OpenDwarfs 2025: Reingeniería de la Benchmark Suite OpenDwarfs para su Uso en Computación Heterogénea

Juan José Ropero, Manuel de Castro Caballero y Diego R. Llanos[1],

Resumen— **En este trabajo presentamos el proceso de reingeniería de la benchmark suite OpenDwarfs, una colección de benchmarks diseñados para evaluar las capacidades de computación paralela en sistemas heterogéneos, usando OpenCL. El proceso de reingeniería llevado a cabo incluye la corrección de errores, la optimización de la compilación y mejoras en la usabilidad, garantizando su compatibilidad con hardware moderno. Además, realizamos experimentos que demuestran la escalabilidad de los benchmarks que forman la nueva benchmark suite OpenDwarfs 2025 en diversas plataformas de hardware, proporcionando resultados comparativos.**

Palabras clave— **OpenDwarfs, OpenCL, computación heterogénea, benchmarking, GPU, paralelismo.**

I. Introducción

OPENDWARFS es una *benchmark suite*, o colección de benchmarks, cuyo desarrollo inicial se basó en los '13 Dwarfs de Berkeley", como se describe en la publicación *The Landscape of Parallel Computing Research: A View from Berkeley* [1]. Estos *dwarfs* son patrones de computación que capturan los aspectos clave de los algoritmos paralelos, y se utilizan para medir el rendimiento de sistemas heterogéneos. OpenDwarfs fue implementada originalmente usando OpenCL, un estándar abierto para la programación paralela en plataformas heterogéneas. Esta suite fue presentada por primera vez en *OpenCL and the 13 Dwarfs: A Work in Progress* [2], y su desarrollo continuó con la publicación *On the Characterization of OpenCL Dwarfs on Fixed and Reconfigurable Platforms* [3]. Ambas publicaciones marcaron un hito en la evaluación de sistemas paralelos con arquitecturas heterogéneas, proporcionando una herramienta versátil para analizar el rendimiento de diversos sistemas de cómputo, independientemente de sus configuraciones hardware.

El objetivo principal de OpenDwarfs es ser una herramienta que permita realizar evaluaciones de sistemas de cómputo de manera agnóstica al hardware, es decir, sin depender de la arquitectura subyacente. Esto es crucial en el contexto actual, donde las plataformas heterogéneas —compuestas por CPUs multicore, GPUs, FPGAs, y arquitecturas especializadas como la Intel MIC (Many Integrated Core)— son cada vez más comunes. Al no depender del hardware específico, OpenDwarfs ofrece una forma estandarizada de medir el rendimiento, facilitando la comparación entre plataformas con diferentes configuraciones.

[1]Dpto. de Informática, Universidad de Valladolid, España. E-mail: juanjose.ropero23@estudiantes.uva.es, manuel@infor.uva.es, diego@infor.uva.es

Con el tiempo, la suite OpenDwarfs pasó a ser de código abierto [4], lo que permitió a la comunidad contribuir activamente a su desarrollo. Sin embargo, tras el lanzamiento inicial, la suite ha sufrido una falta de mantenimiento adecuado. En los últimos ocho años, la mayor parte de los benchmarks han quedado desactualizados, con la excepción de algunos ajustes menores realizados por desarrolladores independientes. A pesar de que sigue siendo posible utilizar la suite para evaluar sistemas de cómputo actuales, los avances en hardware y software, junto con la escasa documentación disponible, han hecho que su uso sea cada vez más complicado. Además, los registros de cambios del repositorio original y los hilos de discusión en los *Issue Threads* indican que los desarrolladores originales han abandonado el soporte activo de la suite.

En este contexto, surge la necesidad de un proceso de reingeniería completo de la benchmark suite OpenDwarfs. Este proyecto tiene como objetivo restaurar y actualizar la suite, reparando los defectos que dificultan su uso. Los benchmarks actuales presentan diversos problemas derivados del uso de los compiladores actuales, como errores en el proceso de compilación o ejecución, salidas incorrectas o inconsistentes, y falta de compatibilidad con las plataformas más modernas. El proceso de reingeniería consistirá en un análisis exhaustivo del estado actual de cada benchmark, la corrección de los errores identificados y la implementación de mejoras para garantizar que la suite funcione correctamente en una amplia gama de sistemas heterogéneos.

Uno de los principales desafíos de este proceso de reingeniería es asegurar que cada benchmark en la suite proporcione salidas razonables y consistentes, acordes a las tareas que cada uno está diseñado para evaluar. Las salidas de los benchmarks deben poder generarse de forma opcional, permitiendo la verificación de su correcto funcionamiento en diferentes plataformas. Este enfoque permitirá generar un conjunto de datos estándar que podrá utilizarse para validar la ejecución de los benchmarks en arquitecturas paralelas diversas. Además, se implementarán mejoras en la usabilidad de la suite, con el objetivo de facilitar la experiencia del usuario al configurar y ejecutar los benchmarks.

Otro aspecto importante del proceso de reingeniería realizado es la uniformidad en el uso de la suite. Actualmente, cada benchmark requiere configuraciones y procedimientos específicos que pueden

variar considerablemente entre ellos, lo que complica su uso para aquellos que no están familiarizados con los detalles técnicos de cada uno. El proceso de reingeniería abordará este problema mediante la estandarización de los procesos de compilación, ejecución y recolección de resultados, proporcionando una interfaz de usuario más coherente y sencilla de utilizar.

Una vez completado el proceso de reingeniería, se llevará a cabo una serie de experimentos para evaluar el rendimiento de los benchmarks en diferentes plataformas OpenCL. Estos experimentos incluirán pruebas en arquitecturas heterogéneas como CPUs multicore, GPUs, y FPGAs, con el objetivo de analizar las variaciones en los tiempos de ejecución de cada benchmark dadas ciertas entradas específicas. Además de los tiempos de ejecución, se evaluará la escalabilidad de los benchmarks en sistemas con diferentes capacidades de cómputo.

El proyecto concluirá con un análisis detallado de los resultados obtenidos en estas pruebas experimentales. Se compararán los tiempos de ejecución y la eficiencia de los benchmarks en diferentes plataformas, proporcionando una visión global del rendimiento de OpenDwarfs en sistemas paralelos heterogéneos. A partir de este análisis, se discutirán las limitaciones actuales de la suite, así como las áreas de mejora para futuras versiones. Además, se presentarán propuestas para expandir la suite, incorporando nuevos benchmarks que aborden los retos emergentes en la evaluación de sistemas heterogéneos.

