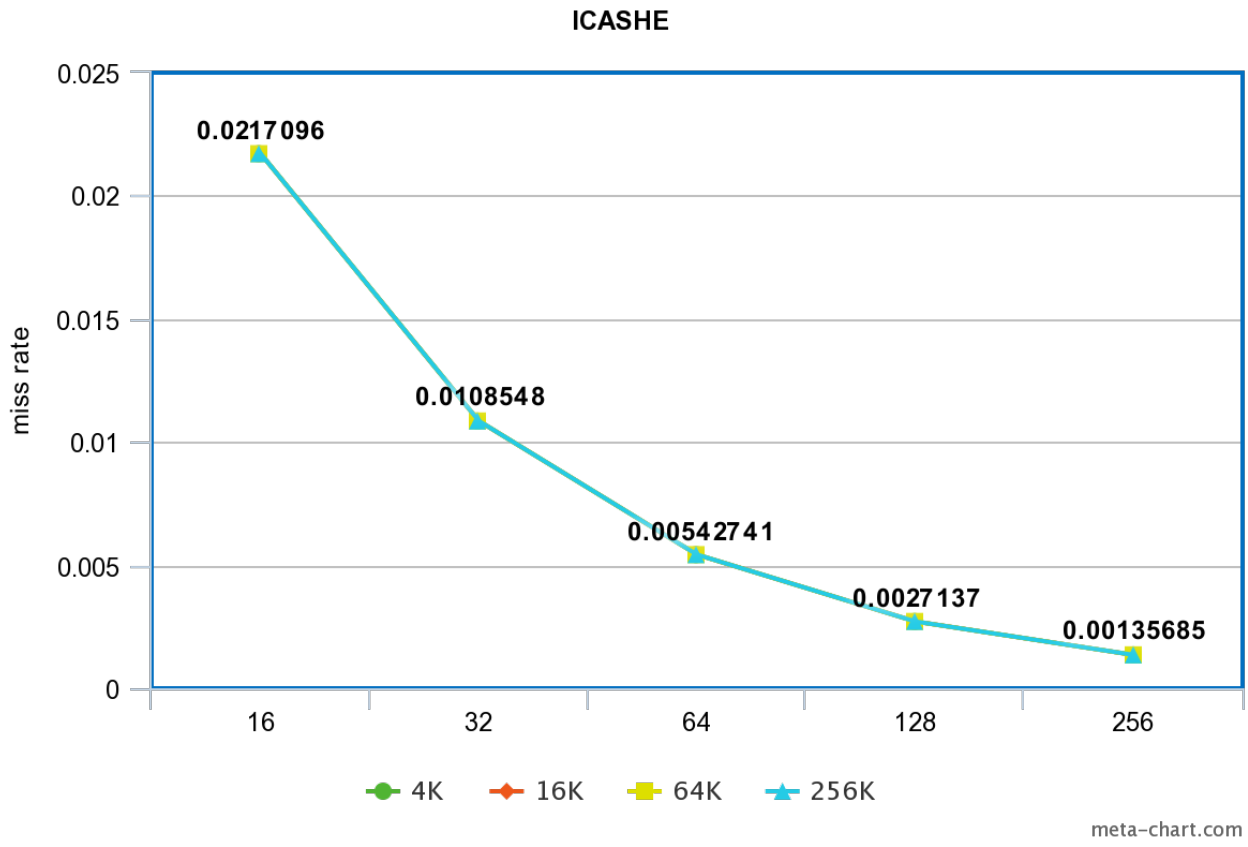
ICACHE

	16	32	64	128	256
4K	0.0217096	0.0108548	0.00542741	0.0027137	0.00135685
16K	0.0217096	0.0108548	0.00542741	0.0027137	0.00135685
64K	0.0217096	0.0108548	0.00542741	0.0027137	0.00135685
256K	0.0217096	0.0108548	0.00542741	0.0027137	0.00135685



