### Question 1 - Direct-mapped Tag/Index/Offset Calculations

### As talked about in class, a cache has three primary configuration parameters:

### S: Cache size (how much data the cache holds)

### B: Block size (the granularity of the data)

### A: Associativity (number of blocks with the same index that can be in the cache at the same time).

### These parameters also specify how a memory reference address is split into its three components: block offset bits, index bits, and tag bits.

### Assuming 32-bit address, calculate the following values for a direct-mapped 32KB cache with 64B blocks (that is: S=32KB, B=64B, and A=1):

### The number of bits in the block offset

### The number of "sets" in the cache

### The number of "ways" in the cache

### The number of index bits

### The number of tag bits

Answer:

Assuming the memory is Byte Addressable, *all* ***log*** *used are base 2*

1. log(B) = log(64) = 6 bits
2. S/B\*A = 2^15 / 2^6 \* 1 = 2^9 = 512 Sets
3. No. of ways = A = 1
4. log(S/B\*A) = 9 bits
5. Total address size in bits -(index+offset bits) = 32-(9+6) = 17 bits

### Question 2 - Set-Associative Tag/Index/Offset Calculations

### For a two-way set associative version of the cache in question 1 (that is: S=32KB, B=64B, and A=2), calculate the following values:

### The number of bits in the block offset

### The number of "sets" in the cache

### The number of "ways" in the cache

### The number of index bits

### The number of tag bits

Answer:

1. log(B) = log(64) = 6 bits
2. S/B\*A = 2^15 / 2^6 \* 2 = 2^8 = 256 Sets
3. No. of ways = A = 2
4. log(S/B\*A) = 8 bits
5. Total address size in bits - (index+offset bits) = 32-(4+6) = 18 bits

## Question 3 - Generalizing the Calculations

Assuming 32-bit address, write formulas in terms of S, B, and A for calculating the following quantities. You may find using log(S), log(B), and log(A) in the formulas easier than S, B, and A directly. For this assignment, you may assume S, B, and A are all powers of 2. Write the formulas for:

1. The number of bits in the block offset
2. The number of "sets" in the cache
3. The number of "ways" in the cache
4. The number of index bits
5. The number of tag bits

Hint: you may find it helpful to first work out the numbers for a direct-mapped cache (that is, consider just Sand B), and then extend it to set-associative caches.

Answer:

1. number of bits in the block offset = log(B)
2. No. of Sets = S/B\*A
3. No. of ways = A
4. No of index bits = log(S/B\*A)
5. No of tag bits = Total address size in bits - (index+offset bits)

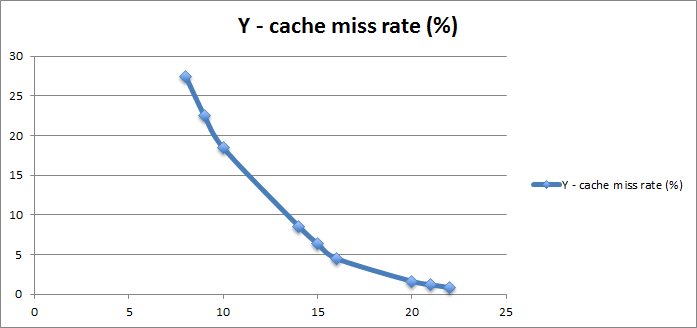
## Question 4 - Miss Rate vs Cache Size

1. How large must the cache be for the miss rate to be less than 10%? How large to be less than 5%?
2. Today's processors generally have 32KB or 64KB first-level data caches. The best improvement you would expect in general from doubling the size of the cache would be a 2x reduction in misses (for example, reducing the miss rate from 20% to 10%). By what ratio does increasing the cache size from 32KB to 64KB reduce the miss rate? (2.0 would be halving the miss rate; 1.0 would be no change in miss rate; less than 1.0 would be an increase in misses).

