INSTITUCIÓN UNIVERSITARIA POLITÉCNICO GRANCOLOMBIANO

FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS

MAESTRÍA EN INGENIERÍA DE SISTEMAS

GRUPO DE INVESTIGACIÓN FICB-PG

LÍNEA DE PROFUNDIZACIÓN: COMPUTACIÓN DE ALTO DESEMPEÑO

Algoritmo para detección de sobrecargas y posibles fallos cluster HPC local

PRESENTA:

John Edison Ramírez Pescador

1520020099

ASESOR TEMÁTICO:

Alexis Rojas Cordero

Junio 2017

ÍNDICE GENERAL

Contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

22. Gráfica megabytes leídos por segundo vs tiempo medio en atender solicitud.
23. Gráfica megabytes escritos por segundo vs tiempo medio en atender solicitud.
24. Técnica del codo para selección de cantidad de clúster.
25. Cantidad de clusters k means.
26. Cantidad de clusters error cuadrático.
27. Cantidad de clusters vecino más cercano.
28. Datos por atributos k means.
29. Datos por atributos error cuadrático.
30. Datos por atributos vecino más cercano.
31. Gráfica paralela, comportamiento de atributos k means.
32. Gráfica paralela, comportamiento de atributos error cuadrático.
33. Gráfica paralela, comportamiento de atributos vecino más cercano.

RESUMEN

Los sistemas High Performance Computing (HPC), ha incrementado el número de unidades de procesamiento (núcleos) significativamente [22]. Actualmente, estos sistemas poseen una gran potencia de cómputo, todo parece indicar que esta tendencia continuará aumentando durante los próximos años, debido a las constantes mejoras tecnológicas, lo que hace posible el incremento tanto del número de núcleos por procesador, como el número de procesadores en estos sistemas. Los usuarios de estos sistemas, desean obtener el máximo partido a toda esta potencia, a la hora de ejecutar sus aplicaciones, ya sea disminuyendo el tiempo de ejecución de sus aplicaciones, o ejecutando sus aplicaciones con una carga mayor de trabajo [31]. Ejecutar programación paralela, en computación de alto desempeño es una tarea sencilla. el número de procesos que se realiza entre las tareas de una aplicación y la cantidad de procesadores, nos muestra que existe un límite en su escalabilidad. Aquí es donde se identifican sobrecargas en el sistema, conllevando a una in eficiencia del mismo. Esto se muestra en las aplicaciones que ejecuten una elevada cantidad de recursos, en largos períodos de tiempo, estas aplicaciones sufren de sobrecargas y posibles fallos, donde se abre un gran campo de investigación en toda la comunidad de ingenieros TI dedicados a la computación de alto desempeño [21].

PALABRAS CLAVE

HPC, Cluster, Lustre, Sistema de Archivos, entradas y salidas de datos, Darshan, software R, clustering.

INTRODUCCIÓN

En el centro de cómputo HPC del politécnico Gran colombiano se realizaron varios experimentos, en esta tesis se observaremos un enfoque, en la caracterización de sobrecargas o posibles fallas a nivel de entradas y salidas de información, esta caracterización fue con el software Darshan en un cluster con sistema operativo ROCKS y sistema de archivos LUSTRE. La composición del sistema de entradas y salidas es complicado, los sistemas de archivos paralelos disponibles, contienen una brecha de rendimiento entre la tasa de entradas y salidas entregado, y el ancho de banda de aplicación alcanzado, lecturas del sistema como las jerarquías de almacenamiento, y la distribución de archivos, nos muestra que el aprovechamiento no es el máximo [16].

Entre los desafíos de los sistemas actuales, se encuentra el acceso a sistemas de almacenamiento, donde las entradas y salidas deben soportar un movimiento de

datos eficiente y escalable entre discos y memorias [6]. Se propone una metodología para realizar la instalación y la configuración de un sistema de archivos paralelo. Se analizarán algunas de sus limitaciones, y su impacto en el rendimiento de entradas y salida, mientras se garantizan datos de acceso simultáneos para este sistema de archivos distribuidos, esta tecnología lleva el paralelismo a grandes escalas [5]. Este análisis ayudará a una mejor comprensión de los requisitos de entradas y salidas de las aplicaciones que se utilizan en este tipo de computación, donde se utilizaron aplicaciones útiles, capaces de modelar los patrones de acceso de estas entradas y salidas, según la carga de trabajo que soporte nuestro sistema, este será el punto de referencia, no sólo para modelar el comportamiento del sistema, sino también para establecer las métricas obtenidas, según el rendimiento entregado de las aplicaciones.

Para explotar eficientemente el creciente número de núcleos y hacer pleno uso de nuestros sistemas. Se debe tener gran cuidado al acceder a los datos de entradas y salidas durante la ejecución de las aplicaciones, porque no es óptimo este acceso, ya que puede afectar negativamente al rendimiento, debido a que ejecutamos varios procesos simultáneamente, y en la gran complejidad que se presenta entre el almacenamiento en caché, la carga de datos que se realiza en el proceso dificulta la caracterización [1].

A pesar de que muchas aplicaciones ya se han adaptado a los distintos sistemas de archivo similares a LUSTRE, el software de sistema Application Programming Interface (API) está todavía en la etapa temprana, se está trabajando en muchas direcciones, con el mínimo de errores entre los millones de archivos de aplicaciones científicas, ya sea creados por proceso o por iteración, en esta etapa la caracterización que nos muestra Darshan es como nos afecta la gran cantidad de operaciones de meta-datos dentro del sistema de archivos paralelo [33], se utilizó nuestro sistema de archivos Lustre así comparamos el desempeño y el análisis de las operaciones en Lustre, esta comparación es con discos de archivos locales [23].

La velocidad de entradas y salidas en el sistema de archivos LUSTRE, mantiene un ritmo alto según la demanda de computación de alto rendimiento, las interconexiones de red son muy importantes debido a la baja latencia, que se presenta debido a que nuestros servidores son de una red Ethernet de 100M, en clusters HPC el acceso a redes de alta velocidad de las interconexiones, se puede comprender como esto afecta la velocidad de almacenamiento en cada sistema, en su sistema de archivos [7]. En esta computación a gran escala se consiguen aplicaciones con terabytes, petabytes y más allá, la utilizada en el proyecto, fue Gigabyte, debido a la capacidad de almacenamiento de el HPC, igual al manejar

esta cantidad de datos nos genera un cuello de botella bajando el rendimiento de la aplicación, lo cual se convierte en un desafío a la hora de realizar pruebas, que funcionen que se ejecute con el sistema de archivos, que pueda Proporcionan entradas y salidas, escalables, y de alto rendimiento, ejecutándose en el entorno del sistema de archivos Lustre, nos brinda un enfoque de resultados empíricos que apoyan, la exploración de alternativas en este tipo de computación [17].

Las tasas de acceso a datos de almacenamiento, nos muestra que se debe tener una gran cantidad de nodos, para obtener la cantidad de procesadores necesarios, así proporcionar entradas y salidas correspondientes a las necesidades de las aplicaciones, según las limitaciones claramente visibles por el hardware, lo cual nos obliga a impulsar la máxima eficiencia en cada proceso [15]. Para evitar el desperdicio de ciclos, causados por inconvenientes de red, escritura en disco, o lectura de disco, en el clúster, establecemos el estado del proceso, con una herramienta de categorización, para ser guardado en el disco en la carpeta de logs definida en Darshan, así guardar un evento del sistema, antes de que el proceso complete su ejecución, con lo cual se tendrá el proceso desde los datos que se guardados en el disco [13].

Presentación del problema

Las sobrecargas y posibles fallos en el almacenamiento de datos, provocado por la cantidad de datos generados por aplicaciones de alto procesamiento, por ejemplo, para medios sociales, distribución de vídeo, computación en red entre otros, presenta bastantes retos para los ingenieros, TI donde nos surgen problemas, que son: cómo identificar un inconveniente en un volumen tan amplio de datos? o cómo identificar una complicación en el rendimiento de almacenamiento de datos? [9]. Todo este aumento en la cantidad de datos se acompaña de un tiempo de acceso más largo. El diseño de los sistemas de almacenamiento de datos, y las sobrecargas que se presentan al momento de almacenar los datos, es esencial para entender dónde se producen los posibles fallos, de ahí es donde surge el problema de investigación, verificando los canales y sistemas de almacenamiento de alto rendimiento para detectar las sobrecargas en la forma que escriben en los discos, en el sistema de archivos LUSTRE [19].

Hipótesis

Las tasas de fracaso de los sistemas HPC son altas. En general, los cuellos de botella, sobrecargas, datos mal escritos, se producen como resultado de, hardware, consumo, fallos de software, factores humanos, ataques maliciosos, congestión de la red, sobrecarga del servidor, y otras causas desconocidas [28]. Se puede desarrollar un algoritmo que según las sobrecargas o posibles fallos presentados, se tome una decisión a partir de los datos de que es lo que puede estar generando la misma.

Objetivos

A continuación se presentaran los objetivos de la investigación propuesta así entender, la estructura y analizar la forma en que se va abordar el proyecto, todos estos están relacionados con sobrecargas en HPC con lo cual no muestra su alineación con la tesis.

Objetivo General

Elaborar un algoritmo para detectar sobrecargas o posibles fallos en un sistema de librería HPC de un cluster local.

Objetivos específicos

1. Realizar experimento con montaje de sistema operativo Rocks 6.2, sistema de librería LUSTRE y herramienta de categorización Darshan para identificación de fallos de sistema según las métricas, caching, taza de transferencia, tipos de archivo, concurrencia.

2. Definir patrones encontrados en las sobrecargas y evaluar algoritmos existentes que fueron implementados para cada sobrecarga.

3. Proponer algoritmo desarrollado en r, sobre un sistema linux centos, y file system LUSTRE.

4. Validación de algoritmo, se entregaran estadísticas de cantidades y caracterizaciones, en los problemas de lectura y escritura de archivos en el sistema de archivos antes mencionado.

REVISIÓN DE LITERATURA / ANTECEDENTES

En la revisión de literatura realizada, la investigación acerca de las sobrecargas y posibles fallos en la lectura y escritura en discos de los sistemas HPC, se observan con problemas en la fiabilidad [6]. Los archivos sueltos, presentan problemas en los sistemas de archivos, según las estructuras de datos con problemas en la cantidad de operación en caché, y tiempo de lectura [7]. Se observó como también se presentan muchos errores en la consistencia de datos, en la mayoría de documentación, nos dice que existen problemas en la transferencias de datos a discos en HPC, nos muestra que se debe mejorar el diseño que hacen necesaria la realización operadores de procesador para transferir con frecuencia a un, mayor volumen [7], toda esta literatura se muestra en la tabla 1.

Articulo	**Autores**	**Publicación**	**Comentarios**
A Checkpoint storage System for Desktop Grid Computing	Samer Al-Kiswany Matei Ripeanu Sudharshan S Vazhkudai Abdullah Gharaibeh	The 28th International Conference on Distributed Computing Systems	Q1 y Q2. Presentan argumentos que apoyan la premisa de que los puntos de control de entradas y salidas es una operación de escritura intensiva, lo que requiere nuevas soluciones de almacenamiento.
HPC Global File System Performance Analysis Using A Scientific-Application Derived Benchmark	Borrill, Julian	Lawrence Berkeley National Laboratory	Q1 y Q2. Estudio que representa uno de los más extensos análisis de entradas y salidas de los sistemas de archivos paralelos modernos, explorando una amplia gama de arquitecturas y configuraciones del sistema.
A Multiobjective Genetic Programming-Based Ensemble for Simultaneous Feature Selection and Classification	Nag, K.a Pal, N.R.b	IEEE Transactions on Cybernetics	Q1 y Q2. Representa la efectividad de los esquemas propuestos de reducción de tamaño fijo y reglas. Además, comparan su método con cuatro métodos de clasificación en conjunto con seis algoritmos de selección de características y conjunto completo de características.

Batching operations to improve the performance of a distributed metadata service	Avilés-González, A Piernas, J González-Férez, P	Journal of Supercomputing	Q2 Mejorar limitaciones en el rendimiento alcanzado por los sistemas de archivos distribuidos y paralelos, debido a los gastos generales de procesamiento de mensajes, las latencias bajas, anchos de banda y las posibles congestiones.
Determining the appropriate number of nodes for fast mining of frequent patterns in distributed computing environments	Lin, W.-T. Chu, C.-P.	International Journal of Parallel, Emergent and Distributed Systems	Q2. La eficiencia en la ejecución y escalabilidad, para la minería patrones frecuentes (fps), nos hace acelerar la ejecución, se han propuesto muchos algoritmos basados en el enfoque generar y prueba utilizando tecnologías paralelas distribuidas en los últimos años.
Performance model-directed data sieving for high-performance I/O	Chen, Y.a Lu, Y.a Amritkar, P.a Thakur, R.b Zhuang, Y.a	Journal of Supercomputing	Q1 y Q2. En este estudio, mejora el rendimiento del enfoque de datos existente considerablemente, y reduce el consumo de memoria, que se han verificado por el análisis teórico y los resultados experimentales.
A parallel file system with application-aware data layout policies for massive remote sensing image processing in digital earth	Wang, L.a Ma, Y.a Zomaya, A.Y.b Ranjan, R.c Chen, D.d	IEEE Transactions on Parallel and Distributed Systems	Q2. La carga de I/O por las cantidades masivas de datos RS y los patrones de acceso de datos RS irregulares, ha mostrado el clúster paralelo basado en sistemas tradicionales de I/O ya no es aplicable.
The dispatch time aligning I/O scheduling for parallel file systems	Liu, Y Qin, J Figueiredo, R	Cluster Computing	Q2. En los sistemas de archivos paralelos (PFS), un archivo de solicitud de datos I/O puede dividirse en múltiples I/O sub-solicitudes a través del sistema de almacenamiento.
A probability-based load balancing algorithm for parallel file systems	Li, Y.a Feng, D.b Shi, Z.b Zheng, Y.b	Journal of the Chinese Institute of Engineers, Transactions of the Chinese Institute of Engineers,Series A/Chung-kuo Kung Ch'eng Hsuch K'an	Q1 y Q2. En los sistemas, se coloca un archivo comúnmente en varios nodos de almacenamiento utilizando técnicas de replicación o seccion de archivo simultáneo I / O. Sin embargo, los nodos de almacenamiento se pueden quitar o agregar como resultado de fallo de un nodo o actualización.
A prediction-based dynamic file assignment strategy for parallel file systems	Long, S.a Zhao, Y.a Chen, W.a Tang, Y.b	Parallel Computing	Cuartiles Q1, Q2 y Q3. Para reducir al mínimo el tiempo medio de respuesta entre discos, en diferentes condiciones de carga de

			trabajo, y una comparación de la PDFA con los algoritmos de asignación de archivos conocidos, tales como HP y SOR.
Research on optimization towards hybrid and hierarchy storage architecture	Zhou, E Zhang, W Dong, Y Lu, Y	Guofang Keji Daxue Xuebao/Journal of National University of Defense Technology	Q1 y Q2. Las evaluaciones en el sistema muestran que la aplicación acoplada puede mejorar el rendimiento de las aplicaciones típicas de uso intensivo de datos de forma espectacular.
IOPro: a parallel I/O profiling and visualization framework for high-performance storage systems	Kim, S.J.a Zhang, Y.b Son, S.W.c Kandemir, M.a Liao, W.-K.d Thakur, R.e Choudhary, A.d	Journal of Supercomputing	Q1 y Q2. Para diseñar aplicaciones paralelas eficientes, la comprensión del flujo de operaciones complicadas de I/O y las interacciones involucradas entre las bibliotecas es por excelencia la base de I/O.
Scalable metadata management through OSD+ devices	Avilés-González, A Piernas, J González-Férez, P	International Journal of Parallel Programming	Q2 El rendimiento de el grupo de metadatos basado en OSD ha sido evaluado y comparado con el alcanzado por Lustre.
Enabling dynamic file I/O path selection at runtime for parallel file system	Li, X.a b Xiao, L.a b Qiu, M.c Dong, B.a b Ruan, L.a b	Journal of Supercomputing	Q1 y Q2. presenta un marco que permite que el archivo de I/O de selección de ruta dinámica con granularidad fina en tiempo de ejecución, permitir que los sistemas de archivos correspondientes eligen optimizaciones para servir peticiones I/O.
Self-tuning optimization on storage servers in parallel file systems	Liao, J	Journal of Circuits, Systems and Computers	Q1 y Q2. Operaciones lógicas de I/O de acceso y su correspondiente acceso físico a los servidores de almacenamiento en un sistema de archivos en paralelo, para construir un sistema de almacenamiento de auto-ajuste.
The experience in designing and evaluating the high performance cluster Netuno	Silva, G.P.a Correa, J.a Bentes, C.b Guedes, S.c Gabioux, M.d	International Journal of Parallel Programming	Q2. Se presenta una evaluación de desempeño detallado de un cluster (Netuno), su rendimiento I / O, así como los resultados para dos aplicaciones del mundo real.
A survey of checkpoint/restart techniques on distributed memory systems	Shahzad, F Wittmann, M Kreutzer, M Zeiser, T Hager, G Wellein, G	Parallel Processing Letters	Q1 y Q2. Se muestra un ambiente de tolerancia a fallos con ejecución de grandes aplicaciones. Punto de control / reinicio (C / R) método clásico y más popular para minimizar el daño fracaso.

