Manejo de Memoria para sistemas HPC

MSc. Computer Science
ITAM
Mexico City, Mexico
salvador.garcia.gonzalez@itam.mx

Salvador Garcia

***Abstract*—El Cómputo de Alto Rendimiento (HPC) utiliza la máxima potencia de un sistema de cómputo para resolver cálculos que permitan la solución de problemas tradicionales en distintas áreas de la ciencia. Este máximo aprovechamiento necesita de optimizaciones dentro de la gestión de memoria, planeación de procesos, entre otros aspectos de los Sistemas Operativos. La optimización no solo se realiza a nivel software, también busca optimizaciones a nivel arquitectura dentro de los distintos componentes del hardware, como la memoria, el caché, el procesamiento y demás componentes. Dentro del Survey se indaga más a profundidad los problemas y optimizaciones realizadas en especifico en el manejo de memoria a nivel software y hardware enfocado a los sistemas que realizan HPC. Aspectos como patrones de acceso, migración, localidad de memoria, particiones, ancho de banda de memoria son optimizados para poder dar el mejor resultado posible.**

***Index Terms*—Memory management, Memory optimization, High Performance Computing**

I. Introducción

El surgimiento de nuevas cantidades de información y datos en las distintas industrias y áreas de la ciencia ha llevado a requerir mayor capacidad de cómputo. Una evidencia de esta necesidad se indica en el International Exascale Project Roadmap [23], donde presentan los desafíos del cómputo de alto rendimiento para las próximas décadas. La alternativa intuitiva para aumentar el poder de cómputo es generar una arquitectura donde se incrementa el numero de ciclos dentro del núcleo. El reto que se tiene es que a mayor rendimiento se requiere un mayor consumo de energía, tanto para hacer funcionar el chip, como para disipar el calor dentro del chip, además de otras limitantes [24]. Ante este escenario, el HPC ha cambiado el paradigma de supercomputadoras con un solo núcleo a un paradigmas de clusters y grids de cómputo con varios núcleos trabajando en conjunto.

Dentro de este nuevo paradigma, no todos los nodos se busca que sean homogéneos entre si, ahora se permite y prefiere que exista heterogeneidad que permita que cada nodo cuente con su propia especialización [2]. Esta heterogeneidad es necesaria ya que la especialización de los nodos trae un mayor rendimiento global del sistema, con un costo de manejo de recursos más complejo. La motivación de este Survey es explorar distintas mejoras de uno de esos recursos limitados: la memoria. En particular, uno de los aspectos a profundizar es la localización de los datos, el cual es un punto crítico dentro del desarrollo de software. Este uso eficiente de la memoria involucra el uso manejo de distintos tipos de jerarquía de memoria, la cual ha ido complicándose conforme avanza el desarrollo en el área.

En la sección 2 se da un breve resumen de la historia del HPC de acuerdo a las distintas clases de arquitecturas y sistemas operativos, incluyendo sistemas con núcleos ligeros y sistemas con núcleos más complejos compatibles con Linux. En la sección 3 se presentan los listados de supercomputadoras más potentes que se utilizan actualmente, así como las métricas de rendimiento empleadas dentro de estas listas. En la sección 4 se introduce el manejo de memoria para los sistemas HPC, que es similar a la gestión de un sistema de cómputo tradicional, pero con ciertas optimizaciones en el patrón de acceso, localización de memoria, migración de memoria, etc. Por último, en la última sección se presentan áreas de investigación y áreas de oportunidad en esta área.

II. Historia de HPC y Métricas de Rendimiento

El objetivo del HPC es muy similar desde su primera concepción. La idea general es emplear tecnología de punta en el hardware para poder generar el máximo rendimiento posible en términos de cómputo [2]. Lo que ha ido cambiando durante las últimas décadas son los distintos paradigmas de los sistemas; es decir, la arquitectura y los Sistemas Operativos que orquestan el hardware. En el último mencionado, uno de los mayores retos es que el Sistema Operativo logre explotar la máxima capacidad del diseño del hardware [2]. En esta sección se presentan los principales hitos que se han tenido dentro del HPC, desde los primeros sistemas desarrollados que cuentan con núcleos pequeños y eficientes, evolucionando a sistemas compatibles con Linux, hasta llegar a sistemas mixtos o bien con distintos Sistemas Operativos.

A. Historia del diseño de los HPC

El Sistema Operativo de los sistemas HPC busca controlar los recursos del hardware empleado para ofrecer un máximo rendimiento, el cual es una tarea compleja que presenta un sacrificio entre el máximo posible rendimiento alcanzado y la facilidad de uso y la productividad de los usuarios. En esta subsección se describen tres clases de sistemas diseñados durante las últimas décadas, desde sistemas con procesadores vectoriales hasta sistemas distribuidos que buscan una paralelización de los procesos [2].

