Configuración y Optimización de Sistemas de Cómputo
Rendimiento de un Nodo

Master Universitario en Ingeniería Informática
Depto. de Informática de Sistemas y Computadores (DISCA)
Universidad Politécnica de Valencia

Rendimiento de un Nodo

- Introducción
- Roofline Model
  - Intensidad aritmética
  - Rendimiento Pico
  - Ancho de banda
- Medir prestaciones de forma práctica
  - Intel Advisor

Rendimiento de un Nodo

- El rendimiento de un nodo depende de:
  - Rendimiento de pico (FLOP/S) (OP/S)
    - Procesador
      - Scalar, Superescalar, …
      - Coprocesadores (Unidad Vectorial)
    - Tarjetas Aceleradoras
      - GPU, ASIC
  - Tiempo medio de acceso a memoria
    - Jerarquía de Cache
    - Memoria
      - DRAM (DDR5/4, …)
      - HBM, …

Rendimiento de un Nodo

- ¿Qué es más importante?
  - Depende del programa
- ¿Cómo lo podemos saber?
  - Caracterizar el rendimiento
    - Tomar medidas sobre el nodo
    - Analizar el código
      - Tipo de operaciones
      - Operaciones /Acceso a memoria
      - Opciones de paralelización / Vectorización

Roofline Model

- Modelo visual e intuitivo para estimar el rendimiento de una aplicación
  - Rendimiento de pico
    - Con las diferentes optimizaciones
    - Para diversas configuraciones
      - Multi-hilo, monoprocesador, con o sin aceleración
  - Ancho de Banda de Memoria
    - DDR / HBM / …
    - Memoria Cache (L1/L2/L3)

Roofline Model

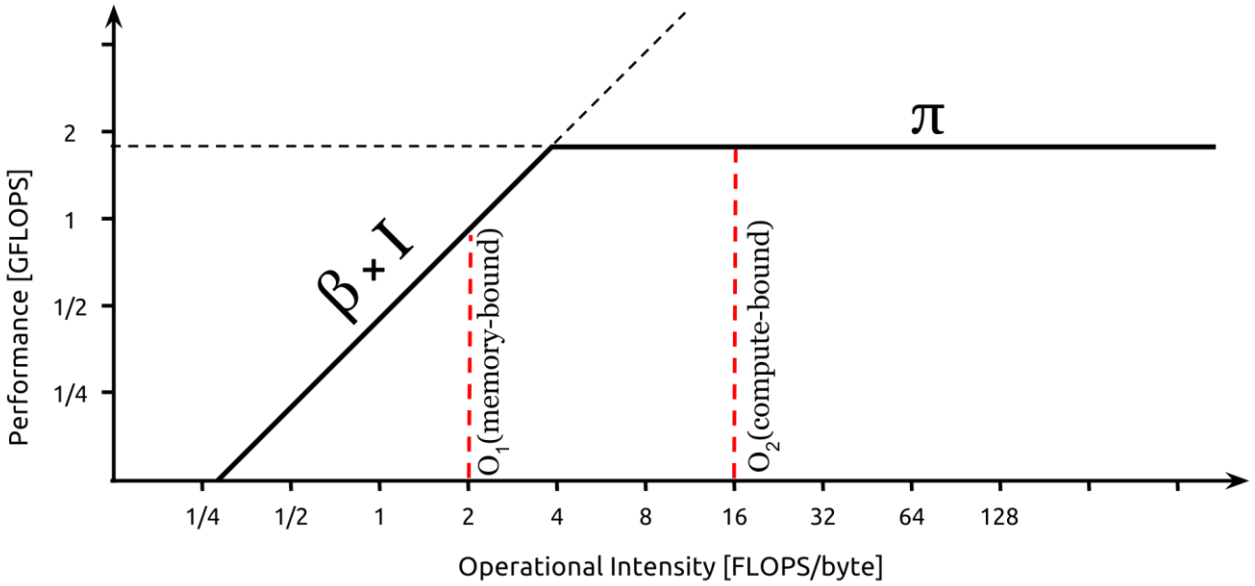
- Rendimiento Alcanzable (RA)

