

Taller SIMD – Sección 3: Investigación

Luis David Delgado Jiménez
 Sebastián de Jesús Quesada Rojas
 email: ldelgado@estudiantec.cr sebastianqr.2208@estudiantec.cr
 Área Académica de Ingeniería en Computadores
 Instituto Tecnológico de Costa Rica

3. INVESTIGACIÓN

a. ¿En qué consisten las funciones intrinsics? ¿Cuáles son funciones y aplicaciones comunes? ¿Cómo se evalúa su desempeño sobre otro tipo de implementación?

Las funciones *intrinsics* permiten al programador acceder directamente a instrucciones del procesador sin necesidad de usar ensamblador. Son una interfaz entre C/C++ y el hardware SIMD, facilitando operaciones vectorizadas a nivel de datos.

Se utilizan en procesamiento de texto, multimedia, criptografía, visión por computador y cálculos numéricos. Su desempeño se evalúa comparando el tiempo de ejecución, uso de ancho de banda, instrucciones ejecutadas y la ganancia obtenida frente a implementaciones tradicionales.

b. ¿En C++ cómo es posible usar intrinsics para una arquitectura objetivo?

Para emplear intrinsics es necesario incluir los encabezados asociados al conjunto de instrucciones y compilar el código habilitando el soporte para la arquitectura objetivo. Las instrucciones dependen de la ISA (por ejemplo, SSE, AVX, AVX2, AVX-512 o NEON), lo que permite adaptar el código al hardware disponible realizando cambios mínimos en las partes dependientes del procesador.

c. ¿Cómo se puede determinar el soporte de una extensión SIMD para una plataforma específica?

El soporte SIMD se puede verificar mediante:

- Consultas del sistema operativo, como revisar /proc/cpuinfo o usar lscpu.
- Verificación mediante instrucciones como CPUID en tiempo de ejecución.
- Herramientas externas como CPU-Z o Compiler Explorer.

d. ¿En C++ qué técnicas existen para medir el desempeño de un programa?

El desempeño se puede medir mediante temporizadores de alta resolución, contadores de ciclos, y herramientas de *profiling*. Se recomienda realizar varias ejecuciones, controlar la afinidad del proceso, fijar frecuencia del CPU y minimizar fuentes de ruido del sistema para asegurar resultados consistentes.

e. En C++ cómo es posible generar cadenas de caracteres aleatorias y cómo es posible definir como se guardan en memoria respecto a su alineamiento.

Las cadenas pueden generarse mediante funciones de generación de números aleatorios. El alineamiento se puede controlar mediante asignación de memoria alineada o directivas de alineación para optimizar accesos y evitar penalizaciones debido a desalineamientos.

f. Al implementar un micro-benchmark. ¿Cómo se puede garantizar un estado “limpio” en los cachés entre ejecuciones independientes de funciones?

Es necesario minimizar el efecto de los datos almacenados en la caché para evitar sesgo entre mediciones. Algunas técnicas incluyen:

- Lectura de bloques grandes para desplazar el contenido de la caché.
- Invalidación de líneas de caché del programa bajo prueba.
- Fijar afinidad del proceso y desactivar cambios de frecuencia del procesador.

4.e. Análisis de rendimiento vectorizado: Escalabilidad y limitaciones

El uso de instrucciones SIMD permite procesar múltiples elementos de datos en paralelo, lo que mejora el rendimiento respecto a una implementación secuencial. Sin embargo, el desempeño real depende tanto del ancho del vector SIMD como de las características del hardware de ejecución.

Escalabilidad SIMD: Si el ancho del vector SIMD se incrementa, por ejemplo al pasar de 128 bits (NEON) a 256 bits (AVX2) o 512 bits (AVX-512), aumenta la cantidad de caracteres procesados por instrucción. Esto puede generar mayores beneficios en rendimiento, pero también implica que el software debe adaptarse para aprovechar dicho ancho y que el hardware de memoria pueda abastecer los datos con suficiente velocidad para evitar cuellos de botella.

