

Estudio de la eficiencia en la escalabilidad de GPU's para el entrenamiento de Inteligencia Artificial

David Cortes, Carlos Juiz, Belen Bermejo¹

Resumen— El entrenamiento de modelos de aprendizaje profundo a gran escala se ha convertido en un desafío clave para la comunidad científica y la industria. Si bien el uso masivo de GPUs puede acelerar considerablemente los tiempos de entrenamiento, este enfoque conlleva un impacto negativo en la eficiencia. En este artículo, presentamos un análisis detallado de los tiempos presentados por MLPerf Training v4.1 en cuatro cargas de trabajo: BERT, Llama2 LoRA, RetinaNet y Stable Diffusion, evidenciando que existen configuraciones que optimizan la relación entre el rendimiento, uso de GPUs y eficiencia. Los resultados señalan un “punto de equilibrio” que permite disminuir tiempos de entrenamiento manteniendo maximizando la eficiencia.

Palabras clave— GPUs; Rendimiento; MLPerf; Deep Learning; Benchmarking; Eficiencia; Aceleración

I. INTRODUCCIÓN

EL entrenamiento de modelos de aprendizaje profundo de gran escala se ha convertido en una prioridad estratégica tanto en entornos académicos como industriales [1]. Con la creciente complejidad de tareas como la comprensión del lenguaje natural, la detección de objetos y la generación de contenido, el uso de infraestructuras con un gran número de GPUs se ha consolidado como la vía para reducir significativamente el tiempo de entrenamiento. Sin embargo, la ampliación masiva de recursos no siempre se traduce en una mejora proporcional de la eficiencia, puesto que la sobrecarga de comunicación y la sincronización entre nodos suelen afectar negativamente el rendimiento de cada acelerador.

En este contexto, MLPerf Training [2] ofrece un marco de facto de evaluación estandarizada para comparar diversas configuraciones y arquitecturas de hardware. A través de cargas de trabajo representativas, como BERT, Llama2 LoRA, Retinanet y Stable Diffusion, diferentes empresas han reportado el tiempo que tardan sus máquinas en entrenar estos modelos aplicando el benchmark en sus sistemas. Estos tiempos resultantes se han reportado en un repositorio de acceso libre [3]. Este estudio presenta un análisis de estos resultados, que evidencian dos tendencias contrapuestas: la disminución del tiempo total de entrenamiento al aumentar el número de GPUs y la consiguiente reducción de la eficiencia por acelerador. En este artículo se aborda el desafío de conciliar estas dos realidades, sugiriendo la existencia de configuraciones equilibradas que combinan tiempos de entrenamiento competitivos con un aprovechamiento

to óptimo de los recursos, el cual puede repercutir posiblemente en minimizar los costes operativos y el impacto energético. Este enfoque cobra importancia no solo para centros de investigación y laboratorios con grandes presupuestos, sino también para aquellas organizaciones que requieren maximizar su sostenibilidad.

Para abordar este objetivo, en este documento se exponen resultados cuantitativos que ilustran cómo la eficiencia varía en función del número de aceleradores, y se discuten los factores que determinan el rendimiento en diferentes arquitecturas de hardware. Con ello, se pretende contribuir a la toma de decisiones informadas al momento de planificar y dimensionar infraestructuras de cómputo para el entrenamiento de modelos de Deep Learning.

Este artículo se estructura de la siguiente manera: en la Sección II, se presentan los conceptos fundamentales y métricas clave de rendimiento y escalabilidad; la Sección III revisa los trabajos relacionados más relevantes; en la Sección IV, se describe la metodología empleada para analizar y comparar los sistemas; la Sección V, expone los resultados experimentales; en la Sección VII, se amplía la discusión; finalmente en VI se ofrecen las conclusiones y posibles líneas de investigación futura.

II. CONCEPTOS PREVIOS

A. Rendimiento y escalabilidad

La computación de alto rendimiento (HPC) consiste en el uso de supercomputadoras de gran capacidad para resolver problemas que requieren un alto volumen de operaciones [1]. En el caso del aprendizaje profundo, HPC se orienta a distribuir el entrenamiento de modelos neuronales complejos entre múltiples aceleradores[1]. En los últimos años, las *Graphics Processing Units*(GPUs) han adquirido un papel esencial en este ámbito, gracias a su capacidad de ejecutar miles de hilos en paralelo y de procesar operaciones matriciales de forma muy eficiente. Estas propiedades son idóneas para las convoluciones y demás cálculos necesarios en redes neuronales, por lo que la infraestructura de HPC suele incluir grandes conjuntos de GPUs interconectadas mediante redes de alta velocidad[4]. En este contexto, este artículo se centra en el tiempo de entrenamiento, es decir, el tiempo necesario para ajustar los parámetros de un modelo utilizando un conjunto de datos determinado[2]. En este marco, definimos el Speedup, en el contexto a la comparación de sistemas, como la re-

¹Dpto. de Ciencias Matemáticas e Informática, Universidad de las Islas Baleares, e-mail: {d.cortes, cjuiz, bbermejo}@uib.es

lación en el aumento de velocidad, que mide cuánto más rápido es un sistema en comparación con otro [5] al que denominaremos de referencia. Puede definirse como la relación entre los tiempos de ejecución, y, en nuestro contexto de formación, se calcula como la relación entre el tiempo de referencia y el tiempo nuevo como se muestra en la **Ecuación 1**.

