

## IMT2112 - Algoritmos Paralelos en Computación Científica

Localidad de los datos

Elwin van 't Wout

22 de agosto de 2019



PONTIFICIA  
UNIVERSIDAD  
CATÓLICA  
DE CHILE

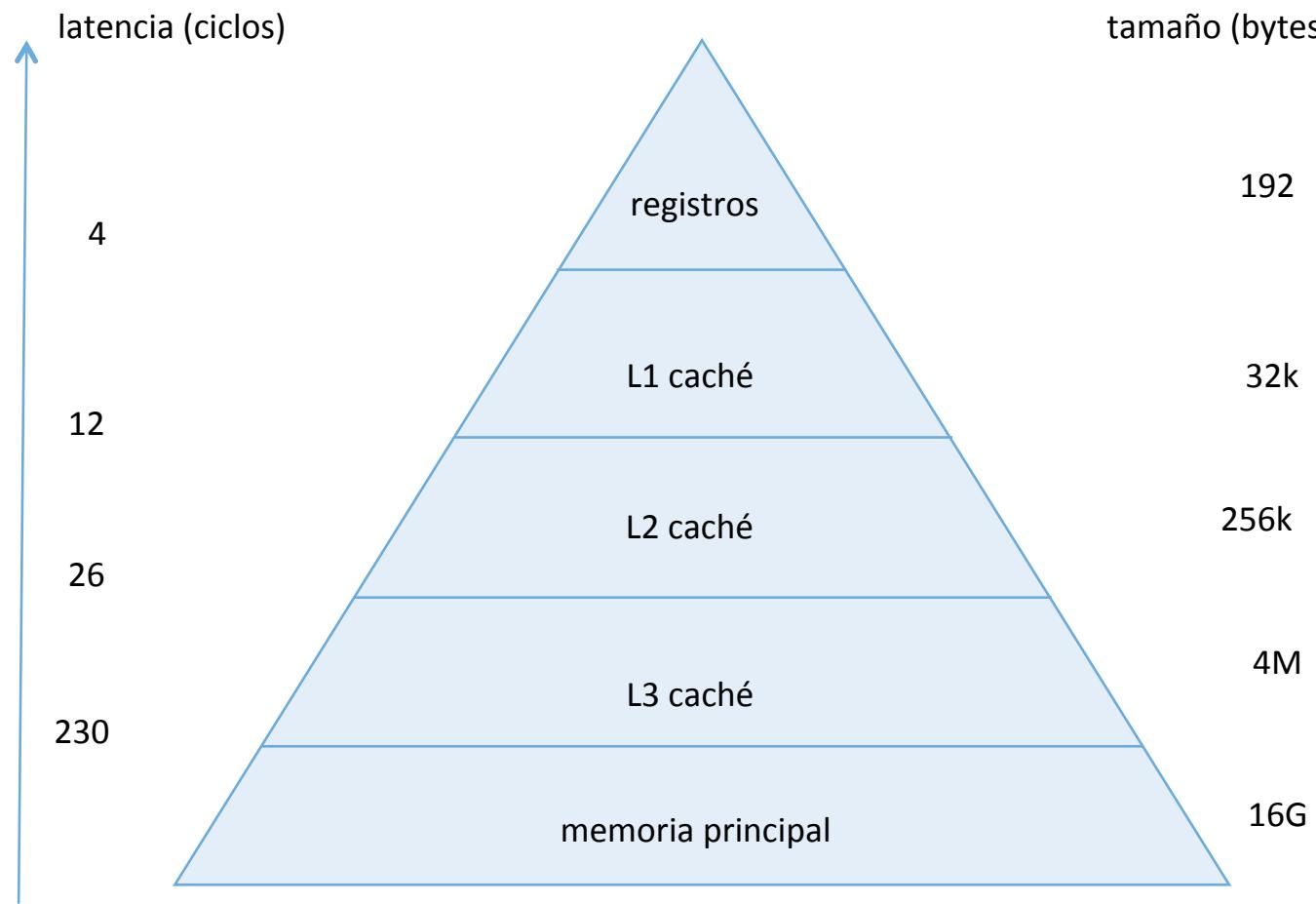
Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl

