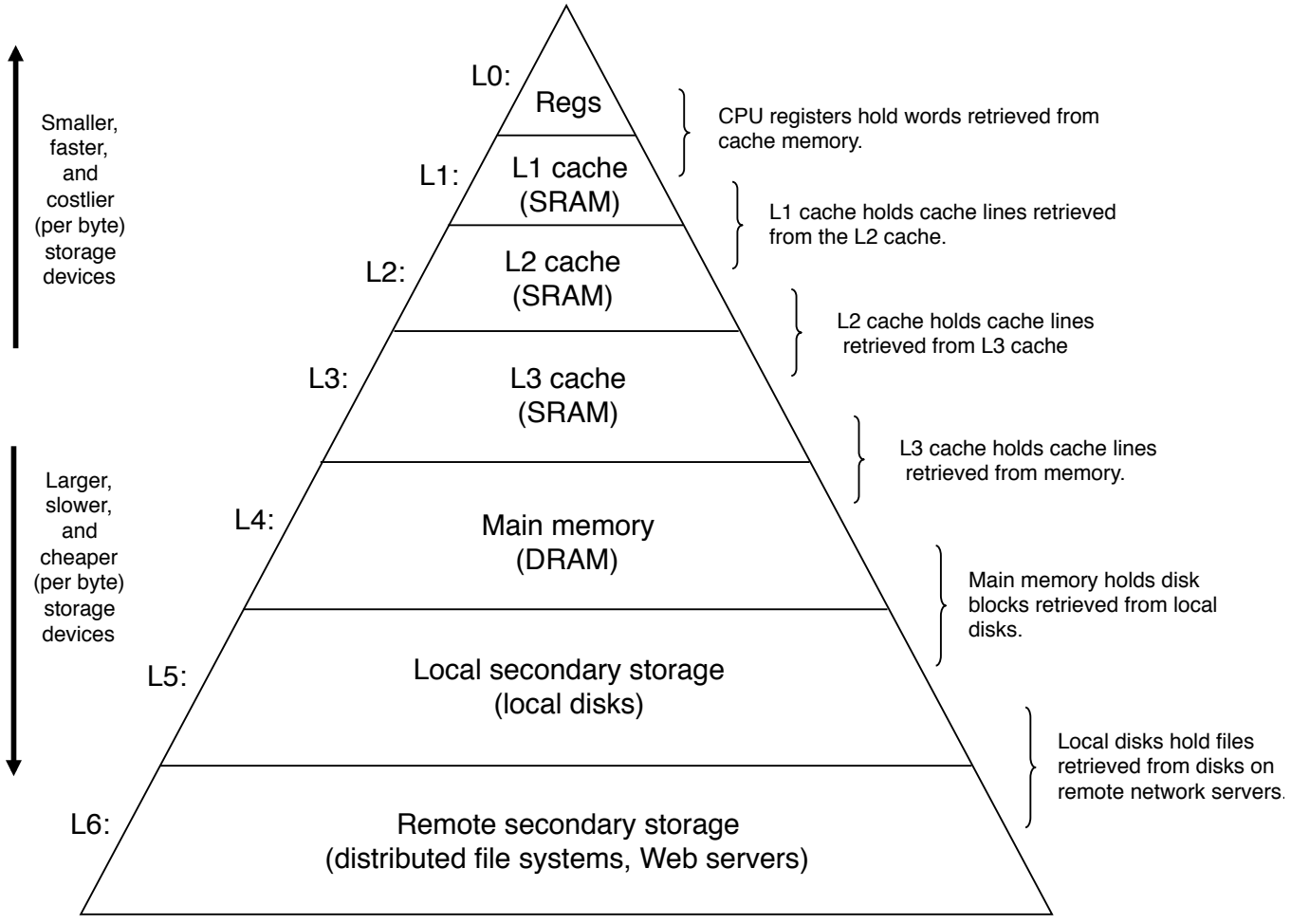
More Cacheing

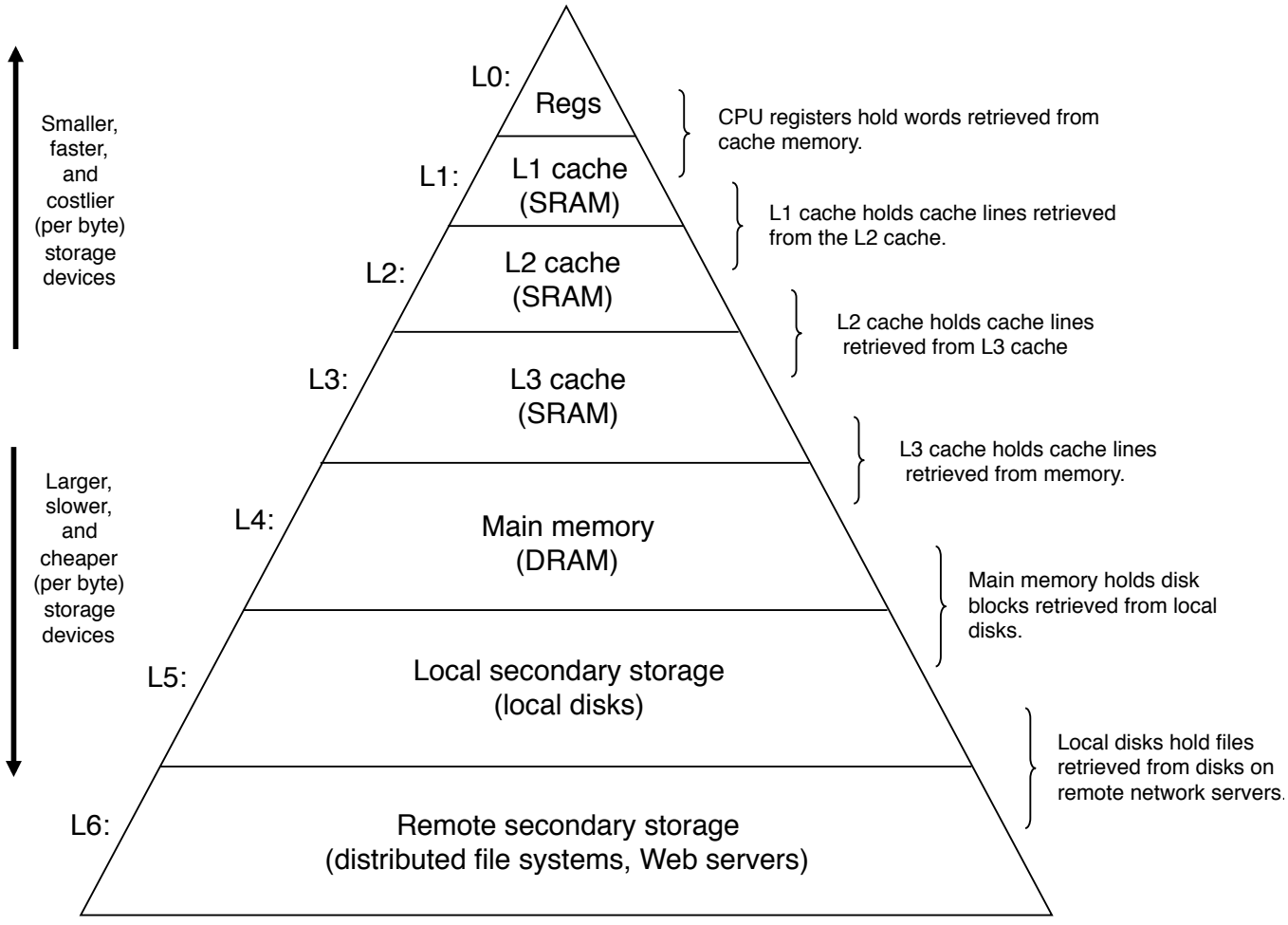
caches are everywhere

- Web caches
- Browser caches
- Google music app
- many other places within and between machines

hierarchy


Smaller, faster, and costlier (per byte) storage devices
Larger, slower, and cheaper (per byte) storage devices
L0:
Regs
L1:
L1 cache (SRAM)
L2:
L2 cache (SRAM)
L3:
L3 cache (SRAM)
L4:
Main memory (DRAM)
L5:
Local secondary storage (local disks)
L6:
Remote secondary storage (distributed file systems, Web servers)
CPU registers hold words retrieved from cache memory.
L1 cache holds cache lines retrieved from the L2 cache.
L2 cache holds cache lines retrieved from L3 cache
L3 cache holds cache lines retrieved from memory.
Main memory holds disk blocks retrieved from local disks.
Local disks hold files retrieved from disks on remote network servers.

hierarchy



- each layer cache for layer below it
- high cache hit rate $\rightarrow$ illusion of much larger, faster memory

how?

- store
  - what you've recently read
  - what you think you'll need soon
- when you read, load an entire *block*

array

some code

```
for (i=0; i<SIZE; i++)
    sum+=A[i]
```

what happens?

- `read A[0]` *miss*
- `load A[0]` and following items
- `read A[1]` *hit*
- `read A[2]` *hit*
- ...
- `read A[k]` *miss*
- `load A[k]` and following items
- `read A[1]` *hit*
- `read A[2]` *hit*

locality

some code

```
for (i=0; i<SIZE; i++)
    sum+=A[i]
```

i, sum good temporal locality

array elements bad temporal, good spatial