Este trabajo de reingeniería no solo restaurará la utilidad de OpenDwarfs como una herramienta de benchmarking, sino que también sentará las bases para su evolución futura, asegurando que continúe siendo una referencia clave en la evaluación de plataformas de computación paralela.

II. Estructura de la suite

La suite OpenDwarfs es una herramienta diseñada para evaluar el rendimiento de sistemas heterogéneos utilizando OpenCL. Está compuesta por un conjunto de benchmarks basados en los "13 Dwarfs de Berkeley" [1], patrones computacionales que encapsulan operaciones fundamentales de algoritmos paralelos. Estos benchmarks evalúan diferentes tipos de cargas de trabajo que pueden ejecutarse en CPUs, GPUs, FPGAs y otras plataformas. La suite ha sido diseñada para ser agnóstica al hardware, permitiendo realizar comparaciones de rendimiento en una variedad de arquitecturas.

A. Descripción general y filosofía

OpenDwarfs sigue la filosofía de proporcionar un conjunto de pruebas que abarquen la mayor diversidad de patrones computacionales posibles, permitiendo evaluar diferentes aspectos del rendimiento en plataformas heterogéneas. Los benchmarks fueron seleccionados para representar una gama de aplicaciones paralelas, cubriendo áreas como álgebra lineal densa y dispersa, transformaciones espectrales, métodos dinámicos, entre otros. Cada benchmark fue desarrollado usando OpenCL, asegurando compatibilidad con una amplia gama de dispositivos, desde procesadores tradicionales hasta hardware especializado.

La suite incluye benchmarks para categorías como:

- **Dense Linear Algebra**: Evalúa operaciones de álgebra lineal sobre matrices densas.
- **Sparse Linear Algebra**: Evalúa operaciones sobre matrices dispersas.
- **Spectral Methods**: Transformaciones de Fourier, entre otras.
- **Dynamic Programming**: Métodos que resuelven problemas grandes a través de subproblemas más pequeños, como Needleman-Wunsch.

En total, la suite cubre 13 categorías diferentes, cada una representada por al menos un benchmark. La suite también permite que los usuarios ejecuten y analicen los resultados de estos benchmarks bajo diferentes configuraciones de hardware y software.

B. Prerrequisitos de compilación

Para poder compilar y ejecutar los benchmarks de OpenDwarfs, es necesario cumplir con una serie de prerrequisitos. Estos incluyen la instalación de un compilador compatible con OpenCL, como GCC/G++ en su versión 11.0 o superior, y las bibliotecas necesarias para compilar programas escritos en OpenCL. OpenDwarfs fue diseñada para ser ejecutada en sistemas GNU/Linux, por lo que la instalación de herramientas como `autoconf`, `automake`, `libtool`, y `make` es fundamental. Estos paquetes permiten la creación de scripts de compilación automáticos y la correcta gestión de dependencias durante la compilación de los benchmarks.

Otro requisito importante es la instalación de una versión de OpenCL compatible con los dispositivos en los que se ejecutarán los benchmarks. Aunque la suite fue desarrollada utilizando OpenCL 1.0, se puede utilizar una versión más moderna del estándar. Para facilitar la configuración, se recomienda el uso de herramientas como `clinfo`, que proporciona información sobre los dispositivos compatibles con OpenCL presentes en el sistema.

C. Proceso de compilación

El proceso de compilación de la suite se divide en varios pasos, asegurando que los benchmarks se configuren y generen correctamente en la máquina donde se ejecutarán. A continuación, se describen los pasos principales para compilar OpenDwarfs:

1. Ejecutar el archivo `autogen.sh` desde la raíz del repositorio. Este script genera automáticamente los archivos de cabecera y otros ficheros necesarios para la compilación.
2. Navegar al directorio `build` del repositorio y ejecutar el script `configure`. Este script genera los `Makefile` necesarios para compilar los benchmarks.
3. Finalmente, ejecutar el comando `make` para compilar los benchmarks seleccionados. En este

paso, se generan los archivos ejecutables para cada benchmark y se crean los kernels de OpenCL necesarios para su ejecución.

En algunos casos, puede ser necesario especificar opciones adicionales al ejecutar el script `configure`, como la selección de benchmarks específicos o la ruta al SDK de OpenCL. Esto permite ajustar el proceso de compilación según las características del sistema donde se ejecutarán los benchmarks.

D. Descripción de los benchmarks

La suite de OpenDwarfs contiene benchmarks que representan algoritmos y métodos utilizados en diversas disciplinas científicas y aplicaciones. A continuación, se presenta una breve descripción de los benchmarks más representativos de la suite:

- **BFS (Breadth-First Search)**: Este benchmark implementa el algoritmo de búsqueda en anchura, que se utiliza en problemas de gráficos. Evalúa el rendimiento al recorrer grandes grafos y calcular los costos de los nodos desde un nodo raíz [5].
- **BWA_HMM (Baum-Welch Algorithm)**: Este benchmark pertenece a la categoría de modelos gráficos y está basado en el algoritmo Baum-Welch, utilizado para entrenar modelos ocultos de Markov. Evalúa la capacidad de ajustar parámetros en modelos gráficos grandes.
- **CFD (Computational Fluid Dynamics)**: Simula problemas de dinámica de fluidos computacional. Este benchmark mide el rendimiento de las operaciones sobre ecuaciones diferenciales parciales utilizadas para modelar el flujo de fluidos [6].
- **FFT (Fast Fourier Transform)**: Implementa la transformada rápida de Fourier, que convierte señales en el dominio del tiempo al dominio de la frecuencia. Este benchmark es fundamental para medir el rendimiento en aplicaciones de procesamiento de señales [7].
- **KMEANS**: Algoritmo de agrupamiento que divide un conjunto de puntos en grupos según su proximidad. Este benchmark se utiliza en aprendizaje automático y minería de datos, evaluando la eficiencia de la agrupación [8].
- **LUD (LU Decomposition)**: Realiza la descomposición LU de matrices. Este método es ampliamente utilizado en la solución de sistemas de ecuaciones lineales [9].
- **NW (Needleman-Wunsch)**: Implementa el algoritmo de alineación de secuencias de ADN. Es un ejemplo de programación dinámica y se utiliza en bioinformática para encontrar el alineamiento óptimo entre dos secuencias de nucleótidos.

