Evaluación Experimental del Impacto Numérico y Energético de la Computación en Precisión Mixta mediante el uso de los Tensor Cores en GPUs NVIDIA para Kernels HPC

Karen Dayana Mateus Gómez y William Andrés Urrutia Torres

Trabajo de grado para optar al título de Ingeniero de Sistemas

Director

Gilberto Javier Díaz Toro

Doctor en Ciencias de la Computación

Universidad Industrial de Santander

Facultad de Ingenierías Físico-Mecánicas

Escuela de Ingeniería de Sistemas e Informática

Programa académico de Ingeniería de Sistemas

Bucaramanga

2026

Dedicatoria

Karen Dayana Mateus Gómez

Con todo mi amor, dedico este trabajo a mi abuelo Juan Gómez, quien rezó por mí cada día de este recorrido. Saber que siempre me tenía presente en sus oraciones me dio fuerzas para seguir adelante, incluso en los momentos más difíciles. A mi abuela Paulina Sánchez, quien me cuidó como a una hija y me educó con amor, paciencia y dedicación. Gracias por cada consejo y cada enseñanza, y por ayudarme a convertirme en la persona que soy hoy. Este logro también les pertenece.

A mi mamá, Alba Marina, mi mayor ejemplo de fortaleza y perseverancia. Gracias por cada palabra de aliento cuando sentí que no podía continuar, por creer siempre en mí y por enseñarme, con tu propia vida, que el esfuerzo y la constancia dan frutos. A mi papá, Héctor Manuel, por tu trabajo incansable y por cada sacrificio silencioso que hizo posible mi educación. Gracias por confiar en mí incluso cuando yo misma dudaba de mis capacidades, y por esforzarte tanto para que nunca sintiera el peso de esos sacrificios. A ambos, gracias por darme las herramientas, el amor y la confianza necesarios para llegar hasta aquí. Todo lo que soy y lo que he logrado lleva una parte de ustedes. Los amo profundamente.

A mi tía Paola Andrea, por tu amor y tu apoyo incondicional. Gracias por tus consejos, por cuidarme con ternura y por estar siempre presente, haciéndome sentir querida y acompañada.

A mi pareja, Carlos Gómez, por ser mi refugio y mi apoyo durante este camino. Gracias por sostenerme en los momentos difíciles, por celebrar conmigo cada alegría y por acompañarme en cada desafío. Tenerte a mi lado ha sido uno de los regalos más bonitos de esta etapa. Este logro también tiene un pedacito de ti.

A mi compañero de tesis, William, por tu entrega, paciencia y compromiso. Gracias por cada esfuerzo compartido, por tu respaldo en los momentos de mayor exigencia y por hacer de este trabajo una experiencia de aprendizaje y crecimiento. Me alegra haber recorrido este camino contigo y haber alcanzado juntos una meta tan importante.

William Andrés Urrutia Torres

Dedico este proyecto, en primer lugar, a mis padres, William Fernando Urrutia e Irma Consuelo Torres. Gracias por trabajar cada día, muchas veces sin descanso, y aun así ser padres presentes, atentos y cariñosos. Hoy valoro más que nunca lo que eso costó y la confianza que siempre depositaron en mí. Este logro también es suyo.

A mi hermana Yessica, la persona más cercana que tengo. Gracias por tus consejos y por esa alegría que sabes transmitir a quienes te rodeamos. Si he intentado ser mejor persona y mejor estudiante ha sido, en buena parte, para ser una vara que valga la pena superar; ojalá la superes pronto y aspires siempre a más.

A mis abuelos, Pastora Urrutia y Tristán Torres. A mi abuelita Pastora, por seguir acompañándome en este camino; y a mi abuelito Tristán, que ya no está, pero cuyos consejos y el orgullo que siempre mostró por mí siguen dándome fuerzas. Me habría encantado compartir este momento contigo.

A mis mejores amigos, Brayan Aparicio, Anderson Lemus, Hammer Muñoz y Jorge Gutiérrez, por escucharme, acompañarme en las alegrías y en los tropiezos, y por hacer de esta etapa una experiencia que recordaré siempre.

Una dedicatoria muy especial para Karen Mateus, mi mejor amiga y compañera en este proyecto y en mucho más. Conocerte marcó un antes y un después en mi vida. Gracias por creer en mí incluso cuando yo no lo hacía, por hacerme sentir capaz de dar mucho más de lo que creía tener, y por sostenerme en cada momento en que me faltaron fuerzas. Nada de esto habría sido igual sin ti. Gracias por estar en las buenas y en las malas. Que vengan muchos logros más y que podamos celebrarlos igual que este.

Agradecimientos

Expresamos nuestro más sincero agradecimiento a la Universidad Industrial de Santander, por brindarnos la oportunidad de formarnos académica y personalmente, permitiéndonos adquirir conocimientos, vivir nuevas experiencias y desarrollar habilidades fundamentales para nuestro futuro profesional.

A nuestro director de proyecto, Ph.D. Gilberto Javier Díaz Toro, y a nuestro tutor, MSc. Luis Alejandro Torres Niño, por su valiosa guía, disposición y acompañamiento constante durante el desarrollo de este trabajo.

Agradecemos a la Universidad de Cartagena por facilitarnos el acceso a la infraestructura de cómputo de alto rendimiento del sistema PACCA, utilizada para ejecutar la campaña experimental de este trabajo.

Finalmente, al grupo CAGE y al SC3 por su apoyo en infraestructura y recursos, los cuales fueron esenciales para la ejecución y culminación satisfactoria de este proyecto.

Tabla de contenido

Introducción 21
1 Objetivos 25
1.1 Objetivo general 25
1.2 Objetivos específicos 25
2 Marco de referencia 26
2.1 Fundamentos teóricos 26
2.1.1 Formatos de representación de punto flotante 26
2.1.2 Arquitectura de Tensor Cores y la operación matricial fundamental 28
2.1.3 Propagación del error numérico y drift en esquemas iterativos 29
2.1.3.1 Métricas de error 29
2.1.3.2 Definición del drift numérico 30
2.1.4 Compensación numérica y asignación selectiva de precisión 30
2.1.5 Métricas de eficiencia energética 32
2.1.6 Síntesis: los tres ejes del compromiso 33
2.2 Estado del arte 34
3 Metodología 38
3.1 Diseño experimental y plataforma de evaluación 38
3.1.1 Selección de kernels y configuraciones de precisión 38
3.1.2 Plataforma de hardware y entorno de software 39
3.1.3 Espacio de parámetros evaluado 39
3.1.4 Protocolo general de ejecución 40
3.2 Construcción de las líneas base FP64 y FP32 40
3.2.1 Implementaciones de referencia 40
3.2.2 Validación numérica 41
3.2.3 Caracterización inicial del rendimiento 41

3.3 Integración de precisión mixta y Tensor Cores 41
3.3.1 Implementaciones FP16 y BF16 41
3.3.2 Uso de cuBLAS, cuDNN, CUTLASS y kernels personalizados 42
3.3.3 Verificación de Tensor Cores con Nsight Compute 42
3.3.4 Comparaciones oficiales de rendimiento para GEMM y Convolución 2D 42
3.4 Evaluación del error numérico y drift 43
3.4.1 Referencia numérica y métricas de error 43
3.4.2 Seguimiento iterativo de los tres kernels 43
3.4.3 Esquemas de compensación numérica evaluados 44
3.4.4 Anclaje periódico en FP64 44
3.5 Medición energética y análisis estadístico 46
3.5.1 Protocolo de telemetría energética 46
3.5.2 Repeticiones y control de variabilidad 46
3.5.3 Tratamiento estadístico 46
3.6 Análisis multiobjetivo 47
3.6.1 Construcción del espacio tiempo–energía–error 47
3.6.2 Identificación del frente de Pareto 47
3.7 Alcances y limitaciones 48
3.7.1 Alcances 48
3.7.2 Limitaciones 49
4 Resultados y discusión 51
4.1 Fase 1: Construcción y caracterización de las líneas base 51
4.1.1 Validación funcional de los kernels 51
4.1.2 Rendimiento de la línea base en GEMM 52
4.1.2.1 Verificación de la ruta FP64: la referencia ya usa Tensor Cores 52
4.1.3 Rendimiento de la línea base en Convolución 2D 54
4.1.4 Rendimiento de la línea base en Stencil 2D 55

4.1.5 Síntesis de la Fase 1 56
4.2 Fase 2: Precisión mixta y utilización de Tensor Cores 57
4.2.1 Verificación experimental de Tensor Cores 57
4.2.2 Rendimiento de GEMM en precisión mixta 57
4.2.3 Rendimiento de Convolución 2D en precisión mixta 59
4.2.4 Rendimiento de Stencil 2D en precisión mixta 61
4.2.5 Interpretación arquitectónica del rendimiento 62
4.2.6 Síntesis de la Fase 2 63
4.3 Fase 3: Comportamiento numérico y costo de la compensación 63
4.3.1 Diseño de la campaña de Fase 3 64
4.3.2 Error numérico frente al tamaño del problema 66
4.3.3 Evolución del error a lo largo del encadenamiento 67
4.3.4 Estabilidad numérica y aparición de valores no finitos 69
4.3.5 Drift y horizonte de estabilidad en Stencil 2D 71
4.3.6 Efecto y costo de la compensación local 73
4.3.7 Rendimiento sostenido en función del tamaño 75
4.3.8 Síntesis de la Fase 3 76
4.4 Fase 4: Anclaje de referencia, sensibilidad y análisis multiobjetivo 77
4.4.1 Configuración experimental y definición del anclaje 77
4.4.2 Estabilidad y variabilidad experimental 78
4.4.3 Efecto de la frecuencia de anclaje sobre el tiempo 82
4.4.3.1 GEMM 85
4.4.3.2 Convolución 2D 86
4.4.3.3 Stencil 2D 87
4.4.3.4 Efecto sobre la comparación entre formatos 87
4.4.4 Efecto del anclaje sobre el error en Stencil compensado 88
4.4.5 Sensibilidad al tamaño del problema 90

4.4.6 Energía y EDP por configuración . . . 91
4.4.6.1 GEMM . . . 92
4.4.6.2 Convolución 2D . . . 93
4.4.6.3 Stencil 2D . . . 95
4.4.6.4 Cobertura . . . 96
4.4.7 Análisis multiobjetivo tiempo–energía–error . . . 96
4.4.7.1 Criterios de admisión . . . 96
4.4.7.2 Frente de Pareto de GEMM . . . 98
4.4.7.3 Frente de Pareto de Convolución 2D . . . 100
4.4.7.4 Kahan local frente a compensación espacial . . . 101
4.4.7.5 Stencil 2D . . . 102
4.4.8 Alcance y limitaciones de la campaña . . . 105
4.4.9 Síntesis de la Fase 4 . . . 105
4.5 Discusión general . . . 106
4.5.1 Influencia del patrón computacional . . . 106
4.5.2 Compromiso entre rendimiento y exactitud . . . 107
4.5.3 Diferencias entre FP16 y BF16 . . . 107
4.5.4 Costo y utilidad de la compensación . . . 107
4.5.5 Atribución energética y alcance de la medición . . . 108
4.5.6 Implicaciones para la selección de precisión . . . 109
4.5.7 Directrices de selección de precisión . . . 109
4.5.7.1 Ejes. . . . 109
4.5.7.2 Procedimiento. . . . 110
4.5.7.3 Aplicación a dos casos. . . . 113
4.6 Síntesis de resultados . . . 113
5 Conclusiones . . . 114
Referencias Bibliográficas . . . 119

Lista de tablas

Tabla 1. Comparación de los cuatro formatos de punto flotante evaluados. El épsilon de máquina se define como $\epsilon = 2^{-t}$, donde t es el número de bits de fracción almacenados. 27

Tabla 2. Validación funcional de las líneas base: GPU vs. CPU dentro de la misma precisión. 52

Tabla 3. Línea base de GEMM: comparación CPU vs. GPU en FP32 y FP64. . . . 53

Tabla 4. Perfil de Nsight Compute de la ruta FP64 de cuBLAS (GEMM, $N = 12288$). 53

Tabla 5. Línea base de Convolución 2D: comparación CPU vs. GPU en FP32 y FP64. 55

Tabla 6. Línea base de Stencil 2D ($N_X = N_Y = 4096$, $N_T = 20$, tiempos por paso): comparación CPU paralela (OpenMP) vs. GPU CUDA. 56

Tabla 7. Utilización del ancho de banda en las líneas base de Stencil 2D. 56

Tabla 8. Indicadores de utilización de Tensor Cores obtenidos mediante NVIDIA Nsight Compute. 58

Tabla 9. GEMM con precisión mixta y Tensor Cores, $N = 12288$. 58

Tabla 10. GEMM con precisión mixta y Tensor Cores, $N = 32768$. 59

Tabla 11. Convolución 2D con precisión mixta y Tensor Cores, $C = 1024$. 60

Tabla 12. Caracterización de rendimiento de las principales rutas de Stencil 2D ($N_X = N_Y = 4096$, $N_T = 20$). 62

Tabla 13. Espacio experimental de la Fase 3. El encadenamiento aplica el mismo operador fijo sobre la salida de la iteración anterior. Los tamaños corresponden a N en GEMM, HW en Convolución 2D y $n_x = n_y$ en Stencil 2D. 65

Tabla 14. Horizonte de desbordamiento en Stencil 2D: iteración del primer valor no finito frente a la predicha por el modelo ($\lambda \approx 1{,}99$; valores para 4096^2, con variación inferior al $0{,}5\,\%$ entre mallas). Las rutas de 16 bits incorporan compensación espacial. 73

Tabla 15. Configuración experimental de la Fase 4. 79

Tabla 16. Variabilidad entre réplicas del tiempo por iteración ($n = 8$, horizonte de 40 iteraciones). En Stencil 2D, las rutas WMMA usan compensación espacial en las tres periodicidades. 80

Tabla 17. Esquemas de compensación de Stencil 2D bajo el operador de estrés (campaña de variabilidad, malla 1024^2, $K = 0, 20$ iteraciones): mediana del tiempo por iteración sobre las ocho réplicas y error relativo L_2. 82

Tabla 18. Sobrecosto temporal del anclaje en FP64, expresado como $T(K)/T(0)$, con intervalos de confianza al $95\,\%$ sobre $n = 8$ réplicas. 83

Tabla 19. Prueba t pareada sobre $\ln[T(K)/T(K = 0)]$ ($n = 8$, gl= 7): valor p sin corregir y con Holm–Bonferroni. GEMM y Convolución 2D forman una única familia de ocho comparaciones; Stencil 2D se corrigió por separado sobre su propia familia de cuatro. Con $n = 8$ y normalidad no verificada, los valores inferiores a 10^{-10} se reportan como $< 10^{-10}$. 84

Tabla 20. Razón de tiempo por iteración entre FP16 y BF16 a igual periodicidad de anclaje, con intervalos de confianza al $95\,\%$ ($n = 8$). 88

Tabla 21. Iteración del primer valor no finito en Stencil 2D por ruta y tamaño de malla, operador de estrés. 90

Tabla 22. GEMM: configuración de menor EDP por categoría y tamaño. 93

Tabla 23. Convolución 2D: configuración de menor EDP por categoría y tamaño. . 95

Tabla 24. Stencil 2D (campaña con operador difusivo): configuración de menor EDP por iteración en cada categoría y malla. Tiempo y energía por iteración proceden de la ventana fiable de cada configuración. 96

Tabla 25. Criterios de admisión aplicados antes del análisis de dominancia. 97

Tabla 26. Estratos admitidos y descartados en el análisis de dominancia. 98

Tabla 27. Kahan local frente a compensación espacial, $K = 0$, malla 8192^2 a $1\,500$ iteraciones: error relativo L_2 y sobrecosto temporal respecto a la ruta sin compensación del mismo formato. 101

Tabla 28. Stencil 2D, operador difusivo: tiempo y energía de GPU por iteración y error relativo L_2 en el horizonte común, para las configuraciones representativas de cada malla. Se indica si la configuración pertenece al frente (candidatas) y, en su caso, si alguna referencia la domina. 103

Tabla 29. Directrices de selección de precisión por kernel, tolerancia y prioridad. Rendimiento frente a GPU FP64 salvo indicación; error relativo L_2 frente a FP64. 111

Lista de figuras

Figura 1. Error relativo L_2 y L_∞ respecto a la referencia FP64 tras 80 iteraciones encadenadas, con y sin compensación local. Las aspas y las bandas grises marcan las configuraciones cuya solución dejó de ser finita; en Convolución 2D las rutas FP16 se representan fuera de la escala válida. 66

Figura 2. GEMM encadenado: error relativo L_2 por iteración para $N = 4096$ y $N = 8192$, con y sin compensación local. Las aspas marcan los puntos de observación en que la solución dejó de ser finita, a partir de la iteración 30 en $N = 8192$. 68

Figura 3. Convolución 2D encadenada: error relativo L_2 por iteración para $HW = 256$ y $HW = 512$, con y sin compensación local. Las rutas FP16 dejaron de ser finitas desde la iteración 20 en los dos tamaños. 68

Figura 4. Estado de la solución en la última iteración registrada, por formato, esquema de compensación y tamaño del problema en GEMM y Convolución 2D. 70

Figura 5. Stencil 2D: evolución del error relativo L_2 respecto a la referencia FP64 en función de la iteración, para las tres mallas evaluadas. Las líneas verticales indican el horizonte de desbordamiento medido de cada formato. . 71

Figura 6. Stencil 2D: horizonte de desbordamiento por formato y tamaño de malla, en escala logarítmica. 72

Figura 7. Compensación local en GEMM y Convolución 2D: sobrecosto temporal respecto a la ruta sin compensar (izquierda) y error relativo L_2 final de las configuraciones finitas (derecha), en función del tamaño del problema. . . 74

Figura 8. Rendimiento sostenido frente al tamaño del problema. GEMM y Convolución 2D: tasa en GFLOP/s de las rutas con y sin compensación local, junto a la referencia GPU FP64. Stencil 2D: *speedup* frente a GPU FP32 de las rutas WMMA con compensación espacial y de GPU FP64. 75

Figura 9. GEMM ($N = 1024$): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una de las ocho réplicas experimentales. 85

Figura 10. Convolución 2D ($HW = 64$): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una réplica. 86

Figura 11. Stencil 2D (1024×1024): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una réplica. 87

Figura 12. Stencil 2D, ruta WMMA BF16 con compensación espacial: costo temporal $T(K)/T(0)$ (izquierda) y error propagado $E(K)/E(0)$ (derecha) según la periodicidad de anclaje, para las tres mallas evaluadas. WMMA FP16 se omite porque pierde la finitud en la iteración 29 para todos los valores de K y mallas. 89

Figura 13. Tiempo por iteración frente al tamaño del problema en los tres kernels. GEMM y Convolución 2D: rutas de 16 bits con compensación local. Stencil 2D: rutas WMMA con compensación espacial frente a la referencia GPU FP32. 91

Figura 14. GEMM: EDP mínimo válido por categoría y tamaño del problema (escala logarítmica). 93

Figura 15. Convolución 2D: EDP mínimo válido por categoría y tamaño del problema (escala logarítmica). 94

Figura 16. GEMM: proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para los dos estratos admitidos. El color codifica el error relativo L_2 en el mismo horizonte; la línea discontinua une las configuraciones no dominadas. FP64 se representa como referencia y no participa en la dominancia. 99

Figura 17. Convolución 2D: proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para los cuatro estratos admitidos. El color codifica el error relativo L_2; la línea discontinua une las configuraciones no dominadas. 100

Figura 18. Stencil 2D, operador difusivo ($\alpha = 3/16$): proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para las dos mallas evaluadas. El color codifica el error relativo L_2 del estado propagado en el horizonte común de cada malla; la forma, formato y compensación; la discontinua une las configuraciones no dominadas. GPU FP64 y GPU FP32 son referencia y no participan en la dominancia. 102

Lista de apéndices

Apéndice A. Participación y reconocimiento en CARLA 2026 **126**

Apéndice B. Resumen aceptado **127**

Apéndice C. Declaración sobre el uso de herramientas de inteligencia artificial **128**

Glosario

Ancho de banda de memoria velocidad a la que se transfieren datos entre la memoria y las unidades de cómputo.

Block FMA modelo de error que trata la operación de un Tensor Core como una única operación de bloque, y no como una suma de operaciones escalares.

Cancelación catastrófica pérdida grande de cifras significativas al restar dos números de magnitud parecida y signo opuesto.

Convolución 2D kernel que aplica un filtro sobre una malla bidimensional; común en procesamiento de señales e imágenes.

Drift numérico desviación acumulada del resultado respecto a la referencia a lo largo de las iteraciones.

EDP *Energy-Delay Product*; producto de la energía y el tiempo de ejecución, usado como métrica conjunta de eficiencia.

Energía a solución energía total consumida hasta completar el cálculo; integral de la potencia sobre la ventana de ejecución.

Épsilon de máquina menor diferencia relativa entre dos números representables consecutivos; $\epsilon = 2^{-t}$.

Frente de Pareto conjunto de configuraciones no dominadas, en las que no se puede mejorar un eje sin empeorar otro.

GEMM multiplicación de matrices densas (*General Matrix Multiply*); kernel de alta intensidad aritmética.

HPC computación de alto rendimiento; uso de sistemas masivamente paralelos para resolver problemas científicos de gran escala.

IEEE 754 estándar que define la representación y el comportamiento de los números de punto flotante.

Intensidad aritmética relación entre las operaciones de punto flotante y los bytes movidos desde memoria.

Kahan (suma compensada de) técnica que recupera la parte perdida al redondear cada suma para reducir el error acumulado.

MMA operación matricial fundamental $\mathbf{D} = \mathbf{A} \times \mathbf{B} + \mathbf{C}$ que ejecuta un Tensor Core en una sola instrucción.

Norma L_2 error euclidiano relativo; mide la desviación global respecto a la referencia.

Norma L_∞ error máximo absoluto; detecta la mayor desviación puntual del resultado.

NVML *NVIDIA Management Library*; interfaz para leer el sensor de potencia integrado en la GPU.

Potencia energía consumida por unidad de tiempo, medida en vatios.

Precisión mixta uso combinado de varios formatos dentro de un mismo cálculo, por ejemplo multiplicar en FP16 y acumular en FP32.

Rango dinámico magnitud de los valores representables, fijada por el número de bits del exponente.

Refinamiento iterativo método que calcula en baja precisión y corrige el residuo en una precisión superior.

Significante bits que codifican las cifras del número (mantisa); determinan la resolución.

Stencil 2D kernel iterativo que actualiza cada punto de una malla a partir de sus vecinos; de baja intensidad aritmética.

Subnormales números muy pequeños representados sin el bit implícito, para acercarse de forma gradual a cero.

Techo de rendimiento (roofline) modelo que fija el máximo rendimiento alcanzable según la intensidad aritmética del kernel; se usa en la Sección «Interpretación arquitectónica del rendimiento» para contrastar GEMM (limitado por cómputo) con Stencil 2D (limitado por ancho de banda de memoria).

Tensor Cores unidades de las GPUs NVIDIA que ejecutan una multiplicación-acumulación matricial en una sola instrucción.

Unidad de redondeo (u) cota del error relativo de una operación elemental; $u = \epsilon/2$.

Resumen

Título: Evaluación Experimental del Impacto Numérico y Energético de la Computación en Precisión Mixta mediante el uso de los Tensor Cores en GPUs NVIDIA para Kernels HPC

Autores: Karen Dayana Mateus Gómez y William Andrés Urrutia Torres

Palabras clave: precisión mixta; Tensor Cores; BF16; desbordamiento numérico; stencil; eficiencia energética; frente de Pareto.

Descripción:

En computación de alto rendimiento, la doble precisión (FP64) impone altos costos en tiempo y energía. Las GPU incorporan Tensor Cores para acelerar FP16 y BF16, pero la precisión reducida introduce errores de redondeo que, en esquemas iterativos, pueden comprometer la validez del resultado. Se evaluó este compromiso en GEMM, Convolución 2D y Stencil 2D, en FP64, FP32, FP16 y BF16, mediante cuBLAS, cuDNN y kernels WMMA propios en una GPU A100, midiendo tiempo, energía de GPU y error frente a FP64. En GEMM y Convolución 2D la precisión mixta aceleró el cálculo hasta $15{,}27\times$ y $28{,}07\times$ frente a FP64; en Stencil 2D la implementación WMMA evaluada no obtuvo ganancia temporal. Bajo un operador amplificante, FP16 dejó de producir valores finitos en la iteración 29 y BF16 igualó el horizonte de FP32 (142 iteraciones); la compensación del residuo de cuantización redujo el error por factores de $1{,}7$ a $5{,}2$ duplicando el tiempo, y el anclaje en FP64 solo redujo el error cuando se aplicó en todas las iteraciones, lo que equivale a calcular en doble precisión, sin extender ese horizonte. Con un operador difusivo, el análisis de Pareto mostró que en Stencil 2D la GPU en FP64 iguala en costo a la mejor ruta tensorial compensada con error ocho órdenes menor. La precisión mixta beneficia a kernels de alta intensidad aritmética; en esquemas iterativos con operadores amplificantes, el compromiso se desplaza del rendimiento al rango dinámico, sin que las estrategias evaluadas lo resuelvan.

Trabajo de grado para optar al título de Ingeniero de Sistemas. Universidad Industrial de Santander, Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática. Director: Gilberto Javier Díaz Toro, Doctor en Ciencias de la Computación.

Abstract

Title: Experimental Evaluation of the Numerical and Energy Impact of Mixed-Precision Computing Using Tensor Cores on NVIDIA GPUs for HPC Kernels

Authors: Karen Dayana Mateus Gómez and William Andrés Urrutia Torres

Keywords: mixed precision; Tensor Cores; BF16; numerical overflow; stencil; energy efficiency; Pareto front.

Description:

In high-performance computing (HPC), the widespread use of double precision (FP64) imposes high costs in execution time and energy. Modern GPUs incorporate Tensor Cores that accelerate reduced formats such as FP16 and BF16, but reduced precision introduces rounding errors that, in iterative schemes, can compromise the validity of the result. This trade-off was evaluated on GEMM, 2D Convolution, and 2D Stencil, in FP64, FP32, FP16, and BF16, using cuBLAS, cuDNN, and custom WMMA kernels on an NVIDIA A100 GPU, measuring time, GPU energy, and error against FP64. In GEMM and 2D Convolution, mixed precision achieved speedups of up to 15.27× and 28.07× over FP64; in 2D Stencil, the evaluated WMMA implementation achieved no speedup. Under an amplifying operator, FP16 stopped producing finite values at iteration 29, while BF16 matched the FP32 horizon (142 iterations); compensating the quantization residual reduced the error by factors of 1.7 to 5.2 while doubling execution time, and FP64 anchoring reduced it only when applied at every iteration, which amounts to computing in double precision, without extending that horizon. Under a diffusive operator, Pareto analysis showed that in 2D Stencil the FP64 GPU baseline matches the cost of the best compensated tensor route with eight orders of magnitude less error. Mixed precision benefits kernels with high arithmetic intensity; in iterative schemes with amplifying operators, the trade-off shifts from performance to dynamic range, and none of the strategies evaluated resolves it.

Undergraduate thesis for the degree of Systems Engineer. Universidad Industrial de Santander, Faculty of Physical and Mechanical Engineering, School of Systems and Informatics Engineering. Director: Gilberto Javier Díaz Toro, Doctor of Computer Science.

Introducción

Históricamente, la computación científica de alto rendimiento (HPC) ha empleado el formato de doble precisión (FP64), del estándar IEEE 754, para preservar la exactitud y la estabilidad numérica en numerosas aplicaciones científicas. Sin embargo, la evolución de las arquitecturas de unidades de procesamiento gráfico (GPU), impulsada en gran medida por el desarrollo de aplicaciones de inteligencia artificial y aprendizaje profundo, ha favorecido la incorporación de unidades especializadas para operaciones matriciales. En las GPU NVIDIA, estas unidades, denominadas Tensor Cores, aceleran operaciones de multiplicación y acumulación matricial mediante esquemas de precisión mixta: los operandos se multiplican en formatos reducidos como FP16 o BF16, mientras que la acumulación se realiza en FP32 para contener parcialmente la pérdida de exactitud.

Esta evolución arquitectónica plantea un desafío fundamental para la ingeniería de software científico: determinar en qué condiciones es posible aprovechar la capacidad computacional y el potencial de eficiencia energética de los Tensor Cores sin comprometer la exactitud numérica requerida por la aplicación. El uso de formatos de menor precisión reduce el tamaño de los datos y puede disminuir el volumen de información transferida en la jerarquía de memoria, además de habilitar rutas de ejecución con mayor capacidad de cómputo. Sin embargo, estas ventajas están acompañadas por una menor resolución numérica y, dependiendo del formato, por un rango dinámico más limitado: FP16 sacrifica rango y precisión respecto a FP32, mientras que BF16 conserva el rango dinámico de FP32 a costa de un significante más corto. Como consecuencia, los errores de redondeo pueden propagarse y acumularse durante el cálculo, en particular en algoritmos iterativos como el Stencil 2D, hasta producir desviaciones significativas o incluso comportamientos divergentes respecto a una referencia de mayor precisión.

En este contexto, la presente investigación evalúa experimentalmente el compromiso entre rendimiento computacional, consumo energético y exactitud numérica asociado al uso de precisión mixta sobre tres kernels representativos de HPC: multiplicación de matrices densas (GEMM), Convolución 2D y Stencil 2D. El estudio considera configuraciones basadas en FP64, FP32, FP16 y BF16, incluyendo rutas aceleradas mediante Tensor Cores cuando resultan aplicables. La

degradación numérica se cuantifica respecto a una referencia FP64 y, en los tres kernels, se analiza además la evolución del *drift* a lo largo de iteraciones encadenadas y el efecto de mecanismos de compensación numérica. Asimismo, se caracteriza el comportamiento temporal y energético de las configuraciones evaluadas.

A partir de estas mediciones, el trabajo busca caracterizar el espacio conjunto tiempo–energía–error e identificar las configuraciones no dominadas mediante un Frente de Pareto, entendiendo por configuración no dominada aquella para la cual no existe otra alternativa que sea igual o mejor en los tres criterios y estrictamente mejor en al menos uno de ellos. El propósito no consiste en asumir que la reducción de precisión constituye siempre una alternativa superior a FP64, sino en establecer, a partir de evidencia experimental, bajo qué condiciones una configuración de precisión mixta puede ofrecer un compromiso favorable entre rendimiento, consumo energético y exactitud numérica de acuerdo con la tolerancia de la aplicación.

Planteamiento y justificación del problema

En el contexto de la computación a exaescala, el uso generalizado de doble precisión (FP64) en aplicaciones científicas puede imponer costos significativos en tiempo de ejecución y consumo energético. Aunque FP64 continúa siendo un formato de referencia por sus ventajas en exactitud y estabilidad numérica, mantener uniformemente alta precisión puede limitar el aprovechamiento del hardware cuando determinadas etapas del cálculo admiten esquemas de precisión mixta cuidadosamente controlados (Kashi et al., 2024). Paralelamente, el creciente costo energético de los sistemas HPC ha convertido la eficiencia energética en un criterio relevante de diseño y ejecución (Suarez et al., 2025).

Las arquitecturas GPU contemporáneas presentan diferencias importantes entre la capacidad de cómputo disponible para distintos formatos numéricos. Los formatos reducidos, como FP16 y BF16, disminuyen el tamaño de los operandos y habilitan unidades especializadas para operaciones matriciales. En las GPU NVIDIA, esta capacidad se materializa principalmente mediante los Tensor Cores, diseñados para ejecutar operaciones de multiplicación y acumulación matricial bajo esquemas de precisión mixta. Su utilización puede incrementar significativamente el *through-*

put; sin embargo, la ventaja depende del kernel, de la configuración de precisión y de la capacidad de la implementación para utilizar efectivamente dichas unidades. El comportamiento numérico también depende de la representación de los operandos, la precisión de acumulación y la semántica aritmética efectiva del hardware (Khattak & Mikaitis, 2025).

La dimensión energética amplía el problema más allá del rendimiento. Diversos trabajos han explorado estrategias para reducir el consumo asociado al cómputo tensorial (Atoofian, 2023), lo que evidencia que la precisión reducida debe analizarse tanto desde la velocidad de ejecución como desde la energía necesaria para completar el cálculo. Una reducción de potencia instantánea o de tiempo de ejecución no garantiza por sí sola una menor energía total, por lo que ambas magnitudes deben medirse experimentalmente.

A estas consideraciones se añade la restricción numérica. El empleo de formatos de menor precisión incrementa los errores de redondeo y puede favorecer fenómenos como la cancelación catastrófica y la absorción en sumas de magnitudes dispares. Su relevancia depende del formato, del algoritmo y de los datos, por lo que una configuración ventajosa en rendimiento no necesariamente conserva la exactitud requerida. En consecuencia, es necesario determinar qué degradación numérica acompaña a la aceleración y si permanece dentro de niveles aceptables para el problema considerado.

Este problema resulta especialmente relevante al comparar kernels con patrones de cómputo diferentes, como GEMM, Convolución 2D y Stencil 2D. En GEMM y Convolución 2D, la exactitud depende, entre otros factores, de la representación de los operandos y del modo en que se acumulan los productos parciales. En las rutas de precisión mixta evaluadas, la acumulación en FP32 permite contener parcialmente la degradación numérica. En Stencil 2D, además, la salida de una iteración puede convertirse en la entrada de la siguiente, permitiendo que pequeñas desviaciones se propaguen a lo largo del horizonte temporal. Esto permite estudiar no solo el error final, sino también su evolución iterativa. Asimismo, operadores stencil pueden reformularse mediante operaciones matriciales para aprovechar Tensor Cores (Chen et al., 2024).

La posibilidad de ejecutar estos kernels mediante rutas tensoriales no resuelve, por sí sola,

el problema de investigación. Verificar la utilización de Tensor Cores confirma la ruta de ejecución empleada, mientras que una reducción del tiempo evidencia una mejora de rendimiento. Sin embargo, estas observaciones no establecen si el resultado conserva la calidad numérica requerida ni si existe una mejora energética global. Por ello, el rendimiento debe evaluarse junto con la desviación respecto a una referencia de mayor precisión y la energía consumida, considerando además que la semántica aritmética de los Tensor Cores puede afectar el comportamiento numérico observado (Khattak & Mikaitis, 2025).