$$Speedup = \frac{\text{Tiempo Referencia}}{\text{Tiempo Nuevo}} \quad (1)$$

Fórmula 1

La Eficiencia se define como la relación entre la velocidad alcanzada y el número de aceleradores o procesadores utilizados [5]. Se expresa como se muestra en la **Fórmula 2**. Esta métrica indica la eficacia con la que se utilizan los recursos disponibles. La eficiencia alcanza un valor de 1 en casos ideales, en los que la mejora del rendimiento es proporcional al número de aceleradores empleados [5].

$$\text{Eficiencia} = \frac{Speedup}{\text{Número de aceleradores}} \quad (2)$$

Fórmula 2

$$\text{Eficiencia}' (E') = \frac{\frac{\text{Tiempo Referencia}}{\text{Tiempo Nuevo}}}{\frac{\text{Número de aceleradores}}{\text{Aceleradoras de referencia}}} \quad (3)$$

Fórmula 3

La escalabilidad es la capacidad de un sistema para adaptarse a un aumento de la carga de trabajo manteniendo o mejorando su rendimiento[5]. Este concepto incluye la planificación de la capacidad, que permite la asignación flexible de recursos en función de las demandas variables, como se observa en las aplicaciones distribuidas o en los sistemas bajo demanda [5].

B. Clasificación

La agrupación, es una técnica fundamental en el aprendizaje no supervisado, cuyo objetivo principal es organizar y clasificar puntos de datos de objetos o *clusters* basándose en sus similitudes intrínsecas, sin necesidad de etiquetas predefinidas[6]. Esta técnica se aplica ampliamente en campos como la segmentación de mercados, el análisis de datos genómicos y el procesamiento de imágenes, debido a su capacidad para descubrir patrones ocultos y estructuras subyacentes en los datos[6]. Existen varios métodos de agrupación, cada uno con características y aplicaciones distintas. En este estudio, utilizamos el métodos, *K-means*, el cual es uno de los algoritmos de agrupación más populares y sencillos[7]. Divide los datos en «k» grupos predefinidos, asignando cada punto al *cluster* con el centroide más cercano y minimizando iterativamente la distancia entre los puntos y su centroide asignado[7].

C. Deep Learning

En el campo específico del Aprendizaje Automático, los sistemas aprenden patrones directamente a partir de datos, sin ser programados explícitamente para cada tarea[6]. Una de las ramas más potentes de esta área es el Aprendizaje Profundo ó *Deep Learning* (DL), que se caracteriza por el uso de redes neuronales artificiales profundas con múltiples capas[8]. Estas redes han demostrado un rendimiento sobresaliente en retos como la clasificación de imágenes, la traducción automática o la generación de texto, logrando resultados que compiten e incluso superan, en muchos casos, el desempeño humano[9]. Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano, consistentes en neuronas artificiales organizadas en capas. Cada neurona recibe un conjunto de entradas, realiza una operación matemática y transmite su salida a otras neuronas [10]. Estas redes pueden aprender relaciones complejas dentro de los datos ajustando las neuronas mediante algoritmos de entrenamiento[10]. A diferencia de los métodos tradicionales de aprendizaje automático, en los que la extracción intervención manual [8], DL permite la extracción automática de características de alto nivel directamente de los datos entrada[9].

Dentro del aprendizaje profundo, se distinguen varias subáreas con objetivos y metodologías particulares, como lo son el Procesamiento de Lenguaje Natural (NLP) que se encarga de que las máquinas puedan entender, interpretar y generar lenguaje humano [11]. Modelos como BERT y Llama2-LoRA se sitúan en esta categoría, aunque con matices diferenciales. BERT se especializa en tareas de comprensión del lenguaje [11] y Llama2-LoRA, por su parte, apunta a la generación y manipulación de texto, y representa un modelo de lenguaje de gran escala que utiliza la técnica de Low-Rank Adaptation (LoRA), que es una técnica para entrenar eficientemente modelos grandes ajustando solo una pequeña parte de los parámetros mediante matrices de bajo rango [12].

Otro campo es el de Visión por Computador, que comprende el reconocimiento y análisis automatizados de información contenida en imágenes o secuencias de vídeo. Donde el modelo RetinaNet, orientado a la detección de objetos[13], es un ejemplo destacado en esta rama. Se caracteriza por su arquitectura de red neuronal que combina un backbone, típicamente ResNet, el cual es un modelo para clasificación de imágenes en el campo de DL [14], [15]. Este se usa para la extracción de características con una subred de clasificación y regresión de bounding boxes, que son los marcos alrededor del objeto [13].

Así mismo los Modelos Generativos enfocados en la creación de contenido sintético, tales como imágenes o secuencias de texto. Como Stable Diffusion, incluido en este documento, es un modelo generativo capaz de crear imágenes de alta calidad a partir de ruido aleatorio o de descripciones textuales[16].