  RA = min { Rendimiento pico (RP),
  IA * Ancho de Banda Pico (BW)}

  - RA (Flops/s)
  - RP (Flops/s) > RA (Flops/s)
  - IA (Intensidad Aritmética)
    - Flops/ byte
  - BW (Bytes/s)

Roofline Model

- β (Ancho de Banda) π (Rendimiento Pico)



Intensidad Aritmética

- Medida de cuantas operaciones (coma flotante) se hacen para un número de bytes que se carga o almacena de memoria

$$IA = \frac{\frac{Flops}{s}}{Transferencias\ Memória} = \frac{flop\ s/s\ econd}{byte\ s/s\ econd} = \frac{flops}{bytes}$$

Intensidad Aritmética

- Ejemplo ([4]): Funcion y = ax + y

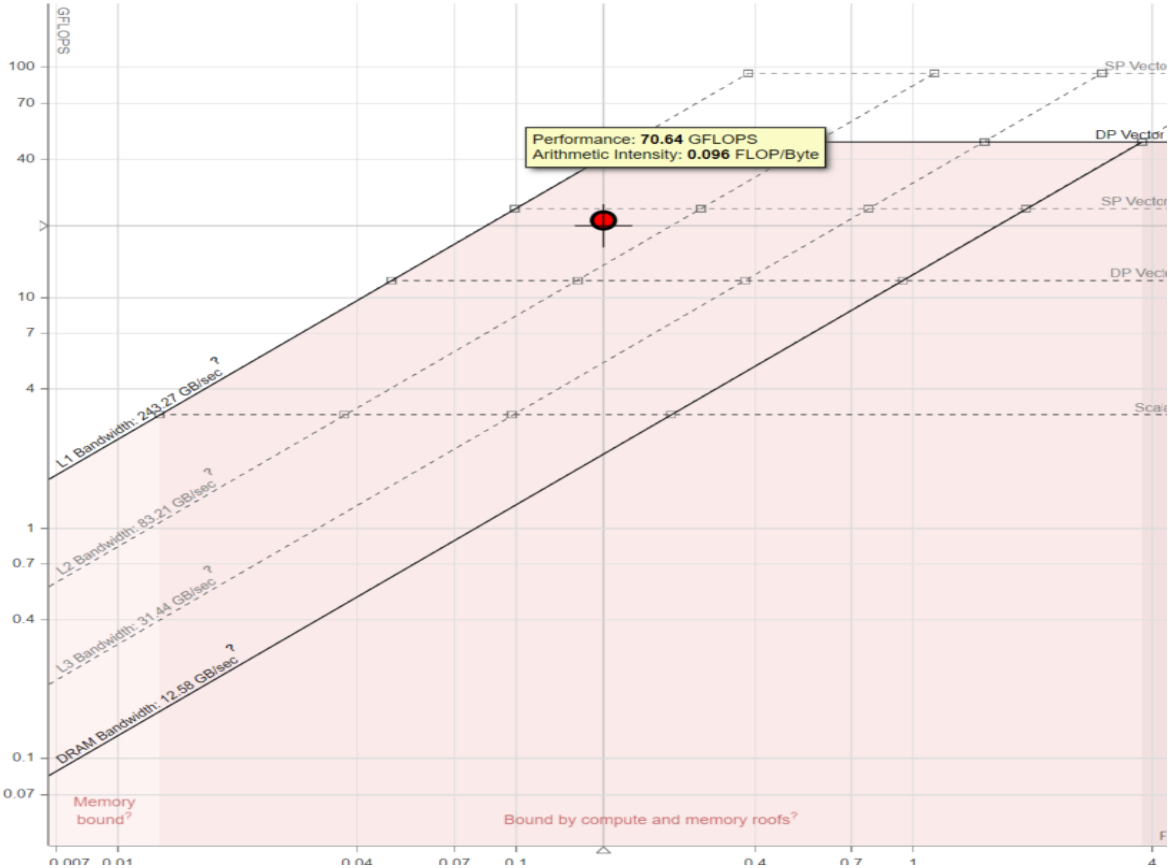
```
void faxpy(size_t n, double a, const double *x, double *y) {
    size_t i;
    for (i = 0; i < n; i++) {
        y[i] = a * x[i] + y[i];
    }
}
```

  - Tráfico de memoria (TM) = (2 loads + 1 store)*n
  - TM = (8*2+8)*n = 24n
  - flops = (mult + add)*n = 2n
  - IA = 2n/24n = 1/12