A pesar de los avances en precisión mixta y hardware tensorial, aún son limitados los criterios para determinar bajo qué condiciones una configuración de precisión reducida resulta adecuada para un kernel científico específico. Muchos estudios se concentran en rendimiento y exactitud numérica, mientras que la dimensión energética recibe menor atención. En particular, son escasos los trabajos que comparen, bajo una misma plataforma y condiciones experimentales controladas, kernels con patrones de cómputo diferentes considerando simultáneamente tiempo de ejecución, energía y degradación numérica.

En consecuencia, esta investigación busca aportar evidencia experimental para utilizar las capacidades de precisión mixta del hardware moderno sin desvincular las mejoras de rendimiento y eficiencia energética de sus consecuencias numéricas. El propósito es caracterizar este compromiso y establecer criterios para identificar configuraciones apropiadas según el kernel y la tolerancia al error de la aplicación. A partir de esta necesidad se formula la siguiente pregunta de investigación: ¿En qué medida la implementación de esquemas de computación en precisión mixta, mediante el uso de Tensor Cores en GPUs NVIDIA, permite optimizar el rendimiento computacional y el consumo energético en kernels representativos de HPC, como GEMM, Convolución 2D y Stencil 2D, manteniendo el error numérico dentro de límites aceptables para la aplicación?

1 Objetivos

1.1 Objetivo general

Determinar empíricamente las configuraciones de computación en precisión mixta con uso de Tensor Cores en GPUs NVIDIA que ofrezcan compromiso entre rendimiento computacional, consumo energético y exactitud numérica en kernels representativos de HPC, mediante una evaluación sistemática basada en mediciones de potencia, energía y métricas de error durante la ejecución, sin exceder niveles aceptables de error para la aplicación.

1.2 Objetivos específicos

- Implementar y adaptar versiones paramétricas de kernels representativos de HPC (GEMM, Convolución 2D y Stencil 2D) en distintos formatos de precisión FP64, FP32, FP16 y BF16, activando explícitamente los Tensor Cores mediante el uso de bibliotecas oficiales (cuBLAS, cuDNN, CUTLASS) y kernels personalizados.

- Medir y caracterizar el rendimiento computacional y el consumo energético de cada variante de precisión, cuantificando la latencia, el throughput efectivo (TFLOPS) y el Energy Delay Product (EDP), así como el consumo de potencia y energía durante la ejecución mediante herramientas de perfilamiento y telemetría, como Nsight Compute, NVML para la GPU y RAPL para la CPU host.

- Evaluar la evolución del error numérico y el drift por iteración en esquemas iterativos bajo distintas configuraciones de precisión, con y sin algoritmos de compensación, como la suma de Kahan, mediante métricas de error basadas en las normas L_2 y L_∞, con el fin de establecer criterios para la selección de la precisión de acuerdo con la tolerancia numérica de la aplicación.

- Realizar un análisis multiobjetivo del espacio tridimensional conformado por Tiempo, Energía y Error, mediante la construcción del Frente de Pareto, con el fin de identificar configuraciones no dominadas y establecer directrices de ingeniería para la selección de la estrategia de precisión más adecuada según la tolerancia de la aplicación objetivo.

2 Marco de referencia

El problema abordado en esta investigación involucra tres dominios que suelen estudiarse por separado: la aritmética de punto flotante, la arquitectura de los aceleradores gráficos y la eficiencia energética de los sistemas de cómputo. Ninguno de los tres, considerado de forma aislada, permite decidir si una configuración de precisión reducida es aceptable para una aplicación científica; esa decisión requiere evaluar conjuntamente el tiempo de ejecución, la energía consumida y el error numérico resultante.

El capítulo se organiza de la siguiente manera. Primero se describe el efecto de reducir el número de bits del significante en un formato de punto flotante. A continuación, se describe cómo el hardware tensorial modifica la semántica de la operación aritmética elemental. Luego se formalizan la propagación y la medición del error, los mecanismos algorítmicos para contenerlo y las métricas de costo energético. Finalmente, se revisa la literatura en ese mismo orden, identificando para cada eje los resultados establecidos y los problemas abiertos que motivan la pregunta de investigación formulada en la Introducción.

2.1 Fundamentos teóricos

2.1.1 Formatos de representación de punto flotante

En computación científica, los números reales se aproximan mediante el estándar IEEE 754. Para un número *normalizado*, el valor representado es

$$V = (-1)^s \times (1 + m) \times 2^{e-\text{bias}}, \tag{1}$$

donde s es el bit de signo, $m \in [0, 1)$ es la parte fraccionaria del significando y e es el exponente sesgado. Los formatos FP64, FP32, FP16 y BF16 siguen esta representación, diferenciándose principalmente en la distribución de los bits destinados al exponente y a la fracción (Higham, 2002).

Cuando $e = 0$, IEEE 754 define los números *subnormales* como

$$V = (-1)^s \times m \times 2^{1-\text{bias}}, \tag{2}$$

sin el bit implícito del significando. Esta distinción es relevante porque las unidades tensoriales

pueden aplicar *flush-to-zero* a valores subnormales, introduciendo diferencias respecto al comportamiento IEEE 754 convencional (Fasi et al., 2021).

La precisión de cada formato depende del número de bits de fracción almacenados, mientras que los bits del exponente determinan su rango dinámico. El épsilon de máquina se define como $\epsilon = 2^{-t}$, donde t es el número de bits de fracción almacenados, y la unidad de redondeo como $u = \epsilon/2$ bajo redondeo al más cercano (Higham, 2002). Estas magnitudes condicionan la resolución y la propagación del error numérico.

FP16 y BF16 ocupan 16 bits, pero distribuyen sus bits de manera diferente. FP16 dispone de 10 bits de fracción y ofrece mayor resolución, aunque con un rango dinámico más limitado. BF16 utiliza 7 bits de fracción y conserva un rango dinámico comparable al de FP32. Por ello, FP16 presenta menor error de redondeo asociado al formato, mientras que BF16 ofrece mayor capacidad para representar valores de magnitud extrema.

Tabla 1

Comparación de los cuatro formatos de punto flotante evaluados. El épsilon de máquina se define como $\epsilon = 2^{-t}$, donde t es el número de bits de fracción almacenados.

Formato	**Bits**	**Exp.**	**Frac.**	ϵ	**Máx. representable**
FP64	64	11	52	$2{,}22 \times 10^{-16}$	$\sim 1{,}8 \times 10^{308}$
FP32	32	8	23	$1{,}19 \times 10^{-7}$	$\sim 3{,}4 \times 10^{38}$
FP16	16	5	10	$9{,}77 \times 10^{-4}$	$6{,}55 \times 10^{4}$
BF16	16	8	7	$7{,}81 \times 10^{-3}$	$\sim 3{,}4 \times 10^{38}$

Reducir la precisión disminuye el volumen de datos transferidos y puede habilitar unidades de hardware especializadas con mayor capacidad de cómputo (Abdelfattah et al., 2021). Sin embargo, también incrementa la sensibilidad al redondeo y a la pérdida de significancia (Blanchard et al., 2020). En consecuencia, la selección del formato constituye un compromiso entre rendimiento, consumo energético y exactitud numérica, dimensiones que se evalúan conjuntamente en este tra-

bajo.

2.1.2 Arquitectura de Tensor Cores y la operación matricial fundamental

Los Tensor Cores son unidades funcionales especializadas, introducidas por NVIDIA a partir de la arquitectura Volta, diseñadas para acelerar operaciones matriciales de multiplicación y acumulación (*Matrix Multiply-Accumulate*, MMA):

$$\mathbf{D} = \mathbf{A} \times \mathbf{B} + \mathbf{C}, \tag{3}$$

donde $\mathbf{A}$ y $\mathbf{B}$ son los operandos de entrada, $\mathbf{C}$ el acumulador y $\mathbf{D}$ el resultado. Estas unidades operan sobre bloques matriciales de tamaño fijo y permiten agrupar múltiples productos y acumulaciones, con un rendimiento superior al de las rutas escalares convencionales (Markidis et al., 2018).

En las configuraciones de precisión mixta consideradas en este trabajo, los operandos se representan en formatos reducidos como FP16 o BF16, mientras que la acumulación se realiza en FP32. Esta combinación busca aprovechar el mayor rendimiento de los formatos de 16 bits y, al mismo tiempo, reducir la pérdida de exactitud durante la acumulación. Por ello, una configuración debe distinguir entre la precisión de los operandos y la precisión utilizada para acumular.

Desde el punto de vista numérico, el comportamiento de estas operaciones no puede describirse únicamente mediante el modelo escalar clásico de punto flotante. Blanchard *et al.* (2020) desarrollan un análisis para el *block FMA* en precisión mixta que permite establecer cotas sobre el error producido por estas operaciones y constituye la base del análisis de la Sección «Propagación del error numérico y drift en esquemas iterativos».

Las arquitecturas recientes también incorporan formatos como TF32 y rutas tensoriales en doble precisión (Andersch et al., 2022). En paralelo, trabajos como MixPert (Lin et al., 2024) exploran la emulación de precisión mixta sobre Tensor Cores de tipo entero, ampliando el espacio de formatos explotables más allá de FP16 y BF16. Además, bibliotecas de alto nivel pueden seleccionar automáticamente estas rutas, por lo que la precisión nominal utilizada por el programa no siempre permite identificar la unidad de cómputo ejecutada. Por esta razón, en este trabajo la

activación de Tensor Cores se verifica mediante herramientas de perfilado.

Finalmente, su aplicación no se limita a GEMM y convoluciones. Algunos operadores *stencil* pueden reformularse mediante operaciones matriciales para utilizar Tensor Cores (Chen et al., 2024). Sin embargo, debido a su baja intensidad aritmética, el rendimiento de Stencil 2D puede estar limitado por el ancho de banda de memoria (Zhang et al., 2025), lo que lo convierte en un caso relevante para evaluar si el uso de Tensor Cores produce una mejora efectiva.

2.1.3 Propagación del error numérico y drift en esquemas iterativos

Cada operación en aritmética de precisión finita introduce un error de redondeo relacionado con la unidad de redondeo $u = \epsilon/2$. Para un producto interno de longitud n, el análisis clásico acota la acumulación del error mediante $\gamma_n = nu/(1-nu)$, siempre que $nu < 1$ (Higham, 2002). Esta cota muestra que el error puede crecer con el número de operaciones y verse amplificado por fenómenos como la cancelación catastrófica y el condicionamiento del problema.

En operaciones tensoriales, el comportamiento también depende del orden y la precisión de acumulación. Blanchard *et al.* (2020) extienden el análisis al *block FMA* en precisión mixta, mientras que Fasi *et al.* (2021) muestran que la estructura de acumulación puede modificar la propagación del error. Asimismo, El Arar *et al.* (2025) evidencian que el error depende tanto del formato como del condicionamiento del cálculo. Por ello, en este trabajo la degradación numérica se cuantifica empíricamente para cada configuración.

2.1.3.1 Métricas de error. Sea X_{ref} el resultado obtenido en FP64, utilizado como referencia numérica de alta precisión, y sea $\hat{X}$ el resultado de una configuración de menor precisión. El error relativo en norma L_2 mide la desviación global:

$$E_{L_2} = \frac{\left\|\hat{X} - X_{\text{ref}}\right\|_2}{\left\|X_{\text{ref}}\right\|_2}, \qquad \left\|\hat{X} - X_{\text{ref}}\right\|_2 = \sqrt{\sum_{i=1}^{N} \left(\hat{X}_i - X_{\text{ref},i}\right)^2}. \tag{4}$$

De forma complementaria, la norma L_∞ mide el máximo error absoluto:

$$E_{L_\infty} = \left\|\hat{X} - X_{\text{ref}}\right\|_\infty = \max_i \left|\hat{X}_i - X_{\text{ref},i}\right|. \tag{5}$$

Ambas métricas se utilizan conjuntamente: L_2 caracteriza la desviación global, mientras que L_∞ permite detectar errores localizados que podrían quedar ocultos en una medida agregada (Higham, 2002).

2.1.3.2 Definición del drift numérico. En esquemas iterativos, la salida de una iteración se convierte en la entrada de la siguiente, por lo que pequeñas desviaciones pueden propagarse y acumularse (Higham, 2002). Croci y Giles (2022) muestran que, en problemas iterativos ejecutados en baja precisión, el comportamiento del error puede cambiar significativamente con el esquema de redondeo. Por esta razón, este trabajo analiza la evolución del error iteración por iteración.

Sea $u_{\text{ref}}^{(n)}$ el estado de referencia FP64 en la iteración n y $u_p^{(n)}$ el estado obtenido con la configuración $p \in \{\text{FP32}, \text{FP16}, \text{BF16}\}$. El *drift* global relativo se define como

$$D_2^{(p)}(n) = \frac{\left\| u_p^{(n)} - u_{\text{ref}}^{(n)} \right\|_2}{\max\left(\left\| u_{\text{ref}}^{(n)} \right\|_2, \varepsilon \right)}, \tag{6}$$

donde ε es una constante positiva pequeña utilizada para evitar divisiones por cero. El máximo error puntual se expresa mediante

$$D_\infty^{(p)}(n) = \left\| u_p^{(n)} - u_{\text{ref}}^{(n)} \right\|_\infty = \max_i \left| u_{p,i}^{(n)} - u_{\text{ref},i}^{(n)} \right|. \tag{7}$$

La evolución de $D_2^{(p)}(n)$ permite determinar si el error permanece acotado, crece durante la simulación o alcanza un comportamiento que invalida el resultado. Este seguimiento constituye la base del análisis de drift asociado al tercer objetivo específico.

2.1.4 Compensación numérica y asignación selectiva de precisión

La precisión mixta busca reducir el costo computacional sin perder de forma incontrolada la exactitud numérica. Higham y Mary (2022) distinguen estrategias que actúan tanto sobre la acumulación de las operaciones como sobre la asignación de precisión en diferentes etapas del algoritmo.

Una de estas estrategias es la suma compensada de Kahan, que mantiene una variable auxiliar para recuperar parte del error perdido durante las acumulaciones (Higham, 2002). El método

puede reducir significativamente el error de suma, aunque introduce operaciones adicionales y requiere mantener el término de compensación. En este trabajo se evalúa sobre Stencil 2D una variante por celda inspirada en ella (Kahan local, Sección «Esquemas de compensación numérica evaluados»).

En las rutas tensoriales, la acumulación interna realizada por los Tensor Cores no puede modificarse desde el algoritmo. Por esta razón, en este trabajo la compensación se aplica fuera de la instrucción MMA: se conserva en FP32 el residuo de cuantización del estado y se reincorpora en la iteración siguiente (Sección «Esquemas de compensación numérica evaluados»).

En un operador de vecindario, como el Stencil 2D, la compensación puede aplicarse de dos formas. En la variante *Kahan local*, cada celda corrige únicamente el error de redondeo de sus propias escrituras, de forma análoga a la suma de Kahan clásica (Higham, 2002). En la compensación *espacial*, el residuo de cada celda se propaga a las celdas vecinas que la leen, de modo que también se incluye al aplicar el operador. La variante Kahan local no corrige el residuo de las celdas vecinas que intervienen en el cálculo de cada celda nueva. La compensación espacial, propuesta en este trabajo, utiliza la linealidad del operador de diferencias finitas ($L(v + c) = L(v) + L(c)$) para aplicar el residuo de cada celda sobre sus vecinas manteniendo la ruta de Tensor Cores. Los resultados de esta comparación se presentan en la Sección «Kahan local frente a compensación espacial».

Otra estrategia consiste en utilizar diferentes precisiones según la sensibilidad numérica de cada etapa del cálculo. Métodos como el refinamiento iterativo ejecutan operaciones costosas en baja precisión y reservan una precisión mayor para las etapas de corrección (Haidar et al., 2020; Higham & Mary, 2022). De forma similar, El Arar *et al.* (2025) muestran que una asignación no uniforme de precisión puede mejorar el compromiso entre costo y exactitud.

Una variante de esta idea distribuye la precisión a lo largo del tiempo y no entre etapas del algoritmo: en un esquema encadenado, una fracción de las iteraciones se evalúa en precisión alta y el resto en precisión reducida. El mecanismo difiere del refinamiento iterativo en qué corrige. El refinamiento iterativo calcula un residuo del sistema y lo corrige en una precisión superior, de

modo que su beneficio depende de la convergencia de ese proceso de corrección.

Una iteración evaluada en precisión alta, en cambio, evita el redondeo de esa iteración concreta, pero no reconstruye la desviación ya acumulada en las anteriores. El residuo de almacenamiento que conserva le permite reconstruir el estado propio de la ruta de precisión reducida, no la desviación de ese estado respecto a una trayectoria de referencia externa, que es la magnitud que habría que conocer para eliminar el error previamente acumulado. Su beneficio depende entonces de la frecuencia con que se aplique y no de una propiedad de convergencia. En cuanto a la magnitud sobre la que actúa, al igual que la compensación, opera sobre la resolución del estado y no amplía el rango de exponente del formato en que este se almacena, de modo que no desplaza el punto en que un estado creciente agota ese rango. Este trabajo evalúa esa estrategia bajo el nombre de anclaje periódico en FP64 (Sección «Anclaje periódico en FP64»).

En consecuencia, la precisión mixta no implica sustituir globalmente un formato por otro, sino combinar precisiones y mecanismos de compensación de acuerdo con las características del algoritmo. Su efectividad debe determinarse experimentalmente para cada configuración.

2.1.5 Métricas de eficiencia energética

La eficiencia energética constituye una dimensión relevante en sistemas HPC, donde una mejora de rendimiento no implica necesariamente una reducción del consumo energético (Suarez et al., 2025). Para distinguir ambas magnitudes, la potencia $P(t)$ representa la tasa de consumo de energía en un instante, mientras que la energía a solución corresponde a la energía total empleada durante la ejecución:

$$E_{\text{total}} = \int_{t_0}^{t_0+T_{\text{exec}}} P(t)\,\mathrm{d}t, \tag{8}$$

donde T_{exec} es el tiempo de ejecución dentro de la ventana de medición. En este trabajo, dicha ventana se delimita alrededor del kernel para evitar incluir etapas ajenas al cálculo evaluado.

La precisión reducida puede modificar tanto el tiempo de ejecución como el comportamiento energético, debido a la disminución del volumen de datos y al uso de unidades de cómputo especializadas (Mukunoki & Ogita, 2020). Sin embargo, una menor potencia instantánea

no garantiza una menor energía total: una configuración de baja potencia pero mayor duración puede resultar menos eficiente. Por esta razón, se analizan por separado el tiempo, la potencia media y la energía consumida. La estimación energética mediante telemetría también presenta limitaciones. Interfaces como NVML obtienen mediciones con una resolución temporal finita, lo que puede afectar especialmente a kernels de corta duración (van der Vlugt et al., 2025; Yang, Adámek, & Armour, 2024). En consecuencia, las configuraciones se evalúan mediante múltiples repeticiones y se considera la variabilidad de las mediciones antes de comparar sus diferencias energéticas.

Como métrica conjunta de tiempo y energía se utiliza el *Energy-Delay Product* (EDP):

$$\mathrm{EDP} = E_{\text{total}} \times T_{\text{exec}}, \tag{9}$$

donde un valor menor representa, en términos generales, un compromiso más favorable entre ambas dimensiones. No obstante, el EDP no incorpora la calidad numérica del resultado. Por ello, en esta investigación se utiliza como una métrica complementaria, mientras que la selección final de configuraciones considera conjuntamente tiempo de ejecución, energía y error numérico mediante el análisis multiobjetivo descrito en la Sección «Síntesis: los tres ejes del compromiso».

2.1.6 Síntesis: los tres ejes del compromiso

Las subsecciones anteriores establecen las tres dimensiones que articulan esta investigación: tiempo de ejecución, energía consumida y error numérico. Estas magnitudes deben analizarse conjuntamente, ya que una mejora en una de ellas no garantiza una mejora global de la configuración.

El eje temporal depende tanto de la capacidad de cómputo habilitada por los formatos reducidos y los Tensor Cores como de las características del kernel. En operaciones de alta intensidad aritmética, como GEMM, estas unidades pueden aprovecharse con mayor eficacia, mientras que kernels como Stencil 2D pueden estar limitados por el ancho de banda de memoria (Zhang et al., 2025). Aunque estos operadores pueden reformularse mediante operaciones matriciales (Chen et al., 2024), el beneficio de utilizar Tensor Cores debe comprobarse experimentalmente.

El eje numérico se caracteriza mediante las normas L_2 y L_∞, así como mediante la evolución

del *drift* en los esquemas iterativos. El error no depende únicamente del formato utilizado, sino también del algoritmo y de la aritmética efectiva del hardware (Fasi et al., 2021; Khattak & Mikaitis, 2025). Por esta razón, las configuraciones se comparan respecto a una referencia común en FP64.

El eje energético considera la energía consumida durante la ejecución y, de forma complementaria, el *Energy-Delay Product* (EDP). Debido a las limitaciones temporales de la telemetría energética, especialmente para kernels de corta duración (Yang, Adámek, & Armour, 2024), las mediciones se realizan mediante repeticiones y se analiza su variabilidad antes de establecer comparaciones.

Estas tres dimensiones pueden presentar relaciones de intercambio. Una reducción de precisión puede mejorar el tiempo o la energía, pero también aumentar el error numérico; de forma similar, mecanismos de compensación pueden reducir el error a costa de trabajo adicional. Por ello, no se supone la existencia de una única configuración óptima.

Sea $\mathcal{C}$ el conjunto de configuraciones evaluadas. Una configuración c_1 domina a c_2 si

$$T(c_1) \leq T(c_2) \ \wedge \ E(c_1) \leq E(c_2) \ \wedge \ \mathrm{Error}(c_1) \leq \mathrm{Error}(c_2), \tag{10}$$

con desigualdad estricta en al menos uno de los tres criterios. El Frente de Pareto está formado por las configuraciones que no son dominadas por ninguna otra alternativa del conjunto evaluado. De esta manera, el análisis conserva por separado tiempo, energía y error, permitiendo seleccionar posteriormente una configuración de acuerdo con las prioridades y la tolerancia numérica de la aplicación.

La contribución de este trabajo consiste en evaluar conjuntamente estas tres dimensiones sobre los mismos kernels, plataforma y condiciones experimentales, con el propósito de caracterizar los compromisos asociados al uso de precisión mixta.

2.2 Estado del arte

La evolución reciente de las arquitecturas de cómputo ha modificado el papel de la precisión numérica en aplicaciones científicas. Dongarra *et al.* (2024) señalan que las tendencias arquitectónicas han ampliado de forma marcada la brecha de rendimiento entre la doble precisión

y las unidades matriciales especializadas de menor precisión. En este contexto, la precisión mixta ha pasado a considerarse una variable de diseño para mejorar el aprovechamiento del hardware. Abdelfattah *et al.* (2021) recopilan técnicas de aritmética mixta aplicadas al álgebra lineal, mientras que Higham y Mary (2022) muestran que las cotas deterministas de peor caso pueden resultar conservadoras frente al comportamiento observado en numerosos cálculos. Kashi *et al.* (2024) señalan, sin embargo, que la incorporación de estas estrategias en aplicaciones científicas de producción continúa siendo más limitada que su desarrollo teórico.

Uno de los ejemplos más consolidados es el refinamiento iterativo en precisión mixta. Haidar *et al.* (2018, 2020) demostraron que es posible ejecutar etapas costosas en baja precisión sobre Tensor Cores y recuperar posteriormente exactitud mediante correcciones calculadas en una precisión superior, bajo determinadas condiciones de convergencia. Este principio se ha extendido a aproximaciones de bajo rango, representaciones tensoriales y problemas de estructura electrónica (Baboulin et al., 2024; Habib et al., 2024; Liu et al., 2024). Estos trabajos muestran que la precisión puede distribuirse selectivamente entre etapas del algoritmo, siempre que exista un mecanismo para identificar o corregir el error introducido. En operadores iterativos donde la salida de una etapa alimenta directamente a la siguiente, esta estructura no siempre está disponible y el error puede propagarse a lo largo de la evolución.

Paralelamente, diversos trabajos han estudiado el comportamiento aritmético de los Tensor Cores. Markidis *et al.* (2018) mostraron tempranamente que el aumento de rendimiento puede ir acompañado de pérdida de exactitud. Fasi *et al.* (2021) analizaron el tratamiento de subnormales, el redondeo y el orden de acumulación, identificando diferencias respecto al modelo IEEE 754 convencional, mientras que Khattak y Mikaitis (2025) extendieron esta caracterización a generaciones posteriores de GPU. Desde el análisis numérico, Blanchard *et al.* (2020) formularon cotas de error para operaciones *block FMA* en precisión mixta y Fasi *et al.* (2023) mostraron que distintos esquemas aritméticos pueden presentar comportamientos diferentes aun bajo cotas teóricas similares. En conjunto, estos estudios evidencian que el formato declarado no determina por sí solo la precisión efectiva de una implementación y que resulta necesario contrastar empíricamente sus resultados

con una referencia de mayor precisión.

Esta necesidad es especialmente relevante en esquemas iterativos. Croci y Giles (2022), al estudiar la ecuación del calor en precisión reducida, muestran que el mecanismo de redondeo puede modificar significativamente la evolución temporal del error. En estos casos, una única métrica final no permite observar cuándo comienza a crecer la desviación ni cómo se propaga entre iteraciones. Las estrategias de compensación y asignación selectiva de precisión buscan limitar esta degradación sin regresar completamente a formatos de mayor costo. Métodos como la suma compensada de Kahan (Higham, 2002) recuperan parte del error perdido durante las acumulaciones, mientras que El Arar *et al.* (2025) y otros trabajos recientes (Chen et al., 2026; Yang, Zhao, et al., 2024) asignan la precisión según la sensibilidad numérica de cada operación. Estas técnicas pueden mejorar la exactitud, aunque introducen costos adicionales que también deben evaluarse.

El uso de Tensor Cores se ha extendido más allá de la multiplicación de matrices densas. Chen *et al.* (2024) muestran que operadores *stencil* pueden reformularse mediante operaciones matriciales; Cui (2024) estudia estrategias similares en productos tensoriales de elementos finitos; y en convolución se han propuesto implementaciones basadas en transformaciones de Winograd y fusión de operaciones (Zhang et al., 2026; Zhang et al., 2024). Asimismo, Lu *et al.* (2024) exploran bloques densos dentro de operaciones originalmente dispersas. Estos trabajos muestran que los Tensor Cores pueden aprovecharse en diversos patrones de cómputo mediante reformulaciones algorítmicas.

Sin embargo, la posibilidad de utilizar Tensor Cores no garantiza una aceleración efectiva. Zhang *et al.* (2025) muestran que, cuando un kernel está limitado por el ancho de banda de memoria, incrementar la capacidad aritmética no elimina necesariamente el cuello de botella. Kashi *et al.* (2025) destacan igualmente que parte del beneficio de la precisión reducida puede provenir de disminuir el volumen de datos transferidos y no solo de aumentar el *throughput*. Esta dependencia resulta relevante al comparar GEMM, Convolución 2D y Stencil 2D: GEMM presenta alta reutilización de datos e intensidad aritmética, mientras que Stencil 2D suele estar más condicionado por el movimiento de datos, y la convolución presenta un comportamiento dependiente de la reuti-

lización y organización de memoria. Por ello, los resultados obtenidos sobre GEMM no pueden extrapolarse directamente a kernels con patrones de cómputo diferentes.

La dimensión energética ha recibido una atención creciente, aunque con menor integración con el análisis numérico. Mukunoki y Ogita (2020) muestran que las configuraciones que minimizan el tiempo de ejecución no necesariamente minimizan la energía. Desde la arquitectura, Atoofian (2023) y Hafezan y Atoofian (2023) proponen mecanismos para reducir el consumo de Tensor Cores, principalmente en aprendizaje profundo. En aplicaciones científicas, Suarez *et al.* (2025) destacan la energía como una restricción creciente en HPC, mientras que Godoy *et al.* (2025) caracterizan tiempo, potencia y energía en códigos científicos sobre diferentes plataformas. En la misma línea, Gedik *et al.* (2026) evalúan la eficiencia energética de cómputos en precisión mixta. Estos resultados indican que una aceleración no implica necesariamente un ahorro energético proporcional. La medición energética introduce, además, limitaciones experimentales. Yang *et al.* (2024) caracterizan la resolución temporal de los sensores de potencia accesibles mediante NVML, especialmente en ejecuciones cortas, mientras que van der Vlugt *et al.* (2025) presentan instrumentación externa de mayor resolución. Esta evidencia justifica considerar la duración de la ventana de medición, realizar múltiples repeticiones y analizar la variabilidad antes de establecer diferencias energéticas entre configuraciones.

En conjunto, la literatura revisada muestra avances en tres líneas principales: el uso de precisión mixta y Tensor Cores para mejorar el rendimiento, el análisis de la degradación y propagación del error numérico, y la caracterización del consumo energético. Sin embargo, estos aspectos se estudian con frecuencia de forma separada o sobre familias específicas de aplicaciones. En particular, son limitados los trabajos que comparan, bajo una misma plataforma, kernels con patrones de cómputo diferentes considerando simultáneamente tiempo de ejecución, energía y error respecto a una referencia común. En operadores iterativos, además, el error suele resumirse mediante métricas finales, mientras que su evolución durante la ejecución recibe menor atención.

Entre los trabajos revisados, Godoy *et al.* (2025) se aproxima a una evaluación integrada de rendimiento y energía en aplicaciones científicas, mientras que contribuciones como Chen *et*

al. (2026) incorporan precisión y consumo energético sobre familias específicas de métodos. No obstante, en la literatura revisada no se identificó una evaluación sistemática que reúna, dentro de un mismo entorno experimental, diversidad de patrones de cómputo, hardware tensorial, medición energética y seguimiento del error iterativo.

Este trabajo aborda esa brecha mediante la evaluación conjunta de GEMM, Convolución 2D y Stencil 2D bajo diferentes configuraciones de precisión. El análisis considera tiempo de ejecución, energía y error numérico respecto a una referencia en FP64, e incorpora el seguimiento del *drift* y mecanismos de compensación en el operador iterativo. Así, se caracterizan los compromisos y configuraciones no dominadas en el espacio tiempo-energía-error.

3 Metodología

Este capítulo describe la metodología experimental seguida para estudiar cómo el uso de precisión mixta afecta el rendimiento, el consumo de energía y la exactitud numérica en distintos kernels representativos de HPC. Los experimentos se realizaron sobre una GPU NVIDIA con soporte para Tensor Cores y se organizaron en seis etapas: (i) definición del diseño experimental y del entorno de ejecución, (ii) obtención de las líneas base en FP64 y FP32, (iii) implementación de las distintas alternativas de precisión mixta, (iv) análisis del error numérico y del *drift*, (v) evaluación del comportamiento energético y análisis estadístico de los resultados, y (vi) análisis conjunto de las métricas mediante el Frente de Pareto.

Con el fin de que las comparaciones fueran consistentes, en cada conjunto de experimentos se conservaron las mismas condiciones de entrada, dimensiones del problema y parámetros de ejecución. Esto permitió analizar las diferencias entre las configuraciones procurando que los cambios observados estuvieran relacionados principalmente con la precisión empleada y con la ruta de cómputo utilizada.

3.1 Diseño experimental y plataforma de evaluación

3.1.1 Selección de kernels y configuraciones de precisión

Se seleccionaron tres kernels con patrones de cómputo diferentes:

- GEMM: alta intensidad aritmética y alta reutilización de los datos de entrada.
- Convolución 2D: comparte con GEMM la estructura de productos matriciales y presenta, además, reutilización de datos con un patrón espacial.
- Stencil 2D: menor intensidad aritmética que los dos anteriores; su rendimiento depende principalmente del acceso a memoria, y la salida de cada iteración es la entrada de la siguiente.

Cada kernel se evaluó en cuatro formatos: FP64, FP32, FP16 y BF16. Los resultados en FP64 se utilizan como referencia numérica. En las operaciones que lo permiten, las rutas de 16 bits acumulan en FP32 para reducir la pérdida de exactitud. Los resultados en FP32 constituyen la línea base de rendimiento frente a la cual se comparan los dos formatos de 16 bits.

3.1.2 Plataforma de hardware y entorno de software

Todas las evaluaciones se ejecutaron en el servidor `paccaA100.unicartagena.edu.co` del sistema PACCA, equipado con una GPU NVIDIA A100-PCIE-40GB de arquitectura Ampere (capacidad de cómputo 8.0, 108 SM). Los códigos se compilaron para la arquitectura `sm_80`. Se utilizaron las bibliotecas cuBLAS y cuDNN, la API WMMA y kernels CUDA propios; la ruta basada en CUTLASS se verificó sobre otra GPU y no forma parte de la campaña reportada (Sección «Alcances»). NVIDIA Nsight Compute se empleó para obtener métricas microarquitectónicas y verificar el uso de Tensor Cores (Sección «Verificación de Tensor Cores con Nsight Compute»). El consumo energético de la GPU se midió mediante NVML; el acceso a los contadores RAPL de la CPU anfitriona estuvo restringido en el nodo, por lo que no se reportan mediciones de energía de la CPU.

3.1.3 Espacio de parámetros evaluado

Los factores del estudio son el kernel, el formato de precisión y la implementación. En GEMM y Convolución 2D se evaluaron dos tipos de implementación: las bibliotecas del fabricante (cuBLAS y cuDNN), que representan el rendimiento alcanzable con código optimizado, y kernels propios basados en WMMA, que permiten controlar explícitamente el uso de los recursos de la

GPU. En Stencil 2D se evaluaron un kernel CUDA convencional y una variante que utiliza Tensor Cores mediante WMMA.

