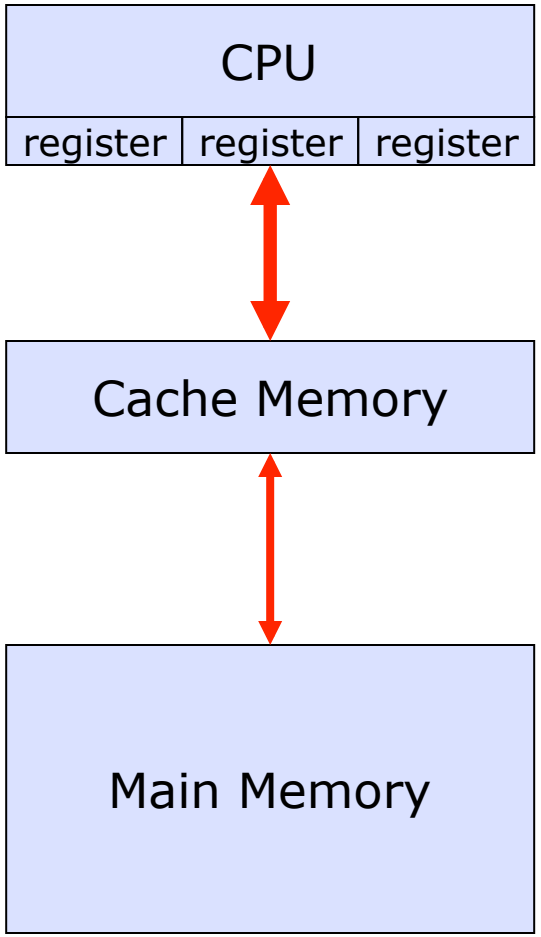
Lecture 5: More on Cache Memory

William Gropp
www.cs.illinois.edu/~wgropp

Simplified Computer Architecture

- Main memory contains the program data
- Cache memory contains a *copy* of the main memory data
  - Cache is *faster* but consumes more space and power
  - In our analysis so far, we assumed *infinitely faster*
- Registers contain working data only

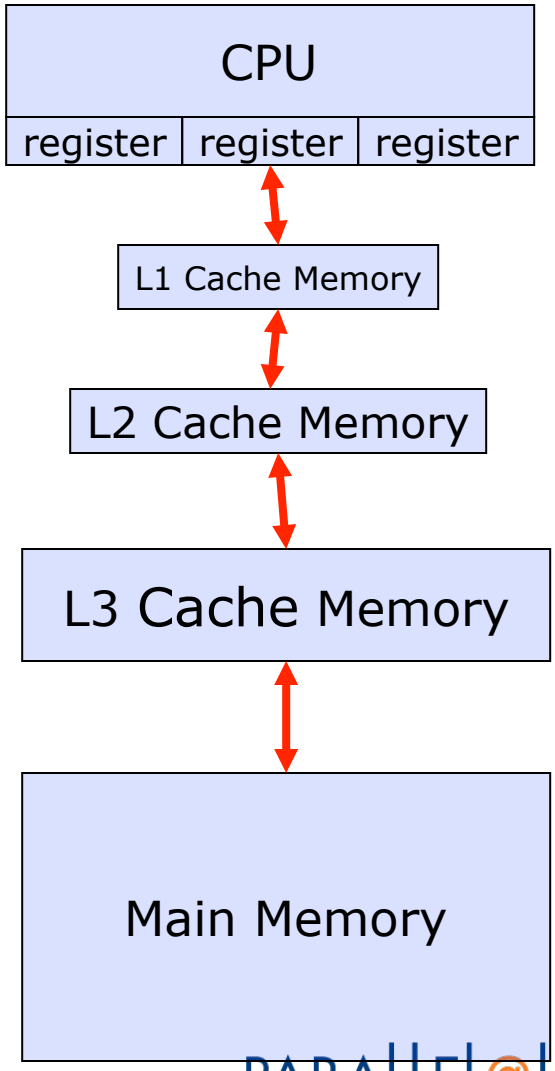


Importance of Memory in Performance Bounds

- We have seen:
  - ♦ Loads and stores can be as important as floating point operations
  - ♦ Simple models that look at just sustained memory bandwidth (and ignore details of cache effects) can provide useful *bounds* on performance
    - Recall the sparse matrix-multiply example
    - True for problems where the majority of data accesses are consecutive
  - ♦ Note that this is a bound, a guaranteed-not-to-exceed value for the performance*
    - * - based on the assumptions of the model

