Índice

1. Contexto del Proyecto

1.1. Nombre del Proyecto

micro-wgpu

1.2. Nombre de la institución o empresa

El proyecto se realizará en como un proyecto de investigación en el Instituto Tecnológico de Costa Rica.

1.3. Requerimientos de confidencialidad y propiedad intelectual

Es un proyecto de investigación que busca hacer todo de manera abierta y transparente. Por lo tanto nada debe ser confidencial y se debe de publicar todo el código e información generada públicamente.

El titular de la propiedad intelectual sería el autor, que publicará todo bajo una licencia abierta.

Una ventaja de que el proyecto se desarrolle de manera abierta es que podrá utilizarse cualquier software de código abierto, notablemente el API de gráficos a utilizar es software de código abierto.

1.4. Descripción del problema

Los desarrolladores e investigadores en áreas de procesamiento gráfico o computación con GPUs enfrentan dificultades para comprender y optimizar el rendimiento de estas en diferentes plataformas.

Esta situación genera incertidumbre sobre la capacidad de las aplicaciones para operar eficientemente en diversos entornos y complica la identificación de diferencias en rendimiento o posibles fallos en las distintas capas por las que se ejecutan estos programas, desde la implementación de APIs gráficos hasta el hardware, lo cual es crucial tanto para proyectos académicos como para la optimización de productos comerciales.

1.5. Objetivo general

Desarrollar un banco de microbenchmarks que permita evaluar y comparar el rendimiento de GPUs en diversas plataformas, con el fin de facilitar la identificación de diferencias en la implementación de APIs gráficos y optimizar aplicaciones gráficas, utilizando WebGPU en navegadores y Vulkan en interfaces de línea de comandos.

1.6. Objetivos específicos

1. Investigar las características y requisitos fundamentales de los microbenchmarks aplicados a GPUs para establecer un banco de estos que sea de beneficio para la comunidad académica y desarrolladora. Esto se logrará mediante la revisión de la literatura existente y la recopilación de datos relevantes.
2. Diseñar una arquitectura modular de pipeline gráfico y microbenchmarking, que facilite la creación de microbenchmarks y permita su ejecución en diferentes contextos (interfaces web y CLI).
3. Implementar las interfaces web y CLI que permitan la ejecución de los microbenchmarks, la recopilación de datos y la interacción con los usuarios, asegurando en el caso de la interfaz web la integración con la base de datos para el almacenamiento y consulta de resultados.
4. Orquestar la infraestructura necesaria para servir la página web, que incluye la configuración del servidor y la base de datos, garantizando la accesibilidad y el rendimiento adecuados para los usuarios finales.

1.7. Interesados

- **Desarrolladores e Ingenieros en GPU**: utilizan el proyecto para optimizar el rendimiento de aplicaciones mediante microbenchmarks específicos de WebGPU. Buscan herramientas precisas para identificar cuellos de botella y mejorar la eficiencia.
- **Investigadores Académicos**: usan los microbenchmarks para estudios y experimentos sobre rendimiento de GPUs, buscando datos precisos para validar teorías y publicar resultados.
- **Empresas de Tecnología y Desarrollo de Software**: implementan los microbenchmarks para mejorar el rendimiento y la experiencia de usuario en sus productos, identificando y corrigiendo problemas antes del lanzamiento.
- **Proveedores de Hardware**: analizan el rendimiento de sus GPUs en diferentes escenarios para ajustar y mejorar sus productos, basándose en los resultados de los microbenchmarks.
- **Usuarios Finales (Desarrolladores y Usuarios de Aplicaciones)**: se benefician indirectamente de las mejoras en el rendimiento de las aplicaciones, lo que impacta en la calidad de su experiencia.