Los sistemas desarrollados a lo largo de las últimas décadas han ido variando entre núcleos pequeños y sistemas más complejos, siendo **la primer clase** la que surgió en la década de 1990 e inicios del siglo XXI que son sistemas con núcleos que buscan la máxima eficiencia posible (LWK por sus siglas en inglés Light Weight Kernel). Algunos ejemplos de estos núcleos pequeños son SUNMOS [25], Puma [26] y Catamount [27]. Estos sistemas buscan una funcionalidad mínima, pero al ser desarrollados de la manera más eficiente posible, no se busca una compatibilidad e interoperabilidad entre ellos, por lo que se vuelve una solución poco escalable y costosa cuando hablamos de grids de cómputo. Esta falta de compatibilidad va en contra de lo propuesto por Linux, en donde se busca una estandarización como POSIX ((Portable Operating System Interface for Unix)). Adicional, la compatibilidad con Linux es importante ya que desde ese momento el Sistema Operativo iba ganando mercado (imagen 1). Por este motivo los sistemas LWK fueron cayendo en desuso para dar paso a más sistemas más complejos (FWK por sus siglas en ingles Full Weight Kernel), de manera que todas las supercomputadoras listadas dentro del top500 (listado que ordena por rendimiento a las 500 supercomputadoras) tienen como sistema operativo uno de la familia basada en Linux [28].

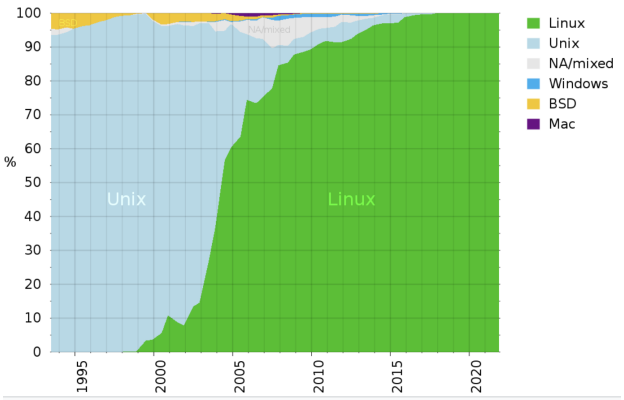

Fig. 1. Evolución de uso de los sistemas operativos en sistemas HPC [1]

La segunda clase de sistemas HPC aprovechó el desarrollo de máquinas con Massively Parallel Processos o MPPs y el surgimiento de POSIX. A través de la estandarización POSIX, la manera en como se envían y reciben los mensajes entre los sistemas se podía realizar de manera portable, con latencia baja y un ancho de banda más alto [35]. En cuanto al procesamiento, se comenzaron a aprovechar familias de procesadores de 32 y 64 bits, observando que desde 2006 se comienza a emplear x86-64 Intel como predominante en los procesadores de los sistemas HPC dentro del top500. Otros como x86-64 AMD, POWER, x86-32 Intel y x86-32 AMD también cuentan con un uso significativo (imagen 2). Ejemplos de esta segunda clase son sistemas como Cray's Compute Node Linux [20], Argo [21] y K Computer [22]

Esta segunda clase plantea aprovechar las ventajas de compatibilidad de Linux al mismo tiempo que se tiene un sistema

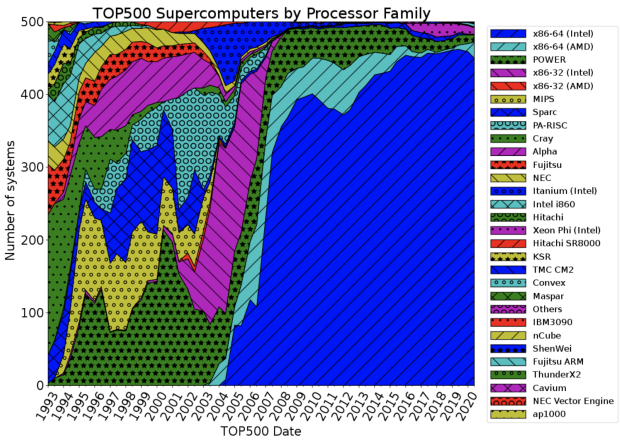

Fig. 2. Familia de procesadores en los SO de los sistemas HPC del Top500 [28]

con buen rendimiento. Siguiendo la idea de los LWK y los FWK, se puede hacer que Linux funcione más como un LWK, o bien que los LWK funcionen como Linux, lo que brinda todo un rango de posibilidades de optimización y compatibilidad [2]. El hacer que Linux actúe más como un LWK busca reducir al mínimo los *daemons* y otros componentes que impactan en el rendimiento del sistema, para no mermar la capacidad total del procesamiento para una aplicación. Adicional a esta optimización se pueden realizar optimizaciones directas sobre la calendarización y la gestión de memoria, ya que los sistemas HPC comúnmente tratan con problemas en lotes que consumen memoria a grandes cantidades [2]. Por este motivo, el interés del Survey se centra en las posibles modificaciones de la gestión de memoria se pueden realizar.

