Artículo de Investigación

Aibi revista de investigación, administración e ingeniería. Volumen 13, Número 3, Pág 01-10, Septiembre - Diciembre de 2025, ISSN 2346-030X

Clúster de computadoras open source para Password Cracking.
Open source computer cluster for Password Cracking.

William Rogelio Marchand-Niño[1], Guilmar Ciriaco-Soto[2], Elkin Fausto Ortiz-Morales[3], Gardyn Olivera-Ruiz[4], Einstein-Arnold Ortiz-Morales[5]

[1,2,3,4,5]*Universidad Nacional Agraria de la Selva, Tingo María - Perú*

Recibido: 20 de febrero de 2025.
Aceptado: 12 de julio de 2025.
Publicado: 01 de septiembre de 2025.

Resumen- Este estudio se centra en la evaluación de la eficiencia en término de tiempo de ejecución para procesos que demandan una significativa capacidad computacional como el password cracking o adivinación de contraseñas a partir del hash de una contraseña, por lo que el problema que se abordó fue ¿Cuál es el nivel de eficiencia del clúster de computadoras open source para el procesamiento de password cracking con el propósito de validar la fortaleza de contraseñas? Para este propósito se utilizó un clúster de computadoras de dos nodos bajo un sistema operativo Linux y OpenMPI, y para el proceso de descubrimiento de contraseñas se utilizó la herramienta Jhon the Ripper (JTR). La metodología aplicada contempla cuatro etapas: selección de la plataforma de hardware, selección de la plataforma de software, implementación del clúster y las pruebas del clúster con dos escenarios, las pruebas con hashes MD5 y SHA-1, y las pruebas de hashes de sistemas operativos NT (para contraseñas de sistemas Windows) y SHA-512 (para contraseñas del sistema operativo Linux Ubuntu 20.04), cuyos resultados indican una disminución entre 52% y 64% del tiempo de procesamiento para descubrir una contraseña con el clúster en su máximo aprovechamiento de procesadores (16 en total) en comparación con el mismo procedimiento pero con una sola computadora o nodo. Por lo tanto, el nivel de eficiencia encontrado es positivamente significativo.

Palabras clave: clúster, HPC, password cracking, ciberseguridad, contraseñas**.**

Abstract— This research focuses on the evaluation of efficiency in execution time for processes that demand significant computational capacity such as password cracking or password guessing from a hash, so the defined problem was, what is the level of efficiency of the open source computer cluster for password cracking processing for the purpose of validating password strength? For this purpose, a two-node computer cluster was used under a Linux operating system and OpenMPI, and the John the Ripper (JTR) tool was used for the password discovery process. The applied methodology includes four stages: selection of the hardware platform, selection of the software platform, cluster implementation and cluster testing, with two scenarios, testing with MD5 and SHA-1 hashes, and testing NT operating system hashes (Windows systems) and SHA-512 (Ubuntu 20.04 Linux operating system). The results of which indicate a decrease between 52% and 64% in the processing time to discover a password with the cluster at its maximum use of processors (16 in total) compared to the same procedure but with a single computer or node. Therefore, the level of efficiency found is positively significant.

Keywords: cluster, HPC, password cracking, cybersecurity, passwords.

*Autor para correspondencia.
Correo electrónico: william.marchand@unas.edu.pe (William Rogelio Marchand Niño).
La revisión por pares es responsabilidad de la Universidad de Santander.

Como citar este artículo: W. R. Marchand-Niño, G. Ciriaco. Soto, E. F. Ortiz-Morales, G. Olivera-Ruiz y E. A. Ortiz-Morales, “Clúster de computadoras open source para Password Cracking”, Aibi revista de investigación, administración e ingeniería, vol. 13, no. 3, pp. 01-10 2025, doi: 10.15649/2346030X.4242

I. INTRODUCCIÓN

La computación de alto rendimiento (HPC, por sus siglas en inglés) desempeña un papel fundamental en la resolución de problemas complejos que abarcan una amplia gama de disciplinas como la criptografía o simulaciones complejas, y una alternativa de solución relacionada con la HPC es el clúster de computación bajo código abierto porque no siempre está al alcance de todas las organizaciones la HPC comercial. La computación de alto rendimiento tiene un impacto en la informática moderna que es innegable, ha desempeñado un papel fundamental en la aceleración de avances científicos, en la toma de decisiones basada en datos y en la creación de tecnologías de punta que están dando forma a nuestro mundo. A medida que continuamos enfrentando desafíos cada vez más complejos y demandantes en términos de procesamiento de datos, la tecnología de clústeres sigue siendo un pilar fundamental en el panorama tecnológico actual.

Por otro lado, la ciberseguridad actualmente es un factor estratégico para las organizaciones independientemente de su naturaleza privada o pública; sin embargo, los riesgos de seguridad informática que enfrentan las empresas son significativamente altas; considerando además, que la pandemia del Covid-19 aceleró la adopción de tecnología informática, en muchos casos sin el componente de ciberseguridad adecuado, produciendo diversos tipos de ataques de exfiltración de datos y perdidas de grandes sumas de dinero.

De hecho, las estadísticas indican que los niveles de cibercrimen siguen en niveles alarmantes, tal es así que, en Latinoamérica durante el año 2023, el 69% de empresas aproximadamente ha sufrido algún tipo de incidente de ciberseguridad, siendo las principales preocupaciones, el robo de información y el acceso indebido a sistemas (Eset, 2023), por ejemplo, los robos de información que incluyen datos personales y credenciales de acceso (nombre de usuario y contraseña cifrada o en texto plano en el peor de los casos) [1].

Las campañas de phishing también es otro de los tipos de ataques preferidos por los ciberdelincuentes con las que intentan entregar software malicioso (*malware*) como *ransomware* y troyanos, con propósitos de exfiltración de información (Positive Technologies, 2023). Entonces, es evidente que las personas siguen siendo el principal objetivo de los ciberdelincuentes para obtener el acceso inicial a los demás activos de información; y las personas también están involucradas con el uso de contraseñas, en las que aún se encuentran deficiencias como: débil nivel de concienciación y compromiso de los usuarios relacionado con el establecimiento y uso de contraseñas (cumplimiento de políticas de complejidad y fortaleza), resistencia de los usuarios para actualizar o cambiar sus contraseñas periódicamente, esto tiene consecuencias negativas porque los ciberdelincuentes se aprovechan de la longevidad de las contraseñas o la reutilización de estas a través del tiempo.

La verificación de la fortaleza de la contraseña se convierte en una necesidad de las empresas. Los administradores de sistemas se ven ante un desafío para mitigar los riesgos por el uso de contraseñas que pueden descubrirse poco tiempo, y si se hace uso de potencia computacional suficiente, resulta en una tarea sencilla para los atacantes.

