Small example:

Cache: - direct mapped (1-way)
  - cacheline = 2 doubles
  - 2 cachelines

Array \( x = x[0], ..., x[j3] \) accessed twice, cold cache

1.) for \( j = 0:1 \)
   for \( i = 0:7 \)
      access \( x[i:j] \)

   access pattern: 012...7012...7

2.) for \( j = 0:1 \)
   for \( i = 0:2:7 \)
      access \( x[i:j] \)
   for \( i = 1:2:7 \)
      access \( x[i:j] \)

   access pattern: 0246135202461357

3.) for \( j = 0:1 \)
   for \( k = 0:1 \)
      for \( i = 0:3 \)
         access \( x[2i+4:j] \)

   access pattern: 0123012345674567

8 misses, 8 hits
neighbor use \( \checkmark \)
data reuse \( \checkmark \)

16 misses, 0 hits
neighbor use \( \checkmark \)
data reuse \( \checkmark \)

4 misses, 12 hits
(optimal)
neighbor use \( \checkmark \)
data reuse \( \checkmark \)

Types of reuse

Data reuse: reuse of data brought into cache before it is evicted

Neighbor reuse: use of data in the same cacheline of the requested data before it is evicted
Accessing a vector

Cache: 2-way
  - cacheline = 4 doubles

Array: \( x[0], \ldots, x[\text{size}-1] \), \( n > \text{cache size} \)
  - accessed once \( \implies \) no data reuse

1. stride = 1: \( 012 \ldots \)
   - neighbour use, \( \frac{n}{4} \) misses

2. stride = 2: \( 024 \ldots (\text{size}-2)135 \ldots (\text{size}) \)
   - some neighbour use, \( \frac{n}{2} \) misses

3. stride \( \geq 4 \): \( 048 \ldots (\text{size}-4)159 \ldots (\text{size}3) \)
   - no neighbour use, \( n \) misses

Where does this occur?

2) Arrays (e.g., images) are stored in row-major order (in C):

Access is column-wise + \( n \) divisible by power of 2
  \( \implies \) no neighbour use

Typical pattern in image processing (e.g., 256, 512, \ldots)