La tercer clase es una alternativa mixta. Se formula a través de un LWK con kernel de Linux para atender a las peticiones de sistema, de manera que pueda atenderlas lo más rápidamente posible. Por otra parte se tienen varios nodos de procesamiento FWK que permitan compatibilidad entre ellos y con estandarizaciones como POSIX [2]. Cuando se tiene una arquitectura de este estilo es común esta diferenciación de nodos, uno o varios dedicados a manejar el sistema, otros para el desarrollo del usuario, otros para Input/Output y algunos nodos de cómputo que se encargan de efectuar la carga de trabajo necesaria para cumplir la tarea propuesta. A nivel hardware, esta alternativa multi-núcleo busca explotar el escalamiento horizontal dentro de cada chip [2], en donde se cuenta con varios nodos FWK que permiten la interoperabilidad entre ellos.

Como se mencionaba previamente, en los últimos años Linux es el principal Sistema Operativo de los sistemas HPC, pero si se observa el mercado total de Linux, los HPC solamente representan una pequeña proporción cuando se consideran todos los Sistemas Operativos basados en Linux usados para otras aplicaciones (dispositivos móviles, dispositivos de cómputo personal, servidores Web, etc). Esto se debe principalmente a que Linux al buscar estandarización

y al ser un proyecto de colaboración masiva, no se adapta rápidamente a las nuevas tendencias de la comunidad HPC. Por este motivo, se ha buscado combinar distintos núcleos de Sistemas Operativos en el mismo sistema HPC cuando existen suficientes núcleos. Ejemplos de estos son FusedOS de IBM [17], Hobbes de Sandia National Laboratories [18] y NixOS [19].

III. Listados de sistemas HPC y métricas de rendimiento

A. Métricas de Rendimiento y Benchmarks

Una pregunta relevante en el área de HPC es como medir de manera justa y estandarizada los sistemas. Esta pregunta se realizó poco después del surgimiento de esta área de investigación cuando en 1979 se publicó el Benchmark High Performance LINPACK [4] que busca medir el poder de cómputo con operaciones de punto flotante. En particular este benchmark busca resolver un sistema denso de ecuaciones lineales [4] y es actualmente el más empleado para medir el rendimiento, de hecho es usado para el crear el listado más relevante de supercomputadoras llamado Top 500. Además de este benchmark existen otros, como el High Performance Conjugate Gradient (HPCG) [29] que evalúa el en situaciones de aplicaciones actuales, con matrices ralas, limitaciones de memoria y limitaciones de interconexión.

Las métricas calculadas derivados empleando estos benchmarks son el rendimiento pico (RPeak) o máximo del sistema medido en términos de la frecuencia de la máquina por segundo; y la segunda, el rendimiento alcanzado en el benchmark, el se llama rendimiento máximo (RMax). Lo común es que dependiendo de la aplicación, el rendimiento pico es menor al rendimiento teórico, ya que como se verá más adelante toma rangos máximos de 80% del teórico en los sistemas consultados.

B. Listados de Sistemas HPC

Desde 1993 se propuso una lista del Top 500 de sistemas que presentan el mejor rendimiento en términos de Mflops/s (tasa de ejecución en término de millones de operaciones de punto flotante por segundo) usando el High Performance LINPACK. En este benchmark se realiza de manera bianual y se puede observar que el Rmax ha ido incrementando desde 60GFlops en sus inicios hasta 1,102 PFlops en la actualidad, rendimiento alcanzado por el sistema Frontier que fue elaborado por HPE (Hewlett Packard Enterprise) en conjunto con AMD y el Laboratorio Nacional Oak Ridge (patrocinado por el Departamento de Energía de Estados Unidos) [30]. En el sistema Frontier, el Rpeak alcanza 1,685 PFlops teóricos (solo un 65% del Rmax). Para poder lograr este rendimiento, las arquitecturas empleadas usan núcleos aceleradores heterogéneos, es decir que tengan núcleos de propósito general y aceleradores específicos para los aplicativos [8].

Adicional al listado de Top 500, existen otros dos listados relevantes que buscan medir el performance de los HPC. El primero de ellos ha tomado relevancia debido al alto consumo energético por parte de los sistemas HPC, derivado de esto el listado recibe el nombre de Green500, y se realiza también de manera bianual [7]. Este listado toma en cuenta no solo el rendimiento computacional usando el Benchmark High Performance LINPACK, si no también la eficiencia energética en términos de Rendimiento por Watt. De nuevo el sistema Frontier encabeza en lugar número 2 esta lista, con 52.227 GFlops/watt [7].

