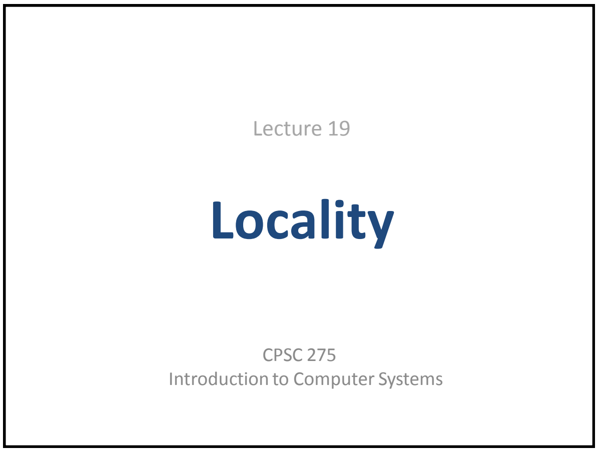
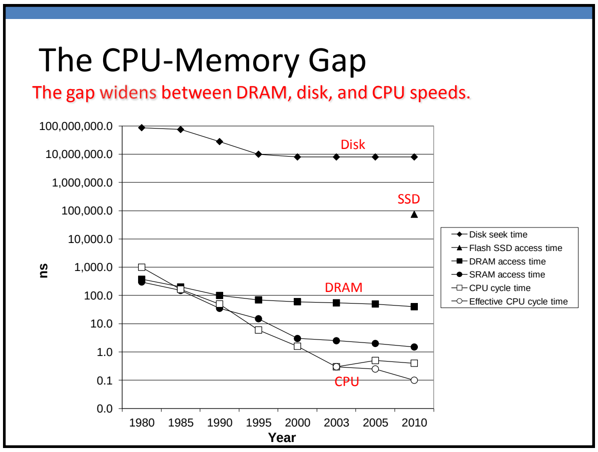


Lecture 19

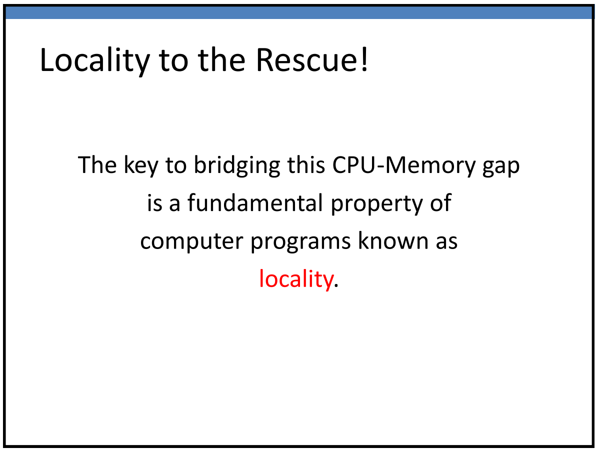
Locality

CPSC 275
Introduction to Computer Systems



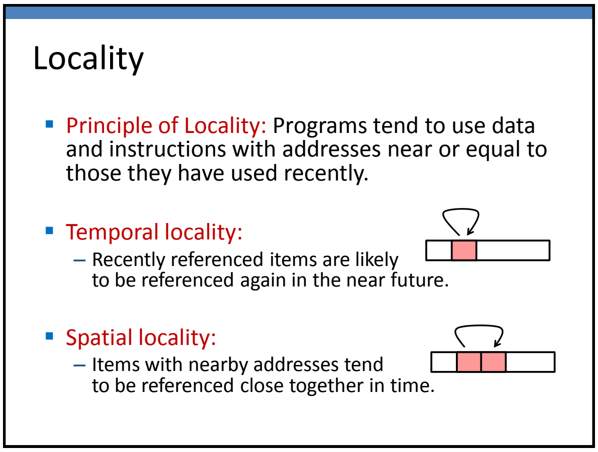
The CPU-Memory Gap

The gap widens between DRAM, disk, and CPU speeds.



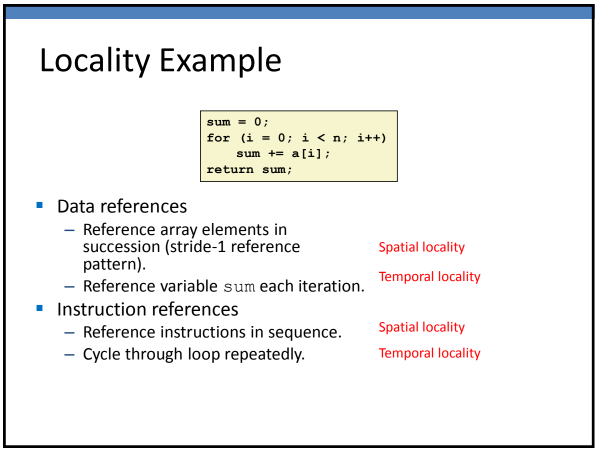
Locality to the Rescue!

The key to bridging this CPU-Memory gap is a fundamental property of computer programs known as locality.



