones en la PC.**




La Figura 26 muestra el impacto en los tiempos de las pruebas en la PC, el despliegue en Nativo es el que menos tiempo genera realizando las pruebas. El tiempo de Inicio es similar para ambos contenedores (±0,25 s), pero Singularity marca la diferencia con el tiempo de Parada (±0,05 s). Docker es el que menos tiempo usa asignando los recursos de ejecución en ambos dispositivos.

Con los resultados presentados en la Figura 25 y Figura 26 se puede afirmar que los contenedores son una buena opción para el despliegue de aplicaciones sobre diferentes arquitecturas. Esto se debe a que, en la mayoría de los casos, aumentan el rendimiento de las aplicaciones, no afecta en gran manera la eficiencia de las aplicaciones y su impacto en el tiempo total de ejecución disminuye a medida que

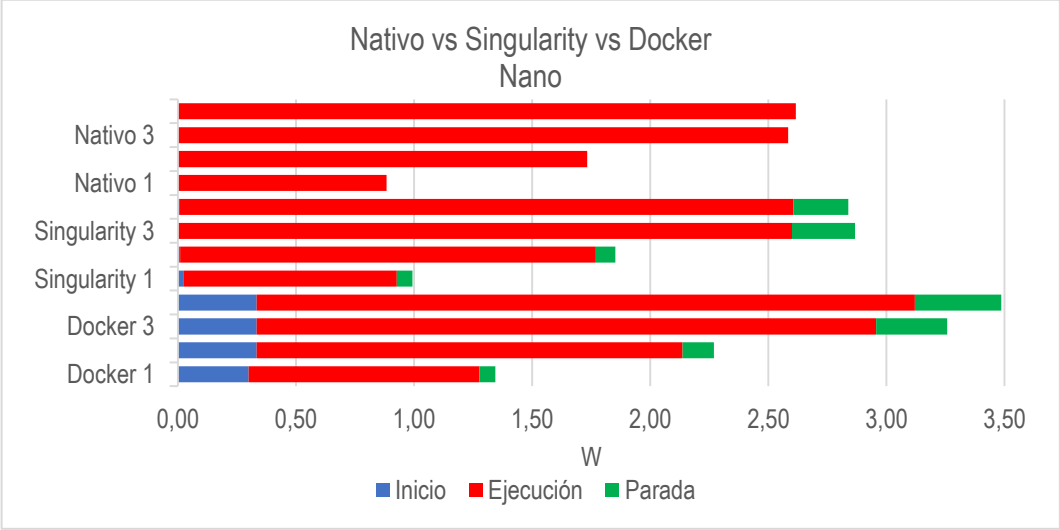
la escala de los recursos aumenta. Se verifica el impacto en el consumo energético de los métodos de despliegue para confirmar si se presenta algún impacto importante.

## 8.4 IMPACTO EN EL CONSUMO SEGÚN SU DESPLIEGUE

Las siguientes figuras muestran el consumo sobre los dispositivos, Las franjas de colores representan lo siguiente:

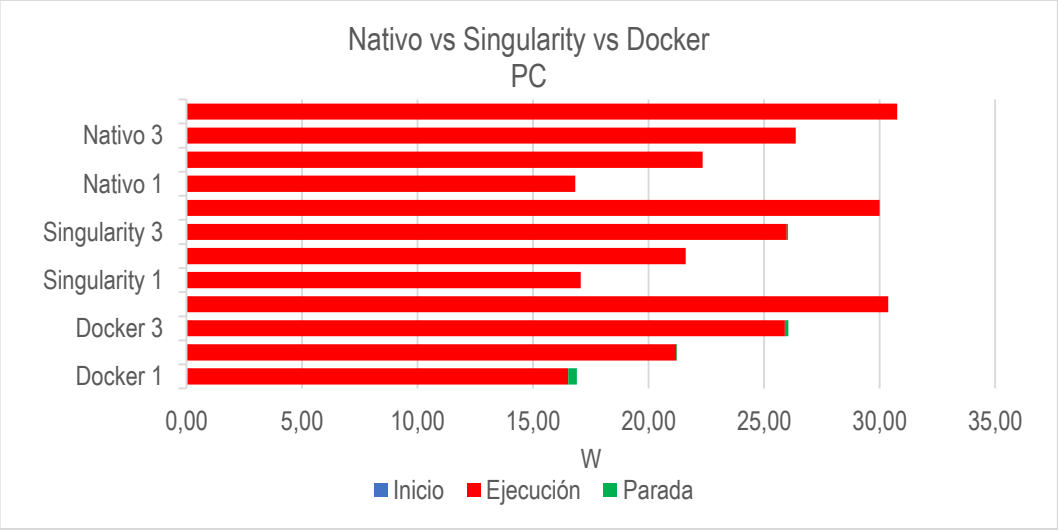
- Franja roja es el consumo promedio durante la ejecución de las pruebas.
- Franja azul es el consumo promedio durante el Inicio del contenedor.
- Franja verde es el consumo promedio durante la Parada del contenedor.