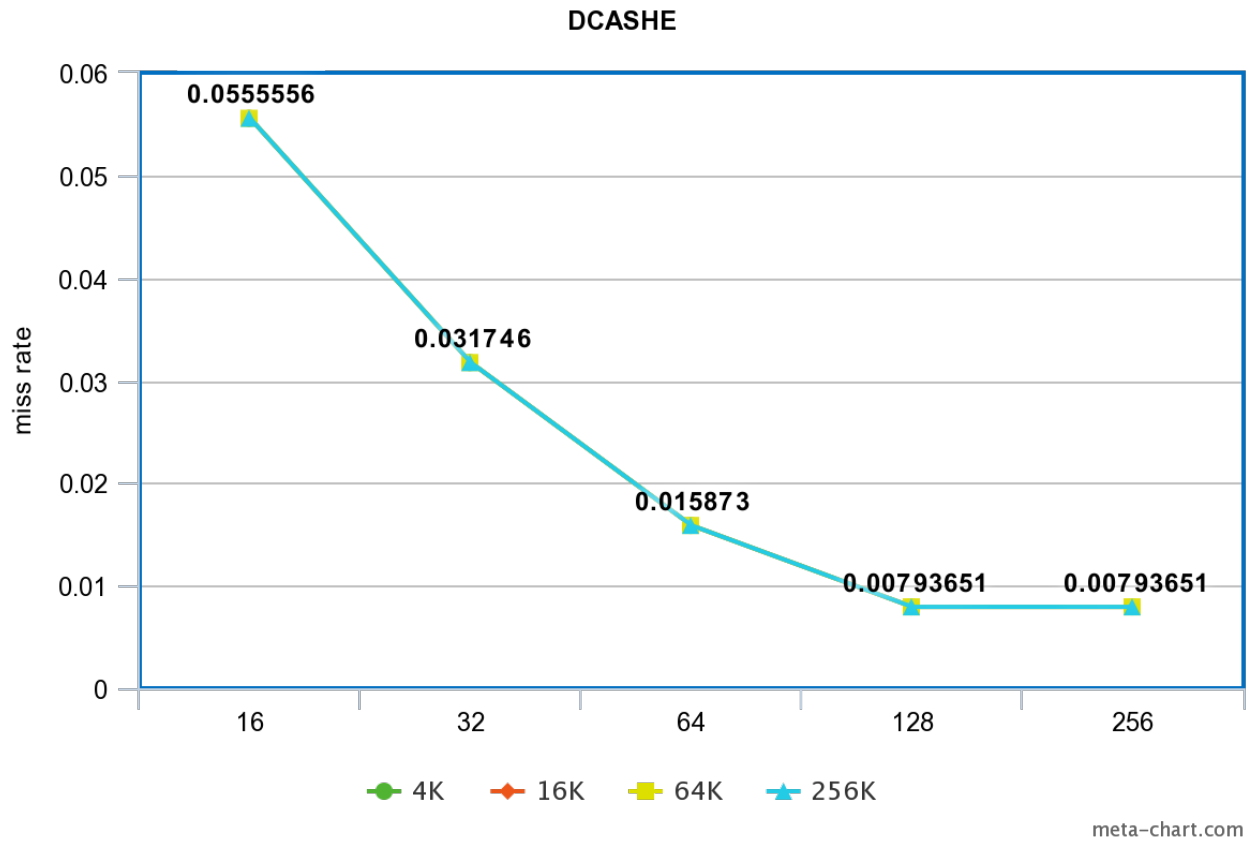
當 cache 固定時，增大 blocks，會因為 spatial locality 而降低 miss rate。

而理論上如果一直增大 blocks 的話，會因為抓到太多不需要的資料，導致 pollution，進而提高 miss penalty，但這邊也許是因為數量不夠大，沒有發生這個現象。

比較特別的還有 cache 愈大，理論上 miss rate 要愈小，這邊會四條都一樣大概也是因為資料數量不夠大吧。

DCACHE

	16	32	64	128	256
4K	0.0555556	0.031746	0.015873	0.00793651	0.00793651
16K	0.0555556	0.031746	0.015873	0.00793651	0.00793651
64K	0.0555556	0.031746	0.015873	0.00793651	0.00793651
256K	0.0555556	0.031746	0.015873	0.00793651	0.00793651