Simplified Computer Architecture II

- Because of the way cache is implemented in hardware, there are tradeoffs between size (number of bytes or capacity), speed, and power
- Main memory contains the program data
- Multiple Cache memories contain a copy of the main memory data
  - Cache is faster but consumes more space and power
  - Cache items accessed by their address in main memory
  - L1 cache is the fastest but has the least capacity
  - L2, L3 provide intermediate performance/size tradeoffs

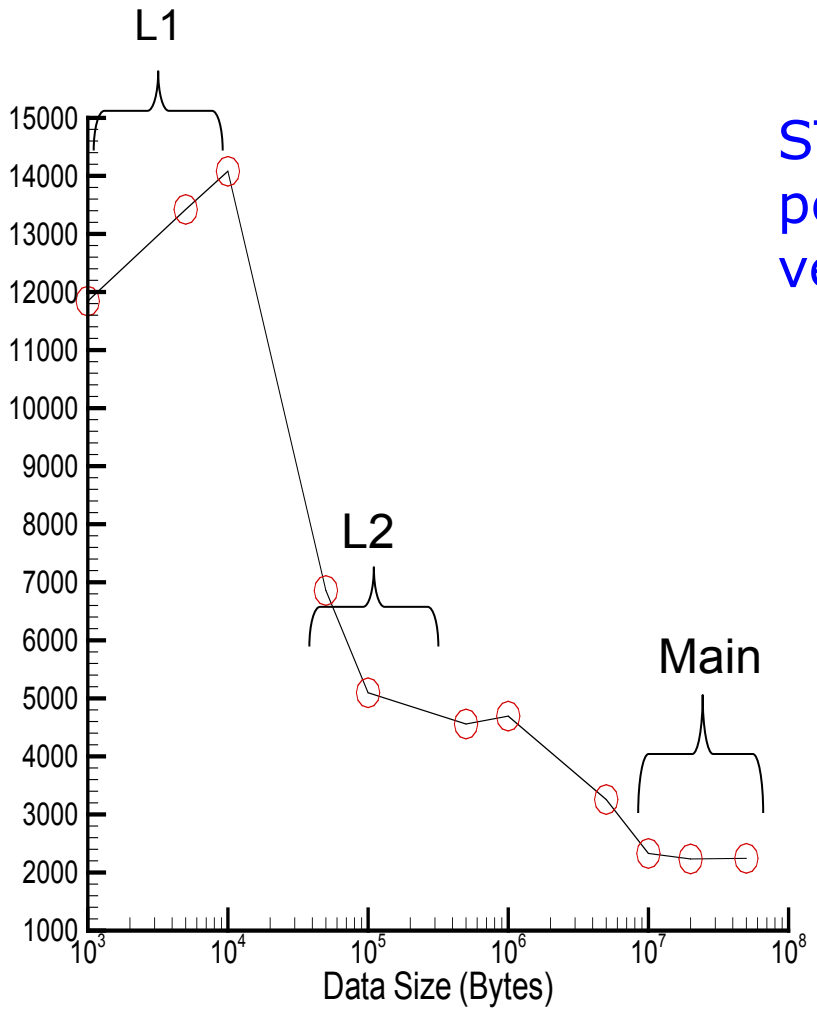


Memory Locality

- Typical access times for cache-based systems

Register	1 cycle	Ratio to previous
L1 Cache	Few cycles	1-3x
L2 Cache	~ 10 cycles	3-10x
Memory	~ 250 cycles	25x
Remote Memory	~2500-5000 cycles	10-20x

Impact of Memory Hierarchy



STREAM performance in MB/s versus data size

Revising the Performance Model

- Use the relevant sustained bandwidth
  - Obtain the STREAM (or equivalent) value for the slowest level of memory that will be used
  - Many problems fit into at least L3 cache