Cada uno de estos benchmarks está diseñado para evaluar un aspecto específico del rendimiento del hardware, ya sea en términos de computación de enteros, punto flotante, acceso a memoria o paralelismo. La ejecución de estos benchmarks en diferentes dispositivos permite una evaluación comparativa de las capacidades de cómputo de dichos sistemas.

E. Compatibilidad y flexibilidad

Una característica clave de OpenDwarfs es su flexibilidad. La suite puede ejecutarse en una amplia gama de dispositivos, incluidos procesadores convencionales, tarjetas gráficas y hardware especializado como FPGAs. Además, la suite permite al usuario ajustar ciertos parámetros de los benchmarks, como el tamaño de las matrices, la cantidad de iteraciones o el tipo de entrada utilizada. Esto facilita la personalización de las pruebas y la adaptación de la suite a diferentes escenarios experimentales.

Además de su flexibilidad, OpenDwarfs proporciona una interfaz de usuario consistente a través de scripts de línea de comandos. Los usuarios pueden compilar, ejecutar y obtener resultados de los benchmarks utilizando una serie de comandos sencillos, sin necesidad de modificar el código fuente.

F. Limitaciones

A pesar de sus muchas ventajas, OpenDwarfs también presenta ciertas limitaciones. Algunos benchmarks requieren configuraciones específicas que pueden no estar disponibles en todos los sistemas, lo que puede dificultar la comparación directa entre diferentes plataformas. Además, el uso de OpenCL como único estándar de programación limita el acceso a dispositivos que no son compatibles con este estándar.

Otro desafío es la necesidad de realizar ajustes manuales en ciertos benchmarks para que funcionen correctamente en hardware más reciente. Esto se debe a que algunos de los benchmarks fueron desarrollados hace varios años y no han sido actualizados para aprovechar las mejoras en las versiones más recientes de OpenCL.

III. Proceso de reingeniería de la suite

El proceso de reingeniería de la suite OpenDwarfs se llevó a cabo con el objetivo de corregir los diversos defectos que dificultaban o impedían su uso. El propósito de este esfuerzo era asegurar que cada uno de los benchmarks proporcionara salidas correctas y consistentes, mejorando la experiencia del usuario y permitiendo la comparación precisa entre distintas plataformas. Este capítulo se centra en los cambios aplicados a los benchmarks, categorizados por orden alfabético, detallando las modificaciones específicas realizadas y los razonamientos detrás de cada cambio.

A. Cambios generales en la suite

Uno de los primeros problemas que se abordaron fue la falta de uniformidad en el uso de las versiones de OpenCL. Al compilar los benchmarks, muchos arrojaban advertencias (*warnings*) debido a que estaban utilizando APIs obsoletas o funciones de versiones antiguas de OpenCL. Los desarrolladores originales no habían actualizado el código para

adaptarlo a las nuevas versiones del estándar, lo que provocaba inconsistencias en las compilaciones. Para solventar este problema, se estableció OpenCL 1.1 como la versión por defecto para todos los benchmarks de la suite, añadiendo la definición del marco `CL_TARGET_OPENCL_VERSION` con el valor 110 en los archivos necesarios. Este cambio se implementó de manera global para evitar la propagación de errores relacionados con versiones incorrectas.

Adicionalmente, se realizaron pruebas exhaustivas de compilación y ejecución para cada uno de los benchmarks modificados. Estas pruebas aseguraron que los cambios realizados en un benchmark no afectaran negativamente a los otros, y que la suite pudiera ser utilizada en una variedad de plataformas sin problemas de compatibilidad. A continuación se detallan los cambios realizados en cada uno de ellos.

B. BFS (Breadth-First Search)

El benchmark BFS, al igual que muchos otros, presentaba una serie de advertencias (*warnings*) al ser compilado con versiones modernas de GCC/G++. Estas advertencias se debían principalmente a la falta de compatibilidad con las versiones más recientes de OpenCL. Se decidió estandarizar el uso de OpenCL 1.1 para todos los benchmarks, introduciendo la macro `CL_TARGET_OPENCL_VERSION` con el valor `110`, lo que solucionó las advertencias. Además, se revisaron las configuraciones de tiempo y se ajustaron los temporizadores internos para asegurar la coherencia de los resultados en diferentes plataformas.

C. BWA_HMM (Baum-Welch Algorithm)

El benchmark BWA_HMM también presentó problemas relacionados con las advertencias de OpenCL, los cuales se corrigieron de manera similar. Sin embargo, este benchmark presentó un problema adicional: cuando se proporcionaban entradas grandes (más de 4000 estados), el benchmark devolvía salidas inconsistentes, incluidos valores `NaN` o errores de *Segmentation Fault*. Se identificó un acceso incorrecto a la memoria en el kernel de OpenCL, el cual fue corregido para permitir la ejecución con entradas más grandes sin errores. Además, se agregaron opciones para generar archivos de salida con los parámetros HMM calculados, facilitando el análisis de los resultados.

D. CFD (Computational Fluid Dynamics)

El benchmark de CFD presentaba problemas menores relacionados con la gestión de las versiones de OpenCL, que se solucionaron con la actualización a OpenCL 1.1. No se encontraron otros defectos significativos en la ejecución, por lo que, tras realizar pruebas exhaustivas, este benchmark se consideró completamente funcional.

E. CRC (Cyclic Redundancy Check)

En el caso del benchmark CRC [10], los problemas iniciales surgieron debido a un error en la conversión de tipos de datos en el kernel de OpenCL, lo que provocaba fallos en GPUs de Nvidia. Tras investigar, se modificó el manejo de las conversiones para permitir la compatibilidad con todas las plataformas. Además, se solucionaron problemas relacionados con la toma de tiempos en las GPUs, lo que mejoró la precisión de los resultados.

F. CSR (Compressed Sparse Row)

El benchmark CSR presentó un problema único: aunque la compilación generaba el ejecutable, no se creaba el archivo del kernel debido a un error en la línea de compilación. Este error se debía a la omisión del flag `-lm` necesario para las funciones matemáticas. Añadiendo este flag a la variable `LIBS` en el `Makefile` se resolvió el problema. También se ajustaron los temporizadores del benchmark para corregir los problemas de cronometraje detectados.