Data Obtained:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cache Size (byte) | X - Log of cache size | Y - cache miss rate (%) | total memory accesses | hit | miss |
| 256 | 8 | 27.4989944 | 49642128 | 35991042 | 13651086 |
| 512 | 9 | 22.50601948 | 49642128 | 38469661 | 11172467 |
| 1024 | 10 | 18.45224282 | 49642128 | 40482042 | 9160086 |
| 16 \* 1024 | 14 | 8.551404162 | 49642128 | 45397029 | 4245099 |
| 32 \* 1024 = 32kb | 15 | 6.352133011 | 49642128 | 46488794 | 3153334 |
| 64 \* 1024 = 64 kb | 16 | 4.502721962 | 49642128 | 47406881 | 2235247 |
| 1024 \* 1024 = 1 MB | 20 | 1.625133798 | 49642128 | 48835377 | 806751 |
| 2\* 1024 \* 1024 = 2 MB | 21 | 1.199763233 | 49642128 | 49046540 | 595588 |
| 4\* 1024 \* 1024 = 4 MB | 22 | 0.827613595 | 49642128 | 49231283 | 410845 |

Graph:



|  |  |  |  |
| --- | --- | --- | --- |
| Answer - Part 1: |  |  |  |
| The cache size should be **more than 8Kb** for the miss rate to be **less than** **10%** | | | |
| The cache size should be **64Kb or more** for the miss rate to be **less than 5%** | | | |
| Answer - Part 2: |  |  |  |
| The **miss rate** would be **reduced by** the following ratio: | | | **1.410731789** |

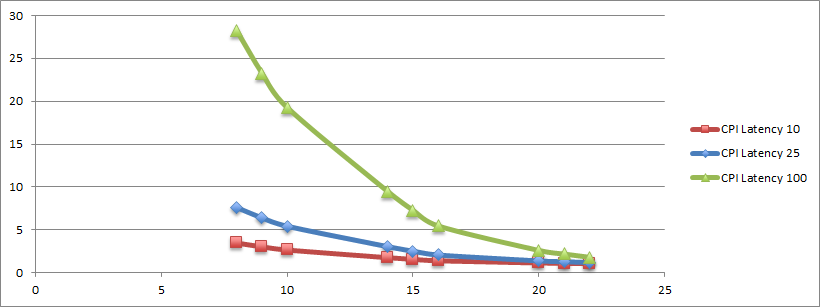
## Question 5 - Performance Impact of Cache Size

1. For each of the cache miss latency, how large must the cache be to achieve a CPI of 2? What does this say about the impact on cache miss latency?
2. In the previous question you calculated the ratio of miss rates for increasing the cache size from 32KB to 64KB. What is the speedup that results from increasing the cache size from 32KB to 64KB (this is basically a "speedup" calculation in which 1.0 is no speedup, less than 1.0 is a slowdown, and greater than 1.0 is a speedup). Calculate the speedup for each of the cache miss latencies.
3. How do these speedup numbers compare to the cache miss ratio you calculated in part b of the previous question?

Answer:

Data Obtained:

|  |  |  |
| --- | --- | --- |
| CPI Latency 10 | CPI Latency 25 | CPI Latency 100 |
| 3.474909496 | 7.599758657 | 28.22400446 |
| 3.025541754 | 6.401444676 | 23.28095929 |
| 2.660701854 | 5.428538277 | 19.26772039 |
| 1.769626375 | 3.052336999 | 9.465890121 |
| 1.571691971 | 2.524511923 | 7.288611681 |
| 1.405244977 | 2.080653271 | 5.457694743 |
| 1.146262042 | 1.390032111 | 2.60888246 |
| 1.107978691 | 1.287943176 | 2.187765601 |
| 1.074485224 | 1.198627263 | 1.819337459 |



Part 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Speed up from 32 KB to 64 KB | Latency 10 | Latency 25 | Latency 100 |
|  | 1.118446959 | 1.213326583 | 1.335474413 |