El segundo listado adicional al Top 500 toma un approach distinto a este y al Green 500 ya que el High Performance LINPACK considera situaciones ideales con matrices densas donde se puede explotar el máximo rendimiento de los sistemas, pero en aplicaciones recientes de Big Data, Machine Learning e Inteligencia Artificial existen operaciones matriciales que no son densas. Para esto, se buscó diseñar otro método para medir el performance en estas situaciones al cual se le dio el nombre de High Performance Conjugate Gradient (HPCG) benchmark. Esta prueba es más realista, pero al observar los resultados se genera una pequeña proporción del Rpeak de los sistemas. Para el sistema Frontier todavía no se tienen datos del rendimiento, pero para comparar los 10 sistemas que encabezan el listado, solo se alcanzan un Rmax el 2.696% del Rpeak cuando en comparación el Top 500, donde el Top 10 alcanza un Rmax el 74.35% del Rpeak.

El avance en el rendimiento medido es debido a varios factores, pero vale la pena compararlo con el incremento del número de transistores en un circuito integrado, donde según la Ley de Moore se duplica aproximadamente cada dos años. Lo interesante es que el crecimiento en rendimiento en los sistemas HPC sobrepasa el avance generado por el número de transistores, lo que se puede explicar debido a optimizaciones específicas en los Sistemas Operativos de los HPC y en el hardware 3. Más adelante se analizarán optimizaciones en la gestión de memoria para el Sistema Operativo más común, así como las arquitecturas empleadas.

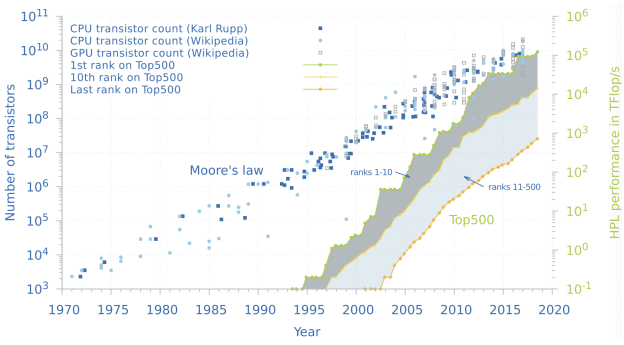

Fig. 3. Crecimiento del número de transistores comparado con el crecimiento del performance en sistemas HPC [2]

IV. Diseño General del manejo de memoria en sistemas HPC

A. Manejo de Memoria en sistemas HPC

Dentro de la clasificación de sistemas con diseños homogéneos y heterogéneos se tienen varios niveles de memoria (jerarquía de memoria). Comúnmente un Sistema Operativo usa técnicas de segmentación y paginación, pero al tener varias jerarquías de memoria se vuelve complicado encontrar la manera más eficiente de usar los tipos de memoria, al mismo tiempo que se busque minimizar el movimiento de los datos [36]. Dentro de esta jerarquía de memoria se cuenta con pocos Megabytes de memoria volátil caché (con distintos niveles), Gigabytes de velocidad media de memoria volátil, y Terabytes de almacenamiento de bajo costo, lento y no volátil. El Sistema Operativo debe abstraer esta jerarquía en un modelo que puede usar para manejar eficientemente la memoria, saber que está ocupado, donde se pueden almacenar datos cuando se necesitan y que datos dejar de usar, almacenar o mover cuando no se necesiten. Estas ultimas tareas se realizan por medio del manejador de memoria [2].

Adicional, en sistemas Grid, la lenta interconexión entre nodos es una dificultad para sistemas heterogéneos, ocasionando que el desarrollo de software sea complicado. Esta complejidad recae en que si se requiere un rendimiento óptimo, se requiere un manejo explicito de los niveles de memoria. Otros aspectos se deben considerar, como la planeación y las políticas de distribución, ya que para asignar la memoria, se debe considerar los requerimientos de los aplicativos, así como el estado del sistema en general. Por último los manejadores que se encargan de estos aspectos utilizan interfaces que permiten no solo gestionar la memoria, si no también brindar información al nodo pertinente y al compilador que le permita optimizar el uso de los recursos.

Múltiples optimizaciones se pueden realizan para la gestión de memoria para aplicaciones de HPC, algunas a nivel hardware y otras a nivel Sistema Operativo. Como primer ejemplo se analiza la memoria en el chip, la cual debe tener un diseño que minimice el movimiento de datos. Adicional a esto, se puede permitir que el Sistema Operativo controle el movimiento de los datos a núcleos para que existe un buen rendimiento. Entre ellas destacan la ScratchPad Memory (SPM), el uso de los memristors, o bien arquitecturas de memoria multicanal que buscan incrementar la tasa de transferencia entre la memoria DRAM y el controlador de memoria añadiendo más canales de comunicación.