# Clase previa

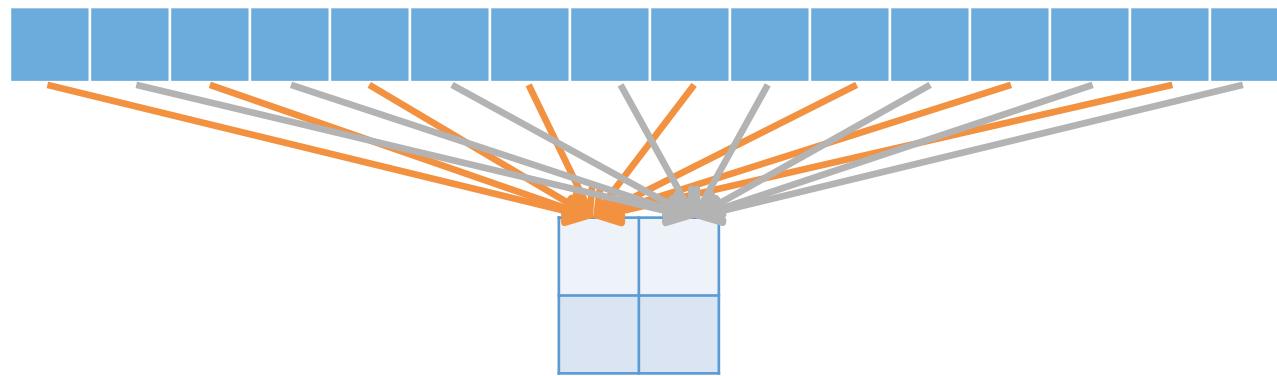
- Jerarquía de memoria
- Mapeo de caché

# Jerarquía de memoria



# Mapeo del caché

Mapeo de caché asociativo  $n$ -vías: combinación de mapa de caché directo y asociativo



# Agenda

- ¿Como se extende la arquitectura de memoria a procesadores de múltiples núcleos?
- ¿Como aprovechar la arquitectura de memoria en el diseño de algoritmos?

# La arquitectura de múltiples núcleos

Sección 1.4 del libro de Eijkhout

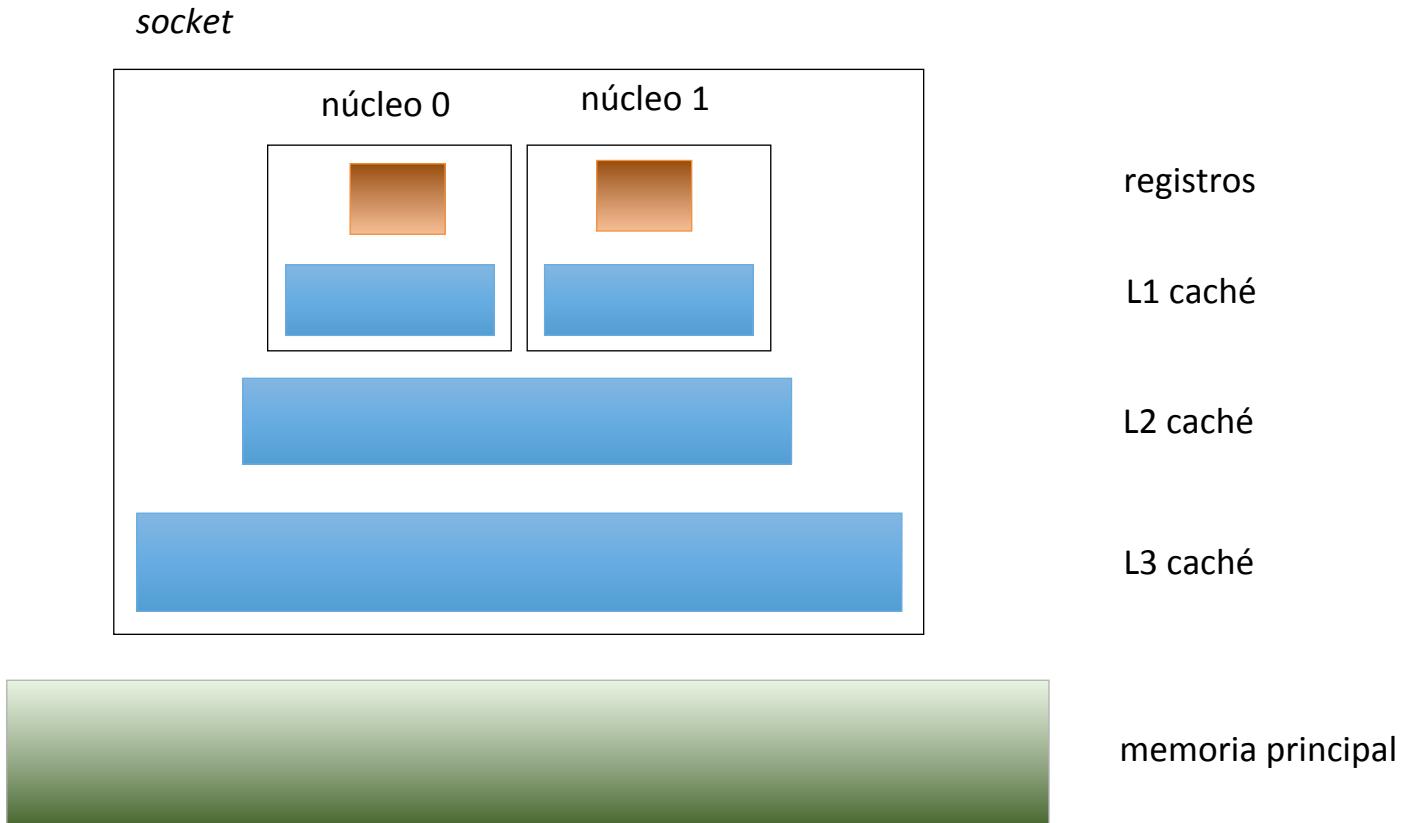
# Las limitaciones de procesadores de un solo núcleo

