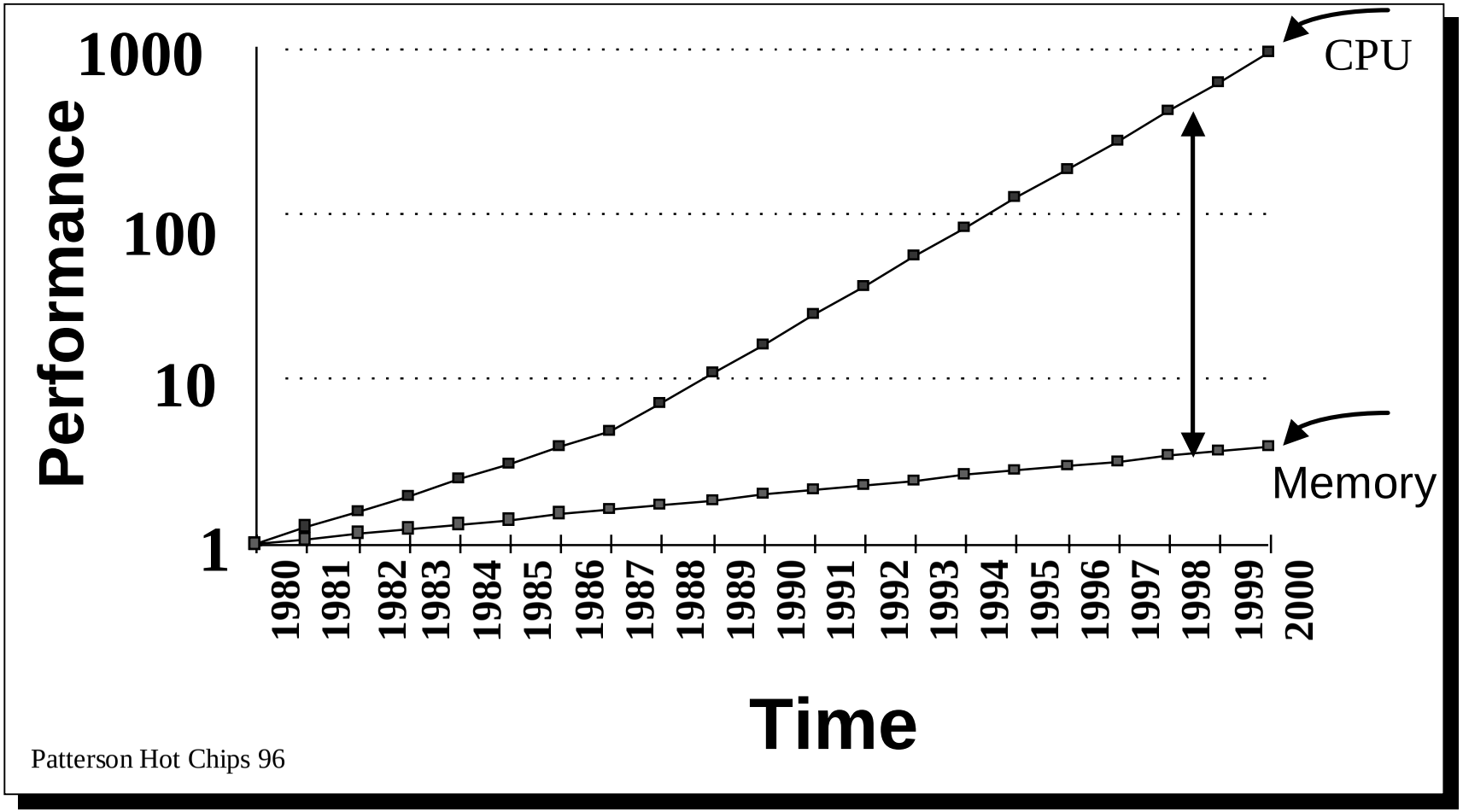
What is (or why have) memory?

- Memory is a state
- Caches intermediate computation
- Finite processing resources
- Pointers give you indirection
- Convenient

Processor-Memory Gap



What is the memory hierarchy and why?

- What
  - Registers
  - Cache(s)
  - Main memory
  - Disk
  - Tape?? / OceanStore
- Why?
  - Cost-performance (now)
  - Computation-Data locality

