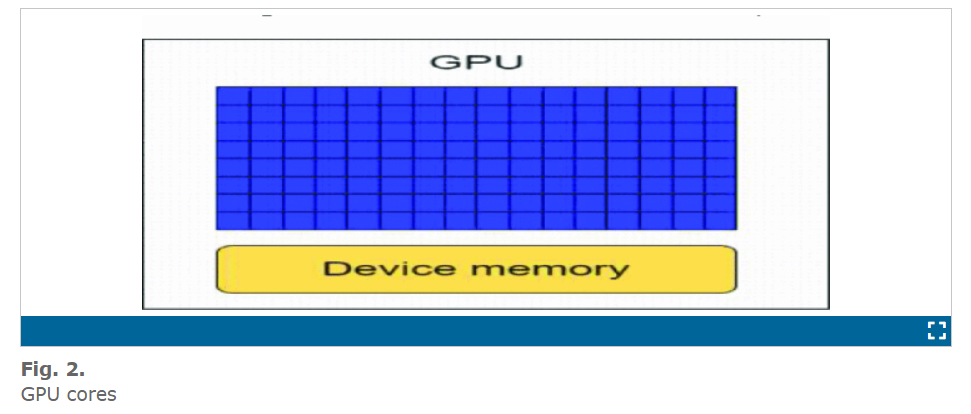
Arquitectura del HPC a nivel de GPU

Small intro about gpus in hps

Hoy en día, el Cómputo de Alto Rendimiento (HPC) requiere el despliegue de es un campo de estudio que siempre está avanzando, pero limitado en ciertos aspectos físicos. Por ejemplo, dado a los núcleos limitados que están presentes en los CPUs, éstas no logran un mayor rendimiento en las aplicaciones HPC. Sin embargo, las GPU pueden proporcionar un rendimiento impresionante en HPC gracias a la disponibilidad de miles de núcleos.

Explanation in gpu

Actualmente conocemos que la Unidad de Procesamiento Gráfico (GPU) es muy popular en el mundo del videojuego. Las GPU se les conocen más como tarjetas gráficas o tarjetas de vídeo. Las tarjetas gráficas se utilizan para obtener imágenes 2D o 3D de alta calidad en plataformas de juego. En uso en computación paralela, la GPU es conocida por procesar y ejecutar cálculos más complejos en menos tiempo que la CPU, gracias a los cientos de núcleos que lleva incorporados. Estos núcleos tienen un número «n» de subprocesos que realizan cálculos complejos simultáneamente [1].



[1]

Arquitectura de la GPU y modelo de ejecución

Una GPU tiene unidades de procesamiento llamadas multiprocesadores de streaming (SM), y cada uno de estos SM tiene varios núcleos, su propia memoria privada llamada archivo de registros, una caché L1, una memoria compartida que los núcleos pueden usar en conjunto y una caché de solo lectura. Las GPUs siguen un modelo llamado SPMD (Single Program, Multiple Data), donde todos los hilos (o threads) ejecutan el mismo programa o kernel, pero con diferentes datos.

Los hilos que ejecutan estos kernels están organizados en una especie de rejilla (grid), y esta rejilla se divide en grupos más pequeños llamados bloques de subprocesos (thread blocks). Luego, estos bloques se dividen en unidades aún más pequeñas llamadas warps, que son conjuntos de hilos que ejecutan la misma instrucción al mismo tiempo bajo el modelo SIMD (Single Instruction Multiple Data).

Dentro de la GPU, los archivos de registros son una memoria que solo pueden usar los hilos individuales, mientras que las otras memorias, como la caché L1 o la memoria compartida, son usadas por todos los hilos del SM. Para acceder a la memoria global (la memoria principal de la GPU), los SM lo hacen a través de transacciones de gran tamaño, que tienen un gran ancho de banda pero también alta latencia. Por eso, las transacciones más grandes ayudan a reducir la cantidad de operaciones en la memoria DRAM.

Cuando los hilos piden datos de memoria y sus direcciones están cerca unas de otras, la GPU puede agrupar esas solicitudes en una sola transacción. Esto se llama coalescencia de memoria y ayuda a que la GPU sea más eficiente al reducir el número de accesos a la memoria.