**Figura 27. Impacto en el Consumo de las aplicaciones en la Nano según su despliegue.**



En la Figura 27, el impacto en el consumo de la Nano muestra que Singularity tiene consumos mínimos durante su Inicio, imperceptibles para el monitor de consumo, solo en la parada se hace visible. Por otro lado, Docker presenta un consumo similar al iniciar un contenedor, independientemente de los recursos que se asignen, aumentando el consumo en las pruebas de manera pareja y en la Parada solo al llegar a la máxima cantidad de recursos se ve un mayor consumo en todas las franjas.

**Figura 28. Impacto en el Consumo de las aplicaciones en la PC según su despliegue.**




La Figura 28 presenta el consumo energético promedio de las pruebas según su método de despliegue en la PC, esto evidencia que el impacto de los contenedores sobre el consumo energético es mínimo. Parte de ese resultado se debe al tiempo de inicio/parada que en el caso de la PC era muy corto y el medidor de consumo no alcanza a registrar. Como se habló en la comparación (8.1), rendimiento (8.2) y tiempos (8.3), solo cuando las capacidades computacionales son pequeñas podemos ver un impacto de los diferentes mecanismos de despliegue. En la Figura

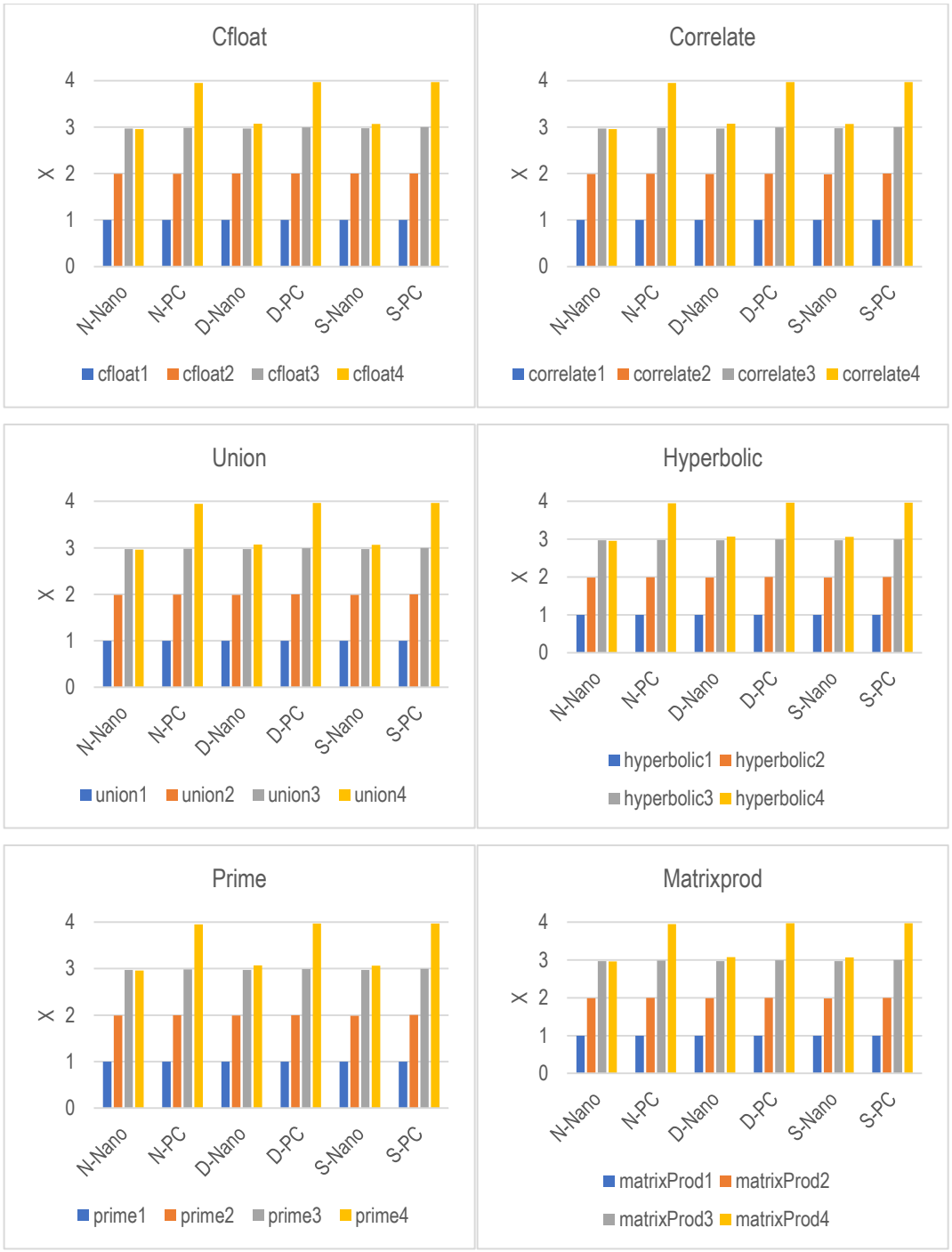
27 se evidencian estos aumentos, pero al aumentar las capacidades computacionales estos impactos se vuelven casi imperceptibles, como en la Figura 28. Esto se relaciona directamente con los tiempos de ejecución, en el caso de la PC son muy cortos, impidiendo que el monitor de consumo pueda detectar si existe alguna variación y aunque el consumo en la PC se evidencie, es menor al de la Nano.