III. TRABAJOS RELACIONADOS

El campo de la evaluación comparativa de hardware y software para el entrenamiento de modelos de DL ha sido objeto de numerosos estudios en los últimos años. Al hacer una segmentación y enfocarnos en los estudios que reflejan el usos comparativo en la aceleración y haciendo énfasis en el hardware, podemos encontrar trabajos como, [17] en el que proponen una aproximación para la gestión eficiente de sistemas con el uso de GPU al ejecutar tareas tanto de entrenamiento como de inferencia. Su método, denominado Cost Efficient Deep Learning Job Allocation, integra técnicas de modelado agnóstico a la arquitectura y un planteamiento consciente con el costo eléctrico. El objetivo es asignar tareas heterogéneas a los nodos GPU de forma que se minimice el coste energético sin degradar el rendimiento. [17], se centra en la planificación de trabajos y la integración de métodos de right-sizing. Nuestro trabajo, en cambio, realiza un análisis comparativo de configuraciones ya definidas en MLPerf, enfocándose en la disminución de la eficiencia por GPU al crecer el número de aceleradores. [18] ofrece un amplio panorama de los aceleradores hardware para la implementación de redes neuronales profundas, comparando arquitecturas como GPU, FPGA, ASIC y CGRA. Se discuten factores de diseño, potencia, throughput, y se repasan avances en el entrenamiento y la inferencia de DNNs en distintos entornos embebidos y de alto rendimiento. En [19] los autores presentan una revisión sistemática enfocada en la optimización y aceleración de grandes modelos de lenguaje Large Language Models, (LLMs). Se discute la evolución de las bibliotecas, frameworks y estrategias de escalado para reducir los costes de entrenamiento y el tiempo de convergencia sin menoscabar el rendimiento. Si bien también se evalúa el impacto del escalado y la complejidad de modelos masivos, su énfasis es en LLMs y sus optimizaciones específicas. En la misma línea [20] estudian la escalabilidad de modelos LLMs en configuraciones con miles de GPUs, analizando cuidadosamente estrategias de paralelismo. En nuestro caso, se incluye un modelo de lenguaje a gran escala Llama2 LoRA dentro de una comparación más amplia de cargas (BERT, Retinanet, Stable Diffusion), analizando principalmente la eficiencia al variar el número de GPUs. En [21], se analizan diferentes plataformas y se revelan cuellos de botella en la en la arquitectura de TPU. Por otro lado [22] examina los principios de benchmarking en el contexto de dispositivos de ML y marcos de software para el entrenamiento de redes profundas. Se destacan varios enfoques de evaluación y se introducen métricas para comparar rendimiento, escalabilidad y facilidad de integración. También se hace referencia a MLPerf como organismo de referencia para medir rendimiento. En [23] cuantifican cómo distintos modelos de DL escalan en GPUs con recursos de energía limitados. Ajustan modelos tipo ley de potencia para describir la forma en que el time-to-train varía según la disponibilidad de cómputo y restricciones energéti-

cas. Sin embargo su enfoque es orientado a modelar el consumo energético. En [24] introducen métodos algorítmicos específicos diseñados para HPC para el entrenamiento distribuido de modelos de ML que mejoran la escalabilidad. A lo largo de los años han aparecido algunas técnicas que intentan reducir el tiempo de entrenamiento, la complejidad y el consumo energético manteniendo la escalabilidad [25], a diferencia de nuestro estudio que compara los resultados de los sistemas con un benchmark específico.

IV. METODOLOGÍA

En este trabajo se analizan, de manera comparativa, los tiempos de entrenamiento reportados en MLPerf Training v4.1 [3], [26] para diferentes máquinas y distintos algoritmos de aprendizaje profundo. No se llevaron a cabo experimentos directos de entrenamiento, sino que se procesaron los resultados oficiales publicados para las cargas de trabajo BERT, Llama2 LoRA, RetinaNet y Stable Diffusion. Cada sistema evaluado presenta un número variable de GPUs, lo que permite estudiar cómo se comporta el rendimiento de las configuraciones más pequeñas frente a las de escala masiva. Cada carga de trabajo presenta un conjunto de sistemas diferentes, como se muestra en la Tabla I, con solo 11 máquinas presentes en las 4 cargas.

Tabla I: Máquinas por cargas de trabajo y sistema de referencia para cada carga

Carga	Cantidad de Sistemas	Cantidad GPUs referencia	Tiempo referencia
BERT	28	2	366.18
Llama2 LoRA	29	4	61.87
RetinaNet	26	2	172.83
Stable Diffusion	25	4	60.52

Este trabajo se llevó a cabo utilizando la siguiente metodología. En primer lugar, se seleccionó una máquina de referencia para cada carga de trabajo caracterizada por su reducido número de GPUs y su mayor latencia en tiempo entrenamiento en comparación con el resto de equipos. A partir de esta elección, se procedió a calcular las métricas del *speedup* y de *E'* utilizando los tiempos publicados en [3], resultados públicos de MLPerf, asegurando la consistencia en la forma de medición de cada sistema y el cálculo de las métricas utilizadas. En el cálculo del *speedup* se utilizaron los valores de la máquina de referencia para cada carga, garantizando que se pudiera cuantificar la ganancia en velocidad de entrenamiento siguiendo la Ecuación 1. Posteriormente, se derivó al cálculo de la *E'*, entendida como la fracción de mejora total atribuible a cada acelerador adicional y que contempla la cantidad de aceleradores de la máquina de referencia, siguiendo la Ecuación 3. Para cada conjunto de sistemas se procedió a realizar un proceso de agrupamiento utilizando el algoritmo de K-means, con la finalidad de identificar patrones de comportamiento. Finalmente, se compararon los resultados en tablas y gráficos para identificar tendencias de escalabilidad

y, sobre todo, para ubicar un rango de configuraciones óptimo en el que se obtiene un equilibrio entre la reducción del tiempo de entrenamiento y el aprovechamiento efectivo de cada GPU denotado por E’.