Updating the Sparse Matrix Example

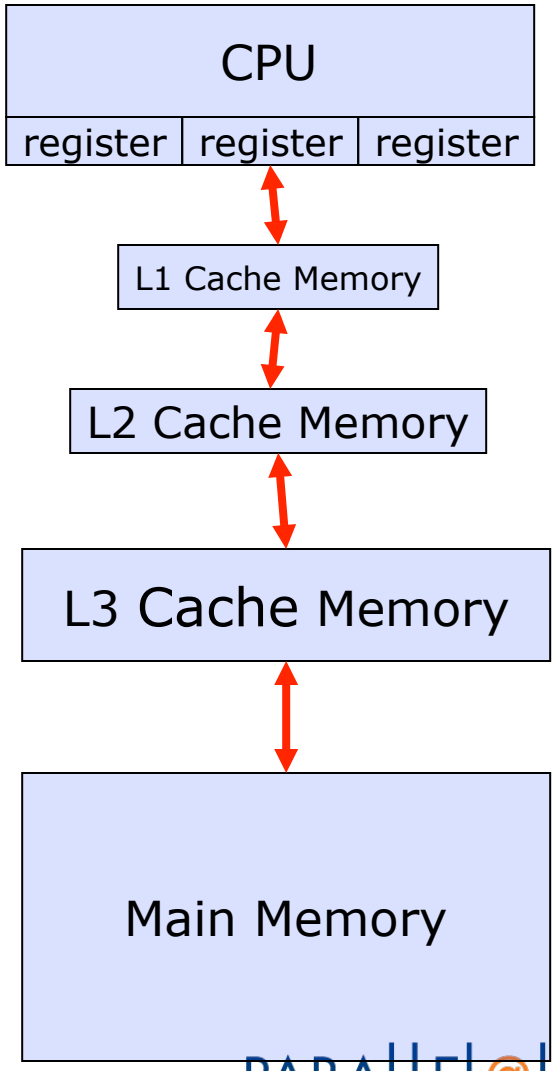
- Original (no cache)
  - Time = nnz(2c + 2.5r) + n(0.5r+w)
- Add in time to load x from Cache:
  - Time = nnz(2c + 1.5r + r_c) + n(1.5r +w)
- Where r_c is the time to read from (the appropriate level) of cache
  - Note that X is read into cache from memory (the 1 in the n(1.5r) term)

More on Cache Operation

- Most communication in a computer is carried out in chunks – blocks of bytes of data that move together
- In the memory hierarchy, data moves between memory and cache, and between different levels of cache, in groups called *lines*

Simplified Computer Architecture III

- Main memory contains the program data
- Multiple Cache memories contain a copy of the main memory data
- Data is moved between levels of memory in groups of words called lines
  - Lines are typically 64-128 bytes, or 8-16 double precision words
  - Even if you don't use the data, it is moved and occupies space in the cache
- Programming Language Relevance
  - This performance feature is not captured in most programming languages



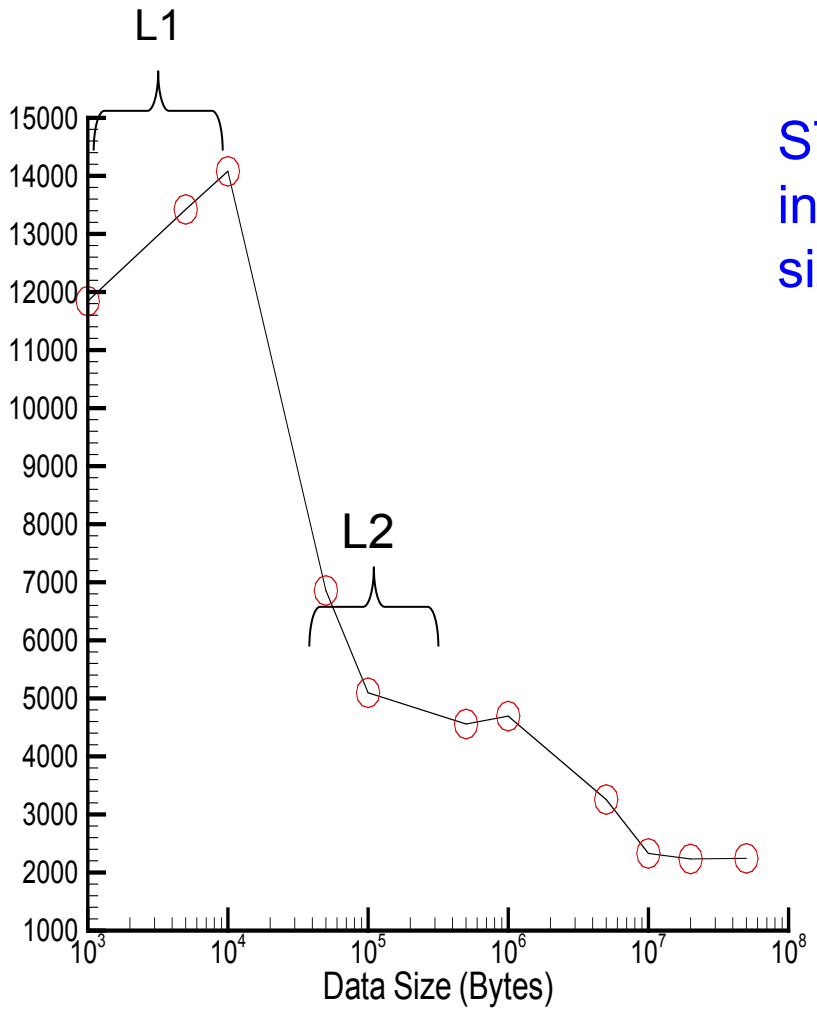
Types of Locality

- Temporal – reuse same data
  - This has been our assumption in the sparse matrix case – data is reused
- Spatial – (re)use "nearby" data
- These match the *hardware*
- Alpern and Carter suggest a different breakdown, based on needs of algorithms:
  - Local: Data has both Temporal and Spatial
  - Semilocal: Data has Spatial locality
  - Nonlocal: Data has neither spatial nor temporal locality