En conclusión, a medida que la escala (tamaño y capacidades) de los dispositivos aumenta, el impacto de los contenedores sobre el consumo energético y los tiempos de ejecución disminuye. Como ultima verificación, se revisa el impacto en la escalabilidad de los métodos de despliegue para confirmar si se presenta algún impacto importante.

## 8.5 IMPACTO EN LA ESCALABILIDAD SEGÚN SU DESPLIEGUE

Para finalizar se presenta el impacto que tiene el método de despliegue sobre la escalabilidad de una aplicación. Estas pruebas no miden picos de rendimiento sino cuantas operaciones pueden ser realizadas durante un tiempo fijo con un porcentaje de carga de trabajo definido en una prueba determinada. Por ello, se selecciona la cantidad de núcleos como la variable para medir la escala, esto se debe a que los resultados son más visibles cuando se cambia la cantidad de Cores. Si se mantiene fija y se modifican otros parámetros, los resultados no sufren cambios importantes.

**Figura 29. Impacto en la escalabilidad de las aplicaciones segun su Despliegue.**



La Figura 29 muestra como aumenta la cantidad de operaciones o, como escala la aplicación cuando se asignan recursos. Se debe entender que la cantidad de operaciones no es igual para la Nano y la PC, aunque se vea así en la figura. Las barras son iguales debido a que los resultados son proporcionales, se toma la cantidad de operaciones que se realizan con un solo núcleo y se saca la proporción cuando se usan más recursos. Ese es el valor denominado como X o la cantidad de veces que aumentan las operaciones cuando se asignan recursos.