V. RESULTADOS EXPERIMENTALES

En esta sección se describen los hallazgos principales del estudio tras comparar la eficiencia en el escalado de GPU’s en el entrenamiento de cuatro modelos diferentes de IA; BERT, Llama2, Retinanet y Stable Diffusion. Los datos muestran, de forma general, que si bien el aumento de aceleradores reduce el tiempo total de entrenamiento, la eficiencia por GPU tiende a disminuir, mostrando configuración con mayor rendimiento.

A. BERT

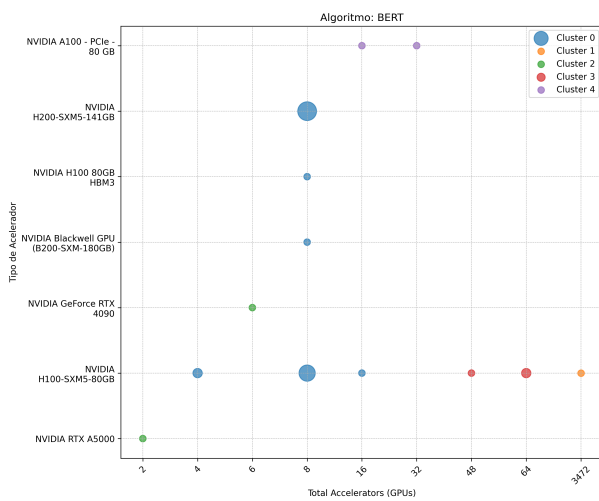


Fig. 1: Máquinas por tipo y cantidad de aceleradoras para la carga BERT

En la Fig. 1 se evidencia una amplia gama de configuraciones para la carga BERT, desde sistemas con tan solo 2 GPUs hasta soluciones de escalado masivo con más de 3000 aceleradores. Esto demuestra la gran heterogeneidad de sistemas utilizados para entrenar el modelo, que abarcan desde GPUs de la serie RTX hasta las más recientes familias H100 y Blackwell. El tamaño de los marcadores indica cuántas máquinas comparten cada configuración, y los colores reflejan la agrupación por clústeres. El tipo de acelerador NVIDIA H100-SXM5-80GB presenta múltiples máquinas con mayor escalado.

Una vez calculada la eficiencia para el modelo BERT, se aprecia que las configuraciones con pocas GPUs, en torno a 8 aceleradores, exhiben los valores más altos de eficiencia, debido a que la sobrecarga de comunicación aún no resulta significativa. A partir de 16 GPUs, la eficiencia se penaliza de manera más pronunciada, tal como se ilustra en la Fig. 2. No obstante, incluso en estas configuraciones más reducidas, se obtiene una mejora sustancial en el *time-to-train* respecto a la máquina base, lo que sugiere que este modelo escala razonablemente bien si la prioridad es reducir el tiempo de entrenamiento. Por otro

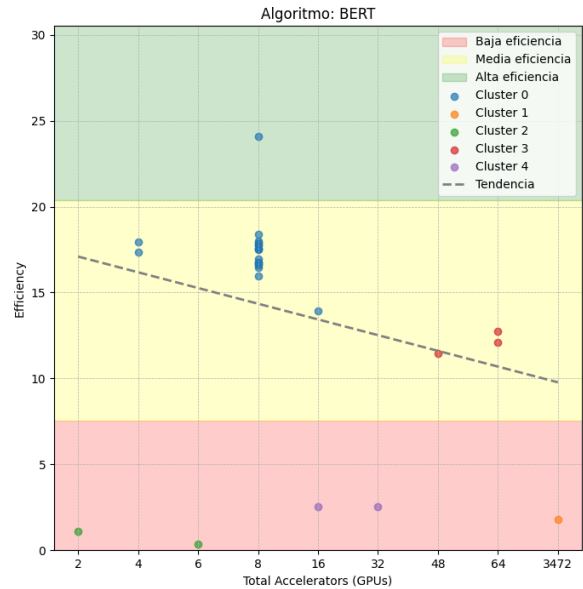


Fig. 2: Comparación de eficiencia en el algoritmo BERT

lado, se observa un hecho llamativo: en una máquina con 3472 aceleradores, la eficiencia es comparable a la de una configuración con solo 2 GPUs, lo que refleja los rendimientos decrecientes provocados posiblemente por las cargas de comunicación al llegar a escalas masivas.