Finalmente, en el modelo de ejecución acelerada de la GPU, las tareas que requieren mucho tiempo de procesamiento se envían a la GPU para que las procese rápidamente. Esto se hace descargando los kernels (código a ejecutar) en la GPU. El CPU, o host, es el encargado de transferir los datos entre la memoria del CPU (memoria del host) y la memoria de la GPU (memoria del dispositivo), y también de invocar los kernels para que se ejecuten en la GPU. [2]

Aplications in software llike cuda nvdia

En el campo del cómputo de alto rendimiento (HPC), las GPUs juegan un rol crucial debido a su capacidad para acelerar aplicaciones complejas y manejar grandes volúmenes de datos en paralelo. El uso de GPUs se apoya principalmente en herramientas como CUDA, que permite a los desarrolladores aprovechar los núcleos de la GPU para ejecutar múltiples hilos de procesamiento de manera simultánea. CUDA es compatible con lenguajes de programación como C, C++ y Fortran, ofreciendo un entorno robusto para escribir aplicaciones que requieren un alto grado de paralelismo.

A nivel de software y compiladores, el entorno de programación en HPC se beneficia de varias herramientas especializadas. Por ejemplo, el **NVIDIA HPC SDK (NVHPC)** proporciona un conjunto completo de compiladores y bibliotecas optimizadas para trabajar con GPUs. Asimismo, **CUDA** es una herramienta fundamental, ya que permite el uso eficiente de los múltiples núcleos en una GPU para distribuir las tareas en bloques de hilos, maximizando el rendimiento paralelo.

Además de CUDA, hay otros modelos de programación paralela ampliamente utilizados como **MPI** (Message Passing Interface) y **OpenMP**, que permiten gestionar la comunicación entre los procesos que se ejecutan en diferentes nodos o dentro de una misma GPU. **OpenACC** es otra opción cuando se quiere simplificar el paralelismo de alto nivel, proporcionando directivas para facilitar la ejecución de código en la GPU sin tener que reescribir el programa desde cero.

En cuanto a las bibliotecas de software, herramientas como **HDF5** y **OpenBLAS** permiten manejar grandes volúmenes de datos y realizar cálculos matemáticos de manera eficiente. Estas bibliotecas están diseñadas para funcionar bien con arquitecturas de alto rendimiento, permitiendo el uso eficiente de los recursos de hardware.

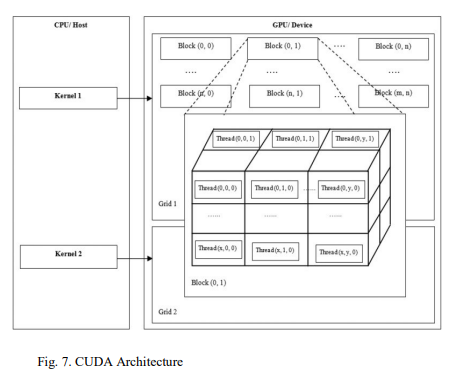
Otro aspecto clave del HPC es la utilización de herramientas de monitoreo y análisis de rendimiento. **NSight Compute SDK** y **Allinea Forge** permiten a los desarrolladores crear perfiles detallados de cómo sus aplicaciones están utilizando la GPU, identificando posibles cuellos de botella o áreas donde se puede optimizar el rendimiento. **Score-P**, por otro lado, es una herramienta que ayuda a medir el rendimiento de las aplicaciones, proporcionando métricas detalladas sobre el uso de recursos.

Para manejar múltiples dependencias y bibliotecas de software en un entorno de HPC, se utiliza **Spack**, un gestor de paquetes diseñado específicamente para aplicaciones científicas. Esto facilita la instalación y gestión de herramientas y librerías como **OpenMPI** para trabajos multinodo, asegurando que todo el software necesario esté configurado correctamente sin necesidad de ajustes manuales.

En resumen, las aplicaciones de HPC que aprovechan GPUs requieren un ecosistema variado de herramientas de software y metodologías de programación para maximizar el rendimiento. El uso de **CUDA**, junto con modelos de programación como **MPI**, **OpenMP**, y bibliotecas especializadas, permite ejecutar aplicaciones complejas en paralelo de manera eficiente. A su vez, las herramientas de análisis de rendimiento y gestión de dependencias garantizan que estas aplicaciones puedan optimizarse y ejecutarse en plataformas modernas sin problemas.