El tamaño del problema se trató de dos formas. En las mediciones de rendimiento y de métricas microarquitectónicas se seleccionaron tamaños que mantienen la GPU en régimen estacionario durante la mayor parte de la ejecución, de modo que las fases de inicialización y finalización tengan un peso despreciable. En los estudios de error y de energía, el tamaño se incluyó como factor experimental; los tamaños mayores garantizan además una duración de ejecución suficiente para obtener lecturas de energía válidas. En Convolución 2D, además del barrido $H = W \in \{64, 128, 256, 512\}$ con $C = K = 64$ usado para la línea base y su caracterización por tamaño, la comparación de precisión mixta de la Fase 2 (Tabla 11) usa un punto operativo de mayor intensidad aritmética: $N = 1$, $C = K = 1024$, $H = W = 256$, $R = S = 3$.

3.1.4 Protocolo general de ejecución

Todas las configuraciones de un mismo experimento utilizaron los mismos datos de entrada. Antes de cada medición se realizaron ejecuciones de calentamiento para excluir la creación del contexto CUDA y la inicialización de las bibliotecas, seguidas de una sincronización del dispositivo. Los tiempos se midieron con eventos CUDA. Para cuantificar la variabilidad entre ejecuciones independientes se utilizó la campaña de variabilidad de la Fase 4 ($n = 8$ réplicas por configuración, Tabla 15); las campañas de producción de las fases 1 a 3 y el análisis multiobjetivo de la Fase 4 registran una única medición por configuración, sin réplicas independientes. En PACCA no fue posible fijar la frecuencia de reloj de la GPU, por lo que los resultados pueden verse afectados por el escalado dinámico de frecuencia (DVFS); la variabilidad resultante se cuantificó mediante las ocho réplicas de la campaña de Fase 4.

3.2 Construcción de las líneas base FP64 y FP32

3.2.1 Implementaciones de referencia

Las líneas base se construyeron en FP64 y FP32. GEMM se implementó mediante cuBLAS, Convolución 2D mediante cuDNN y Stencil 2D mediante kernels CUDA personalizados. Estas configuraciones se utilizaron como punto de comparación para las posteriores variantes de precisión

reducida.

Las ejecuciones FP64 y FP32 se definieron sin solicitar explícitamente rutas de Tensor Core desde el código de usuario. No obstante, dado que las bibliotecas de alto nivel pueden seleccionar internamente rutas tensoriales según la arquitectura y la configuración, la unidad de cómputo efectivamente utilizada se verificó posteriormente mediante perfilamiento, en lugar de inferirse únicamente a partir del formato nominal.

3.2.2 Validación numérica

La corrección funcional de los kernels se verificó comparando sus resultados con una referencia en FP64. Para cada configuración se conservaron las mismas entradas y dimensiones, de forma que la diferencia observada correspondiera al efecto de la precisión y no a cambios en los datos.

En GEMM y Convolución 2D se calcularon métricas globales de error al finalizar la operación. En Stencil 2D, además de verificar la salida final, se conservaron estados intermedios para permitir el análisis iterativo desarrollado posteriormente.

3.2.3 Caracterización inicial del rendimiento

La línea base se caracterizó mediante tiempo de ejecución, latencia y *throughput* efectivo. Para GEMM y Convolución 2D se utilizó el número de operaciones de punto flotante para expresar el rendimiento en GFLOP/s o TFLOP/s. En Stencil 2D se consideró adicionalmente el ancho de banda efectivo, debido a su mayor dependencia del movimiento de datos.

3.3 Integración de precisión mixta y Tensor Cores

3.3.1 Implementaciones FP16 y BF16

A partir de las líneas base se construyeron variantes con operandos FP16 y BF16. Cuando la operación lo permitió, la acumulación se mantuvo en FP32 para limitar la pérdida de exactitud asociada a la reducción del formato.

Para garantizar comparabilidad, las variantes reducidas conservaron la misma formulación matemática y los mismos datos de entrada de sus correspondientes líneas base.

3.3.2 Uso de cuBLAS, cuDNN, CUTLASS y kernels personalizados

GEMM utilizó las funcionalidades de precisión mixta disponibles en cuBLAS, una ruta opcional basada en CUTLASS y una implementación personalizada basada en WMMA. Convolución 2D empleó cuDNN, la misma ruta opcional de CUTLASS y una variante personalizada orientada al uso de operaciones matriciales. En Stencil 2D se desarrollaron implementaciones CUDA específicas para su estructura iterativa, incluyendo una variante con WMMA.

Durante la optimización se consideraron la alineación de memoria, la coalescencia de accesos, la reutilización de datos, el uso de memoria compartida y la configuración de bloques, hilos y *warps*.

3.3.3 Verificación de Tensor Cores con Nsight Compute

La activación efectiva de Tensor Cores se verificó mediante NVIDIA Nsight Compute. El perfilamiento permitió distinguir entre rutas convencionales y tensoriales y analizar métricas asociadas al uso de unidades de cómputo, ocupación y tráfico de memoria.

Esta verificación fue necesaria porque el formato declarado por la aplicación no permite determinar por sí solo qué ruta aritmética seleccionó una biblioteca optimizada. Se perfilaron el kernel WMMA propio de GEMM (instanciación FP16) y la ruta FP64 de cuBLAS. Las rutas FP16 de cuBLAS y cuDNN con Tensor Cores no se perfilaron: su uso de instrucciones HMMA se infiere del modo de invocación de la biblioteca y no de contadores de hardware, y así se indica en las tablas de la Fase 2.

3.3.4 Comparaciones oficiales de rendimiento para GEMM y Convolución 2D

Para evitar fusionar en una sola cifra de *speedup* efectos que miden cosas distintas, los resultados de GEMM y Convolución 2D del apartado «Resultados y discusión» se reportan como tres comparaciones separadas y explícitamente rotuladas, nunca combinadas en un único número:

- Comparación (a): cuBLAS/cuDNN en FP32 escalar *vs.* cuBLAS/cuDNN con Tensor Cores (FP16/BF16) — efecto de la precisión, biblioteca contra biblioteca. **Esta es la comparación oficial del Objetivo Específico 2.**

- Comparación (b): biblioteca FP32 *vs.* kernel WMMA propio — brecha de implementación entre una biblioteca madura y un kernel de control explícito escrito para este trabajo; se reporta como análisis secundario.

- Comparación (c): kernel WMMA propio sin compensación *vs.* WMMA propio con mecanismo de compensación numérica (Sección «Esquemas de compensación numérica evaluados») — efecto de la técnica de corrección de error (Objetivo Específico 3).

Las comparaciones (a) y (b) se reportan en filas distintas de las Tablas 9, 10 y 11; la comparación (c) se reporta en la Fase 3 (Sección «Efecto y costo de la compensación local»).

3.4 Evaluación del error numérico y drift

3.4.1 Referencia numérica y métricas de error

La ejecución en FP64 se utilizó como referencia numérica de mayor precisión. Los resultados de FP32, FP16 y BF16 se compararon mediante las normas L_2 y L_∞ (Ecuaciones (4) y (5)), definidas en la Sección «Propagación del error numérico y drift en esquemas iterativos».

Para cada comparación se mantuvieron constantes las condiciones iniciales, dimensiones y parámetros del kernel. Las métricas obtenidas se almacenaron junto con la configuración correspondiente para relacionar posteriormente el error con el rendimiento y la energía.

3.4.2 Seguimiento iterativo de los tres kernels

Los tres kernels se ejecutaron con encadenamiento real: la salida de cada aplicación del operador se utilizó como entrada de la siguiente ($X^{(n+1)} = X^{(n)}A$ en GEMM, $X^{(n+1)} = \mathrm{conv}(X^{(n)}, W)$ en Convolución 2D y $u^{(n+1)} = \mathcal{L}\,u^{(n)}$ en Stencil 2D). Esto permitió evaluar la propagación del error a lo largo de la trayectoria y no únicamente el error de una operación aislada. Los operadores se construyeron con coeficientes exactamente representables en FP16 y BF16 (matriz de Hadamard escalada en GEMM, Laplaciano de cinco puntos en Convolución 2D y Stencil 2D), de modo que el único error introducido por la reducción de precisión fuese el de la cuantización del estado y no el de la representación del operador.

En los puntos de observación se registraron las métricas $D_2^{(p)}(n)$ y $D_\infty^{(p)}(n)$ respecto a la

trayectoria FP64, así como la posición del elemento de máximo error. A partir de estos registros se construyeron las curvas de *drift* y se determinó el horizonte en el que cada formato mantuvo resultados finitos.

3.4.3 Esquemas de compensación numérica evaluados

Se evaluaron tres mecanismos de compensación, todos basados en conservar en FP32 el residuo de almacenamiento del estado, es decir, la diferencia entre el valor exacto y su cuantización al formato de 16 bits.

En GEMM y Convolución 2D se evaluó la *compensación local*: el término principal se calcula sobre el estado cuantizado y una corrección independiente sobre el residuo, y ambos se suman antes de volver a cuantizar, según $(T + c)\, A = TA + cA$. En Stencil 2D se evaluaron dos variantes. La *compensación espacial* realimenta el residuo de cada celda sobre la vecindad del operador, de modo que las celdas vecinas lo leen al aplicar el operador. La variante *Kahan local* resta, antes de cuantizar la nueva escritura de cada celda, el residuo de la escritura anterior de esa misma celda; corrige solo el redondeo propio y no reincorpora el residuo de las vecinas. El nombre se conserva por la opción del código que la activa, pero ninguno de los tres mecanismos es una suma compensada de Kahan en sentido estricto: el residuo no acumula el error de una reducción, sino el de la representación del estado entre iteraciones.

Las variantes con y sin compensación se compararon utilizando las mismas condiciones iniciales y el mismo número de iteraciones. El análisis consideró tanto la reducción del error como el costo computacional y de memoria introducido por la compensación.

3.4.4 Anclaje periódico en FP64

Además de la compensación se evaluó el anclaje periódico en FP64. El estado encadenado se mantiene como el par (T, c), donde T es el buffer en el formato de 16 bits y c el residuo de almacenamiento, con la convención $Q(v) + c = v$ siendo Q la cuantización al formato reducido. La periodicidad K determina en qué iteraciones el paso se evalúa en doble precisión. En una iteración de anclaje se reconstruye el estado exacto conocido hasta ese punto, $v = Q^{-1}(T) + c$, se aplica el operador completo en FP64 sobre ese estado reconstruido, y el resultado se vuelve a

cuantizar a 16 bits conservando el nuevo residuo. El paso en doble precisión *sustituye* al paso de baja precisión de esa iteración; no se añade a él.

El estado de partida del paso en FP64 es, por tanto, el de la propia ruta de precisión reducida y no el de la trayectoria de referencia. El mecanismo corrige el error de redondeo de la iteración anclada y no reconstruye el error acumulado en las iteraciones anteriores, de modo que no requiere conocer la trayectoria de referencia y es aplicable sin ella. Con $K = 0$ el anclaje queda deshabilitado y la ejecución es equivalente a la de la Fase 3. Con $K = 1$ todas las iteraciones evalúan el operador en FP64, de modo que no permanece ningún paso aritmético en 16 bits y la ruta se reduce a doble precisión con el estado almacenado en 16 bits más su residuo.

El anclaje requiere la compensación activa, porque el residuo es lo que permite reconstruir el estado exacto de entrada del paso en FP64. Las configuraciones con $K > 0$ se comparan, en consecuencia, contra la ruta compensada sin anclaje ($K = 0$) del mismo formato, y no contra la ruta sin compensar. El residuo se conserva en doble precisión durante la re-siembra que sigue a un paso anclado: conservarlo en FP32 introduciría en ese punto un truncamiento ausente de la referencia e impediría que $K = 1$ reprodujera la trayectoria de doble precisión. Esta elección eleva el estado a 10 bytes por celda en FP16 (2 del buffer y 8 del residuo), por encima de los 8 bytes por celda de una ruta FP64, de modo que en las configuraciones ancladas la ventaja de la precisión reducida deja de provenir del volumen de memoria del estado y queda limitada al rendimiento aritmético.

La exactitud observable de una ruta anclada está acotada por el campo de lectura y no por el paso en FP64. En Stencil 2D, el estado propagado por la compensación espacial se reconstruye como $Q^{-1}(T) + c$ en FP32 antes de compararlo con la referencia en FP64; la referencia en doble precisión, en cambio, se compara sin esa conversión. Cuando el error interno cae por debajo de la resolución del campo de lectura, la métrica reportada queda limitada por un piso de instrumentación del orden de la unidad de redondeo de FP32, $u = 2^{-24} \approx 5{,}96 \times 10^{-8}$. Es lo que ocurre con $K = 1$: los valores reportados ($3{,}094 \times 10^{-8}$ en la malla de 4096^2 y $2{,}634 \times 10^{-8}$ en la de 8192^2, esto es $0{,}52$ y $0{,}44$ veces u) miden el redondeo del campo de lectura y no la exactitud interna del mecanismo, cuyo estado reproduce la trayectoria de doble precisión. Que los dos formatos de

16 bits coincidan allí en cinco cifras significativas, pese a que sus unidades de redondeo difieren en un factor de ocho, confirma que la magnitud publicada es independiente del formato almacenado. Esta limitación afecta a la interpretación del error de esas configuraciones; el tiempo y la energía no se ven afectados.

3.5 Medición energética y análisis estadístico

3.5.1 Protocolo de telemetría energética

La energía de la GPU se caracterizó mediante NVML, utilizando lecturas del contador energético antes y después de la ventana de ejecución. Debido a la resolución temporal de la telemetría, los kernels cuya duración individual era demasiado corta se ejecutaron repetidamente dentro de una misma ventana, de modo que el intervalo acumulado fuera suficientemente largo para producir una lectura válida.

La energía por ejecución se obtuvo dividiendo la diferencia acumulada entre el número de lanzamientos incluidos en la ventana. Las lecturas nulas o inválidas se excluyeron del análisis. A partir de la energía y del tiempo se calculó además el *Energy-Delay Product* definido en la Ecuación (9).

3.5.2 Repeticiones y control de variabilidad

No fue posible bloquear las frecuencias de reloj de la GPU mediante las interfaces disponibles en PACCA. Por este motivo, la variabilidad inducida por el comportamiento dinámico del hardware se cuantificó experimentalmente mediante repeticiones por configuración.

Como medida descriptiva se utilizó el coeficiente de variación,

$$C_v = \frac{s}{\bar{x}} \times 100, \tag{11}$$

donde $\bar{x}$ es la media y s la desviación estándar de las repeticiones.

3.5.3 Tratamiento estadístico

Para evaluar la consistencia de las diferencias observadas entre configuraciones, se analizaron las repeticiones experimentales con un nivel de significancia de $\alpha = 0{,}05$. Los

esquemas de compensación de Stencil 2D (ninguna, Kahan local, espacial) se compararon sobre la campaña de variabilidad mediante el tiempo mediano por iteración y el error de cada uno (Tabla 17); como el error es determinista y las diferencias de tiempo superan con amplitud la variabilidad entre réplicas, esa comparación no requirió prueba estadística (Sección «Estabilidad y variabilidad experimental»). Para el efecto de la periodicidad de anclaje K en los tres kernels se utilizó una prueba t pareada de una muestra sobre la razón logarítmica $\ln[T(K)/T(K = 0)]$ por réplica, con $n = 8$ pares y gl $= 7$. La prueba compara cada réplica consigo misma y la transformación logarítmica simetriza la razón; supone normalidad de las diferencias logarítmicas, que no se verificó formalmente con $n = 8$. Cuando se comparó más de una periodicidad por kernel y formato, los valores p se corrigieron por la familia de comparaciones mediante el método de Holm-Bonferroni (Sección «Efecto de la frecuencia de anclaje sobre el tiempo»).

3.6 Análisis multiobjetivo

3.6.1 Construcción del espacio tiempo–energía–error

Las métricas obtenidas en las etapas anteriores se integraron en un espacio de tres dimensiones formado por el tiempo de ejecución, la energía consumida y el error numérico. Cada punto del espacio representa una configuración experimental definida por el kernel, la precisión y la implementación utilizada.

El EDP se conservó como una métrica complementaria para interpretar conjuntamente tiempo y energía, pero no sustituyó estos dos ejes en el análisis multiobjetivo.

3.6.2 Identificación del frente de Pareto

Para cada conjunto comparable de configuraciones se aplicó el criterio de dominancia definido en la Ecuación (10) (Sección «Síntesis: los tres ejes del compromiso»). Una configuración c_1 domina a otra configuración c_2 cuando presenta valores iguales o menores de tiempo, energía y error, y es estrictamente mejor en al menos uno de los tres criterios.

Las configuraciones no dominadas conformaron el Frente de Pareto. Este análisis permitió conservar explícitamente los compromisos entre rendimiento, consumo energético y exactitud numérica, sin imponer una ponderación única entre las tres dimensiones.

3.7 Alcances y limitaciones

3.7.1 Alcances

Esta investigación evalúa experimentalmente esquemas de precisión mixta sobre GPU NVIDIA con soporte para Tensor Cores. Los experimentos se realizaron en un entorno intra-nodo sobre el sistema PACCA, con una GPU NVIDIA A100-PCIE-40GB de arquitectura Ampere y una CPU Intel como anfitrión.

Como casos de estudio se seleccionaron tres kernels representativos de distintos patrones de cómputo en HPC: GEMM, de alta intensidad aritmética; Convolución 2D, con reutilización espacial de datos; y Stencil 2D, dominado por el movimiento de memoria. Los tres se ejecutaron bajo encadenamiento iterativo real, en el que la salida de una iteración alimenta la siguiente (Sección «Seguimiento iterativo de los tres kernels»), lo que permitió estudiar la propagación del error en los tres casos. Stencil 2D es el único kernel con iteraciones suficientes para que todos los formatos, incluido FP64, alcancen su horizonte de desbordamiento. Los tamaños de problema se ampliaron respecto al rango previsto (512 a 4096 en GEMM, 512^2 a 2048^2 en Stencil 2D) hasta 32768 y 16384^2, respectivamente, para alcanzar un régimen de utilización estable y ventanas suficientes para la telemetría energética (Sección «Métricas de eficiencia energética»).

Se evaluaron cuatro formatos de punto flotante: FP64 como referencia numérica, FP32 como línea base, y FP16 y BF16 como formatos reducidos, combinados en las rutas de precisión mixta con acumulación en FP32 cuando la operación y la implementación lo permitieron.

Las implementaciones se desarrollaron sobre el ecosistema CUDA: GEMM mediante cuBLAS y una implementación personalizada basada en WMMA; Convolución 2D mediante cuDNN y una variante WMMA de mayor control; y Stencil 2D mediante kernels CUDA personalizados basados en WMMA. Los dos primeros disponen de una ruta CUTLASS opcional, verificada funcionalmente sobre una GPU distinta a la de la campaña reportada, por lo que el apartado «Resultados y discusión» no incluye mediciones de esta ruta. Puesto que las bibliotecas optimizadas pueden seleccionar internamente distintas rutas aritméticas, la utilización efectiva de Tensor Cores se verificó con NVIDIA Nsight Compute y no se infirió del formato nominal.

La evaluación de rendimiento consideró el tiempo de ejecución y el *throughput* efectivo, complementados con métricas microarquitectónicas de perfilamiento. La caracterización energética de la GPU se realizó mediante telemetría NVML; el acceso a RAPL de la CPU anfitriona estuvo restringido por los permisos del sistema, por lo que no se reportan mediciones de energía de la CPU. La degradación numérica se cuantificó mediante las normas L_2 y L_∞ con FP64 como referencia, sobre operaciones aisladas y sobre la evolución del *drift* en el esquema encadenado, evaluando además mecanismos de compensación: local en GEMM y Convolución 2D, y espacial y Kahan local en Stencil 2D.

3.7.2 Limitaciones

El estudio se restringió a FP64, FP32, FP16 y BF16, quedando fuera los formatos enteros (INT8, INT4) y FP8. El error se midió tomando como referencia el resultado en FP64, no la solución analítica exacta; las magnitudes reportadas son desviaciones respecto a esa referencia. El análisis cubre únicamente el error determinista de redondeo, y no la fiabilidad de los Tensor Cores ante fallas transitorias inducidas por radiación (*soft errors*), un problema distinto abordado en (Basso et al., 2020; Netti et al., 2023).

Todas las mediciones se realizaron sobre una única GPU Ampere, sin escenarios multi-GPU ni ejecución distribuida mediante MPI, para aislar el efecto de la precisión sin que intervinieran los costos de comunicación. Por la misma razón no se implementó selección dinámica de precisión: cada configuración se definió previamente y se evaluó de forma independiente, de modo que las conclusiones comparan alternativas estáticas. Los resultados no son extrapolables de forma directa a aceleradores de otros fabricantes ni a generaciones microarquitectónicamente distintas. En Stencil 2D, la ausencia de ganancia con Tensor Cores corresponde a la implementación WMMA evaluada, que carga cinco tiles completos por punto sin compartir el halo entre vecinos y escribe el resultado en FP32; una formulación que comparta el halo, como ConvStencil (Chen et al., 2024), podría cambiar ese resultado y no se evaluó aquí.

La instrumentación impone limitaciones adicionales. En PACCA no fue posible fijar la frecuencia de reloj de la GPU, por lo que las ejecuciones quedaron sujetas a DVFS, que introduce

variabilidad en tiempo, potencia y energía; para acotarlo se ejecutaron múltiples repeticiones por configuración, se descartaron las ejecuciones de calentamiento y se analizó la dispersión de las mediciones. La resolución temporal finita de NVML afecta sobre todo a los kernels cortos; se compensó ampliando las ventanas de medición e incorporando repeticiones internas. Las mediciones energéticas son, por tanto, válidas para comparar configuraciones entre sí, no como valores absolutos.

La campaña no registró la frecuencia de reloj efectiva del SM durante la ejecución. `cudaDeviceGetAttribute(cudaDevAttrClockRate)` proporciona el valor de «Reloj GPU» impreso en cada trabajo, un atributo estático del dispositivo (1 410 MHz, el reloj de *boost* nominal de la A100 PCIe) y no una muestra en tiempo de ejecución, por lo que esas líneas no permiten confirmar a qué frecuencia operó realmente la GPU. Esta limitación es relevante porque la tasa medida en la ruta FP64 de cuBLAS (10,53 TFLOP/s, Tabla 4) equivale al 54 % del pico DMMA nominal a 1 410 MHz, y las rutas cuBLAS en FP32 (10,36 TFLOP/s, 53 % de su pico) y en FP16 con Tensor Cores (166 TFLOP/s, 53,2 % de su pico) caen en la misma fracción pese a ejecutarse sobre tuberías de cómputo distintas (unidad escalar, DMMA y HMMA). Esa coincidencia entre tres rutas independientes es compatible con que la GPU haya operado cerca de su reloj base (765 MHz, el 54,3 % del máximo), pero no puede confirmarse sin una muestra de reloj en tiempo de ejecución, y no se relanzó la campaña para obtenerla. Los porcentajes de pico reportados en este trabajo (Tabla 4 y Sección «Interpretación arquitectónica del rendimiento») toman como referencia el reloj máximo nominal de la A100 PCIe; si el reloj efectivo fue menor, esos porcentajes subestiman la utilización real de las unidades de cómputo.

Finalmente, el análisis de *drift* por iteración (Sección «Evolución del error a lo largo del encadenamiento») y el frente de Pareto tiempo–energía–error del Objetivo Específico 4 se construyeron sobre los tres kernels, con cobertura desigual: dos tamaños en GEMM, cuatro en Convolución 2D y dos mallas en Stencil 2D. En este último el análisis exigió un operador difusivo distinto del de estrés usado en el resto de la evaluación, porque bajo el operador amplificante ninguna precisión conserva la finitud hasta la ventana energética (Sección «Stencil 2D»).

4 Resultados y discusión

Este capítulo presenta y discute los resultados experimentales obtenidos durante las cuatro fases del estudio. En la primera fase se caracterizan las líneas base en FP64 y FP32 para los tres kernels evaluados. La segunda fase analiza la incorporación de FP16 y BF16, la utilización efectiva de Tensor Cores y las ganancias de rendimiento obtenidas. La tercera fase estudia la degradación numérica, con especial atención al *drift*, el horizonte de desbordamiento y las estrategias de compensación en Stencil 2D. Finalmente, la cuarta fase integra tiempo de ejecución, energía y error numérico mediante análisis estadístico y criterios de dominancia de Pareto.

4.1 Fase 1: Construcción y caracterización de las líneas base

La primera fase estableció los puntos de referencia utilizados posteriormente para evaluar las configuraciones de precisión reducida. Se verificó primero la corrección funcional de los kernels y posteriormente se caracterizó el rendimiento de las implementaciones nominales en FP64 y FP32 para GEMM, Convolución 2D y Stencil 2D, contrastando en todos los casos la ruta GPU contra una implementación de referencia en CPU.

4.1.1 Validación funcional de los kernels

Antes de analizar el rendimiento, se verificó la corrección funcional de las implementaciones comparando la salida de la ruta GPU con la salida de una implementación de referencia en CPU, dentro de la misma precisión (FP32 GPU vs. FP32 CPU, y FP64 GPU vs. FP64 CPU). Esta validación, ejecutada como parte de la campaña de caracterización inicial de la Fase 1, permitió comprobar que la ruta GPU reproduce el resultado de la ruta CPU antes de introducir formatos de precisión reducida en las fases siguientes.

Tabla 2

Validación funcional de las líneas base: GPU vs. CPU dentro de la misma precisión.

Kernel	**Configuración**	**L_2 relativo (GPU vs. CPU)**	**Error máx. abs.**
GEMM (FP32)	12288^3	0,000001	$3{,}66 \times 10^{-4}$
GEMM (FP64)	12288^3	$< 10^{-6}$	0,0
GEMM (FP32)	32768^3	0,000002	$1{,}05 \times 10^{-3}$
Convolución 2D (FP32)	$H = W = 64$	$< 10^{-6}$	$2{,}2 \times 10^{-5}$
Convolución 2D (FP32)	$H = W = 512$	$< 10^{-6}$	$2{,}7 \times 10^{-5}$
Stencil 2D (FP32)	4096×4096	0	0
Stencil 2D (FP64)	4096×4096	0	0

Nota. Los binarios de validación de GEMM y Convolución 2D imprimen el error relativo con seis decimales, de modo que un valor impreso como 0,000000 solo permite afirmar que el error es menor que 10^{-6}; se reporta como $< 10^{-6}$ en esta y en las demás tablas de líneas base. En Stencil 2D el error es exactamente nulo.

La ruta GPU reprodujo la referencia de CPU dentro de la misma precisión con error despreciable en los tres kernels. Por tanto, las diferencias numéricas analizadas en las fases posteriores se atribuyen al formato de precisión y no a errores funcionales de implementación.

4.1.2 Rendimiento de la línea base en GEMM

La línea base de GEMM se caracterizó mediante cuBLAS para dos tamaños de problema, contrastando la ejecución en GPU contra una implementación de referencia en CPU multihilo (OpenBLAS, `cblas_sgemm`/`cblas_dgemm`, con `OPENBLAS_NUM_THREADS` igualado al número de CPU asignadas por SLURM, hasta 16). La Tabla 3 resume los resultados.

La ejecución en GPU redujo el tiempo frente a la CPU en ambas precisiones, con *speedups* de $5{,}5$–$6\times$ en FP32 y $11{,}6$–$12{,}1\times$ en FP64, manteniendo un error L_2 relativo prácticamente nulo. Estos resultados establecen las referencias temporales utilizadas posteriormente para cuantificar las ganancias de las rutas de precisión reducida.

4.1.2.1 Verificación de la ruta FP64: la referencia ya usa Tensor Cores. El perfilado con Nsight Compute ($N = 12288$) confirma que la referencia GPU en FP64 activa Tensor Cores

Tabla 3

Línea base de GEMM: comparación CPU vs. GPU en FP32 y FP64.

Caso / Precisión	**CPU (ms)**	**GPU (ms)**	**Speedup**	**TFLOP/s**	**Error L_2 rel.**
$N = 12\,288$ – FP32	2 144,031	358,355	5,98×	10,36	1×10^{-6}
$N = 12\,288$ – FP64	4 263,707	352,486	12,10×	10,53	$< 10^{-6}$
$N = 32\,768$ – FP32	38 732,708	6 990,083	5,54×	10,07	2×10^{-6}
$N = 32\,768$ – FP64	77 438,986	6 664,549	11,62×	10,56	$< 10^{-6}$

(DMMA). cuBLAS despachó un kernel de CUTLASS con Tensor Cores FP64, y el número de instrucciones DMMA ejecutadas, multiplicado por las 512 operaciones de cada instrucción ($8 \times 8 \times 4$ por *warp*), coincide exactamente con las $2N^3$ operaciones del producto: la totalidad del cálculo pasó por la tubería tensorial FP64 (Tabla 4). La tasa de $10{,}53$ TFLOP/s supera así el pico FP64 sin Tensor Cores de la A100 ($9{,}7$ TFLOP/s) sin que ello constituya un error de medición. En consecuencia, las comparaciones frente a las rutas FP16/BF16 contrastan Tensor Cores contra Tensor Cores, no contra una ruta escalar. Esa tasa equivale al $54\,\%$ del pico DMMA nominal a $1\,410$ MHz; por qué las tres rutas de biblioteca caen en torno a esa misma fracción de su pico respectivo, y qué dice eso sobre la frecuencia de reloj efectiva durante la campaña, se discute como limitación en la Sección «Limitaciones».

Tabla 4

Perfil de Nsight Compute de la ruta FP64 de cuBLAS (GEMM, $N = 12288$).

Métrica	**Valor observado**
Kernel despachado por cuBLAS	`cutlass_80_tensorop_d884gemm`
Instrucciones DMMA ejecutadas	7 247 757 312
Operaciones por la tubería tensorial FP64	$3{,}71 \times 10^{12}$ (100 % de $2N^3$)
Ocupación de la tubería tensorial	99,52 %

4.1.3 Rendimiento de la línea base en Convolución 2D

En Convolución 2D se empleó cuDNN como implementación optimizada en GPU, con TF32 desactivado explícitamente en la ruta FP32 (`CUDNN_FMA_MATH`) para que la línea base refleje la unidad escalar y no Tensor Cores. Se barrieron cuatro tamaños de entrada cuadrada ($H = W \in \{64, 128, 256, 512\}$, $N = 1$, $C = K = 64$, núcleo 3×3) en FP32 y FP64, contrastando la ejecución en GPU contra una referencia en CPU basada en *im2col* (transformación serial) seguida de OpenBLAS (`cblas_sgemm`/`cblas_dgemm`, multihilo, misma configuración de hilos que la referencia de GEMM). El rendimiento en GPU alcanzó hasta $18\,267{,}874$ GFLOP/s en FP32 y $2\,228{,}852$ GFLOP/s en FP64 para $H = W = 512$.

La cifra de FP32 equivale al $93{,}7$ % del pico escalar FP32 de la A100 ($19{,}5$ TFLOP/s) y debe leerse como un rendimiento nominal. El algoritmo de convolución se dejó a la selección automática de cuDNN, y el registro de la campaña muestra que en FP32 eligió el algoritmo de Winograd en los cuatro tamaños, mientras que en FP64 eligió `IMPLICIT_PRECOMP_GEMM`. Winograd ejecuta menos multiplicaciones que la convolución directa, pero el rendimiento se calcula con el conteo de operaciones de la convolución directa ($2NKCHWRS$), de modo que el GFLOP/s de FP32 sobrestima el trabajo aritmético realizado, igual que en la ruta FP32 de la Fase 2 antes de fijar el algoritmo (Sección «Rendimiento de Convolución 2D en precisión mixta»). Por la misma razón, los *speedups* GPU/CPU de FP32 de la Tabla 5 y la razón entre FP32 y FP64 en GPU reflejan, además de la precisión, el cambio de algoritmo. La línea base de la Fase 1 refleja así el rendimiento que cuDNN ofrece por defecto; para la comparación oficial de la Fase 2, en cambio, la ruta FP32 se fijó en `IMPLICIT_GEMM` (Sección «Rendimiento de Convolución 2D en precisión mixta»).

Tabla 5

Línea base de Convolución 2D: comparación CPU vs. GPU en FP32 y FP64.

Caso / Precisión	**CPU (ms)**	**GPU (ms)**	**Speedup**	**Error L_2 rel.**
$H = W = 64$ – FP32	16,316	0,053	307,01×	$< 10^{-6}$
$H = W = 64$ – FP64	33,688	0,160	211,16×	$< 10^{-6}$
$H = W = 128$ – FP32	122,940	0,110	1 119,95×	$< 10^{-6}$
$H = W = 128$ – FP64	254,251	0,741	343,09×	$< 10^{-6}$
$H = W = 256$ – FP32	1 019,105	0,315	3 233,33×	$< 10^{-6}$
$H = W = 256$ – FP64	1 303,194	2,326	560,27×	$< 10^{-6}$
$H = W = 512$ – FP32	4 894,800	1,058	4 626,48×	$< 10^{-6}$
$H = W = 512$ – FP64	5 459,403	8,671	629,58×	$< 10^{-6}$

A diferencia de GEMM, en Convolución 2D el *speedup* GPU/CPU no decrece sino que crece con el tamaño del problema: en FP32 pasa de $307{,}01\times$ ($H = W = 64$) a $4\,626{,}48\times$ ($H = W = 512$), y en FP64 de $211{,}16\times$ a $629{,}58\times$, porque el tiempo de CPU escala peor que el de GPU al aumentar la superficie espacial de la convolución. En ambas precisiones el error L_2 relativo GPU vs. CPU permanece despreciable en los cuatro tamaños, consistente con lo observado en GEMM.

4.1.4 Rendimiento de la línea base en Stencil 2D

El comportamiento de Stencil 2D difirió del observado en GEMM. Para la configuración $N_X = N_Y = 4096$ con $N_T = 20$ pasos temporales, expresando los tiempos por paso, la implementación FP32 alcanzó $535{,}252$ GFLOP/s en GPU con un ancho de banda efectivo de 857 GB/s, mientras que FP64 alcanzó $388{,}420$ GFLOP/s y $1\,244$ GB/s. La referencia de CPU paraleliza el operador con OpenMP (`#pragma omp parallel for`, `OMP_NUM_THREADS` igualado a las CPU asignadas por SLURM, hasta 16) y no es, por tanto, una referencia serial. Los *speedups* respecto a esa CPU paralela fueron de $174{,}98\times$ y $140{,}51\times$ respectivamente, y en ambos casos el error L_2 relativo fue exactamente nulo.