En términos de Sistema Operativo, solamente se centrará en las optimizaciones basadas en manejo de memoria de Linux, ya que es el Sistema Operativo más empleado por sistemas HPC. En general Linux ofrece una aplicación general y un uso fácil para desarrolladores; sin embargo, para lograr optimizaciones adicionales relativas a arquitecturas complejas y heterogéneas, se requieren funcionales adicionales. A continuación se exploran distintos aspectos del Sistema Operativo y modificaciones que permiten lograr un mejor rendimiento para las aplicaciones. Entre ellas se exploran los patrones de acceso, el proceso de migración de memoria, la localidad de memoria y las particiones de memoria.

B. Taxonomía

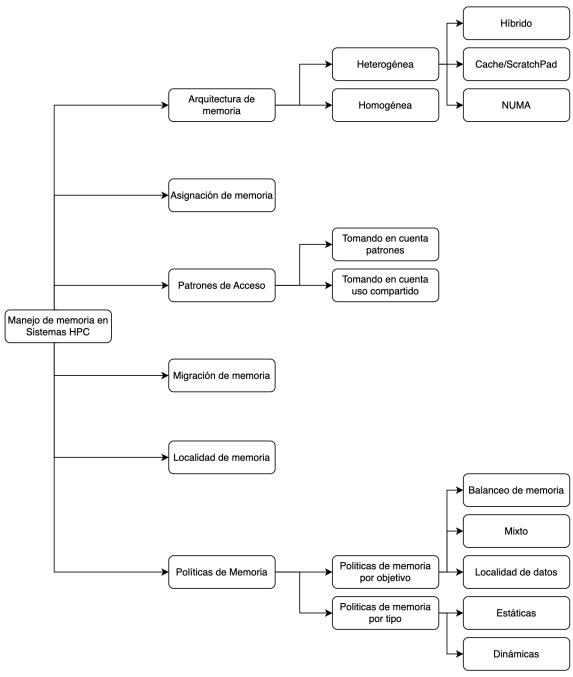

Fig. 4. Taxonomía de manejo de memoria en sistemas HPC

C. ScratchPad Memory (SPM)

Una primera optimización que se puede realizar consiste en contar con una memoria interna de alta velocidad en cada núcleo que permita almacenar cálculos o algunos datos temporales. Esta memoria interna requiere que físicamente se encuentre cercano a la ALU o bien a los núcleos. La ScratchPad Memory (SPM) [9] introducida en 2002 es un sistema que entra dentro de la clasificación NUMA, teniendo latencias similares a estos sistemas. Con esta memoria se busca resolver el problema de selección de memoria en chip para aplicaciones con un requerimiento alto de cómputo. Esta SPM es manejada directamente por los programas, explícitamente pueden acceder y escribir o leer los datos que almacena, teniendo como consecuencia un rendimiento determinístico contrario a la caché. Los resultados obtenidos de esta alternativa es que se reduce el consumo de energía en un 40%, esto debido a que la memoria en chip ocupa más del 50% del área, al reducir esta área implica que se reduce el número de transistores y por consiguiente, se obtiene una reducción de energía [9].

D. Memristors y Resistive Random Access Memory

Actualmente al utilizar Random Access Memory (RAM) se tiene una memoria que accede y almacena a una tasa alta, pero ante una falla energética los datos no son guardados, es decir es volátil. En cambio una memoria no volátil tiene una tasa de almacenamiento y lectura más baja pero permite que ante una falla energética los datos sigan almacenados. Con el surgimiento de las implementaciones de los *memristors* en [10], que soporta la Resistive Random Access Memory (ReRAM) se puede contar con una memoria rápida y no volátil. Además se cuentan con ventajas como requerir menos energía para funcionar y presentan un mayor rendimiento en términos de espacio, ya que tienen una mayor densidad de área con respecto a la RAM estática.

E. Patrones de acceso

Los patrones de acceso a memoria nos indica la forma en que el Sistema Operativo accede a la memoria disponible, esto se realiza tomando en cuenta las distintas latencias de lectura y escritura de datos en la DRAM y SRAM. Para esto existen distintas optimizaciones que se pueden lograr como PAMS [8] y SAMMU [13] que se presentan a continuación.

E1: Pattern Aware Memory System (PAMS)

La primer alternativa para optimizaciones es una que considere los patrones utilizados al acceder a la memoria, para esto la propuesta de Patter Aware Memory System (PAMS) busca acelerar la transferencia de datos entre las estructuras de datos ordenando los accesos con un descriptor de memoria 5. El descriptor de memoria gestiona las estructuras de datos usando bloques y organizando los accesos en patrones de manera que se evite acceder a los datos múltiples veces. Como consecuencia se evita la transferencia de datos, mejorando en rendimiento y consumo energético. Como segundo paso, calendariza los accesos de acuerdo a las necesidades de la aplicación [8]. Otro ejemplo de sistema que considera patrones es brindado por [15] en donde se tiene un filtro de bloom que accede a una detección de patrón, esto en conjunto con un criterio para determinar cuando mover los datos. Con esta optimización se logra una mejora de 41% contra sistemas tradicionales de caché vía hardware [15].