ALDM: Adaptive loading data migration in distributed file systems	Tan, Z Zhou, W Feng, D Zhang, W	IEEE Transactions on Magnetics	Q1 y Q2 Se analizan los problemas de balance de carga causada por acceso a datos en sistemas de archivos paralelos y le da una forma precisa para estimar la carga de los servidores de datos.
Parallel universal access layer: A scalable I/O library for integrated tokamak modeling	Galonska, A.a Gibbon, P.a Imbeaux, F.b Frauel, Y.b Guillerminet, B.b	Computer Physics Communications	Q1 y Q2 Se describe una extensión paralela I/O de la Capa de Acceso Universal, una interfaz que se utiliza para el intercambio de objetos de datos estandarizados entre Integrado de Modelado Tokamak (ITM) códigos.
Parallel file system analysis through application I/O Tracing	Wright, S.A.a Hammond, S.D.b Pennycook, S.J.a Bird, R.F.a Herdman, J.A.c Miller, I.c	Computer Journal	Q1 y Q2 Las operaciones de entrada y salida (I/O) pueden representar una proporción significativa del tiempo de ejecución de aplicaciones informáticas paralelas, aunque ha habido varios avances en las bibliotecas de formato de archivo, el diseño del sistema de archivos y de I/O de hardware, existe una creciente divergencia entre el desempeño de los sistemas de archivos paralelos y las torres de ordenadores que apoyan.
Improved algorithm FGF-LRU for I/O performance of Lustre	Linlin, L Liangxu, S	Journal of Theoretical and Applied Information Technology	Q1 y Q2 Se analiza el modo de acceso de I/O de Lustre, y el procedimiento de servicio y la página de recogida de algoritmo, y propuso algoritmo LRU (FGF-LRU).
Parallel file system performances in fusion data storage	Iannone, F.a Podda, S.b Bracco, G.b Manduchi, G.c Maslennikov, A.d Migliori, S.b	Fusion Engineering and Design	Muestran las altas tasas de I/O en un banco de pruebas emulado sobre la base de los sistemas de archivos paralelos como Lustre y GPFS, que se utiliza comúnmente en computación de alto rendimiento.
A dynamic and adaptive load balancing strategy for parallel file system with large-scale I/O servers	Dong, B Li, X Wu, Q Xiao, L Ruan, L	Journal of Parallel and Distributed Computing	Q1 y Q2 Se presenta SALB, un algoritmo de balanceo de carga dinámica y adaptable que se basa totalmente en una arquitectura distribuida.
Delegation-based I/O mechanism for high performance computing systems	Nisar, A.a Liao, W.-K.b Choudhary, A.c	IEEE Transactions on Parallel and Distributed Systems	Q1 y Q2. Se muestra como la consistencia de datos estrictos adoptados de los sistemas de archivos tradicionales son inadecuados para las

			plataformas de computación en paralelo.
Design and evaluation of an online anomaly detector for distributed storage systems.	Chen, X.a He, X.b Guo, H.c Wang, Y.d	Journal of Software	Q1 y Q2. Se presenta un detector de anomalías rendimiento que es capaz de detectar de manera eficiente anomalías en rendimiento y precisión identificar las fuentes defectuosas en un nodo del sistema de un sistema de almacenamiento distribuido.
Analysis of long-term file system activities on cluster systems	Cho, H Kim, S Lee, S	World Academy of Science, Engineering and Technology	Q1 y Q2. Miden y analizan las actividades del sistema de archivos en dos sistemas de cluster a gran escala que tenían los recursos de computación de alto rendimiento de nivel TFlops.
Design and evaluation of MPI file domain partitioning methods under extent-based file locking protocol	Liao, W.-K.	IEEE Transactions on Parallel and Distributed Systems	Q1 y Q2. Muestran un MPI colectiva de I/O como método eficaz para el acceso en paralelo de archivos compartidos y el mantenimiento de las órdenes canónicas de datos estructurados en archivos.
Design and evaluation of multiple-level data staging for blue gene systems	Isaila, F.a Garcia Blas, J.a Carretero, J.a Latham, R.b Ross, R.b	IEEE Transactions on Parallel and Distributed Systems	Q1 y Q2. Analizan las aplicaciones paralelas y como actualmente sufren de un desequilibrio importante entre la potencia de cálculo y ancho de banda de I/O disponibles

Fuente: Elaboración Propia a partir de Scopus (2016) y verificación de cuartil Scimago Journal & Country Rank

ESTRATEGIA METODOLÓGICA

El método de trabajo está muy acoplado a los entregables, siendo esta una investigación cuantitativa, descriptiva donde basamos la investigación en la literatura revisada, los entregables fueron,

1. Recolección de más información adquirida y hacer toda la revisión literaria.
2. Definir claramente el marco de trabajo. Identificando cada una de las variables y etiquetarlas. Diseñar el experimento.
3. Realizar la colección de datos, para su análisis e interpretación. En este momento se podrá realizar con mucha claridad la deducción conociendo claramente si hay respuesta a la pregunta de investigación.

4. Realizar una muy buena documentación con los resultados de la investigación.
5. Elaborar el algoritmo de detección de fallos.
6. Validar el algoritmo de detección de fallos.

Análisis de experimento

La estructura de nuestro HPC no es separada método de comportamiento de los HPC descritos en la literatura revisada, con lo que observamos un buen impacto en el proceso de diseño, teniendo claro lo externo y lo interno, para una buena encapsulación de los procesos, separando la lógica de lo físico en otras palabras lo más abstracto a lo más concreto.

El siguiente paso fue ir de lo general a lo específico, realizando algo explícito primero y a partir de esto lo implícito dándonos un mapa funcional de los conceptos y las relaciones, así se obtuvo un gran nivel de abstracción, que coincide con el nivel establecido como nuestro objetivo general [29].

Cronograma

Nombre de la tarea	Fecha de inicio	Fecha de vencimiento	Listo	Asignado a	Estado
Revision de literatura Experimentos realizados	20/08/16	31/08/16		John Ramirez	Completo
Revision de literatura montajes HPC	03/09/16	10/09/16		John Ramirez	Completo
Revision de servidores para centrtos de computo	10/09/16	17/09/16		John Ramirez	Completo
Instalacion de sistemas operativos en maquinas	17/09/16	24/09/16		John Ramirez	Completo
Establecer y montar software de pruebas que sature centro	17/09/16	24/09/16		John Ramirez	Completo
Instalación y pruebas herramienta de categorizacion	24/09/16	08/10/16		John Ramirez	Completo
Recoleccion de datos de sistema de archivos PFS	08/10/16	12/11/16		John Ramirez	Completo
Recolecion de datos de sistema de archivos LUSTRE	08/10/16	12/11/16		John Ramirez	Completo
Realizar articulo para Tesis 1	12/11/16	26/11/16		John Ramirez	Completo
Realizar presentación para Coloquio avance de proyecto	12/11/16	26/11/16		John Ramirez	Completo
Validar literatura revisada y segun los datos buscar nueva	26/11/16	10/12/16		John Ramirez	Completo
Validar algoritmos existentes y organizar para uso	10/12/16	04/02/17		John Ramirez	Completo
Diseñar algoritmo ajustado a datos recolectados	04/02/17	01/04/17		John Ramirez	Completo
Desarrollo propio en lenguaje r	04/02/17	20/05/17		John Ramirez	Completo
Probar algoritmo en distintos escenarios	20/05/17	02/06/17		John Ramirez	Completo
Elaboracion de documento con resultados	20/05/17	02/06/17		John Ramirez	Completo

Figura 1: Cronograma del proyecto

DESARROLLO E IMPLEMENTACIÓN

Presentación de la investigación

Esta investigación cuantitativa, es basada en la literatura de las tasas de fracaso de los sistemas HPC, que son altas como lo describimos en el contexto de nuestro

problema de investigación, en general, el fenómeno sobre el cual se va a profundizar en este proyecto son las sobrecargas las cuales se desean detectar según sea su causa [14].

Etapa física

Se toma desde el alistamiento de los equipos de cómputo hasta el cableado de la red para estos equipos se brindan un total de 10 servidores de la universidad politécnico gran colombiano y se logran dejar en correcto funcionamiento 7 equipos con los cuales se deja un máster con las mejores características del grupo de equipos y los otros 6 equipos como nodos, las características de estos se mostraran en la etapa de armado, se montan en un rack descrito más adelante, y se conectan con un switch 10/100 para su comunicación, se monta parte eléctrica y se conectan todos los equipos con esto terminamos nuestra etapa física así tenemos todo listo para el montaje de nuestro experimento.

Etapa lógica

En la etapa lógica se comenzó con la instalación del sistema operativo en el máster, seleccionando los complementos de rocks, la configuración de los discos, la IP pública y la IP privada, de ahí se configura en la Bios las IP de las tarjetas de red para que se puedan conectar los nodos al máster, e instalar el sistema operativos en los nodos, posterior se crean los discos para OST y MDT de LUSTRE estos según nuestra arquitectura inicial son creados dentro del máster, la universidad adquirió una arreglo de discos Dell pero será ya utilizados en los futuros proyectos de la universidad, se instala lustre y se montan los discos, luego se montan en los nodos por comunicación de red OSS y verificamos su funcionamiento, con esto proseguimos a instalar MPICH este se instala primero en el máster y después proseguimos a instalarlo en cada nodo, después de verificar que en las pruebas de MPICH utilice los procesadores de los nodos según las pruebas proseguimos a instalar Darshan, este se instala creando una carpeta donde se realizara la configuración del mismo y otra para los logs crean las variables de entorno, con esto se finaliza el experimento.

Desarrollo de la investigación

En la etapa de investigación tomaremos todos los datos arrojados por Darshan y se utilizara el lenguaje R para desarrollar una algoritmo que nos muestre las sobrecargas y posibles fallas comparando los atributos más relevantes en estos datos y utilizando una técnica de minería de datos llamada clustering donde encontraremos los datos que se comportan de una manera irregular y se verificara con el log que genera Darshan así añadir una etiqueta a que sucede en esos datos particulares, y así mediante comparación de logs definir cuál fue la causa de la sobrecarga, esto se podrá extrapolar a distintos sistemas de archivos, distintos tipos de cluster y distintas herramientas de caracterización.

Computer Name	Cores	Ram	Hard disk
Master	8	24 GB	1.33 TB with raid 1 and 5
Node 1	4	4 GB	600 GB
Node 2	4	4 GB	292 GB
Node 3	8	8 GB	600 GB
Node 4	8	6 GB	292 GB
Node 5	8	8 GB	292 GB
Node 6	4	2 GB	292 GB

Figura 2: Equipo Master y Nodos de cluster Politécnico Gran Colombiano

Creamos el primer centro de computación de alto desempeño del politécnico gran colombiano, con los que después de revisar funcionamiento procesador, funcionamiento de discos, funcionamiento de memorias RAM, baterías de los CMOS, fuentes redundantes, y demás componentes después de esta exhaustiva revisión, resultaron 7 equipos con características como se muestra en la figura 6.1. Se montan estos en el rack adquirido por la universidad, se realizan las conexiones eléctricas y de red, se conectan los nodos y el máster en un switch 10/100 QPCOM, se podría cambiar por un switch gigabit pero no cambiaría mucho debido a que las tarjetas de red de los servidores ya descritos son de 100M lo cual nos resulta una arquitectura como la descrita en la figura 2.

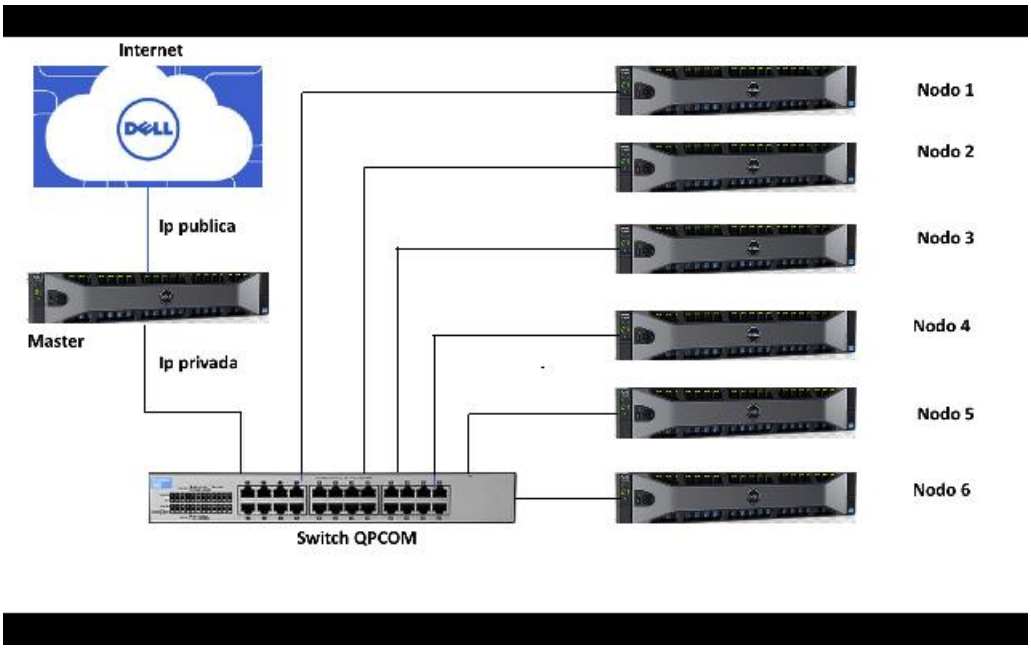

Figura 3: Equipo Master y Nodos de cluster Politécnico Gran Colombiano

Armado de centro de computación de alto desempeño y red

El montaje de los 7 equipos en el rack se hizo de una forma tal que no se logre una condición peligrosa debido a una carga mecánica irregular, se observó que la conexión del equipo al circuito de alimentación eléctrica fue adecuada que no suceda un efecto de sobrecarga en sus circuitos, observamos las protecciones de sobre corriente y el cableado, con una potencia adecuada, una puesta a tierra confiable, llevadas a las regletas de potencia del rack, de ahí conectamos los 7 equipos, el switch, y un monitor, dejamos funcionando nuestra red 10/100.

Prueba de equipos y red

Se realizó encendido de los equipos y se realizó un diagnostico por BIOS del estado de los equipos, se observa que todos tienen arreglos RAID corriendo en sus equipos con lo cual se inicia la eliminación de estos en cada uno de los equipos, una operación larga, se corren live cd de centos 7, para verificar conexión a Internet en cada una de las dos tarjetas en los 7 equipos, observando el correcto funcionamiento de los equipos, para iniciar con nuestro proceso de instalación de nuestro cluster [12].

Creación de arreglos RAID para equipos Máster y Nodos

Se crean los arreglos según la capacidad y funcionalidad de cada equipo en los nodos se realiza un arreglo RAID 1, el que es denominado mirroring, figura 3, el cual nos crea un espejo del disco que utilizamos del nodo en caso de daño de un disco el otro lo sustituirá mientras lo remplazamos, en el master lo que se realiza es un RAID 5 o distribuido con paridad, figura 4, debido a que necesitamos buen espacio para la creación de discos OST y MDT, para el funcionamiento de nuestro lustre,

este proceso al igual que el borrado toma bastante tiempo, debido a nuestro servidor [12].

Figura 4: Creación de Arreglo en nodos

Figura 5: Creación de arreglo en máster

Configuración en BIOS para desarrollo del sistema

Esta parte es muy necesaria para el reconocimiento de entorno de ejecución de pre-arranque PXE, debido a que este es el que se solicita al momento de vincular los nodos al máster, así se intente instalar con CD cada nodo no va a encontrar el master y nos arrojara error en la instalación, se configuraron las ip en una opción de la BIOS dejando el master con IP 35.0.0.2 y dejando en los nodos de la 35.0.0.10 hasta la 35.0.0.15 cómo se visualizan en las figuras 5 y 6.