Descripción del trabajo

2.1. Descripción de la solución

La solución al problema planteado consiste en desarrollar un framework integral para la ejecución y evaluación de microbenchmarks en GPUs, utilizando las API de WebGPU y Vulkan. Esta solución se desglosa en los siguientes componentes principales:

- **Framework de Pipeline Gráfico**: Se creará una biblioteca que ofrece un pipeline gráfico simplificado utilizando wgpu, diseñado para ejecutar funciones gráficas, como compute shaders, y medir su rendimiento. Esta biblioteca también incluirá utilidades para facilitar la creación y adición de microbenchmarks.
- **Biblioteca de Microbenchmarks**: Se desarrollará una segunda biblioteca que proporciona varios microbenchmarks predefinidos. Esta biblioteca estará compuesta por funciones que ejecutan los microbenchmarks y devuelven resultados estructurados, permitiendo evaluar el rendimiento de distintas operaciones gráficas.
- **Aplicación CLI**: Se implementará una aplicación binaria con una interfaz de línea de comandos (CLI) que actuará como un envoltorio ligero sobre la biblioteca de microbenchmarks. La CLI permitirá ejecutar los microbenchmarks y presentar los resultados de forma clara en la terminal.
- **Servidor Web**: Se establecerá un servidor que aloje una página web con una interfaz gráfica y scripts de WebAssembly (WASM) para ejecutar los microbenchmarks desde el navegador. El servidor gestionará la comunicación de los resultados de los microbenchmarks y ofrecerá funcionalidades para acceder a datos históricos y realizar descargas de datos en formato CSV.

2.2 Entregables y criterios de aceptación

A continuación se detallan los entregables y sus criterios de aceptación por objetivo específico.

Entregables Objetivo Específico 1

Identificador	Entregable	Criterio de aceptación
101	Documento de requisitos. Este documento detallará todas las funcionalidades del sistema, así como la composición específica del banco de microbenchmarks.	El documento de requisitos entregado detalla las cantidades y tipos de microbenchmarks que se realizarán e incluirá al menos 5 referencias.

Entregables Objetivo Específico 2

Identificador	Entregable	Criterio de aceptación
201	Documento de diseño. Este documento detallará las funcionalidades específicas de la biblioteca, su arquitectura y cómo se utilizará.	El documento de diseño incluye ejemplos claros de cómo registrar un benchmark.
202	Biblioteca documentada que tenga funciones para crear pipeline gráfico y registrar benchmarks.	Se puede utilizar la directiva `#![deny(missing_docs)]` en la biblioteca para evitar que compile si no está documentada. La funcionalidad de la biblioteca será aceptable si efectivamente se puede escribir el banco de microbenchmarks con ella.

Entregables Objetivo Específico 3

Identificador	Entregable	Criterio de aceptación
301	Programa de interfaz CLI.	La interfaz cumple con todas las características detalladas en el documento de requerimientos.
302	Programa de servidor que sirve la página para la interfaz web y se comunica con la base de datos.	La interfaz cumple con todas las características detalladas en el documento de requerimientos.

Entregables Objetivo Específico 4

Identificador	Entregable	Criterio de aceptación
401	Paquete de Nix del servidor. Esto asegurará poder levantar el servidor en cualquier entorno de manera simple.	Se puede levantar el servidor a través de la herramienta Nix.
402	Despliegue de la página web accesible desde internet con un URL público y estable.	En el URL definido se sirve la interfaz web y se verifica el funcionamiento con la base de datos detallado en el documento de requerimientos.

2.3. Actividades y propuesta de esfuerzo

Identificador	Actividad	Horas estimadas
100	Experimentar y familiarizar con wgpu.	10
200	Definir requisitos.	10
300	Crear setup inicial para pruebas con pipeline mínimo y una prueba de multiplicación matricial.	15
400	Agregar una prueba de ancho de banda de memoria (la de copias entre buffers) para ya tener una prueba de los 2 tipos principales que hay y tener una mejor idea de qué es necesario.	8
500	Plan de proyecto.	8
600	Diseñar API y arquitectura de biblioteca de framework para crear microbenchmarks.	5
601	Diseñar estructura de página web.	2
602	Diseñar arquitectura de servidor que sirve página web y comunica con base de datos.	4
603	Redactar el documento de diseño formal.	8
700	Implementar API y arquitectura de biblioteca, reescribiendo las pruebas existentes para adaptarse a la biblioteca ya definida.	12
701	Escribir microbenchmarks existentes como una biblioteca separada.	4
702	Implementar interfaz CLI como wrapper de la biblioteca de microbenchmarks.	4
800	Implementar servidor que sirve página web con configuración local.	8
801	Agregar detalles de producción al servidor (como configuración de DB real)	2
900	Crear el archivo de nix para empaquetar el servidor.	3
901	Poner el servidor en el servicio de hosting.	4
1000	Implementar microbenchmark de convolución.	5
1001	Implementar microbenchmark de reducción.	5
1100	Implementar microbenchmark de scan.	5
1101	Implementar microbenchmark de accesos de memoria secuenciales.	5
1200	Implementar microbenchmark de accesos de memoria desordenados.	5