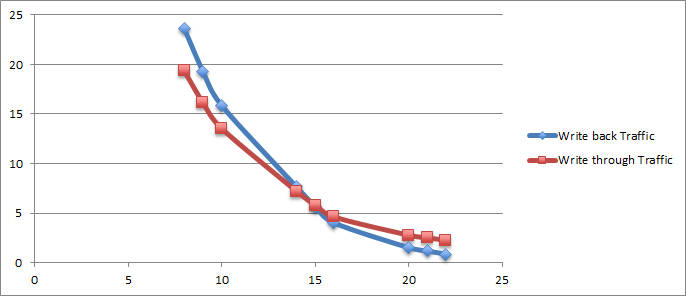
## Question 6 - Traffic of Write-Back vs Write-Through Caches

1. At what cache size do the two write policies generate approximately the same amount of traffic?
2. Why does the difference between the two schemes diverge at large cache sizes?
3. Why does the difference between the two schemes diverge at small cache sizes?

Answer:

Data too large to fit here.The data can be found in that attached xlsx sheet.

Graph obtained from the data:



1. At 32 kb, the two write policies generate approx same amount of traffic
2. Number of dirty evictions decrease with greater cache sizes, thus decreasing the out traffic for write back policy, in comparison to write through this out traffic metric remains same irrespective of the size of cache, which is the reason why the two schemes diverge
3. At smaller cache size, there are a large number of dirty evictions, each eviction increases the traffic by a factor of 64(block size), therefore write back generates more traffic than write through at smaller cache sizes

## Question 7 - Set-Associative Caches

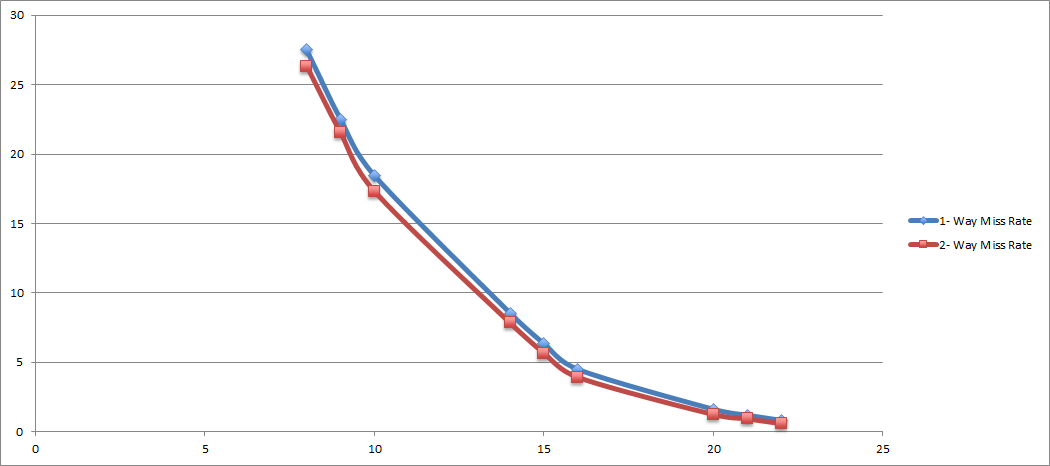
1. How large must the set-associative cache be for the miss rate to be less than 10%? How large to be less than 5%?
2. How large must the direct-mapped cache be before it equals or exceeds the performance of the 16KB two-way set-associative cache (ignore any noise in which the direct-mapped cache might slightly outperform the set-associative cache for some configurations)?
3. This graph shows that the performance gap between set-associative and direct-mapped caches narrows as the cache size grows very large. Explain what likely causes this narrowing.