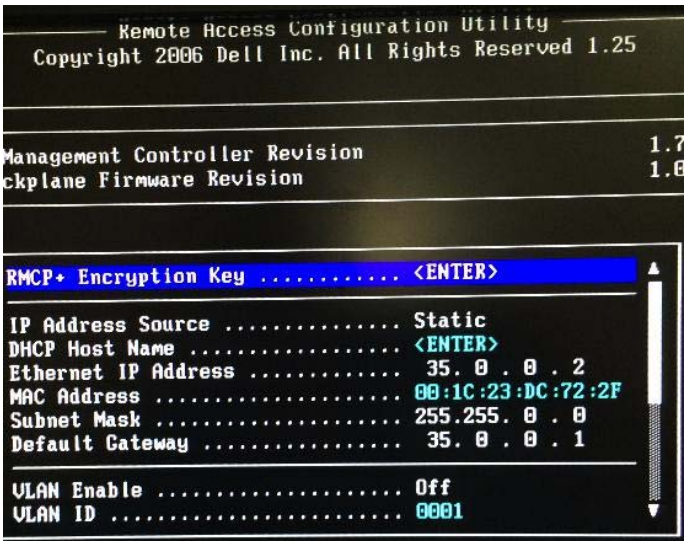

Figura 6: Configuración tarjeta de red máster

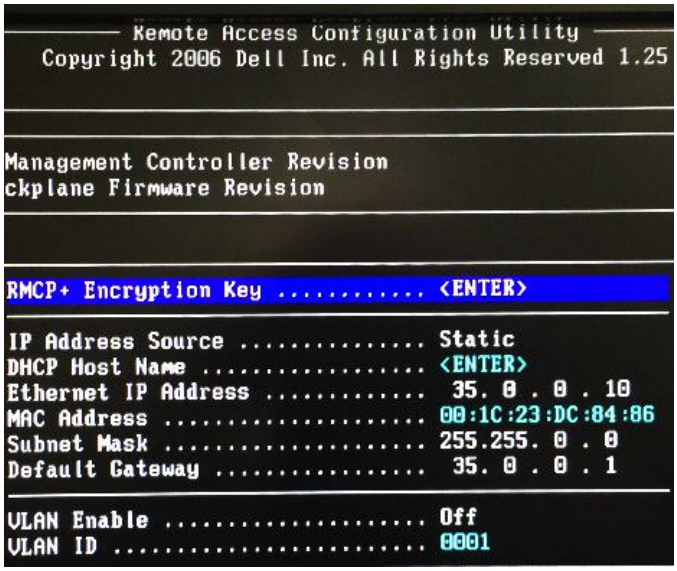

Figura 7: Configuración tarjeta de red nodos

Instalación de sistema operativo en máster y selección de características

Se descarga la versión de rocks 6.2 estable de la página de rock mencionada en la referencia de este párrafo, se verifica el funcionamiento de las dos tarjetas de red para la configuración de la ip pública y la ip privada, al insertar el cd boot escribimos build, para iniciar la instalación del sistema operativo , e seleccionan las características que vamos a utilizar, y no seleccionar algunas que utilicen procesamiento que no se va a utilizar, seleccionamos base, ganglia, kernel, os, y web server , luego se configura la información del cluster, donde solo se modifica el nombre de host, de ahí se prosigue con la configuración de las ip en las tarjetas de red dejando una para el acceso a internet, con ip brindada por el departamento de sistemas del politécnico, y la otra con la ip privada seleccionada 35.0.0.2, mascara /24, que es con la que nos conectaremos con los nodos, asignamos la contraseña de usuario root, configuramos la zona horaria y escribimos el servidor NTP, y por último se dejan las particiones en modo automático, con esto ingresamos al sistema operativo y verificamos que se acceda a internet con esto finalizamos la instalación de nuestro master.

Instalación de software en nodos

Para conectar los nodos al máster se inserta el cd de rocks en cada nodo, pero antes de encenderlo se debe correr un comando en el terminal del máster, el cual es insert-ethers esto nos abre una ventana, con la cual se irán registrando en el máster cada nodo agregado, escogemos la opción compute, y proseguimos con el encendido del nodo, en la ventana inicial como la de la figura 6.6 no se digita nada, y por el protocolo PXE se inicia automáticamente, en el máster nos aparecerá en la ventana abierta, un nombre compute-0-0 y así fueron apareciendo a medida que se ingresaron más nodos apareció compute-0-1 y así sucesivamente, por ultimo nos

conectamos desde terminal por ssh a cada uno de los nodos, y verificamos que todos quedaron funcionales [30].

Pruebas e información de funcionamiento de ROCKS

Para verificar el funcionamiento de nuestro cluster, lo realizamos de tres maneras, una de ellas, fue verificando el listado de nodos, con el comando en terminal rocks list host y observamos el listado de nodos, con lo cual prosigo a realizarle ping a cada nodo del listado, con esto ya verificado se observa la cantidad de equipos y núcleos, abriendo un navegador y se escribe en la url http://localhost/ganglia lo que nos muestra los equipos conectados y la cantidad de núcleos disponibles para nuestros procesos de computación paralela, por ultimo realizamos la instalación de un monitor más detallado que nos muestra, memoria discos kernel, para este se crea una carpeta que en este caso fue nombrada nmon, creada en /root, y se ingresa a la carpeta creada y escribimos dos comando de descargas,

“wget http://sourceforge.net/projects/nmon/files/lmon16g.c”,

y el segundo el archivo que se encarga de compilar este

“wget http://sourceforge.net/projects/nmon/files/makefile”,

Después de descargar estos, se crea un link con el comando

“ln lmon16g.c lmon.c”, se compila con el comando "make", y se ejecuta con el comando “./nmon_powe_rhel3”, con esto ya nos mostrara nuestro monitor, con esto verificamos todas las variables de importancia de nuestro cluster [10].

Figura 8: Monitor de cluster nmon

Configuración de discos OST y MDT

En la etapa de definición de equipos, se mostró en la figura 5, en el inicio como el master tenía dos tipos de arreglos el RAID5 y el RAID1 en este se dejaron 2 disco con arreglo RAID 1 con capacidades de 300 GB y 100 GB para OST y MDT respectivamente estos se formatean en el máster hasta que los visualizamos como sdb y sdc con el comando fdisk -l observando los tres disco el del sistema operativo /dev/sda, el disco MDT /dev/sdb, el Disco OST /dev/sdc, con esta construcción de discos podemos proseguir a la instalación de LUSTRE [24].

Instalación de LUSTRE en máster

Se creó una carpeta en /root llamada archivos-master en esta carpeta descargamos uno a uno los rpm necesarios para la instalación de lustre con los siguientes comandos dentro de la carpeta creada,

"wget http://repositorio.systerminal.com/lustre/server/kernel-2.6.32-431.17.1.el6_lustre.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/kernel-firmware-2.6.32-431.17.1.el6_lustre.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/lustre-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/lustre-modules-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_64.
"wget http://repositorio.systerminal.com/lustre/server/lustre-osd-ldiskfs-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_6
"wget http://repositorio.systerminal.com/lustre/server/e2fsprogs-1.42.9.wc1-7.el6.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/e2fsprogs-libs-1.42.9.wc1-7.el6.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/libcom_err-1.42.9.wc1-7.el6.x86_64.rpm",
"wget http://repositorio.systerminal.com/lustre/server/libss-1.42.9.wc1-7.el6.x86_64.rpm",

Le otorgamos todos los permisos a esta carpeta con chmod 777 * donde está ubicado los rpm descarga-dos, se verifico que todo haya sido descargado y con permisos con el comando ls –lrt [23].

A partir de este momento se iniciamos con la instalación de lustre hay unos archivos propios de rocks que crearon conflicto y se deberán eliminar otros que necesitaron instalarse en una secuencia específica, proceso el cual describiremos a continuación,

"rpm -ivh kernel-firmware-2.6.32-431.17.1.el6_lustre.x86_64.rpm", instalación de firmware, "rpm -ivh kernel-2.6.32-431.17.1.el6_lustre.x86_64.rpm”, instalación de kernel,

para instalar el siguiente paquete que es uno más actualizado se tuvo que eliminar el anterior con, "rpm -e –nodeps libss-1.41.12-18.el6.x86_64"y ahora si instalar el más actualizado con

"rpm -ivh libss-1.42.9.wc1-7.el6.x86_64.rpm" sucede lo mismo con el siguiente paquete "rpm -e –nodeps libcom_err-1.41.12-18.el6.x86_64" y ahora si se instaló el más actualizado con

"rpm -ivh libcom_err-1.42.9.wc1-7.el6.x86_64.rpm" nuevamente se elimina los que nos crea conflicto "rpm -e –nodeps e2fsprogs-1.41.12-18.el6.x86_64" "rpm -e –nodeps e2fsprogs-libs-1.41.12-18.el6.x86_64" y se instalan los mas actualizados con

"rpm -ivh e2fsprogs-libs-1.42.9.wc1-7.el6.x86_64.rpm" "rpm -ivh e2fsprogs-1.42.9.wc1-7.el6.x86_64.rpm" seguimos con los modulos de lustre,

"rpm -ivh lustre-modules-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_64.rpm" y "rpm -ivh lustre-osd-ldiskfs-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_64.rpm"

Se prosiguió con la instalación de la librería snmp, para comunicación con los nodos con, "yum install net-snmp-lib" "yum list installer *net-snmp*"

para instalar finalmente lustre con, "rpm -ivh lustre-2.5.2-2.6.32_431.17.1.el6_lustre.x86_64.x86_64.rpm" después de la instalación de LUSTRE se debe des habilitar selinux, lo cual se realizó modificando el archivo grub.conf con el comando "vi /boot/grub/grub.conf” añadiendo en la línea de kernel “selinux=0”, otro procedimiento que se realiza es deshabilitar el firewall con “chkconfig iptables off” y se reinició el cluster, para esto se apagaron primero los nodos y de esta forma se prosiguió con el reinicio del máster, lo siguiente es crear el archivo lustre.conf, con el siguiente comando, “vi /etc/modprobe.d/lustre.conf” este se crea para que al iniciar el cluster nos cargue las funciones de red de lustre, se dígito dentro del archivo “options lnet networks=tcp0(eth0)”, y se reinició nuevamente el equipo [23].

Con este proceso terminado se realizó el montaje de los dos discos OST y MDT en los discos que ya se han configurado /dev/sdb y dev/sdc, se realiza el formateo del disco formato MGS del sdb con el comnado "mkfs.lustre –fsname=lustre –mgs –mdt –index=0 /dev/sdb", luego se prosiguió con punto de montaje con el comando “mkdir /mdt”, y por último se monta con el comando “mount –t lustre /dev/sdb /mdt” se verifico con el comando "mount", se continua con el formateo de siguente disco con formato OSS, con el comando "mkfs.lustre –fsname=lustre –mgsnode=35.0.0.2@tcp –ost –index=1 /dev/sdc", nuevamente se creó el punto de montaje con el comando “mkdir /lustre” y se montó con el comando “mount –t lustre /dev/sdc /lustre” y se verifico con el comando mount se observa ya montados los dos discos en la figura 8, después de esto, se agregaron dos lineas en el archivo

fstab ubicado en /etc/fstab las cuales son "/dev/sdb /mdt lustre defaults 0 0" y "/dev/sdd /lustre lustre defaults 0 0", con esto los discos arrancaran montados cuando se realice algún apagado o reinicio [23].

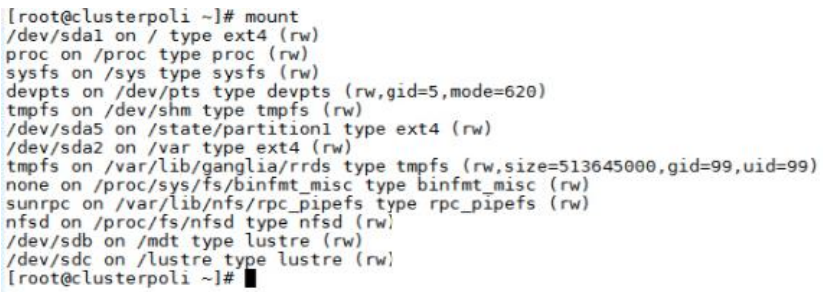

Figura 9: Particiones creadas en el máster para LUSTRE

Instalación de LUSTRE en nodos

Tal como realizamos la instalación de lustre en la anterior sección se inició con la creación de una carpeta en /root llamada archivos-cliente en esta carpeta descargamos uno a uno los rpm necesarios para la instalación de lustre en este caso son 4 se descargan con los siguientes comandos,

"wget http://repositorio.systerminal.com/lustre/cliente//lustre-client-2.5.2-2.6.32_431.17.1.el6.x86_64.x86_64.rpm"
"wget http://repositorio.systerminal.com/lustre/cliente/lustre-client-modules-2.5.2-2.6.32_431. 17.1.el6.x86_64.x86_64.rpm"
"wget ftp://mirror.switch.ch/pool/4/mirror/scientificlinux/6.3/x86_64 /updates/security /kernel-2.6.32-431.17.1. el6.x86_64.rpm",
"wget ftp://mirror.switch.ch/pool/4/ mirror/scientificlinux/6.5/i386/updates/security /kernel-firmware-2.6.32-431.17.1. el6.noarch.rpm",

se validan que se hayan descargado correctamente con ls y se ejecuta el comando chmod 777 * para proceder a copiar a cada uno de los 6 nodos con el comando "scp -rp /archivos-cliente/ compute-0-0:/root/lustre", y lo realizamos con cada uno cambiando compute-0-0 por compute-0-1, compute-0-2, y así sucesivamente hasta el sexto nodo después de realizar la transferencia de los archivos se prosiguió con la instalación digitando los siguientes comandos,

"rpm -ivh kernel-firmware-2.6.32-431.17.1.el6.noarch.rpm",
"rpm -ivh kernel-2.6.32-431.17.1.el6.x86_64.rpm",
"rpm -ivh lustre-client-modules-2.5.2-2.6.32_431.17.1.el6.x86 _64.x86_64.rpm",
"rpm -ivh lustre-client-2.5.2-2.6.32_431.17.1.el6.x86_64.x86_64.rpm".

Después de instalar los componentes de LUSTRE cliente se procedió a montar el archivo de sistema lo primero que realizamos es crear una carpeta en la raíz, y realizamos el montaje con el comando “mount –t lustre 35.0.0.2@tcp:/lustre /lustre” y nuevamente verificamos con el comando mount se observa lo de la figura 10.

```
[root@compute-0-0 lustre_clientes]# mkdir /lustre
[root@compute-0-0 lustre_clientes]# mount -t lustre 35.0.0.1@tcp:/lustre /lustre
[root@compute-0-0 lustre_clientes]# mount
/dev/sda1 on / type ext4 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
tmpfs on /dev/shm type tmpfs (rw)
/dev/sda5 on /state/partition1 type ext4 (rw)
/dev/sda2 on /var type ext4 (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
35.0.0.2@tcp:/lustre on /lustre type lustre (rw)
[root@compute-0-0 lustre_clientes]#
```

Figura 10: Montaje de LUSTRE en nodos

Instalación MPICH en master

para la instalación de MPICH se realizó a través de una secuencia de pasos para obtener una configuración predeterminada primero se realiza la descarga de la página de mpich el archivo tar mpich-3.1.3.tar.gz, se necesita en el cluster un compilador C (gcc), el cual ya se tiene en rocks, también un requisito opcional es un compilador de Fortran, este también lo contiene la distribución de rocks, específicamente gfortran, se elije un directorio de instalación, en este caso se utilizo, "mkdir /opt/mpich/mpich3", movemos la descarga de mpich y la descomprimimos alli, ahora configuramos los paquetes, con los comandos, "./configure –prefix=/opt/mpich/mpich3/", y se valido la carpeta en donde esta, con "./configure –enable-romio –enable-timer-type=gettimeofday prefix=/opt/mpich/mpich3", con esto prosigo a instalar, con "make", y tan pronto finalizo, "make install", después de esto se debe agregar la variable de entorno con, "PATH=/usr/local/bin/mpich2/bin:$PATH", con esto ya nos quedó instalado MPICH igual se debe verificar que todo está correctamente con los comandos: "which mpich" "which mpiexec" nos muestra que quedaron correctos.

Tabla 2: Lista de nodos para ejecución mpich

compute-0-1	4
compute-0-2	4
compute-0-3	8
compute-0-4	8
compute-0-5	8
compute-0-6	4

Después de esto compilamos uno de los ejemplos, que se encuentra en la ruta "/opt/mpich/mpich3/examples" con el comando "mpicc hellow.c" y ejecutamos el

compilado con comando "mpirun -np 4 hola" hola fue el nombre que se le dejo al resultado de la compilación, lo cual nos muestra los procesos con los cuatro núcleos del máster[4].