G. FFT (Fast Fourier Transform)

FFT presentó un problema significativo: no era compatible con GCC en versiones 11 o posteriores debido a la deprecación de ciertas excepciones dinámicas en C++. Para solucionar este problema, se eliminaron las excepciones dinámicas del código y se actualizaron los métodos de manejo de errores, lo que permitió que el benchmark funcionara correctamente en versiones más recientes del compilador sin afectar a su funcionamiento en versiones anteriores.

H. GEM (General Electrostatics Module)

El benchmark GEM, destinado al cálculo de potencial electrostático en biomoléculas, presentaba errores durante la compilación. El problema estaba relacionado con conflictos entre macros definidas en el código y en las bibliotecas de C++. Estos conflictos se resolvieron eliminando o redefiniendo las macros problemáticas. Además, se corrigió la gestión de las opciones de entrada del benchmark para hacerlas coherentes con la documentación.

I. KMEANS

KMEANS presentó problemas menores relacionados con las advertencias de OpenCL, que se solucionaron con la actualización a la versión 1.1. Además, se implementó la opción de almacenar los resultados de los centroides en archivos de texto en lugar de mostrarlos en pantalla, lo que facilitó el análisis de las salidas. También se añadió la posibilidad de ajustar el tamaño de `LocalSize` en las ejecuciones de OpenCL.

J. LUD (LU Decomposition)

LUD, aunque funcional, presentaba problemas al manejar matrices grandes, lo que resultaba en salidas incorrectas durante la verificación. Se descubrió que las diferencias en los resultados se debían a errores en el proceso de comparación de matrices, que se solucionaron ajustando el algoritmo de verificación. Además, se mejoró el generador de entradas para crear matrices con descomposición LU válida.

K. NQUEENS

El benchmark NQUEENS presentó un problema relacionado con la gestión de los parámetros `GlobalSize` y `LocalSize`. Si el número de `threads` era menor que el `Blocksize`, el benchmark fallaba con un error de *Segmentation Fault*. Se modificó la lógica para asegurar que `GlobalSize` y `LocalSize` tuvieran valores consistentes, resolviendo el problema. Además, se actualizó la documentación y las opciones de entrada para que coincidieran con el comportamiento del código.

L. NW (Needleman-Wunsch)

NW presentó problemas similares a otros benchmarks en términos de advertencias de OpenCL. Sin embargo, el problema principal surgió cuando se utilizaban entradas muy grandes, lo que resultaba en errores de *Segmentation Fault*. Tras varios intentos fallidos de solución, se decidió limitar temporalmente el tamaño máximo de las entradas mientras se continúa investigando una solución definitiva.

M. SRAD (Speckle Reducing Anisotropic Diffusion)

SRAD [11] solo presentó advertencias menores de OpenCL que se resolvieron de manera similar a los otros benchmarks. No se detectaron problemas adicionales durante las pruebas, por lo que se consideró completamente funcional tras el proceso de reingeniería.

N. SWAT

El benchmark SWAT, que procesa secuencias de ADN, presentaba limitaciones en cuanto al tamaño de las entradas, que se amplió durante el proceso de reingeniería. Además, se crearon dos nuevos generadores de archivos de entrada, `createswat_query` y `createswat_sampledb`, para generar secuencias de ADN con reglas de composición basadas en el principio de Chargaff. Estos cambios mejoraron significativamente la funcionalidad y usabilidad del benchmark.

M. TDM (Time-Dependent Markov)

El benchmark TDM solo presentó problemas menores relacionados con las advertencias de OpenCL, que se resolvieron con la actualización estándar. Se incluyeron nuevos ejemplos de entrada para facilitar el uso del benchmark, pero no se encontraron problemas adicionales durante el proceso de reingeniería.

El proceso de reingeniería de la suite OpenDwarfs descrito corrigió múltiples defectos que afectaban la ejecución de los benchmarks. Cada benchmark fue probado exhaustivamente en plataformas heterogéneas, asegurando la compatibilidad y funcionalidad en entornos modernos. Los cambios realizados no solo restauraron la utilidad de la suite, sino que también mejoraron su usabilidad y precisión.

IV. Mejora de la usabilidad

El proceso de reingeniería de OpenDwarfs no solo se centró en corregir los defectos de los benchmarks, sino también en mejorar la usabilidad general de la suite. Esta sección aborda los cambios realizados para facilitar la ejecución de los benchmarks y asegurar que el sistema sea más accesible para los usuarios, además de permitir una evaluación experimental más sistemática.

A. Adaptaciones para una evaluación experimental sistemática

Se establecieron diferentes *input sets* para los experimentos, inspirados en los definidos por el *SPEC CPU* [12]. Se implementaron tres categorías de *input sets*:

- **Test input set**: Este conjunto está diseñado para pruebas rápidas, con una duración de ejecución de aproximadamente un segundo. Su objetivo es garantizar que los benchmarks están correctamente instalados y funcionan adecuadamente bajo cargas de trabajo mínimas.
- **Train input set**: Este conjunto simula cargas de trabajo moderadas. Las ejecuciones duran aproximadamente 30 segundos y son utilizadas para simular condiciones de trabajo más exigentes.
- **Reference input set**: Este conjunto está diseñado para simular cargas de trabajo realistas y pesadas, con una duración de ejecución de un minuto. Los resultados obtenidos se utilizan para comparar el rendimiento de los benchmarks en diferentes plataformas de hardware.

Los tiempos de ejecución estimados fueron obtenidos usando temporizadores incluidos en los benchmarks por los desarrolladores originales. El objetivo era observar cómo se comportan los benchmarks en diferentes plataformas de OpenCL, especialmente comparando CPU y GPU. Las pruebas realizadas sugirieron que la GPU debería ofrecer un *speedup* significativo en comparación con la CPU para ciertos benchmarks, como CSR, GEM o LUD, debido a su capacidad para realizar cálculos matemáticos de forma más eficiente y en paralelo.

B. Invocaciones por línea de comandos

Uno de los principales cambios para mejorar la usabilidad fue la homogeneización de las invocaciones por línea de comandos de los diferentes benchmarks. Este cambio permitió reducir la confusión en el uso de los benchmarks, ofreciendo una interfaz más consistente para su ejecución.