2D array. Difference in performance?

```
sum=0;
for (i=0; i<NROWS; i++)
    for (j=0; j<NCOLS; j++)
        sum+=A[i][j];
```

```
sum=0;
for (i=0; i<NCOLS; i++)
    for (j=0; j<NROWS; j++)
        sum+=A[j][i];
```

some math

Suppose that reading from the *slow* memory is 100x slower than the *fast* memory.

- a hit takes 1x time
- a miss is 100x time

some math

Average access time

$$\text{avg} = p(\textit{hit}) * \text{hit time} + p(\textit{miss}) * \text{miss time}$$

some math

Average access time

$$\text{avg} = p(\textit{hit}) * \text{hit time} + p(\textit{miss}) * \text{miss time}$$

Hit rate of 97%

$$\begin{aligned} \text{avg} &= (0.97) * 1 + (0.03) * (1 + 100) \\ &= 4 \text{ units} \end{aligned}$$

some math

Average access time

$$\text{avg} = p(\textit{hit}) * \text{hit time} + p(\textit{miss}) * \text{miss time}$$

Hit rate of 97%

$$\begin{aligned}\text{avg} &= (0.97) * 1 + (0.03) * (1 + 100) \\ &= 4 \text{ units}\end{aligned}$$

Hit rate of 99%

$$\begin{aligned}\text{avg} &= (0.99) * 1 + (0.01) * (1 + 100) \\ &= 2 \text{ units}\end{aligned}$$

some math

Average access time

$$\text{avg} = p(\textit{hit}) * \text{hit time} + p(\textit{miss}) * \text{miss time}$$

Hit rate of 97%

$$\begin{aligned}\text{avg} &= (0.97) * 1 + (0.03) * (1 + 100) \\ &= 4 \text{ units}\end{aligned}$$

Hit rate of 99%

$$\begin{aligned}\text{avg} &= (0.99) * 1 + (0.01) * (1 + 100) \\ &= 2 \text{ units}\end{aligned}$$

So if you increase the hit rate by 2%, the performance doubles.

Intel i7 Caches