E2: Sharing Aware Memomy Management (SAMMU)

En cambio existen otras alternativas que analizan el comportamiento de acceso a memoria a nivel hardware. Uno de estos sistemas es el Sharing Aware Memomy Management (SAMMU) que analiza el comportamiento de acceso de memoria a nivel hardware [13]. Adicional, se plantea el mapeo de hilos y datos para incrementar la localidad de la memoria y así mejorar el rendimiento y disminuir la energía consumida. Con esta alternativa se busca lograr que el acceso a los datos este cerca en la jerarquía de memoria. SAMMU analiza el acceso a la memoria de los datos que están mapeados a su nodo NUMA. El rendimiento con esta alternativa plantea mejoras del 13.1% en promedio. [13].

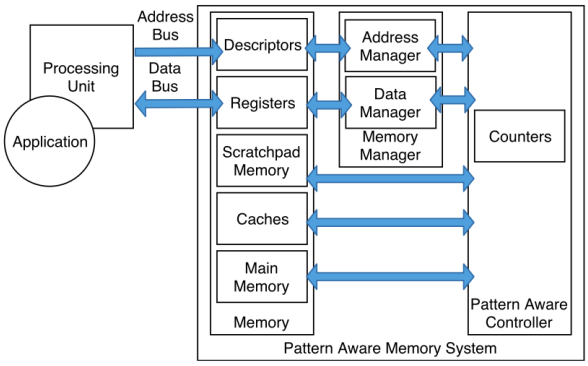

Fig. 5. Arquitectura del Pattern Aware Memory System [2]

Modelos generales de estos sistemas que buscan ser conscientes del patrón de acceso, se presenta en [1].

F. Migración de memoria

La idea en la optimización en migración de memoria es migrar las tareas junto con las páginas de memoria junto que utiliza a otro nodo, en caso contrario se puede incurrir en una ineficiencia al intentar acceder remotamente a la memoria. Al igual que las alternativas pasadas, esta migración puede ser manejada por el Sistema Operativo o directamente por el hardware. La desventaja es que esta migración tiene un costo computacional adicional, en especial cuando se migran datos para sistemas multi-núcleo con jerarquías de memoria complejas.

F1: Utility-Based Hybrid Memory (UH-MEM)

Existen alternativas que evalúan el beneficio esperado de la aplicación y del sistema completo ante una migración de página, una de ellas es la Utility-Based Hybrid Memory (UH-MEM) [32]. La mayor complejidad de esta alternativa en NUMA es que los datos se almacenan en páginas, al migrar una página se migran todos los datos de la página, lo que puede ocasionar que exista acceso remoto de otro proceso en lugar de acceso local. Para esto, existen soluciones propuestas que ayudan a mantener cercanía entre datos que se usan en conjunto [33].

Adicional a determinar como priorizar la migración de páginas de acuerdo a su utilidad, se tienen optimizaciones en los mecanismos de migración de memoria. De la misma manera se tienen dos alternativas, gestionadas por el Sistema Operativo o gestionada por el hardware. Para gestión por Sistema Operativo se han realizado benchmarks como los presentados en [14], donde se exploran distintas configuraciones de hardware incluyendo sistemas NUMA y se evalúan mecanismos disponibles en Linux como *move_pages* y *mem_cpy*. Adicional se presentan los resultados de como la localidad de memoria y el paralelismo pueden ser mejorados a través de la migración de datos [14]. Alternativas basadas en hardware se pueden ver en [34], donde se exploran los Assymetric

Multicore Processors systems (AMPs) bajo distintas pruebas entre ellas migración de datos.

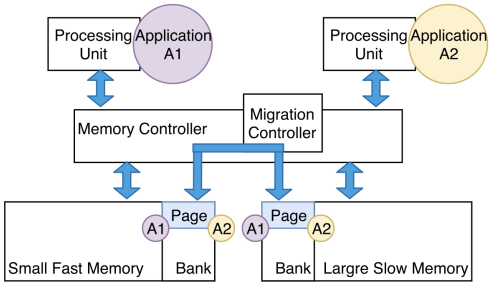

Fig. 6. Arquitectura del Pattern Aware Memory System [2]

G. Localidad de memoria

Para localidad de memoria a nivel Sistema Operativo existen alternativas para Linux como *hwlock* [37], donde se busca tener una abstracción de la jerarquía de memoria (aunque también permite de los núcleos, hilos, etc). Esta librería se explora en [16]. Otra herramienta que permite encontrar y exponer los recursos es *hexe* que divide la asignación de memoria en tres partes: [39]

- Se asigna la dirección virtual
- Se mapea la dirección virtual a una dirección física
- se mapea la dirección virtual a una página exacta de dirección física que se usará por el sistema operativo