B.1 Selección de plataforma y dispositivo OpenCL

Se revisó y mejoró la implementación original para la selección de plataformas y dispositivos OpenCL. Los usuarios pueden ahora utilizar las siguientes opciones para especificar la plataforma y el dispositivo de ejecución:

- `-p x -d y`: Estas dos opciones se utilizan conjuntamente para seleccionar el dispositivo correcto. Con `-p` se especifica el #id de la plataforma y con `-d` el #id del dispositivo dentro

de esa plataforma. El comando `clinfo` puede utilizarse para obtener los ids correspondientes.
- `-t n`: Permite seleccionar el tipo de dispositivo (CPU, GPU, MIC o FPGA) usando un valor numérico. Si no se especifica una plataforma (`-p`), la ejecución se realizará en la plataforma OpenCL por defecto del sistema.

Por defecto, si no se especifica un dispositivo, el sistema ejecutará los benchmarks en la CPU de la plataforma #0. Sin embargo, se realizaron ajustes para analizar todas las plataformas disponibles en busca de una CPU con soporte de OpenCL, facilitando la ejecución en configuraciones complejas.

B.2 Generadores de archivos de entrada

Se implementaron generadores de archivos de entrada para diversos benchmarks, siguiendo la estrategia usada originalmente para CRC y CSR. Estos generadores permiten crear archivos de entrada de mayor tamaño para realizar pruebas adicionales. Algunos ejemplos incluyen:

- `createswat_query` y `createswat_sampledb`: Crean secuencias de ADN basadas en las reglas de composición de Erwin Chargaff, necesarias para el benchmark SWAT [13].
- `createbfs`: Genera archivos de entrada para el benchmark BFS. Se desarrolló a mano, replicando los archivos originales incluidos en la suite, y se añadió también un generador similar proveniente de la benchmark suite Rodinia [14].
- `createlud`: Genera matrices con descomposiciones LU para el benchmark LUD, garantizando que las matrices creadas sean válidas. Se incorporó el uso de OpenMP para mejorar la generación de grandes matrices de forma paralela.

Estas mejoras aseguraron que los benchmarks de OpenDwarfs sean más fáciles de usar y configurar, proporcionando una interfaz más uniforme y permitiendo la realización de experimentos más precisos y repetibles en múltiples plataformas. Además, se añadieron opciones para personalizar el tamaño de `LocalSize` en varios benchmarks, mejorando la flexibilidad en la ejecución en sistemas heterogéneos. Confiamos en que los cambios implementados en la usabilidad de la suite OpenDwarfs faciliten su uso en experimentos futuros, mejorando la precisión y consistencia de los resultados obtenidos en plataformas modernas.

V. Resultados experimentales

En esta sección se presentan los resultados obtenidos de los experimentos realizados con los diferentes *input sets* desarrollados para la suite OpenDwarfs. Los experimentos se llevaron a cabo en distintas plataformas hardware con el objetivo de evaluar el rendimiento de los benchmarks en arquitecturas heterogéneas (CPU y GPU). A continuación, se describen los experimentos, las plataformas utilizadas y los resultados obtenidos.

A. Plataformas de experimentación

Las pruebas se realizaron en el clúster Gorgon, configurado con los siguientes componentes:

- 2× procesadores AMD EPYC 7713, cada uno con 64 núcleos físicos y 128 threads, proporcionando un total de 128 núcleos y 256 threads.
- 2× GPU Nvidia RTX 4500.
- 1× GPU Nvidia A100.
- 1× GPU AMD W7800.
- 1 TB de memoria RAM.

Este sistema fue utilizado para realizar las ejecuciones de todos los benchmarks en CPU y GPU, utilizando diferentes conjuntos de entradas (*input sets*) para las pruebas: *Test*, *Train* y *Reference*. Cada una de estas categorías simula distintas cargas de trabajo: baja, moderada y elevada, respectivamente.

B. Tiempos de ejecución

Los tiempos de ejecución de los benchmarks se calcularon a partir de la media de 30 repeticiones para cada *input set*. Esta estrategia permitió reducir el ruido en los resultados y obtener mediciones más representativas. Se ejecutaron aproximadamente 6300 ejecuciones en total, lo que se tradujo en aproximadamente 50 horas de experimentación.

En las figuras 1, 2 y 3 se muestran los tiempos medios obtenidos para cada benchmark, con los resultados organizados por *input set* y por tipo de arquitectura (CPU y GPU). En las gráficas que se presentarán, el eje Y representa los tiempos de ejecución en segundos, y el eje X identifica los distintos benchmarks.

C. Speedup

Además de los tiempos de ejecución, también se calculó el *speedup* de las ejecuciones en GPU respecto a la CPU. El *speedup* refleja la relación de rendimiento entre las arquitecturas heterogéneas, permitiendo ver de manera clara las mejoras obtenidas con GPUs. Los speedups correspondientes a los tres input sets pueden verse en las figuras 1, 2 y 3.

Se añadió un campo adicional denominado *Geo Mean Speedup*, que refleja la media geométrica de los *speedups* obtenidos para cada benchmark, proporcionando una visión más global del rendimiento.

D. Conclusiones de los experimentos

Tras analizar los resultados, se pueden extraer las siguientes conclusiones:

- En general, las ejecuciones en GPU son más rápidas que en CPU. Sin embargo, en muchos casos la diferencia no es significativa, lo que puede explicarse por el hecho de que las versiones de CPU de los benchmarks no son secuenciales, sino que aprovechan todos los threads disponibles en la CPU.
- La variación en los *speedups* se incrementa a medida que aumenta el tamaño de los problemas a resolver. Esto sugiere que las GPUs son más eficaces para tareas más grandes y paralelizables.