Por lo tanto, las empresas se pueden ver en la necesidad de acceder a recursos computacionales suficientes, para procesar y verificar la fortaleza de contraseñas que ayude a mitigar el uso de términos o cadenas de caracteres adivinables haciendo uso de las mejores técnicas y herramientas vigentes de *password cracking*. Estos recursos computacionales representados por supercomputadoras (computación de alto rendimiento) no siempre está al alcance de todas las organizaciones, y el no acceder a recursos de cómputo de alto rendimiento, pone en desventaja estratégica a los defensores de los sistemas frente a los atacantes cibernéticos.

Este estudio se centra en la evaluación de la eficiencia mediante la ejecución de procesos que demandan una significativa capacidad de procesamiento computacional, por lo que el problema que se abordó fue ¿Cuál es el nivel de eficiencia del clúster de computadoras *open source* para el procesamiento de *password cracking* con el propósito de validar la fortaleza de contraseñas?

Uno de los primeros aspectos a considerar, son los estudios relacionados con la implementación y configuración de clúster de HPC, y en ese sentido trabajos como los de García-Ojeda et al. [2], en el cual se implementó un clúster de computadoras para la gestión de big data, basado en componentes como Apache Hadoop para el almacenamiento distribuido (HDFS), cómputo distribuido (MapReduce), y gestión de tareas en el clúster. También abordaron el factor energético, concluyendo que en general el clúster consume un promedio de 66% más energía que un equipo unitario, sin embargo, el rendimiento del clúster es mayor en un 25% respecto a un equipo unitario sobre las mismas tareas.

A propósito del factor energético, en contraste a lo corroborado en la investigación anterior, Kocot et al. [3] abordan una revisión de literatura respecto a la eficiencia energética, encontrando el uso de DFVS (Escalado Dinámico de Voltaje y Frecuencia) como mecanismo para controlar la potencia/energía de los dispositivos informáticos, algunos combinan DVFS y DPM (incluido el apagado de máquinas) y un número limitado de trabajos que utilizan una limitación explícita de la potencia.

En esa misma línea, alternativas de uso de componentes electrónicos de bajo consumo de energía, y alto rendimiento como los dispositivos de *edge computer*, arrojaron resultados que muestran que es posible lograr un equilibrio favorable entre el bajo consumo de energía y el alto rendimiento al implementar algoritmos de agrupación en clústeres en dispositivos avanzados de *edge computer*, como la placa *Nvidia Jetson* Nano. Si bien se puede maximizar la eficiencia energética, esto se produce a costa de una disminución del rendimiento, lo que puede no ser ideal para aplicaciones específicas en tiempo real [4].

Como parte de las aplicaciones o usos de un clúster de computadoras, se realizaron experimentos de regresión lineal para predicción de retrasos de vuelos, utilizando un clúster de computadoras de hasta 5 unidades. El resultado muestra que, al incluir 5 computadoras en un entorno de clúster con especificaciones iguales, se puede aumentar el rendimiento de la computación hasta 35,8% y 39,76 % en comparación con uno independiente; por lo tanto, incorporar más nodos al clúster puede acelerar significativamente el proceso [5], [6].

Otro campo de aplicación de computación de alto rendimiento está en el procesamiento de imágenes y el entrenamiento de algoritmos de aprendizaje automático. El estudio de López-Martínez et al. [7], demuestra el papel crucial de la HPC en el procesamiento de imágenes, específicamente utilizando Deep Learning distribuida para clasificar diferentes tipos de malezas (objeto de estudio). Mediante el uso de HPC, se reduce significativamente los tiempos de procesamiento y el esfuerzo requerido para entrenar modelos o implementar algoritmos computacionalmente exigentes. En comparación con el uso de una sola computadora, la propuesta de los autores ofrece un beneficio considerable

en términos de eficiencia y rentabilidad, lo que la convierte en una alternativa valiosa para la comunidad universitaria y científica que puede no tener acceso a una infraestructura HPC de gama alta.

En relación con la ciberseguridad, específicamente con la criptografía, existen algunos trabajos que abordan el *password cracking* desde la perspectiva de generación de diccionarios para la adivinación de contraseñas con técnicas de aprendizaje automático, como lo realizado por Zhou et al. [8] donde, a partir de diccionarios públicos como rockyou y yandex entrenan un modelo para generar un promedio de dos mil millones de posibles contraseñas, utilizando técnicas de incrustación de palabras. En el experimento que realizaron utilizaron como hardware de procesamiento a una computadora de escritorio y un servidor con 16 GB y 64 GB de memoria RAM respectivamente, además de tarjetas gráficas Nvidia, que no son considerados como grandes recursos computacionales.

En otros estudios de la adivinación de contraseñas, se comparan algoritmos de búsquedas que utilizan algunas herramientas como *john the ripper* o *hashcat*, y en esa línea, se aplicaron estructuras de datos como los filtros *cuckoo* para optimizar los tiempos de búsqueda y el consumo de memoria [9]. Un aspecto importante que destacan las investigaciones relacionadas con el *password cracking* o adivinación de contraseñas, es el consumo de recursos de los algoritmos y de las herramientas; es por eso, que algunos investigadores como Alkhwaja et al. [10], realizaron pruebas comparativas usando programación paralela con uso de tarjetas gráficas (GPU) y procesadores (CPU), resultando en que el procesamiento paralelo combinado (GPU+CPU) es la fórmula más rápida para el *password cracking*.

II. MARCO TEÓRICO

a. Computación de Alto Rendimiento (HPC)

La computación de alto rendimiento es una tecnología que se enfoca en el desarrollo de supercomputadoras, algoritmos de procesamiento en paralelo (también denominado computación paralela) y software relacionado. Su utilidad radica en la necesidad de procesar grandes problemas complejos expresados en sistemas de ecuaciones matemáticas [11]. Es importante resaltar que la HPC permite a los investigadores el acceso a herramientas de simulación y análisis que antes eran inasequibles, con el resultado de que, en muchas áreas de la ciencia, el ritmo de la investigación se ha duplicado o triplicado [1], [12].

Sin embargo, lo recursos que requieren la computación de alto rendimiento son altos, de igual manera las exigencias de las aplicaciones que necesitan de este tipo de sistemas, por lo que, los diseñadores de supercomputadoras siempre están mejorando los niveles de rendimiento mediante el uso de tecnología de CPU y GPU de última generación. Pero el ancho de banda y la latencia en los dispostivos de red pueden tener un impacto importante en el rendimiento del sistema de computación de alto rendimiento [13].

Las supercomputadoras que forman parte de un sistema de computación de alto rendimiento se basan en procesadores RISC (*Reduced Instruction Set Computer*), que están diseñados para integrarse con facilidad en un sistema multiprocesador y que acceden a única memoria por medio del multiprocesamiento simétrico (SMP, por sus siglas en inglés). Al mismo tiempo, al formar un clúster de computadoras requieren estar interconectados por una red de alta velocidad, propietaria (patentada) o una red de area local tradicional, utilizando mecanismos de procesamiento en paralelo como PVM (*Parallel Virtual Machine*) o MPI (*Message-Passing Interface*) [14], que permita aprovechar las capacidades del HPC.

