Ten Advanced Optimizations of Cache Performance

Can categorize the 10 advancements into 5 categories:

1. Reducing hit time: small and simple fast-level caches and way-prediction. Both techniques also generally decrease power consumption
2. Increasing cache bandwidth: pipelined caches, multibanked caches, and nonblocking caches. These techniques have varying impacts on power consumption
3. Reducing miss penalty: Critical word first and merging write buffers. These optimizations have little impact on power
4. Reducing miss rate: Compiler optimizations. Obviously any improvement at compile time improves power consumption.
5. Reducing the miss penalty or miss rate via parallelism: Hardware prefetching and compiler prefetching. These optimizations generally increase power consumption, primarily due to prefetched data that are unused.

First Optimization: Small and Simple First-Level Caches

For better clock cycles we can do two things to cache:

- Increase size of first-level cache
- More associativity

Higher associativity is used more than increasing cache size.

- Cost of higher associativity is either low or negligible
- Processors take at least two clock cycles to access the cache, and thus the impact of a longer hit time may not be critical
- Better avoids translation lookaside buffer (TLB) (keeping out of critical path) improves hit times
- With multithreading, conflict misses can increase, making higher associativity more attractive

Second Optimization: Way Prediction to Reduce Hit Time

Way prediction: extra bits are kept in the cache to predict the way, or block within the set of the next cache access.

- Prediction means the multiplexor is set early to select the desired block, and only a single tag comparison is performed that clock cycle in parallel with reading the cache data
- A miss results in checking the other block for matches in the next block cycle (extra 1 clock cycle)
- Simulation shows 90% accuracy for two way, 80% on 4-way, and better performance on I-caches than D-caches
- Way prediction yields lower average memory access time for a two-way associative cache

Way selection: Extend form of way prediction to be used to reduce power consumption

- Does so by using way prediction bit to decide which cache block to actually access (way bits are like extra address bits)
- This saves power when the prediction is correct, but adds significant time on a misprediction since the access, not just the tag match and selection, must be repeated
- This optimization only makes sense on lower-power processors
- Makes pipelining in cache access difficult

Third Optimization: Pipelined Cache Access to Increase Cache Bandwidth