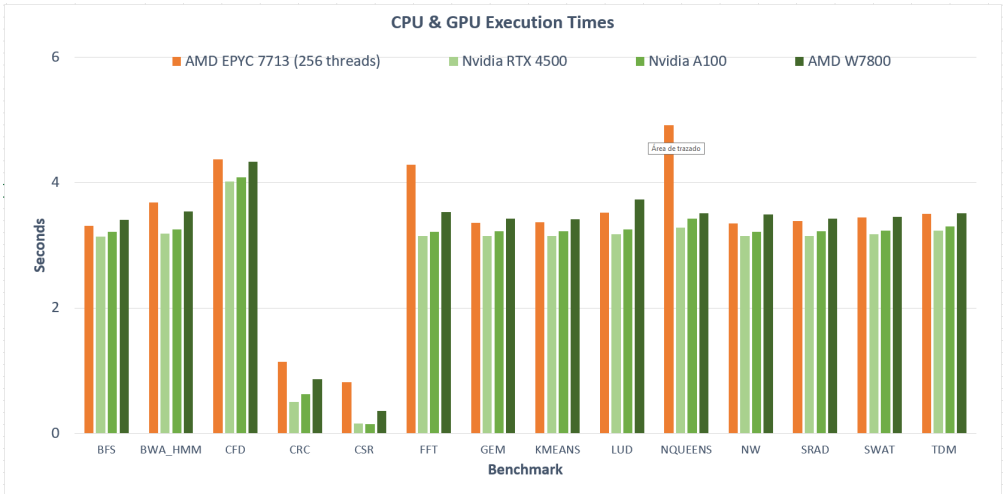

Fig. 1: Tiempos medios del *input set* Test.

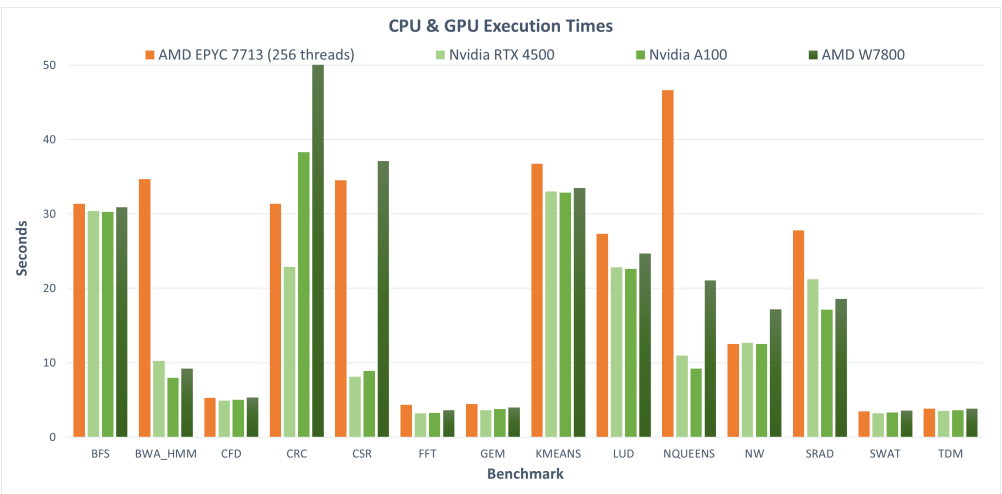

Fig. 2: Tiempos medios del *input set* Train.

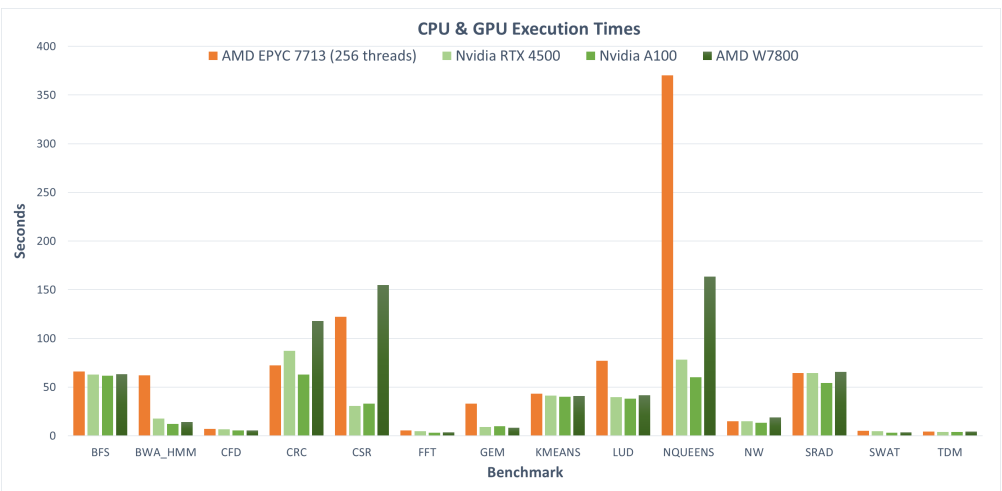

Fig. 3: Tiempos medios del *input set* Reference.

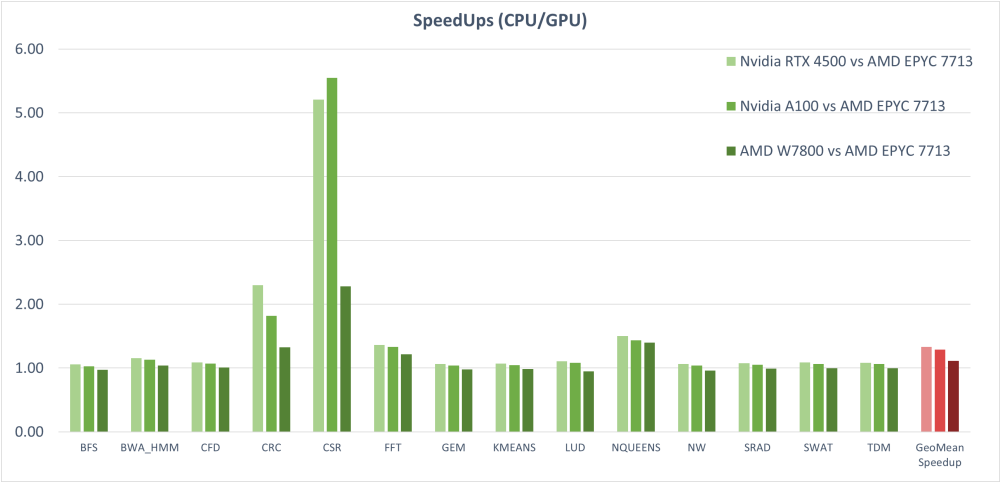

Fig. 4: *Speedup* del *input set* Test.

