A good cache should be fast and cost effective. We measure how fast a cache is by its CPI, or Cycles per Instruction, and we measure its cost effectiveness by comparing how much it costs compared to how much space is wasted. Together a good cache should cost as little as possible, while maintaining a good standard. We have compiled a report analyzing the effects of different types of caches on 3 separate trace files, but unless stated otherwise the majority of data will come from “A-9\_new\_1.5.pdf.trc” with the other two files used as margin of errors and comparisons.

Intuitively it might be thought that the more expensive caches are worth their cost because they’re more efficient but looking at Figures 1 and 2 we see that is not the case. While cost rises exponentially, CPI is relatively the same for cache sizes 64 KB to 8 MB. Even 32 KB isn’t a huge difference with one run recording a CPI of 4.6723. When compared to 4.5521 as the best recorded CPI in our dataset, or 4.9258 as a recorded worst for an 8 MB cache, 32 KB is still a completely viable option if you know that your average data limits would fit inside.

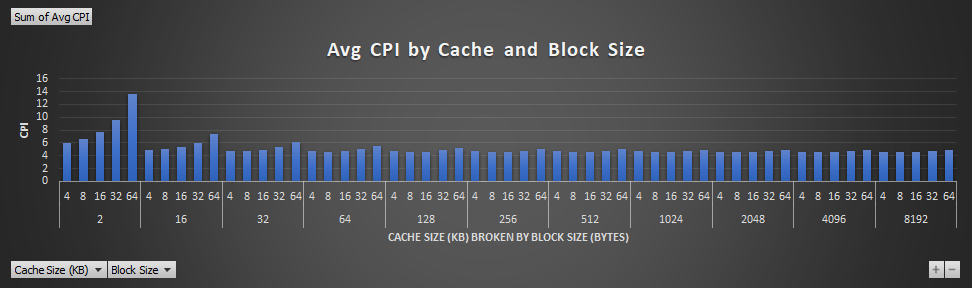
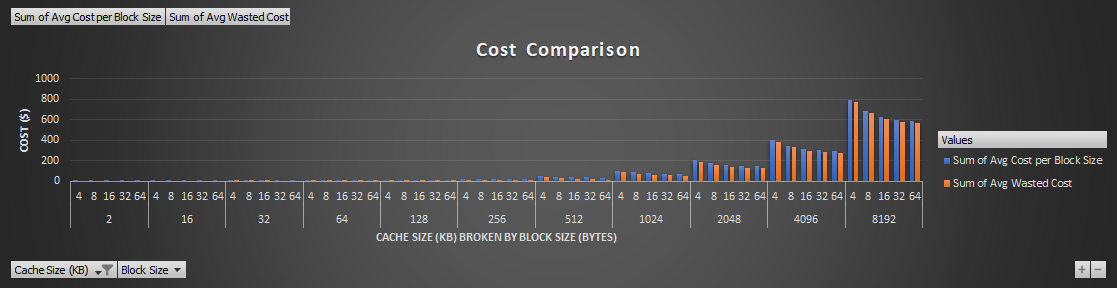
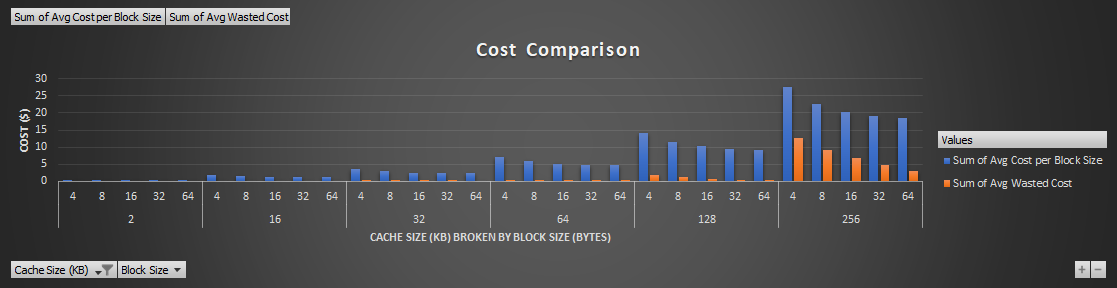
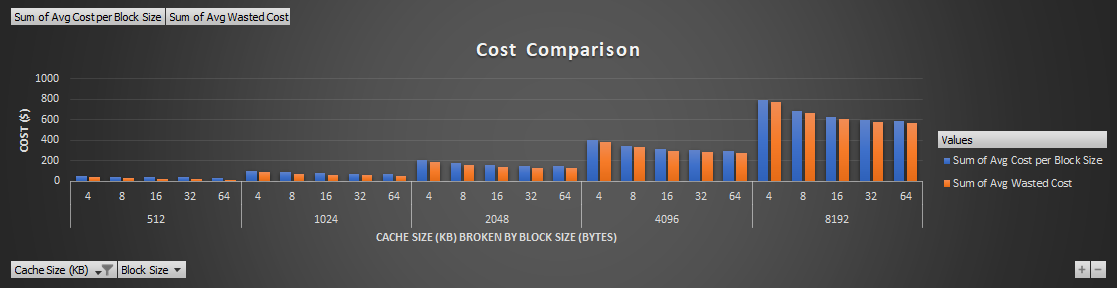


