Vectorización

Técnicas de Programación Concurrente

Introducción

- Queremos realizar un cómputo "simple" sobre un conjunto grande de datos.
  - Ejemplos: procesar sonido, imágenes, video, entrenar redes neuronales, etc, etc.
- Cada dato es independiente o un pequeño subconjunto lo es.
- Fork join?

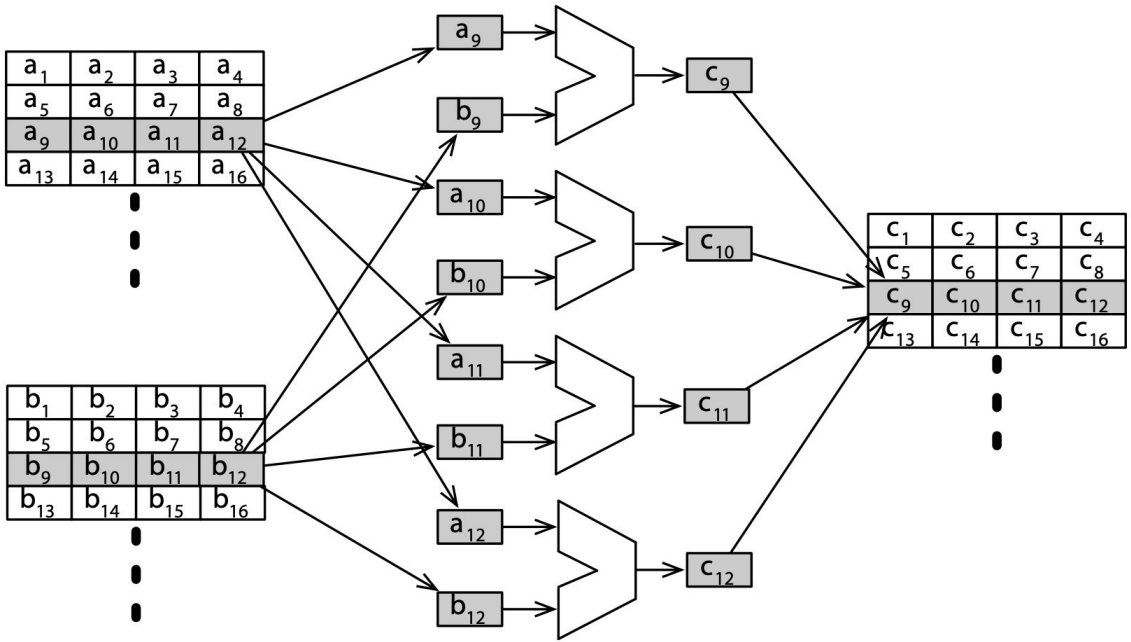
Introducción

- Lanzar incluso una task tiene un costo en cómputo y accesos a memoria (localidad)
- Las CPU son limitadas (de hecho alguna vez fue una sola!)

Operaciones vectoriales

Al "detenerse" la ley de Moore, el aumento de transistores no se tradujo en un aumento de velocidad. En su lugar se agregaron múltiples ALUs para operar sobre los mismos registros de forma concurrente. Esto dió lugar a la introducción de juegos de instrucciones SIMD (Single Instruction Multiple Data) como ser MMX, SSE, AVX en x86 y NEON en ARM.

Operaciones Vectoriales

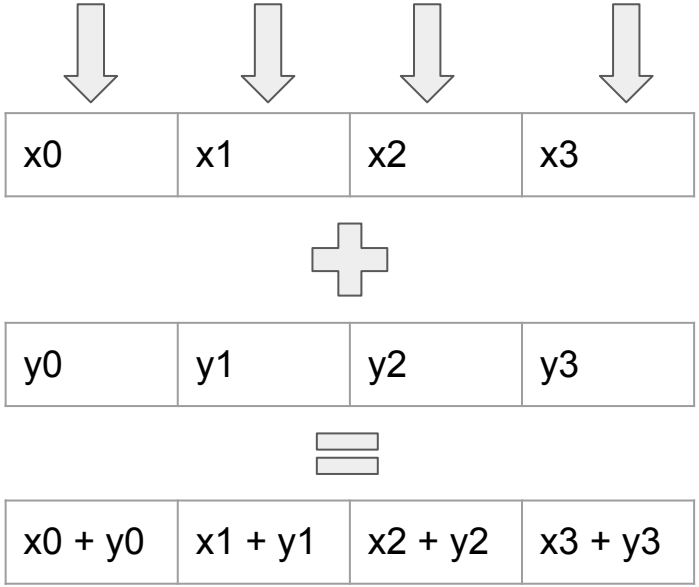


Operaciones vectoriales

- Cada variable (registro) es un vector de tamaño fijo, generalmente de entre 128 y 512 bits
- Se divide en N "lanes" (carriles) según el tamaño de los datos. Por ejemplo un registro de 512 bits puede alojar 16 enteros de 32 bits, o bien 8 flotantes de 64.

Operaciones vectoriales

- Operaciones "verticales" son entre distintos registros en el mismo lane

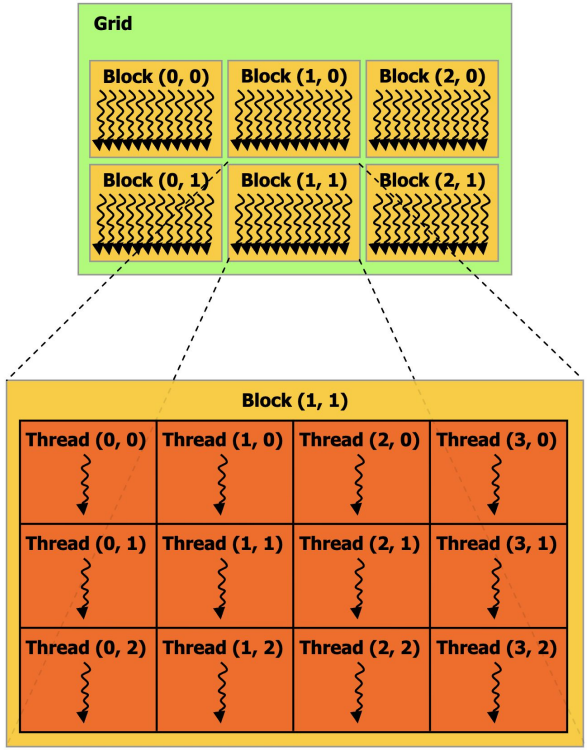


Operaciones vectoriales

- Operaciones "horizontales" reducen un vector a un escalar. Son más lentas.

CUDA

- Compute Unified Device Architecture
- Standard de facto de NVDIA para trabajar sobre GPUs
- Modela "threads" en bloques que trabajan sobre una pequeña porción de memoria independiente, y se pueden direccionar en 1D, 2D, 3D.
- Permite hasta 2048 threads por "Streaming Processor"



CUDA

Ejemplo de código de un "thread" para sumar dos vectores

```
pub unsafe fn add(a: &[f32], b: &[f32], c: *mut f32) {
    let idx = thread::index_1d() as usize;
    if idx < a.len() {
        let elem = &mut *c.add(idx);
        *elem = a[idx] + b[idx];
    }
}
```

Referencias

- Seven Concurrency Models in Seven Weeks: When Threads Unravel (The Pragmatic Programmers), Paul Butcher