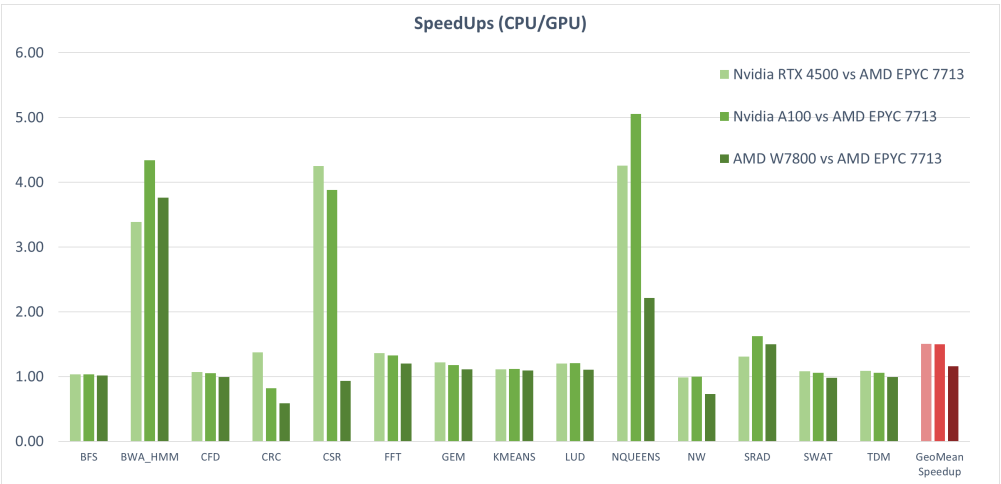

Fig. 5: *Speedup* del *input set* Train.

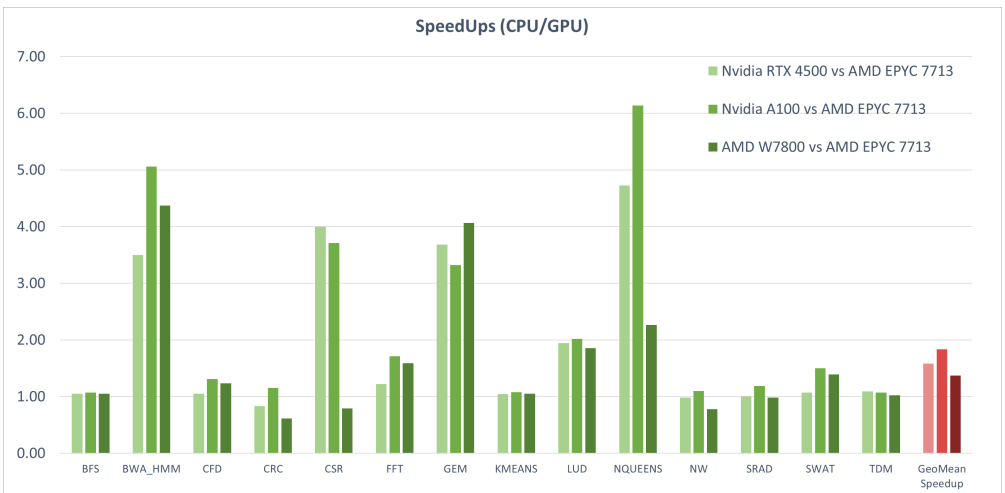

Fig. 6: *Speedup* del *input set* Reference.

Fig. 7: Escalabilidad de CPU (tiempos).

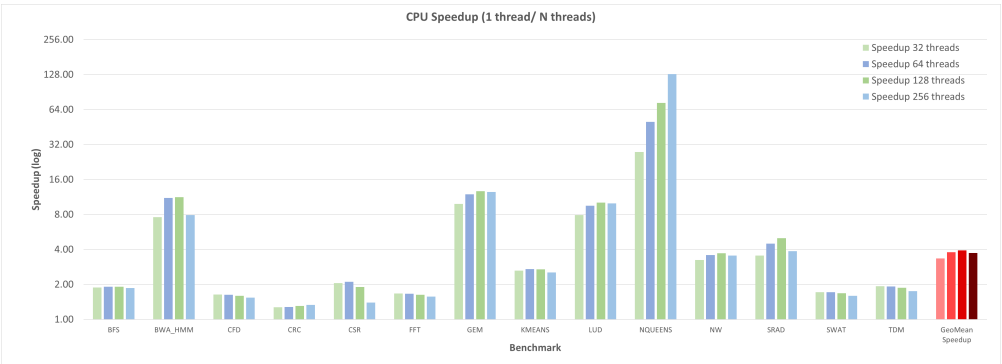

Fig. 8: Escalabilidad de CPU (*speedup*).

- Algunos benchmarks presentan anomalías en su *speedup*, como CSR y BWA_HMM, que mostraron tiempos de ejecución más largos al aumentar el número de hilos utilizados. Esto puede deberse a desbalanceo de carga, sobrecoste en la creación de hilos o problemas de contención de memoria.

E. Experimentación de escalabilidad

Se realizaron experimentos adicionales para estudiar la escalabilidad de los benchmarks utilizando solo el *input set Train*, variando el número de hilos en la CPU (1, 32, 64, 128 y 256 hilos). Los resultados mostraron que la mayoría de los benchmarks no presentan una escalabilidad ideal, con *speedups* significativamente inferiores al valor esperado en la mayoría de los casos. NQUEENS fue la excepción, con un *speedup* de hasta $\times 128$ al utilizar 256 hilos.

Para algunos benchmarks, como BWA_HMM y CSR, se observó que incrementar el número de hilos resultaba en tiempos de ejecución mayores. Esto puede deberse a problemas de optimización del código o a contención en los accesos a memoria. Se sugiere realizar análisis de *profiling* más detallados para determinar la causa exacta de estos comportamientos.

Los resultados experimentales muestran que OpenDwarfs es una herramienta útil para evaluar el rendimiento en plataformas heterogéneas. Aunque las GPUs suelen ofrecer mejores tiempos de ejecución en general, se encontraron excepciones y anomalías en algunos benchmarks. La experimentación de escalabilidad reveló que pocos benchmarks escalan de manera óptima con el número de hilos en CPU, lo que sugiere áreas de mejora en la implementación de ciertos algoritmos.