Figure 1

Figure 2

Not only do the larger caches not make a comparative difference in CPI but they’re not being used to their full potential either. Figure 3 shows the average wasted cost per block and cache size in comparison to its average cost. As the cache size grows the difference between how much the cache cost and how much is wasted diminishes. Another way to look at it is the unused block percentage. 128 KB cache can range from 0-30% wasted space, while just going up to 512 KB has a range from 41-73% and 8 MB caches ranging 96-98% wasted space. That space wouldn’t be wasted with more data references to store though. Knowing your data needs is the best way to guide you on what size cache to buy.

Figure 3 (please take note of the changing cost values on the y axis)



The difference between 128 KB cache and 256 KB cache is mostly cost related. On average their CPI’s are very similar, both with a low around 4.6 cycles per instruction. The 128 KB has a high of 5.2 while the 256 KB has a high of 5.01, but [insert error range from looking at other data]. Both have an average miss rate of (128: 1.67 - 8.58, 256: 1.34-8.72). Cost ranges (128: 9.31-13.72 and 256: 18.59-26.32). From this, unless your data needs are very large, a 128 KB cache is perfect. If you need something cheaper the 64 KB cache is comparatively good as well but with the cache being near capacity from our testing, I think spending a few extra dollars to get a little bit more leeway for memory would be ideal. The caveat being if you’re buying cache in bulk for many computers at once. If these are okay with a few hiccups and aren’t top speed performance sensitive, the few dollars being saved can save you more in the long run. See Figure 4 for graphs comparing the 128 KB and 256 KB cache.