Tabla 6

Línea base de Stencil 2D ($N_X = N_Y = 4096$, $N_T = 20$, tiempos por paso): comparación CPU paralela (OpenMP) vs. GPU CUDA.

Precisión	CPU (ms)	GPU (ms)	Speedup	Error L_2 rel.
FP32	27,397	0,156570	174,98×	0
FP64	30,316	0,215757	140,51×	0

Expresado como fracción del ancho de banda pico de la memoria HBM2 de la GPU NVIDIA A100, esto corresponde al $55{,}1\,\%$ en FP32 y al $80{,}0\,\%$ en FP64.

Tabla 7

Utilización del ancho de banda en las líneas base de Stencil 2D.

Precisión	Ancho de banda efectivo (GB/s)	Fracción del pico HBM2
FP32	857	$55{,}1\,\%$
FP64	1 244	$80{,}0\,\%$

Stencil 2D alcanzó *speedups* de $140{,}51$–$174{,}98\times$ frente a la CPU paralela (OpenMP) y hasta el $80{,}0\,\%$ del ancho de banda pico del dispositivo, lo que confirma que su rendimiento está determinado por el acceso a memoria. Por esta razón, el análisis posterior de Stencil 2D considera el ancho de banda efectivo junto con las métricas temporales y no únicamente el *throughput* aritmético.

4.1.5 Síntesis de la Fase 1

Las líneas base mostraron comportamientos diferentes entre los tres kernels. GEMM presentó una fuerte componente aritmética con *speedups* moderados ($5{,}5$–$12{,}1\times$); Convolución 2D alcanzó las ganancias mayores (307–$4\,626\times$ en FP32, con el algoritmo de Winograd, y 211–$630\times$ en FP64), crecientes con el tamaño espacial de la entrada; y Stencil 2D, dominado por el movimiento de datos, alcanzó $140{,}51$–$174{,}98\times$. Estas diferencias constituyen el punto de partida para interpretar las ganancias obtenidas mediante precisión reducida y Tensor Cores. Estos *speedups* no son

comparables entre kernels: las tres referencias de CPU son multihilo, pero mediante mecanismos distintos (GEMM y Convolución 2D delegan el paralelismo a OpenBLAS, con el número de hilos igualado a las CPU asignadas por el trabajo de SLURM; Stencil 2D paraleliza directamente con un bucle OpenMP propio), de modo que un mayor *speedup* en un kernel no implica una GPU relativamente más ventajosa: la comparación válida es GPU frente a CPU dentro de cada kernel, no el *speedup* de un kernel frente a otro.

4.2 Fase 2: Precisión mixta y utilización de Tensor Cores

La segunda fase evaluó las implementaciones con operandos de precisión reducida y verificó experimentalmente la utilización del hardware tensorial. Una vez comprobada la ejecución sobre Tensor Cores, se comparó el rendimiento de las rutas de precisión mixta con las referencias establecidas en la Fase 1.

4.2.1 Verificación experimental de Tensor Cores

La utilización efectiva de Tensor Cores se verificó mediante NVIDIA Nsight Compute sobre la instanciación FP16 del kernel `wmma_gemm_kernel` ($m = n = k = 12288$) en una GPU NVIDIA A100-PCIE-40GB. Se registraron 906 millones de instrucciones HMMA y $3{,}71 \times 10^{12}$ operaciones por la ruta tensorial `fp16` $\rightarrow$ `fp32`, que concentró el $100\,\%$ de los FLOP del kernel: evidencia microarquitectónica de que el cálculo se canalizó por las unidades tensoriales y no solo con operandos en precisión reducida.

La totalidad de los FLOP se ejecutó por la ruta `fp16` $\rightarrow$ `fp32`, confirmando que la acumulación se mantiene en FP32 tal como se especificó en el diseño experimental.

4.2.2 Rendimiento de GEMM en precisión mixta

GEMM presentó la mayor aceleración entre los kernels estudiados. Para $N = 12288$, la ruta FP16 mediante cuBLAS con Tensor Cores redujo el tiempo medio de $352{,}487$ ms en la referencia FP64 a $23{,}086$ ms, alcanzando $160\,741{,}22$ GFLOP/s ($160{,}741$ TFLOP/s) y un *speedup* de $15{,}27\times$. Frente a la ruta cuBLAS FP32 escalar ($358{,}354$ ms), que es la comparación oficial (a) de la Sección «Comparaciones oficiales de rendimiento para GEMM y Convolución 2D», la aceleración es de $15{,}52\times$. La implementación WMMA personalizada en FP16 también aceleró el

Tabla 8

Indicadores de utilización de Tensor Cores obtenidos mediante NVIDIA Nsight Compute.

Métrica	**Valor observado**
Ciclos activos de la tubería tensorial	22,50 %
Instrucciones HMMA ejecutadas	905 969 664
Operaciones tensoriales FP16→FP32	$3{,}71 \times 10^{12}$ (100 %)
Operaciones tensoriales BF16→FP32	0
Throughput de los SM	64,12 %
Warps activos	74,77 %
Throughput de DRAM	39,15 %
Bytes leídos de DRAM	58,75 GB
Bytes escritos en DRAM	605,64 MB
Registros por hilo	34

cálculo, con 97,494 ms y 38 062,38 GFLOP/s (38,062 TFLOP/s), equivalente a $3{,}62\times$ y $4{,}22$ veces más lenta que cuBLAS.

Tabla 9

GEMM con precisión mixta y Tensor Cores, $N = 12288$.

Implementación	**Tiempo medio (ms)**	**Rendimiento (GFLOP/s)**	**Speedup vs FP64 GPU**	**Error máx. abs.**	**Error L_2 rel.**	**Tensor Cores**
GPU cuBLAS (FP64) [Referencia]	352,487	10 527,60	—	—	—	Sí (DMMA)
GPU cuBLAS (FP32)	358,354	10 355,26	0,98×	0,013808	0,000002	No (FP32)
GPU cuBLAS TC (FP16)	23,086	160 741,22	15,27×	0,965415	0,000225	Sí (HMMA)†
GPU WMMA custom (FP16)	97,494	38 062,38	3,62×	0,965415	0,000225	Sí (HMMA)

† Uso de HMMA inferido del modo de invocación de cuBLAS, no perfilado con Nsight Compute (Sección «Verificación de Tensor Cores con Nsight Compute»); las filas WMMA y FP64 sí se perfilaron (Tablas 8 y 4). Las filas FP64 y FP32 proceden de una corrida distinta de la de la Tabla 3, por lo que los tiempos difieren en el último dígito.

Para $N = 32768$ se mantuvo el mismo comportamiento: cuBLAS con Tensor Cores alcanzó 423,553 ms y 166 139,16 GFLOP/s (166,139 TFLOP/s), es decir $91{,}82\times$ frente a la CPU y $16{,}50\times$ frente a la ruta GPU clásica en FP32, operando al 53,2 % del pico teórico en FP16. La ruta WMMA

personalizada alcanzó $1\,836{,}201$ ms y $38\,322{,}99$ GFLOP/s ($38{,}323$ TFLOP/s), equivalente a $3{,}81\times$ sobre la ruta GPU clásica, pero solo al $12{,}3\,\%$ del pico teórico.

Tabla 10

GEMM con precisión mixta y Tensor Cores, $N = 32768$.

Implementación	Tiempo (ms)	Rendimiento (GFLOP/s)	% pico teórico	Speedup vs CPU	Speedup vs GPU clásico	Error L_2 rel.	Tensor Cores
CPU BLAS	38 891,891	1 809,34	—	1,00×	—	$< 10^{-6}$	No (CPU)
GPU cuBLAS clásico	6 988,929	10 068,60	51,6 % (FP32)	5,56×	1,00×	0,000005	No (FP32)
GPU cuBLAS TC (FP16)	423,553	166 139,16	53,2 % (FP16)	91,82×	16,50×	0,000308	Sí (HMMA)†
GPU WMMA custom (FP16)	1 836,201	38 322,99	12,3 % (FP16)	21,18×	3,81×	0,000308	Sí (HMMA)

† Uso de HMMA inferido del modo de invocación de cuBLAS, no perfilado (Sección «Verificación de Tensor Cores con Nsight Compute»). Las filas CPU BLAS y cuBLAS clásico proceden de una corrida distinta de la de la Tabla 3 (misma configuración FP32), por lo que sus tiempos y su error difieren ligeramente. $< 10^{-6}$: valor por debajo de la resolución de impresión (Tabla 2).

Los errores máximos absolutos correspondientes fueron $0{,}002974$ (CPU BLAS), $0{,}020150$ (GPU cuBLAS clásico) y $1{,}082245$ en ambas rutas con Tensor Cores. La diferencia entre cuBLAS con Tensor Cores y la implementación WMMA personalizada muestra que activar las unidades tensoriales no garantiza por sí solo el máximo rendimiento: ambas rutas producen el mismo error numérico, pero la biblioteca alcanza un *throughput* más de cuatro veces superior. La diferencia se atribuye a la organización de memoria, el bloqueo de datos (*tiling*) y las optimizaciones internas de la biblioteca. El perfilado con Nsight Compute cubrió el kernel WMMA propio y no la ruta `cublasGemmEx` con Tensor Cores: que esta última ejecute instrucciones HMMA se infiere de su modo de invocación (`CUBLAS_COMPUTE_32F_FAST_16F`) y es consistente con su rendimiento, más de ocho veces superior al pico FP32 escalar de la A100 ($19{,}5$ TFLOP/s), pero no se verificó con contadores de hardware (Sección «Verificación de Tensor Cores con Nsight Compute»). Lo mismo aplica a la ruta tensorial de cuDNN de la sección siguiente.

4.2.3 Rendimiento de Convolución 2D en precisión mixta

La Convolución 2D también presentó una mejora al utilizar la ruta tensorial de cuDNN. Para $C = 1024$, el tiempo de ejecución se redujo de $164{,}095$ ms en la referencia FP64 y $89{,}479$ ms en la ruta FP32 clásica a $5{,}847$ ms en la ruta Tensor Core FP16: un *speedup* de $28{,}07\times$ respecto a FP64 y de $15{,}30\times$ respecto a FP32 escalar con `IMPLICIT_GEMM`. La ruta tensorial alcanzó

211,570 TFLOP/s, con un error L_2 relativo del orden de 10^{-4}. La implementación WMMA personalizada en FP16 obtuvo 25,890 ms y 47,778 TFLOP/s, equivalente a $6{,}34\times$ respecto a FP64.

La referencia FP32 de la comparación oficial (a) se fijó en el algoritmo `IMPLICIT_GEMM` de cuDNN, con TF32 desactivado (`CUDNN_FMA_MATH`). Con la selección automática, cuDNN elige Winograd, cuya tasa aparente (30,96 TFLOP/s) supera el pico escalar FP32 de la A100 por contar menos multiplicaciones de las que atribuye el rendimiento nominal (Sección «Rendimiento de la línea base en Convolución 2D»), de modo que no es comparable operación por operación con la ruta tensorial. `IMPLICIT_GEMM` ejecuta la convolución directa y queda por debajo de ese pico (13,824 TFLOP/s). La elección condiciona la cifra: en la misma corrida, cuDNN FP32 con su algoritmo por defecto tardó 39,950 ms, de modo que la ruta FP16 con Tensor Cores es 6,83 veces más rápida que ese FP32, frente a las 15,30 veces que resultan con `IMPLICIT_GEMM`. La Tabla 11 declara el algoritmo de cada fila.

Tabla 11

Convolución 2D con precisión mixta y Tensor Cores, $C = 1024$.

Implementación	Tiempo medio (ms)	Rendimiento (TFLOP/s)	Speedup vs FP64 GPU	Error máx. abs.	Error L_2 rel.	Tensor Cores
GPU cuDNN (FP64) [Referencia]	164,095	7,538	—	—	—	No (FP64)
GPU cuDNN clásico (FP32)	89,479	13,824	1,83×	0,001318	0,000002	No (FP32)
GPU Tensor Core (FP16)	5,847	211,570	28,07×	0,337585	0,000574	Sí (HMMA)†
GPU WMMA custom (FP16)	25,890	47,778	6,34×	0,270692	0,000538	Sí (HMMA)

† Uso de HMMA inferido del modo de invocación de cuDNN, no perfilado con Nsight Compute (Sección «Verificación de Tensor Cores con Nsight Compute»). Algoritmo de cada fila: FP64, selección automática de cuDNN (el binario no la registra; en la línea base, con el mismo tamaño y un tiempo equivalente, cuDNN eligió `IMPLICIT_PRECOMP_GEMM`); FP32, `IMPLICIT_GEMM` fijado, con TF32 desactivado; Tensor Core FP16, selección automática de cuDNN con Tensor Cores habilitados (no registrada); WMMA, kernel propio (*im2col* y GEMM con WMMA), sin cuDNN. La fila FP32 procede de una corrida posterior a las demás, con la misma configuración y el binario modificado para fijar el algoritmo.

Al igual que en GEMM, el beneficio depende tanto del formato numérico como del nivel de optimización de la implementación: la ruta tensorial de cuDNN es 4,43 veces más rápida que la implementación WMMA propia sobre el mismo hardware.

4.2.4 Rendimiento de Stencil 2D en precisión mixta

En Stencil 2D el comportamiento fue diferente al observado en GEMM y Convolución 2D: las rutas WMMA en FP16 y BF16 fueron más lentas que la ruta CUDA en FP32 ($0{,}216$ y $0{,}224$ ms frente a $0{,}157$ ms por paso, es decir, $0{,}73\times$ y $0{,}70\times$) y alcanzaron un ancho de banda efectivo menor (Tabla 12). Su menor intensidad aritmética hace que el movimiento de datos domine el tiempo de ejecución, por lo que la mayor capacidad de cómputo tensorial no produce aceleración.

El menor ancho de banda efectivo de las rutas WMMA frente a GPU FP32 es consistente con que el kernel WMMA carga, por cada punto de salida, cinco tiles completos desde memoria global (uno por vecino del operador de cinco puntos, sin compartir el halo entre ellos) y escribe el resultado en FP32 en lugar del formato de 16 bits de almacenamiento, ambos por encima de lo que supondría leer y escribir una sola vez a 16 bits. Este resultado caracteriza, por tanto, a la implementación WMMA evaluada y no al límite del hardware: una formulación que comparta el halo entre vecinos, como ConvStencil (Chen et al., 2024), reduce ese tráfico. Por esta razón, las rutas de Stencil se interpretan mediante el tiempo de ejecución y el ancho de banda efectivo, y su caracterización detallada se aborda en la Fase 3 junto con el análisis del *drift*. Las rutas WMMA de esta fase no llevan compensación; en la Fase 3 todas las rutas de 16 bits incorporan la compensación espacial, que añade cinco lecturas y una escritura en FP32 por celda, lo que es consistente con que allí su tasa relativa a GPU FP32 baje a $0{,}60$–$0{,}67\times$ (Sección «Rendimiento sostenido en función del tamaño»).

Tabla 12

Caracterización de rendimiento de las principales rutas de Stencil 2D ($N_X = N_Y = 4096$, $N_T = 20$).

Ruta	Precisión	Tiempo	Ancho de banda efectivo
GPU CUDA	FP32	0,156570 ms	857 GB/s
GPU CUDA	FP64	0,215757 ms	1 244 GB/s
WMMA	FP16	0,215603 ms	467 GB/s
WMMA	BF16	0,224102 ms	449 GB/s

4.2.5 Interpretación arquitectónica del rendimiento

Los resultados muestran que el beneficio de los Tensor Cores depende del patrón computacional del kernel. GEMM presenta alta reutilización de datos y elevada intensidad aritmética, condiciones favorables para las unidades tensoriales. Convolución 2D también obtiene una aceleración importante, aunque su comportamiento depende de la organización del problema y de la implementación seleccionada. Stencil 2D, en cambio, está condicionado por el movimiento de datos, por lo que incrementar la capacidad aritmética no elimina el cuello de botella de memoria.

La Tabla 8 es coherente con esa lectura: el kernel WMMA de GEMM alcanza un *throughput* de SM del 64,12 % frente a un 39,15 % de DRAM, de modo que no está limitado por el ancho de banda, aunque tampoco satura las unidades tensoriales (tubería activa solo el 22,50 % de los ciclos). La línea base de Stencil 2D, por el contrario, opera entre el 55 y el 80 % del ancho de banda pico de HBM2 (Tabla 7).

El modelo del techo de rendimiento (*roofline*) cuantifica esta distinción. Con un ancho de banda pico de HBM2 de $\sim$1 555 GB/s (el que corresponde a los porcentajes de la Tabla 7), el punto de inflexión de la A100 es de $\sim$12,5 FLOP/byte para FP32 escalar (19,5 TFLOP/s) y de $\sim$200 FLOP/byte para FP16 con Tensor Cores ($\sim$312 TFLOP/s, el pico implícito en el 53,2 % de la Tabla 10). GEMM con $N = 12288$ en FP16 tiene una intensidad aritmética algorítmica de

$N/4 = 3\,072$ FLOP/byte ($2N^3$ operaciones sobre $8N^2$ bytes, si cada operando se lee una vez y el resultado se escribe en FP32), muy por encima de ambos puntos de inflexión, lo que lo sitúa en la región limitada por cómputo. El kernel WMMA propio, en cambio, movió $59{,}4$ GB desde y hacia DRAM (Tabla 8), lo que da ~ 62 FLOP/byte efectivos: por su menor reutilización de datos queda bajo el punto de inflexión de FP16, coherente con que no alcance el techo tensorial. Stencil 2D opera a $\sim 0{,}62$ FLOP/byte en FP32 y $\sim 0{,}31$ FLOP/byte en FP64 ($535{,}252$ GFLOP/s sobre 857 GB/s y $388{,}420$ GFLOP/s sobre $1\,244$ GB/s), dos órdenes de magnitud por debajo del punto de inflexión de FP32: su rendimiento lo fija el ancho de banda.

4.2.6 Síntesis de la Fase 2

La activación efectiva de Tensor Cores se comprobó mediante perfilamiento en el kernel WMMA propio de GEMM, con el $100\,\%$ de los FLOP ejecutados por la ruta `fp16` $\rightarrow$ `fp32`, y en la ruta FP64 de cuBLAS (DMMA); en las rutas tensoriales de cuBLAS y cuDNN se infiere del modo de invocación. GEMM obtuvo las mayores ganancias: en las rutas FP16 de cuBLAS, $15{,}27\times$ respecto a la referencia FP64 ($N = 12288$) y $16{,}50\times$ sobre la ruta GPU clásica en FP32 ($N = 32768$). Convolución 2D alcanzó $28{,}07\times$ respecto a la referencia FP64 y $15{,}30\times$ respecto a la configuración FP32 con `IMPLICIT_GEMM` ($6{,}83\times$ frente al algoritmo FP32 por defecto de cuDNN). En Stencil 2D, en cambio, las rutas WMMA de 16 bits fueron más lentas que la ruta CUDA en FP32, por su mayor dependencia del subsistema de memoria. La brecha entre las rutas de biblioteca y las implementaciones WMMA propias en GEMM y Convolución 2D confirma que el beneficio de la precisión mixta no puede generalizarse a partir de un único kernel ni de la sola activación del hardware tensorial.

4.3 Fase 3: Comportamiento numérico y costo de la compensación

La tercera fase evaluó el costo numérico de la reducción de precisión bajo *encadenamiento real*: la salida de cada aplicación del operador se utilizó como entrada de la siguiente, de modo que el error medido fue el que se propaga a través de las iteraciones y no el de una operación aislada. Todas las métricas se calcularon contra la trayectoria de referencia en FP64 en los mismos puntos de observación, donde se registraron el error relativo L_2, el error relativo L_∞ y si la solución seguía

siendo finita.

4.3.1 Diseño de la campaña de Fase 3

Los tres operadores de la Tabla 13 tienen coeficientes exactamente representables en FP16 y BF16 (matriz de Hadamard escalada en GEMM, Laplaciano de cinco puntos en Convolución 2D y Stencil 2D), de modo que la reducción de precisión no introduce error en la representación del operador: el error medido procede solo de la cuantización del estado. Cada configuración se ejecutó una vez; la variabilidad con réplicas corresponde a la Fase 4.

Salvo en GEMM con $N = 1024$ y $N = 4096$, donde el factor de amplificación por iteración es exactamente 1, los operadores encadenados son *amplificantes*: el estado crece por un factor $\sqrt{2}$ por iteración en GEMM con $N = 2048$ y $N = 8192$, 2 en Convolución 2D y cerca de $1{,}99$ en Stencil 2D (Secciones «Estabilidad numérica y aparición de valores no finitos» y «Drift y horizonte de estabilidad en Stencil 2D»). Se trata de un caso de estrés: permite observar, dentro de horizontes accesibles, tanto la acumulación del error como la pérdida de finitud de cada formato, incluida la de FP64 en Stencil 2D. No representa a un operador estable como el de una difusión real (la ecuación del calor explícita $u \leftarrow u + r\,\mathcal{L}u$ con $r \leq 1/4$), que no hace crecer el estado. Las conclusiones sobre rango dinámico y horizonte de desbordamiento valen, por tanto, para operadores amplificantes; el comportamiento bajo un operador estable se examina en la Fase 4 (Sección «Stencil 2D»).

Tabla 13

Espacio experimental de la Fase 3. El encadenamiento aplica el mismo operador fijo sobre la salida de la iteración anterior. Los tamaños corresponden a N en GEMM, HW en Convolución 2D y $n_x = n_y$ en Stencil 2D.

Kernel	**Operador encadenado**	**Tamaños**	**Iteraciones**
GEMM	$X^{(n+1)} = X^{(n)}A$, $A = c\,H$, $c = 2^p$	1024, 2048, 4096, 8192	80
Convolución 2D	$X^{(n+1)} = \mathrm{conv}(X^{(n)}, W)$	64, 128, 256, 512	80
Stencil 2D	$u^{(n+1)} = \mathcal{L}\,u^{(n)}$	4096^2, 8192^2, 16384^2	120 (pase numérico); 1500 o 4000 según la malla (pase de energía)*

* Ambas cifras corresponden a la misma campaña con el operador de estrés: el pase numérico (120 iteraciones) alimenta las figuras de *drift* y error frente al tamaño; el pase de energía (4000 en 4096^2, 1500 en 8192^2 y 16384^2) se diseñó con margen suficiente para observar el desbordamiento de FP64 en la iteración 1044–1045 (Tabla 14).

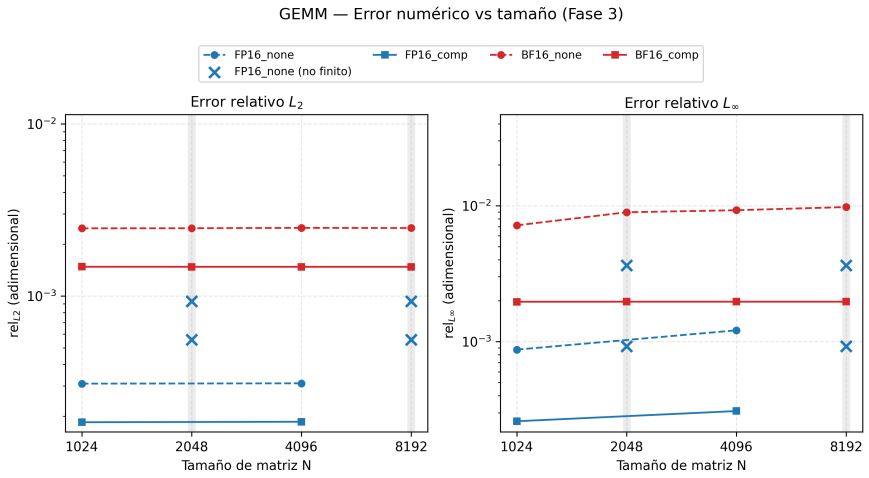
En GEMM y Convolución 2D la compensación se evaluó activada y desactivada, y el mecanismo es *local*: explota la linealidad del operador, $(T + r)\,A = TA + rA$, calculando el término principal sobre el estado cuantizado T y la corrección sobre el residuo de almacenamiento r, conservado en FP32 y reincorporado antes de volver a cuantizar. No es suma compensada de Kahan: lo que se conserva es el error de representación del estado, no el de una reducción. En Stencil 2D el mecanismo es la compensación espacial, que realimenta ese residuo sobre la vecindad del operador; como en esta fase todas las rutas de 16 bits la incorporan, el contraste entre compensar y no compensar solo se evalúa en GEMM y Convolución 2D. Para Stencil 2D ese contraste, junto con la variante Kahan local, se evalúa en la Fase 4 (Secciones «Estabilidad y variabilidad experimental» y «Kahan local frente a compensación espacial»).

4.3.2 Error numérico frente al tamaño del problema

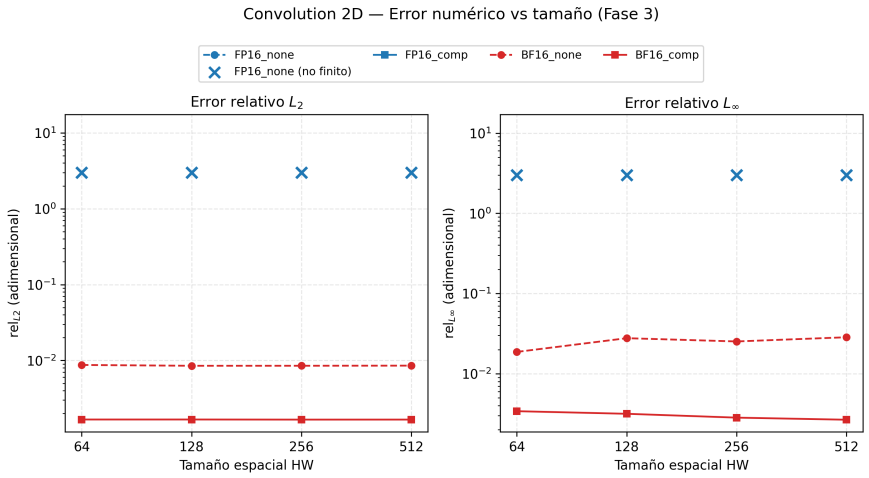
Esta sección determina si el error propagado depende del tamaño del problema o solo del formato. La Figura 1 presenta el error relativo de GEMM y Convolución 2D tras 80 aplicaciones del operador.

Figura 1

Error relativo L_2 y L_∞ respecto a la referencia FP64 tras 80 iteraciones encadenadas, con y sin compensación local. Las aspas y las bandas grises marcan las configuraciones cuya solución dejó de ser finita; en Convolución 2D las rutas FP16 se representan fuera de la escala válida.



(a) *GEMM, en función del tamaño de matriz N.*



(b) *Convolución 2D, en función del tamaño espacial HW.*

En las configuraciones que conservaron la finitud, el error L_2 de GEMM no mostró dependencia del tamaño de matriz: $3{,}09 \times 10^{-4}$ y $3{,}11 \times 10^{-4}$ en FP16 sin compensación para $N = 1024$ y $N = 4096$, los dos únicos con solución finita, y entre $2{,}47 \times 10^{-3}$ y $2{,}49 \times 10^{-3}$ en BF16 a lo largo de los cuatro. El cociente entre ambos formatos, $7{,}99$, coincide con la razón de sus unidades de redondeo (2^{-11} en FP16 y 2^{-8} en BF16): el error sin compensar lo fija la resolución del formato, no la longitud de la reducción, que crece con N.

La compensación local redujo ese error a $1{,}85 \times 10^{-4}$ en FP16 y $1{,}48 \times 10^{-3}$ en BF16, entre el $40{,}2$ y el $40{,}6$ % en todos los casos finitos. En norma L_∞ el efecto fue distinto: sin compensación el error creció con el tamaño (de $7{,}16 \times 10^{-3}$ a $9{,}76 \times 10^{-3}$ en BF16 y de $8{,}72 \times 10^{-4}$ a $1{,}21 \times 10^{-3}$ en FP16), mientras que con compensación BF16 se mantuvo constante en $1{,}96 \times 10^{-3}$. El error máximo puntual bajó entre $3{,}4$ y $5{,}0$ veces y, en BF16, dejó de depender del tamaño.

En Convolución 2D el resultado dominante no fue la magnitud del error, sino la pérdida de estabilidad: las dos rutas FP16, con y sin compensación, dejaron de producir valores finitos desde la iteración 20 en los cuatro tamaños, por lo que su error no resulta comparable en la misma escala. Las rutas BF16 tampoco dependieron del tamaño espacial: el error L_2 se mantuvo entre $8{,}50 \times 10^{-3}$ y $8{,}72 \times 10^{-3}$ sin compensación y en $1{,}66 \times 10^{-3}$ con ella, una reducción del 80 % (factor $5{,}1$–$5{,}2$). En norma L_∞ la ruta sin compensar osciló entre $1{,}87 \times 10^{-2}$ y $2{,}85 \times 10^{-2}$ sin tendencia definida y la compensada decreció monótonamente de $3{,}41 \times 10^{-3}$ a $2{,}67 \times 10^{-3}$: la separación creció de $5{,}5$ en $HW = 64$ a $10{,}7$ en $HW = 512$.

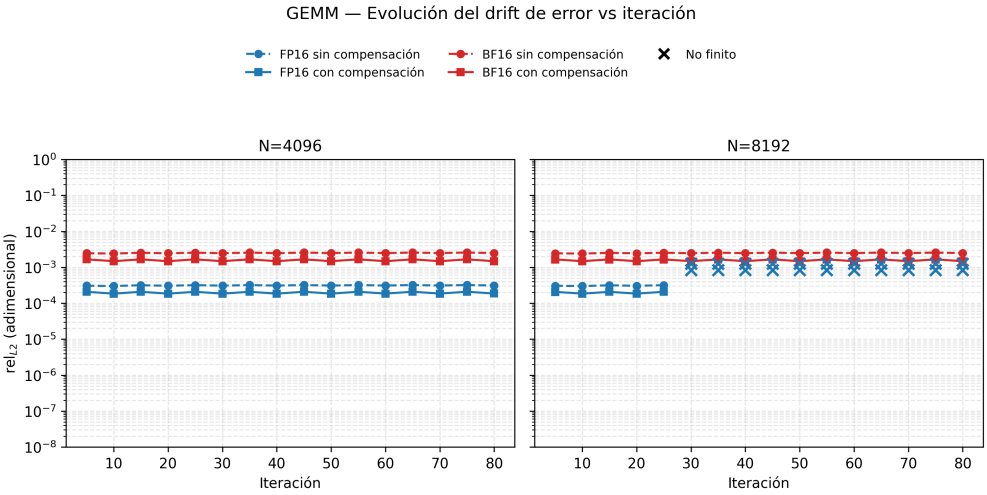
4.3.3 Evolución del error a lo largo del encadenamiento

Que el error no dependa del tamaño no implica que no dependa de la iteración. Las Figuras 2 y 3 muestran el error en función del número de aplicaciones del operador.

En GEMM el error relativo no se acumuló con la iteración. En $N = 4096$ se mantuvo entre $3{,}01 \times 10^{-4}$ y $3{,}21 \times 10^{-4}$ en FP16 sin compensación y entre $2{,}41 \times 10^{-3}$ y $2{,}57 \times 10^{-3}$ en BF16 sin compensación durante las 80 iteraciones, alternando entre dos ramas según la paridad del punto de observación: $A = c\,H$ con H de Hadamard simétrica cumple $H^2 = N\,I$, de modo que el estado se reproduce salvo un factor de escala cada dos aplicaciones. Cada rama converge a un

Figura 2

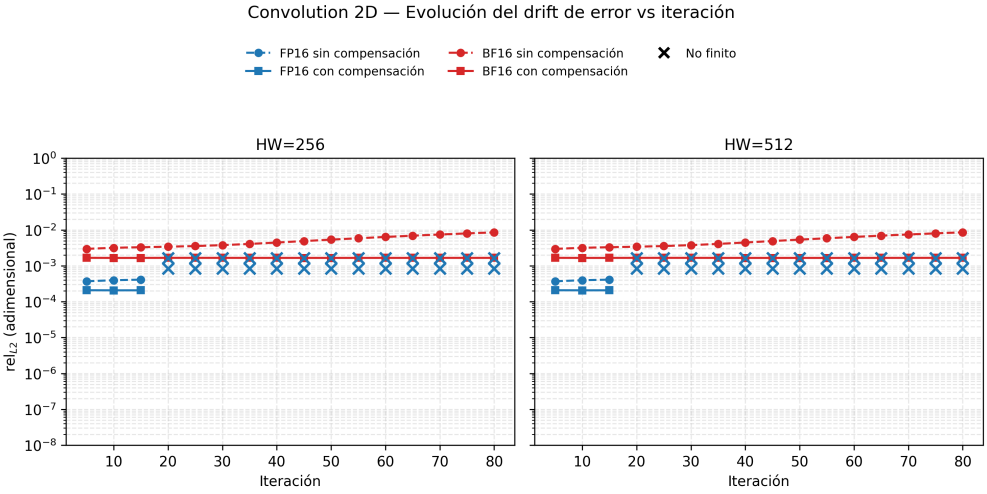
GEMM encadenado: error relativo L_2 por iteración para $N = 4096$ y $N = 8192$, con y sin compensación local. Las aspas marcan los puntos de observación en que la solución dejó de ser finita, a partir de la iteración 30 en $N = 8192$.



Línea discontinua/círculo: sin compensación; línea continua/cuadrado: con compensación local.

Figura 3

Convolución 2D encadenada: error relativo L_2 por iteración para $HW = 256$ y $HW = 512$, con y sin compensación local. Las rutas FP16 dejaron de ser finitas desde la iteración 20 en los dos tamaños.



Línea discontinua/círculo: sin compensación; línea continua/cuadrado: con compensación local.

valor asintótico, $3{,}21 \times 10^{-4}$ y $3{,}11 \times 10^{-4}$ en FP16, con incrementos decrecientes. Al ser $H/\sqrt{N}$ ortogonal, el operador amplifica por igual la solución y su error, y la trayectoria de referencia y la de baja precisión lo hacen de forma idéntica: el error *relativo* no crece. El caso $N = 8192$ aporta el límite de estabilidad: las dos rutas FP16 registraron su último estado finito en la iteración 25 y el primero no finito en la iteración 30, el mismo umbral que en $N = 2048$, mientras que ambas rutas BF16 completaron las 80 iteraciones.