B. Llama2-LoRA

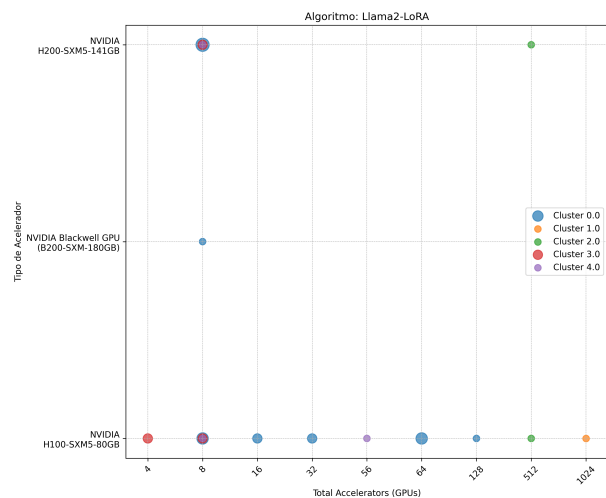


Fig. 3: Máquinas por tipo y cantidad de aceleradoras para la carga Llama2-LoRA

Para la carga de Llama2 LoRA, el número de GPUs va aproximadamente de 4 hasta 1024, como se puede ver en la Fig. 3, reflejando una amplia variedad en la escala de los sistemas analizados. La mayoría de configuraciones se agrupan en torno a un rango bajo-medio de aceleradores, 4 a 64, aunque se aprecia un punto con 1024 GPUs en la parte derecha de la gráfica. La presencia de aceleradores de la NVIDIA H100-SXM5-80GB domina buena parte de los sistemas, si bien también aparecen GPU Blackwell y

sistemas de la serie H200.

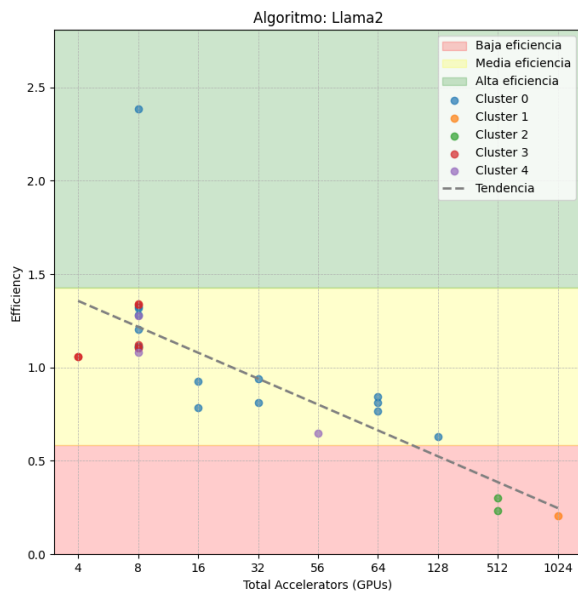


Fig. 4: Comparación de eficiencia en el algoritmo Llama2 LoRA

El comportamiento de la eficiencia en el modelo de Llama2 LoRA que exige una alta demanda de memoria y comunicaciones, se logra evidenciar una caída de eficiencia acelerada al escalar en número de GPUs. Nuevamente, en configuraciones modestas, 4 a 8 aceleradores, se aprecia una eficiencia superior a la unidad, lo que significa un buen aprovechamiento de cada GPU adicional. Sin embargo, a partir de 16 aceleradores, la curva de eficiencia decrece visiblemente, como se aprecia en la Fig. 4. Aun así, al comparar el time-to-train, los sistemas con mayor escalado, 1024 GPUs, concluyen el entrenamiento con una rapidez significativamente mayor, algo valioso si la prioridad es reducir drásticamente los tiempos de convergencia. De manera similar a BERT, se identificó un rango óptimo con eficiencias medias, en el que se combinan tiempos razonables con un uso equilibrado de recursos.

C. RetinaNet

En el caso de RetinaNet, la distribución de los sistemas va desde configuraciones con dos sistemas de 4 GPUs hasta otras que superan las 2000 unidades. De nuevo, se distinguen varias familias de aceleradores, H100, Blackwell, H200, lo cual revela enfoques muy diversos de escalabilidad. Los clústeres reflejan no solo la variación en la cantidad de GPUs, sino también las diferencias en los modelos de GPU empleados. Al igual que en BERT y Llama2-LoRA, el acelerador más presente con diferentes configuraciones es el NVIDIA H100-SXM5-80GB, como se ve en la Fig. 5.

Retinanet, un modelo de detección de objetos, mostró un comportamiento de escalabilidad más lineal que Llama2 LoRA, posiblemente al requerir me-