Los clústeres se utilizan en aplicaciones que tienen alta complejidad de los datos, por lo que necesitan de una velocidad operativa adecuada, y algunas formas de configuración de clústeres como el de equilibrio de carga, orientado a la eficiencia; los clústeres de alta disponibilidad, orientados a la continuidad del procesamiento y redundancia de componentes y conectividad; finalmente, los clústeres de altas capacidades para el cálculo y procesamiento de gran intensidad y problemas matemáticos complejos [15].

La computación paralela consiste en el uso de múltiples procesadores conectados para realizar cálculos complejos de una manera más rápida y eficiente. En ese sentido, un solo equipo no puede albergar todos los procesadores que se necesitan, por lo que deben ser instalados en múltiples computadoras e interconectarse (clúster) para realizar las operaciones. Se pueden considerar dos tipos de computadoras que participan para la computación paralela; el dedicado, que está destinado exclusivamente para el trabajo del clúster, y el no dedicado, que además del trabajo del clúster pueden estar destinados a tareas individuales [16].

b. Password Cracking

Las contraseñas son un elemento básico de la vida moderna y se utilizan para proteger todo tipo de información, desde el correo electrónico, las cuentas bancarias hasta las cuentas de acceso al sistema de nuestros dispositivos electrónicos, en definitiva, podemos afirmar que es el elemento guardián de nuestra información. ¿Pero, como ocurre el proceso de autenticación mediante el uso de contraseñas? Cuando un usuario introduce una contraseña, se genera un *hash* de la contraseña introducida y se compara con un hash almacenado de la contraseña real del usuario; si los *hashes* coinciden, el usuario queda autenticado [17].

Es importante precisar que un *hash* es una función matemática pseudoaleatoria unidireccional que acepta una cantidad ilimitada de bits como entrada y produce una cantidad finita de bits como salida; algunos algoritmos *hash* utilizan funciones de comprensión especialmente diseñadas para crear una salida de longitud fija independientemente del tamaño o el contenido de entrada, tales como MD5, SHA-1, SHA-256 [18].

Por lo tanto, es fácil entender por qué descifrar contraseñas se considera una grave amenaza a la seguridad, específicamente para usuarios, sistemas, dispositivos del ámbito comercial, educativo, financiero, gubernamental y militar sólo por mencionar algunos tipos. El *password cracking* o descifrado de contraseñas es el proceso que implica utilizar diversas técnicas y herramientas para obtener acceso no autorizado a recursos protegidos a partir de los tres estados de los datos; datos en proceso, datos en tránsito o datos almacenados. El proceso utilizado para descifrar las contraseñas consiste en formular una estimación de cuál es la contraseña cifrada o hash de la contraseña especulada, utilizando el mismo algoritmo que el sistema atacado y comparar sistemáticamente la estimación de la contraseña cifrada o en hash resultante con el valor

en *hash* o cifrado real que se capturó; asimismo, para descifrar contraseñas se emplean varios métodos, tales como el preprocesamiento de conjunto de contraseñas, la división de estructuras, la recombinación de cadenas de caracteres y la generación de contraseñas de forma iterativa [19].

En ese orden de ideas, los tipos de ataque de descifrado de contraseñas son; ataques de fuerza bruta, que consiste en probar todas las combinaciones posibles de caracteres hasta encontrar la contraseña correcta, este método es lento y poco eficiente para contraseñas largas y complejas, pero puede ser efectivo contra contraseñas cortas y simples, sin embargo, un ataque de fuerza bruta siempre tendrá éxito si alcanza a cubrir todo el espacio de combinaciones posibles en un tiempo determinado. Los atacantes por lo general utilizan herramientas automatizadas para adivinar las contraseñas metódicamente. Algunos ejemplos de contraseñas débiles son “123456”, “qwerty”, “password” o “111111” [20], [21]. Un ataque de diccionario consiste en utilizar listas de palabras comunes, frases o contraseñas conocidas para intentar acceder al sistema, este método es más rápido que los ataques de fuerza bruta, pero es menos efectivo si la contraseña no está en la lista [22], [23].

Una forma de *password cracking* puede ser considerado a los ataques de phishing para obtener credenciales, que consiste en engañar a los usuarios para que revelen sus contraseñas a través de correos electrónicos o sitios web maliciosos, estos ataques a menudo se basan en ingeniería social para manipular a los usuarios para que hagan clic en enlaces o abran archivos adjuntos infectados [24], [25].

c. Fortaleza de Contraseñas

La fortaleza de una contraseña se refiere a su capacidad para resistir intentos de descifrado por métodos no autorizados, la fortaleza de una contraseña se mide por la dificultad que presentan los ataques de fuerza bruta, diccionario, y otros métodos de cracking para adivinarla [26].

Los factores determinantes de la fortaleza de contraseñas son; la longitud de contraseña que es un factor crucial en su fortaleza, ya que las contraseñas más largas necesitan más combinaciones, lo que dificulta los ataques de fuerza bruta [27]. Asimismo, la complejidad de una contraseña implica el uso de una combinación de letras mayúsculas, minúsculas, números y caracteres especiales, esto también incrementa el número de combinaciones necesarias y dificulta los ataques de diccionario; un ejemplo práctico es la contraseña "Password123" es significativamente menos segura que "P@ssw0rd!23" debido a la mayor complejidad de la segunda, sin embargo, esta última también puede ser adivinada con relativa facilidad con ataques por diccionario con herramientas que aplican diversos tipos de reglas o combinaciones sobre una lista de palabras [26], [28].

En contraste, las estrategias que permiten mejorar la fortaleza de las contraseñas tales como, la educación y conciencia del usuario [13], [29] son mecanismos defensivos que deben ser permanentes. Las políticas de contraseñas robustas, que exigen longitudes mínimas, complejidad y cambios periódicos pueden fortalecer significativamente la seguridad de las contraseñas. La autenticación multi factor (MFA) añade una capa adicional de seguridad al requerir más de un factor de verificación, como algo que el usuario sabe (contraseña) y algo que el usuario tiene (token o dispositivo móvil) [30].

III. METODOLOGÍA

Para desarrollar la investigación se ejecutaron las siguientes etapas:

a. Selección de la plataforma de hardware.

Se utilizó dos computadoras de escritorio con características homogéneas que se detallan en la tabla 1.

Tabla 1: Especificaciones técnicas de nodos del clúster.

Dispositivo	Procesador	Velocidad	Núcleos	RAM	Disco
Master	Intel Core i7-9700	3.00 GHz	8	32 GB	2 TB
Nodo 1	Intel Core i7-9700	3.00 GHz	8	32 GB	2 TB

Fuente: Elaboración propia.

Del mismo modo, las 2 computadoras se conectan a un regulador automático de voltaje con una capacidad de 1200VA/600W y voltaje de 220V que contiene 8 puertos de salida. Para la conectividad, las interfaces de red cableada de cada computadora utilizadas cuentan con 1 Gbps de velocidad, además de un switch con puertos GigaEthernet.

Para el sistema operativo base del clúster, se tuvo en cuenta algunos aspectos como la licencia, la arquitectura, la versión más reciente de kernel, así como los requisitos mínimos de memoria RAM y espacio en disco. Se evaluaron dos distribuciones de Linux, Debian y Ubuntu. Las características de cada sistema operativo se visualizan en la tabla 2.