Los resultados de la Figura 29 muestran que las aplicaciones usadas para el estudio escalan de manera lineal cuando se les asignan recursos, independientemente del método de despliegue, las capacidades o tamaño del dispositivo. Esto se puede usar como base para asegurar que desarrollar aplicaciones en dispositivos de bajo costo no limita su implementación en arquitecturas de mayor escala. Incluso, puede ser tomado como ventaja en aquellos proyectos donde el presupuesto sea limitado y se necesite de resultados preliminares para la obtención de recursos de mayor escala para la finalización del proyecto.

Se concluye que el impacto sobre la eficiencia, el rendimiento y la escalabilidad es mínimo independientemente de su método de despliegue, esto afirma que los contenedores son una herramienta excelente para el despliegue de aplicaciones. También, los resultados presentados reflejan que el impacto en los tiempos de ejecución o el consumo se reduce a medida que los recursos computacionales o la escala de los dispositivos aumentan. Otro punto a favor de los contendores es que, con un solo archivo de construcción de una imagen, se puede desplegar la misma aplicación en diferentes arquitecturas, reduciendo la creación de imágenes que fallen al momento de ser iniciadas.

# 9. CONCLUSIONES

El principal enfoque de las arquitecturas Post-Moore es la integración de chips de propósito específico, esto provee a las arquitecturas Post-Moore de fabricación tradicional de dos características fundamentales: ser multinúcleo para aprovechar el paralelismo y altamente heterogéneas para cubrir necesidades específicas de las aplicaciones. Las características Post-Moore identificadas permiten la selección de dispositivos de bajo costo monetario y bajo consumo energético que pueden ser usados en el despliegue de una aplicación en diferentes escalas (tamaño, cantidad de datos, cantidad de hilos, etc.) de manera fluida, flexible y simple usando los mecanismos de despliegue presentados en el estudio.

Los dispositivos Post-Moore seleccionados usando características tales como multinúcleo y heterogeneidad pueden aprovecharse en sistemas embebidos, IoT o Edge Computing como vehículos autónomos, domótica, robótica, microsatélites, etc. En estos dispositivos pueden implementar aplicaciones de IA, ML o DL que permiten un funcionamiento autónomo completo como se muestra en la sección 5.

La estrategia y lineamientos propuestos en la sección 6 son evaluados por los mecanismos expuestos en la sección 7, de esta manera se evidencian su impacto en las aplicaciones. Los resultados de la sección 8 demuestran que dichas propuestas son beneficiosas para el despliegue de las aplicaciones, mejorando en algunos casos el rendimiento. Las métricas y flujo de pruebas propuestas son viables para el desarrollo de otros estudios que implementen un banco de pruebas diferente. En resumen, la estrategia utilizada en el estudio generaliza la selección y evaluación de dispositivos Post-Moore de fabricación tradicional.

Para evitar problemas con la ejecución de imágenes en los contenedores es recomendable desarrollar un archivo de construcción como se plantea en los lineamientos. Es recomendable compartir dicho archivo y no la imagen generada. El archivo puede usarse para despliegues en diferentes contenedores, solo se debe hacer la traducción al formato de cada contenedor. Si se desea realizar un despliegue rápido por parte de usuarios con pocos conocimientos, la contenerización es la opción más viable.

El objetivo principal de este trabajo es evaluar el impacto en el rendimiento computacional según su despliegue durante la ejecución de aplicaciones sobre arquitecturas Post-Moore. Los resultados dejan claro que el impacto sobre la eficiencia, el rendimiento o la escalabilidad es mínimo independientemente de su método de despliegue. El impacto en los tiempos de ejecución o consumo se reduce a medida que los recursos computacionales y el tamaño de los dispositivos aumentan. Dicho impacto para la eficiencia, rendimiento o escala es la reducción de la cantidad de Ops cuando se usa uno u otro método de despliegue. El impacto en los tiempos de ejecución o consumo es el tiempo agregado y consumo adicional al ejecutar aplicaciones usando los diferentes métodos de despliegue.