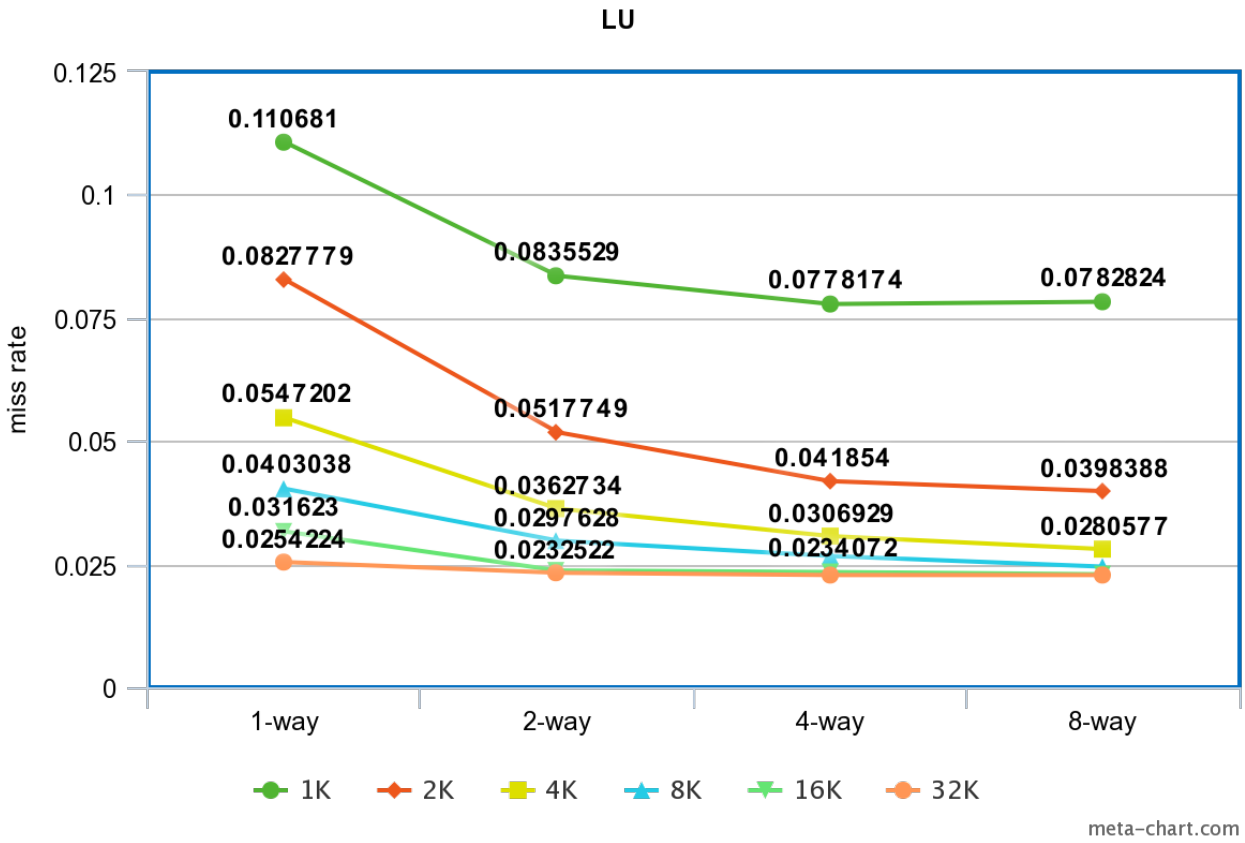
miss rate 相較於 ICACHE 高了許多，原因可能是因為 icache 有 737 筆，dcache 只有 126 筆，因為資料筆數較少，因此一個 miss 的 penalty 會比 icache 大很多，導致 miss rate 較大。

而 block 在 128 與 256 兩種情況下的 miss rate 會一樣，原因應該是 cache 大小大於資料大小，也就是發生 miss 的情況都是第一次放入 cache 的情況，因此已經達到最低的 miss rate 了。

其他特殊的原因大致上都與 ICACHE 情況相同。

LU

	1-way	2-way	4-way	8-way
1K	0.110681	0.0835529	0.0778174	0.0782824
2K	0.0827779	0.0517749	0.041854	0.0398388
4K	0.0547202	0.0362734	0.0306929	0.0280577
8K	0.0403038	0.0297628	0.0266625	0.0244923
16K	0.031623	0.0237173	0.0234072	0.0229422
32K	0.0254224	0.0232522	0.0227872	0.0227872