Identificador	Actividad	Horas estimadas
1201	Implementar microbenchmark de ancho de banda de copiar de buffer->textura.	5
1300	Implementar microbenchmark de ancho de banda de copiar entre texturas.	5
1301	Agregar benchmarks faltantes a herramienta CLI.	2
1500	Agregar benchmarks faltantes a servidor web, incluyendo alteraciones necesarias en la base de datos y cambios en la interfaz web.	10
1600	Mejorar la documentación del repositorio para guíar desarrolladores en el uso de la herramienta CLI, ejecutar el servidor de manera local o producción y cómo agregar microbenchmarks.	4
1601	Redactar informe final.	10

El total de horas estimadas para el proyecto es de 168 horas, lo cual implica un promedio de poco de más de 10 horas por semana dedicadas al proyecto.

2.4. Análisis de riesgos

Riesgos personales

- Falta de familiaridad con tecnologías.
  - Descripción: Al iniciar el proyecto el desarrollador no tiene mucha experiencia en APIs y pipelines gráficos y no ha utilizado `wgpu` antes. Se dedicará tiempo del proyecto a familiarizarse con las tecnologías pero el riesgo es que no sea suficiente o que la falta de familiaridad cause problemas después.
  - Probabilidad: 0.3
  - Impacto en esfuerzo: Podría requerir hasta 40 horas adicionales de investigación y aprendizaje.

Resigos de herramientas

- Incompatibilidades o fallos en el sistema de desarrollo (APIs gráficos, drivers)
  - Descripción: Podrían haber fallos con drivers o con los APIs gráficos de manera que no se puedan ejecutar bien en el entorno de desarrollo.
  - Probabilidad: 0.05
  - Impacto en esfuerzo: Si es un problema pequeño por ejemplo con drivers, alerdedor de 10 a 20 horas para solucionar cualquier problema en el sistema.
- Problemas con la infraestructura web (servidor, hosting)
  - Descripción: Fallos o carencia de capacidades del hosting podrían afectar la capacidad de hostear la página web y su base de datos.
  - Probabilidad: 0.05
  - Impacto en esfuerzo: 10 horas para solucionar problemas de configuración o encontrar un nuevo proveedor.

Riesgos en procesos

- Cambios en los requisitos del proyecto
  - Descripción: Si los requisitos cambian, puede ser necesario rediseñar parte del proyecto.
  - Probabilidad: 0.25
  - Impacto en esfuerzo: Altamente dependiente del cambio de requerimientos, pero se estima que podría ser 10 horas para alterar diseños, o hasta 30 horas si hay que cambiar implementaciones.

Riesgos en insumos

- Costos de hosting más elevados de lo presupuestado.
  - Descripción: Podría ser que a la hora de levantar el servidor que hostea la página web y la base de datos este resulte mucho más caro de lo esperado. Si esto sucede se buscarán otros proveedores y alternativas.
  - Probabilidad: 0.3
  - Impacto en esfuerzo: 30 horas para analizar proveedores alternativos y montar toda la infraestructura de nuevo.
- Falta de acceso a hardware necesario para pruebas
  - Descripción: Se podría determinar que el hardware con el que actualmente se dispone no es adecuado para realizar pruebas.
  - Probabilidad: 0.15
  - Impacto en esfuerzo: 10 horas para conseguir acceso al hardware necesario.

2.5. Cronograma

A continuación se proporciona el cronograma de actividades a realizar en cada semana y el estimado de horas a trabajar por semana.

Las actividades señaladas para cada semana se planean iniciar y completar en esa misma semana.

Semana	Actividades a realizar	Horas estimadas de trabajo por semana
1	100	10
2	200	10
3	300	15
4	400	8
5	500	8
6	600, 601, 602	11
7	700	8
8	800	12
9	900, 901	8
10	1000, 1001	10
11	1100, 1101, 1102	12
12	1200, 1201	10
13	1300, 1301	10
14	1400, 1401	10
15	1500, 1501	12
16	1600, 1601	14