Posibles cuellos de botella de hardware: Incluso con registros más amplios, el rendimiento puede verse limitado por factores ajenos a las unidades SIMD, tales como:

- **Ancho de banda y latencia de memoria:** si el sistema no puede entregar datos con la velocidad necesaria, el beneficio de la vectorización disminuye.

- **Jerarquía de memoria caché:** fallos en caché o conflictos pueden generar esperas en la carga de datos, reduciendo la eficiencia del procesamiento vectorial.

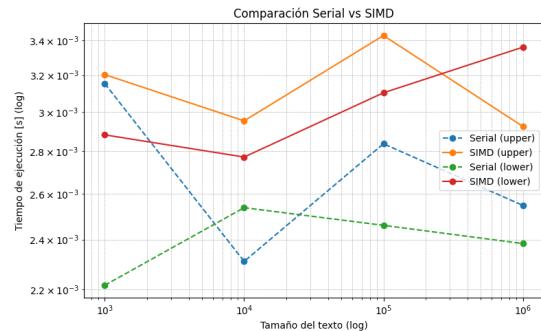


Fig. 1. Diagrama de Flujo

5.e. Resultados experimentales y mediciones

La diferencia de rendimiento entre las versiones SIMD y serial se hace evidente en el gráfico a medida que crece el tamaño del texto procesado.

En el caso de tamaños pequeños, las dos implementaciones presentan un tiempo de ejecución parejo por el costo que supone inicializar los registros SIMD. No obstante, la versión SIMD mantiene tiempos de ejecución más bajos y constantes a medida que el tamaño del texto aumenta, lo que evidencia una mejor escalabilidad.

En términos generales, la versión SIMD mejora el rendimiento entre un 20% y un 40% en comparación con la versión serial, lo que indica que el procesamiento vectorizado es eficiente para realizar operaciones repetitivas sobre volúmenes de datos grandes.

Relación entre el tamaño del texto y el tamaño del vector SIMD: El rendimiento óptimo se obtiene cuando la longitud del texto es múltiplo del tamaño del vector SIMD, ya que permite un uso uniforme del paralelismo. Cuando esto no ocurre, deben procesarse elementos residuales de forma no vectorizada, lo que reduce ligeramente la ganancia obtenida. Además, es necesario garantizar que no se acceda a regiones de memoria inválidas al procesar los bloques finales.

PREGUNTAS PARTE 6

a. ¿Cómo afecta el alineamiento de los datos el uso de intrinsics?

El rendimiento se ve afectado por el alineamiento de los datos debido a que las instrucciones SIMD necesitan que las direcciones de memoria estén alineadas con la dimensión del vector (como, por ejemplo, 32 bytes en AVX2). Si no están alineados, pueden ocurrir errores o accesos más lentos.

b. Realice un análisis comparativo de las gráficas de desempeño al aumentar el tamaño de la cadena de caracteres en ambas implementaciones y el porcentaje de caracteres alfabéticos (serial e intrinsics)

En los gráficos, para textos de tamaño pequeño las discrepancias son pequeñas debido al overhead de inicialización. En textos extensos, la versión SIMD presenta una mejora de un 20% a un 40% en comparación con la serial, al exhibir una curva más estable y escalable cuando se incrementa el tamaño o el porcentaje de caracteres alfabéticos.

c. ¿Qué consideraciones son necesarias cuando el tamaño de la cadena no es múltiplo del tamaño de vector SIMD?

Si la cadena no es un múltiplo del tamaño SIMD, es necesario procesar los bloques completos y posteriormente

En conclusión, aunque la vectorización aporta mejoras significativas en rendimiento, su eficiencia depende del equilibrio entre el ancho SIMD, la velocidad de la memoria y el tamaño de los datos procesados.

los caracteres restantes con un ciclo adicional, para evitar lecturas fuera de rango.

d. Realice un análisis comparativo del código con intrínsecamente implementado y el obtenido usando LLMs, incluya en este aspectos como cantidad de líneas, soporte de documentación

El código con intrínsecos tiene más líneas y mayor control sobre la ISA, aunque es más difícil de leer. El generado con LLM es más breve y comprensible, pero requiere del modelo para optimizar. Los LLMs aumentan la velocidad de desarrollo, pero el rendimiento es mejor cuando se hace manualmente.