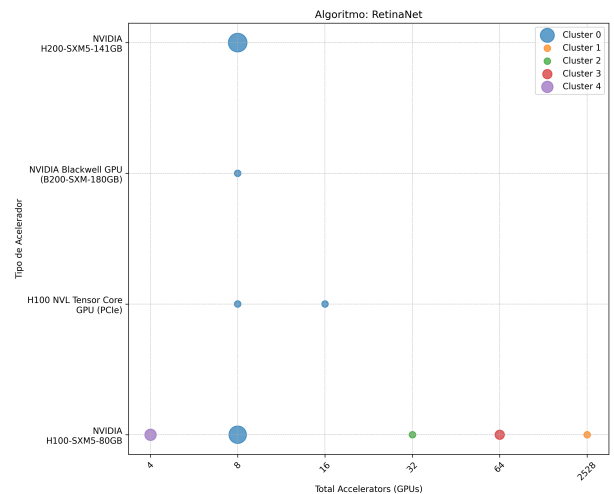


Fig. 5: Máquinas por tipo y cantidad de aceleradoras para la carga RetinaNet

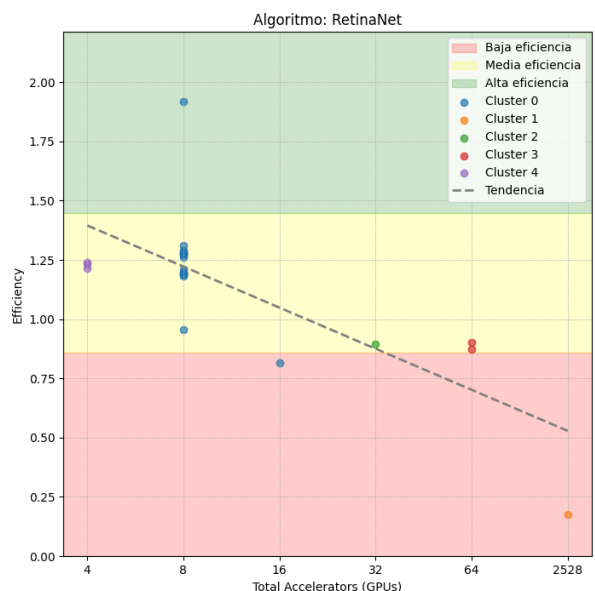


Fig. 6: Comparación de eficiencia en el algoritmo RetinaNet

nos sincronización de parámetros en cada iteración. No obstante, también se constató una reducción gradual en la eficiencia por GPU. Como se aprecia en la Fig. 6. En las máquinas de 8 GPUs, la eficiencia se mantiene en niveles medios o altos, indicando que cada GPU adicional sigue mejorando el rendimiento de forma tangible. Al sobrepasar este límite, la pendiente de la curva de eficiencia se hace más pronunciada, confirmando que los beneficios de escalar se vuelven cada vez menores. Colando en la zona de baja eficiencia a máquinas con 2528 aceleradores.

D. Stable Diffusion

En la Fig. 7, se observan configuraciones que van de 4 a 1024 GPUs. Además de las arquitecturas de NVIDIA, la carga de Stable Diffusion, cuenta con sistemas con aceleradores TPU-v5p, lo cual añade otra dimensión de diversidad al análisis, pues se trata de un tipo diferente de acelerador. El tamaño de los

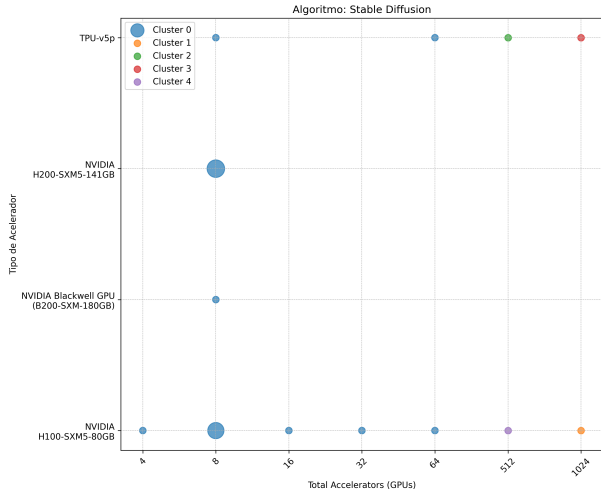


Fig. 7: Máquinas por tipo y cantidad de aceleradoras para la carga Stable Diffusion

puntos indica que múltiples máquinas comparten esa configuración. Nuevamente la NVIDIA H100-SXM5-80GB presenta una deversidad en escalado.

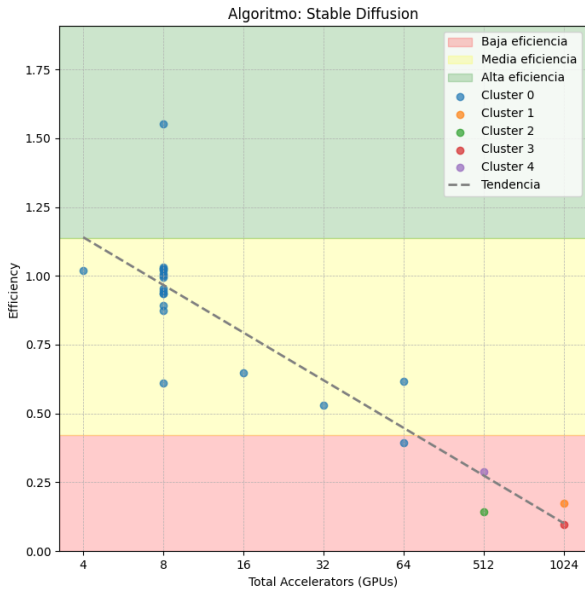


Fig. 8: Comparación de eficiencia en el algoritmo Stable Diffusion

Finalmente el caso de Stable Diffusion resulta especialmente exigente, pues la combinación de un autoencoder y una U-Net con múltiples pasos de denoising incrementa la comunicación entre GPUs. Sin embargo algunas configuraciones moderadas, de 4 a 8 GPU's presentan una eficiencia moderada y alta, posiblemente a que aprovecha la capacidad de paralelismo en el espacio latente y en la red convolucional. Sin embargo como se aprecia en la Fig. 8, conforme se incluyen más aceleradores, 16, 32 o incluso cientos, la eficiencia cae de manera notable, posiblemente por cuellos de botella en la comunicación de gradientes y en la sincronización de las distintas partes del modelo. Como en los demás algoritmos, se destaca

un punto intermedio que maximiza la relación entre rapidez y rendimiento por GPU.