Instalación MPICH en nodos

Para realizar la instalación de mpich en los nodos, la página citada de mpich nos sugiere, que realicemos el traslado de la carpeta de instalación de mpich, que salio después de instalarlo en el máster, de ejecutar los comandos "make" y "make install", se realizó pero al correr las pruebas de mpich no realizo la parte de pruebas de computación paralelo, y salió mensaje error en comunicación con compute-0-0, compute-0-1, etc. lo cual se decide instalar mpich en cada nodo, con el procedimiento similar a como se realizó en el master en la carpeta descomprimida de mpich con,

"./configure –prefix=/opt/mpich/mpich3/ ./configure –prefix=/opt/mpich/mpich3/"
"./configure –prefix=/opt/mpich/mpich3/"
"yum install gcc gcc-gfortran gcc-c++" "./configure –prefix=/opt/mpich/mpich3/"
"make"
"make install"

Para realizar las pruebas de computación en paralelo, se creo un archivo con la relación de host este archivo llamado host_file como aparece en la tabla 6.1 y ejecutamos el comando con la mayoría de núcleos, y visualizar en ganglia el funcionamiento de todos los núcleos de los nodos, después de que se ejecutó el comando "mpiexec -f host_file -n 34 ./examples/cpi", arrojándonos una salida con los procesos del compilado cpi realizado por todos los nodos [4].

Instalación Darshan

Esta fue la última etapa de nuestro experimento la cual es la instalación de nuestra herramienta de caracterización de HPC, para la instalación de este software se debe tener unos requerimientos básicos, que se presentaran más adelante, así como las configuraciones necesarias datos de nuestro lustre, de nuestro mpich, y de nuestro máster, con esto poder observar en tiempo real el funcionamiento de nuestro HPC [2].

Requerimientos obligatorios y opcionales

La versión de darshan utilizada es darshan-util 3.1.3, esta es la colección de herramientas para HPC la cual se instaló en el máster y allí es donde guardaron los registros, este software se desarrolló para entornos Linux. Entre los requerimientos

obligatorios fueron: un compilador C. Biblioteca de desarrollo zlib, zlib-dev o similar. Y entre los requisitos opcionales según el módulo de Darshan que se utilice, entre los cuales se instalaron: Encabezados y biblioteca de desarrollo libbz2, libbz2-dev o similar. Perl. Gnuplot 4.2 o posterior. Epstopdf. Después de esto proseguimos con la instalación y configuración [2].

Instalación y configuración herramienta de caracterización Darshan

La herramienta darshan-util fue utilizada, como se observó en la literatura revisada, en las caracterizaciones revisada esto significa que se deja CC a su configuración predeterminada, o especificar un compilador local que en este caso es mpich, con esto se estableció CC a mpicc porque la biblioteca de tiempo de ejecución se utilizará en el entorno del máster, se especificó –prefix para instalar Darshan en una ubicación específica, estas se realizan en el directorio principal / no root se consultó ./configure –help para especificar rutas alternativas para librerías de desarrollo de zlib y libbz2 [2]. Para iniciar con esta instalación después de que se realizó la configuración previa, y tener darshan-3.1.3.tar.gz, lo descomprimimos con el comando:

"tar -xvzf darshan-3.1.3.tar.gz" descomprimido en la raiz "/",

Ingresamos a este folder descomprimido, con "cd darshan-3.1.3/darshan-util", después damos uno de los pasos más importantes que es la configuración de darshan,

"./configure –with-mem-align=16 –with-log-path=$WORK/darshan/log –prefix=$HOME/darshan –with-jobid-env=PBS_JOBID –with-zlib=/darshan/src/zlib/1.2.8/gnu_434 –enable-group-readable-logs -enable-shared CC=mpicc"

la descripción de cada uno de estos valores se encuentra en la tabla 3, por último se digito "make", y "make install" para finalizar con nuestra instalación [2].

Tabla 3: Opciones de configuración Darshan

Configuración	Necesidad	Descripción
		Este valor dependió de nuestro sistema

–with-mem-align	Obligatorio	rocks y Darshan lo utilizo para determinar si el búfer para una operación de lectura o escritura está alineado en la memoria.
–with-log-path	Obligatorio	Especifica el directorio donde se colocaron los registros de Darshan
–with-jobid-env	Obligatorio	la variable de entorno que Darshan comprobó para Determinar el jobid de un trabajo. Los valores comunes son PBS_JOBID o COBALT_JOBID.
CC=	Obligatorio	El compilador MPI C que se utilizo para la compilación
–with-log-path-by-env	Opcional	Variable de entorno que se utilizo para determinar la ruta de registro en tiempo de ejecución.
–with-log-hints	Opcional	las sugerencias a utilizar al escribir el archivo de registro de Darshan.
–with-zlib	Opcional	Una ubicación alternativa para el encabezado y la biblioteca de desarrollo de zlib

Argumentos de configuración y compilación

En un sistema como rocks, en los nodos, se otorgaron los permisos del directorio de registro, estos archivos de registro escritos por Darshan tienen permisos establecidos para permitir sólo el acceso de lectura por el propietario del archivo, esto se modificó con la opcion –enable-group-readable-logs que colocamos en el

momento de la configuración, con esto permitimos que todos los registros sean legibles por un usuario final otro argumento que dejamos el argumento -enable-shared para configurar se puede usar para habilitar la compilación de una versión compartida de la biblioteca darshan-util.

Módulos seleccionados para adquisición de datos Darshan

Se definieron los parámetros de Darshan, según lo que se tenia instalado, en total son 6 los parámet-ros, los cuales se configuraron con los siguientes valores, LUSTRE_OSTS=1, LUSTRE_MDTS=1,LUSTRE_STRIPE_OFFSET=4, LUSTRE_STRIPE_SIZE=1 MB, LUSTRE_STRIPE_WIDTH=-1, con esto describimos nuestro LUSTRE_COUNTERS, archivo de configuración del módulo adicional de LUSTRE que se encuentra en darshan [2].

Para ejecutar las pruebas necesarias en la herramienta Darshan, después de su instalación se recurrieron a todas las pruebas de las carpetas Darshan-util, y Darshan-test, resultando información importante de cada una de las características de los nodos, entre esta información se encuentra, accesos y porcentaje de maquina usado, tamaño de barrido, numero de procesos, numero de iteraciones, tamaño de los bloques, tiempo mínimo, máximo y promedio de escritura, tiempo mínimo, máximo, y promedio de lectura, ancho de banda de escritura y lectura, entre otras, se mostrara una muestra de una de las pruebas en la que se realizaron varias iteraciones.

darshan log version: 3.10 # compression method: ZLIB # exe: /tmp/test/mpi-io-test -f /tmp/test/mpi-io-test.tmp.dat # uid: 1000 # jobid: 22309 # start_time: 1491926876 # start_time_asci: Tue Apr 11 11:07:56 2017 # end_time: 1491926876 # end_time_asci: Tue Apr 11 11:07:56 2017 # nprocs: 4 # run time: 1 # metadata: lib_ver = 3.1.4 # metadata: h = romio_no_indep_rw=true;cb_ nodes=4

log file regions # ———————————————————-

header: 360 bytes (uncompressed)

job data: 208 bytes (compressed)

record table: 90 bytes (compressed)

POSIX module: 148 bytes (compressed), ver=3

MPI-IO module: 122 bytes (compressed), ver=2

STDIO module: 67 bytes (compressed), ver=1

mounted file systems (mount point and fs type) # ———————————————————

- # mount entry: /proc/sys/fs/binfmt_misc autofs # mount entry: /run/user/1000/gvfs fuse.gvfsd-fuse

mount entry: /sys/fs/pstore pstore # mount entry: /dev/mqueue mqueue # mount entry: /mount-old ext4 # mount entry: /dev devtmpfs # mount entry: / ext4

********** # POSIX module data # **********

Per-file summary of I/O activity. # —— # <record_id>: darshan record id for this file # <file_name>: full file name # <nprocs>: number of processes that opened the file # <slowest>: (estimated) time in seconds consumed in IO by slowest process # <avg>: average time in seconds consumed in IO per process

<record_id> <file_name> <nprocs> <slowest> <avg>

6331129185542144414 /tmp/test/mpi-io-test.tmp.dat 4 0.029831 0.024042

********** # MPI-IO module data # **********

Per-file summary of I/O activity. # —— # <record_id>: darshan record id for this file # <file_name>: full file name # <nprocs>: number of processes that opened the file # <slowest>: (estimated) time in seconds consumed in IO by slowest process # <avg>: average time in seconds consumed in IO per process

<record_id> <file_name> <nprocs> <slowest> <avg>

6331129185542144414 /tmp/test/mpi-io-test.tmp.dat 4 0.030100 0.024307

********** # STDIO module data # **********

Per-file summary of I/O activity. # —— # <record_id>: darshan record id for this file # <file_name>: full file name # <nprocs>: number of processes that opened the file # <slowest>: (estimated) time in seconds consumed in IO by slowest process # <avg>: average time in seconds consumed in IO per process

<record_id> <file_name> <nprocs> <slowest> <avg>

15920181672442173319 <STDOUT> 1 0.000047 0.000047 7238257241479193519 <STDERR>

1 0.000000 0.000000

Con esto se logra caracterizar y analizar todos los parámetros de lo que le sucede a un HPC.

Listas de archivo Darshan

Los tipos de lista que arroja la herramienta de caracterización, es algo clave sobre lo que se quiere trabajar, ya que se encuentran listas donde nos muestra el tiempo de entradas y salidas para archivos únicos, que archivo fue abierto por cada rango

en el trabajo, que archivo es independiente (abierto por un proceso), y archivos compartidos, se dio una breve descripción del archivo utilizado, los tipos de archivo según su categoría solo se muestran los escogidos.

Archivos únicos: slowest_rank_io_time: el tiempo total de entradas y salidas para archivos únicos, (incluyendo metadatos + tiempo de transferencia de datos), sobre los archivos compartidos se utilizó, time_by_cumul_, que nos agrega el tiempo acumulativo en todos los procesos y divide por el número de procesos, y time_by_cumul_io_only, que nos incluye metadatos Y tiempo de transferencia de datos, en el rendimiento total de nuestro HPC en cada proceso también será mostrado con el rendimiento a en MB / s todo esto con la opción –file acompañado de las utilidades de la carpeta darshan-util, utilizando mpiexec con cada uno de sus ejecutables, así se obtuvieron totales basados en el uso de archivos, resultando datos en varias listas de archivos que se observó con la opción –file-list que nos produjo una lista de archivos abiertos por la aplicación junto con estimaciones de la cantidad de tiempo dedicado a acceder a cada archivo [2].

Presentación de datos Darshan

En la presentación de datos arrojada por Darshan, son archivos de texto separados por tabulación con atributos y características como, Read_only, archivos que sólo se leyeron, Write_only, archivos que sólo se escribieron, Read_write, archivos que se han leído y escrito, Unique, archivos que se abrieron en un solo rango, y Shared, archivos que fueron abiertos por más de un rango, y la opcion–total que nos mostrara los datos como un total agregado en lugar de desglosadas por archivo, para un algoritmo mas ajustado a grandes potencias de supercomputación se toman datos de ALCF I/O Data Repository [3], cada uno de los campos mencionados se tomaron como atributos con R nos dio valores muy específicos entre los archivos y el proceso con mínimos y máximos globales, cuyos valores como tiempo abierto y tiempo de cierre por cada dato estadísticas que no tuvieran sentido en conjunto [2].

Parámetros y atributos tomados para detección de sobrecargas

los datos que se consideraron importantes para la investigación, y fueron considerados para cada columna de atributos, son los siguientes, <Time> este dato se tomara como id ya que es un sello de tiempo predeterminado, <r/s>,número de solicitudes de lectura emitidas por segundo, <w/s>, número de solicitudes de escritura emitidas por segundo, <rMB/s>, numero de Megabytes leidos por segundo, <wMB/s>, numero de Megabytes escritos por segundo, <svctm>, tiempo medio en atender cada solicitud en ms, <%util>, este es el porcentaje de saturación del ancho de banda, <MB_r>, numero de Megabytes leidos, <MB_w>, numero de Megabytes escritos, se caracterizan los datos, en archivos independientes que contienen estadísticas desglosadas por archivo, lo cual al momento de categorizar un proceso, nos genero varios archivos de datos con un total alto, así observar la cantidad de registros aproximadamente, después de realizarles a cada uno, una gráfica de dispersión en Excel, estos registros se toman para realizar nuestro

algoritmo en R el que nos muestra mayor inestabilidad en los datos, con el fin de garantizar, la sobrecarga, o posible fallo, se observa en la figura 10, la variación de datos [2].

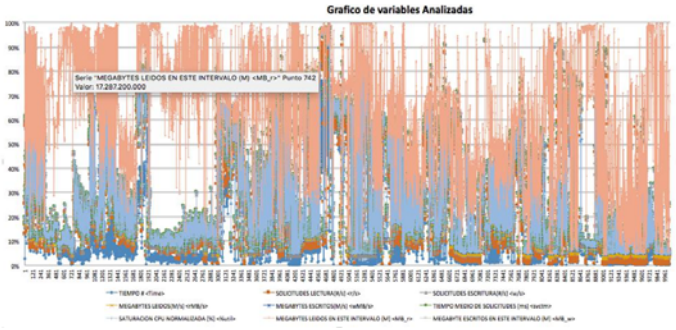

Figura 11: Gráfica con todas las variables a evaluarse en algoritmo

Algoritmo

Organización de datos para implementación de algoritmo en r para linux

Para iniciar el trabajo con los datos generados por Darshan, se realizó una completa revisión de como este nos exhibe los datos, y observar datos que nos puedan generar un diagnóstico, así tener la posibilidad de añadirlo al algoritmo y darle una descripción particular, en el momento de implementación, para realizar esta tarea, debido a la cantidad de datos, se utilizaron dos herramientas una fue complemento de minería de datos para Excel 2013 llamada "Complementos de minería de datos de Microsoft SQL Server 2012 SP1 para Microsoft Office" y la otra es rapidminer licencia educativa, algo que me dio la posibilidad de relacionar los datos y poder establecer el mejor algoritmo, y no dejar escapar ningún dato en la caracterización en el proceso de limpieza de datos [25].

R lenguaje de programación con enfoque estadístico

El Software Estadístico Libre R es un Proyecto libre de código abierto, este utiliza una sintaxis de línea de comando, con funcionalidades de abrir archivos y ver gráficos, este es disponible para las versiones Debian, Red Hat, Suse y Ubuntu de Linux, en este software, se puede, tener como apuntar y hacer clic para controlar los cálculos, ayuda en la escritura de sintaxis, con terminación automática de palabras y un amplio menú de paquetes que nos permiten muchos complementos para la programación, los paquetes permiten que cualquier persona contribuya, muchos programadores pueden desarrollar paquetes, o elegir el paquete que requiera según lo que uno esté trabajando, la edición de otros datos se debe realizar a través de la interfaz de línea de comando, r puede producir casi cualquier análisis [11].

Técnica de análisis de datos

La técnica que se utilizo es clustering y fue basada según los grupos de datos obtenidos en sus distintos algoritmos, la idea fue aplicar cada uno de estos y así

encontrar los patrones que evidencian los tipos de sobrecargas en los datos, estos algoritmos revisados, fueron los de tipos aglomerativos, jerárquicos, y por particionamiento, se escogen los de particionamiento ya que nos caracterizan mejor los tipos de datos, esto se explicó en la sección de adelante, resultando como proceso el k-means, el de error cuadrático, y uno de densidad llamado vecino más cercano para identificación de cada tipo de sobrecarga, en el algoritmo de detección [18].