Refining the Bounds

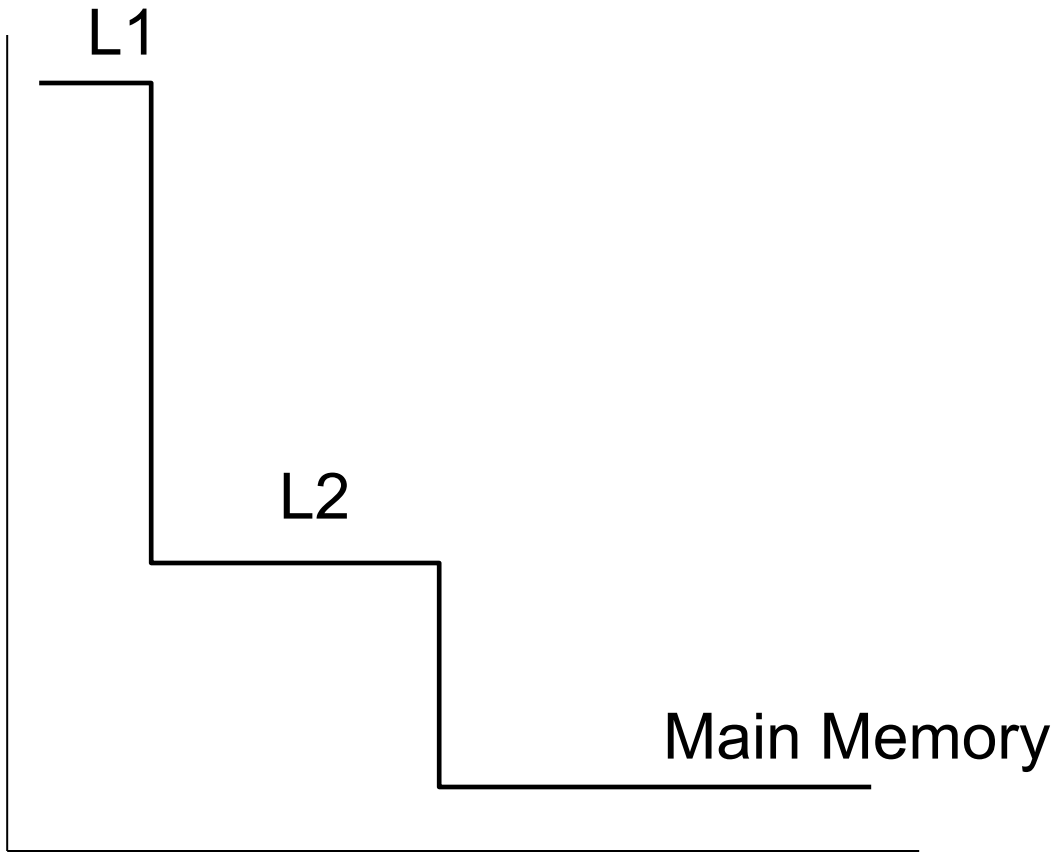
- Achievable memory bandwidth depends on small, fast, cache memory
  - *Temporal locality* reduces need for sustained memory bandwidth
    - Data is present in faster memory
  - Leads to an important rule-of-thumb in performance modeling
    - Identify dominant terms
    - Architectures are too complex for any other approach
    - Similar to approaches for mathematical modeling
  - "Does it fit in cache" provides a first model
    - Assume all of cache is available
    - For multiple cache levels, use performance of the smallest (fastest) cache in which the data fits

Impact of Memory Hierarchy: Measured



STREAM performance in MB/s versus data size

Modeled: Memory Bandwidth vs Data Size



Refining the Bounds: Spatial Locality

- Non-consecutive memory accesses expose more details about the memory structure
  - ♦ Effective cache size reduces when not all of the data on the same cache line is used
    - *Spatial Locality* important to get full use of cache
    - Cache line size helps in refining performance bounds
      - Reduce effective cache size to same fraction of cache line used.
      - E.g., if 1 8 byte value used from a 128-byte cache line, or 1/16 of the line, the effective cache size is 1/16 of the full size.