- La velocidad del reloj ha alcanzado un límite
  - el aumento de la velocidad del reloj sobrecalentaría el hardware
- El paralelismo a nivel de instrucción ha alcanzado un límite
  - cantidad de datos intrínsecamente paralelos
  - capacidades del compilador

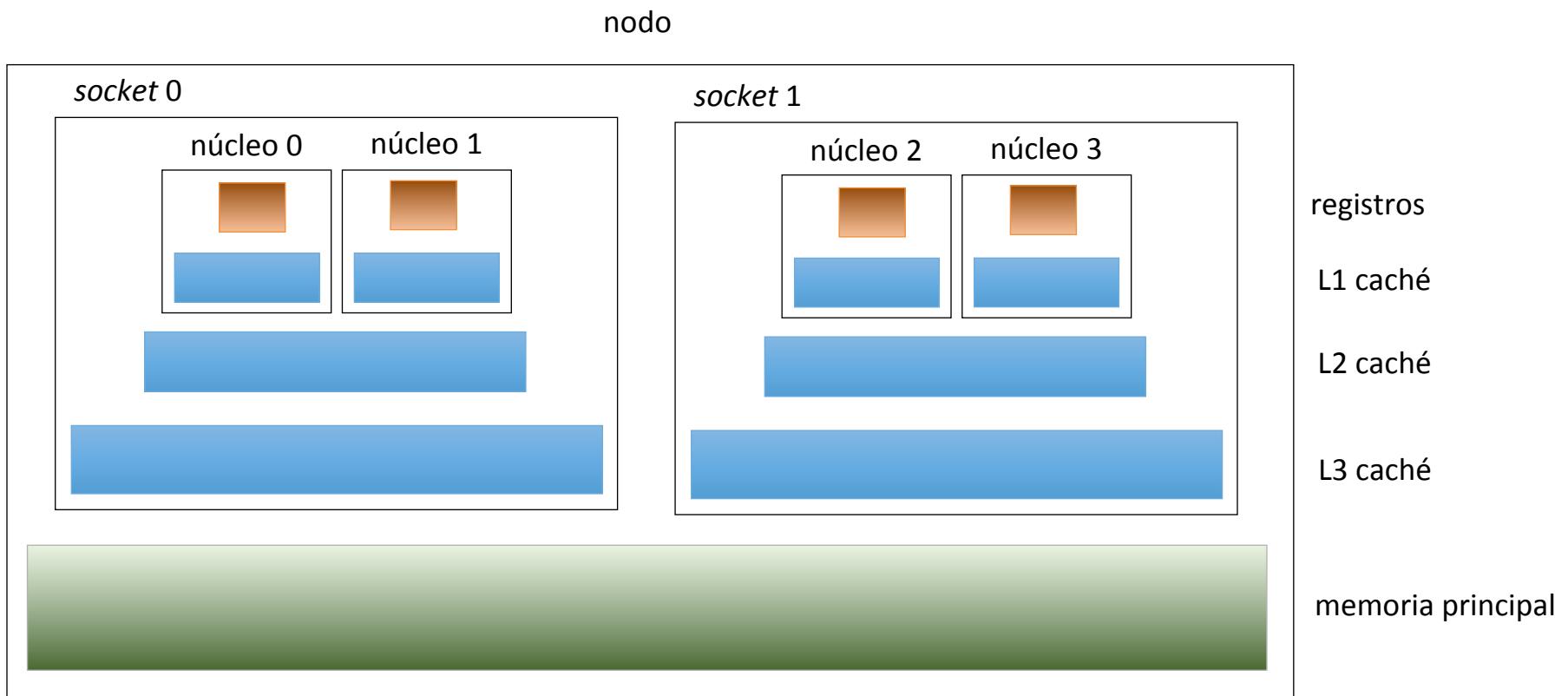
# Los procesadores *multi-core*

- En lugar de la sofisticación de un único procesador, use múltiples núcleos de procesamiento en el chip
- Dividir el procesador en diferentes unidades de procesamiento
  - entre L1 y L2, o entre L2 y L3