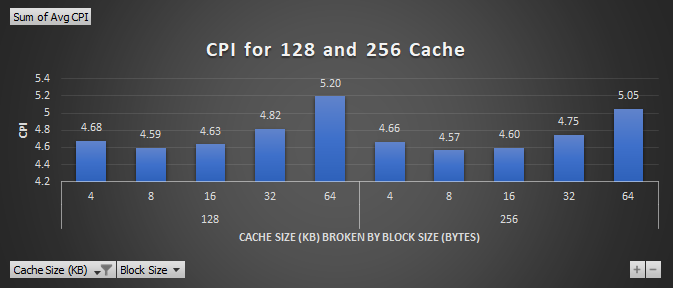
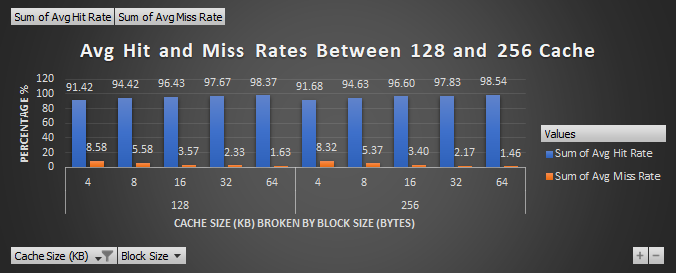
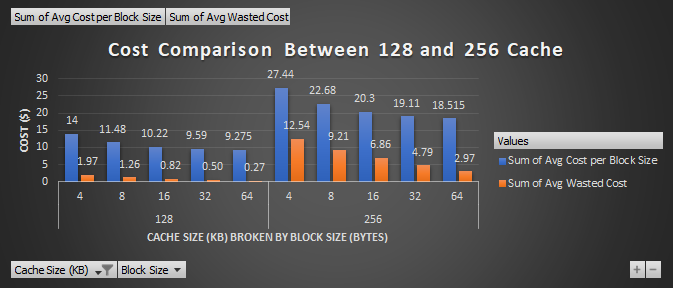
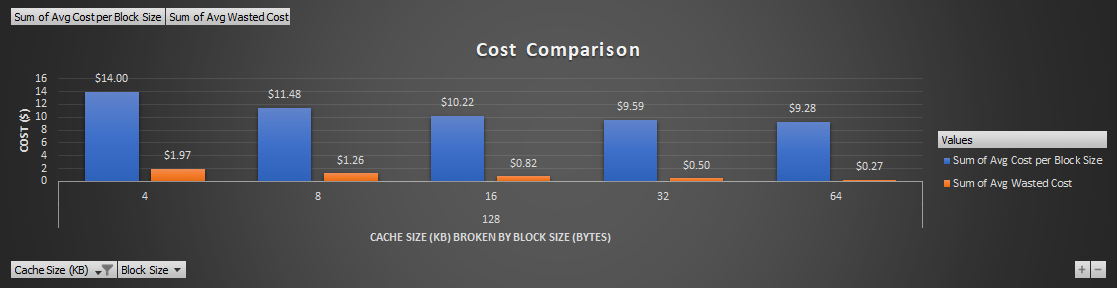
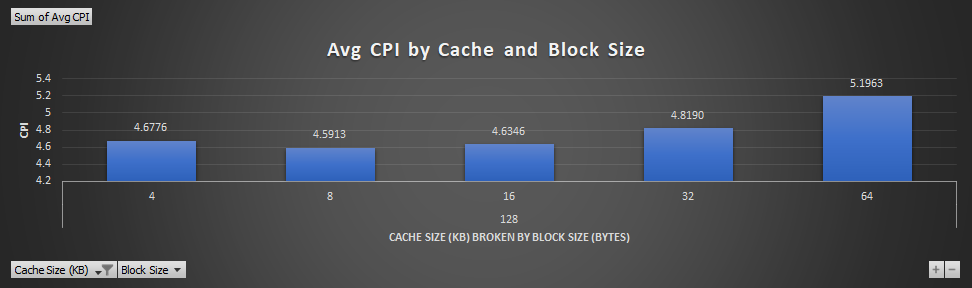
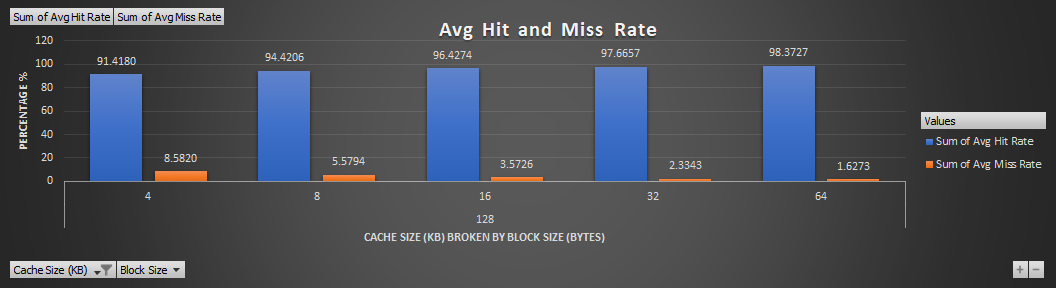