Tabla 2: Comparación de sistemas operativos.

Características	Debian	Ubuntu
Licencia	GNU GPL y otras licencias	GNU GPL y otras licencias
Sistema de archivos	Ext4 (predeterminado)	Ext4 (predeterminado)
Arquitecturas compatibles	x86, x86-64, ARM, AMD64 e Intel 64, MIPS, Power Systems	x86, x86-64, ARM, entre otros
Versión del kernel	6.1 (en Debian 12)	6.2 (en Ubuntu 23.04)
Basado en	Nativo	Basado en Debian
Memoria RAM mínima	512 MB (recomendado)	512 MB (recomendado)
Disco duro mínimo	10 GB (recomendado)	25 GB (recomendado)

Fuente: Elaboración propia.

Después del análisis de los sistemas operativos, el elegido para implementar el clúster fue Ubuntu versión 20.04 denominado "Focal Fossa", ya que es una versión estable y con soporte hasta abril de 2025, además está disponible la documentación necesaria y el soporte por parte de la comunidad de usuarios de Linux para la distribución, a diferencia de las versiones más recientes.

b. Selección de la plataforma de software

Otro componente importante es el middleware para el clúster, que es el software que actúa como intermediario entre diferentes aplicaciones, sistemas operativos o componentes de software para facilitar la comunicación y la interoperabilidad. En la tabla 3 se muestra la comparación de los aspectos más relevantes de las características de los sistemas middleware. Las alternativas examinadas son: Slurm, OpenMosix, OpenMPI, Mosix, Intel Cluster Ready y Mpich.

Tabla 3: Cuadro comparativo de sistemas middleware para el clúster.

Middleware	Descripción	Componentes	Soporte	Comentario
Slurm	• Sistema de gestión de trabajos de código abierto para clústeres HPC. • Prioridades de trabajos. • Monitorización y registro. • Libre	• Slurmctld (Controller) • Slurmd (Demonio) • Slurmdbd (Base de datos)	Comunidad activa, actualizaciones regulares	• Requiere conocimientos avanzados para la instalación y configuración • Admite administración de recursos. • Poca documentación
OpenMosix	• Extensión del kernel de Linux que crea un clúster de computadoras que parece ser una sola máquina virtual. • Balanceo de carga. • Libre.	• Kernel de Linux con extensiones OpenMosix	Proyecto descontinuado, sin soporte actual	• Desarrollo y mantenimiento discontinuado desde 2008. • No es adecuado para tareas que requieran baja latencia o alta velocidad de comunicación.
OpenMPI	• Implementación de código abierto de la interfaz de paso de mensajes (MPI) estándar. • Alto rendimiento en sistemas de memoria distribuida. • Flexibilidad y portabilidad en diversas arquitecturas. • Libre	• Biblioteca MPI, utilidades de línea de comandos	Comunidad activa, soporte continuo	• Facilita la comunicación y coordinación entre procesos en clústeres y sistemas distribuidos para aplicaciones paralelas. • Capacidad de agregar y quitar nodos según las necesidades de carga de trabajo.
Mosix	• Tecnología de código cerrado similar a OpenMosix, que permite un clúster de computadoras con Linux que se comporta como una sola máquina virtual. • Equilibrio de carga • Detección automática de recursos. • Comercial	• Kernel de Linux con extensiones Mosix	Proyecto descontinuado, sin soporte actual	• Limitado soporte para sistemas operativos modernos y nuevas arquitecturas. • No adecuado para tareas con requisitos de baja latencia. • Enfoque en la migración automática de procesos entre nodos para aprovechar la capacidad de procesamiento disponible.
Intel Cluster Ready	• Suite de herramientas de Intel para implementar y administrar clústeres. • Interoperabilidad y estabilidad. • Comercial.	• Intel Cluster Checker • Intel Cluster Tools • Intel Cluster Studio	Soporte de Intel	• Optimizado para hardware de Intel y ofrece herramientas para el desarrollo de aplicaciones paralelas. • Compatible con hardware Intel. • Puede requerir la compra de hardware adicional o especifico de Intel.
MPICH	• Implementación de código abierto de la interfaz de paso de mensajes (MPI) para clústeres y sistemas distribuidos. • Estabilidad y confiabilidad. • Libre.	• Biblioteca MPI, utilidades de línea de comandos	Comunidad activa, soporte continuo	• Enfoque en rendimiento, escalabilidad y portabilidad para aplicaciones MPI. • Probables problemas en la escalabilidad en clústeres muy grandes o complejas.

Fuente: Elaboración propia.

Luego del análisis de las características de los sistemas middleware, se llegó a la conclusión de que OpenMPI es la opción más adecuada para la implementación del clúster de alto rendimiento por las siguientes razones:

- Capacidad de agregar y quitar nodos: OpenMPI también permite agregar y quitar nodos en cualquier momento, facilitando la expansión o reducción del clúster según las necesidades de carga de trabajo.

- Soporte constante: OpenMPI cuenta con una comunidad activa para el desarrollo y soporte, lo que garantiza actualizaciones continuas.

- Paralelización de aplicaciones: OpenMPI es ampliamente utilizado en aplicaciones paralelas y distribuidas ya que permite que las aplicaciones ejecuten múltiples procesos de manera simultánea y así aprovechar los recursos del clúster de alto rendimiento.

OpenMPI es una biblioteca que facilita el desarrollo y ejecución de aplicaciones paralelas en sistemas de alto rendimiento como clústeres, lo cual permite la comunicación y coordinación entre procesos ejecutándose en paralelo en varios nodos, para que se pueda resolver problemas complejos que requieran potencia de cómputo.

En la tabla 4 se presentan los distintos tipos de arquitectura de clúster y sus características más relevantes. Esta información provee del conocimiento necesario para tomar una decisión acertada al seleccionar la arquitectura adecuada para la implementación del proyecto.

Tabla 4: Comparación de tipos de arquitectura.

Arquitectura	Nodos	Red de comunicación	Herramienta de programación	Sistema operativo
Solaris	Nodos basados en solaris y máquinas virtuales	Red de alto rendimiento	Bibliotecas de programación de alto rendimiento, entorno MPI	Solaris(Sistema operativo Oracle)
Beowulf	Nodos de uso común	Red de bajo costo(Ethernet)	Código abierto, MPI(OpenMPI,MPICH)	Linux
Hadoop	Nodos de servidores	Data Center Network, Red Local (Ethernet)	Apache Hadoop, MapReduce, HDFS, Java, Python	Linux
Kubernets	Nodos de servidores	Red de datos(Ethernet)	Docker, Python, Java, etc.	Linux

Fuente: Elaboración propia.

Tras evaluar las particularidades de diversas arquitecturas clúster, se eligió la arquitectura tipo Beowulf, debido a su economía al utilizar el hardware de uso común, sistemas operativos de código abierto y redes de alta velocidad. Además, ofrece una variedad de herramientas de programación paralela para optimizar el rendimiento del clúster en tareas complejas; su flexibilidad, escalabilidad, y eficiencia la hacen una opción óptima para alcanzar los objetivos del proyecto.