Answer:

Data obtained:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Cache Size (byte) | X - Log of cache size | 1- Way Miss Rate | total memory accesses | hit | miss | Hit - 2Way | Miss2-way | 2- Way Miss Rate |
| 256 | 8 | 27.4989944 | 49642128 | 35991042 | 13651086 | 36567016 | 13075112 | 26.33874197 |
| 512 | 9 | 22.50601948 | 49642128 | 38469661 | 11172467 | 38951479 | 10690649 | 21.5354366 |
| 1024 | 10 | 18.45224282 | 49642128 | 40482042 | 9160086 | 41040904 | 8601224 | 17.32646111 |
| 16 \* 1024 | 14 | 8.551404162 | 49642128 | 45397029 | 4245099 | 45750288 | 3891840 | 7.839792847 |
| 32 \* 1024 = 32kb | 15 | 6.352133011 | 49642128 | 46488794 | 3153334 | 46826794 | 2815334 | 5.671259701 |
| 64 \* 1024 = 64 kb | 16 | 4.502721962 | 49642128 | 47406881 | 2235247 | 47692004 | 1950124 | 3.928365037 |
| 1024 \* 1024 = 1 MB | 20 | 1.625133798 | 49642128 | 48835377 | 806751 | 49021494 | 620634 | 1.250216349 |
| 2\* 1024 \* 1024 = 2 MB | 21 | 1.199763233 | 49642128 | 49046540 | 595588 | 49177236 | 464892 | 0.936486848 |
| 4\* 1024 \* 1024 = 4 MB | 22 | 0.827613595 | 49642128 | 49231283 | 410845 | 49349275 | 292853 | 0.589928377 |

Graph:



Answer:

1. For lesser than 10%, the cache should be 16 Kb. For lesser than 5%, cache should be 64 Kb
2. 4 Mb
3. As the cache size increases, the no. of blocks increases. This causes an improvement in the hit rate in both cases ( direct mapped cache as well as in set associative cache) since now more number of memory addresses can be kept in the cache it does not matter how they are being stored in 1-way (direct mapped cache) or 2-way cache as far as miss rates are concerned, although this does impact the look up time of these addresses in case of set associative cache. As there would be several ways to look at, on the same index before the address is found.

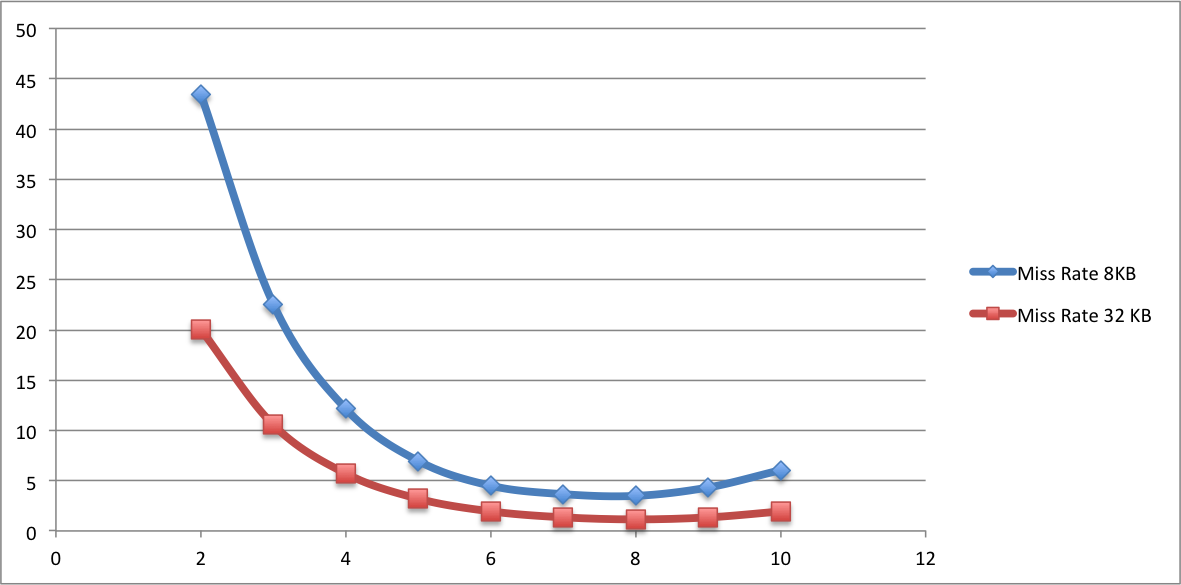
Bonus Questions :