Roofline Model: Memoria

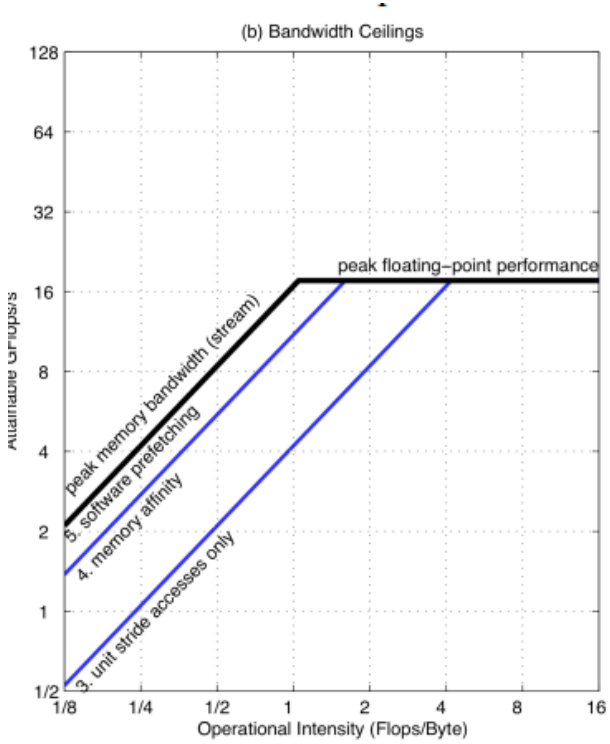
- No todas las transferencias de memoria cuestan lo mismo
  - Latencia Registro < L1 < L2 < L3 < DRAM
- Como afecta esto al roofline model
  - Cada estructura de memoria tiene un ancho de banda cuantificable experimentalmente
  - Dependiendo donde se ubiquen los datos de nuestra aplicación tendremos un limite debido a ancho de banda u otro

Roofline Model: Memoria



Roofline Model: Memoria

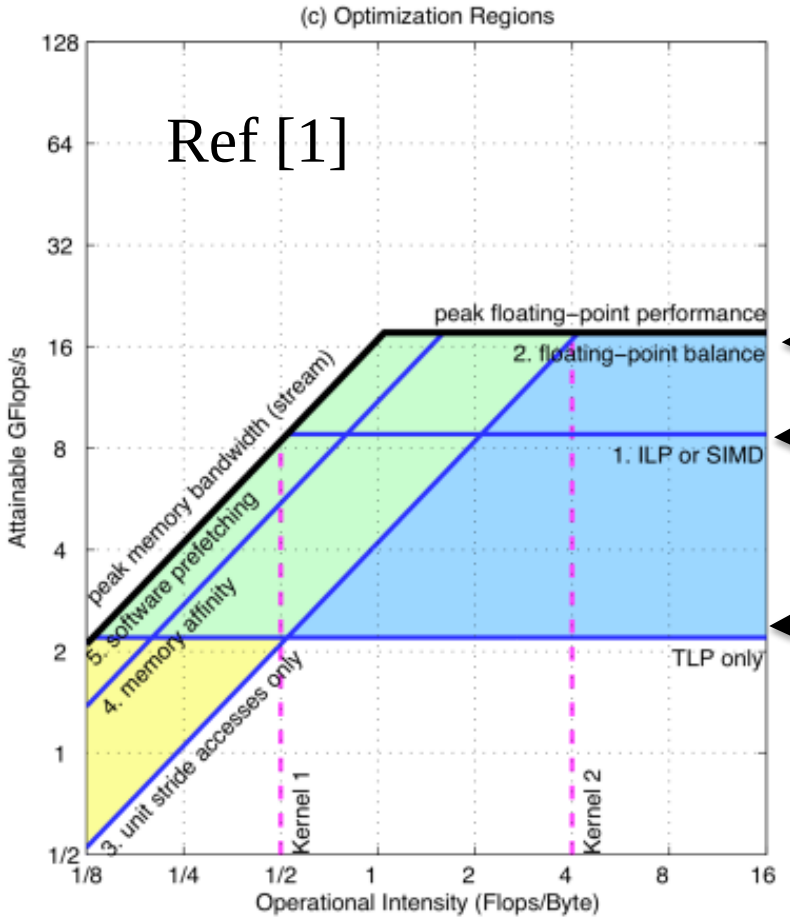
- El ancho de banda efectivo también se puede modular con mecanismos SW
  - Software prefetching:
    - No afecta al BW pero si disminuye el tiempo de acceso medio a memoria