Figure 4

Once you have an idea for your data size, breaking it down to block size and associativity is where knowing how much you want to spend/save vs hit rate vs CPI comes into play. Figure 5 shows a 128 KB cache and how its cost, CPI, and average hit and miss rates are broken down by block size. While this is a specific cache size, the trends will mirror most caches (cache’s below 32 KB tend to have their lowest CPI at a block size of 4 instead of 8). Going from 4- to 6-byte blocks will see an increase in hit rate and a decrease for cost no matter the cache size. If you’re looking for the cheapest one, the 64 byte block has the lowest cost, lowest wasted cost percentage wise, and also has the lowest miss rate. That is counteracted by having the highest CPI. While it wouldn’t miss as often, the larger block size means there are more data reads and writes per block causing the CPI to escalate. We would recommend if you’re looking for a good average a 16 block size would be the best. It’s the middle of the cost, CPI, and is comparative to the larger block size for hit and miss rate. From least to greatest CPI it will normally go, 8, 16, 4, 32, 64 byte blocks. Looking at the other two trace files though, for “A-10\_new\_1.5\_a.pdf.trc” it goes 16, 8, 32, 4, 64-byte blocks and the “Corruption2.trc” trace file actually has 4 byte blocks worse than 64-byte blocks starting at 512 KB sized caches. From this if you want the best CPI and know you’re going above 32 KB for your cache size, we recommend staying either at 8- or 16-byte blocks. We say 16 is a good average size between the two extremes, it tends to be either the best or second-best CPI in a block size, as well having a better hit rate than an 8-byte block. Overall, knowing the kind of data you’re primarily using the cache for is key. If the data is generally small, smaller block sizes would be perfect, but if you’re constantly referencing large chunks of continuous data from the cache, putting them in bigger block sizes might be more suitable. They’re cheaper, but the size makes the CPI slightly slower.

Figure 4



For associativity there’s a downward trend in CPI and miss rate for the higher the association size (for cache sizes greater than 8 KB), but the higher the size the more it costs. Depending on the block size is also a factor, for 128 KB Cache (see Figure 5 for a visual breakdown), a smaller 2-byte block size will be a difference of a $1 between the lowest and highest cost between associations while the 64-byte block will give a range around 7 cents.

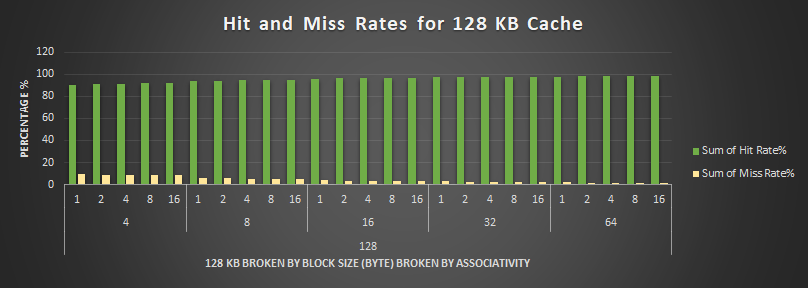
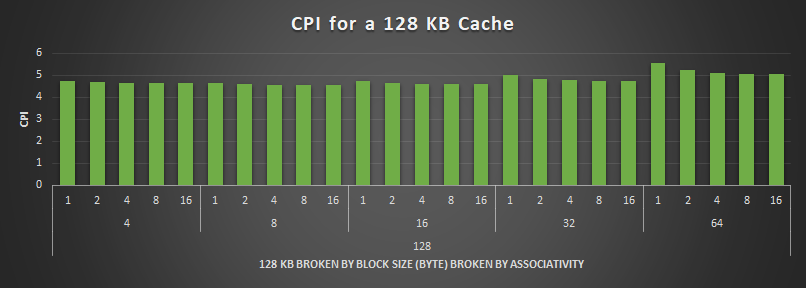
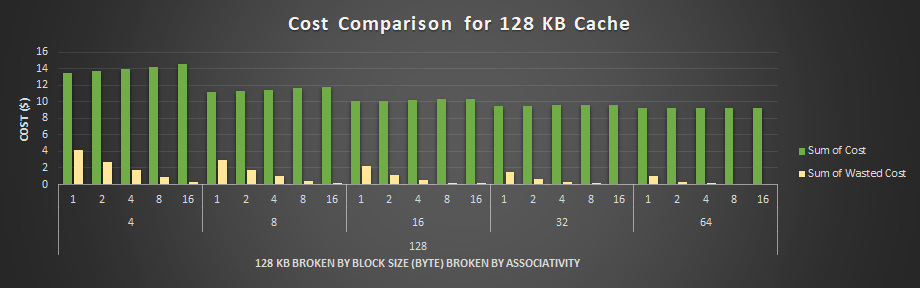