# Los procesadores *multi-core*



# Los procesadores *multi-core*



# Terminología

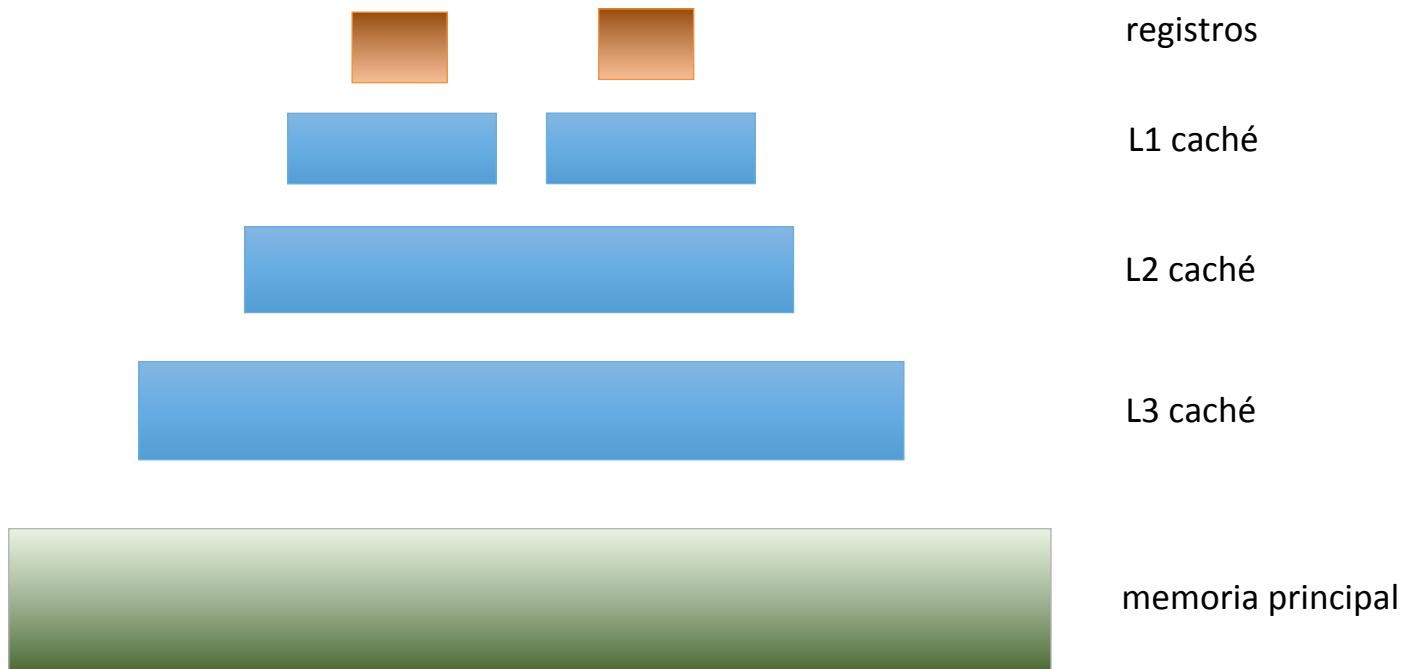
- El dispositivo informático contiene
  - núcleo (*core*): la unidad de ejecución y el caché L1
  - *socket*: múltiples núcleos y un caché L2 compartido
  - nodo (*node*): múltiples *sockets* y una memoria principal compartida
  - red (*network*): múltiples procesadores conectadas con memoria distribuida
- Tenga en cuenta que existen diferentes arquitecturas
  - los núcleos pueden tener un caché L2 privado
  - distintos núcleos/*sockets/nodos* pueden tener comunicación directa