En la figura 1 se visualiza la arquitectura clúster de alto rendimiento para su adecuado funcionamiento, este diseño es la arquitectura estándar utilizadas en los HPC, donde el nodo principal, conocido como máster se enlaza con la red pública y desempeña el papel de controlador central, emitiendo comandos a los nodos esclavos que a su vez están conectados con una interfaz de red a un switch. Esta interconexión entre los nodos y el nodo principal hace uso de la capa de aplicación para la comunicación y el procesamiento de datos.

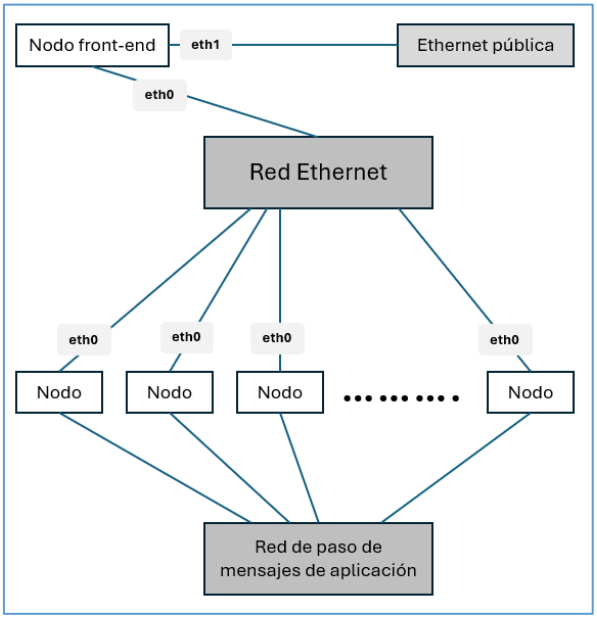

Figura 1: Arquitectura de clúster HPC.
Fuente: Rocks Multi-computer Cluster, por Bairpetro, (2015).

c. Implementación del clúster

La topología utilizada se muestra en la figura 2, donde se puede apreciar el diseño del clúster con el nodo máster y el nodo esclavo, formando una red de tipo estrella, que a su vez tienen acceso a Internet. Se configuró direccionamiento IP estático.

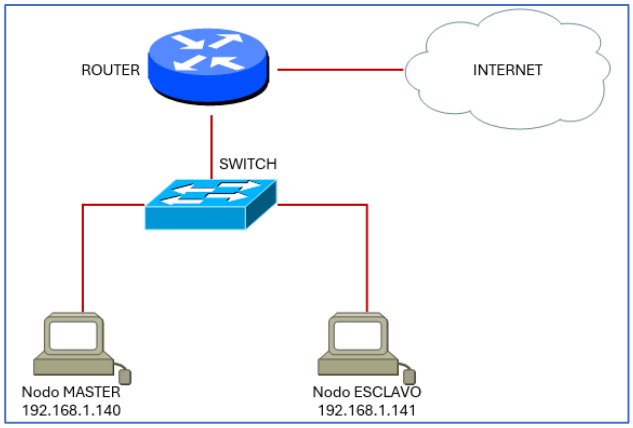

Figura 2: Diagrama de topología del clúster.
Fuente: Elaboración propia.

Después de la instalación del sistema operativo Ubuntu en ambos equipos, se debe asegurar la instalación de paquetes adicionales como net-tools; openmpi (https://www.open-mpi.org/software/ompi/v4.1/), openmpi-bin, libopenmpi-dev y libssl-dev. Configurar y habilitar el servicio SSH.

Para facilitar el intercambio de archivos entre los nodos del clúster es esencial habilitar el sistema de archivos NFS con la instalación de paquetes adicionales como nfs-kernel-server (master y esclavo) y nfs-common portmap (solo esclavo), para luego habilitar un directorio compartido verificando su accesibilidad desde el nodo esclavo con el comando "showmount -e" y "mount -t nfs IP_address: ruta_absoluta_directorio_compartido". También se debe realizar los ajustes en el archivo "hosts" para la identificación de los nombres específicos de los dispositivos miembros del clúster.

Es importante que el montaje del directorio compartido se configure para ejecutarse automáticamente cada vez que se inicie el sistema operativo del nodo esclavo; para ello se puede editar el archivo /etc/fstab y agregar, por ejemplo, la siguiente línea: "192.168.1.140: home/cluster/DirCompartido//home/cluster/DirCompartido nfs rw,sync,hard,intr 0 0".

En el nodo master se debe registrar la cantidad de procesadores disponibles por nodo, esto se realiza editando el archivo ".mpi_hostfile" tal como se muestra en la figura 3.

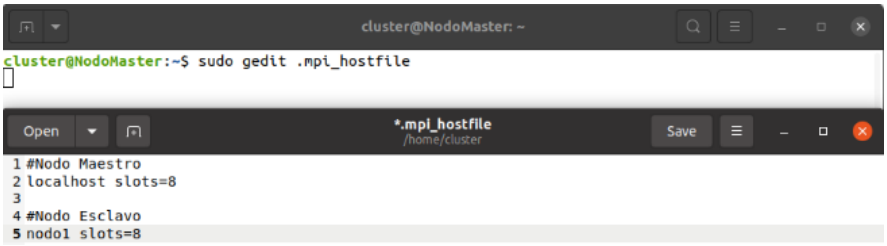

Figura 3: Creación y configuración del archivo mpi_hostfile.
Fuente: Elaboración propia.

d. Pruebas del clúster

Para las pruebas del clúster y aplicación de este para el proceso de password cracking se diseñaron dos escenarios generales:

- Primer escenario: Prueba de rendimiento con hashes sha-1 y md5. Para este escenario se calcularon los hashes básicos de algunas contraseñas de diversa complejidad y fortaleza asumiendo que existen aplicaciones que aún utilizan este tipo de algoritmos.

- Segundo escenario: Prueba de rendimiento con hashes de contraseñas de sistemas Windows y Linux. Se optó por usar los hashes de contraseñas de los usuarios locales de un sistema Windows que están almacenados en formato NT, y para Linux, se utilizó la distribución Ubuntu 20.04 que almacena los hashes de contraseña en formato de SHA-512 con SALT.

El diccionario utilizado para las pruebas se basó en el archivo de palabras por defecto password.lst de la herramienta John the Ripper con aproximadamente diez mil palabras. El tiempo de ejecución por proceso se limitó a 72 horas por disponibilidad de equipamiento.