VI. DISCUSIÓN

Una lectura clave que se deriva del hecho de que la comunicación y sincronización de gradientes en redes de gran escala penaliza de forma notable el rendimiento individual por acelerador. Esto puede verse con especial claridad en modelos que requieren un gran intercambio de datos, como Llama2 LoRA, que maneja grandes volúmenes de parámetros y, en consecuencia, experimenta una caída acelerada de la eficiencia. Establecer un balance adecuado entre la capacidad de cómputo y la complejidad de la comunicación se puede traducir en una mejor utilización de cada GPU, lo que cobra importancia para centros de computación con restricciones de presupuesto o de consumo energético. Adicional posiblemente la distribución de las cargas en el entrenamiento también incide en la eficiencia.

Estos hallazgos resaltan la necesidad de una planificación cuidadosa al dimensionar infraestructuras de cómputo. Más GPUs no siempre implican una optimización global, ya que la sobrecarga de comunicación puede neutralizar las ventajas del paralelismo. Así, la elección de la configuración “óptima” depende del objetivo principal —rapidez absoluta en el entrenamiento o maximización de la eficiencia por acelerador— y de los recursos disponibles. Sin embargo, es importante señalar que los datos utilizados provienen exclusivamente de los reportes oficiales de MLPerf Training v4.1, lo cual restringe el análisis a las configuraciones publicadas y podría representar solo una fracción del panorama real en entornos de producción.

En este sentido, el contar con un mayor número de sistemas con el mismo modelo y generación de GPU, topología de red y configuración de software, permitiría caracterizar de manera más robusta la relación entre el escalado y la eficiencia. La muestra disponible incluye equipos con distinto hardware y distintas generaciones de GPUs, lo que introduce diversidad en los resultados, pero dificulta la comparación estrictamente homogénea. Aun así, se logra describir y contrastar la ganancia de rendimiento por cada GPU adicional en la configuración establecida por MLPerf, sin profundizar en la incidencia de posibles optimizaciones de software o ajustes de hiperparámetros.

Otro aspecto relevante es que las pruebas están ceñidas a la configuración y versión del software especificados por MLPerf, sin variaciones de optimización o ajustes más finos que podrían alterar los resultados de desempeño y eficiencia. Este planteamiento, centrado principalmente en el rendimiento estandarizado, que se refiera a una visión en el que cada sistema se enfrenta a la misma prueba en las mismas condiciones. No obstante, entendemos que la seleccionan distintos optimizadores o implementación de paralelismo híbrido podría maximizar la eficiencia.

Por último, cabe destacar que la comparación no pretende evaluar el rendimiento de cada GPU de

manera individual, sino describir el comportamiento global del sistema a medida que crece el número de aceleradores. Las diferencias generacionales y en capacidad de procesamiento entre las GPUs analizadas constituyen otro factor que podría incidir en la eficiencia, pero no se ha abordado en profundidad en este estudio. Pese a ello, la muestra disponible permite visualizar con claridad cómo el escalado masivo no siempre se traduce en ganancias lineales y que existen configuraciones intermedias que logran un equilibrio más adecuado entre tiempo de entrenamiento y aprovechamiento de recursos.

VII. CONCLUSIONES

El análisis de los resultados de MLPerf Training v4.1 para diferentes máquinas y cargas de trabajo, BERT, Llama2 LoRA, Retinanet y Stable Diffusion, pone de manifiesto que en máquinas con miles de GPUs, el time-to-train es muy corto, el incremento en el número de GPUs reduce efectivamente los tiempos de entrenamiento, pero a costa de degradar la eficiencia por acelerador.

Por otro lado, aquellas configuraciones más modestas, con menos aceleradores, exhiben una eficiencia notablemente superior, si bien su tiempo global de entrenamiento aumenta. Este comportamiento apunta a la importancia de identificar puntos de equilibrio que optimicen el rendimiento sin incurrir en un sobredimensionamiento de recursos. Asimismo, se destaca que el comportamiento de la escalabilidad no es idéntico en todas las cargas, los modelos de Stable Diffusion y Llama2 LoRA, presentan una mayor caída de eficiencia con grandes volúmenes de GPUs.