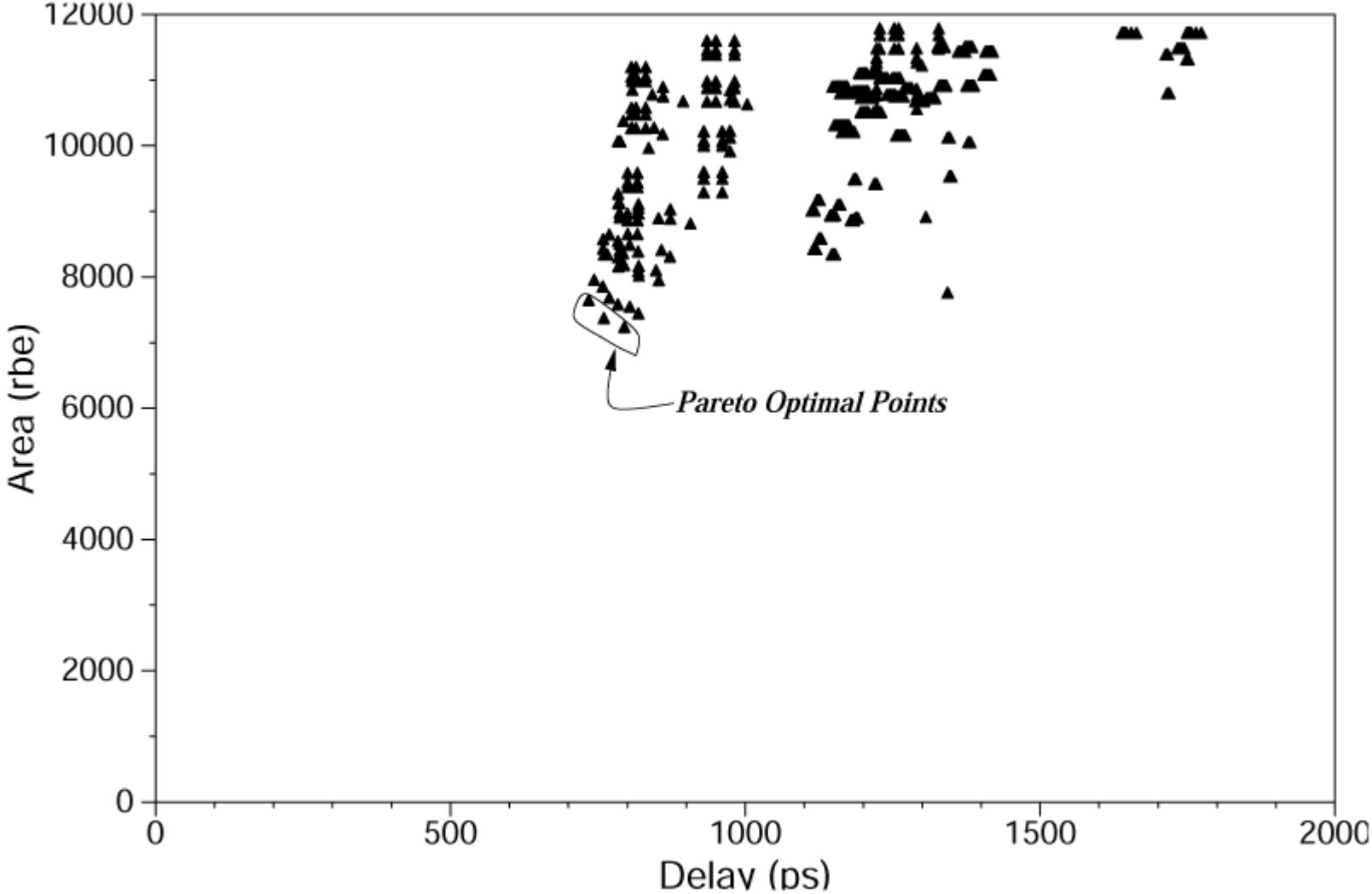
A Cache is a bet..

- What do you lose?
  - If you miss it costs you more
- And you win because
  - Locality

Stream buffers – implicit prefetching

- His baseline was direct mapped
  - Inflated results compared to today

Cache size/speed tradeoff



Where are we going from here?

- Problems
  - Processors still are faster to memory access
  - General laws of physics
    - Can’t just keep making caches larger
    - L2 caches are now the size of main memory
    - IBM L3 cache: 64 MB!
    - Diminishing returns to layering
- Ideas
  - Continuous layers?
  - Does RAM have to be R?
  - More specialized hardware for application-domain structure

(swiped from Luna)

Don't Need to Pay Much for Your Miss (II) -- Lockup-free Instruction Fetch/Prefetch

```
if (catch-hit)
    get-from-cache
else if (catch-miss){
    Judge the miss states from MSHR (in-input-stack indicator, partial write codes, valid indicator);
    if (totally written)
        read from cache;
    else if (in-input-stack)
        read from input stack;
    else if (partially written || already-asked-for)
        by-pass;
    else{
        initiate MSHR;
        when data available do 1, 2,  3 parallely; }
```

1. if (send-to-CPU)
       send to CPU;
2. if (!totally written || !MSHR obsolete){
       if (input-stack full)
           FIFO remove one;
       write to input-stack;
       set MSHR.in-input-stack;
   }
3. write to catch and set MSHR.partial-write-code
   if (written || obsolete MSHR)
       MSHR.num-of-words-processed++;
   if (MSHR.num-of-words processed overflow)
       clear MSHR.valid-indicator;