En Convolución 2D se observó el comportamiento opuesto. Su filtro es el Laplaciano discreto de cinco puntos, cuyo factor de amplificación depende de la frecuencia espacial y alcanza un radio espectral de 2 en Nyquist: las componentes de error próximas a esa frecuencia crecen más rápido que la solución, y el error relativo se acumula. En $HW = 512$, la ruta BF16 sin compensación creció monótonamente de $2{,}96 \times 10^{-3}$ en la iteración 5 a $8{,}54 \times 10^{-3}$ en la iteración 80, un factor de $2{,}9$, con una trayectoria que difiere en menos del $0{,}5\,\%$ de la de $HW = 256$.

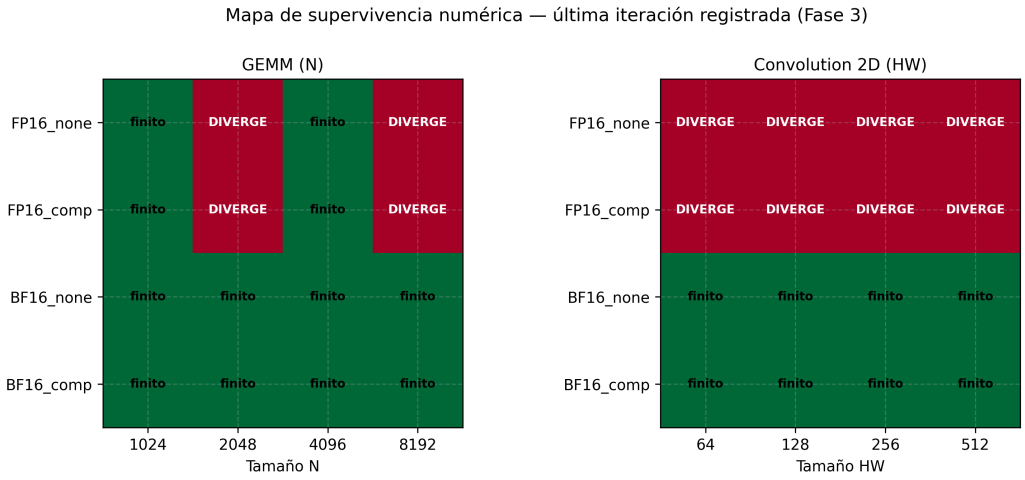
La compensación local suprimió esa acumulación en los dos formatos: la ruta BF16 compensada se mantuvo entre $1{,}655 \times 10^{-3}$ y $1{,}664 \times 10^{-3}$ durante las 80 iteraciones, una variación inferior al $0{,}6\,\%$, y la ruta FP16 compensada en $2{,}07 \times 10^{-4}$ durante el intervalo en que permaneció finita, frente a la ruta FP16 sin compensar, que creció de $3{,}69 \times 10^{-4}$ a $4{,}11 \times 10^{-4}$ entre las iteraciones 5 y 15 antes de perder la finitud.

4.3.4 Estabilidad numérica y aparición de valores no finitos

Se distinguen dos fenómenos: el crecimiento del error, compatible con una solución finita, y la pérdida de finitud, que invalida la solución. La Figura 4 resume el estado de la solución en la última iteración registrada para cada configuración de GEMM y Convolución 2D.

Figura 4

Estado de la solución en la última iteración registrada, por formato, esquema de compensación y tamaño del problema en GEMM y Convolución 2D.



La finitud de FP16 no dependió del tamaño del problema, sino de cuánto amplifica el operador en cada iteración. En GEMM ese factor es $c\sqrt{N}$ con c potencia de dos, de modo que vale exactamente 1 cuando $\sqrt{N}$ también lo es ($N = 1024$ y $N = 4096$) y $\sqrt{2}$ cuando no ($N = 2048$ y $N = 8192$). Con factor 1 el estado no crece y ninguna ruta desborda; con $\sqrt{2}$ bastan unas 32 iteraciones para agotar el rango de FP16, y la pérdida de finitud se registró entre los puntos de observación 25 y 30. En Convolución 2D el operador amplifica por 2, el margen se reduce a unas 16 iteraciones y FP16 perdió la finitud en los cuatro tamaños.

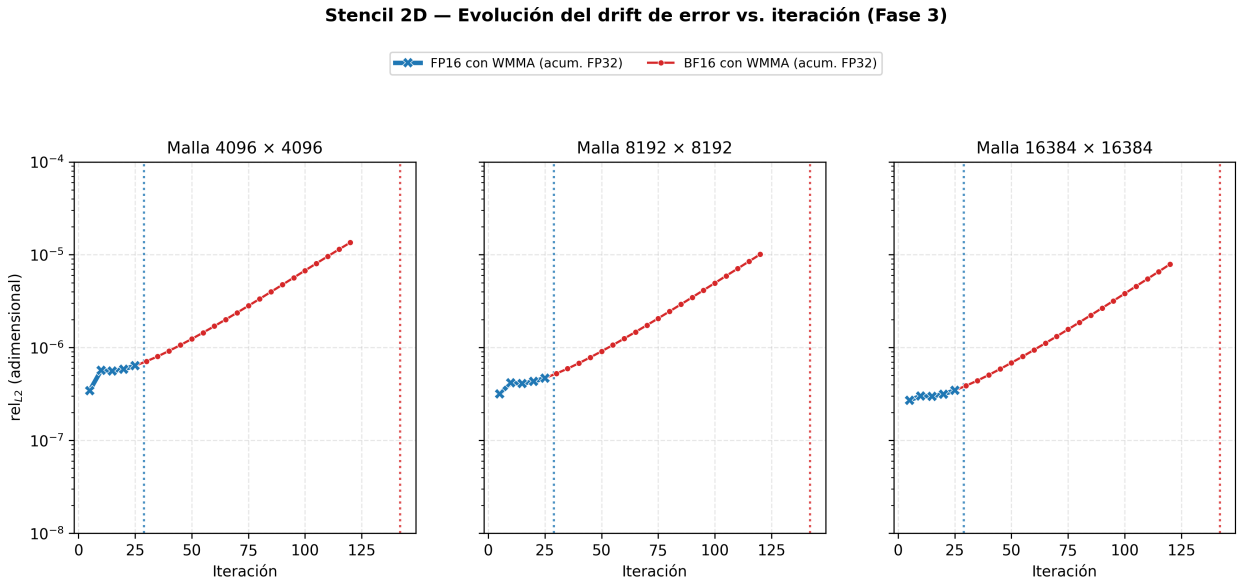
BF16 completó las 80 iteraciones en las dieciséis configuraciones evaluadas (dos kernels, cuatro tamaños, con y sin compensación): comparte con FP32 un exponente de ocho bits, que sitúa su desbordamiento más allá de la iteración 128 incluso con el operador más amplificante. La compensación no alteró la finitud de ninguna configuración, pues actúa sobre la resolución del estado y no sobre su rango. Bajo operadores amplificantes, la diferencia práctica entre FP16 y BF16 es, por tanto, de rango dinámico y no de resolución.

4.3.5 Drift y horizonte de estabilidad en Stencil 2D

Stencil 2D admite un número de iteraciones mucho mayor que los otros dos kernels y permite resolver la evolución del *drift* sobre una trayectoria larga. La Figura 5 presenta el error relativo L_2 por iteración para las tres mallas evaluadas en las rutas WMMA con acumulación en FP32 y compensación espacial.

Figura 5

Stencil 2D: evolución del error relativo L_2 respecto a la referencia FP64 en función de la iteración, para las tres mallas evaluadas. Las líneas verticales indican el horizonte de desbordamiento medido de cada formato.



El drift creció de forma sostenida con la iteración, a diferencia de lo observado en GEMM. En la iteración 120 la ruta BF16 alcanzó $1{,}35 \times 10^{-5}$ en la malla de 4096^2, $1{,}01 \times 10^{-5}$ en 8192^2 y $7{,}89 \times 10^{-6}$ en 16384^2: aumentar la malla no agravó el error, sino que lo redujo en un factor de $1{,}71$ entre los dos extremos.

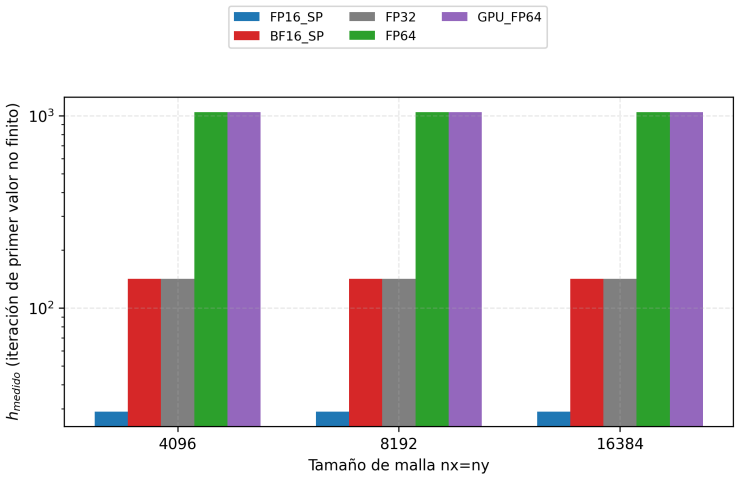
FP16 dejó de producir valores finitos en la iteración 29. Hasta ese punto su trayectoria y la de BF16 son indistinguibles: en la iteración 25 difieren en menos del $0{,}1\,\%$ en las tres mallas ($6{,}363$ frente a $6{,}366 \times 10^{-7}$ en 4096^2). En ambas rutas el estado se propaga a través de un residuo en FP32 y la acumulación se realiza en FP32, de modo que la precisión propagada la fija el formato

de acumulación y no el de los operandos de entrada. La diferencia entre FP16 y BF16 no está en el error, sino en el intervalo durante el cual ese error puede medirse.

Esa diferencia se cuantifica en el horizonte de desbordamiento, definido como la iteración del primer valor no finito (Figura 6).

Figura 6

Stencil 2D: horizonte de desbordamiento por formato y tamaño de malla, en escala logarítmica.



El horizonte fue el mismo en las tres mallas para cada formato, salvo una iteración en FP64 (Tabla 14): lo fijan el rango de exponente del formato y la tasa de amplificación del operador, no el número de celdas. Un modelo de crecimiento geométrico, con el estado multiplicándose por $\lambda \approx 1{,}99$ hasta agotar el máximo representable, lo predice con un error inferior al 7 % en los cuatro formatos y del $0{,}4$ % en FP64 (313 puntos de ajuste, $R^2 = 1{,}000$).

Tabla 14

Horizonte de desbordamiento en Stencil 2D: iteración del primer valor no finito frente a la predicha por el modelo ($\lambda \approx 1{,}99$; valores para 4096^2, con variación inferior al $0{,}5\,\%$ entre mallas). Las rutas de 16 bits incorporan compensación espacial.

Ruta	4096^2	8192^2	16384^2	**Predicho**
FP16 + compensación espacial	29	29	29	27,0
BF16 + compensación espacial	142	142	142	136,0
FP32	142	142	142	144,6
FP64 (CPU)	1045	1044	1044	1040,6
GPU FP64	1045	1044	1044	1040,6

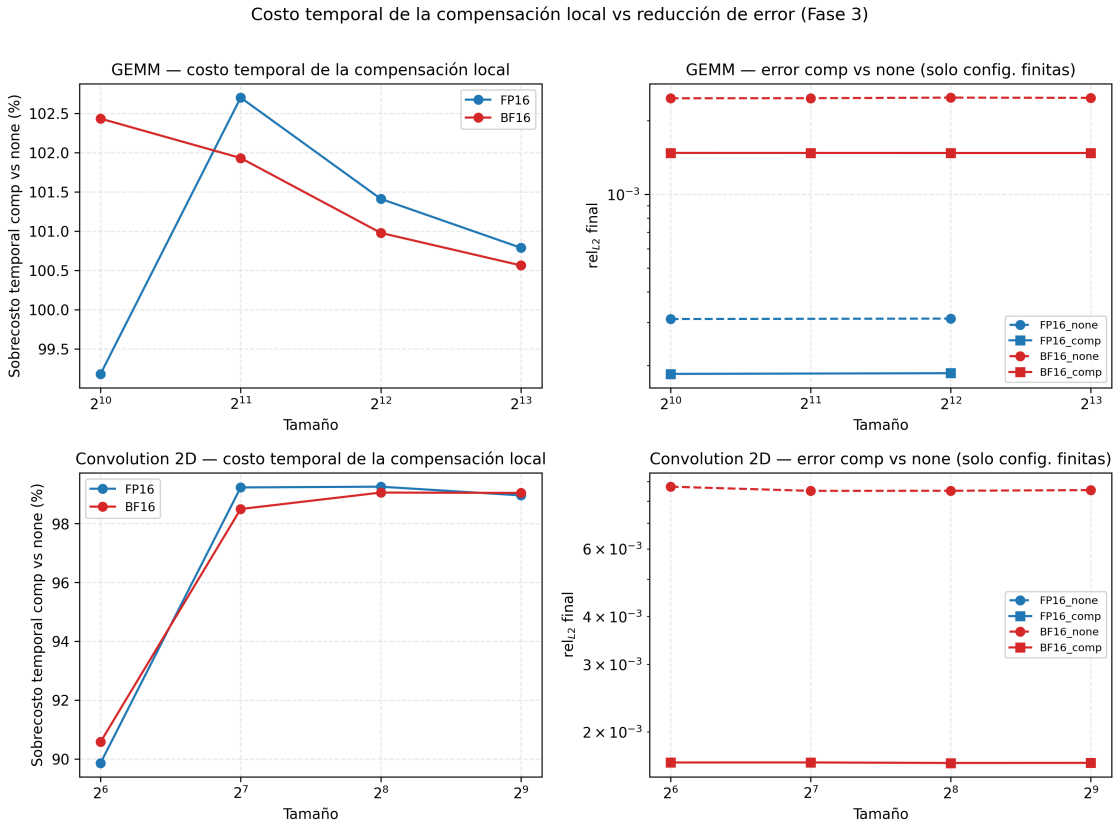
La progresión $29 \rightarrow 142 \rightarrow 1044$ reproduce esa jerarquía: FP16 desborda al superar 2^{16}; BF16 y FP32 comparten un campo de exponente de 8 bits y, con él, el mismo horizonte; FP64 lo alcanza cerca de 2^{1024}. Que BF16 iguale a FP32 pese a tener la mantisa más corta de los cuatro formatos indica que, bajo operadores amplificantes, el límite de los formatos de 16 bits lo fija el rango dinámico. En todos los formatos el desbordamiento ocurre; el formato solo determina en qué iteración.

4.3.6 Efecto y costo de la compensación local

Esta sección cuantifica el costo temporal de la compensación local y la reducción de error que produce. La Figura 7 contrasta ambas magnitudes para las configuraciones que permanecieron finitas.

Figura 7

Compensación local en GEMM y Convolución 2D: sobrecosto temporal respecto a la ruta sin compensar (izquierda) y error relativo L_2 final de las configuraciones finitas (derecha), en función del tamaño del problema.



El sobrecosto temporal estuvo entre el 99,2 y el 102,7 % en GEMM y entre el 89,9 y el 99,3 % en Convolución 2D, sin dependencia del tamaño ni del formato: la compensación duplicó el tiempo de ejecución, como corresponde a una implementación que ejecuta un segundo producto tensorial por iteración.

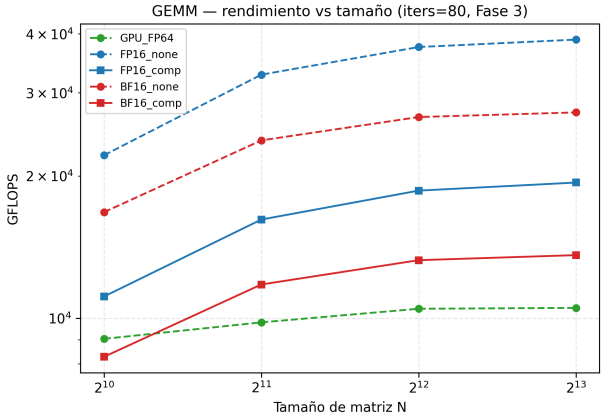
La ganancia numérica depende del operador: un factor de 1,68 en GEMM frente a 5,1–5,2 en Convolución 2D, ambos independientes del tamaño. En GEMM el error no se acumula y la compensación solo recorta el nivel estacionario; en Convolución 2D impide además su crecimiento por iteración. Esa mejora no alcanza a la finitud: ninguna configuración compensada la recuperó (Sección «Estabilidad numérica y aparición de valores no finitos»).

4.3.7 Rendimiento sostenido en función del tamaño

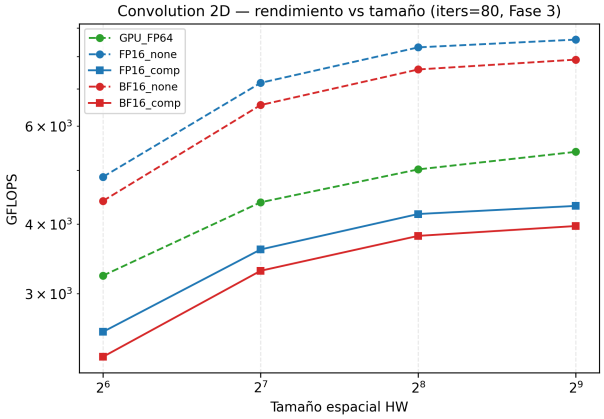
Esta sección reporta el rendimiento asociado a cada uno de los comportamientos numéricos anteriores (Figura 8). En GEMM y Convolución 2D la tasa se calcula sobre las operaciones *útiles* (una aplicación del operador por iteración), también en las rutas compensadas: el trabajo adicional de la corrección no cuenta como operación útil y aparece por tanto como una caída del rendimiento.

Figura 8

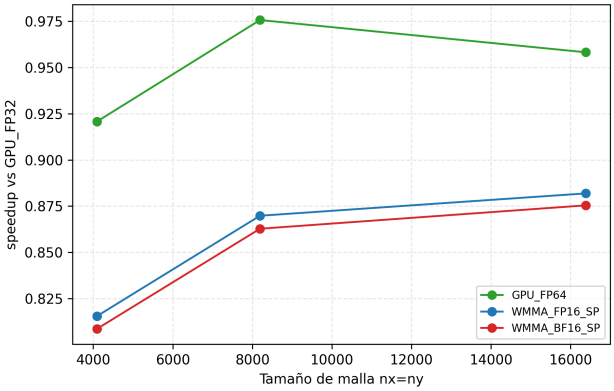
Rendimiento sostenido frente al tamaño del problema. GEMM y Convolución 2D: tasa en GFLOP/s de las rutas con y sin compensación local, junto a la referencia GPU FP64. Stencil 2D: speedup *frente a GPU FP32 de las rutas WMMA con compensación espacial y de GPU FP64.*



(a) *GEMM*



(b) *Convolución 2D*



(c) *Stencil 2D*

En GEMM el rendimiento aumentó con N hasta saturar: FP16 sin compensación pasó de $22{,}1$ a $38{,}9$ TFLOP/s y BF16 de $16{,}8$ a $27{,}3$ TFLOP/s, frente a $9{,}04$–$10{,}5$ TFLOP/s de GPU FP64. La mayor tasa correspondió a una configuración no finita; entre las estables, BF16 sin compensar alcanzó $2{,}59\times$ la referencia en $N = 8192$. Las variantes compensadas llegaron a $19{,}4$ y $13{,}6$ TFLOP/s en FP16 y BF16, esto es, $0{,}494$–$0{,}502$ de la tasa no compensada en ambos kernels, consistente con el sobrecosto cercano al $100\,\%$ (Sección «Efecto y costo de la compensación local»).

En Convolución 2D, GPU FP64 alcanzó $3{,}23$–$5{,}39$ TFLOP/s y las rutas compensadas $2{,}31$–$4{,}31$ TFLOP/s, por lo que fueron más lentas que FP64 en todos los tamaños: en $HW = 512$, $4{,}31$ frente a $5{,}39$ TFLOP/s ($20\,\%$ menos). Solo las variantes sin compensación superaron la referencia, con $8{,}58$ y $7{,}90$ TFLOP/s en FP16 y BF16.

Stencil 2D tampoco obtuvo aceleración: las rutas Tensor Core con compensación espacial alcanzaron $0{,}60$–$0{,}67\times$ GPU FP32, y GPU FP64, $0{,}79$–$0{,}81\times$. En términos absolutos, FP64 sostuvo 365–374 GFLOP/s frente a 270–311 GFLOP/s de las rutas de 16 bits. La aceleración observada en GEMM y Convolución 2D no se trasladó, por tanto, a Stencil 2D. Determinar si esta limitación proviene de ancho de banda, latencia u ocupación requeriría métricas de Nsight Compute no disponibles en esta campaña.

4.3.8 Síntesis de la Fase 3

De los resultados de esta fase se desprenden tres conclusiones. Primera: la diferencia práctica entre FP16 y BF16 bajo operadores amplificantes es de rango dinámico y no de resolución. FP16 perdió la finitud en dos de los cuatro tamaños de GEMM, en los cuatro de Convolución 2D y, en Stencil 2D, tras 29 iteraciones. BF16 completó el intervalo evaluado en los dos primeros kernels y, en Stencil 2D, igualó el horizonte de FP32 (142 iteraciones) pese a su mantisa más corta; un modelo de agotamiento del exponente reproduce esos horizontes con un error inferior al $7\,\%$. Mientras ambos permanecieron finitos, su error difirió en menos del $0{,}1\,\%$, porque el estado se propaga a través de un residuo en FP32.

Segunda: la acumulación del error depende del operador y no del formato. En GEMM

permaneció estacionaria durante las 80 iteraciones e invariante frente a N; en Convolución 2D creció un factor de $2{,}9$ sobre el mismo intervalo; en Stencil 2D creció de forma sostenida y decreció al aumentar la malla.

Tercera: el costo de la compensación local debe ponderarse frente a su ganancia. Redujo el error por factores de $1{,}68$ en GEMM y $5{,}1$–$5{,}2$ en Convolución 2D, y suprimió su crecimiento por iteración en el segundo caso, pero duplicó el tiempo de ejecución y no restauró la finitud en ninguna configuración que la hubiese perdido; en Convolución 2D ese costo situó a las rutas de 16 bits por debajo de la referencia en doble precisión.

Estos resultados motivan la Fase 4, donde el costo de la compensación y del anclaje se estudia simultáneamente en tiempo, energía y error; en Stencil 2D el análisis multiobjetivo se construye además con un operador difusivo, numéricamente estable.

4.4 Fase 4: Anclaje de referencia, sensibilidad y análisis multiobjetivo

La Fase 3 mostró que la compensación mejora la resolución, pero no el rango: redujo el error por factores de $1{,}68$ en GEMM y $5{,}1$–$5{,}2$ en Convolución 2D sin desplazar el horizonte de desbordamiento. La Fase 4 evalúa una estrategia distinta: reemplazar periódicamente, cada K iteraciones, el paso de baja precisión por uno completo en FP64. Como se define en la Sección «Anclaje periódico en FP64», este paso parte del estado reconstruido de la propia ruta reducida, no de la trayectoria de referencia.

La fase estudia el costo temporal del anclaje, su efecto sobre el error al variar K y la posición de las configuraciones en el espacio tiempo–energía–error tras aplicar criterios explícitos de admisión.

4.4.1 Configuración experimental y definición del anclaje

La periodicidad K indica cada cuántas iteraciones se sustituye el paso de baja precisión por FP64; $K = 0$ representa ausencia de anclaje y sirve como referencia. Con $K = 1$, todas las iteraciones se ejecutan en FP64, por lo que ese punto no compara directamente formatos de 16 bits.

En GEMM y Convolución 2D se evaluaron rutas de 16 bits con y sin compensación local. En Stencil 2D se usaron las rutas WMMA con compensación espacial en FP16 y BF16; la campaña

de variabilidad incluyó además las rutas sin compensación y con Kahan local (Sección «Estabilidad y variabilidad experimental»). Para el análisis multiobjetivo se añadió una campaña de Stencil con operador difusivo ($\alpha = 3/16$, condición inicial monomodo), incluyendo variantes sin compensación y con Kahan local (Sección «Stencil 2D»).

El anclaje requiere compensación activa para reconstruir el estado de entrada al paso FP64; por ello, compensación y periodicidad no forman un diseño factorial completo. No existen configuraciones con $K > 0$ sin compensación y cada configuración anclada se compara con la ruta compensada del mismo formato en $K = 0$.

La campaña separa las pasadas numéricas, destinadas al seguimiento del error, de las energéticas, cuyo mayor horizonte garantiza una medición fiable mediante NVML. Por ello, en el análisis multiobjetivo de la Sección «Análisis multiobjetivo tiempo–energía–error», el error se evalúa en un horizonte común, mientras que tiempo y energía se expresan como tasas por iteración obtenidas en ventanas fiables. Las $n = 8$ réplicas corresponden únicamente a la campaña de variabilidad ($N = 1024$; $HW = 64$; 1024^2): las campañas de producción (operador de estrés y operador difusivo) tienen una sola medición por configuración, por lo que sus frentes de Pareto se basan en dominancia determinista y no admiten inferencia estadística.

Las periodicidades se compararon mediante una prueba t pareada de una muestra sobre $\ln[T(K)/T(0)]$ ($n = 8$, $\mathrm{gl} = 7$), calculada réplica por réplica: la transformación logarítmica simetriza la razón y el diseño pareado compara cada réplica consigo misma. Para GEMM y Convolución 2D ($K \in \{1, 5\}$), los valores p se corrigieron mediante Holm–Bonferroni sobre una familia de ocho comparaciones; Stencil 2D se corrigió por separado, sobre su propia familia de cuatro (Tabla 19).

4.4.2 Estabilidad y variabilidad experimental

La variabilidad entre réplicas fue baja: en las dieciocho configuraciones, el coeficiente de variación del tiempo por iteración osciló entre $0{,}13$ y $0{,}79\,\%$. Los valores mayores correspondieron a Stencil 2D, cuyo menor tiempo por iteración lo hace más sensible a las variaciones dinámicas del reloj.

Tabla 15

Configuración experimental de la Fase 4.

Factor	**GEMM / Convolución 2D**	**Stencil 2D**
Formato	FP16, BF16	FP16, BF16
Periodicidad K	Producción: $\{0, 1, 5, 20\}$; variabilidad: $\{0, 1, 5\}$	Variabilidad: $\{0, 1, 5\}$; estrés: $\{0, 1, 8, 32\}$; difusivo: $\{0, 1, 2, 32, 64, 128\}$
Compensación	Local	Espacial (estrés); ninguna, Kahan local y espacial (variabilidad y difusivo)
Operador	—	Estrés; difusivo $\alpha = 3/16$
Tamaño de variabilidad	$N = 1024$; $HW = 64$	1024^2
Tamaños de sensibilidad	$N = 2^{10}$–2^{13}; $HW = 2^6$–2^9	4096^2–16384^2
Multiobjetivo	—	4096^2, 8192^2 (difusivo)
Horizontes numéricos	20, 40, 80	4 000 y 1 500 (difusivo)
Horizontes energéticos	GEMM: 24 000 y 500; Conv.: 37 000 y 2 500	4 000–16 000 y 1 500–3 000 (difusivo)
Réplicas / unidad	$n = 8$ por configuración en variabilidad; réplica independiente, usando la mediana intra-réplica por configuración.	
Inferencia	Prueba t pareada sobre $\ln[T(K)/T(0)]$ ($n = 8$, gl$= 7$), con corrección de Holm–Bonferroni.	
Métricas	t_{iter}, energía de GPU y error relativo L_2.	

Nota. $K = 0$ indica ausencia de anclaje. Las configuraciones con $K > 0$ emplean compensación activa (Sección «Anclaje periódico en FP64»). En GEMM y Convolución 2D, $K = 20$ pertenece únicamente a producción; la campaña de variabilidad ($n = 8$) usa $K \in \{0, 1, 5\}$.

Tabla 16

Variabilidad entre réplicas del tiempo por iteración ($n = 8$, horizonte de 40 iteraciones). En Stencil 2D, las rutas WMMA usan compensación espacial en las tres periodicidades.

Kernel	**Formato**	K	**Media (ms)**	**CV (%)**	**IC 95 % (ms)**
GEMM	BF16	0	0,25509	0,13	[0,25480; 0,25537]
		1	0,27862	0,17	[0,27823; 0,27900]
		5	0,26989	0,20	[0,26944; 0,27034]
	FP16	0	0,19321	0,15	[0,19296; 0,19345]
		1	0,27876	0,19	[0,27832; 0,27920]
		5	0,22065	0,23	[0,22023; 0,22107]
Convolución 2D	BF16	0	0,13083	0,32	[0,13047; 0,13118]
		1	0,11516	0,35	[0,11481; 0,11550]
		5	0,13377	0,39	[0,13332; 0,13421]
	FP16	0	0,11797	0,39	[0,11759; 0,11835]
		1	0,11536	0,38	[0,11499; 0,11572]
		5	0,12378	0,49	[0,12328; 0,12428]
Stencil 2D	BF16	0	0,03608	0,44	[0,03594; 0,03621]
		1	0,07554	0,79	[0,07504; 0,07604]
		5	0,05756	0,48	[0,05733; 0,05779]
	FP16	0	0,03562	0,35	[0,03552; 0,03572]
		1	0,07630	0,58	[0,07593; 0,07667]
		5	0,05789	0,27	[0,05776; 0,05802]

Esa estabilidad es relevante porque no fue posible fijar la frecuencia de reloj de la GPU (Sección «Limitaciones»); las ocho réplicas se adoptaron para acotar la variabilidad inducida por el escalado dinámico. Los intervalos resultantes son estrechos, pero con $n = 8$ solo sostienen las comparaciones cuyos intervalos no se solapan.

Sobre esta misma campaña de variabilidad se compararon los tres esquemas de compensación de Stencil 2D (ninguna, Kahan local y espacial, Sección «Esquemas de compensación numérica evaluados»). La Tabla 17 reporta, para cada esquema, el tiempo por iteración y el error relativo L_2 tras 20 iteraciones, el horizonte en que las rutas FP16 aún son finitas. El error es determinista a $K = 0$: coincide en las ocho réplicas, de modo que las diferencias entre esquemas son exactas y no requieren prueba estadística. Las diferencias de tiempo, de hasta un $42\,\%$, son muy superiores a la variabilidad entre réplicas (coeficientes de variación inferiores al $1\,\%$, Tabla 16). Las ocho réplicas de Tabla 17 proceden de una tanda de trabajos distinta de las ocho réplicas oficiales ($r1$–$r8$) de la Tabla 16, aunque comparten configuración nominal (malla 1024^2, $K = 0$, compensación espacial); repetir la comparación sobre las réplicas $r1$–$r8$ a 20 iteraciones da $0{,}0357$ ms en FP16, coherente con el horizonte de 40 iteraciones de esa tabla ($0{,}0356$ ms), lo que descarta el horizonte como causa de la diferencia frente a los $0{,}0385$ ms de Tabla 17 y la atribuye a la variación entre tandas de ejecución independientes, ya señalada como limitación por el DVFS no fijable en PACCA (Sección «Limitaciones»).

Tabla 17

Esquemas de compensación de Stencil 2D bajo el operador de estrés (campaña de variabilidad, malla 1024^2*,* $K = 0$*,* 20 *iteraciones): mediana del tiempo por iteración sobre las ocho réplicas y error relativo* L_2*.*

Formato	**Esquema**	**Tiempo por iteración (ms)**	**Error L_2 rel.**
FP16	Sin compensación	0,0274	$7{,}63 \times 10^{-2}$
FP16	Kahan local	0,0276	$7{,}63 \times 10^{-2}$
FP16	Compensación espacial	0,0385	$1{,}28 \times 10^{-6}$
BF16	Sin compensación	0,0271	$5{,}71 \times 10^{-1}$
BF16	Kahan local	0,0276	$5{,}71 \times 10^{-1}$
BF16	Compensación espacial	0,0386	$1{,}28 \times 10^{-6}$

En ambos formatos, la compensación espacial es la única que reduce el error, unas 6×10^4 veces en FP16 y 4×10^5 en BF16, a cambio de unos $0{,}011$ ms más por iteración (un 41–42 %). Kahan local y la ruta sin compensación producen el mismo error y difieren en tiempo en menos de un 2 %, consistente con que ninguna de las dos corrige el error bajo el operador de estrés (Sección «Drift y horizonte de estabilidad en Stencil 2D»).

4.4.3 Efecto de la frecuencia de anclaje sobre el tiempo

La Tabla 18 expresa el costo del anclaje como la razón $T(K)/T(0)$, de modo que un valor superior a la unidad indica sobrecosto.

Tabla 18

Sobrecosto temporal del anclaje en FP64, expresado como $T(K)/T(0)$, con intervalos de confianza al $95\,\%$ sobre $n = 8$ réplicas.

Kernel	**Formato**	$K = 1$	$K = 5$
GEMM	FP16	1,443 [1,440; 1,446]	1,142 [1,140; 1,144]
GEMM	BF16	1,092 [1,091; 1,094]	1,058 [1,056; 1,060]
Convolución 2D	FP16	0,978 [0,973; 0,982]	1,049 [1,044; 1,054]
Convolución 2D	BF16	0,880 [0,877; 0,884]	1,022 [1,018; 1,027]
Stencil 2D	FP16	2,142 [2,130; 2,154]	1,625 [1,619; 1,631]
Stencil 2D	BF16	2,094 [2,080; 2,107]	1,595 [1,586; 1,605]

Las doce razones difieren significativamente de 1 tras la corrección de Holm–Bonferroni (Tabla 19). En Convolución 2D, dos de ellas son inferiores a la unidad: anclar cada iteración resultó más rápido que no anclar ($0{,}978$ en FP16 y $0{,}880$ en BF16), efecto que las trayectorias por réplica permiten interpretar.