Relaciones posibles entre atributos para detección de sobrecargas y posibles fallos

Para la identificación de todo tipo de cusas de las sobrecargas de este sistema se verificaron cada una de las variables de entrada en relación a las salidas, y al tiempo medio de solicitud entonces nos resultaron una cantidad considerable de gráficas, debido a la cantidad de gráficas se realiza una tabla para resumirlo, es la tabla 4, con esto se observaron más claramente que clúster encerrara que tipo de sobrecarga según su causa, gracias a los datos obtenidos de estas gráficas, los tipos de sobrecargas fueron por ancho de banda, carga de trabajo, tipo o velocidad de disco, y cantidad de solicitudes según nodo, todas evidenciadas con nuestras gráficas en las figuras, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, con estas gráficas las más observadas fueron con el tiempo de solicitud, Megabytes escritos y Megabytes leídos

Tabla 4: Gráficas que se desarrollaron para verificar datos de inestabilidad

Variables eje X	Variables eje y
r/s	MB_r
w/s	MB_w
rMB/s	MB_r
wMB/s	MB_w
r/s	%util
w/s	%util
rMB/s	%util
wMB/s	%util
r/s	svctm
w/s	svctm
rMB/s	svctm
wMB/s	svctm

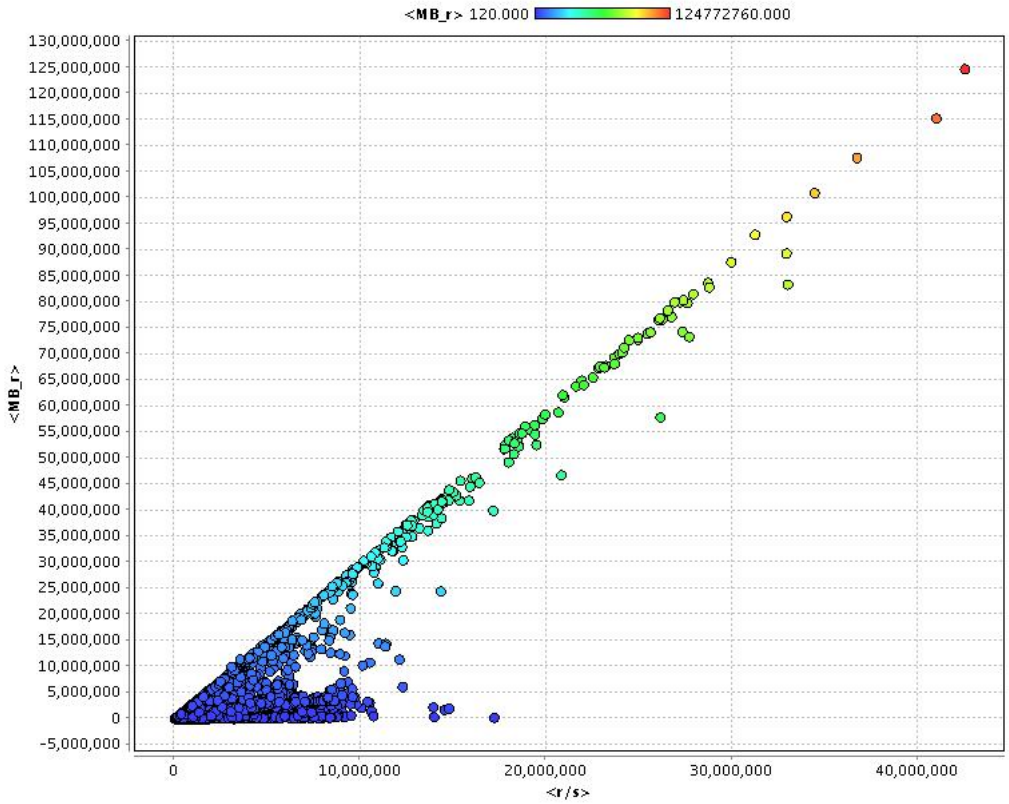

Figura 12: Gráfica cantidad de solicitudes de lectura vs MB de archivos leídos en intervalo

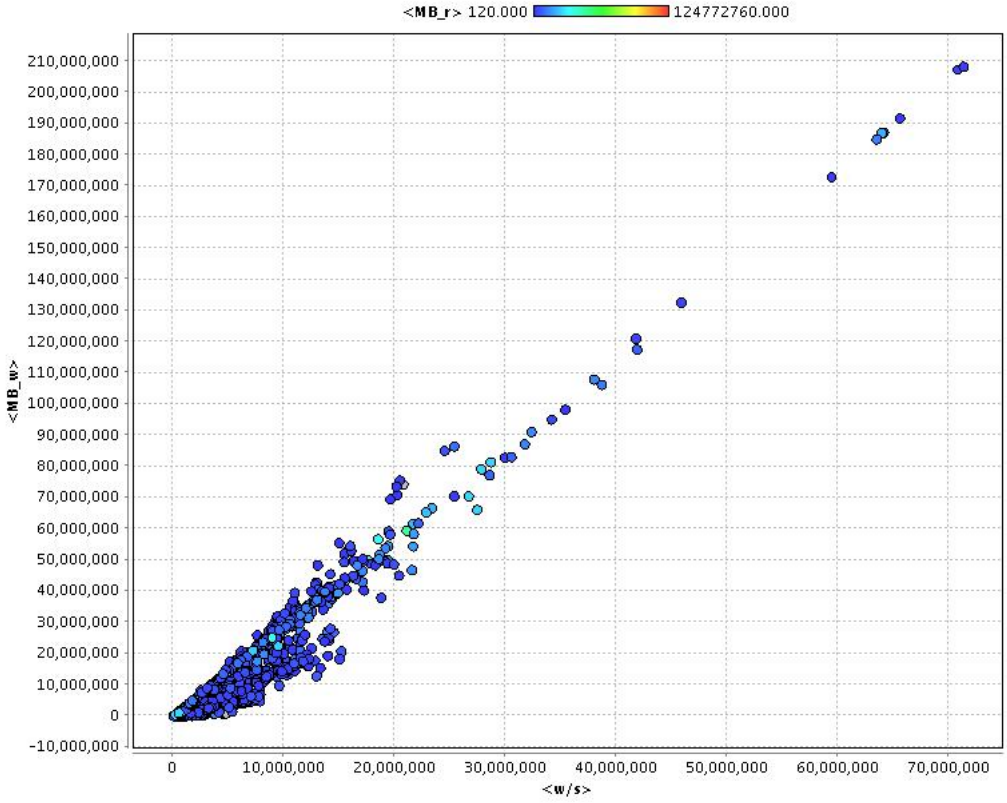

Figura 13: Gráfica cantidad de solicitudes de escritura vs MB de archivos escritos en intervalo

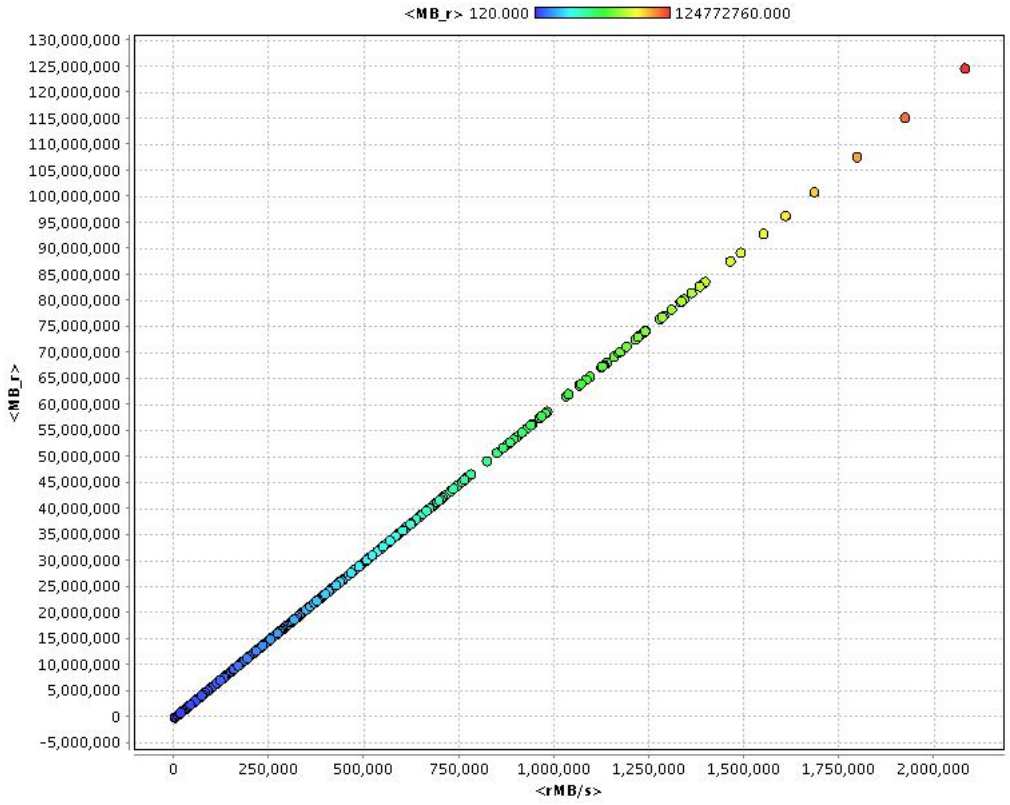

Figura 14: Gráfica megabytes leídos por segundo vs MB de archivos leídos en intervalo

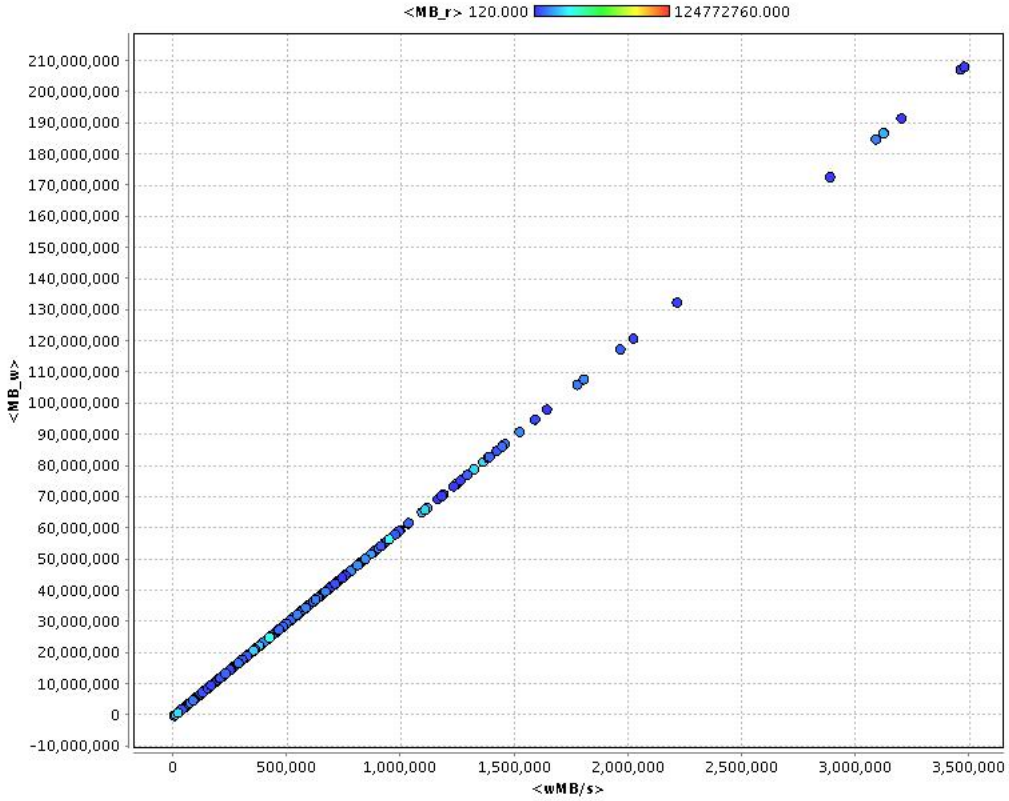

Figura 15: Gráfica megabytes escritos por segundo vs MB de archivos escritos en intervalo