IV. RESULTADOS Y DISCUSIÓN

Para la obtención de los resultados esperados, lo primordial fue evidenciar el correcto funcionamiento de los nodos configurados en el clúster; para este propósito, en el nodo master se ejecutó un script que efectúa el cálculo de la suma de números primos a partir de un programa compilado escrito en el lenguaje C, y también se realizó los mismo cálculos con el nodo máster y el nodo esclavo en conjunto, es decir funcionando como clúster, teniendo como resultado que el tiempo necesario para culminar el procesamiento es de 30 segundos con 8 procesadores solo con el nodo máster y de 16 segundos operando como clúster (nodo master y esclavo) con 16 procesadores en conjunto. Por lo que se evidencia en esta primera prueba que existe una eficiencia significativa al usar ambos nodos, esto se contrasta con los trabajos realizados con García-Ojeda et al. [2] en el que manifiesta que el uso de un clúster de HPC mejora al menos un 25% respecto a la ejecución de tareas en nodos unitarios; y en este caso la mejora en la eficiencia respecto al tiempo fue del 46%.

Para la ejecución del procesamiento de password cracking, en principio fue necesario contar con una instalación y configuración adecuada de OpenMPI y JTR, en ese sentido, para esta investigación se utilizó la versión john-1.9.0-jumbo-1. Además, se realizó la transformación del código fuente de JTR en un ejecutable funcional, para esto se utilizó el comando "./configure && make". Asimismo, se procedió a activar OpenMPI utilizando el comando "./configure --enable-mpi", este paso es crucial para habilitar la compatibilidad y el uso eficiente de OpenMPI dentro del entorno de JTR. Para finalizar el proceso de configuración, se ejecuta el comando "make -s clean && make -sj4" y posteriormente el comando "make -sj4" que inicia la compilación utilizando cuatro hilos de proceso simultáneos lo que acelera significativamente el proceso de construcción del programa.

Una vez que la compilación ha concluido se procede a llevar a cabo la prueba de testeo de JTR ejecutando el comando "./john -test", esta acción es crucial para verificar la funcionalidad y el rendimiento adecuado de JTR después de su compilación debido a que se somete el programa a una serie de pruebas específicas para identificar posibles problemas y asegurarse de que todas las funciones operen según lo esperado para garantizar resultados precisos y confiables en el proceso de password cracking offline.

Se ejecutaron pruebas en el clúster en dos escenarios, el primer escenario engloba una prueba de rendimiento con hashes md5 y sha-1 utilizando contraseñas de 7 y 9 caracteres, el segundo escenario contempla una prueba de rendimiento con una lista de usuarios con sus hashes extraídos de sistema operativo Windows a partir de los archivos del sistema "system" y "sam", y de un sistema operativo Ubuntu 20.04 a partir

de los hashes almacenados en el archivo "shadow". Estas pruebas tuvieron como propósito validar la mejora al emplear dos o más nodos trabajando de manera paralela en comparación con el uso de un solo nodo en un clúster HPC, bajo la topología de red diseñada.

El tiempo de procesamiento definido arbitrariamente para cada prueba, fue de 72 horas como máximo, considerando la disponibilidad de los recursos del equipamiento.

a. Primer escenario: Prueba de rendimiento con hashes sha-1 y md5

Se utilizaron los hashes SHA-1 de las contraseñas "guilmar" y "rousandia" para ejecutar la prueba de cracking con los nodos del clúster HPC, y tal como se muestra en la tabla 5, los resultados del tiempo de ejecución cuando se utilizan 8 y 16 procesadores tienen una diferencia significativa.

Cuando se ejecuta el password cracking con un solo nodo se puede usar el siguiente comando: “time mpirun -np 8 –hostfile ../.mpi_hostfile ../john-1.9.0-jumbo-1/run/john –format=raw-sha1 hash.txt”; y cuando se ejecuta con ambos nodos se puede ejecutar el siguiente comando: “time mpirun -np 16 --hostfile ../.mpi_hostfile ../john-1.9.0-jumbo-1/run/john --format=raw-sha1 hash.txt”. El diccionario utilizado es la lista por defecto de la herramienta JTR.

Tabla 5: Resultado del password cracking utilizando el hash sha-1.

Numero de nodos clúster	Numero de procesadores	Tiempo de ejecución(minutos)
1	8	0:05:18
2	16	0:02:00

Fuente: Elaboración propia.

En la tabla 5 se observa el tiempo de ejecución utilizado para realizar el password cracking. Se puede evidenciar que, utilizando solo un nodo del clúster, el tiempo ejecutado es de 5 minutos con 18 segundos, mientras que al utilizar 2 nodos el tiempo se reduce a 2 minutos, obteniendo una disminución de 3 minutos con 18 segundos, lo que se interpreta como una eficiencia aproximada de 64%. Estos resultados se contrastan con lo que afirma López-Martínez et al. [7], ya que al utilizar el HPC se reduce significativamente los tiempos de esfuerzo y procesamiento requerido en actividades gestionadas por el usuario donde se utilizan recursos computacionales.

También se utilizó la contraseña “ust3du5” que fue generada de manera aleatoria, con la cual se calculó el hash MD5 y SHA-1. Se precisa que la regla utilizada fue por defecto y el diccionario contenía aproximadamente 10 mil palabras, y en contraste con la perspectiva planteada por Zhou et al. [8] donde se utilizan diccionarios públicos para generar posibles contraseñas. Los resultados se muestran en la tabla 6 y 7.

Tabla 6: Resultado del password cracking utilizando el hash md5 de la contraseña “ust3du5”.

Numero de nodos clúster	Numero de procesadores	Tiempo de ejecución(minutos)
1	8	0:17:52
2	16	0:06:29

Fuente: Elaboración propia.

El tiempo de ejecución al realizar password cracking, utilizando un nodo de ocho procesadores es de 17 minutos con 52 segundos, mientras que, al utilizar los dos nodos con 16 procesadores en total, el tiempo se reduce a 6 minutos con 29 segundos, obteniendo una disminución de 11 minutos con 23 segundos, lo que se interpreta como una eficiencia aproximada también de 64%.

Tabla 7: Resultado del password cracking utilizando el hash sha-1 de la contraseña “ust3du5”.

Numero de nodos clúster	Numero de procesadores	Tiempo de ejecución(minutos)
1	8	0:32:09
2	16	0:11:35

Fuente: Elaboración propia.

De acuerdo con la tabla 7, el tiempo de ejecución para el hash SHA-1 para obtener el resultado es de 32 minutos con 9 segundos con 8 procesadores y 11 minutos con 35 segundos con 16 procesadores, obteniendo una diferencia de 20 minutos con 34 segundos, lo que es equivalente al 64% menos del tiempo consumido con un solo nodo.

Estas observaciones confirman también que, el tiempo requerido para descifrar las contraseñas, está directamente relacionado con el algoritmo de hash utilizado. También es importante destacar que la diferencia en el proceso de descifrado se mantiene constante en 64% utilizando ambos nodos del clúster, por lo que se entiende que, a mayor cantidad de nodos se obtiene una diferencia mayor en el tiempo de procesamiento.

