

Today

- Learning Outcomes
 - Define stride.
 - Relate stride to locality.
 - Evaluate the cache performance of strided access patterns.
- Reading
 - 6.5, 6.6

Locality: Spatial and Temporal

- So far, we've examined access patterns where spatial and temporal locality are similar, if not identical.
- When we read an array sequentially, the data we access is physically close together (spatial locality) and we are accessing it at approximately the same time (temporal locality).
- What happens when we change that?

Building Intuition (1a)


- Imagine that I have an array that fits in the L1 cache.
- Let's say that I read through the array many, **many** times, reading a single byte on each access.
- What would the latency be of each access?

Building Intuition (1b)

- Imagine that I have an array that fits in the L1 cache.
- Let's say that I read through the array many, **many** times, reading a single byte on each access.
- What would the latency be of each access?
 - **Whatever the latency of the L1 cache is.**
 - Sure, I might take misses the first time through the array, but if my cacheline is of size N , we take one miss to $N-1$ hits.
 - Example: $N = 64$
 - 1 miss and 63 hits
 - That's a miss rate of 1.6%; 98.4% of the accesses are hits
 - And on every subsequent iteration, all the accesses are hits.

Building Intuition (1c)

- Imagine that I have an array that fits in the L1 cache.
- Let's say that I read through the array many, **many** times, reaching a single byte on each access.
- What would the latency be of each access?
 - Whatever the latency of the L1 cache is.
- Now, let's say that instead of reading every byte, *I read every 8th byte*; what will be the latency of each access?
 - Should still be the latency of the L1 cache



This is called **strided** access: the **stride** is the number of bytes between accesses.

Building Intuition (2a)

- Now, let's imagine I have a big (huge) array – larger than my cache.
 - If I access every byte, what will my latency be (approximately)?

Building Intuition (2b)

- Now, let's imagine I have a big (huge) array – larger than any of my caches.
 - If I access every byte, what will the latency be for each access (approximately)?
 - Close to the latency of the cache -- how far away will depend on the cache size.
 - Examples: Let's say a miss is 5x slower than a hit.
 - 32-byte cacheline: $(5 + 31) / 32 = 1.13$ the cost of a hit
 - 64-byte cacheline: $(5 + 63) / 64 = 1.06$ the cost of a hit
 - 128-byte cacheline: $(5 + 127) / 128 = 1.03$ the cost of a hit
 - The larger the cache line, the closer to the hit time purely sequential access.

Building Intuition (2c)

- Now, let's imagine I have a big (huge) array – larger than any of my caches.
 - If I access every byte, what will the latency be for each access (approximately)?
 - Close to the latency of the cache -- how far away will depend on the cache size.
 - Examples: Let's say a miss is 5x slower than a hit.
 - 32-byte cacheline: $(5 + 31) / 32 = 1.13$ the cost of a hit
 - 64-byte cacheline: $(5 + 63) / 64 = 1.06$ the cost of a hit
 - 128-byte cacheline: $(5 + 127) / 128 = 1.03$ the cost of a hit
 - Now let's say that the L1 is 4 KB and I access one byte out of every 4 KB, what will be the latency of each access?

Building Intuition (2d)

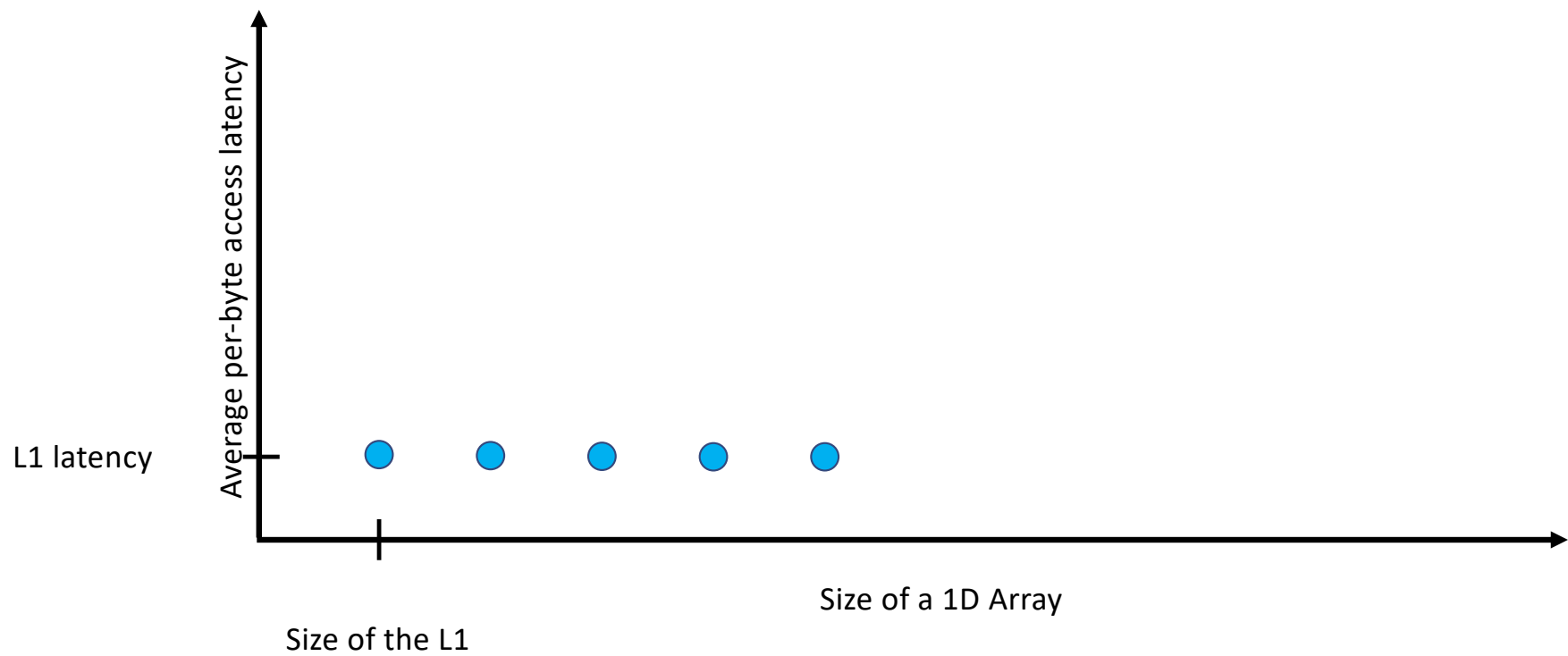
- Now, let's imagine I have a big (huge) array – larger than any of my caches.
 - If I access every byte, what will the latency be for each access (approximately)?
 - Close to the latency of the cache -- how far away will depend on the cache size.
 - Examples: Let's say a miss is 5x slower than a hit.
 - 32-byte cacheline: $(5 + 31) / 32 = 1.13$ the cost of a hit
 - 64-byte cacheline: $(5 + 63) / 64 = 1.06$ the cost of a hit
 - 128-byte cacheline: $(5 + 127) / 128 = 1.03$ the cost of a hit
 - Now let's say that the L1 is 4 KB and I access one byte out of every 4 KB, what will be the latency of each access?
 - Approximately the latency of the data source (or the lower level cache).