La idea de estas propuestas es que el rendimiento se mejora cuando se maneja explícitamente la localidad de la memora, esto es una propuesta que se ha vuelto altamente sugerida en las nuevas tecnologías de memoria en sistemas HPC [1]. En estas librerías presentadas, la asignación de memoria requiere conocimiento de la arquitectura de jerarquía de memoria que se tiene en el sistema, de manera que se logre una mejora teniendo un manejo explicito en lugar de uno automático [1]. En la siguiente imagen se puede observar un esquema general de la localidad de memoria en HPC:

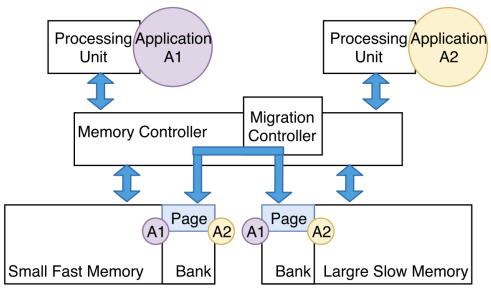

Fig. 7. Localidad de memoria en HPC [2]

H. Políticas

Para las políticas se ha mostrado que las basadas en localidad tienen un mejor rendimiento que las basadas en balanceo de memoria [1]. Una tercer clase se puede tener, en donde se tenga un sacrificio entre la localidad y el balanceo de memoria [39]. El asignar los datos basados en rendimiento puede resultar una confiabilidad del sistema menor, por esto en este documento se plantean distintas maneras de asignar los datos de una manera estática y dinámica de acuerdo a que tan usado son las páginas de memoria.

V. Critica y áreas de investigación

Una área de oportunidad dentro de los sistemas HPC es dentro de los benchmarks que se toman en cuenta para su medición. Los principales usados para estos no consideran aspectos como tolerancia a fallas; es decir, si falla un nodo como se evita la pérdida de información. En este aspecto hay algunas investigaciones que involucran puntos de control [6] y otras que no tienen puntos de control [4]. Adicionalmente, pruebas en escenarios más realistas se pueden generar, en donde se prueben para aspectos computacionales actuales donde se ocupa alta capacidad de cómputo, como en Artificial Inteligence, Machine Learning y Deep Learning.

Dentro de las áreas de investigación que se exploran distintas alternativas, entre ellas los contenedores para sistemas heterogéneos que tengan distintos sistemas operativos, pero algunos de ellos en común. En parte para aislar ejecuciones especializadas de acuerdo al tipo de nodo empleado.

Dentro del área de manejo de memoria, un área de interés que puede considerarse es que existen mejoras a nivel hardware, pero Linux al ser colaborativo no necesariamente sigue el mismo ritmo de las actualizaciones. El software requiere soporte adicional para nuevas tecnologías, algunas mencionadas dentro de los papers son High-Band Memory (HBM), Resistive Random Access Memory (RRAM), así como mejoras en el control del ancho de banda y de la latencia en términos de comunicación inter-nodo, persistencia y encriptación.

REFERENCES

[1] .Pupykina, Anna, and Giovanni Agosta. "Survey of memory management techniques for hpc and cloud computing." IEEE Access 7 (2019): 167351-167373.

[2] Gerofi, Balazs, et al., eds. Operating Systems for Supercomputers and High Performance Computing. Vol. 1. Singapore city, Singapore: Springer, 2019.

[3] Asch, Mark, et al. "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry." The International Journal of High Performance Computing Applications 32.4 (2018): 435-479.

[4] Davies, Teresa, et al. "High performance linpack benchmark: a fault tolerant implementation without checkpointing." Proceedings of the international conference on Supercomputing. 2011.

[5] Wong, Kenneth F., and Mark Franklin. "Checkpointing in distributed computing systems." Journal of parallel and distributed computing 35.1 (1996): 67-75.

[6] Dongarra, Jack J., Piotr Luszczek, and Antoine Petitet. "The LINPACK benchmark: past, present and future." Concurrency and Computation: practice and experience 15.9 (2003): 803-820.

[7] Feng, Wu-chun, and Kirk Cameron. "The green500 list: Encouraging sustainable supercomputing." Computer 40.12 (2007): 50-55.

[8] Hussain, Tassadaq. "A novel hardware support for heterogeneous multicore memory system." Journal of Parallel and Distributed Computing 106 (2017): 31-49.

[9] Banakar, Rajeshwari, et al. "Scratchpad memory: A design alternative for cache on-chip memory in embedded systems." Proceedings of the Tenth International Symposium on Hardware/Software Codesign. CODES 2002 (IEEE Cat. No. 02TH8627). IEEE, 2002.

[10] Hartmann, Matthias, et al. "Memristor-based (reram) data memory architecture in asip design." 2013 Euromicro Conference on Digital System Design. IEEE, 2013.

[11] Schroeder, Bianca, and Garth A. Gibson. "A large-scale study of failures in high-performance computing systems." IEEE transactions on Dependable and Secure Computing 7.4 (2009): 337-350.