En conjunto, estos hallazgos demuestran la necesidad de sopesar detenidamente tanto el time-to-train como la eficiencia por GPU al dimensionar infraestructuras de cómputo para el entrenamiento distribuido de modelos de Deep Learning. El estudio respalda la idea de que no siempre es ventajoso apuntar a configuraciones con miles de GPUs, especialmente cuando se prioriza la sostenibilidad, el control de costes o el aprovechamiento óptimo de cada acelerador. Como trabajo futuro, se profundizará en estrategias de paralelismo y optimizaciones de la comunicación para reducir la caída de eficiencia, así como incorporar mediciones de consumo energético y métricas de sostenibilidad que guíen la toma de decisiones de forma más holística.

REFERENCIAS

- [1] Sergei Kurgalin and Sergei Borzunov, *A practical approach to high-performance computing*, Springer International Publishing, 11 2019.
- [2] Peter Mattson, Christine Cheng, and Diamos et. al, “Mlperf training benchmark,” in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., 2020, vol. 2, pp. 336–349.
- [3] Peter Mattson, Christine Cheng, and Cody Coleman et. al, “Mlcommons - training — tableau public,” .
- [4] Jose González-Abad, Álvaro López García, and Valentin Y. Kozlov, “A container-based workflow for distributed training of deep learning algorithms in hpc clusters,” *Cluster Computing*, vol. 26, pp. 2815–2834, 10 2023.
- [5] Xavier Molero, Carlos Juiz, and Miguel Rodeño, *Evaluación y modelado del rendimiento de los sistemas informáticos*, Pearson Educación London, 1 edition, 2004.
- [6] Dongkuan Xu and Yingjie Tian, “A comprehensive survey of clustering algorithms,” *Annals of Data Science* 2015 2:2, vol. 2, pp. 165–193, 8 2015.
- [7] J MacQueen, “Some methods for classification and analysis of multivariate observations,” *books.google.com/J MacQueen Proceedings of the fifth Berkeley symposium on mathematical, 1967*•books.google.com, 1965.
- [8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2323, 1998, Deep Learning.
- [9] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton, “Deep learning,” *Nature* 2015 521:7553, vol. 521, pp. 436–444, 5 2015.
- [10] Sun-Chong Wang, “Artificial neural network,” *Interdisciplinary Computing in Java Programming*, pp. 81–100, 2003.
- [11] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 10 2018.
- [12] Simon Lermen, Charlie Rogers-Smith, and Jeffrey Lashish, “LoRA Fine-tuning Efficiently Undoes Safety Training in Llama 2-Chat 70B,” *arXiv*, 10 2023.
- [13] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar, “Focal Loss for Dense Object Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 8 2017.
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 12 2015, Micro ResNet.
- [15] Keiron O’Shea and Ryan Nash, “An introduction to convolutional neural networks,” *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, pp. 943–947, 11 2015.
- [16] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2022-June, pp. 10674–10685, 12 2021.
- [17] Dong Ki Kang, Ki Beom Lee, and Young Chon Kim, “Cost efficient gpu cluster management for training and inference of deep learning,” *Energies* 2022, Vol. 15, Page 474, vol. 15, pp. 474, 1 2022.
- [18] Pudi Dhillewarao, Srinivas Boppu, M. Sabarimalai Manikandan, and Linga Reddy Cenkeramaddi, “Efficient hardware architectures for accelerating deep neural networks: Survey,” *IEEE Access*, vol. 10, pp. 131788–131828, 2022.
- [19] Zhyar Rzgar K. Rostam, Sandor Szenasi, and Gabor Kertesz, “Achieving peak performance for large language models: A systematic review,” *IEEE Access*, vol. 12, pp. 96017–96050, 2024.
- [20] Jared Fernandez, Luca Wehrstedt, Leonid Shamis, Mostafa Elhoushi, Kalyan Saladi, Yonatan Bisk, Emma Strubell, and Jacob Kahn, “Hardware scaling trends and diminishing returns in large-scale distributed training,” *arXiv preprint arXiv:2411.13055*, 11 2024.
- [21] Yu Emma Wang, Gu-Yeon Wei, and David Brooks, “Benchmarking tpu, gpu, and cpu platforms for deep learning,” 7 2019.
- [22] Wei Dai and Daniel Berleant, “Benchmarking contemporary deep learning hardware and frameworks: A survey of qualitative metrics,” *Proceedings - 2019 IEEE 1st International Conference on Cognitive Machine Intelligence, CogMI 2019*, pp. 148–155, 12 2019.
- [23] Nathan C. Frey, Baolin Li, Joseph McDonald, Dan Zhao, Michael Jones, David Bestor, Devesh Tiwari, Vijay Gadepally, and Siddharth Samsi, “Benchmarking resource usage for efficient distributed deep learning,” *2022 IEEE High Performance Extreme Computing Conference, HPEC 2022*, 1 2022.
- [24] Yang You, Aydin Buluc, and James Demmel, “Scaling deep learning on gpu and knights landing clusters,” *In-*

ternational Conference for High Performance Computing, Networking, Storage and Analysis, SC, vol. 2017-November, 8 2017.

- [25] Luis Cruz Varona, “Paralelización del entrenamiento de redes neuronales en sistemas heterogéneos,” 7 2021.
- [26] Peter Mattson, Christine Cheng, and et. al, “Mlperf training benchmark,” in *Proceedings of Machine Learning and Systems*, I Dhillon, D Papailiopoulos, and V Sze, Eds., 2020, vol. 2, pp. 336–349.