El impacto percibido en escalas pequeñas depende de la disponibilidad de recursos o del manejo que el Sistema Operativo (OS) le dé al dispositivo Post-Moore. Esto evidencia la necesidad de optimizar el OS para ejecutar la aplicación o la implementación de contenedores, ya que estas arquitecturas están compuestas por hardware que facilita la configuración a medida para sacar la mayor eficiencia computacional de una aplicación. No se debe considerar la pérdida de rendimiento como un factor para descartar la contenerización como un método de despliegue.

# 10. TRABAJOS FUTUROS

Desarrollar un manejador de paquetes basado en contenedores, los manejadores de paquetes simplifican la implementación de las aplicaciones y los contenedores albergaran todo lo necesario para el funcionamiento de una aplicación. El manejador de paquetes puede generar un archivo de construcción a la medida para la arquitectura en que se va a desplegar, simplificado la instalación, mantenimiento, actualización o eliminación de la aplicación.

El estudio se puede usar como base para la investigación y desarrollo de aplicaciones en dispositivos embebidos. Por ello, es necesario probar arquitecturas embebidas con mayores capacidades para evidenciar el impacto de las aplicaciones sobre estas arquitecturas y reforzar la estrategia, lineamientos, métricas y flujo de pruebas presentados en este estudio.

Usando la estrategia, lineamientos de despliegue, métricas y flujo de pruebas se pueden realizar estudios sobre otros dispositivos como FPGA, DPU, Unidades de procesamientos emergentes, etc.

# BIBLIOGRAFÍA


«CUDA.,» [En línea]. Available: https://developer.nvidia.com/cuda-zone. [Último acceso: 1 Febrero 2020]

«Foro Histórico de las Telecomunicaciones,» 2019. [En línea]. Available: https://forohistorico.coit.es/index.php/personajes/personajes-internacionales/item/moore-gordon-e. [Último acceso: 6 Febrero 2021]

«Khronos OpenCL Registry,» Khronos Group, 27 Abril 2020. [En línea]. Available: https://www.khronos.org/registry/OpenCL/. [Último acceso: 10 Julio 2020]

ALTSCHULER, F. y GALLMEIER, J. «Heterogeneous system architecture: Multicore image processing using a mix of CPU and GPU elements.,» *Embedded Computing Design,* 2012

Apple, «MacOS,» [En línea]. Available: https://www.apple.com/co/macos/big-sur/. [Último acceso: 17 Febrero 2021]

Asus, «Asus Tinker Board,» [En línea]. Available: https://tinker-board.asus.com/product/tinker-board.html. [Último acceso: 23 Febrero 2021]

BENEDICIC, L.; CRUZ, F. A.; MADONNA, A. y MARIOTTI, K. «Portable, high-performance containers for HPC,» 2017

Canonical Ltd., «Linux Containers,» [En línea]. Available: https://linuxcontainers.org/. [Último acceso: 19 Febrero 2021]

GÓMEZ HERNÁNDEZ, C. E. Análisis de Sistemas Embebidos HPC usando Manejadores de Paquetes, Bucaramanga: Universidad Industrial de Santander, 2020

HALE, J. S.; LI, L.; RICHARDSON, C. N. y WELLS, G. N.«Containers for Portable, Productive, and Performant Scientific Computing,» *Computing in Science & Engineering,* pp. 40 - 50, 2017

Hardkernel, «Odroid,» [En línea]. Available: https://wiki.odroid.com/. [Último acceso: 23 Febrero 2021]

KOVÀCS, À. «Comparison of different Linux containers,» *40th International Conference on Telecommunications and Signal Processing (TSP),* pp. 47-51, 2017

KUITY, A. y PEDDOJU, S. K. «Performance Evaluation of Container-Based High Performance Computing Ecosystem Using OpenPOWER,» *International Conference on High Performance Computing,* pp. 290-308, 2017

KUSZYK, A. y HAMMOUDEH, M. «Contemporary Alternatives to Traditional Processor Design in the Post Moore's Law Era,» *ICFNDS'18: International Conference on Future Networks and Distributed Systems,* pp. 1-5, 2018