Locality

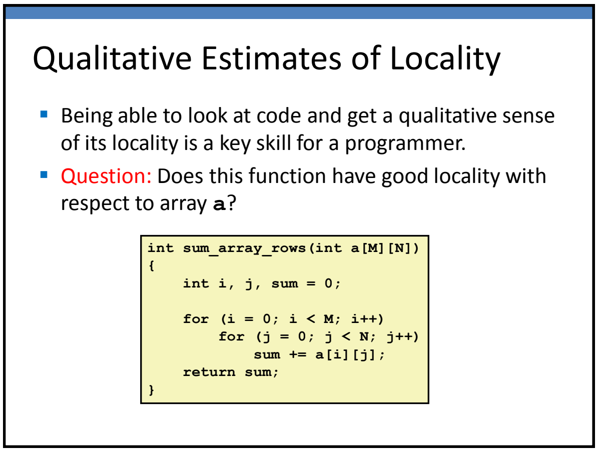
- Principle of Locality: Programs tend to use data and instructions with addresses near or equal to those they have used recently.
- Temporal locality:
  - Recently referenced items are likely to be referenced again in the near future.
- Spatial locality:
  - Items with nearby addresses tend to be referenced close together in time.



Locality Example

```
sum = 0;
for (i = 0; i < n; i++)
    sum += a[i];
return sum;
```

- Data references
  - Reference array elements in succession (stride-1 reference pattern). Spatial locality
  - Reference variable `sum` each iteration. Temporal locality
- Instruction references
  - Reference instructions in sequence. Spatial locality
  - Cycle through loop repeatedly. Temporal locality



Qualitative Estimates of Locality

- Being able to look at code and get a qualitative sense of its locality is a key skill for a programmer.
- Question: Does this function have good locality with respect to array **a**?

```
int sum_array_rows(int a[M][N])
{
    int i, j, sum = 0;

    for (i = 0; i < M; i++)
        for (j = 0; j < N; j++)
            sum += a[i][j];
    return sum;
}
```

Locality Example

- Question: Does this function have good locality with respect to array **a**?

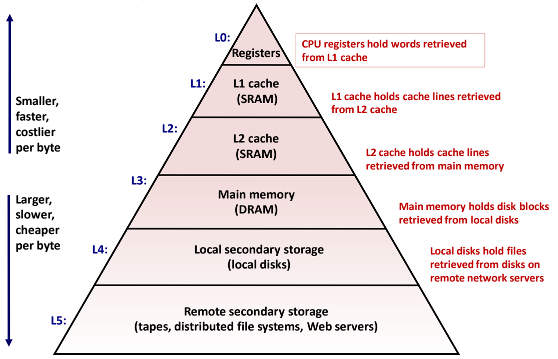
```
int sum_array_cols(int a[M][N])
{
    int i, j, sum = 0;

    for (j = 0; j < N; j++)
        for (i = 0; i < M; i++)
            sum += a[i][j];
    return sum;
}
```

Memory Hierarchies

- Some fundamental and enduring properties of hardware and software:
  - Fast storage technologies cost more per byte, have less capacity, and require more power.
  - The gap between CPU and main memory speed is widening.
  - Well-written programs tend to exhibit good locality.
- They suggest an approach for organizing memory and storage systems known as a memory hierarchy.

An Example Memory Hierarchy



Smaller, faster, costlier per byte

Larger, slower, cheaper per byte

L0: Registers — CPU registers hold words retrieved from L1 cache

L1: L1 cache (SRAM) — L1 cache holds cache lines retrieved from L2 cache

L2: L2 cache (SRAM) — L2 cache holds cache lines retrieved from main memory

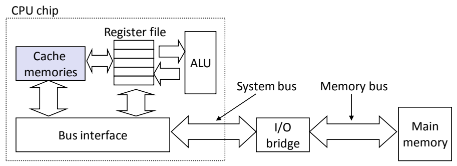
L3: Main memory (DRAM) — Main memory holds disk blocks retrieved from local disks

L4: Local secondary storage (local disks) — Local disks hold files retrieved from disks on remote network servers

L5: Remote secondary storage (tapes, distributed file systems, Web servers)

Caches

- *Cache:* A smaller, faster storage device that acts as a staging area for a subset of the data in a larger, slower device.



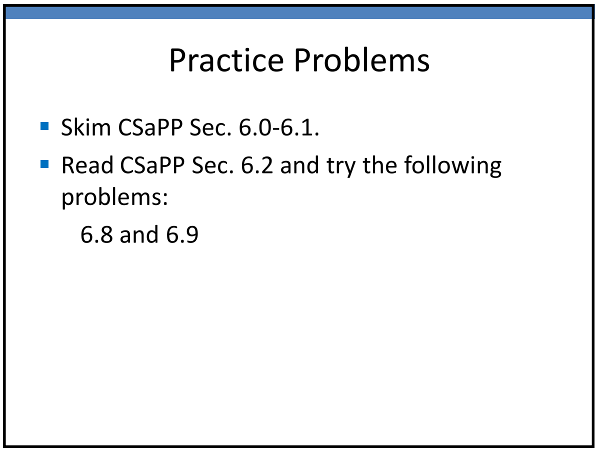
Caches, cont'd

- Fundamental idea of a memory hierarchy:
  - For each *k*, the faster, smaller device at level *k* serves as a cache for the larger, slower device at level *k*+1.
- Why do memory hierarchies work?
  - Because of locality, programs tend to access the data at level *k* more often than they access the data at level *k*+1.
  - Thus, the storage at level *k*+1 can be slower, and thus larger and cheaper per bit.

Big Idea

The memory hierarchy creates a large pool of storage that costs as much as the cheap storage near the bottom,

but that serves data to programs at the rate of the fast storage near the top.



Practice Problems

- Skim CSaPP Sec. 6.0-6.1.
- Read CSaPP Sec. 6.2 and try the following problems:

  6.8 and 6.9