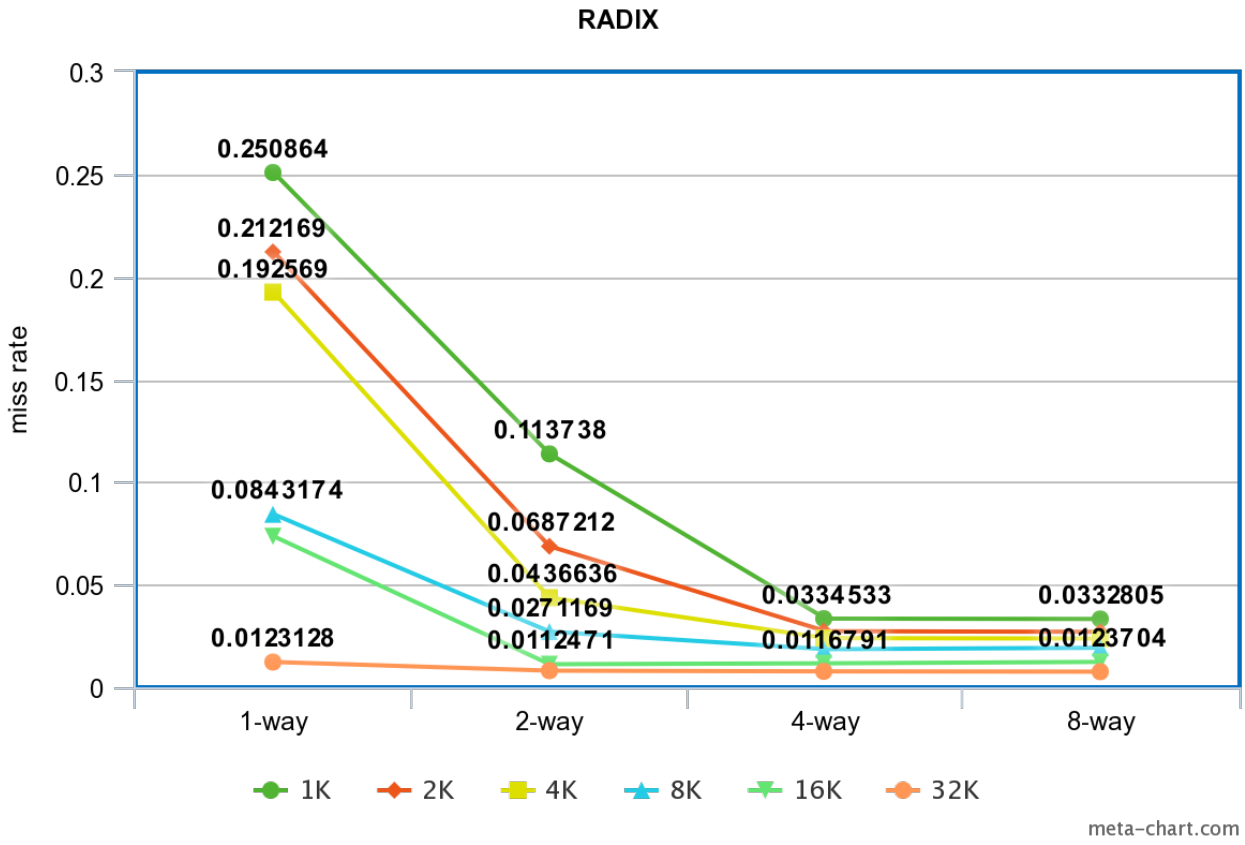
當 cache 增大時，因可容納的資料變多了，miss rate 也就跟著下降，在這邊很正常的表現出來。

而相同的 cache 情況下，加大 associativity 讓一個 set 可放置的資料數變多，可以降低 miss rate。

而 cache 越來越大時，加大 associativity 的變化也會趨近飽和。

RADIX

	1-way	2-way	4-way	8-way
1K	0.250864	0.113738	0.0334533	0.0332805
2K	0.212169	0.0687212	0.0273329	0.0270161
4K	0.192569	0.0436636	0.0239631	0.0237471
8K	0.0843174	0.0271169	0.0185628	0.019254
16K	0.0735599	0.0112471	0.0116791	0.0123704
32K	0.0123128	0.00809332	0.0077621	0.00761809



當 cache 增大時，因可容納的資料變多了，miss rate 也就跟著下降，在這邊很正常的表現出來。

而相同的 cache 情況下，加大 associativity 讓一個 set 可放置的資料數變多，可以降低 miss rate。

而 cache 越來越大時，加大 associativity 的變化也會趨近飽和。