# Terminología

- Memoria compartida (*shared memory*): la memoria es compartida por diferentes unidades de ejecución
  - distintas unidades usan el mismo espacio de memoria
- Memoria distribuida (*distributed memory*): la memoria se separa para las diferentes (conjuntos de) unidades de ejecución
  - una unidad sólo puede usar el parte de memoria asignado
- Esto normalmente se refiere a la memoria principal
  - en comparación con el caché *privado* o *local*

# Coherencia de caché

Problema: distintas copias del mismo dato puede terminar en diferentes ubicaciones en el caché y los registros



# Coherencia de caché

- ¿Cómo evitar que diferentes núcleos cambien el mismo dato?
- Para la arquitectura de memoria compartida, esto es parte del hardware y sistema operativo
  - ‘cache coherence protocol’
- Para la arquitectura de memoria distribuida, esta es responsabilidad del programador

# Localidad y reutilización de datos

Sección 1.6 del libro de Eijkhout

# La localidad en computación científica

- El concepto de localidad se puede encontrar en
  - problemas de ingeniería
  - algoritmos numéricos
  - códigos de software
  - jerarquía de memoria

# La localidad en problemas de ingeniería

- Muchos problemas de ingeniería son fenómenos locales
  - propagación de ondas
  - difusión de calor
- Otros son fenómenos globales
  - campos gravitacionales

# La localidad en métodos numéricos

- Muchos algoritmos en computación científica tienen localidad de datos

- Esquema de Euler para la integración del tiempo

$$y'(t) = f(t, y(t)) \Rightarrow y_{n+1} = y_n + h f(t_n, y_n)$$

- Esquema de diferencia finita para ecuaciones diferenciales parciales

$$-\frac{\partial^2 u}{\partial x^2} - k^2 u = f \Rightarrow -\frac{u(x_{n+1}) - 2u(x_n) + u(x_{n-1})}{(\Delta x)^2} - k^2 u(x_n) = f_n$$

- Los algoritmos globales incluyen métodos de elementos de borde

# La localidad y reutilización de datos

- El mapeo de caché conserva la localidad de los datos
  - los datos que son locales en el código del software pueden permanecer juntos en los niveles de caché
  - la localidad de los datos mejora la reutilización de datos en el caché y, por lo tanto, el rendimiento

# La intensidad aritmética

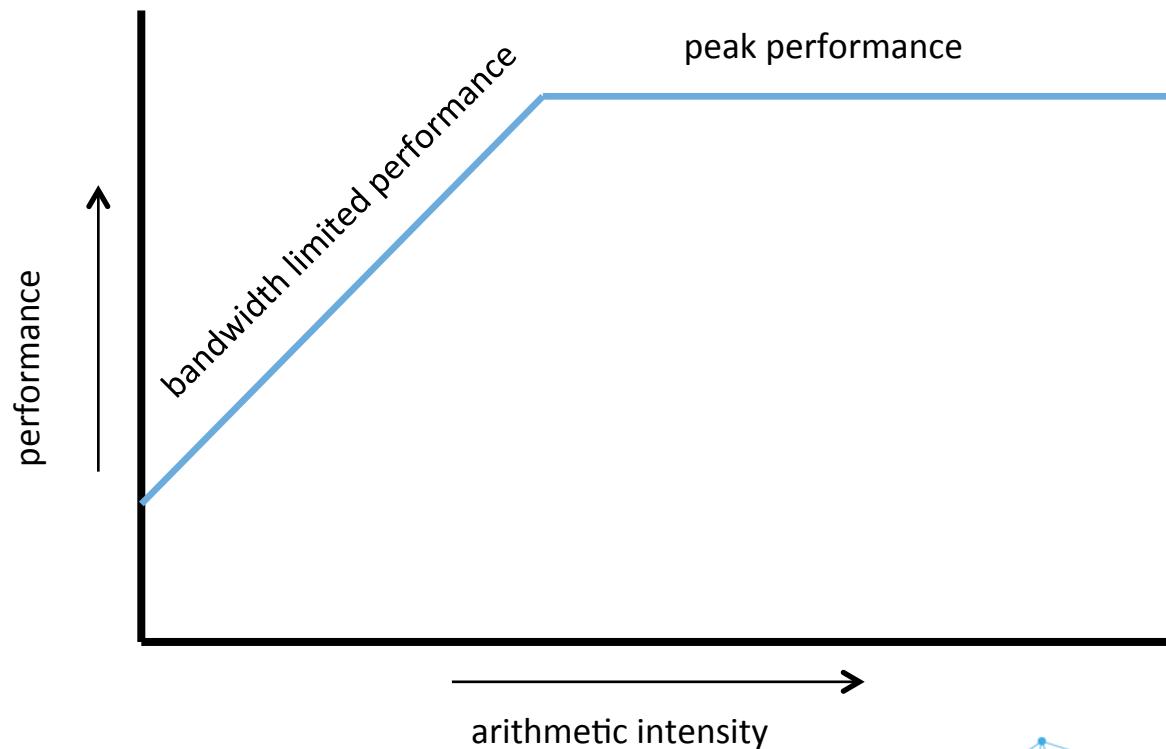
- No todos los algoritmos son adecuados para la reutilización de datos
- Una medida para la capacidad de reutilización es la ‘intensidad aritmética’
  - para  $n$  el número de elementos de datos y  $f(n)$  el número de operaciones, la intensidad aritmética se define como:  
 $f(n) / n$
  - es la relación de operaciones por elemento de datos cargado
  - una mayor intensidad aritmética proporciona más oportunidades para la reutilización de datos