Esta medición refleja la influencia positiva de la estrategia de distribución de carga en el clúster, aprovechando de manera eficiente la potencia de procesamiento combinada de ambos nodos. La reducción significativa en el tiempo de descifrado resalta la efectividad del enfoque paralelo, subrayando la capacidad del clúster para agilizar procesos de descifrado y demostrando su ventaja en términos de velocidad y rendimiento comparado con soluciones de procesamiento convencionales.

b. Segundo escenario: Prueba de rendimiento con hashes de contraseñas de sistemas Windows y Linux

En este escenario, se llevó a cabo dos tipos de pruebas. La primera fue realizada con los hashes NT de contraseñas de usuarios de sistemas Windows, y en la segunda se utilizó hashes de contraseñas del sistema operativo Linux Ubuntu. Las pruebas se realizaron utilizando el diccionario de la herramienta JTR password.lst con aproximadamente 10,000 palabras.

Prueba de rendimiento con hashes de contraseña de usuarios de Windows

Con la ejecución del comando "time mpirun -np 16 --hostfile ../.mpi_hostfile ../john-1.9.0-jumbo-1/run/john --format=NT win-hash", se obtiene como resultado el descifrado del hash de contraseña "admin123" en tan solo 3 segundos. Sin embargo, el proceso de cracking para hashes de contraseñas más fuertes y complejas como "c4rl0sUSER", "M@rl42023" o"Ju4ns1t022" fue necesario 40 minutos, 7 horas y 26 horas respectivamente; con el uso de los 16 procesadores del clúster; mientras que con 8 procesadores fue necesario 102 minutos, 13 horas y 53 horas respectivamente. Estos resultados indican una diferencia en el tiempo de procesamiento de 52% en promedio.

Los hashes de contraseñas de usuario local de un sistema Windows utilizados en las pruebas fueron de la versión Windows 7 con formato NT, similar al siguiente: "68bbbb18b3c27d941cb6224353be1d0a".

Es evidente determinar que la agregación de nodos y distribución de la carga entre más nodos aceleraría significativamente el proceso y permitiría abordar contraseñas más complejas, tal como se contrasta con los estudios de Alkhwaja et al. [10]. Esta situación resalta la importancia de la escalabilidad y la capacidad de procesamiento en tareas de password cracking.

Prueba de rendimiento con hashes de contraseñas de usuario de Linux Ubuntu.

Para el procesamiento de hashes de contraseñas de usuarios locales de la distribución Ubuntu versión 20.04 del sistema operativo Linux, se utilizó el comando "time mpirun -np 16 --hostfile ../.mpi_hostfile ../john-1.9.0-jumbo-1/run/john --format=sha512crypt shadowUbuntu20.txt". Los hashes utilizados para las pruebas corresponden a las contraseñas "admin123", "c4rl0sUSER", "M@rl42023" y"Ju4ns1t022". Los resultados obtenidos que utilizaron el clúster con ambos nodos fueron de 4 segundos para el primero, y 70 minutos para el segundo, y 16 horas para el tercero. El hash de la contraseña "Ju4ns1t022" no fue descubierta en el tiempo determinado y con el diccionario definido previamente.

Los hashes de contraseñas de usuario local de la distribución Ubuntu 20.04 del sistema Linux utilizan el algoritmo SHA-512 con SALT, lo que hace que los hashes sean mas robustos y se requiera mayor tiempo de procesamiento. El formato de un hash para este sistema operativo es: "6v84umER7vqAFBk48$h.b0TbiBVf7DeB3M5qJTxJsZpnhA7To2tPFg6Kdd32248nTtGHof7WFTvK.YvkoRwtQIldWLLkA8da28yN1Hc1:19573:0:99999".

La razón por la que no se logró encontrar la contraseña restante en el proceso de descifrado de hashes en Ubuntu 20.04 se debe a la complejidad de las contraseñas, la fortaleza del algoritmo que en este caso utiliza un componente adicional como el valor de SALT y la limitación en la cantidad de nodos disponibles en el clúster. En esta situación, la escalabilidad y la optimización de recursos se vuelven factores críticos para abordar con éxito el desafío de descifrar contraseñas más complejas en un tiempo razonable.

V. CONCLUSIONES

Se llevó a cabo una exhaustiva exploración en la implementación y evaluación de un clúster de alto rendimiento utilizando la herramienta de código abierto OpenMPI para pruebas de fortaleza de contraseñas con técnicas de password cracking offñine. Esta experiencia proporciona una visión integral del potencial y la eficiencia de los clústeres en el ámbito del descifrado de contraseñas. Se pudo constatar que la configuración de un clúster basado en OpenMPI es factible y brinda ventajas sustanciales en términos de capacidad y eficiencia de procesamiento.

La distribución de tareas y la ejecución paralela permiten aprovechar al máximo los recursos computacionales disponibles, mejorando la eficiencia y la velocidad en la resolución de problemas que demandan un alto poder de procesamiento. La interconexión de los nodos sobre una topología de red Ethernet y herramientas como SSH y NFS facilita la comunicación y compartición de recursos, creando un ambiente de trabajo flexible.

El uso de un clúster en el estudio ha permitido distribuir la carga de trabajo de manera eficiente entre dos nodos, lo que ha llevado a una notable reducción en el tiempo necesario para realizar tareas de procesamiento intensivo. La colaboración entre el nodo máster y nodo esclavo, junto con la adecuada configuración de OpenMPI, ha demostrado su eficiencia en la ejecución paralela de procesos, optimizando los recursos disponibles y mejorando el tiempo de procesamiento del password cracking con una diferencia promedio de 52% a 64 % respecto al procesamiento con un solo nodo.

Los resultados de las pruebas realizadas evidencian el impacto directo del paralelismo en el tiempo necesario para la ejecución de tareas como el password cracking utilizando herramientas John the Ripper y OpenMPI. La distribución de la carga de trabajo entre múltiples procesadores o núcleos resulta en una drástica reducción en el tiempo requerido para descifrar contraseñas. Esto resalta la importancia de la computación en clúster y su aplicabilidad en tareas exigentes de potencia cómputo.

VI.RECOMENDACIONES

Respecto a la planificación y diseño; antes de iniciar la implementación de un clúster de computadoras, es crucial realizar una planificación detallada que abarque la infraestructura, los recursos necesarios, la distribución de tareas y las herramientas adecuadas para la tarea específica que se desea realizar. Esto permitirá evitar problemas futuros y garantizar una implementación exitosa.

Sobre la optimización de recursos, es esencial ajustar los recursos, como el número de nodos y procesadores, de acuerdo con la carga de trabajo y los requerimientos del sistema. Un exceso de recursos puede resultar en subutilización, mientras que una escasez puede limitar el rendimiento.

VII. REFERENCIAS

[1] IBM, "¿Qué es la computación de alto rendimiento (HPC)?" Accessed: Jul. 13, 2024. [Online]. Available: https://www.ibm.com/es-es/topics/hpc.

[2] J. C. García-Ojeda, M. L. Ortíz, R. S. García, J. H. Cáceres, and A. Argoti, "A Computer Cluster for Big Data and Data Analytics Management: Design, Implementation, and Assessment," in Proceedings of the Euro American Conference on Telematics and Information Systems, in EATIS '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3293614.3293626.

[3] B. Kocot, P. Czarnul, and J. Proficz, "Energy-Aware Scheduling for High-Performance Computing Systems: A Survey," Energies (Basel), vol. 16, no. 2, 2023, doi: 10.3390/en16020890.