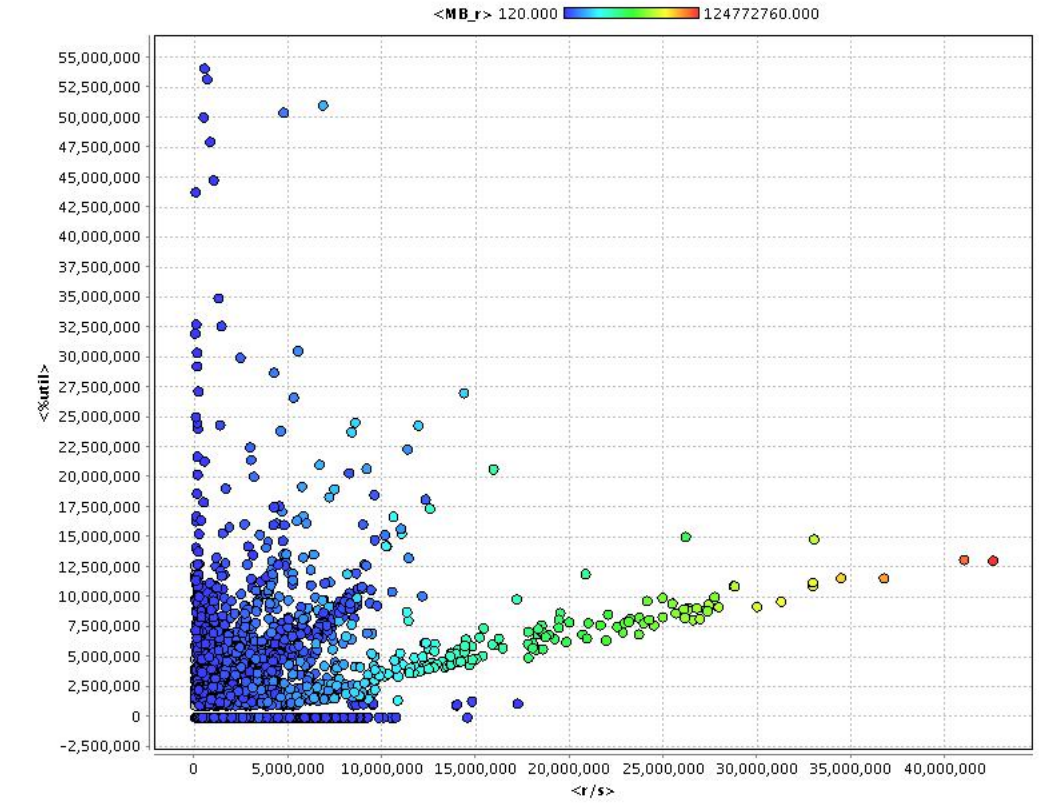

Figura16: Gráfica cantidad de solicitudes de lectura vs porcentaje de CPU

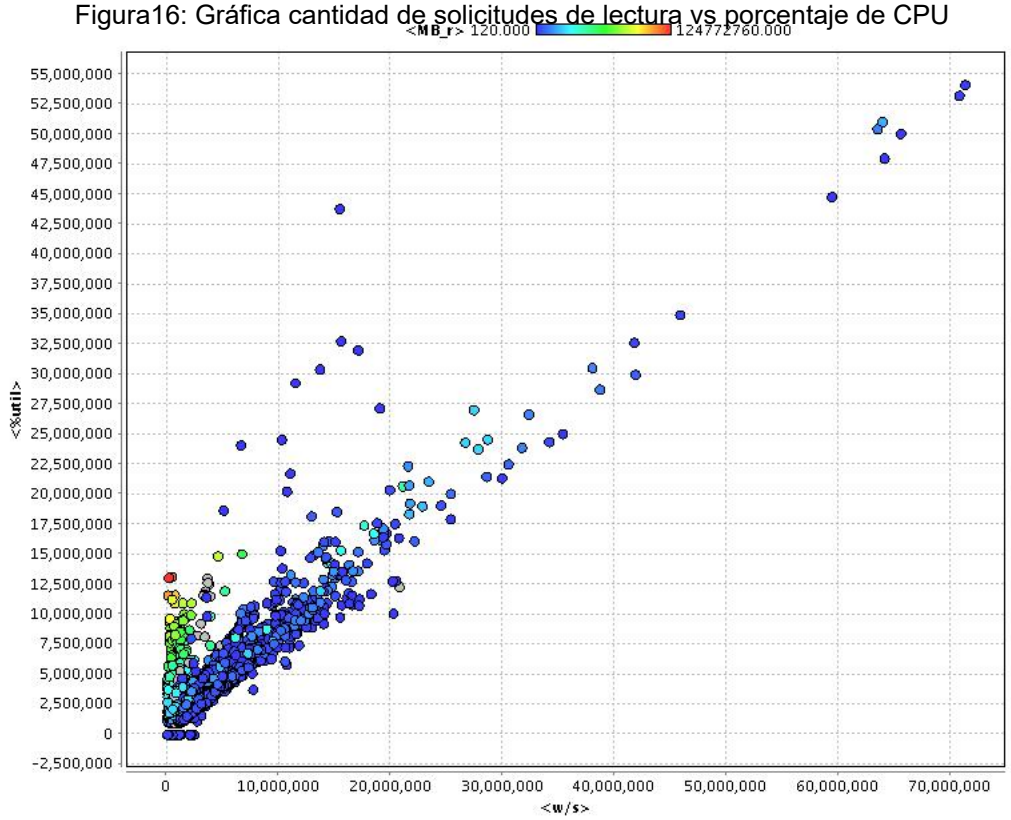

Figura 17: Gráfica cantidad de solicitudes de escritura vs porcentaje de CPU

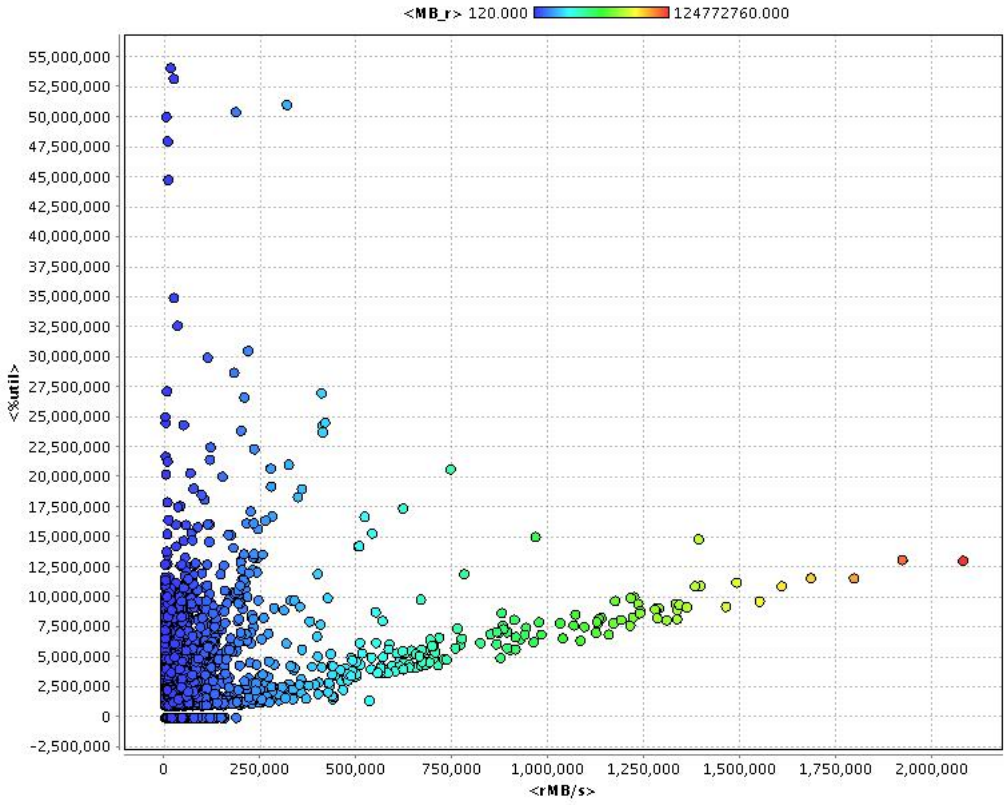

Figura 18: Gráfica Megabytes leidos por segundo vs porcentaje de CPU

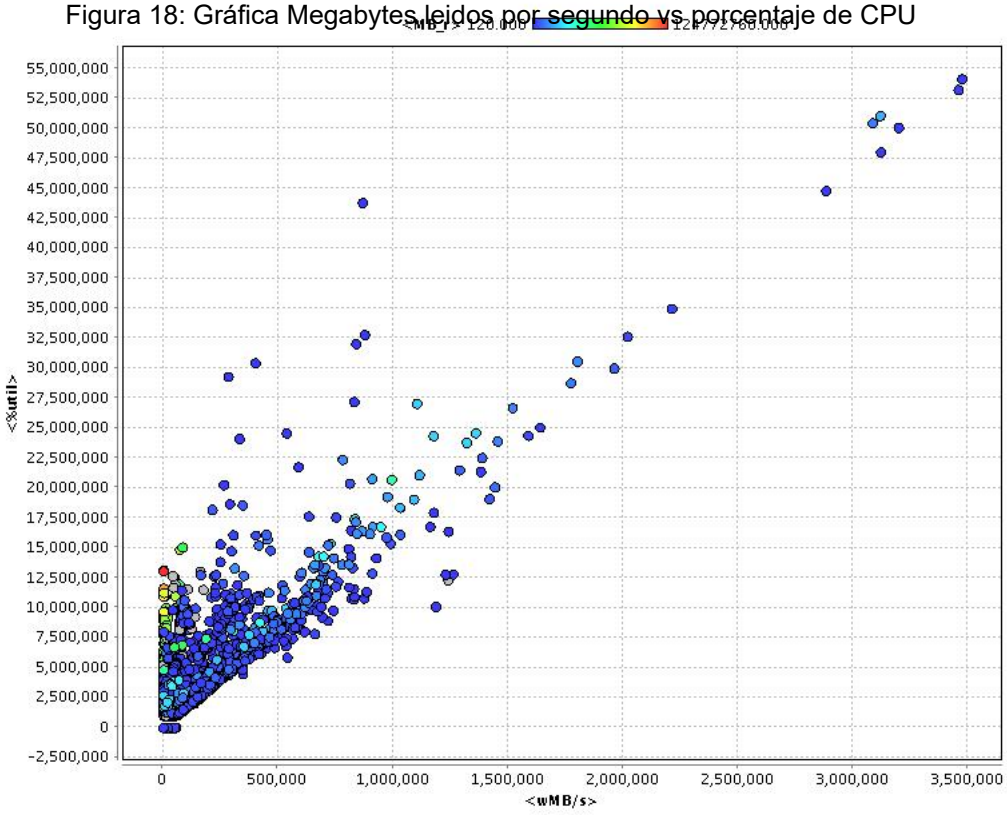

Figura 19: Gráfica Megabytes escritos por segundo vs porcentaje de CPU

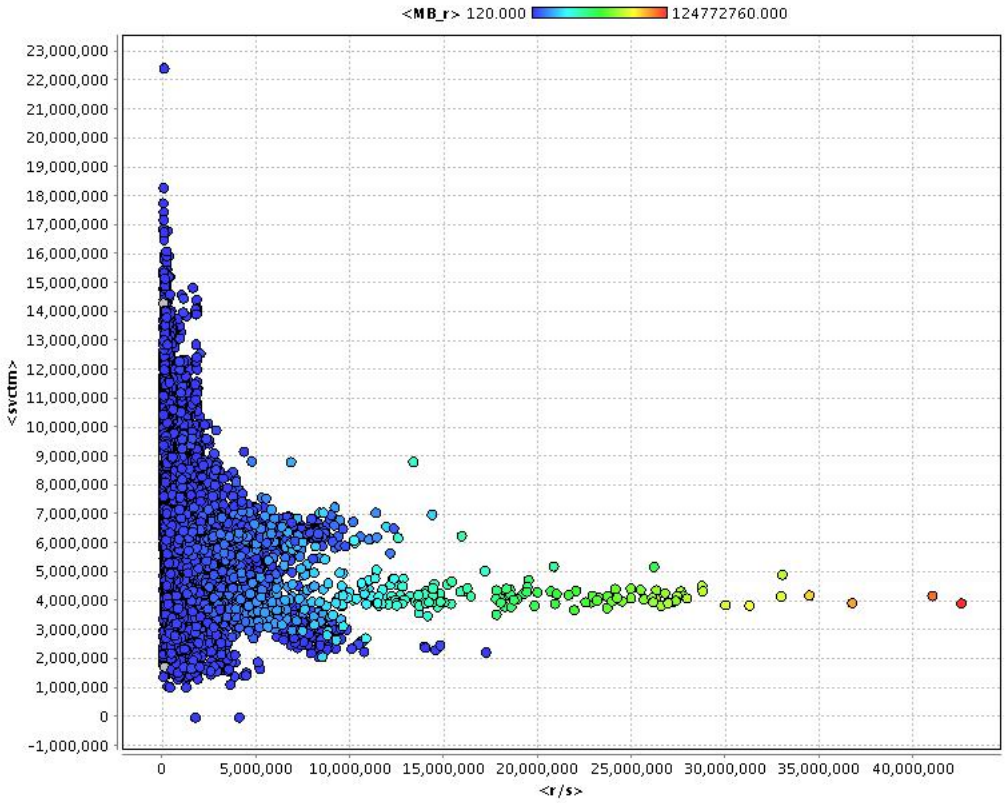

Figure 20: Gráfica cantidad de solicitudes de lectura vs MB tiempo medio en atender solicitud

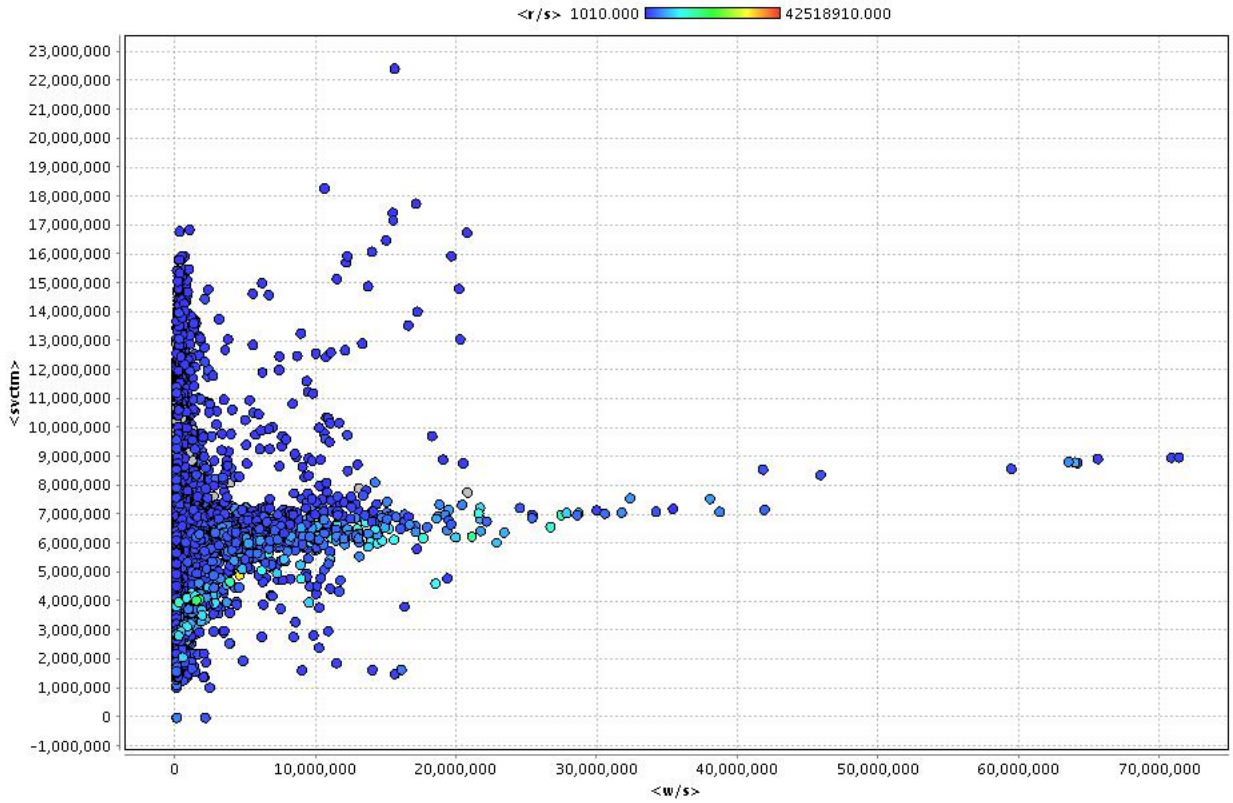

Figura 21: Gráfica cantidad de solicitudes de escritura vs tiempo medio en atender solicitud

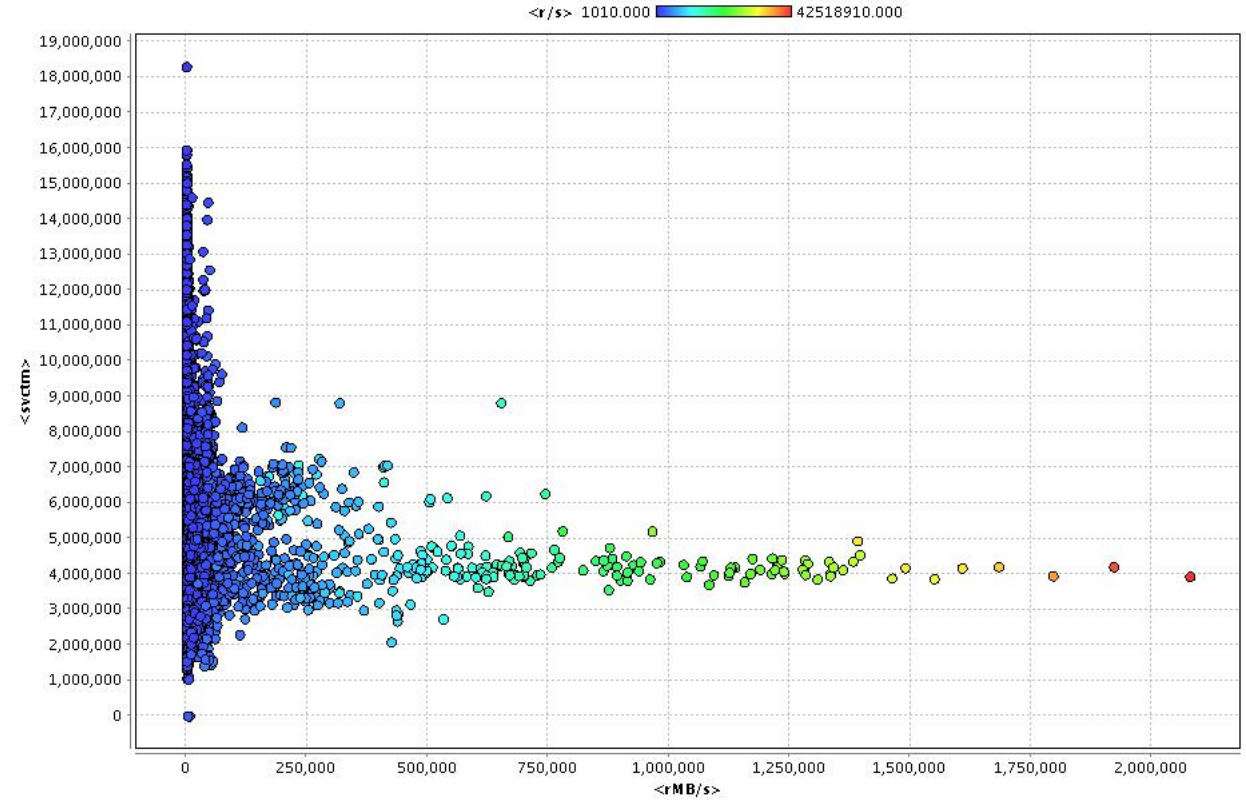

Figura 22: Gráfica megabytes leídos por segundo vs tiempo medio en atender solicitud

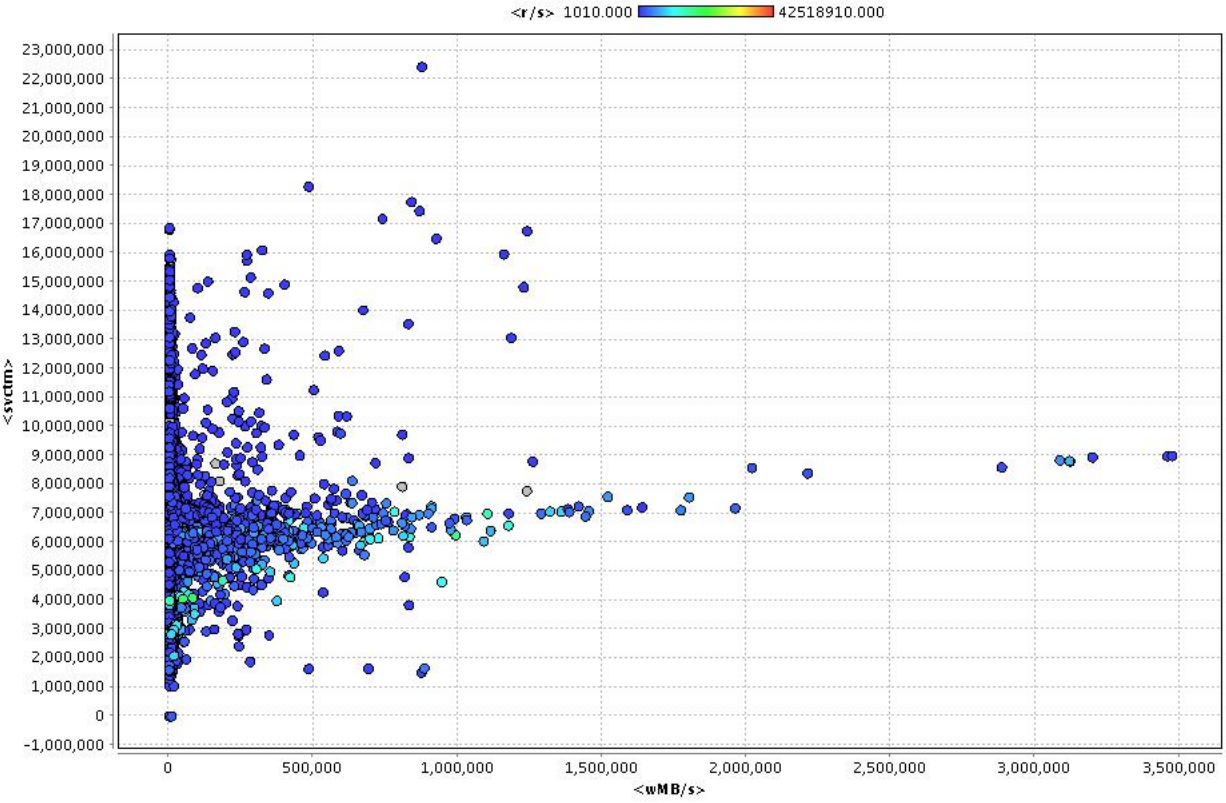

Figura 23: Gráfica megabytes escritos por segundo vs tiempo medio en atender solicitud

Aplicación clustering en datos HPC

Se decidió que la mejor forma de identificar las sobrecargas en la caracterización realizada, para clustering o agrupación, es con el algoritmo K-means, este es un agrupamiento particional más utilizado, a pesar de gradiente descendencia, ósea sensible a donde se ubiquen los centros de agrupación, debido a esto se evaluó si era suficiente con este algoritmo, con la ejecución de otros dos algoritmos, el de particionamiento denominado de error cuadrático, y un algoritmo basado en densidad el de vecinos más cercanos estos conjuntos utilizando diversos criterios de rendimiento, según lo permitió la configuración de cada uno, así poder analizar los resultados experimentales utilizando, pruebas tácticas y proporcionar una clara identificación del por qué se causó el tipo de sobrecarga, en la sección anterior utilizamos una de las primeros pasos para una buena agrupación esta fue con una clara vista en datos, donde se detectaron anomalías, rasgos salientes, entre otros, por este motivo esta técnica es buena para este gran conjunto de datos [32].