Figure 6

The difference gets higher the higher the cache size as well, with the largest cache having a difference of $72 between associations for its small block. But there’s also a downward trend to wasted cost for the higher the association per block, meaning that at least more blocks are being used. The CPI difference between association also increases as block size goes up, but .06 was the worst difference between associations inside a single block size (starting at 128 KB, it gets lower as cache size goes up, with 8 MB having .03 cycles per instruction as the largest difference between associations). With the cost range being so minimal between 2-16 way associations, getting the best CPI and miss rate at a 16 way association is recommended, though an 8 way usually has a difference of only a few thousandth of a CPI and could save you a few pennies, or go towards a difference of only a few hundredth of a cycle and go to a 4 way. Any lower and you risk getting a difference of a few tenths of a CPI though. The direct association seems to be the worst CPI and miss rate, and if you have lots of large data that needs to be stored in order it’ll definitely not be worth the savings in cost.

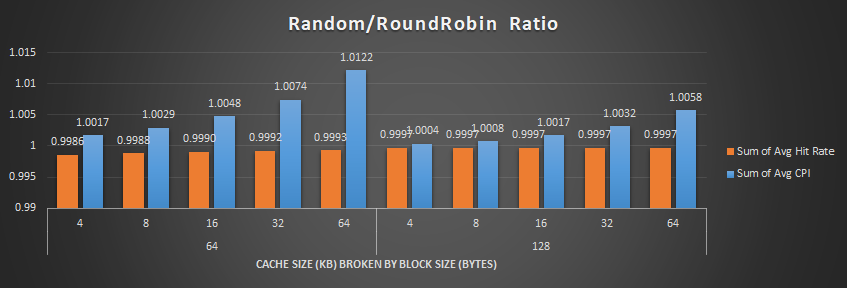
 Finally, for the replacement policy we looked at the “Corruption2.trc” data for a Random policy and a Round Robin Policy. We created a ratio where RND/RR where 1 is no difference, <1 means RR was a bigger number, and >1 RND was the bigger number. In Figure 7 we had to show up to 4 decimal places to show that it’s not truly a perfect 1 or 1.01 (in 3 cases) for each comparison. The average hit rate gets mildly better if you use a Random replacement policy but Round Robin has a slightly better CPI. From this data we don’t believe that either replacement policy produces a significant enough difference to give a recommendation either way.

Figure 6

Overall if you’re looking for the best CPI, the data shows block sizes of 8 bytes tend to be the most favorable when looking at the “A-9\_new\_1.5.pdf.trc” data as a whole, with associations of 16 and 8 being comparative. If you’re looking for a cache size that has the least amount of wasted spaced, but still has just enough to grow into, a 128 KB cache with an 8 block and 16-way association will cost you $11.76 with a CPI of 4.5605. Best performance but you want to save a little more is a 64 KB cache, with an 8 block and 16-way association that will cost $5.95 and a CPI of 4.5994. Both have a bit higher miss rates than if you went with a block size of 16 or 32 bytes, but the CPI gets worse as well. The CPI being worse is a difference of a few tenths of the CPI though; so, if performance isn’t needed to be the fastest it could be, saving a few dollars going to a bigger block size wouldn’t make too much of a noticeable difference if you keep it at an 8- or 16-way association.