# El modelo *roofline*

- El rendimiento máximo está determinado solo por la arquitectura de la computadora
  - frecuencia de reloj y número de núcleos
  - se supone que el flujo de datos no genera retrasos
- El rendimiento también está limitado por la transmisión de los datos
  - p.ej. el ancho de banda y la latencia
  - recuerde que ancho de banda es datos por segundo y la intensidad aritmética operaciones por dato, entonces:  
$$\text{operaciones / segundo} = \text{intensidad aritmética} \times \text{ancho de banda}$$

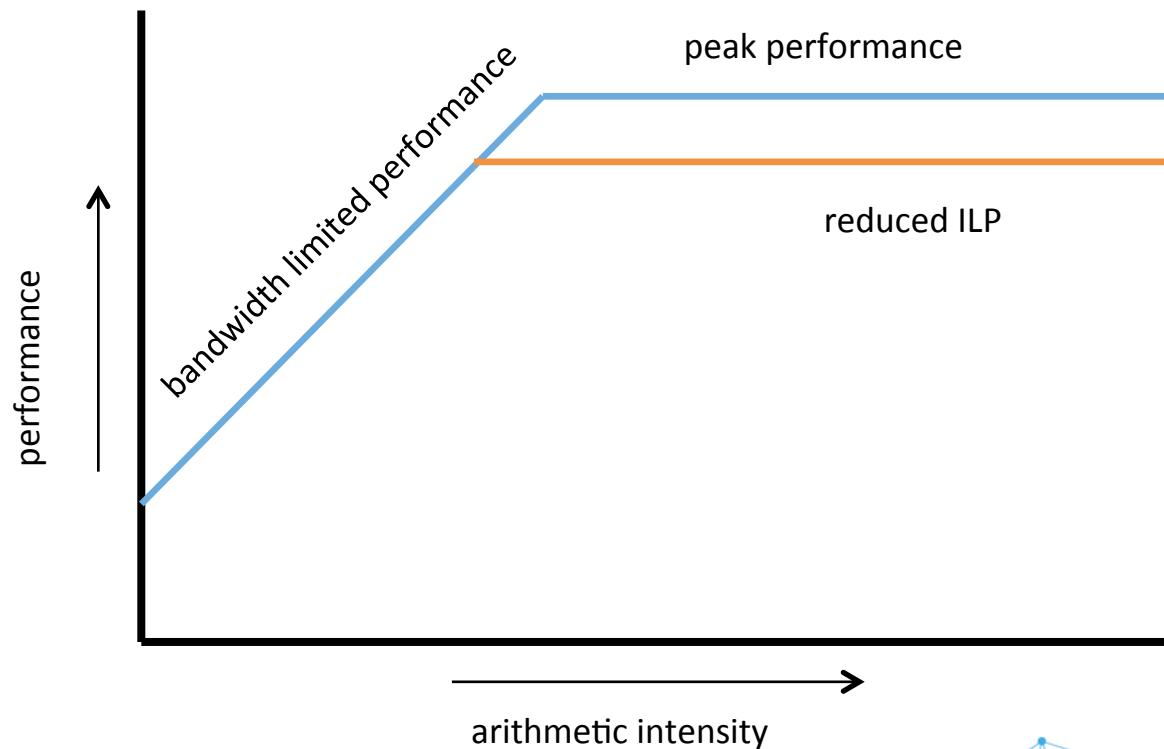
# El modelo *roofline*

El rendimiento de los algoritmos pueden ser limitados por el cálculo (*compute-bound*) o la transmisión de datos (*bandwidth-bound*)