El despliegue de nuevos superordenadores requiere una evaluación continua de las nuevas plataformas y la comprensión de las ventajas y desventajas de portar las aplicaciones existentes a diferentes arquitecturas. Dado que muchos de los protagonistas de la tecnología HPC están creando aceleradores especializados o de uso general, cada vez es más importante conocer el nivel de inversión de tiempo humano necesario para que las aplicaciones estén listas para la producción en cualquiera de estas plataformas aceleradas, así como las ventajas de rendimiento que se espera obtener con este esfuerzo. [3 ]

NVIDIA is the distributor of CUDA. By harnessing the power of GPUs, we can speed up the execution of complex applications by CUDA. C, C++, and Fortran are used to write programs in CUDA. Apparently, CUDA consists of hundreds of cores, where each core acts as a multiprocessor. The total available threads will be arranged into blocks of threads. [1]

[1]

El entorno de programación y el software del sistema se han mantenido tal cual durante toda la evaluación (abril y mayo de 2022). Decidimos conscientemente no variar constantemente el entorno y crear una línea de base fija. Los nodos Wombat arrancan su sistema operativo desde la red, y todos los nodos están equipados con la misma imagen informática preconfigurada basada en CentOS 8.1 con kernel 4.18. El envío y la ejecución de los trabajos se orquestan de forma centralizada. El envío y la ejecución de trabajos se orquestan mediante SLURM. Los compiladores e intérpretes disponibles son NVIDIA HPC SDK (NVHPC) 22.1, Arm Compiler for HPC 22, CUDA 11.5.1, GNU 11.1, LLVM 15.0.0 con OpenMP activado, Python 3.9.0 y Julia 1.7.0. [3]

El soporte de red es proporcionado por OFED 5.4 y UCX 1.11.1 y aunque la mayoría de los experimentos son de nodo único, OpenMPI 4.1.2a1 está instalado para trabajos multinodo. NSight Compute SDK, Allinea Forge y Score-P están disponibles para la creación de perfiles.

Utilizamos Spack para librerías y herramientas científicas adicionales de terceros, incluyendo por ejemplo HDF5, OpenBLAS, y Score-P. No modificamos manualmente ningún indicador de optimización del compilador utilizado por Spack, con el objetivo de obtener una experiencia «lista para usar» sin filtros. Los paquetes que no contaban con recetas de Spack se instalaron individualmente.

Cada equipo de aplicación se encargó de compilar su aplicación respectiva, instalar las dependencias adicionales y vincular las bibliotecas adecuadas. [3]

Por definición, cualquier banco de pruebas puede carecer de algunas características que se encuentran en los sistemas de producción finales. Este hecho debe tenerse en cuenta al analizar los resultados de rendimiento obtenidos. A efectos de esta evaluación, la puntuación de rendimiento más comúnmente utilizada en HPC, el tiempo hasta la solución, no es la principal figura de mérito. En lugar de profundizar en las características de rendimiento de cada aplicación, realizamos un estudio general para evaluar la preparación del ecosistema de software de la plataforma. Este enfoque sienta las bases para nuevas mejoras en la configuración y ajuste del sistema, con el objetivo de aumentar la robustez. Además, se pueden identificar mejoras en la arquitectura del sistema. Tras una convocatoria de contribuciones, 13 equipos de aplicaciones aceptaron participar en el proceso de evaluación y 10 equipos llevaron a cabo el trabajo de evaluación hasta su finalización. El Cuadro 1 resume la lista final de aplicaciones y sus características principales. La lista abarca ocho ámbitos científicos diferentes e incluye códigos escritos en Fortran, C y C++. Los modelos de programación paralela utilizados fueron MPI, OpenMP/OpenACC, Kokkos, Alpaka y CUDA. En las actividades de portabilidad no se incluyeron cambios en los códigos de aplicación. [3]

REFERENCES:

[1] B. N. Chandrashekar, K. Aditya Shastry, B. A. Manjunath and V. Geetha, "Performance Model of HPC Application On CPU-GPU Platform," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-6, doi: 10.1109/MysuruCon55714.2022.9972737.