[12] Perissakis, Stylianos, et al. "Embedded DRAM for a reconfigurable array." 1999 Symposium on VLSI Circuits. Digest of Papers (IEEE Cat. No. 99CH36326). IEEE, 1999.

[13] Cruz, Eduardo HM, et al. "A sharing-aware memory management unit for online mapping in multi-core architectures." European Conference on Parallel Processing. Springer, Cham, 2016.

[14] Perarnau, Swann, et al. "Exploring data migration for future deep-memory many-core systems." 2016 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2016.

[15] Arima, Eishi, and Martin Schulz. "Pattern-aware staging for hybrid memory systems." International Conference on High Performance Computing. Springer, Cham, 2020.

[16] Goglin, Brice. "Exposing the locality of heterogeneous memory architectures to HPC applications." Proceedings of the Second International Symposium on Memory Systems. 2016.

[17] Park, Yoonho, et al. "FusedOS: Fusing LWK performance with FWK functionality in a heterogeneous environment." 2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing. IEEE, 2012.

[18] Brightwell, Ron, et al. "Hobbes: Composition and virtualization as the foundations of an extreme-scale OS/R." Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers. 2013.

[19] Dolstra, Eelco, and Andres Löh. "NixOS: A purely functional Linux distribution." Proceedings of the 13th ACM SIGPLAN international conference on Functional programming. 2008.

[20] Wallace, David. "Compute Node Linux: Overview, progress to date, and roadmap." Proceedings of the 2007 Cray User Group Annual Technical Conference. 2007.

[21] Perarnau, Swann, et al. "Distributed monitoring and management of exascale systems in the Argo project." IFIP International Conference on Distributed Applications and Interoperable Systems. Springer, Cham, 2015.

[22] Ajima, Yuichiro, et al. "The K Computer and Beyond." Transition of HPC Towards Exascale Computing 24 (2013): 3.

[23] Dongarra, Jack, et al. "The international exascale software project roadmap." The international journal of high performance computing applications 25.1 (2011): 3-60.

[24] Hennessy, John L., and David A. Patterson. Computer architecture: a quantitative approach. Elsevier, 2011.

[25] Maccabe, Arthur B., et al. SUNMOS for the Intel Paragon-a brief users guide. No. SAND-94-1583C; CONF-9406205-2. Sandia

[26] National Labs., Albuquerque, NM (United States), 1994. Wheat, Stephen R., et al. "PUMA: An operating system for massively parallel systems." Scientific Programming 3.4 (1994): 275-288.

[27] Kelly, Suzanne M., and Ron Brightwell. "Software architecture of the light weight kernel, Catamount." Proceedings of the 2005 Cray User Group Annual Technical Conference. 2005.

[28] Dongarra, Jack J., Hans W. Meuer, and Erich Strohmaier. "TOP500 supercomputer sites." Supercomputer 13 (1997): 89-111.

[29] Dongarra, Jack, Michael A. Heroux, and Piotr Luszczek. "High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems." The International Journal of High Performance Computing Applications 30.1 (2016): 3-10.

[30] Schneider, David. "The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second." IEEE Spectrum 59.1 (2022): 34-35.

[31] Soltesz, Stephen, et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." Proceedings of the 2Nd ACM SIGOPS/EuroSys european conference on computer systems 2007. 2007.

[32] Y. Li, S. Ghose, J. Choi, J. Sun, H. Wang, and O. Mutlu, "Utility-based hybrid memory management," in Proc. IEEE Int. Conf. Cluster Comput. (CLUSTER), Sep. 2017, pp. 152–165. [Online]. Available: http://ieeexplore.ieee.org/document/8048927/

[33] S. Williams, L. Ionkov, M. Lang, and J. Lee, "Heterogeneous memory and arena-based heap allocation," in Proc. Workshop Memory Centric High Perform. Comput. (MCHPC). New York, NY, USA: ACM, 2018, pp. 67–71. [Online]. Available: http://doi.acm.org/10.1145/3286475.3286568

[34] J.-H. Ding, Y.-T. Chang, Z.-D. Guo, K.-C. Li, and Y.-C. Chung, "An efficient and comprehensive scheduler on asymmetric multicore architecture systems," J. Syst. Archit., vol. 60, no. 3, pp. 305–314, Mar. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762113000738

[35] Walli, Stephen R. "The POSIX family of standards." StandardView 3.1 (1995): 11-17.

[36] Tanenbaum, Andrew. Modern operating systems. Pearson Education, Inc.,, 2009.

[37] Goglin, Brice. "Managing the topology of heterogeneous cluster nodes with hardware locality (hwloc)." 2014 International Conference on High Performance Computing Simulation (HPCS). IEEE, 2014.

[38] Oden, Lena, and Pavan Balaji. "Hexe: A toolkit for heterogeneous memory management." 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2017.

[39] Gupta, Manish, et al. "Reliability-aware data placement for heterogeneous memory architecture." 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2018.