[4] M. Lapegna, V. Mele, and D. Romano, "Clustering Algorithms for Enhanced Trustworthiness on High-Performance Edge-Computing Devices," Electronics (Basel), vol. 12, no. 7, 2023, doi: 10.3390/electronics12071689.

[5] C. Paramita, F. A. Rafrastara, U. Sudibyo, and R. Wibowo, "Performance Evaluation of Linear Regression Algorithm in Cluster Environment," Nov. 2020. doi: 10.13140/RG.2.2.20357.91360.

[6] C. Paramita, S. Catur, S. Luqman Afi, and R. Fauzi Adi, "The Use of Cluster Computing and Random Forest Algoritm for Flight Delay Prediction," Mar. 2022, Zenodo. doi: 10.5281/zenodo.6377016.

[7] M. López-Martínez et al., "A High-Performance Computing Cluster for Distributed Deep Learning: A Practical Case of Weed Classification Using Convolutional Neural Network Models," Applied Sciences, vol. 13, no. 10, 2023, doi: 10.3390/app13106007.

[8] E. Zhou, Y. Peng, G. Shao, F. Deng, Y. Miao, and W. Fan, "Password cracking using chunk similarity," Future Generation Computer Systems, vol. 150, pp. 380–394, 2024, doi: https://doi.org/10.1016/j.future.2023.09.013.

[9] M.-D. Cano, A. Villafranca, and I. Tasic, "Performance evaluation of Cuckoo filters as an enhancement tool for password cracking," Cybersecurity, vol. 6, no. 1, p. 57, 2023, doi: 10.1186/s42400-023-00193-6.

[10] I. Alkhwaja et al., "Password Cracking with Brute Force Algorithm and Dictionary Attack Using Parallel Programming," Applied Sciences, vol. 13, no. 10, 2023, doi: 10.3390/app13105979.

[11] S. Jamshed, "Chapter 2 - Introduction to High-Performance Computing," in Using HPC for Computational Fluid Dynamics, S. Jamshed, Ed., Oxford: Academic Press, 2015, pp. 21–40. doi: https://doi.org/10.1016/B978-0-12-801567-4.00002-7.

[12] J. O'Reilly, "Chapter 9 - High-Performance Computing," in Network Storage, J. O'Reilly, Ed., Boston: Morgan Kaufmann, 2017, pp. 151–161. doi: https://doi.org/10.1016/B978-0-12-803863-5.00009-1.

[13] G. Lee, "Chapter 10 - High-Performance Computing Networks," in Cloud Networking, G. Lee, Ed., Boston: Morgan Kaufmann, 2014, pp. 179–189. doi: https://doi.org/10.1016/B978-0-12-800728-0.00010-2.

[14] C. Severance, K. Dowd, and O. T. Library, High Performance Computing. in Online access: Center for Open Education Open Textbook Library. OpenStax CNX, 2010. [Online]. Available: https://books.google.com.pe/books?id=9MaQzQEACAAJ.

[15] R. Raj and B. Pavithra, "Cluster Computing," Indian Scientific Journal Of Research In Engineering And Management, vol. 6, no. 6, Jun. 2022, doi: http://dx.doi.org/10.55041/ijsrem14249.

[16] D. Jiménez and A. Medina, "Cluster de Alto Rendimiento," Journal Innovación y Tecnología, pp. 16–27, 2014, Accessed: Jul. 13, 2024. [Online]. Available: http://revistasbolivianas.umsa.bo/scielo.php?script=sci_arttext&pid=S1234-12342014000100004&lng=pt&nrm=iso.

[17] Kaspersky, "Cómo almacenar correctamente tus contraseñas de usuario | Blog oficial de Kaspersky." Accessed: Jul. 18, 2024. [Online]. Available: https://www.kaspersky.es/blog/how-to-store-passwords/29183/.

[18] S. K. Jena, R. C. Barik, and R. Priyadarshini, "A systematic state-of-art review on digital identity challenges with solutions using conjugation of IOT and blockchain in healthcare," Internet of Things, vol. 25, p. 101111, Apr. 2024, doi: 10.1016/J.IOT.2024.101111.

[19] C. Wright, "Information Gathering," The IT Regulatory and Standards Compliance Handbook, pp. 73–114, Jan. 2008, doi: 10.1016/B978-1-59749-266-9.00005-9.

[20] Daniel W. Dieterle, Password Cracking with Kali Linux | Security | eBook, 1st Edition. 2024, 2024. Accessed: Jul. 18, 2024. [Online]. Available: https://www.packtpub.com/en-ar/product/password-cracking-with-kali-linux-9781835888544.

[21] BBC, "Millions using 123456 as password, security study finds." Accessed: Jul. 18, 2024. [Online]. Available: https://www.bbc.com/news/technology-47974583.

[22] Kaspersky, "¿Qué es un ataque de diccionario?" Accessed: Jul. 18, 2024. [Online]. Available: https://latam.kaspersky.com/resource-center/definitions/what-is-a-dictionary-attack.

[23] V. Nair and D. Song, "Multi-Factor Credential Hashing for Asymmetric Brute-Force Attack Resistance," Proceedings - 8th IEEE European Symposium on Security and Privacy, Euro S and P 2023, pp. 56–72, Jun. 2023, doi: 10.1109/EuroSP57164.2023.00013.

[24] N. Tihanyi, T. Bisztray, B. Borsos, and S. Raveau, "Privacy-Preserving Password Cracking: How a Third Party Can Crack Our Password Hash Without Learning the Hash Value or the Cleartext," IEEE Transactions on Information Forensics and Security, vol. 19, pp. 2981–2996, Jun. 2023, doi: 10.1109/TIFS.2024.3356162.

[25] Cybersecurity & Infraestructure Security Agency, "Avoiding Social Engineering and Phishing Attacks | CISA." Accessed: Jul. 18, 2024. [Online]. Available: https://www.cisa.gov/news-events/news/avoiding-social-engineering-and-phishing-attacks.

[26] J. Bonneau and S. Preibusch, "The password thicket: technical and market failures in human authentication on the web," 2010.

[27] P. G. Kelley et al., "Guess again (and again and again): Measuring password strength by simulating password-cracking algorithms," Proc IEEE Symp Secur Priv, pp. 523–537, 2012, doi: 10.1109/SP.2012.38.

[28] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The Tangled Web of Password Reuse," 2014, Accessed: Jul. 18, 2024. [Online]. Available: http://dx.doi.org/doi-info-to-be-provided-later.

[29] B. Ur et al., How Does Your Password Measure Up? The Effect of Strength Meters on Password Creation. 2012. Accessed: Jul. 18, 2024. [Online]. Available: https://www.cs.bham.ac.uk/~garciaf/publications/Gone_in_360_seconds_Hijacking_with_Hitag2_poster_2012.pdf.

[30] B. Ur et al., Measuring real-world accuracies and biases in modeling password guessability. 2015. Accessed: Jul. 18, 2024. [Online]. Available: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-ur.pdf.