Q : 8

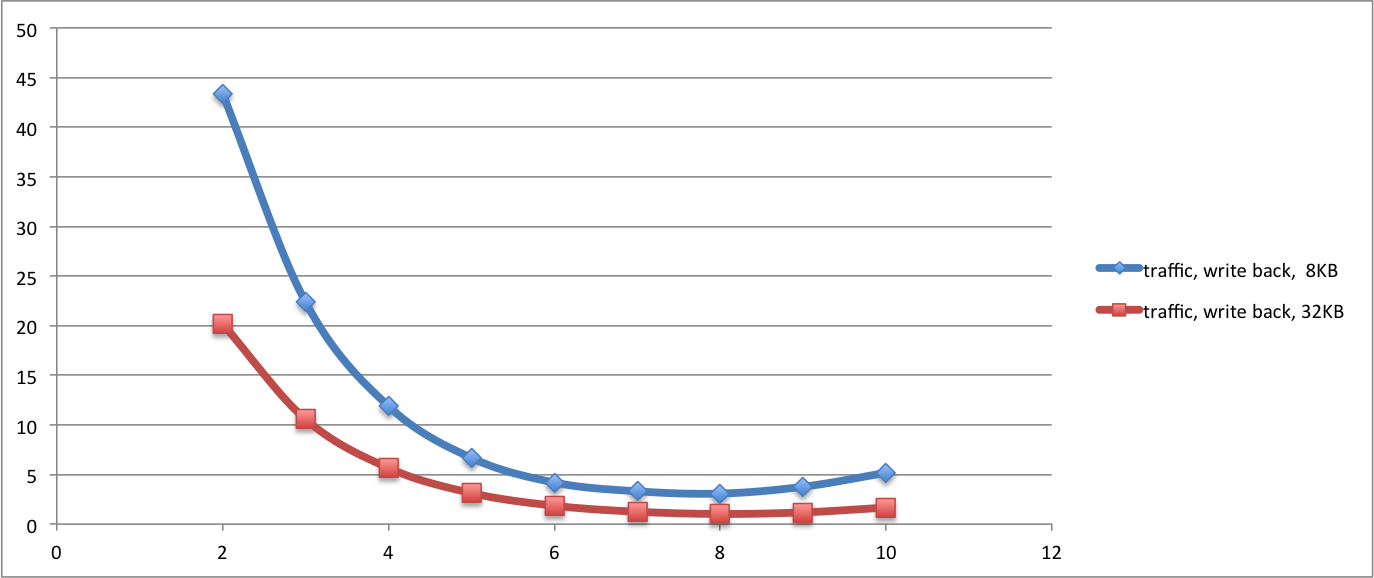
1. What is the block size with the lowest **miss rate** for the 8KB cache? And the 32KB cache?
2. What is the block size with the lowest *traffic* (bytes per memory reference) for the 8KB cache? And the 32KB cache?
3. What are the two (or three) sources of additional traffic as the block size grows? Explain why each component grows.
4. Given that current processors typically use, say, 64B blocks, which metric (miss rate or traffic) are today's caches designed to minimize?

Graphs :