Uso de los algoritmos para segmentación de sobrecargas

El primer problema que se enfrentó y más escogiendo este tipo de solución para el problema de detección de sobrecargas, fue escoger el número adecuado de cluster, ya que cada uno de estos cluster nos puede brindar algo de información acerca del comportamiento de nuestro HPC, existen gran cantidad de métodos para solucionar esta primera parte del problema, los cuales son el método elbow, criterio calinsky, propagación affinity, GAP, Dendogramas, entre otros, de los cuales solo se utilizó, elbow, para esto se utilizó la herramienta R, donde en nuestra forma elbow.

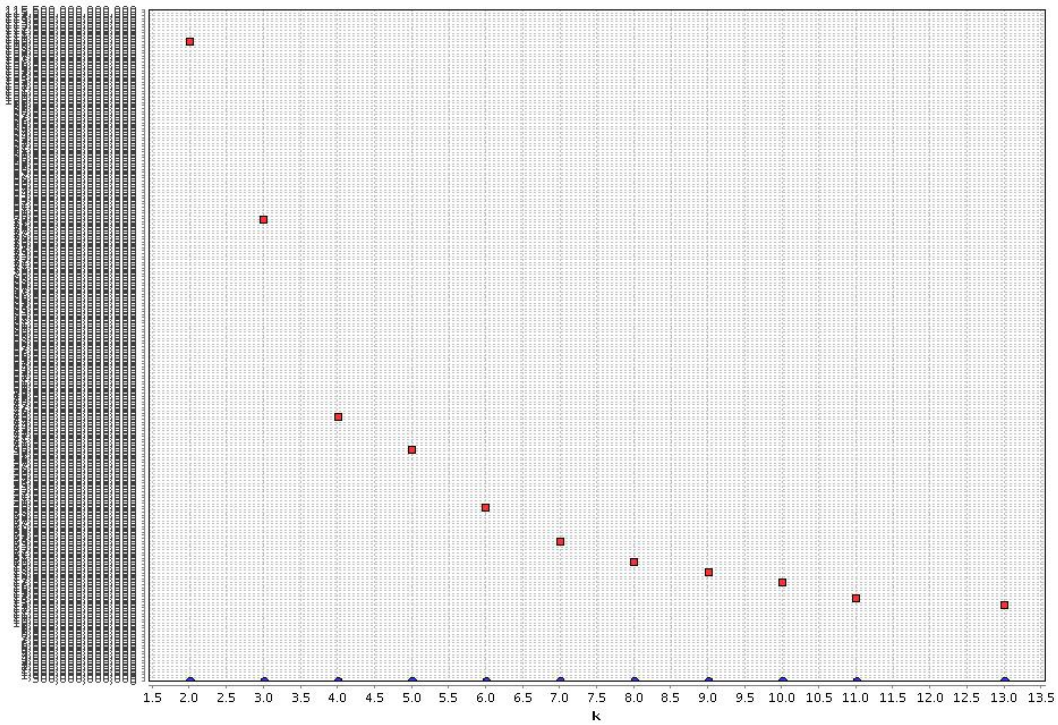

Figura 24: Técnica del codo para selección de cantidad de clusters

se realizaron los tres algoritmos de clustering, ya definidos en la sección anterior, resultandos en los dos métodos de particionamiento 4 clusteres debido a que al realizarse las pruebas con 5 y 6 cluster, resultaban cluster con datos sueltos que se podían agregar de una manera fácil a otro cluster, verificando lo efectivo que fue nuestro elbow para encontrar la cantidad de cluster con lo cual no resulto necesario ejecutar otro método de selección de cantidad de cluster, resultado, 3 cluster como herramienta de categorización de sobrecargas, y un cluster como datos de escritura y lectura normal, el algoritmo de densidad de vecino más cercano, no resulto ser adecuado para la tarea ya que solo nos creó 2 cluster en donde no se podía evidenciar el tipo de sobrecarga ya que en la densidad tomo todos los lejanos y los volvió un cluster el código utilizado el siguiente.

```
#Librerias requeridas

library(psych) # Contiene funciones de exploraciÃ³n de datos

library(fpc) # Contiene funciones de anÃ¡lisis de estabilidad de clusters


directorio <- "/Users/jramirez/GestiondeDatos/Datos"

setwd(directorio)


#Cargar datos de la bodega de Darshan

AdvWorks <-  read.csv("AdventureWorksPlano.csv")

AdvWorks<- AdvWorks[sample(nrow(AdvWorks), 500), ]


#Vincular los datos para trabajar con R

attach(AdvWorks)

summary(AdvWorks)

describe(AdvWorks)


#PreparacionDatos

#Identifica los atributos numéricos de AdventureWorks
```

```
nums <- sapply(AdvWorks, is.numeric)
#Subconjunto de AdventureWorks con solo los atributos numÃ©ricos
AdvWorksNumerical<- AdvWorks[ , nums]
summary(AdvWorksNumerical)

#Normalizar los atributos numéricos
AWNormalizado<-scale(AdvWorksNumerical)
summary(AWNormalizado)
pcenter<-attr(AWNormalizado,"scaled:center")
pscale<-attr(AWNormalizado,"scaled:scale")

matrizDistancias <- dist(AWNormalizado,method = "euclidean")
#Otros posibles métodos "maximum", "manhattan", "canberra", "binary", "minkowski"

#########Generación de los clusters jerárquicos#############
dendograma <- hclust(matrizDistancias,method="ward.D")
plot(dendograma)

#SelecciÃ³n de 4 clusters de todo el dendograma
clusters<-cutree(dendograma,k=4)



printClusters <- function (labels, k, originalDataFrame) {
  for(i in 1:k){
    print(paste("*********Cluster ",i,"*********"))
    print(
      originalDataFrame[labels==i,c("Time","rs","ws","rMBs","wMB/s","svctm","util","MB_r","MB_w
")]
      )
```

```
 }
}


#Imprimir los registros pertenecientes a cada cluster en un archivo
#mode(dendograma)
sink("ClustersResultantesUsandoJerarquia.txt")
printClusters(clusters,4,AdvWorks)
sink()
#########Clusters usando algoritmos por particionamiento#########
desiredClusters <- 4
kmeansClusters <- kmeans(AdvWorksNumerical, centers=desiredClusters, nstart = 100, iter.max = 100)
summary(kmeansClusters)
#Matriz de centroides
kmeansClusters$centers
#Tamano de los clusters
kmeansClusters$size
#Integrantes de los clusters
kclusters<-kmeansClusters$cluster
printClusters(kclusters,desiredClusters,AdvWorks)



#######Evaluación de estabilidad de los clusters
desiredClusters<-4
#EvaluaciÃ³n de los clusters usando mÃ©todo  jerÃ¡rquico
hclusters <- clusterboot(AdvWorksNumerical,B=10,clustermethod = hclustCBI , method="ward.D", seed=3, k=desiredClusters)
summary(hclusters)
hgroups <- hclusters$result$partition
printClusters(hgroups,kbest.p,AdvWorks)
```

```
#Obtiene la estabilidad de cada cluster

hclusters$bootmean


#EvaluaciÃ³n de los clusters usando mÃ©todo por particionamiento

desiredClusters<-4

kclusters <-
clusterboot(AdvWorksNumerical,B=10,clustermethod=kmeansCBI,k=desiredClusters,seed=3)

summary(kclusters)

hgroups <- kclusters$result$partition

printClusters(hgroups,desiredClusters,AdvWorks)

#Obtiene la estabilidad de cada cluster

kclusters$bootmean


#EvaluaciÃ³n del valor del k

# Primero evaluar usando el criterio CH Indice Calinski Harabasz

set.seed(2) #Para evitar aleatoriedad en los resultados

clustering.ch <- kmeansruns(AdvWorksNumerical,krange=2:10,criterion="ch",iter.max=100, runs=
100,critout=TRUE)

clustering.ch$bestk

?kmeansruns

#clustering.ch$cluster


# Segundo evaluar usando el criterio ASW

set.seed(2) #Para evitar aleatoriedad en los resultados

clustering.asw <- kmeansruns(AdvWorksNumerical,krange=2:10,criterion="asw",iter.max=100,
runs= 100,critout=TRUE)

clustering.asw$bestk

#clustering.asw$cluster
```

Cuando utilizamos un mayor número de cluster, donde solo resultan pocos datos en comparación con la carga de trabajo de nuestro HPC, para la presentación de gráficas, utilizamos rapidminer.

RESULTADOS

Los primeros resultados que se generaron fueron para escoger el mejor algoritmo que fue el de kmeans un algoritmo muy implementado en esta herramienta.

Cluster Model

```
Cluster 0: 9289 items
Cluster 1: 148 items
Cluster 2: 526 items
Cluster 3: 34 items
Total number of items: 9997
```

Figura 25: cantidad de clusters k means

Cluster Model

```
Cluster 0: 9096 items
Cluster 1: 44 items
Cluster 2: 179 items
Cluster 3: 678 items
Total number of items: 9997
```

Figura 26: cantidad de clusters error cuadrático

Cluster Model

```
Cluster 0: 196 items
Cluster 1: 9801 items
Total number of items: 9997
```

Figura 27: cantidad de clusters vecino más cercano

Attribute	cluster_0	cluster_1	cluster_2	cluster_3
<r/s>	1161965.133	17912378.716	2423888.365	3284157.059
<w/s>	605194.539	925589.932	9223007.719	36489476.17
<rMB/s>	21458.897	856620.405	89827.962	131845.529
<wMB/s>	14342.527	13357.554	396046.926	1789992.765
<svctm>	6907091.934	4220606.324	6868177.234	8521568.676
<%util>	543599.251	6505989.277	8331579.243	28007753.79
<MB_r>	1207707.163	51397224.324	5329382.852	7775362.853
<MB_w>	816717.201	801453.243	23762815.551	107399565.8

Figura 28: Datos por atributos k means

cluster	<r/s>	<w/s>	<rMB/s>	<wMB/s>	<svctm>	<%util>	<MB_r>	<MB_w>
cluster_0	1076637.303	521991.227	19488.367	12214.536	6932974.016	435178.254	1087317.537	688107.643
cluster_1	4892450.227	32709802....	201922.409	1558837.341	7270233.136	26118235....	12113575....	93530240....
cluster_2	16242759....	941250.503	772926.542	14106.575	4175018.777	5905486.905	46375592....	846394.525
cluster_3	2824982.375	8104478.289	78671.313	327681.049	6721933.625	7643640.493	4673053.168	19660862....

Figura 29: Datos por atributos error cuadratico

Row No.	cluster	<r/s>	<w/s>	<rMB/s>	<wMB/s>	<svctm>	<%util>	<MB_r>
1	cluster_0	8478120	495360	424593	16567	2086712	2122093	25475580
2	cluster_1	3515990	1457770	190615	70155	6251966	2282011	11436900

The position of the example in the (filtered) view on the example table.