Tabla 19

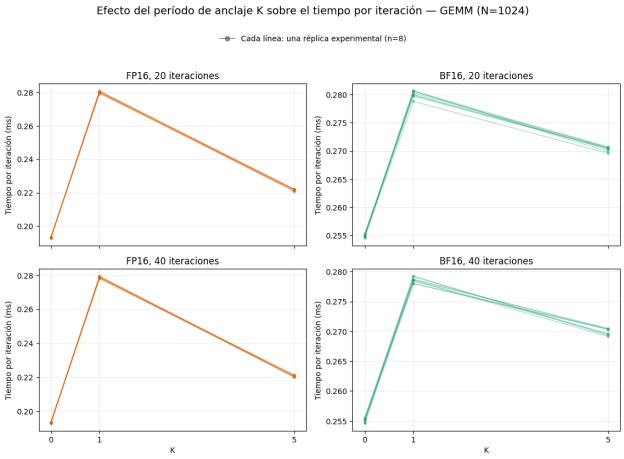
Prueba t pareada sobre $\ln[T(K)/T(K=0)]$ ($n = 8$, gl= 7): valor p sin corregir y con Holm–Bonferroni. GEMM y Convolución 2D forman una única familia de ocho comparaciones; Stencil 2D se corrigió por separado sobre su propia familia de cuatro. Con $n = 8$ y normalidad no verificada, los valores inferiores a 10^{-10} se reportan como $< 10^{-10}$.

Kernel	**Formato**	K	p **sin corregir**	p **Holm**
GEMM	FP16	1	$< 10^{-10}$	$< 10^{-10}$
	FP16	5	$< 10^{-10}$	$< 10^{-10}$
	BF16	1	$< 10^{-10}$	$< 10^{-10}$
	BF16	5	$< 10^{-10}$	$< 10^{-10}$
Convolución 2D	FP16	1	$8{,}7 \times 10^{-6}$	$1{,}1 \times 10^{-5}$
	FP16	5	$5{,}3 \times 10^{-8}$	$1{,}6 \times 10^{-7}$
	BF16	1	$< 10^{-10}$	$< 10^{-10}$
	BF16	5	$5{,}7 \times 10^{-6}$	$1{,}1 \times 10^{-5}$
Stencil 2D	FP16	1	$< 10^{-10}$	$< 10^{-10}$
	FP16	5	$< 10^{-10}$	$< 10^{-10}$
	BF16	1	$< 10^{-10}$	$< 10^{-10}$
	BF16	5	$< 10^{-10}$	$< 10^{-10}$

El costo del anclaje difiere entre kernels y sigue el orden de intensidad aritmética establecido en las fases anteriores. Las tres subsecciones siguientes examinan cada caso sobre las trayectorias por réplica, que revelan estructuras que la razón agregada no refleja.

Figura 9

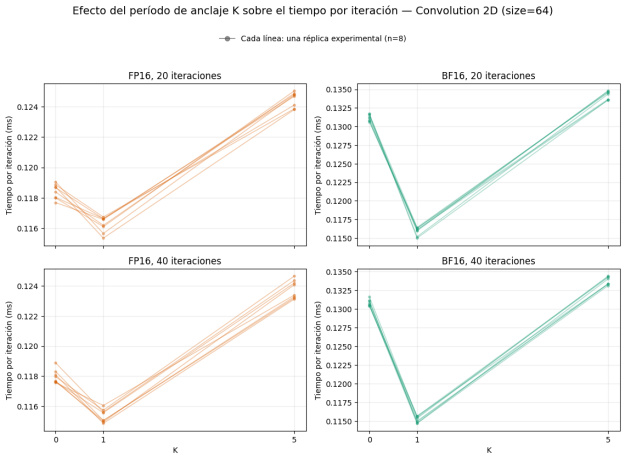
GEMM ($N = 1024$): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una de las ocho réplicas experimentales.



4.4.3.1 GEMM. Las ocho réplicas se superponen, en concordancia con coeficientes de variación inferiores al $0{,}25\,\%$. El tiempo alcanza su máximo en $K = 1$ y desciende en $K = 5$. Además, las dos rutas parten de valores distintos sin ancla —$0{,}193$ ms en FP16 frente a $0{,}255$ ms en BF16— y convergen en $K = 1$ en torno a $0{,}279$ ms. Cuando $K = 1$ y cada iteración se ejecuta mediante un único paso completo en FP64, el tiempo medido deja de distinguir los formatos de 16 bits.

Figura 10

Convolución 2D ($HW = 64$): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una réplica.

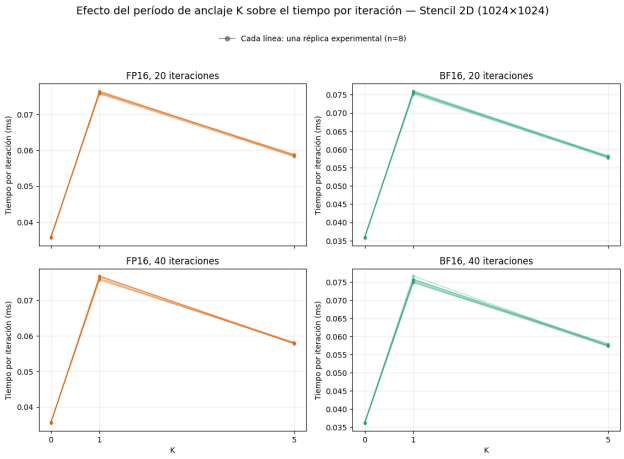


4.4.3.2 Convolución 2D. En Convolución 2D, $K = 1$ produjo el menor tiempo, con ambas rutas convergiendo a $\sim 0{,}1155$ ms. La razón $T(1)/T(0)$ fue inferior a uno en ambos formatos ($0{,}880$ en BF16), con intervalos de confianza que excluyen la unidad.

Esto no implica que el trabajo en FP64 acelere el kernel, sino que $K = 0$ y $K = 1$ ejecutan cargas distintas. Con $K = 0$, la compensación local aplica dos convoluciones por iteración, sobre el estado cuantizado y el residuo (Sección «Esquemas de compensación numérica evaluados»), con un sobrecosto cercano al $100\,\%$ en la Fase 3. Con $K = 1$, ese esquema se sustituye por una única convolución FP64 (Sección «Anclaje periódico en FP64»). Para $HW = 64$ y $C = K = 64$, esta última resultó más rápida que las dos convoluciones de 16 bits compensadas, explicando $T(1)/T(0) < 1$. La convergencia de FP16 y BF16 en $K = 1$ confirma que allí el tiempo lo determina el paso FP64, no el formato reducido.

Figura 11

Stencil 2D (1024×1024*): tiempo por iteración frente a la periodicidad de anclaje, por formato y horizonte. Cada línea corresponde a una réplica.*



4.4.3.3 Stencil 2D. En Stencil 2D el paso de $K = 0$ a $K = 1$ más que duplica el tiempo, y ambos formatos describen trayectorias prácticamente idénticas: en este operador el formato de 16 bits no determina el tiempo de ejecución. Atenuar la periodicidad a $K = 5$ reduce el sobrecosto de $2{,}14$ a $1{,}63$, pero no en proporción a la frecuencia. Esta razón corresponde a la campaña de variabilidad ($n = 8$, malla 1024^2); la Sección «Efecto del anclaje sobre el error en Stencil compensado» reporta una razón distinta ($2{,}9$–$3{,}2\times$) para la campaña de producción, sobre mallas mayores y sin réplicas independientes.

4.4.3.4 Efecto sobre la comparación entre formatos. La Tabla 20 cuantifica la convergencia observada en las tres figuras anteriores mediante la razón $T_{\text{FP16}}/T_{\text{BF16}}$.

Tabla 20

Razón de tiempo por iteración entre FP16 y BF16 a igual periodicidad de anclaje, con intervalos de confianza al $95\,\%$ *(*$n = 8$*).*

Kernel	$K = 0$	$K = 1$	$K = 5$
GEMM	0,757 [0,757; 0,758]	1,001 [1,000; 1,001]	0,818 [0,817; 0,818]
Convolución 2D	0,902 [0,901; 0,902]	1,002 [1,001; 1,003]	0,925 [0,924; 0,927]
Stencil 2D	0,987 [0,984; 0,990]	1,010 [1,007; 1,014]	1,006 [1,003; 1,009]

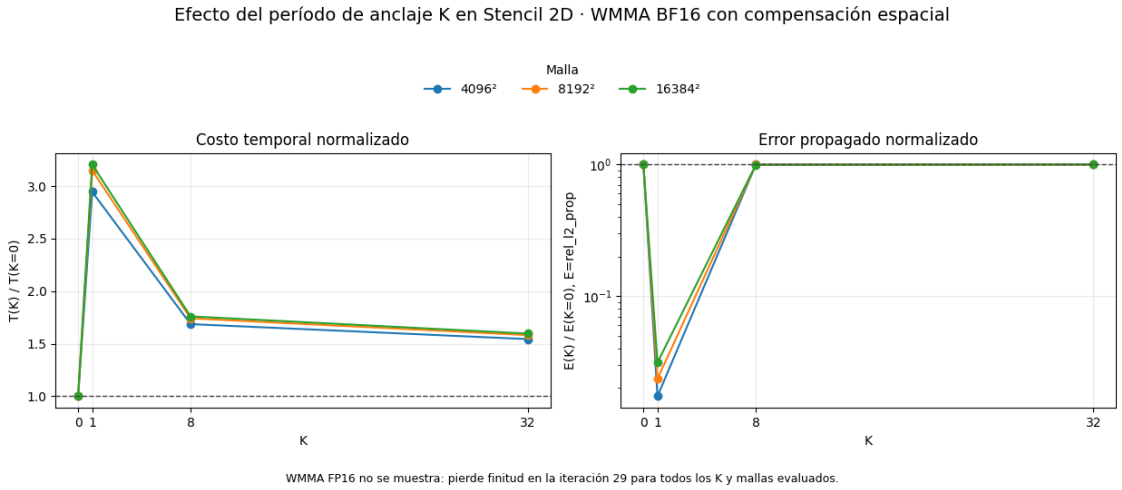
Sin ancla, FP16 supera a BF16 en GEMM ($24{,}3\,\%$) y Convolución 2D ($9{,}8\,\%$), mientras que en Stencil 2D la diferencia es de solo $1{,}3\,\%$, al mover ambos formatos el mismo volumen de datos. Con $K = 1$, esta ventaja desaparece: las razones son estadísticamente indistinguibles de la unidad en los tres kernels. En ese régimen, la elección entre FP16 y BF16 deja de depender del rendimiento.

4.4.4 Efecto del anclaje sobre el error en Stencil compensado

El costo temporal del anclaje debe evaluarse junto con la reducción de error que produce. La Figura 12 contrasta ambas magnitudes sobre la ruta WMMA BF16 con compensación espacial, normalizadas respecto a la ruta sin ancla, sobre la campaña de producción (mallas 4096^2 a 16384^2, $K \in \{0, 1, 8, 32\}$, sin réplicas independientes); no debe confundirse con la razón $T(1)/T(0) = 2{,}14$ de la Sección «Efecto de la frecuencia de anclaje sobre el tiempo», calculada sobre la campaña de variabilidad ($n = 8$, malla 1024^2, $K \in \{0, 1, 5\}$).

Figura 12

Stencil 2D, ruta WMMA BF16 con compensación espacial: costo temporal $T(K)/T(0)$ (izquierda) y error propagado $E(K)/E(0)$ (derecha) según la periodicidad de anclaje, para las tres mallas evaluadas. WMMA FP16 se omite porque pierde la finitud en la iteración 29 para todos los valores de K y mallas.



Con $K = 1$, el anclaje reduce el error propagado entre uno y dos órdenes de magnitud, con $E(1)/E(0) = 0{,}01$–$0{,}03$, a un costo de 2,9–3,2×. Para $K = 8$ y $K = 32$, el error vuelve aproximadamente a su nivel sin ancla, mientras el costo permanece en torno a $1{,}7$ y $1{,}6\times$, respectivamente. Así, espaciar el anclaje conserva parte del sobrecosto pero elimina su beneficio numérico, sin observarse un régimen intermedio útil. Esto concuerda con el crecimiento geométrico de la Fase 3: con $\lambda \approx 2$, ocho iteraciones sin corrección amplifican la desviación en un factor del orden de 2^8.

La campaña difusiva confirma que este comportamiento no depende únicamente de la amplificación. En 8192^2 y $1\,500$ iteraciones, donde la solución de referencia conserva magnitud, $K = 1$ reduce el error de $3{,}1 \times 10^{-5}$ (sin ancla) a $2{,}6 \times 10^{-8}$, $K = 2$ solo a $2{,}2 \times 10^{-5}$ (cerca de un 30 % menos) y $K = 32$ y $K = 64$ lo dejan en $3{,}1 \times 10^{-5}$, sin mejora apreciable (Tabla 28). La malla de 4096^2 a $4\,000$ iteraciones no permite esta comparación: allí la referencia decayó a $\sim 10^{-7}$ y todos los errores relativos, salvo el de $K = 1$, son iguales o superiores a $3{,}4$. Los valores obtenidos con $K = 1$ corresponden al piso de instrumentación (Sección «Anclaje periódico en FP64»): solo

permiten afirmar que la desviación quedó por debajo de la resolución del campo de lectura. El anclaje evita el redondeo del paso corregido, pero no elimina el residuo acumulado entre anclas; por ello, solo $K = 1$ reproduce la trayectoria de doble precisión.

Bajo el operador de estrés, el anclaje tampoco evita el desbordamiento de FP16: la ruta pierde la finitud incluso con $K = 1$, porque el mecanismo reduce el error de cada paso, pero no limita el crecimiento de la magnitud del estado.

Tabla 21

Iteración del primer valor no finito en Stencil 2D por ruta y tamaño de malla, operador de estrés.

Ruta	4096^2	8192^2	16384^2
WMMA FP16 + comp. espacial	29	29	29
WMMA BF16 + comp. espacial	142	142	142
GPU FP32	142	142	142
CPU FP32	142	142	142
GPU FP64	1045	1044	1044
CPU FP64	1045	1044	1044

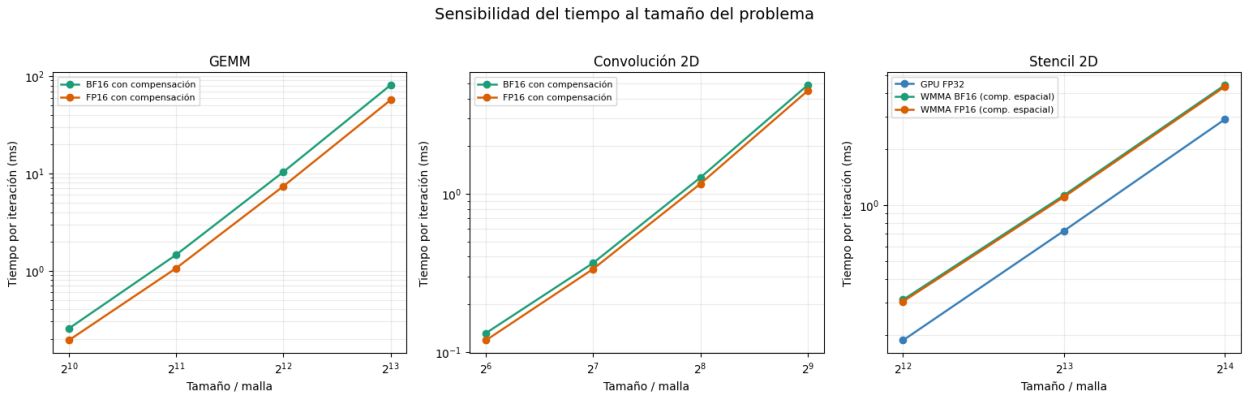
Estos horizontes proceden de la misma campaña que la Tabla 14 de la Fase 3 y la reproducen. Que BF16 y FP32 compartan horizonte indica que el límite de los formatos de 16 bits bajo operadores amplificantes es de rango dinámico y no de resolución: comparten un campo de exponente de ocho bits pese a que BF16 tiene la mantisa más corta de los cuatro formatos evaluados.

4.4.5 Sensibilidad al tamaño del problema

Las comparaciones anteriores se establecieron a un tamaño fijo por kernel. La Figura 13 verifica si se sostienen al variar la escala.

Figura 13

Tiempo por iteración frente al tamaño del problema en los tres kernels. GEMM y Convolución 2D: rutas de 16 bits con compensación local. Stencil 2D: rutas WMMA con compensación espacial frente a la referencia GPU FP32.



En GEMM la ventaja de FP16 sobre BF16 se ensancha con el tamaño: ambas rutas son indistinguibles en 2^{10} y se separan de forma apreciable en 2^{13}. En Convolución 2D escalan superpuestas en todo el rango, de modo que a esas escalas la ruta aritmética no determina el tiempo.

En el panel de Stencil 2D, las rutas WMMA con compensación espacial son entre $1{,}4$ y $1{,}7$ veces más lentas que GPU FP32 en las tres mallas, sin una tendencia clara con el tamaño, y resultan indistinguibles entre sí, en concordancia con la razón $T_{\mathrm{FP16}}/T_{\mathrm{BF16}}$ de $0{,}987$ de la Tabla 20. En este operador, ni los Tensor Cores ni la elección de formato reducen el tiempo frente a una implementación clásica en simple precisión.

4.4.6 Energía y EDP por configuración

Para cada kernel se calculó el *Energy-Delay Product* (Ecuación (9)) de toda configuración que superó el filtro de admisión: error relativo L_2 finito y estrictamente positivo en el punto de observación numérico común, y energía de GPU medida en una ventana fiable. Las configuraciones admitidas se agruparon por esquema de compensación y anclaje. Cada barra de las Figuras 14 y 15 es el EDP *mínimo* observado dentro de su categoría y tamaño, no un promedio; las Tablas 22–24 identifican la configuración que alcanza ese mínimo. GEMM y Convolución 2D proceden de la campaña con el operador de estrés; Stencil 2D, de la campaña difusiva (Sección «Stencil 2D»).

La comparación es válida dentro de cada tamaño. En GEMM y Convolución 2D todas las candidatas de un mismo tamaño comparten la ventana energética, de modo que el EDP de la ventana las ordena correctamente; las ventanas difieren entre tamaños, por lo que de aquí no se infiere una ley de escalamiento del EDP con N. En Stencil 2D las candidatas de una misma malla se midieron con ventanas de distinta longitud (entre $1\,500$ y $16\,000$ iteraciones), y como el EDP de una ventana crece con el cuadrado de su longitud, allí se reporta el EDP por iteración, producto de la energía y el tiempo por iteración.

4.4.6.1 GEMM. La ruta sin compensación tuvo el menor EDP en los cuatro tamaños: FP16 en $N = 1024$ y $N = 4096$, BF16 en $N = 2048$ y $N = 8192$, donde ninguna configuración FP16 sin compensación superó el filtro de admisión. Ninguna ruta compensada ni anclada alcanzó el menor EDP en ningún tamaño: su costo temporal y energético no se compensa dentro de esta métrica agregada (Tabla 22, Figura 14). Este filtro evalúa el error en el punto de observación de la pasada numérica (iteración 80), donde BF16 con $N = 2048$ conserva una solución finita con error positivo; el análisis de dominancia lo evalúa en el horizonte común del estrato ($24\,000$ iteraciones), donde el registro de $N = 2048$ es nulo y se descarta (Tabla 26). Por eso $N = 2048$ participa en el EDP y no en el frente de Pareto.

Figura 14

GEMM: EDP mínimo válido por categoría y tamaño del problema (escala logarítmica).

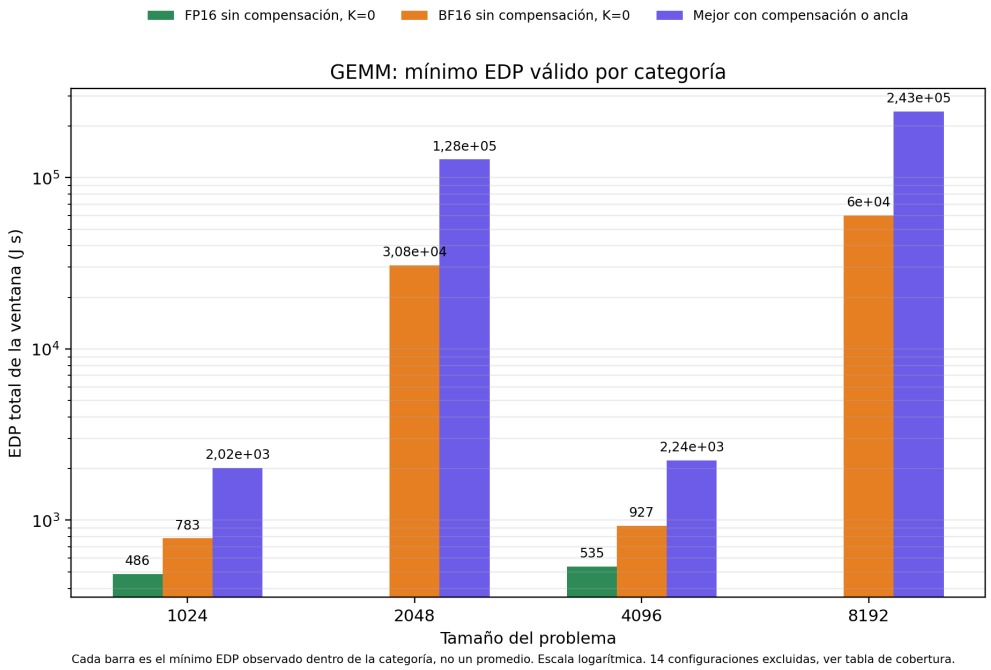


Tabla 22

GEMM: configuración de menor EDP por categoría y tamaño.

Serie	Configuración	Tiempo (ms)	Potencia (W)	Energía (J)	EDP (J s)
BF16 sin compensación, $K = 0$ ($N = 1024$)	BF16, sin compensación	3 009	86,50	260,3	783,4
BF16 sin compensación, $K = 0$ ($N = 2048$)	BF16, sin compensación	17 230	103,8	1 789	30 820
BF16 sin compensación, $K = 0$ ($N = 4096$)	BF16, sin compensación	2 574	139,9	360,1	927,0
BF16 sin compensación, $K = 0$ ($N = 8192$)	BF16, sin compensación	20 150	147,7	2 977	60 000
FP16 sin compensación, $K = 0$ ($N = 1024$)	FP16, sin compensación	2 268	94,47	214,2	485,8
FP16 sin compensación, $K = 0$ ($N = 4096$)	FP16, sin compensación	1 829	160,0	292,7	535,5
Mejor con compensación o ancla ($N = 1024$)	FP16, compensación local, $K = 0$	4 586	96,14	440,9	2 022
Mejor con compensación o ancla ($N = 2048$)	BF16, compensación local, $K = 0$	34 850	105,1	3 661	127 600
Mejor con compensación o ancla ($N = 4096$)	FP16, compensación local, $K = 0$	3 686	164,8	607,3	2 239
Mejor con compensación o ancla ($N = 8192$)	BF16, compensación local, $K = 0$	40 420	148,5	6 004	242 700

4.4.6.2 Convolución 2D. BF16 sin compensación presentó el menor EDP en los cuatro tamaños ($HW = 64$, 128, 256 y 512). FP16 quedó fuera del ranking porque su error terminal

en iter $= 80$ fue nulo o no finito; aquí, un error nulo se considera inadmisible y no evidencia de exactitud. Las referencias GPU FP64 tampoco participaron al carecer de ese punto de observación. La compensación y el anclaje mostraron mayor EDP que la ruta sin compensar en todos los tamaños (Tabla 23, Figura 15).

Figura 15

Convolución 2D: EDP mínimo válido por categoría y tamaño del problema (escala logarítmica).

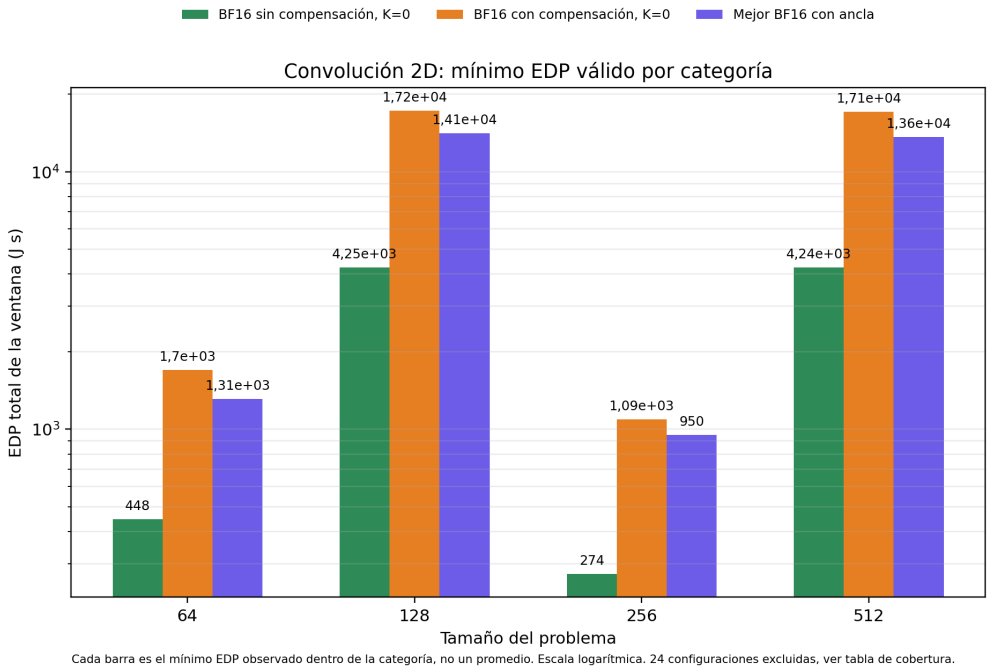


Tabla 23

Convolución 2D: configuración de menor EDP por categoría y tamaño.

Serie	Configuración	Tiempo (ms)	Potencia (W)	Energía (J)	EDP (J s)
BF16 sin compensación, $K = 0$ ($HW = 64$)	BF16, sin compensación	2 483	72,62	180,3	447,7
BF16 sin compensación, $K = 0$ ($HW = 128$)	BF16, sin compensación	6 749	93,36	630,1	4 252
BF16 sin compensación, $K = 0$ ($HW = 256$)	BF16, sin compensación	1 583	109,4	173,3	274,3
BF16 sin compensación, $K = 0$ ($HW = 512$)	BF16, sin compensación	6 095	114,2	696,1	4 242
BF16 con compensación, $K = 0$ ($HW = 64$)	BF16, compensación local	4 797	73,65	353,3	1 695
BF16 con compensación, $K = 0$ ($HW = 128$)	BF16, compensación local	13 460	95,05	1 280	17 230
BF16 con compensación, $K = 0$ ($HW = 256$)	BF16, compensación local	3 154	109,7	346,2	1 092
BF16 con compensación, $K = 0$ ($HW = 512$)	BF16, compensación local	12 150	115,8	1 407	17 100
Mejor BF16 con ancla ($HW = 64$)	BF16, compensación local, $K = 1$	4 184	74,80	313,0	1 310
Mejor BF16 con ancla ($HW = 128$)	BF16, compensación local, $K = 1$	11 600	104,8	1 216	14 110
Mejor BF16 con ancla ($HW = 256$)	BF16, compensación local, $K = 1$	2 791	122,0	340,6	950,4
Mejor BF16 con ancla ($HW = 512$)	BF16, compensación local, $K = 1$	10 440	125,2	1 307	13 640

4.4.6.3 Stencil 2D. En la campaña con el operador difusivo, medido por iteración, el orden de las tres categorías es el mismo en ambas mallas: la ruta sin compensación tiene el menor EDP, la compensación espacial un EDP entre $4{,}5$ y $4{,}6$ veces mayor, y Kahan local el mayor, entre $5{,}2$ y $6{,}3$ veces el de la ruta sin compensar (Tabla 24). Dentro de la compensación espacial, el menor EDP corresponde a $K = 0$ en ambas mallas: todo anclaje añade costo sin reducir el tiempo ni la energía por iteración.

Tabla 24

Stencil 2D (campaña con operador difusivo): configuración de menor EDP por iteración en cada categoría y malla. Tiempo y energía por iteración proceden de la ventana fiable de cada configuración.

Serie	**Configuración**	**Malla**	T_{iter} **(ms)**	E_{iter} **(J)**	**EDP por iter. (J ms)**
Sin compensación, $K = 0$	WMMA BF16	4096^2	0,1034	0,0238	0,0025
	WMMA FP16	8192^2	0,3697	0,0903	0,0334
Kahan local, $K = 0$	WMMA FP16	4096^2	0,2485	0,0512	0,0127
	WMMA FP16	8192^2	0,9403	0,2224	0,2091
Comp. espacial, K óptimo	WMMA FP16, $K = 0$	4096^2	0,2104	0,0523	0,0110
	WMMA BF16, $K = 0$	8192^2	0,8267	0,1838	0,1520

4.4.6.4 Cobertura. Superaron el filtro de admisión 20 de 44 configuraciones en Convolución 2D, 30 de 44 en GEMM y las 36 de 36 evaluadas en Stencil 2D; ninguna configuración de Stencil 2D se excluyó en esta campaña. La categoría «FP16 sin compensación, $K = 0$» de GEMM no tiene configuración válida en $N = 2048$ ni en $N = 8192$: el hueco no se completó con una estimación ni con la barra de otra categoría. Reproducirlo exige una campaña con un punto de observación del error válido en esos tamaños, que no forma parte de las campañas realizadas.

4.4.7 Análisis multiobjetivo tiempo–energía–error

4.4.7.1 Criterios de admisión. Para situar una configuración en las tres coordenadas, las tres métricas deben proceder de la misma configuración canónica y ser individualmente válidas. El análisis se construyó sobre una auditoría que aplica los criterios de la Tabla 25 antes de evaluar dominancia alguna, y que emite para cada configuración descartada el motivo explícito de su exclusión.

Tabla 25

Criterios de admisión aplicados antes del análisis de dominancia.

Criterio	**Implementación**
Identidad de la configuración	Agregación canónica por (tamaño, formato, compensación, ruta, K efectivo), con cruce uno a uno entre métricas. $K = 0$ para rutas de referencia y sin anclaje.
Horizonte común del error	El error de todas las candidatas de un estrato se evalúa en el mismo horizonte de iteraciones.
Tasas por iteración	Tiempo y energía se expresan por iteración y proceden de la ventana de medición fiable más larga disponible para la configuración; se verifica que el tiempo por iteración no dependa de la longitud de la corrida.
Fiabilidad de la energía	Ventana de medición válida; las rutas WMMA de Stencil 2D exigen al menos 1 s. Si no se cumple, la configuración se excluye.
Validez del error terminal	Punto de observación presente, solución finita y error estrictamente positivo. Se excluye la configuración sin punto de observación, con solución no finita o con error nulo.
Referencias	FP64 y FP32 se representan pero nunca compiten; cada estrato exige al menos una candidata con error finito positivo.
Estratos comparables	Se descarta el estrato con menos de dos candidatas admisibles.
Dominancia	Por estrato (kernel, tamaño), minimizando tiempo y energía de GPU por iteración y error relativo L_2 en el horizonte común.

Separar el horizonte del error de la ventana energética es lo que permite construir el análisis en los tres kernels. El tiempo y la energía por iteración son tasas de régimen estacionario: en Convolución 2D, el tiempo por iteración medido en la pasada energética ($37\,000$ o $2\,500$ iteraciones) coincide con el de la pasada numérica a 40 iteraciones con una razón entre $0{,}97$ y $1{,}00$ en todas las configuraciones, incluidas las rutas FP16 que ya no son finitas. El error, en cambio, depende del horizonte y por eso se fija en uno común para todo el estrato. La Tabla 26 resume los estratos resultantes.

Tabla 26

Estratos admitidos y descartados en el análisis de dominancia.

Kernel	Estrato	Resultado
GEMM	$N = 1024$, error a 24 000 it.	Admitido
	$N = 4096$, error a 500 it.	Admitido
	$N = 2048$, error a 24 000 it.	Descartado: dato terminal inválido (error reportado nulo)
	$N = 8192$, error a 500 it.	Descartado: solución no finita
Conv. 2D	$HW = 64$–512, error a 40 it.	Admitidos los cuatro; solo candidatas BF16 (FP16 no finita en el horizonte)
Stencil 2D	4096^2, error a 4 000 it.	Admitido: 16 candidatas
	8192^2, error a 1 500 it.	Admitido: 16 candidatas

Las exclusiones responden a dos causas distintas: invalidez del dato terminal ($N = 2048$, con un error reportado nulo) y pérdida de finitud ($N = 8192$ en GEMM y las rutas FP16 de Convolución 2D). Solo la segunda constituye evidencia de comportamiento numérico; la primera es una limitación de cobertura experimental. Un error nulo no se interpreta como exactitud: con el factor de amplificación $\sqrt{2}$ por iteración que el operador tiene en $N = 2048$ (Sección «Estabilidad numérica y aparición de valores no finitos»), ningún formato conservaría una solución finita tras $24\,000$ iteraciones, por lo que ese valor se trata como un registro inválido y no como una solución exacta. En Stencil 2D, con el operador difusivo, ninguna configuración fue excluida.

4.4.7.2 Frente de Pareto de GEMM. Las cuatro rutas candidatas —FP16 y BF16, con y sin compensación local— aparecen en ambos estratos, con el anclaje barrido en las cuatro periodicidades de la campaña de producción ($K = 0, 1, 5, 20$). La referencia FP64 se sitúa en $0{,}233$ ms y $0{,}0260$ J por iteración en $N = 1024$, y en $13{,}12$ ms y $1{,}562$ J por iteración en $N = 4096$.

La figura 16 muestra dos resultados. Primero, tiempo y energía son casi colineales en este kernel: las configuraciones se alinean a lo largo de una recta de pendiente positiva en ambos es-

tratos. En GEMM, por tanto, el eje energético no aporta información independiente del temporal, y el compromiso efectivo es bidimensional, entre rendimiento y exactitud. Segundo, las rutas sin compensación ocupan el extremo de menor costo del frente y las compensadas el de mayor costo, con la referencia FP64 situada dentro del rango de costo de estas últimas: en estos estratos, el costo de la doble precisión es comparable al de una ruta de 16 bits con compensación.