VI. Conclusiones y trabajo futuro

A lo largo de este proyecto, se han implementado una serie de mejoras significativas en la suite de benchmarks OpenDwarfs. La reingeniería de los códigos que forman la suite permitió solucionar problemas graves, especialmente en lo referente a la compatibilidad con la versión 11.0 del compilador GCC/G++. Además, se añadieron varios generadores de ficheros para diversos benchmarks, lo que facilita la creación de entradas personalizadas para experimentos específicos.

Uno de los logros más importantes es haber producido una documentación detallada de cada benchmark y de los cambios realizados durante el proyecto. Esta documentación es esencial para que futuros desarrolladores puedan comprender mejor la suite y realizar modificaciones adicionales de manera más eficiente.

A pesar de los avances realizados, aún quedan varias tareas pendientes que podrían mejorar significativamente la funcionalidad y usabilidad de la suite OpenDwarfs. Entre los trabajos futuros previstos se incluyen los siguientes:

- **Modificación de la cantidad de recursos de ejecución**: Actualmente, no es posible modificar de manera sencilla la cantidad de recursos asignados a los benchmarks. Sería recomendable permitir a los usuarios seleccionar la cantidad de recursos de hardware que desean utilizar, lo cual podría lograrse permitiendo modificar el tamaño de `LocalWorkSize` en OpenCL. Esto aumentaría la flexibilidad de la suite para realizar

evaluaciones más detalladas.

- **Compatibilidad con sistemas FPGA**: Aunque la suite fue diseñada originalmente para ser compatible con FPGAs de Altera, no se ha comprobado si las modificaciones recientes mantienen esta compatibilidad. Se recomienda realizar pruebas con distintas plataformas FPGA para asegurar que la suite sigue siendo compatible con este tipo de dispositivos y que pueda cumplir con su propósito de evaluación en entornos heterogéneos.
- **Ampliación de las entradas disponibles**: Aunque se han creado nuevos generadores de ficheros para algunos benchmarks, aún existen varios que no disponen de entradas adecuadas. El trabajo futuro debería centrarse en proporcionar generadores de ficheros para todos los benchmarks que lo requieran y revisar aquellos ya incorporados para asegurar su correcto funcionamiento.
- **Corrección de comportamientos anómalos**: Se detectaron comportamientos erróneos en algunos benchmarks, como el caso del benchmark NEEDLE, que presenta fallos al procesar entradas grandes. Estos problemas deben ser investigados y corregidos en futuras versiones para garantizar que todos los benchmarks puedan manejar un mayor rango de entradas.
- **Versión en SYCL**: El siguiente paso natural es migrar OpenDwarfs a SYCL. SYCL, una abstracción moderna de C++ de código único sobre hardware heterogéneo, ha ganado en importancia gracias a la disponibilidad de herramientas como oneAPI de Intel, ComputeCpp de Codeplay y otros entornos de ejecución de código abierto. En futuras versiones planeamos ofrecer un mecanismo de compilación dual para que los usuarios puedan trabajar sin problemas con OpenDwarfs en entornos OpenCL o SYCL.

En resumen, creemos que el trabajo realizado ha mejorado considerablemente la suite OpenDwarfs, pero existe un potencial significativo para seguir mejorando su funcionalidad y compatibilidad en futuras versiones. Estas mejoras asegurarán que la suite pueda seguir siendo una herramienta valiosa para la evaluación de sistemas heterogéneos en un futuro cercano. El nuevo benchmark suite OpenDwarfs 2025 está disponible en el siguiente enlace: *https://github.com/uva-trasgo/OpenDwarfs2025*.

Agradecimientos

El presente trabajo ha sido financiado en parte por el proyecto NATASHA (PID2022-142292NB-I00), del Ministerio de Ciencia, Innovación y Universidades de España. Manuel de Castro también cuenta con el apoyo del citado Ministerio, a través del programa "Ayudas para la Formación de Profesorado Universitario FPU 2022".

Referencias

[1] Krste Asanović, Ras Bodik, and et al., "The landscape of parallel computing research: A view from Berkeley," Tech. Rep. UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 12 2006.

[2] Wu-chun Feng, Heshan Lin, and et al., "OpenCL and the 13 Dwarfs: A work in progress," in *ACM/SPEC Intl. Conf. on Performance Engineering (ICPE)*, New York, USA, 2012, p. 291–294, ACM.

[3] Konstantinos Krommydas, Wu Feng, and et al., "On the characterization of OpenCL Dwarfs on fixed and reconfigurable platforms," in *IEEE Intl. Conf. on Application-Specific Systems, Architectures and Processors (ASAP)*, 06 2014, pp. 153–160.

[4] "Github: vtsynergy/opendwarfs," .

[5] Pawan Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA," in *Intl. Conf. on High Performance Computing (HiPC)*, 2007, p. 197–208.

[6] Andrew Corrigan, Fernando Camelli, Rainald Lohner, and John Wallin, "Running unstructured grid based CFD solvers on modern graphics hardware," *Intl. J. for Numerical Methods in Fluids*, vol. 66, pp. 221 – 229, 05 2011.

[7] James W Cooley, Peter AW Lewis, and Peter D Welch, "The fast Fourier transform and its applications," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 27–34, 1969.

[8] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek, "The global K-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[9] Richard H Bartels and Gene H Golub, "The simplex method of linear programming using LU decomposition," *Communications of the ACM*, vol. 12, no. 5, pp. 266–268, 1969.

[10] David C Feldmeier, "Fast software implementation of error detection codes," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, pp. 640–651, 1995.

[11] Yongjian Yu and Scott Acton, "Speckle reducing anisotropic diffusion," *IEEE Trans. on Image Processing*, vol. 11, pp. 1260–70, 02 2002.

[12] "Standard performance evaluation corporation," www.spec.org/benchmarks.html.

[13] Nicole Kresge, Robert D Simoni, and Robert L Hill, "Chargaff's rules: The work of Erwin Chargaff," *Journal of Biological Chemistry*, vol. 280, no. 24, pp. 172–174, 2005.

[14] Shuai Che, Michael Boyer, and et al., "Rodinia: A benchmark suite for heterogeneous computing," in *IEEE Intl. Symp. on Workload Characterization (IISWC)*, USA, 2009, p. 44–54, IEEE Computer Society.