Roofline Model: Computo

- Los procesadores actuales disponen de multiples configuraciones
- ¿Cuál es el rendimiento de pico a considerar?
  - Se definen cotas superiores según el tipo de operaciones/configuración
    - Pico para operaciones FMA (Fused multiply accumulate)
    - Pico monoprocesador /multiprocesador
    - Pico scalar / superscalar / vectorial

Roofline Model: Computo



Ganancias

Multicore + SIMD + FMA

Multicore + SIMD

Multicore

Roofline Model

- Ideas clave
  - 1) Por mucha capacidad de computo que tengas (GPU/Vectores/...) si tu algoritmo esta esperando para traer datos de memoria → **Memory Bound**
  - 2) Con un código muy eficiente y/o una memoria muy rápida existe un limite en el rendimiento marcado por la arquitectura → **Compute Bound**

Ejemplo Práctico

- Análisis del Roofline Model esta soportado en muchas plataformas de desarrollo
  - Las arquitecturas y las aplicaciones son complejas de modelar
  - En sistemas complejos es una herramienta muy útil
    - Aplicaciones formadas por muchos “kernels”
    - Arquitecturas avanzadas, distribuidas y heterogéneas
    - Arquitecturas paralelas
      - Multithread -→ Multi Node

Roofline: Ejemplo Práctico

- Obtener Roofline Model para CPU de Intel
  - Requisitos
    - Intel Advisor
    - Visual Studio
    - Intel C++ compiler
  - Ejemplo de [3]
    - Diferentes "kernels" que se pueden activar/desactivar
      - Diferentes tipos de acceso estructuras y tipos de operaciones
      - Varios niveles de IA (intensidad aritmética)

Roofline: Ejemplo Práctico

- Pasos (visual studio)
  - Abrir proyecto con Visual studio
    - Seleccionar el Compilador de Intel C++
    - Seleccionar la opciones deseadas de compilación
  - Compilar la aplicación
- Pasos (Intel Advisor)
  - Crear un proyecto nuevo
  - Seleccionar donde se encuentra el ejecutable de la aplicación
  - Ejecutar un análisis (selección FLOPS, trip counts, …)