Graph 1 : Miss rate of 2-way set associative cache with varying block sizes



Graph 2 : Traffic per memory operation for write-back cache

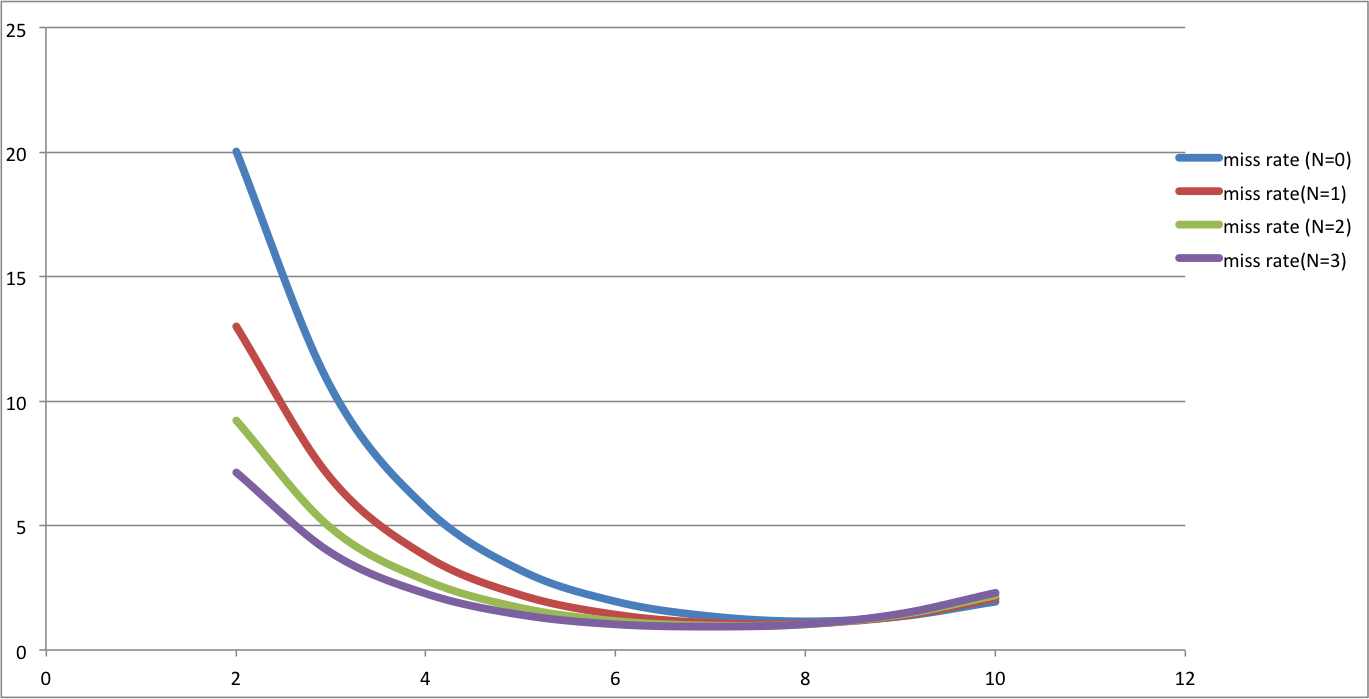


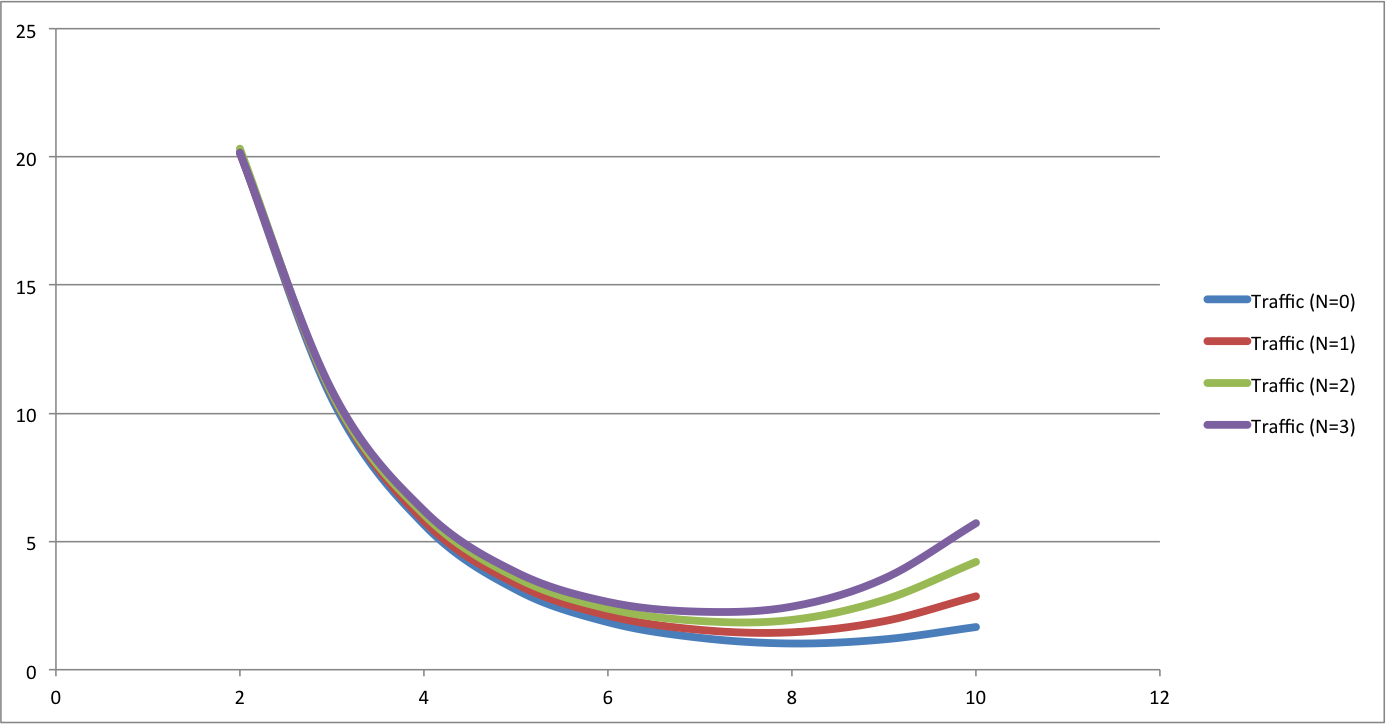
Answer :

1. The block size of lowest miss rate for both 8 Kb and 32 Kb is 256 Bytes.
2. The block size of lowest traffic for both 8 Kb and 32 Kb is 256 Bytes.
3. When the block size gets too big, every eviction of a dirty block causes additional traffic in writing back a bigger chunk of data.
4. Today’s processors are designed to minimize traffic.

Q9:

Answer:

Graphs :



1. For the 32KB cache with a 64B cache block size, which of these four values of *N* has the lowest miss rate?

* N=3 has the lowest miss rate , 1.03.

1. The main cost of prefetching is the additional traffic it generates. For the *N* above with the lowest miss rate, how much extra *traffic* does the prefetching introduce over the N=0 baseline? Give your answer as a percent increase (100% increase would be double the traffic).

* There is a 24% increase in traffic at N = 3 as compared to N=0.

1. How does prefetching impact the choice of block size? For each N, what is the block size with the lowest miss rate?

* for N=0 , Lowest Miss rate is for 256 B block size
* for N=1 , Lowest Miss rate is for 256 B block size
* for N=2 , Lowest Miss rate is for 128 B block size
* for N=3 , Lowest Miss rate is for 128 B block size

1. What combination of block size and prefetch *N* gives the lowest overall miss rate? Is this block size large, smaller, or the same as the best block size without prefetching (N=0).

- N=2 at 128 byte block size gives the lowest overall miss rate. This is smaller than lowest miss rate without prefetching , which is 256 bytes block size.

1. In the above description, we specified that prefetched blocks are put into the cache marked as least recently used (LRU) and not most recently used (MRU). Why did we make this choice? Would the prefetching be more or less effective if blocks were prefetched into MRU? (Feel free to change this policy and see how the results change, but it isn't required to answer the question). Explain.

-If prefetched blocks are marked as MRU , then it would be less effective as we are replacing the block is proven to be most recently used with a block we suppose could be used next. Setting the prefetched block as LRU, brings the block into cache and also gives us the freedom to evict it in case a new block needs to take its place. We cannot ignore temporal locality in order to accommodate for spatial locality.