KYRIAZIS, G. «Heterogeneous System Architecture: A Technical Review,» AMD, 2012

LOPEZ, G. «Teldat Blog,» 3 Octubre 2018. [En línea]. Available: https://www.teldat.com/blog/es/contenedores-contenerizacion-virtualizacion-de-sistema-operativo-sd-wan/. [Último acceso: 11 Septiembre 2020

MATSUOKA, «Cambrian explosion of computing and big data in the post-moore era,» *HPDC '18: Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing,* p. 105, 2018

MATSUOKA, S.; AMANO, H.; NAKAJIMA, K.; INOUE, K.; KUDOH, T.; MARUYAMA, N.; TAURA, K.; IWASHITA, T.; KATAGIRI, T.; HANAWA, T. y ENDO, T. «From FLOPS to BYTES: disruptive change in high-performance computing towards the post-moore era,» *CF '16: Proceedings of the ACM International Conference on Computing Frontiers,* p. 274–281, 2016

Microsoft, «Windows,» [En línea]. Available: https://www.microsoft.com/es-co/windows. [Último acceso: 16 Febrero 2021]

NERSC, «Shifter - Containers for HPC,» [En línea]. Available: https://github.com/NERSC/shifter. [Último acceso: 20 Febrero 2021]

Nvidia Developer, «Jetson Nano,» [En línea]. Available: https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/. [Último acceso: 24 Febrero 2021]

Nvidia, «NGC Catalog,» [En línea]. Available: https://www.nvidia.com/en-us/gpu-cloud/. [Último acceso: 21 Febrero 2021]

ORACLE, «VirtualBox,» Oracle, [En línea]. Available: https://www.virtualbox.org/wiki/Documentation. [Último acceso: 10 Febrero 2021]

QEMU, «QEMU,» [En línea]. Available: https://www.qemu.org/. [Último acceso: 12 Febrero 2021]

QUIROGA, D. «NAMESPACES - AÍSLAR LOS PROCESOS GNU/LINUX EN SUS PROPIOS ENTORNOS DE SISTEMA,» [En línea]. Available: https://clibre.io/blog/por-secciones/codigo/item/384-namespaces-aislar-los-procesos-de-linux-en-sus-propios-entornos-de-sistema. [Último acceso: 15 Febrero 2021]

Raspberry Foundation, «Raspberry pi,» [En línea]. Available: https://www.raspberrypi.org/products/. [Último acceso: 22 Febrero 2021]

Red Hat, «KVM,» [En línea]. Available: http://www.linux-kvm.org/page/Guest_Support_Status. [Último acceso: 18 Febrero 2021]

RODRIGUEZ, R. «Recetas Docker,» 2017. [En línea]. Available: https://recetas-docker.readthedocs.io/es/latest/index.html. [Último acceso: 25 Febrero 2021]

RUIZ, C.; JEANVOINE, E. y NUSSBAUM, L. «Performance evaluation of containers for HPC,» *European Conference on Parallel Processing,* pp. 813-824, 2015

VMware, «VMware,» [En línea]. Available: https://www.vmware.com/company.html. [Último acceso: 11 Febrero 2021]

VTA, «Tomacorriente Inteligente VTA-84630,» [En línea]. Available: https://www.vta.co/wp-content/uploads/2021/08/VTA-84630_manual_web.pdf. [Último acceso: 24 Febrero 2021]

Xunlong Software CO, «Orange pi,» [En línea]. Available: http://www.orangepi.org/Docs/mainpage.html. [Último acceso: 23 Febrero 2021]

YEPES, P. J. R.; HERNANDEZ, C. J. B. y STEFFENEL, L. A. «Low Energy Consumption on Post-Moore Platforms for HPC Research,» *ACI Avances en Ciencias e Ingenierías,* 2021

YOUNGE, A. J.; PEDRETTI, K.; GRANT, R. E. y BRIGHTWELL, R. «A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds,» *Cloud Computing Technology and Science (CloudCom),* pp. 74-81, 2017