[2] Zane Fink, Konstantinos Parasyris, Giorgis Georgakoudis, and Harshitha Menon. 2023. HPAC-Offload: Accelerating HPC Applications with Portable Approximate Computing on the GPU. In The International Conference for High Performance Computing, Networking, Storage and Analysis (SC ’23), November 12–17, 2023, Denver, CO, USA. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3581784.3607095>

[3] @inproceedings{10.1145/3581576.3581621, author = {Elwasif, Wael and Godoy, William and Hagerty, Nick and Harris, J. Austin and Hernandez, Oscar and Joo, Balint and Kent, Paul and Lebrun-Grandie, Damien and Maccarthy, Elijah and Melesse Vergara, Veronica and Messer, Bronson and Miller, Ross and Oral, Sarp and Bastrakov, Sergei and Bussmann, Michael and Debus, Alexander and Steiniger, Klaus and Stephan, Jan and Widera, Rene and Bryngelson, Spencer and Le Berre, Henry and Radhakrishnan, Anand and Young, Jeffrey and Chandrasekaran, Sunita and Ciorba, Florina and Simsek, Osman and Clark, Kate and Spiga, Filippo and Hammond, Jeff and John, Stone and Hardy, David and Keller, Sebastian and Piccinali, Jean-Guillaume and Trott, Christian}, title = {Application Experiences on a GPU-Accelerated Arm-based HPC Testbed}, year = {2023}, isbn = {9781450399890}, publisher = {Association for Computing Machinery}, address = {New York, NY, USA}, url = {https://bibliotecas.ups.edu.ec:2582/10.1145/3581576.3581621}, doi = {10.1145/3581576.3581621}, abstract = {This paper assesses and reports the experience of ten teams working to port, validate, and benchmark several High Performance Computing applications on a novel GPU-accelerated Arm testbed system. The testbed consists of eight NVIDIA Arm HPC Developer Kit systems, each one equipped with a server-class Arm CPU from Ampere Computing and two data center GPUs from NVIDIA Corp. The systems are connected together using InfiniBand interconnect. The selected applications and mini-apps are written using several programming languages and use multiple accelerator-based programming models for GPUs such as CUDA, OpenACC, and OpenMP offloading. Working on application porting requires a robust and easy-to-access programming environment, including a variety of compilers and optimized scientific libraries. The goal of this work is to evaluate platform readiness and assess the effort required from developers to deploy well-established scientific workloads on current and future generation Arm-based GPU-accelerated HPC systems. The reported case studies demonstrate that the current level of maturity and diversity of software and tools is already adequate for large-scale production deployments.}, booktitle = {Proceedings of the HPC Asia 2023 Workshops}, pages = {35–49}, numpages = {15}, location = {Raffles Blvd, Singapore}, series = {HPCAsia '23 Workshops} }

[4] Kohei Yoshida, Shinobu Miwa, Hayato Yamaki, Hiroki Honda,

Analyzing the impact of CUDA versions on GPU applications,

Parallel Computing,

Volume 120,

2024,

103081,

ISSN 0167-8191,

https://doi.org/10.1016/j.parco.2024.103081.

(<https://www.sciencedirect.com/science/article/pii/S016781912400019X>)

INTRODUCCION

La computación de alto rendimiento (HPC) es una herramienta indispensable que ha revolucionado el panorama de la investigación en una gran variedad de campos, desde la meteorología y la climatología hasta la física nuclear y el descubrimiento de fármacos.

Los investigadores confían cada vez más en los sistemas HPC para abordar problemas complejos y de alta carga computacional que antes se consideraban insuperables. La HPC aprovecha arquitecturas de hardware avanzadas y técnicas de procesamiento paralelo para realizar cálculos a velocidades que superan con creces las capacidades de los sistemas informáticos convencionales. Estas máquinas sobrealimentadas son capaces de manejar conjuntos de datos masivos, realizar simulaciones complejas y ejecutar algoritmos complejos con una eficiencia extraordinaria. Gracias a ello, los investigadores de campos tan diversos como la astrofísica, la genómica, la modelización del clima o la ingeniería aeroespacial pueden explorar fenómenos, realizar experimentos y analizar datos a escalas y resoluciones sin precedentes.

El impacto de la HPC en los investigadores es polifacético. En primer lugar, agiliza la investigación al reducir significativamente el tiempo necesario para el análisis de datos, la simulación y la experimentación. Lo que en los sistemas informáticos tradicionales podía llevar meses o incluso años, ahora puede hacerse en cuestión de días u horas. Esta aceleración también ha aumentado el ritmo de los descubrimientos y facilita la iteración rápida, fomentando la innovación.[1]

Referencias

[1] HIGH PERFORMANCE COMPUTING (HPC). (2024). *Scientific Computing World*, 39. <https://bibliotecas.ups.edu.ec:3057/apps/doc/A773092352/GPS?u=ups_cons&sid=bookmark-GPS&xid=fdd790c9>