Figura 30: Datos por atributos vecino más cercano

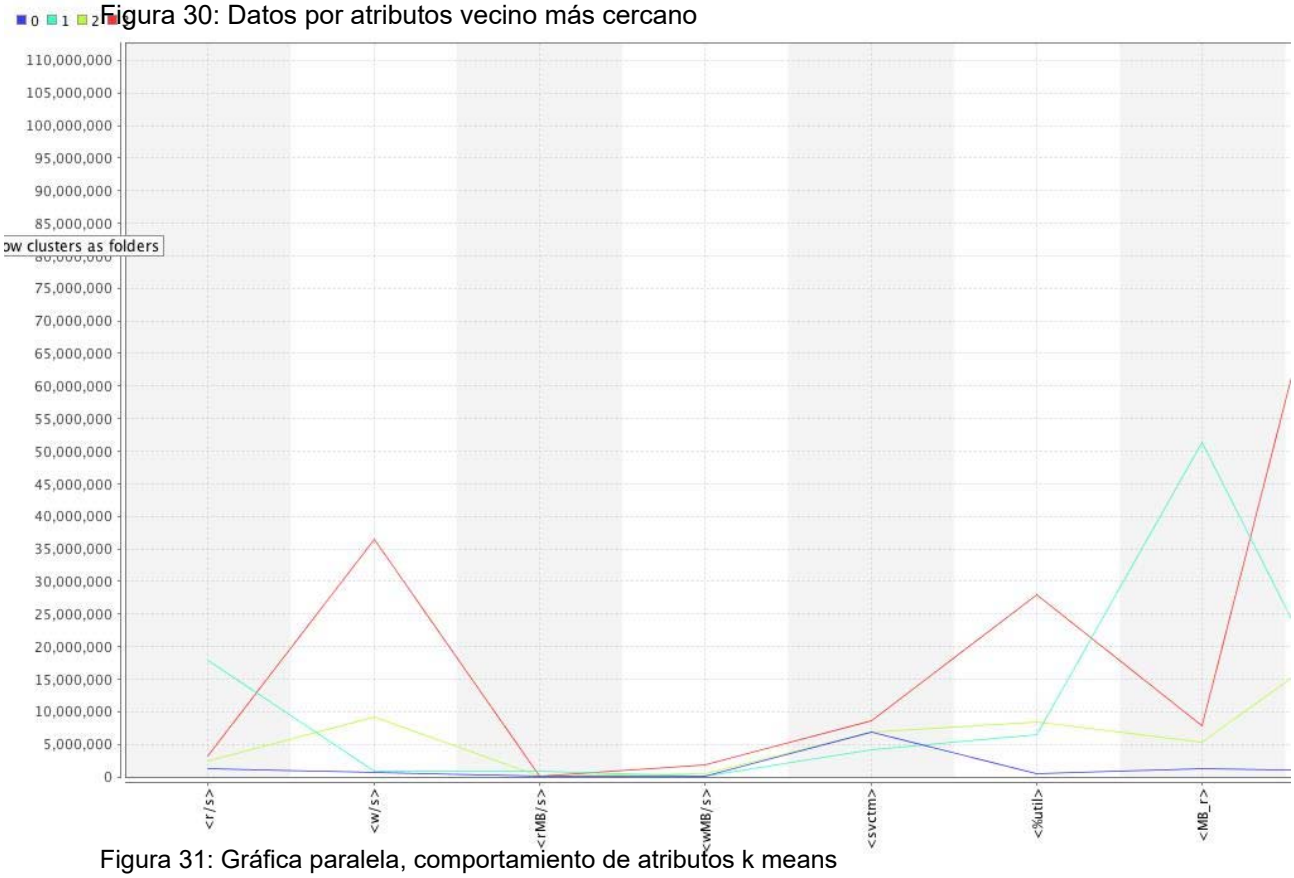

Figura 31: Gráfica paralela, comportamiento de atributos k means

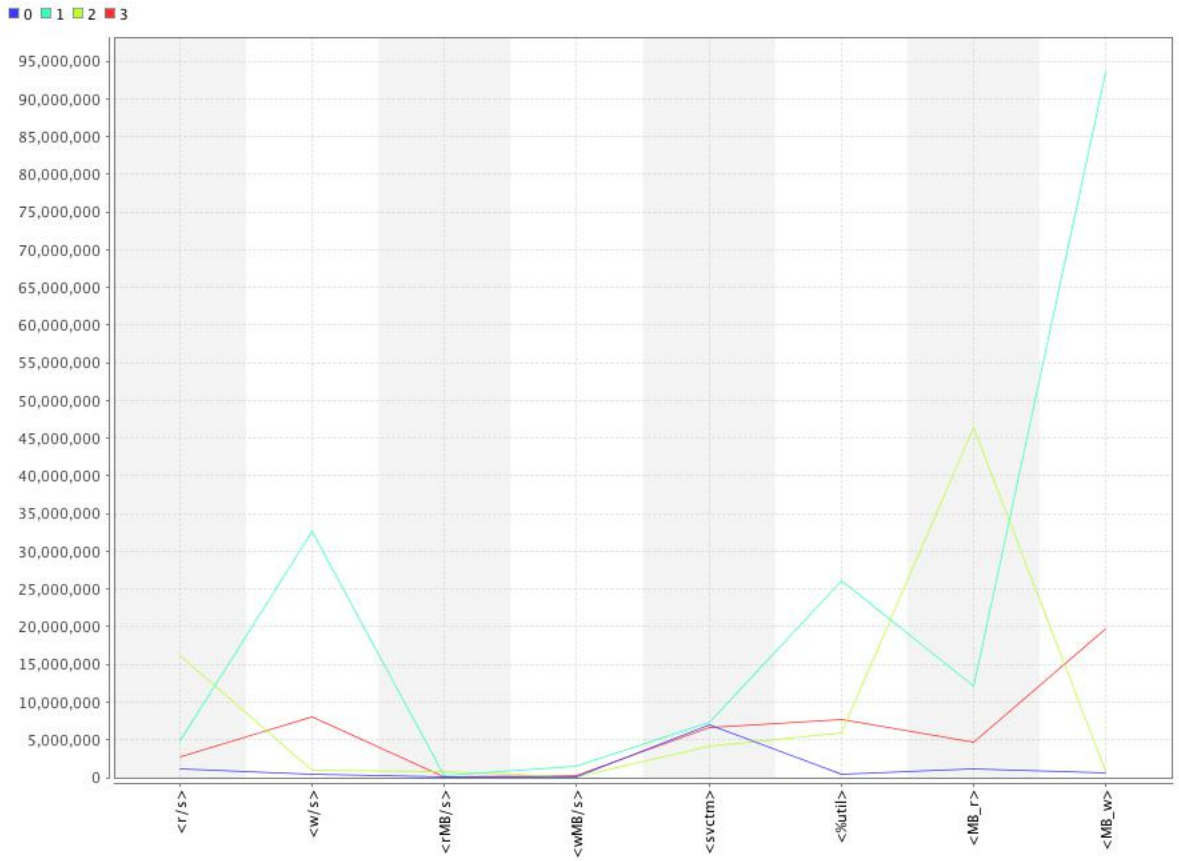

Figura 32: Gráfica paralela, comportamiento de atributos error cuadrático

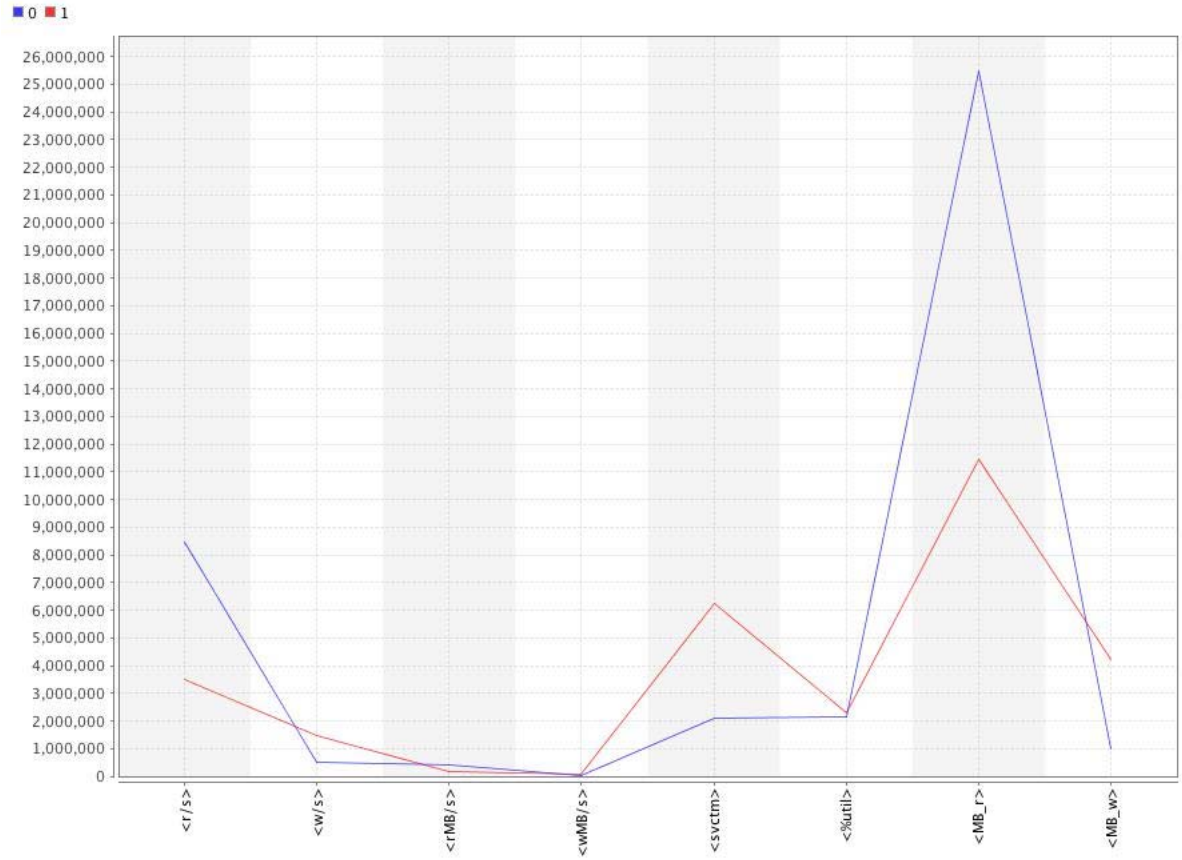

Figura 33: Gráfica paralela, comportamiento de atributos vecino más cercano

Pruebas de algoritmo con nuevas secuencias de datos generadas por darshan

De la cantidad de datos tomados de ALCF I/O Data Repository, generados con la herramienta darshan, y con la ayuda de herramienta de minería para excel, se tomaron 5 muestras de datos, para la elaboración del algoritmo se tomó la que mostro en las gráficas iniciales de variables de entrada contra variables de salida, publicadas unas secciones antes, ahora se realizaron pruebas con las 4 restantes para observar el funcionamiento del algoritmo, los resultados obtenidos fueron óptimos mostrándonos los datos de fallas en tiempo de solicitud, en cantidad de datos escritos, y cantidad de solicitudes, datos similares a la caracterización realizada con las configuraciones, establecidas en el código de r.

Presentación de sobrecargas y posibles fallos definidos

Se definieron tres tipos de fallos los cuales son correspondientes al algoritmo definido kmeans las tres fallas establecidas son representadas en el cluster 1, 2 y 3 son con tiempo medio en responder una solicitud, % de saturación de ancho de banda y cantidad de datos escritos en la gráfica paralela de estos atributos se observó que el cluster 3 que es el de menor cantidad de datos nos presenta un comportamiento inestable en las variables, escritura por segundo y % de saturación de ancho de banda hay se encontraría un posible cuello de botella como se observó en la literatura revisada ya que al observar la cantidad de solicitudes fue de las más altas en la gráfica; en el cluster 2 donde es uno de los datos más altos que sigue después del comportamiento normal, se observa un comportamiento en grafica similar al del cluster 3, con la diferencia que el tiempo medio en atender cada solicitud en ms en comparación con este cluster mencionado, es alto, lo cual se asociaría a un problema en los discos, ya sea por alguna de las causas definida en la literatura revisada, calentamiento, revoluciones, tipo de tecnología, entre otras, y el ultimo en revisarse es el cluster 1 con una sola variable inestable la cual es la cantidad de datos leídos, esta inestabilidad se asociaría a carga de trabajo ya que en la cantidad de datos leídos por segundo inicialmente también es alta y en el resto observamos comportamiento estable.

Los resultados presentados fueron óptimos, debido a la efectividad de las herramientas utilizadas, para la compresión del sistema HPC, se debe mejorar la infraestructura de nuestro centro de computación de alto desempeño, algo que ya se encuentra en proceso, aumento de capacidad de los servidores, nueva arquitectura, arreglos de discos para los sistemas de archivos, todo para desarrollos

de proyectos en la facultad, se deja una herramienta para que se utilice en los futuros proyectos desarrollados por la universidad, se encontraron los tres fallos más concurrentes y que se presentan en toda la literatura se combinan con los registros de caracterización de cada proceso, afectando el tiempo necesario para escribir los archivos en el disco.

DISCUSIÓN Y CONCLUSIONES

Se analizó el rendimiento de un HPC en un equipo con diferentes configuraciones donde tenemos en cuenta la cantidad de procesadores, distintas capacidades de memoria, y factores que determinan patrones particulares acerca del rendimiento de las entradas y salidas tratando de cuantificar la sobrecarga de datos en nuestro sistema de archivo.

Las solicitudes de entradas y salidas entre los nodos y el master efectúan un almacenamiento en caché de datos y metadatos, donde se evidencia el gran reto en la medición del rendimiento de los sistemas de archivos paralelos, con una configuración orientada a la aplicación, se analiza empíricamente este rendimiento con la herramienta de caracterización la cual según, la distribución de archivos en una aplicación de gran escala, brinda en el análisis, todos los datos de los registros.

En nuestro experimento se observa que el establecer un algoritmo, sin importar la complejidad del sistema, se basa en una secuencia de pasos para resolver un problema, lo más relevante es tener la comprensión del problema en este caso, el sistema de archivos paralelo donde se contienen, los atributos de cada objeto, allí es donde se presenta la sobrecarga como se mostró es especialmente en la escritura de los datos.

Trabajos Futuros

Más adelante pretenderé analizar el comportamiento de más sistema de archivos a partir de comunicación, en las diferentes transmisiones, y también cambio en la arquitectura de los nodos y el máster, y poder aportar a nivel de software en mejorar del código c de los módulos de Darshan, en una próxima maestría o estudios de doctorado.

REFERENCIAS

[1] Nawab Ali et al. “Scalable I/O forwarding framework for high-performance computing systems”. In: (2009), pages 1–10.

[2] Argonne. “Darshan HPC I/O Characterization Tool”. In: (2017). URL: http://www.mcs.anl.gov/research/projects/darshan/ (visited on 02/24/2017)

[3] Argonne. “Mathematics and Computer Science Tools and Technology for Solving Critical Scientific Problems”. In: (2017). URL: ftp://ftp.mcs.anl.gov/pub/ (visited on 03/24/2017).

[4] argonne. “MPICH 3.0”. In: (2017). URL: http://www.rocksclusters.org (visited on 02/04/2017).

[5] Rajesh Vellore Arumugam, Lin Wujuan, and Su Hnin Wut Yi. “Scalable Distributed Meta-Data Management in Parallel NFS”. In: (2016), pages 930–935.

[6] Silas Boyd-Wickizer et al. “An Analysis of Linux Scalability to Many Cores.” In:10.13 (2010), pages 86–93.

[7] Peter J Braam and Nathaniel Rutman. “Data integrity in a networked storage system”. In: (Dec. 2015). US Patent 9,225,780.

[8] Yann Chanel. “Study of the Lustre file system performances before its installation in a computing Cluster”. In: (2008).

[9] Chao Chen et al. “Decoupled I/O for Data-Intensive High Performance Computing”. In: (2014), pages 312–320.

[10] Linux Cluster. “NMON monitor”. In: (2016). URL: https://linuxcluster.wordpress.com/category/monitoring/nmon/ (visited on 11/04/2017).

[11] CRAN. “The R Project for Statistical Computing”. In: (2017). URL: https://www.r-project.org/ (visited on 03/27/2017).

[12] DELL. “soporte DELL Poweredge 2960”. In: (2016). URL: http://www.dell.com/support/article/us/en/4/SLN297878/instalaci%C3%B3n-y-configuraci%C3 % B3n - de - raid - en - un - servidor - dell - poweredge ? lang = ES (visited on 10/24/2017).

[13] James Dickson et al. “Enabling portable I/O analysis of commercially sensitive HPC applications through workload replication”. In: Cray User Group 2017 Proceedings (2017).

[14] Ifeanyi Paulinus Egwutuoha. “A proactive fault tolerance framework for high performance computing (HPC) systems in the cloud”. In: (2013).

[15] Ana Gainaru et al. “Scheduling the I/O of HPC applications under congestion”. In: (2015), pages 1013–1022.

[16] B.a Guillerminet et al. “Gateway: New high performance computing facility for EFDA task force on integrated Tokamak modelling”. In: Fusion Engineering and Design 85.3-4 (2010), pages 410–414. ISSN: 09203796. DOI: 10.1016/j.fusengdes.2009.12.010. URL: https://www.scopus.com/inward/record.uri?eid=2- s2.0-78049324940&partnerID=40&md5=908d88b5a4633508b2dcbbc740f09169.

[17] Jaehyun Han, Deoksang Kim, and Hyeonsang Eom. “Improving the Performance of Lustre File System in HPC Environments”. In: (2016). DOI: 10 . 1109 / FAS -W.2016.29.

[18] Tapas Kanungo et al. “An efficient k-means clustering algorithm: Analysis and implementation”. In: IEEE transactions on pattern analysis and machine intelligence 24.7 (2002), pages 881–892.

[19] Youngjae Kim and Raghul Gunasekaran. “Understanding I/O workload characteristics of a Peta-scale storage system”. In: The Journal of Supercomputing 71.3 (2015) pages 761–780.

[20] Ibad Kureshi. “Establishing a University Grid for HPC Applications”. In: (2010).

[21] Riccardo Lancellotti. “A Scalable Monitor for Large Systems”. In: 512 (2016), page 100.

[22] Top 500 List. “https://www.top500.org/”. In: (2017). URL: https://www.top500.org/ (visited on 03/24/2017).

[23] Lustre. “Lustre manual”. In: (2017). URL: https://www.lustre.org/ (visited on 03/24/2017).

[24] Avantika Mathur et al. “The new ext4 filesystem: current status and future plans”. In: 2 (2007), pages 21–33.

[25] Ingo Mierswa. “The Wisdom of Crowds: Best Practices for Data Prep & Machine Learning Derived from Millions of Data Science Workflows”. In: (2016), pages 411.

[26] Oracle. “Virtual Box”. In: (2016). URL: http://www.virtualbox.org/ (visited on 08/24/2016).

[27] Philip M Papadopoulos, Mason J Katz, and Greg Bruno. “NPACI Rocks: Tools and techniques for easily deploying manageable linux clusters”. In: Concurrency and Computation: Practice and Experience 15.7-8 (2003), pages 707–725.

[28] T Kling Petersen and John Fragalla. “Optimizing performance of HPC storage systems”. In: (2013).

[29] Hernández Sampieri Roberto, Fernández Collado Carlos, and Baptista Lucio Pilar. “Metodología de la Investigación”. In: (2006).

[30] Rocks. “Rocks cluter”. In: (2016). URL: http://www.rocksclusters.org (visited on 11/04/2016).

[31] Rushikesh Salunkhe et al. “In Search of a Scalable File System State-of-the-art File Systems Review and Map view of new Scalable File system.” In: ().

[32] Mahdi Soltanolkotabi. “Algorithms and theory for clustering and nonconvex quadratic programming”. In: (2014).

[33] Brent Welch and Geoffrey Noer. “Optimizing a hybrid SSD/HDD HPC storage system based on file size distributions”. In: (2013), pages 1–12.

[34] Yushu Yao and Phil Carns. “Darshan Introduction”.