La identificación nominal de cada configuración del frente se encuentra en la tabla de auditoría asociada a la figura, que registra para cada punto su ruta, formato, periodicidad de anclaje y estado de dominancia.

Figura 16

GEMM: proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para los dos estratos admitidos. El color codifica el error relativo L_2 en el mismo horizonte; la línea discontinua une las configuraciones no dominadas. FP64 se representa como referencia y no participa en la dominancia.

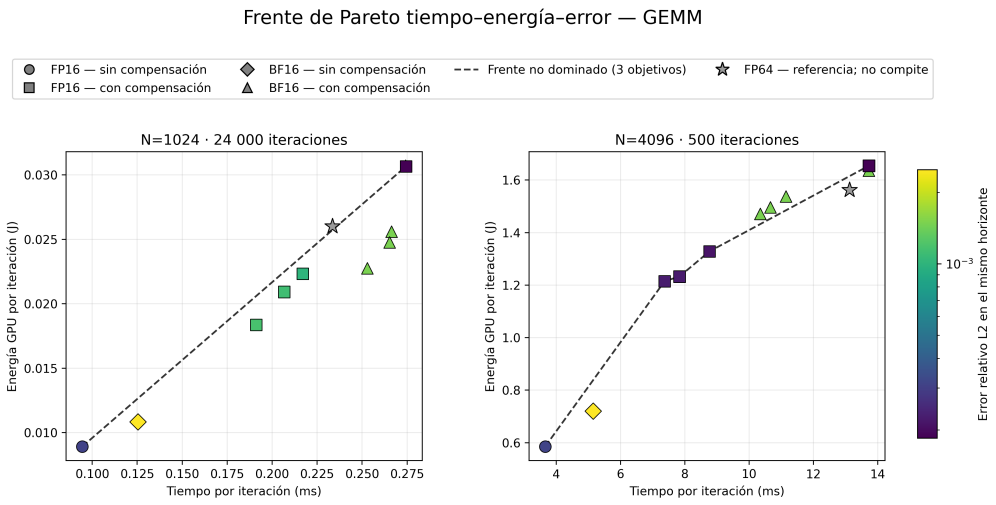
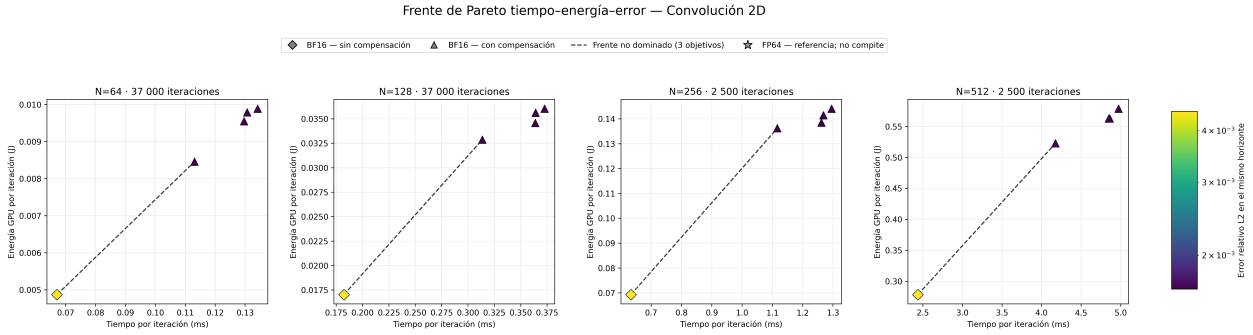


Figura 17

Convolución 2D: proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para los cuatro estratos admitidos. El color codifica el error relativo L_2; la línea discontinua une las configuraciones no dominadas.



4.4.7.3 Frente de Pareto de Convolución 2D. Convolución 2D admite cuatro estratos, el doble que GEMM, pero con un espacio de candidatas más estrecho: únicamente las rutas BF16, con y sin compensación local, ancladas con las mismas cuatro periodicidades de GEMM ($K = 0, 1, 5, 20$). Todas las rutas FP16 fueron descartadas por pérdida de finitud en el punto de observación terminal, en los cuatro tamaños, de forma consistente con la Fase 3, que la midió a partir de la iteración 20 en los cuatro tamaños evaluados, con y sin compensación.

La referencia FP64 no aparece en ninguno de los cuatro paneles: la auditoría no pudo verificar su finitud en el punto terminal de estos estratos, y el criterio de referencia verificada impidió representarla. Es una limitación de la instrumentación de esta campaña, no un resultado sobre la doble precisión.

La estructura del frente se repite en los cuatro tamaños y es más simple que la de GEMM: la ruta sin compensación ocupa el extremo de menor costo, con el error mayor del estrato, y una única configuración compensada completa el frente; las demás compensadas quedan dominadas en la región de mayor costo. La colinealidad entre tiempo y energía observada en GEMM se reproduce en los cuatro estratos: en los kernels de alta intensidad aritmética evaluados, la energía de GPU es aproximadamente proporcional al tiempo.

4.4.7.4 Kahan local frente a compensación espacial. La comparación entre las dos variantes de compensación de Stencil 2D, Kahan local y espacial (Sección «Esquemas de compensación numérica evaluados»), se evalúa sobre la campaña con el operador difusivo ($\alpha = 3/16$), con ambos mecanismos bajo la misma configuración ($K = 0$, sin anclaje); bajo el operador de estrés se compararon en la Sección «Estabilidad y variabilidad experimental». Se usa la malla de 8192^2 a $1\,500$ iteraciones, donde la solución de referencia conserva magnitud y los errores relativos son interpretables; en la de 4096^2 a $4\,000$ iteraciones la referencia ya decayó a $\sim 10^{-7}$ y ninguna ruta de 32 o 16 bits conserva cifras correctas (Sección «Stencil 2D»). El error relativo se mide contra la referencia CPU FP64; el sobrecosto temporal, contra la ruta sin compensación del mismo formato (Tabla 27).

Tabla 27

Kahan local frente a compensación espacial, $K = 0$, malla 8192^2 a $1\,500$ iteraciones: error relativo L_2 y sobrecosto temporal respecto a la ruta sin compensación del mismo formato.

Formato	Esquema	Error relativo	Sobrecosto de tiempo
FP16	Kahan local	$2{,}5 \times 10^{-2}$	$2{,}544\times$
FP16	Compensación espacial	$3{,}1 \times 10^{-5}$	$2{,}130\times$
BF16	Kahan local	$2{,}0 \times 10^{-1}$	$2{,}431\times$
BF16	Compensación espacial	$3{,}1 \times 10^{-5}$	$2{,}084\times$

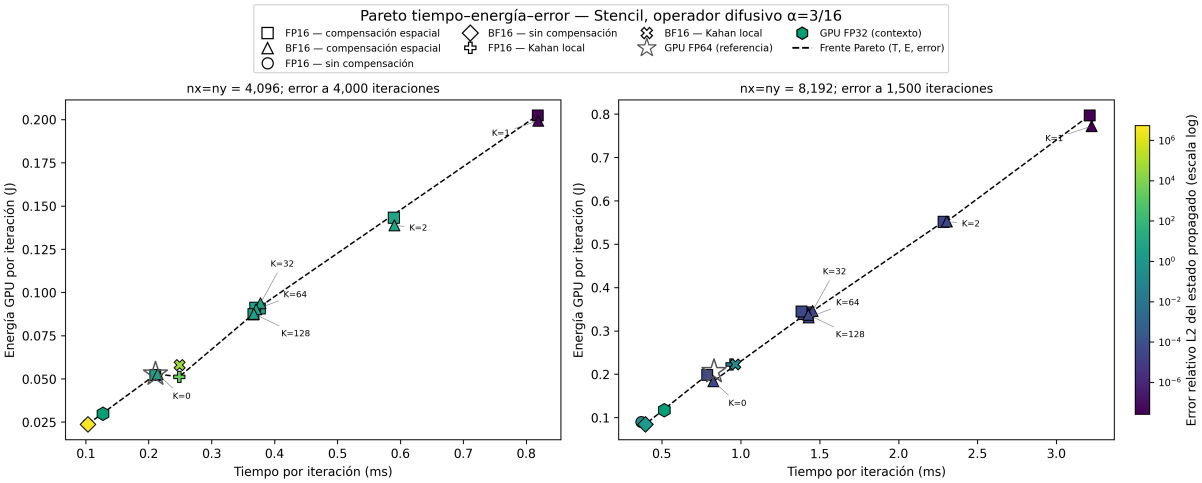
En FP16, la compensación espacial redujo el error relativo a $3{,}1 \times 10^{-5}$ frente a $2{,}5 \times 10^{-2}$ con Kahan local, unas 800 veces menos; en BF16, a $3{,}1 \times 10^{-5}$ frente a $2{,}0 \times 10^{-1}$, unas $6\,500$ veces menos. También presentó menor sobrecosto temporal: $2{,}130\times$ y $2{,}084\times$ en FP16 y BF16, frente a $2{,}544\times$ y $2{,}431\times$ con Kahan local ($0{,}787$ frente a $0{,}940$ ms por iteración en FP16; Tabla 28). En esta campaña, la compensación espacial fue simultáneamente más rápida y más exacta que Kahan local.

Este resultado corresponde al operador difusivo, donde ninguna ruta pierde la finitud, y no debe extrapolarse al operador de estrés de la Fase 3 (Sección «Costo y utilidad de la compen-

sación»). Además, no se dispone de mediciones de tráfico DRAM para comparar ambos esquemas en ese recurso.

Figura 18

Stencil 2D, operador difusivo ($\alpha = 3/16$): proyección tiempo–energía del análisis de dominancia sobre tres objetivos, para las dos mallas evaluadas. El color codifica el error relativo L_2 del estado propagado en el horizonte común de cada malla; la forma, formato y compensación; la discontinua une las configuraciones no dominadas. GPU FP64 y GPU FP32 son referencia y no participan en la dominancia.



Operador difusivo α=3/16, condición inicial monomodo (campaña 7757). Error = rel_l2 del estado propagado frente a FP64 al horizonte común de cada malla (4096²: 4000 it; 8192²: 1500 it), igual para todos los candidatos. T y E = tasas por iteración de la ventana de energía fiable más larga disponible (4096²: [4000, 8000, 16000] it, 8192²: [1500, 3000] it; las rutas WMMA exigen ≥1 s de ventana NVML). A 4000 it (4096²) la solución de referencia ha decaído varios órdenes de magnitud: errores relativos del orden de la unidad sin ancla indican que el residuo de redondeo supera la solución remanente, no una divergencia (todo es finito). Frente = no dominado en (T, E, error) solo entre candidatos de precisión reducida; GPU FP64/FP32 se muestran como referencia y no compiten. Caso límite: en 4096², FP16 con Kahan local permanece en el frente por consumir aproximadamente 2 % menos energía que FP16 espacial con K=0 (0,0512 frente a 0,0523 J), aunque su error es cerca de 700 veces mayor; se considera un empate técnico en energía.

4.4.7.5 Stencil 2D. Bajo el operador de estrés no existe una ventana con tiempo, energía y error simultáneamente válidos: FP16 pierde la finitud en la iteración 29, BF16 y FP32 en la 142, y FP64 en la 1045 (Tabla 21), antes de obtener una medición energética fiable. Por ello, el frente de Stencil 2D se construyó con una campaña difusiva ($\alpha = 3/16$, condición inicial monomodo), en la que todas las rutas permanecen finitas. Un único modo de Fourier es un autovector del Laplaciano discreto, de modo que la solución decae como ese solo modo, sin mezcla entre tasas de decaimiento distintas, lo que simplifica la evolución del error. Las conclusiones de esta campaña quedan acotadas, por tanto, a ese caso particular; no se evaluó una condición inicial multimodo.

Se evaluaron 4096^2 y 8192^2, con dieciséis candidatas por malla: WMMA en FP16 y BF16

con compensación espacial y $K \in \{0, 1, 2, 32, 64, 128\}$, además de variantes sin compensación y con Kahan local. El error se midió a 4000 iteraciones en 4096^2 y a 1500 en 8192^2; las rutas más rápidas requirieron ventanas energéticas de hasta $16\,000$ iteraciones para superar el umbral de fiabilidad de un segundo. Ninguna configuración fue excluida.

Tabla 28

Stencil 2D, operador difusivo: tiempo y energía de GPU por iteración y error relativo L_2 en el horizonte común, para las configuraciones representativas de cada malla. Se indica si la configuración pertenece al frente (candidatas) y, en su caso, si alguna referencia la domina.

Malla	Configuración	T (ms)	E (J)	Error	Frente
4096^2	WMMA FP16 sin compensación	0,103	0,024	$2{,}1 \times 10^{5}$	Sí
	GPU FP32 (referencia)	0,127	0,030	5,0	—
	WMMA FP16 espacial, $K = 0$	0,210	0,052	3,6	Sí
	GPU FP64 (referencia)	0,211	0,053	$1{,}6 \times 10^{-8}$	—
	WMMA FP16 Kahan local	0,248	0,051	$2{,}5 \times 10^{3}$	Sí*
	WMMA FP16 espacial, $K = 128$	0,366	0,087	3,5	Sí†
	WMMA FP16 espacial, $K = 2$	0,589	0,143	5,7	No
	WMMA FP16 espacial, $K = 1$	0,818	0,202	$3{,}1 \times 10^{-8}$	Sí†
8192^2	WMMA FP16 sin compensación	0,370	0,090	0,29	Sí
	GPU FP32 (referencia)	0,515	0,117	$4{,}7 \times 10^{-5}$	—
	WMMA FP16 espacial, $K = 0$	0,787	0,198	$3{,}1 \times 10^{-5}$	Sí
	GPU FP64 (referencia)	0,831	0,206	$1{,}3 \times 10^{-13}$	—
	WMMA FP16 Kahan local	0,940	0,222	$2{,}5 \times 10^{-2}$	No
	WMMA FP16 espacial, $K = 32$	1,427	0,343	$3{,}1 \times 10^{-5}$	Sí†
	WMMA FP16 espacial, $K = 2$	2,284	0,552	$2{,}2 \times 10^{-5}$	Sí†
	WMMA FP16 espacial, $K = 1$	3,210	0,797	$2{,}6 \times 10^{-8}$	Sí†

* Empate técnico en energía: permanece en el frente por consumir un 2 % menos que la ruta espacial con $K = 0$, con un error unas 700 veces mayor. † No dominada entre las candidatas, pero sí por GPU FP64. Las rutas BF16 con compensación espacial reproducen el error de FP16 con diferencias bajo el 1 %. El error de las filas con $K = 1$ (0,52 y 0,44 veces $u = 2^{-24}$) es el piso de instrumentación del campo de lectura (Sección «Anclaje periódico en FP64»), no una medida de la exactitud interna del anclaje. GPU FP64 domina esas configuraciones en tiempo y energía, donde las supera por un factor de 3,9 en ambas mallas, con independencia del error.

La Figura 18 confirma la colinealidad entre tiempo y energía, por lo que el compromiso

efectivo vuelve a ser bidimensional. Las rutas sin compensación ocupan el extremo de menor costo —incluso por debajo de GPU FP32—, pero con errores del orden de 10^5–10^6 en 4096^2 y $0{,}3$–$1{,}6$ en 8192^2. La compensación espacial con $K = 0$ reduce ese error entre cuatro y seis órdenes de magnitud a costa de duplicar aproximadamente el tiempo, mientras que solo $K = 1$ lo lleva al orden de 10^{-8}, con un costo cercano a cuatro veces el de $K = 0$; este último valor corresponde al piso de instrumentación del campo de lectura (Sección «Anclaje periódico en FP64»), no a una medida directa de la exactitud interna.

La referencia GPU FP64 ocupa una posición favorable. En 4096^2 iguala prácticamente a la ruta espacial con $K = 0$ en tiempo y energía ($0{,}211$ frente a $0{,}210$ ms y $0{,}053$ frente a $0{,}052$ J por iteración), con un error ocho órdenes de magnitud menor. En 8192^2, las rutas espaciales solo la superan hasta en un $5\,\%$ en tiempo y un $11\,\%$ en energía, también con un error ocho órdenes mayor. Todas las configuraciones ancladas quedan dominadas por FP64; Kahan local también en 8192^2 y, en 4096^2, solo evita la dominancia por unos $2\,\%$ en energía. Ello es consistente con el costo de memoria del esquema: una ruta FP16 anclada conserva 10 bytes por celda frente a 8 bytes en FP64 (Sección «Anclaje periódico en FP64»), lo que elimina la ventaja de almacenamiento en este operador limitado por memoria.

La compensación espacial también supera a Kahan local en tiempo y exactitud, y bajo este esquema FP16 y BF16 producen el mismo error, lo que indica que la precisión propagada está determinada por la acumulación. Los errores de ambas mallas no deben compararse directamente: a 4000 iteraciones, la referencia de 4096^2 decae de 7×10^{-3} en la iteración 500 a $1{,}4\times10^{-7}$ en la 4000, de modo que errores relativos del orden de la unidad reflejan que el redondeo supera la solución remanente, no una divergencia. La comparación válida es, por tanto, entre configuraciones dentro de cada malla. Además, en 4096^2 ninguna ruta de 32 o 16 bits conserva exactitud a ese horizonte, salvo las ancladas con $K = 1$, que evalúan todas las iteraciones en FP64: las demás, incluida GPU FP32 (error relativo de $5{,}0$), tienen errores relativos iguales o superiores a $3{,}5$ (Tabla 28), de modo que el orden del frente en esa malla refleja sobre todo el costo y no una diferencia de exactitud utilizable. La conclusión sobre la posición de GPU FP64 se apoya, por ello, en el estrato 8192^2,

donde GPU FP32 y la compensación espacial conservan errores del orden de 10^{-5} y la brecha de ocho órdenes de magnitud frente a FP64 es significativa.

4.4.8 Alcance y limitaciones de la campaña

Cuatro restricciones delimitan estos resultados. Primero, la energía incluye solo la GPU; al excluir el consumo del anfitrión, los frentes de Pareto pueden sobreestimar el beneficio energético de las rutas más rápidas.

Segundo, error y energía proceden de pasadas distintas: el error se evalúa en un horizonte común y la energía en ventanas más largas y fiables. Su emparejamiento supone que tiempo y energía por iteración no dependen de la longitud de la corrida; la auditoría conserva la procedencia de cada métrica.

Tercero, el análisis multiobjetivo de Stencil 2D usa el operador difusivo, no el de estrés empleado para tiempo, variabilidad y supervivencia. Además, se limita a 4096^2 y 8192^2, cuyos horizontes de error no son comparables entre sí.

Por último, los intervalos de confianza se basan en ocho réplicas. Permiten respaldar diferencias claras —como el sobrecosto del anclaje en GEMM y Stencil 2D, la ausencia de sobrecosto en Convolución 2D y la pérdida de ventaja de FP16 con anclaje—, pero no diferencias más sutiles.

4.4.9 Síntesis de la Fase 4

El costo del anclaje depende del kernel. En la campaña de variabilidad, $K = 1$ elevó el tiempo de Stencil 2D a aproximadamente $2{,}1\times$ y el de GEMM entre un 9 y un $44\,\%$, mientras que en Convolución 2D ($HW = 64$) no produjo sobrecosto. En el barrido de sensibilidad de Stencil 2D sobre 4096^2–16384^2, el costo de $K = 1$ fue de 2,9–3,2×.

En Stencil 2D, $K = 1$ redujo el error entre uno y dos órdenes de magnitud, mientras que con $K = 8$ o $K = 32$ el costo permaneció en 1,6–1,7× y el error volvió aproximadamente al nivel sin ancla. Con $K = 1$, la métrica alcanza el piso de instrumentación (Sección «Anclaje periódico en FP64»), por lo que la reducción observada constituye una cota. La campaña difusiva confirma que espaciar el anclaje elimina su beneficio numérico sin reducir proporcionalmente su costo.

El anclaje también modifica la comparación entre formatos: sin ancla, FP16 supera a BF16

en GEMM en un $24{,}3\,\%$, pero con $K = 1$ esa diferencia desaparece en los tres kernels. La elección pasa entonces a depender de la estabilidad numérica: BF16 alcanza 142 iteraciones frente a 29 de FP16, y el anclaje no extiende ese horizonte.

Finalmente, el análisis multiobjetivo pudo construirse en los tres kernels separando el horizonte de error de las ventanas energéticas fiables. Tiempo y energía fueron casi colineales, por lo que el compromiso efectivo se concentra en rendimiento y exactitud. En Stencil 2D difusivo, GPU FP64 igualó aproximadamente el costo de la mejor ruta WMMA compensada y logró un error ocho órdenes de magnitud menor; dominó todas las configuraciones ancladas y a Kahan local salvo el empate técnico de energía en 4096^2. Así, en este operador limitado por memoria, la precisión reducida solo conserva ventaja de costo cuando renuncia a la corrección, a cambio de un error considerablemente mayor.

4.5 Discusión general

Los resultados obtenidos en las cuatro fases muestran que el impacto de la precisión mixta no puede caracterizarse únicamente mediante el *throughput*. La utilidad de los Tensor Cores depende del patrón computacional, de la implementación utilizada y de la sensibilidad numérica del algoritmo.

4.5.1 Influencia del patrón computacional

GEMM presentó el escenario más favorable para el aprovechamiento de Tensor Cores. La ruta FP16 optimizada mediante cuBLAS alcanzó $15{,}27\times$ respecto a la referencia FP64 en GPU para $N = 12\,288$, y $16{,}50\times$ respecto a la ruta GPU clásica en FP32 para $N = 32\,768$. En Convolución 2D la ruta tensorial de cuDNN alcanzó $28{,}07\times$ frente a FP64 y $15{,}30\times$ frente a FP32 con `IMPLICIT_GEMM` ($6{,}83\times$ frente al algoritmo FP32 por defecto de cuDNN).

Stencil 2D mostró un comportamiento diferente. Debido a su menor intensidad aritmética, el movimiento de datos representa una fracción mayor del costo de ejecución y limita el beneficio obtenible mediante una mayor capacidad de cómputo tensorial. La línea base ya operaba al 55–$80\,\%$ del ancho de banda pico de HBM2, dejando escaso margen de mejora por la vía aritmética.

En consecuencia, los resultados obtenidos para GEMM no pueden extrapolarse directa-

mente a kernels con patrones de acceso a memoria diferentes.

4.5.2 Compromiso entre rendimiento y exactitud

La reducción de precisión permitió incrementar de forma apreciable el rendimiento en los kernels con mayor intensidad aritmética, pero introdujo diferencias numéricas respecto a FP64, cuya magnitud y evolución dependió del formato y de la estructura del algoritmo.

En una operación aislada, GEMM y Convolución 2D pueden caracterizarse mediante el error de la salida, del orden de 10^{-4} en las rutas tensoriales FP16; en los esquemas encadenados es necesario analizar además cómo esa diferencia evoluciona entre iteraciones. Una única métrica de error final es insuficiente para describir un esquema iterativo: en Convolución 2D el error relativo de BF16 sin compensación creció un factor de $2{,}9$ en 80 iteraciones, y en los tres kernels la solución de 16 bits puede perder la finitud antes del final de la trayectoria.

4.5.3 Diferencias entre FP16 y BF16

La campaña de *drift* permitió separar el efecto del rango dinámico del de la resolución. En Stencil 2D, FP16 alcanzó el desbordamiento en la iteración 29 y BF16 en la 142, igualando el horizonte de FP32 pese a tener la mantisa más corta de los cuatro formatos evaluados. En GEMM y Convolución 2D, donde el error puede compararse antes de la pérdida de finitud, BF16 sin compensación presentó errores relativos en torno a ocho veces mayores que FP16, en correspondencia con la razón entre sus unidades de redondeo.

La Fase 4 mostró que el anclaje modifica ese balance. Sin ancla, FP16 aventaja a BF16 en un $24{,}3\,\%$ en GEMM; con anclaje en cada iteración la diferencia temporal desaparece en los tres kernels y la elección pasa a decidirse exclusivamente sobre el horizonte numérico.

Por tanto, la conveniencia de FP16 o BF16 no depende solo de sus formatos numéricos, sino del algoritmo y de las estrategias de corrección empleadas.

4.5.4 Costo y utilidad de la compensación

Las Fases 3 y 4 responden a preguntas distintas. La Fase 3 estudia la evolución del error y el horizonte numérico en un régimen que permite observar la pérdida de finitud de las configuraciones reducidas; la Fase 4 evalúa el costo y el efecto de sustituir periódicamente el paso de baja precisión

por un paso en FP64 (anclaje).

En la Fase 3, la compensación local redujo el error por factores de $1{,}68$ (GEMM) y $5{,}1$–$5{,}2$ (Convolución 2D) sin desplazar el horizonte de desbordamiento, con un sobrecosto temporal cercano al $100\,\%$. La comparación entre esquemas mostró además que no todos presentan el mismo balance: la compensación espacial, que realimenta el residuo sobre la vecindad del operador, tuvo menor costo y mayor efectividad que la variante Kahan local, aplicada dentro de cada celda (Sección «Kahan local frente a compensación espacial»).

Bajo el operador de estrés, Kahan local no mejoró la exactitud de forma apreciable frente a la ruta sin compensar. La causa cualitativa es la misma en ambos regímenes: el residuo de cada celda solo lo relee ella misma, mientras que el operador de vecindario combina valores cuyos residuos entran sin corregir. El operador difusivo de la Fase 4 confirma ese balance fuera del régimen de estrés, con cifras concretas: la compensación espacial fue más rápida que Kahan local y, en la malla de 8192^2, presentó un error relativo unas 800 veces menor en FP16 y unas $6\,500$ veces menor en BF16 (Tabla 27). Ninguna medición de tráfico DRAM permite cuantificar la diferencia de ese recurso entre ambos esquemas. La eficacia de una técnica de compensación depende, por tanto, de la estructura del operador y no únicamente de su formulación aritmética.

4.5.5 Atribución energética y alcance de la medición

La energía reportada corresponde al consumo del acelerador medido mediante NVML, sin incluir la componente del sistema anfitrión. Esa delimitación acota el alcance de las conclusiones energéticas: acelerar el kernel en la GPU no reduce proporcionalmente el consumo total del sistema si el anfitrión permanece activo durante toda la ejecución, de modo que una evaluación restringida al acelerador sobrestima el beneficio energético de la precisión mixta. La limitación aplica directamente al análisis multiobjetivo de la Sección «Análisis multiobjetivo tiempo–energía–error», construido exclusivamente sobre energía de GPU, cuyas conclusiones deben interpretarse dentro de ese alcance.

4.5.6 Implicaciones para la selección de precisión

Los resultados indican que la selección de precisión debe considerar conjuntamente las características arquitectónicas del kernel y la tolerancia numérica de la aplicación. En kernels de alta intensidad aritmética, las rutas tensoriales pueden ofrecer ganancias importantes de rendimiento; en kernels limitados por memoria, el beneficio depende más del movimiento de datos y de la eficiencia de la implementación.

Elegir solo la configuración de menor error favorece precisiones mayores, mientras que priorizar únicamente la alternativa más rápida puede introducir una degradación numérica excesiva. El análisis de Pareto permite representar este compromiso entre tiempo, energía y error sin imponer una ponderación universal, siempre que el error se evalúe en un horizonte común y el costo en ventanas fiables. En Stencil 2D, además, FP64 en GPU igualó en tiempo y energía a la mejor ruta tensorial con compensación. La Sección «Directrices de selección de precisión» traduce estas implicaciones en una regla de decisión.

4.5.7 Directrices de selección de precisión

Esta sección traduce los resultados en una regla de decisión por kernel, tolerancia y prioridad, cerrando el Objetivo Específico 4. Su validez se limita al alcance medido: GPU A100-PCIE-40GB, tamaños evaluados, energía de GPU y los dos operadores estudiados (estrés y difusivo con $\alpha = 3/16$). No se incluyen formatos no evaluados.

4.5.7.1 Ejes. La tolerancia ε corresponde al error relativo L_2 admisible frente a FP64 en el horizonte requerido por la aplicación, con cuatro bandas: laxa ($\varepsilon \geq 10^{-2}$), moderada ($10^{-3} \leq \varepsilon < 10^{-2}$), estricta ($10^{-5} \leq \varepsilon < 10^{-3}$) y muy estricta ($\varepsilon < 10^{-5}$). Una configuración se recomienda en una banda solo si su error cumple la tolerancia más exigente de esa banda; por ejemplo, en la banda moderada, su error debe ser menor que 10^{-3}. Las bandas se fusionan cuando la recomendación coincide; en Stencil 2D, la estricta se divide en 5×10^{-5} porque GPU FP32 alcanza $4{,}7 \times 10^{-5}$.

La prioridad usa un único eje de costo, pues tiempo y energía de GPU por iteración fueron casi colineales en los tres kernels (Sección «Análisis multiobjetivo tiempo–energía–error»; Figuras 16, 17 y 18). Con *prioridad costo* se elige la configuración admisible más barata; con *prior-*

idad exactitud, la de menor error entre las que superan en velocidad a GPU FP64, o GPU FP64 si ninguna lo hace.

4.5.7.2 Procedimiento.

- **Clasificar el operador** según su factor de amplificación λ. Si $\lambda \leq 1$ (Stencil difusivo con $\alpha = 3/16$; GEMM con $N = 1024$ o 4096), no hay horizonte de desbordamiento y se pasa a la elección de la banda de tolerancia.

- **Comprobar el horizonte** si $\lambda > 1$:

$$n_{\max}(f) \approx \frac{\log_2(M_f/s_f)}{\log_2 \lambda}, \tag{12}$$

  donde M_f es el máximo representable y s_f la amplitud inicial del modo dominante. El formato es elegible si el horizonte requerido no supera $0{,}9\, n_{\max}(f)$, margen que cubre el error del modelo ($< 7\,\%$; Sección «Drift y horizonte de estabilidad en Stencil 2D»).

  En Stencil 2D ($\lambda \approx 2$), el modelo predice $27{,}0$, $136{,}0$, $144{,}6$ y $1\,040{,}6$ iteraciones para FP16, BF16, FP32 y FP64, respectivamente, frente a 29, 142, 142 y $1\,044$–$1\,045$ medidas (Tabla 14). Para GEMM ($\lambda = \sqrt{2}$) y Convolución 2D ($\lambda = 2$), FP16 alcanza aproximadamente 32 y 16 iteraciones, coherente con la pérdida de finitud observada en la Fase 3. Más allá del horizonte de FP64, ningún formato evaluado es válido.

- **Elegir la banda de tolerancia** según el error admisible.

- **Consultar la celda correspondiente** de la Tabla 29.

Nota sobre el modelo. En la Tabla 14 se usa $s_f = \max(A, u_f)$, con $A \approx 1{,}0 \times 10^{-5}$ y $u_f = 2^{-11}$ en FP16 y 2^{-8} en BF16. Este término diferencia BF16 de FP32 pese a compartir rango de exponente. Como ambos horizontes medidos son 142, el modelo sobrestima ligeramente BF16; el margen de la comprobación del horizonte cubre esta diferencia.

Tabla 29

Directrices de selección de precisión por kernel, tolerancia y prioridad. Rendimiento frente a GPU FP64 salvo indicación; error relativo L_2 frente a FP64.