Strided Access Patterns are Common

- Strided patterns occur when we:
 - Pack structures in an array and access only a particular field in the structure
 - In matrix computations
 - In many scientific applications.
- Strided access patterns can also be used to reveal the characteristics of your memory hierarchy.

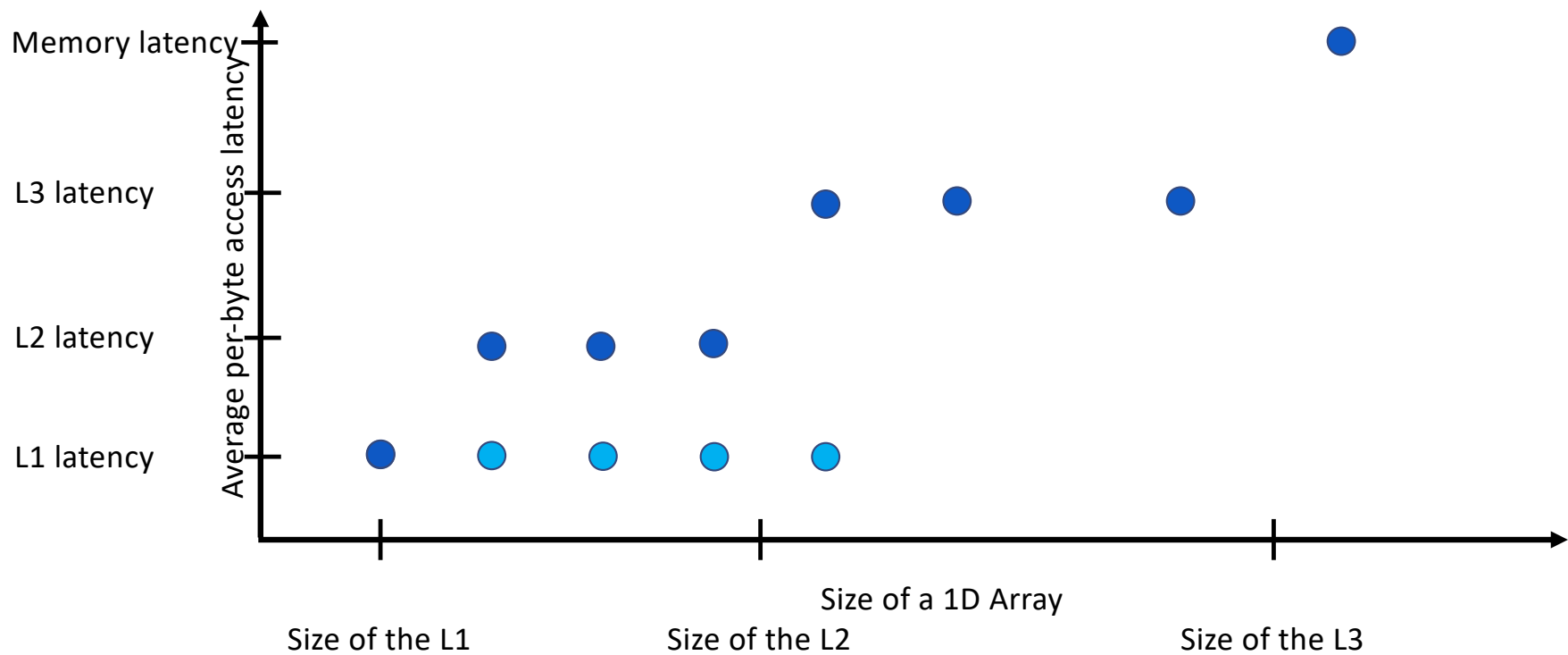
Performance Patterns

Workload: Read every byte of the array many times.



Performance Patterns

Workload: Read every N^{th} byte, where N is the size of an L1 cache line.



Points to Ponder

- How might this look different if our stride were $\frac{1}{2}$ a cache line?
- Twice a cache line?
- What happens if our accesses are larger than single bytes?