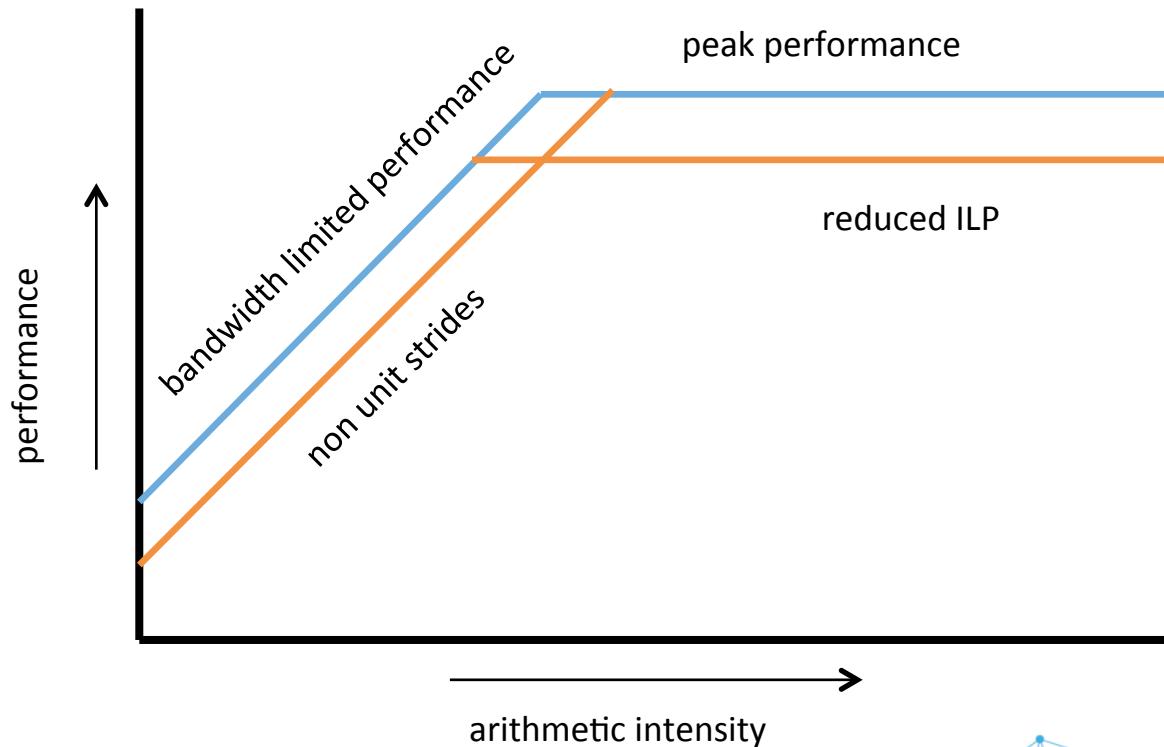
# El modelo *roofline*

El rendimiento máximo depende p.ej. del paralelismo de nivel de instrucción



# El modelo *roofline*

El rendimiento limitado por el ancho de banda depende p.ej. del paso en los bucles



# Localidad

- El rendimiento depende en gran medida de la transmisión de datos
  - es mas común tener algoritmos *bandwidth-bound* que *compute-bound*
- Mejorar la reutilización de datos en el caché
  - localidad temporal
  - localidad espacial

# Localidad temporal

- Es probable que los datos utilizados recientemente aún se puedan encontrar en la memoria caché
  - los cachés normalmente usan la política de reemplazo menos utilizada recientemente (LRU)
- Aquí, “reciente” significa el tiempo antes de que se vacíe el *slot* del caché, que depende del tamaño de caché y la asociatividad
  - depende del tamaño de caché y la asociatividad
- Estrategia del programador:
  - localizar el uso de datos dentro de unas pocas líneas de código

# Localidad espacial

- La dirección de memoria tiene un orden
  - es una cadena de bits que se puede interpretar como un número
- La dirección de memoria determina:
  - la ubicación de los datos en una línea de caché
  - la ubicación de la línea de caché en un conjunto de caché
- El mapeo directo de caché preserva las secuencias de datos
- Estrategia del programador:
  - dejar que las instrucciones funcionen en elementos de datos vecinos