Tolerancia	Prioridad costo	Prioridad exactitud	Evidencia
GEMM, operación aislada ($N = 12\,288, 32\,768$) o encadenada con $\lambda = 1$ ($N = 1024, 4096$)			
$\varepsilon \geq 10^{-3}$ (laxa, moderada)	**cuBLAS TC FP16**: 15,27× frente a FP64 y 15,52× frente a cuBLAS FP32; error 2,3–3,1 × 10^{-4}. Encadenada: FP16 sin compensación, 3,1 × 10^{-4}, 2,4× en $N = 1024$.	**FP16 + comp. local**: 1,85 × 10^{-4} encadenada, ≈1,2× en $N = 1024$ (la compensación duplica el tiempo).	Tablas 9, 10; Secc. «Error numérico frente al tamaño del problema», «Rendimiento sostenido en función del tamaño»
$\varepsilon < 10^{-3}$ (estricta, muy estricta)	**GPU FP64** (cuBLAS, DMMA). cuBLAS FP32 da 2–5 × 10^{-6}, pero no acelera (0,98×).	**GPU FP64**.	Tablas 9, 10
GEMM, encadenada con $\lambda = \sqrt{2}$ ($N = 2048, 8192$) y más de ∼29 iteraciones (FP16 no elegible)			
$\varepsilon \geq 10^{-2}$ (laxa)	**BF16 sin comp.**: 2,5 × 10^{-3}, 2,59× ($N = 8192$, 80 it.).	**BF16 + comp. local**: 1,48 × 10^{-3}, ≈1,3× ($N = 8192$).	Secc. «Error numérico frente al tamaño del problema», «Estabilidad numérica y aparición de valores no finitos», «Rendimiento sostenido en función del tamaño»
$10^{-3} \leq \varepsilon < 10^{-2}$ (moderada)	**BF16** sin comp. si $\varepsilon \geq 2{,}5 \times 10^{-3}$; **BF16 comp.** si $\varepsilon \geq 1{,}5 \times 10^{-3}$; si no, **GPU FP64**.	Como en costo.	Ídem
$\varepsilon < 10^{-3}$	**GPU FP64**.	**GPU FP64**.	Ídem

Tolerancia	Prioridad costo	Prioridad exactitud	Evidencia
Convolución 2D, operación aislada ($C = K = 1024$, $H = W = 256$)			
$\varepsilon \geq 10^{-3}$ (laxa, moderada)	**cuDNN TC FP16**: 28,07× frente a FP64 y 15,30× frente a cuDNN FP32 con `IMPLICIT_GEMM` (6,83× con su algoritmo por defecto); error $5{,}7 \times 10^{-4}$.	**cuDNN FP32 clásico**: 2×10^{-6}, 1,83×.	Tabla 11
$\varepsilon < 10^{-3}$	**cuDNN FP32 clásico** (1,83×) si $\varepsilon \geq 2 \times 10^{-6}$; si no, **GPU FP64**.	Como en costo.	Tabla 11
Convolución 2D, encadenada con $\lambda = 2$ ($HW = 64$–512, 80 it.; FP16 no elegible tras $\sim$14 it.)			
$\varepsilon \geq 10^{-2}$ (laxa)	**BF16 sin comp.**: $8{,}5$–$8{,}7 \times 10^{-3}$, 1,47× ($HW = 512$).	**GPU FP64**: las rutas compensadas son más lentas que FP64 en todos los tamaños.	Secc. «Error numérico frente al tamaño del problema», «Rendimiento sostenido en función del tamaño»
$\varepsilon < 10^{-2}$	**GPU FP64** (BF16 compensado, $1{,}66 \times 10^{-3}$, es más lento que FP64). FP32 encadenado: sin evidencia en este trabajo.	**GPU FP64**.	Ídem
Stencil 2D, operador difusivo ($\lambda \leq 1$), malla 8192^2, 1 500 it.			
$\varepsilon \geq 5 \times 10^{-5}$ (laxa, moderada, estricta alta)	**GPU FP32** (CUDA escalar): $4{,}7 \times 10^{-5}$, 1,61× (0,515 frente a 0,831 ms/it.).	**WMMA + comp. espacial**, $K = 0$: $3{,}1 \times 10^{-5}$, 1,06×.	Tabla 28
$10^{-5} \leq \varepsilon < 5 \times 10^{-5}$	**Comp. espacial** $K = 0$ (1,06×) si $\varepsilon \geq 3{,}1 \times 10^{-5}$; si no, **GPU FP64**.	**GPU FP64**.	Tabla 28
$\varepsilon < 10^{-5}$ (muy estricta)	**GPU FP64** ($1{,}3 \times 10^{-13}$). El anclaje $K = 1$ llega a $2{,}6 \times 10^{-8}$, pero cuesta 3,9 veces más.	**GPU FP64**.	Tabla 28

Tolerancia	Prioridad costo	Prioridad exactitud	Evidencia
Stencil 2D, operador de estrés ($\lambda \approx 1{,}99$), mallas 4096^2–16384^2			
Todas	**GPU FP64** hasta $\sim$940 it. ($0{,}9 \times 1\,044$). Las rutas WMMA compensadas son más lentas que GPU FP64 ($0{,}60$–$0{,}67\times$ frente a $0{,}79$–$0{,}81\times$ la tasa de GPU FP32). GPU FP32 (horizonte 142): error bajo este operador sin evidencia en este trabajo.	**GPU FP64**.	Tabla 14; Secc. «Drift y horizonte de estabilidad en Stencil 2D», «Rendimiento sostenido en función del tamaño»

Nota. En Stencil 2D difusivo, la ruta sin compensación ($0{,}29$) y Kahan local ($2{,}5 \times 10^{-2}$) no son admisibles en ninguna banda. En 4096^2 a $4\,000$ iteraciones, con la solución ya decaída, solo GPU FP64 conserva exactitud (Sección «Análisis multiobjetivo tiempo–energía–error»). Las cifras encadenadas de GEMM y Convolución 2D proceden de la Fase 3 y las de operación aislada, de la Fase 2.

4.5.7.3 Aplicación a dos casos. Un usuario de GEMM con tolerancia 10^{-3} está en la banda moderada. Si ejecuta la operación aislada o un encadenamiento que no amplifica el estado, la ruta indicada es cuBLAS con Tensor Cores en FP16, con un error de $2{,}3$–$3{,}1 \times 10^{-4}$ y una aceleración de $15{,}27\times$ frente a FP64. Si el encadenamiento amplifica por $\sqrt{2}$ y supera unas 29 iteraciones, FP16 deja de ser elegible y los errores de BF16 ($2{,}5 \times 10^{-3}$ sin compensar y $1{,}48 \times 10^{-3}$ compensado) exceden la tolerancia, de modo que la ruta indicada es GPU FP64. Un usuario de Stencil 2D con tolerancia 10^{-5} debe usar GPU FP64 en ambos operadores. Con el operador difusivo, ninguna ruta de 32 o 16 bits alcanza ese error en 8192^2: la compensación espacial con $K = 0$ queda en $3{,}1 \times 10^{-5}$ y GPU FP32 en $4{,}7 \times 10^{-5}$, y solo el anclaje con $K = 1$ lo logra, a $3{,}9$ veces el costo de GPU FP64. Con el operador de estrés, GPU FP64 es además más rápida que las rutas WMMA compensadas, y es válida mientras la aplicación no supere unas 940 iteraciones.

4.6 Síntesis de resultados

En conjunto, los experimentos muestran que la computación en precisión mixta puede producir mejoras importantes de rendimiento, particularmente en kernels con elevada intensidad arit-

mética y alta reutilización de datos. Sin embargo, estas ganancias no son uniformes entre aplicaciones y deben interpretarse junto con el comportamiento del subsistema de memoria y la calidad numérica del resultado.

El análisis de Stencil 2D mostró además que el error de los esquemas iterativos debe analizarse como una trayectoria y no únicamente mediante su valor final. La compensación puede reducir esa desviación y el anclaje periódico en FP64 también lo hace cuando se aplica en cada iteración; al espaciarlo, el beneficio numérico disminuye o desaparece. Ambas estrategias introducen costos de tiempo y energía cuya magnitud depende del kernel y de su configuración.

Finalmente, la integración de las tres dimensiones mostró que ninguna alternativa minimiza simultáneamente tiempo, energía y error, y que la selección debe realizarse según las necesidades numéricas y computacionales de la aplicación. Además, construir un frente de tres objetivos exige que el error de todas las alternativas se evalúe en un mismo horizonte y que el costo se mida en ventanas fiables, aunque ambos procedan de pasadas distintas.

5 Conclusiones

El uso de precisión mixta y Tensor Cores no produce un beneficio uniforme entre kernels HPC: su efectividad depende del patrón computacional, del aprovechamiento del subsistema de memoria y de la sensibilidad numérica del problema. La selección de precisión no puede realizarse, por tanto, buscando únicamente el mayor rendimiento.

Implementación y verificación. Se implementaron y evaluaron GEMM, Convolución 2D y Stencil 2D en cuatro formatos, con bibliotecas optimizadas y kernels CUDA personalizados. NVIDIA Nsight Compute permitió comprobar la utilización efectiva de las unidades tensoriales en el kernel WMMA propio y en la ruta FP64 de cuBLAS, verificación necesaria porque operar con FP16 o BF16 no garantiza que el cálculo se ejecute sobre Tensor Cores; en las rutas tensoriales de cuBLAS y cuDNN ese uso se infirió del modo de invocación. El perfilado mostró además que la referencia FP64 de cuBLAS ya ejecuta el producto completo sobre Tensor Cores (instrucciones DMMA), de modo que las aceleraciones de GEMM frente a FP64 comparan dos rutas tensoriales y no una tensorial frente a una escalar. La ruta CUTLASS prevista en el primer objetivo específico

quedó como opción verificada funcionalmente en otra GPU, sin mediciones en la campaña reportada.

Rendimiento. En GEMM, la ruta FP16 de cuBLAS alcanzó $160{,}7$ TFLOP/s y $15{,}27\times$ frente a FP64 ($N = 12\,288$), y $16{,}50\times$ frente a FP32 clásico ($N = 32\,768$); en Convolución 2D, cuDNN alcanzó $28{,}07\times$ frente a FP64 y $15{,}30\times$ frente a FP32 con `IMPLICIT_GEMM` ($6{,}83\times$ frente al algoritmo FP32 por defecto de cuDNN). La brecha frente a los kernels WMMA propios (superior a un factor de cuatro, con error idéntico en GEMM y similar en Convolución 2D) muestra que activar el hardware tensorial es condición necesaria pero no suficiente. En Stencil 2D no hubo ganancia con la implementación WMMA evaluada: su línea base ya operaba entre el 55 y el $80\,\%$ del ancho de banda pico de HBM2, y las rutas WMMA con compensación espacial sostuvieron entre $0{,}60\times$ y $0{,}67\times$ la tasa de GPU FP32.

Comportamiento numérico. En un esquema iterativo el error no debe evaluarse solo al final, y su acumulación depende del operador más que del formato: en GEMM permaneció estacionaria durante las 80 iteraciones, en Convolución 2D creció un factor de $2{,}9$ en ese mismo intervalo y en Stencil 2D creció de forma sostenida. La diferencia práctica entre FP16 y BF16 bajo operadores amplificantes es de rango dinámico y no de resolución: en Stencil 2D, FP16 dejó de producir valores finitos en la iteración 29 y BF16 igualó el horizonte de FP32 (142 iteraciones) pese a su mantisa más corta, y un modelo de agotamiento del exponente reproduce esos horizontes con un error inferior al $7\,\%$. En las rutas con compensación espacial, mientras ambos formatos permanecieron finitos su error difirió en menos del $0{,}1\,\%$, porque la precisión propagada la fija la acumulación en FP32; en GEMM y Convolución 2D sin compensación, en cambio, el error de BF16 fue unas ocho veces mayor que el de FP16.

Corrección del error frente a control del rango. Son problemas distintos, y ninguna de las estrategias evaluadas resolvió el segundo: ambas actúan sobre la resolución del estado, no sobre su rango. La compensación local redujo el error por factores de $1{,}68$ en GEMM y de $5{,}1$–$5{,}2$ en Convolución 2D, a cambio de duplicar el tiempo y sin restaurar la finitud en ninguna configuración que la hubiese perdido; en Convolución 2D ese costo dejó a las rutas de 16 bits por debajo de

GPU FP64. En Stencil 2D la eficacia dependió de la estructura del esquema: la compensación espacial fue más rápida que Kahan local y, con el operador difusivo en 8192^2, su error fue unas 800 veces menor en FP16 y unas $6\,500$ en BF16.

El anclaje periódico en FP64 reduce el error propagado entre uno y dos órdenes de magnitud con $K = 1$ (valor que en Stencil 2D corresponde al piso de instrumentación del campo de lectura, no a la exactitud interna del mecanismo), pero no desplaza el horizonte de desbordamiento. Con $K > 1$ el costo se mantiene (entre $1{,}6$ y $1{,}7$ veces con $K = 8$ o $K = 32$ bajo el operador de estrés) y la reducción de error desaparece bajo ambos operadores, porque el paso en FP64 evita el redondeo de la iteración anclada pero no el acumulado en las demás. Su costo sigue el orden de intensidad aritmética de los kernels (duplica el tiempo en Stencil 2D, cuesta entre un 9 y un $44\,\%$ en GEMM y no incrementa el tiempo en Convolución 2D al tamaño evaluado) y elimina la ventaja temporal de FP16 sobre BF16 en los tres. Con $K = 1$ la ruta anclada evalúa todas las iteraciones en FP64 y almacena 10 bytes por celda frente a 8 en FP64, y en Stencil 2D con el operador difusivo cuesta $3{,}9$ veces el tiempo de GPU FP64. Tal como se definió en este trabajo, el anclaje no produjo ninguna configuración con mejor compromiso que ejecutar directamente en FP64.

Energía. La energía se midió solo en la GPU, mediante NVML: el acceso a los contadores RAPL de la CPU anfitriona, previsto en el segundo objetivo específico, estuvo restringido en la plataforma, de modo que las cifras sirven para comparar configuraciones entre sí y no como valores absolutos. Dentro de ese alcance, la configuración de menor EDP fue siempre una ruta sin compensación ni anclaje: en los cuatro tamaños de GEMM y de Convolución 2D y en las dos mallas de Stencil 2D con el operador difusivo, donde la compensación espacial tuvo un EDP por iteración entre $4{,}5$ y $4{,}6$ veces mayor y Kahan local entre $5{,}2$ y $6{,}3$ veces mayor. El EDP no incorpora el error, por lo que ese orden describe el costo de cada ruta y no su utilidad.

Análisis multiobjetivo. Se construyó en los tres kernels. Como conclusión metodológica, la resolución del sensor obliga a separar las pasadas numéricas de las energéticas, de modo que el análisis solo es válido si el error de todas las alternativas se evalúa en un horizonte común y el tiempo y la energía se toman como tasas por iteración de ventanas fiables. En los tres kernels tiempo

y energía resultaron casi colineales, por lo que el compromiso efectivo se reduce a rendimiento frente a exactitud. En GEMM y Convolución 2D las rutas sin compensación ocupan el extremo barato del frente y las compensadas el caro. En Stencil 2D, la referencia GPU FP64 igualó en tiempo y energía a la mejor ruta WMMA con compensación espacial, con un error ocho órdenes de magnitud menor, y dominó a toda configuración anclada, y a Kahan local salvo por un empate técnico en energía en 4096^2; solo las rutas sin anclaje conservaron una ventaja de costo, y la más barata, sin compensación, a cambio de un error que la hace inadmisible en todas las bandas de tolerancia.

Directrices de selección. El cuarto objetivo específico se cierra con una regla de decisión por kernel, tolerancia y prioridad (Sección «Directrices de selección de precisión»). En GEMM y Convolución 2D, las rutas FP16 de biblioteca son la opción más barata para tolerancias de 10^{-3} o mayores mientras el horizonte no agote el rango de FP16; para tolerancias más estrictas conviene GPU FP64 en GEMM, donde FP32 no acelera, y FP32 clásico en la convolución aislada. En Stencil 2D la elección es, según la tolerancia, GPU FP32 o GPU FP64; la única ruta de 16 bits que resulta la opción admisible más barata es la compensación espacial sin anclaje, en el estrecho intervalo de tolerancias entre $3{,}1$ y $4{,}7 \times 10^{-5}$, con un ahorro del $5\,\%$ frente a GPU FP64. En esquemas iterativos con operadores amplificantes, la elegibilidad de cada formato la decide primero su horizonte de desbordamiento y solo después su error.

Respuesta a la pregunta de investigación. La respuesta depende del kernel. En kernels de alta intensidad aritmética con implementaciones maduras, la precisión mixta ofrece aceleraciones de entre 15 y 28 veces frente a FP64 en GPU, con errores del orden de 10^{-4} en la operación aislada. En Stencil 2D, cuya línea base está limitada por el ancho de banda de memoria, la implementación WMMA evaluada no obtuvo ganancia temporal y el compromiso se desplaza al eje numérico: bajo operadores amplificantes, el rango dinámico impone un horizonte de validez que ninguna de las estrategias evaluadas logró extender; bajo operadores estables, la doble precisión en GPU tiene un costo similar al de la mejor ruta tensorial compensada y es mucho más exacta, de modo que la precisión reducida solo aporta ahorro si se acepta un error considerablemente mayor.

Limitaciones y trabajo futuro. Estas conclusiones corresponden a una única GPU NVIDIA A100, a energía medida solo en el acelerador y a las implementaciones evaluadas. Las campañas de producción tienen una medición por configuración, por lo que los frentes de Pareto se basan en dominancia determinista; las ocho réplicas de la campaña de variabilidad, con coeficientes de variación inferiores al 1 %, respaldan las diferencias de tiempo amplias y no las sutiles. El frente de Stencil 2D se obtuvo con un operador difusivo y una condición inicial monomodo. De estas restricciones se derivan las siguientes líneas de continuación: perfilar con Nsight Compute las rutas FP16 de cuBLAS y cuDNN y la ruta WMMA de Stencil 2D, para confirmar el uso de instrucciones HMMA y establecer si el límite de esta última es de ancho de banda, latencia u ocupación; evaluar una formulación de Stencil 2D que comparta el halo entre vecinos, como ConvStencil (Chen et al., 2024); estudiar mecanismos de control del rango, como el reescalado periódico del estado, puesto que ni la compensación ni el anclaje extendieron el horizonte de desbordamiento; incorporar la energía del anfitrión, réplicas en las campañas de producción y condiciones iniciales multimodo; y repetir la evaluación en otras generaciones de GPU y con selección dinámica de precisión. Una versión preliminar de estos resultados se presentó en la sesión de pósteres de CARLA 2026, donde recibió el premio a Mejor Póster.

Referencias Bibliográficas

Abdelfattah, A., Anzt, H., Boman, E. G., Carson, E., Cojean, T., Dongarra, J., Fox, A., Gates, M., Higham, N. J., Li, X. S., Loe, J., Luszczek, P., Pranesh, S., Rajamanickam, S., Ribizel, T., Smith, B. F., Swirydowicz, K., Thomas, S., Tomov, S., … Yang, U. M. (2021). A survey of numerical linear algebra methods utilizing mixed-precision arithmetic [Revisión de métodos de álgebra lineal numérica que utilizan aritmética de precisión mixta]. *The International Journal of High Performance Computing Applications*, *35*(4), 344–369. https://doi.org/10.1177/10943420211003313

Andersch, M., Palmer, G., Krashinsky, R., Stam, N., Mehta, V., Brito, G., y Ramaswamy, S. (2022, March). *NVIDIA Hopper architecture in-depth* [Análisis detallado de la arquitectura NVIDIA Hopper] [Consultado en julio de 2025]. NVIDIA Technical Blog. https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

Atoofian, E. (2023). PTTS: Power-aware tensor cores using two-sided sparsity [PTTS: Tensor Cores con gestión del consumo energético mediante esparsidad bilateral]. *Journal of Parallel and Distributed Computing*, *173*, 70–82. https://doi.org/10.1016/j.jpdc.2022.11.004

Baboulin, M., Donfack, S., Kaya, O., Mary, T., y Robeyns, M. (2024). Mixed precision randomized low-rank approximation with GPU Tensor Cores [Aproximación aleatorizada de bajo rango en precisión mixta mediante Tensor Cores de GPU]. *Euro-Par 2024: Parallel Processing*, 31–44. https://doi.org/10.1007/978-3-031-69583-4_3

Basso, P. M., dos Santos, F. F., y Rech, P. (2020). Impact of Tensor Cores and mixed precision on the reliability of matrix multiplication in GPUs [Impacto de los Tensor Cores y la precisión mixta en la fiabilidad de la multiplicación de matrices en GPU]. *IEEE Transactions on Nuclear Science*, *67*(7), 1560–1565. https://doi.org/10.1109/TNS.2020.2977583

Blanchard, P., Higham, N. J., Lopez, F., Mary, T., y Pranesh, S. (2020). Mixed precision block fused multiply-add: Error analysis and application to GPU Tensor Cores [Multiplicación y suma fusionadas por bloques en precisión mixta: Análisis de error y aplicación a los

Tensor Cores de GPU]. *SIAM Journal on Scientific Computing*, *42*(3), C124–C141. https://doi.org/10.1137/19M1289546

Chen, Y., de Oliveira Castro, P., Bientinesi, P., Jansson, N., y Iakymchuk, R. (2026). Enabling mixed-precision in spectral element codes [Habilitación de la precisión mixta en códigos de elementos espectrales]. *Future Generation Computer Systems*, *174*, 107990. https://doi.org/10.1016/j.future.2025.107990

Chen, Y., Li, K., Wang, Y., Bai, D., Wang, L., Ma, L., Yuan, L., Zhang, Y., Cao, T., y Yang, M. (2024). ConvStencil: Transform stencil computation to matrix multiplication on Tensor Cores [ConvStencil: Transformación del cómputo de Stencil en multiplicación de matrices sobre Tensor Cores]. *Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming (PPoPP '24)*, 333–347. https://doi.org/10.1145/3627535.3638476

Croci, M., y Giles, M. B. (2022). Effects of round-to-nearest and stochastic rounding in the numerical solution of the heat equation in low precision [Efectos del redondeo al más cercano y del redondeo estocástico en la solución numérica de la ecuación del calor en baja precisión]. *IMA Journal of Numerical Analysis*, *43*(3), 1358–1390. https://doi.org/10.1093/imanum/drac012

Cui, C. (2024). Acceleration of tensor-product operations with Tensor Cores [Aceleración de operaciones de productos tensoriales mediante Tensor Cores]. *ACM Transactions on Parallel Computing*, *11*(4), 15:1–15:24. https://doi.org/10.1145/3695466

Dongarra, J., Gunnels, J., Bayraktar, H., Haidar, A., y Ernst, D. (2024). *Hardware trends impacting floating-point computations in scientific applications* [Tendencias de hardware que afectan a los cálculos de punto flotante en aplicaciones científicas] [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2411.12090

El Arar, E.-M., Filip, S.-I., Mary, T., y Riccietti, E. (2025). *Mixed precision accumulation for neural network inference guided by componentwise forward error analysis* [Acumulación

en precisión mixta para inferencia de redes neuronales guiada por el análisis del error hacia adelante por componentes] [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2503.15568

Fasi, M., Higham, N. J., Lopez, F., Mary, T., y Mikaitis, M. (2023). Matrix multiplication in multiword arithmetic: Error analysis and application to GPU Tensor Cores [Multiplicación de matrices con aritmética de múltiples palabras: Análisis de error y aplicación a los Tensor Cores de GPU]. *SIAM Journal on Scientific Computing*, *45*(1), C1–C19. https://doi.org/10.1137/21M1465032

Fasi, M., Higham, N. J., Mikaitis, M., y Pranesh, S. (2021). Numerical behavior of NVIDIA Tensor Cores [Comportamiento numérico de los Tensor Cores de NVIDIA]. *PeerJ Computer Science*, *7*, e330. https://doi.org/10.7717/peerj-cs.330

Gedik, G., Schöne, R., y Iakymchuk, R. (2026). *Mixed-precision for energy efficient computations* [Precisión mixta para cálculos energéticamente eficientes] [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2606.27949

Godoy, W. F., Hernandez, O., Kent, P. R. C., Patrou, M., Asifuzzaman, K., Miniskar, N. R., Valero-Lara, P., Vetter, J. S., Sinclair, M. D., Lowe-Power, J., y Bruce, B. R. (2025). Characterizing GPU energy usage in exascale-ready portable science applications [Caracterización del consumo energético de GPU en aplicaciones científicas portables preparadas para la exaescala]. *High Performance Computing. ISC High Performance 2025 International Workshops*, *16091*, 177–190. https://doi.org/10.1007/978-3-032-07612-0_14

Habib, A., Finkelstein, J., y Niklasson, A. M. N. (2024). Efficient mixed-precision matrix factorization of the inverse overlap matrix in electronic structure calculations with AI-hardware and GPUs [Factorización eficiente de la matriz inversa de solapamiento en cálculos de estructura electrónica mediante precisión mixta, hardware de IA y GPU]. *Journal of Chemical Theory and Computation*, *20*(16), 7102–7112. https://doi.org/10.1021/acs.jctc.4c00584

Hafezan, M., y Atoofian, E. (2023). Mixed-precision architecture for GPU Tensor Cores [Arquitectura de precisión mixta para Tensor Cores de GPU]. *Proceedings of the 23rd IEEE International Conference on Scalable Computing and Communications (ScalCom)*, 1–8. https://doi.org/10.1109/SWC57546.2023.10448789

Haidar, A., Bayraktar, H., Tomov, S., Dongarra, J., y Higham, N. J. (2020). Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems [Refinamiento iterativo en precisión mixta mediante Tensor Cores en GPU para acelerar la solución de sistemas lineales]. *Proceedings of the Royal Society A*, *476*(2243), 20200110. https://doi.org/10.1098/rspa.2020.0110

Haidar, A., Tomov, S., Dongarra, J., y Higham, N. J. (2018). Harnessing GPU Tensor Cores for fast FP16 arithmetic to speed up mixed-precision iterative refinement solvers [Aprovechamiento de los Tensor Cores de GPU para acelerar solucionadores de refinamiento iterativo en precisión mixta mediante aritmética FP16 rápida]. *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, 603–613. https://doi.org/10.1109/SC.2018.00050

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms* [Exactitud y estabilidad de los algoritmos numéricos] (2nd ed.). SIAM. https://doi.org/10.1137/1.9780898718027

Higham, N. J., y Mary, T. (2022). Mixed precision algorithms in numerical linear algebra [Algoritmos de precisión mixta en álgebra lineal numérica]. *Acta Numerica*, *31*, 347–414. https://doi.org/10.1017/S0962492922000022

Kashi, A., Koukpaizan, N., Lu, H., Matheson, M., Oral, S., y Wang, F. (2025). Scaling the memory wall using mixed-precision — HPG-MxP on an exascale machine [Superación del muro de memoria mediante precisión mixta: HPG-MxP en una máquina de exaescala]. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '25)*, 281–297. https://doi.org/10.1145/3712285.3759877

Kashi, A., Lu, H., Brewer, W., Rogers, D., Matheson, M., Shankar, M., y Wang, F. (2024). *Mixed-precision numerics in scientific applications: Survey and perspectives* [Cálculo en precisión mixta en aplicaciones científicas: Revisión y perspectivas] [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2412.19322

Khattak, F. A., y Mikaitis, M. (2025). *Accurate models of NVIDIA Tensor Cores* [Modelos precisos de los Tensor Cores de NVIDIA] [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2512.07004

Lin, Z., Sun, A., Zhang, X., y Lu, Y. (2024). MixPert: Optimizing mixed-precision floating-point emulation on GPU integer tensor cores [MixPert: Optimización de la emulación de punto flotante en precisión mixta sobre Tensor Cores de enteros en GPU]. *Proceedings of the 25th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES '24)*, 34–45. https://doi.org/10.1145/3652032.3657567

Liu, X.-Y., Hong, H., Zhang, Z., Tong, W., Kossaifi, J., Wang, X., y Walid, A. (2024). High-performance tensor-train primitives using GPU Tensor Cores [Operaciones fundamentales de trenes de tensores de alto rendimiento mediante Tensor Cores de GPU]. *IEEE Transactions on Computers*, *73*(11), 2634–2648. https://doi.org/10.1109/TC.2024.3441831

Lu, Y., Zeng, L., Wang, T., Fu, X., Li, W., Cheng, H., Yang, D., Jin, Z., Casas, M., y Liu, W. (2024). AmgT: Algebraic multigrid solver on Tensor Cores [AmgT: Solucionador de multigrid algebraico sobre Tensor Cores]. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '24)*, 823–838. https://doi.org/10.1109/SC41406.2024.00058

Markidis, S., Chien, S. W. D., Laure, E., Peng, I. B., y Vetter, J. S. (2018). NVIDIA Tensor Core programmability, performance & precision [Programabilidad, rendimiento y precisión de los Tensor Cores de NVIDIA]. *IEEE International Parallel and Distributed Processing*

Symposium Workshops (IPDPSW), 522–531. https://doi.org/10.1109/IPDPSW.2018.00091

Mukunoki, D., y Ogita, T. (2020). Performance and energy consumption of accurate and mixed-precision linear algebra kernels on GPUs [Rendimiento y consumo energético de kernels de álgebra lineal precisos y de precisión mixta en GPU]. *Journal of Computational and Applied Mathematics*, *372*, 112701. https://doi.org/10.1016/j.cam.2019.112701

Netti, A., Peng, Y., Omland, P., Paulitsch, M., Parra, J., Espinosa, G., Agarwal, U., Chan, A., y Pattabiraman, K. (2023). Mixed precision support in HPC applications: What about reliability? [Compatibilidad con precisión mixta en aplicaciones HPC: ¿qué ocurre con la fiabilidad?]. *Journal of Parallel and Distributed Computing*, *181*, 104746. https://doi.org/10.1016/j.jpdc.2023.104746

Suarez, E., Amaya, J., Frank, M., Freyermuth, O., Girone, M., Kostrzewa, B., y Pfalzner, S. (2025). Energy efficiency trends in HPC: What high-energy and astrophysicists need to know [Tendencias de eficiencia energética en HPC: Lo que necesitan saber los físicos de altas energías y los astrofísicos]. *Frontiers in Physics*, *13*, 1542474. https://doi.org/10.3389/fphy.2025.1542474

van der Vlugt, S., Oostrum, L., Schoonderbeek, G., van Werkhoven, B., Veenboer, B., Doekemeijer, K., y Romein, J. W. (2025). PowerSensor3: A fast and accurate open source power measurement tool [PowerSensor3: Herramienta de código abierto rápida y precisa para medir potencia]. *Proceedings of the 2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 213–226. https://doi.org/10.1109/ISPASS64960.2025.00028

Yang, D., Zhao, Y., Niu, Y., Jia, W., Shao, E., Liu, W., Tan, G., y Jin, Z. (2024). Mille-feuille: A tile-grained mixed-precision single-kernel conjugate gradient solver on GPUs [Mille-feuille: Solucionador de gradiente conjugado en un solo kernel y precisión mixta con granularidad por mosaicos para GPU]. *Proceedings of the International Conference*

for High Performance Computing, Networking, Storage and Analysis (SC '24), 916–931. https://doi.org/10.1109/SC41406.2024.00064

Yang, Z., Adámek, K., y Armour, W. (2024). Accurate and convenient energy measurements for GPUs: A detailed study of NVIDIA GPU's built-in power sensor [Mediciones de energía precisas y prácticas para GPU: Estudio detallado del sensor de potencia integrado de NVIDIA GPU]. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '24)*, 307–323. https://doi.org/10.1109/SC41406.2024.00028

Zhang, L., Huang, J., Di, S., Matsuoka, S., y Wahib, M. (2025). Can Tensor Cores benefit memory-bound kernels? (no!) [¿Pueden los Tensor Cores beneficiar a los kernels limitados por memoria? (¡no!)]. *Proceedings of the 17th Workshop on General Purpose Processing Using GPU (GPGPU '25)*, 28–34. https://doi.org/10.1145/3725798.3725803

Zhang, S., Liang, J., Fu, Y., Hua, R., Yu, J., y Li, Q. (2026). Accelerating dilated Winograd convolution with fused GPU kernel using tensor cores [Aceleración de la convolución Winograd dilatada mediante un kernel de GPU fusionado que utiliza Tensor Cores]. *Journal of King Saud University – Computer and Information Sciences*, *38*(5), 225. https://doi.org/10.1007/s44443-025-00452-1

Zhang, Z., Zhang, P., Xu, Z., Yan, B., y Wang, Q. (2024). Im2col-Winograd: An efficient and flexible fused-Winograd convolution for NHWC format on GPUs [Im2col-Winograd: Convolución Winograd fusionada, eficiente y flexible para el formato NHWC en GPU]. *Proceedings of the 53rd International Conference on Parallel Processing (ICPP '24)*, 1072–1081. https://doi.org/10.1145/3673038.3673039

Apéndices

Apéndice A. Participación y reconocimiento en CARLA 2026

La Latin American High Performance Computing Conference (CARLA) es el encuentro anual de la comunidad latinoamericana de computación de alto rendimiento, y reúne a grupos de investigación, centros de supercómputo, universidades e industria de la región en torno a la computación de alto rendimiento, la inteligencia artificial a escala HPC, sus aplicaciones y la formación en el área. Su sesión de pósteres constituye uno de los espacios de divulgación de trabajos en curso ante esa comunidad. Los pósteres allí expuestos son evaluados por el Poster Committee de la conferencia.

El presente trabajo se presentó en la edición CARLA 2026, realizada entre el 21 y el 25 de septiembre de 2026 en el Centro Cultural de la Universidad Nacional de Córdoba (Obispo Trejo 314, Córdoba, Argentina). El póster, titulado *Experimental Evaluation of the Numerical and Energy Impact of Computing in Mixed-Precision Using Tensor Cores on NVIDIA GPUs for HPC Kernels*, fue firmado por Karen Dayana Mateus Gómez y William Andrés Urrutia Torres (Universidad Industrial de Santander, UIS), Luis A. Torres N. (UIS; SC3UIS/CAGE) y Carlos J. Barrios H. (UIS; SC3UIS/CAGE; LIG/INRIA-Grenoble; INSA Lyon, INRIA, CITI, UR3720).

Entre los pósteres expuestos en la sesión, el Poster Committee de CARLA 2026 otorgó a este trabajo el premio a Mejor Póster de la edición.

Apéndice B. Resumen aceptado

Se transcribe el resumen tal como fue sometido y aceptado por la convocatoria de pósteres, sin correcciones posteriores. Su contenido corresponde al estado del trabajo en el momento de la postulación y no necesariamente a los resultados finales consignados en el apartado «Resultados y discusión».

Abstract. In high-performance computing (HPC), the default use of double precision (FP64) has become a bottleneck regarding execution time and power consumption. Modern NVIDIA GPUs incorporate Tensor Cores, hardware units that accelerate matrix operations in reduced formats such as FP16 and BF16; however, their adoption in scientific software remains limited by numerical stability constraints. The reduction in precision introduces rounding errors that, in iterative schemes, can accumulate and compromise the validity of the results. Few studies comprehensively analyze the intersection of execution time, energy consumption, and numerical error. This poster empirically evaluates this trade-off across three representative kernels, HPC-GEMM, 2D convolution, and 2D stencil, implemented in FP64, FP32, FP16, and BF16 utilizing cuBLAS, cuDNN, and custom WMMA kernels. Performance and energy metrics are quantified using Nsight Compute, NVML, and RAPL, while numerical degradation is assessed via the L_2 and L_∞ norms relative to the FP64 baseline. For the iterative stencil, the cumulative drift per iteration is characterized, both with and without Kahan-compensated summation, and an "overflow horizon" is modeled to predict the iteration at which the reduced-precision field diverges. Preliminary verification confirms the effective activation of Tensor Cores (HMMA instructions in SASS) and demonstrates that BF16 tolerates more iterations than FP16 before overflow, as it shares the FP32 exponent range. The results are integrated using statistical analysis and a Pareto front in the time-energy-error space to derive precision selection guidelines based on the application's error tolerance.

Keywords: mixed-precision computing; Tensor Cores; NVIDIA GPUs; HPC kernels; numerical error propagation; overflow horizon; Kahan sum; Energy-Delay Product; Pareto front.

Apéndice C. Declaración sobre el uso de herramientas de inteligencia artificial

Durante la preparación editorial de esta versión se utilizaron Claude Code, de Anthropic, para apoyar la adaptación del proyecto LaTeX a las normas APA 7 y a la guía de la Biblioteca UIS, y Codex, de OpenAI, para apoyar correcciones de redacción, ortografía y estilo, la revisión de citas y referencias y los ajustes del código LaTeX y su compilación. Las sugerencias de estas herramientas se incorporaron según la revisión de los autores. Su uso en esta etapa se limitó a la presentación del documento y no incluyó la generación de datos experimentales ni la modificación de mediciones, ecuaciones, figuras, resultados o conclusiones científicas. Los autores asumen la responsabilidad por el contenido final.