Roofline: Ejemplo Práctico

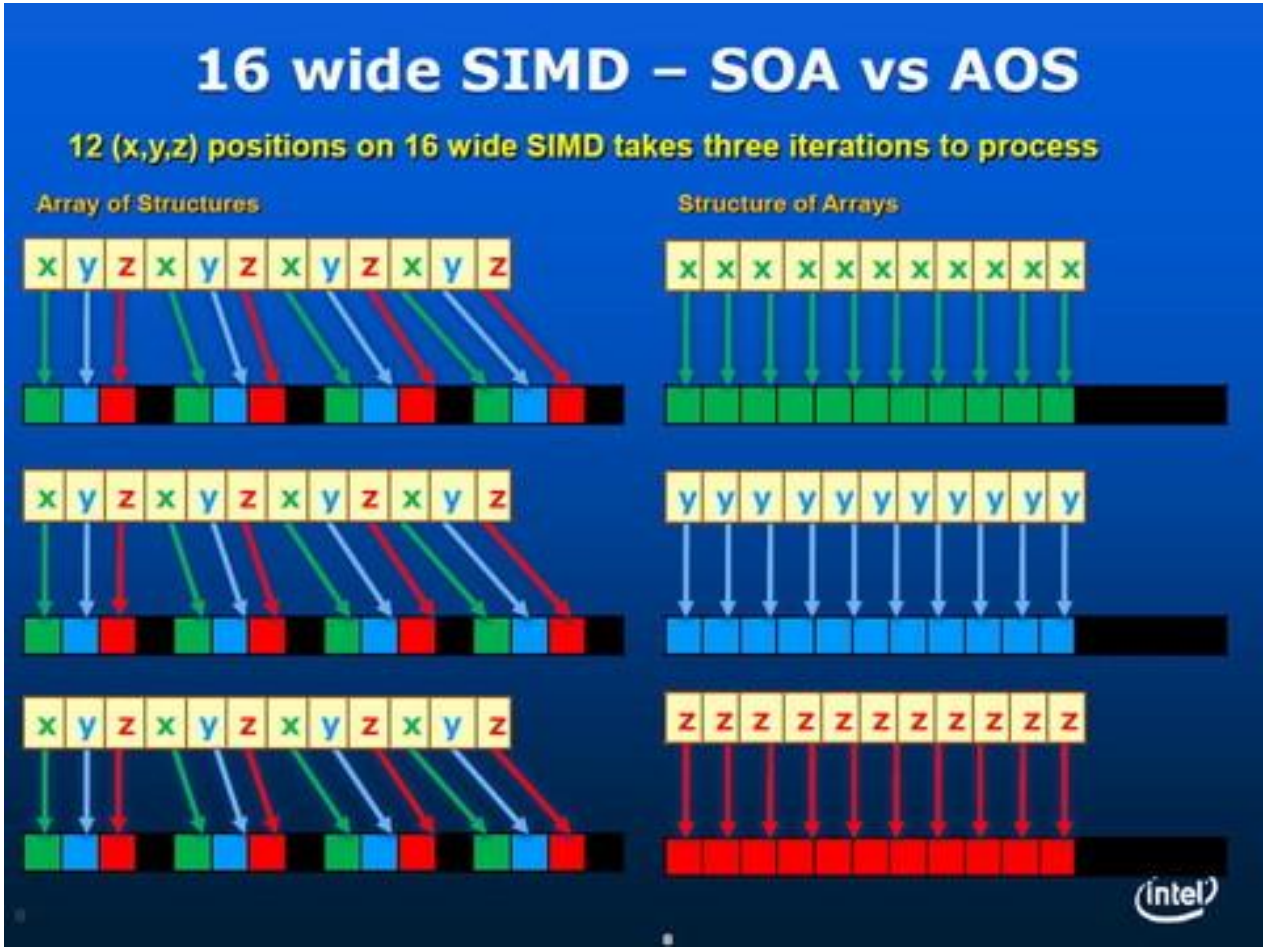
- Evaluación de kernels aritméticos sobre diferentes estructuras de datos
- AoS (Array of Structures) - SoA (Structure of Arrays)

```
struct point3D {
    float x;
    float y;
    float z;
};
struct point3D points[N];
```

```
struct pointlist3D {
    float x[N];
    float y[N];
    float z[N];
};
struct pointlist3D points;
```

SOAs alineados en memoria de forma conveniente facilitan acessos a memoria de stride unitario compatibles con los acessos SIMD

Roofline: Ejemplo Práctico



Bibliografia

- [1] Samuel Williams, Andrew Waterman, and David Patterson, "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures", Communications of the ACM April 2008.
- [2] Intel Advisor Software https://www.intel.com/content/www/us/en/developer/tools/oneapi/advisor.html
- [3] Roofline Model with Intel Advisor https://www.intel.com/content/www/us/en/developer/articles/training/training-sample-intel-advisor-roofline.html
- [4] https://dando18.github.io/posts/2020/04/02/roofline-model
- [5]https://www.intel.com/content/dam/www/public/us/en/documents/presentation/improving-vectorization-efficiency.pdf

Intensidad Aritmética

- Ejemplo ([2]): GEMM (multiplicación de matrices)

```
void dgemm(size_t n, const double *A, const double *B,
const double *C) {
    size_t i, j, k;
    double sum;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            sum = 0.0;
            for (k = 0; k < n; k++) {
                sum = sum + A[i*n+k] * B[k*n+j];
            }
            C[i*n+j] = sum;
        }
    }
}
```

Intensidad Aritmética

- Ejemplo ([2]): GEMM (multiplicación de matrices)

```
for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            sum = 0.0;
            for (k = 0; k < n; k++) {
                sum = sum + A[i*n+k] * B[k*n+j];
            }
            C[i*n+j] = sum;
        }
    }
}
```

- Flops = 2n3
- TM = loads (Aik, Bik, Cij) + Stores (Cij) =
- IA =