# Localidad del núcleo

- En procesadores multi-core, los diferentes núcleos tienen niveles de caché privado
- La rama en el caché afecta a la localidad espacial
  - los datos que son locales en la memoria principal pueden terminar en conjuntos de caché separados
  - esto está *hard-wired* en la arquitectura del procesadores
- La coherencia de caché afecta a la localidad temporal
  - los datos en el caché privado podrían haber sido invalidados por otro núcleo

# *El prefetch stream*

- Recuerde la arquitectura de von Neumann de buscar, ejecutar, almacenar
  - no es necesario esperar hasta que termina este proceso para iniciar otra instrucción
- Si se busca líneas de caché consecutivas, las siguientes ya se buscarán previamente antes de solicitarlas
  - La localidad espacial de las líneas de caché consecutivas no conduce a prohibir los vaciamientos de caché
  - Es implementado en el hardware y sistema operativo

# Asignación matricial

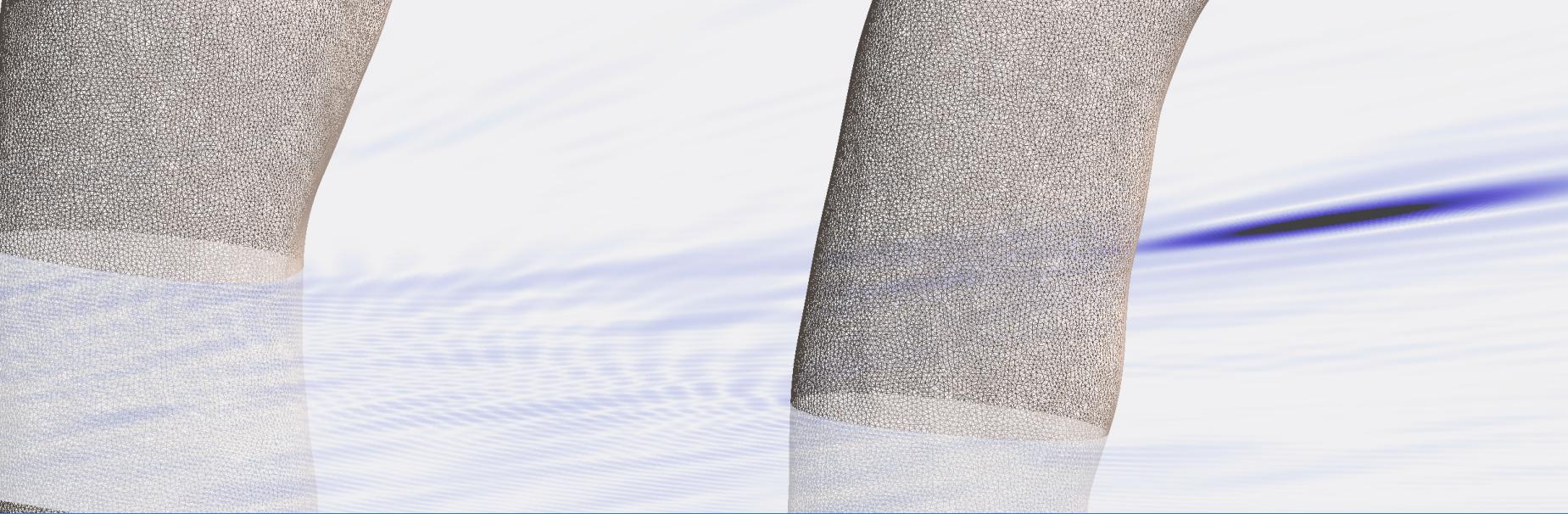
- El orden de los datos en la memoria viene dado por la dirección de memoria
  - la dirección es funcional, no física
- La asignación de memoria de los arreglos de datos se realiza consecutivamente
- Las matrices multidimensionales requieren una numeración unidimensional
  - *row-major ordering* (C, Python)
  - *column-major ordering* (Fortran)

# Resumen

- Coherencia de caché
- Localidad de los datos
- Intensidad aritmética
- El modelo *roofline*

# Clase siguiente

- Eficiencia paralela



## IMT2112 - Algoritmos Paralelos en Computación Científica

Localidad de los datos

Elwin van 't Wout

22 de agosto de 2019



PONTIFICIA  
UNIVERSIDAD  
CATÓLICA  
DE CHILE

Facultad de Matemáticas • Escuela de Ingeniería

imc.uc.cl