Project Report

Cache Design

Team Name- Architects

1) Gourav Anirudh B J (IMT2023005)
2) Sathish Adithiyaa S V(IMT2023030)
3) Subhash H (IMT2023104)

Project Description

This project evaluates the performance of a set-associative cache under different configurations. By varying cache size, block size, and associativity while keeping other parameters constant, we aim to understand how these factors influence hit and miss rates. To achieve this, we simulate the cache's behavior using various memory trace files.

Requirements for the code:

The code file and the trace files should be in the same directory.
The following packages need to be installed using the pip install 'package_name' command.

- Colorama, Tabulate, pandas.

Code Design

Implementation of the cache is done using an object-oriented design in Python, adhering to the set-associative cache architecture with a Least Recently Used (LRU) replacement policy. The cache simulates hit and miss behavior without storing actual data, focusing on tag matching and the valid bit.

- **Classes:**
    - **Cache**: Models the cache with parameters for cache size, block size, and associativity. It has methods check() which checks for a hit or miss, evictor() which evicts a block based on the LRU policy and lru_handling() which takes care of the lru updates after each step.
    - **Block**: Represents an individual cache block with attributes like tag, valid bit, and LRU counter.

- **Main Functions:**
    - **parta**: Simulates a 4-way associative cache with a fixed size of 1024KB and block size of 4 bytes.
    - **partb**: Varies cache sizes from 128KB to 4096KB and analyzes the hit/miss rates.
    - **partc**: Varies the block size from 1 to 128 bytes while keeping the cache size fixed at 1024KB.
    - **partd**: Varies the associativity from 1-way to 64-way for a fixed cache size of 1024KB.

RESULTS

Part A: 4-Way SA cache, 1024KB cache size and 4-byte block size.

For the default configuration of 1024KB cache size and 4-byte block size, we calculated the number of sets (cache lines) using the formula

$$\textit{Cache size (in bytes)} = \textit{(block-size)} * \textit{(associativity)} * \textit{(no.of cache lines)}$$

We obtained the following hit and miss rates for the trace files:

Trace File	Hit Rate (%)	Miss Rate (%)
gcc.trace	93.8356	6.16445
gzip.trace	66.7055	33.2945
mcf.trace	1.03241	98.9676
swim.trace	92.6225	7.37748
twolf.trace	98.7615	1.23855

Part B: Varying Cache Size (128KB - 4096KB)

Here we are increasing the cache size from 128KB to 4096KB for a 4-way set-associative cache with block size of 4 bytes.

By increasing the cache size, the miss rate generally decreased, even though the decrease is not that evident in the miss rate, it is evident in the miss count. Increasing the cache size beyond 1024 KB did not change the miss count at all, indicating that even though a larger cache can store more data, leading to fewer misses, beyond a certain threshold it makes little to no difference. However, different trace files exhibited different levels of sensitivity to cache size changes.

Output for all trace file with changing Cache size:

gcc.trace

	Cache Size (in KB)	Hit count	Miss count	Hit Rate	Miss Rate
0	128	483719	31964	93.8016	6.19838
1	256	483871	31812	93.8311	6.16891
2	512	483893	31790	93.8354	6.16464
3	1024	483894	31789	93.8356	6.16445
4	2048	483894	31789	93.8356	6.16445
5	4096	483894	31789	93.8356	6.16445

gzip.trace

	Cache Size (in KB)	Hit count	Miss count	Hit Rate	Miss Rate
0	128	320883	160161	66.7055	33.2945
1	256	320883	160161	66.7055	33.2945
2	512	320883	160161	66.7055	33.2945
3	1024	320883	160161	66.7055	33.2945
4	2048	320883	160161	66.7055	33.2945
5	4096	320883	160161	66.7055	33.2945

mcf.trace

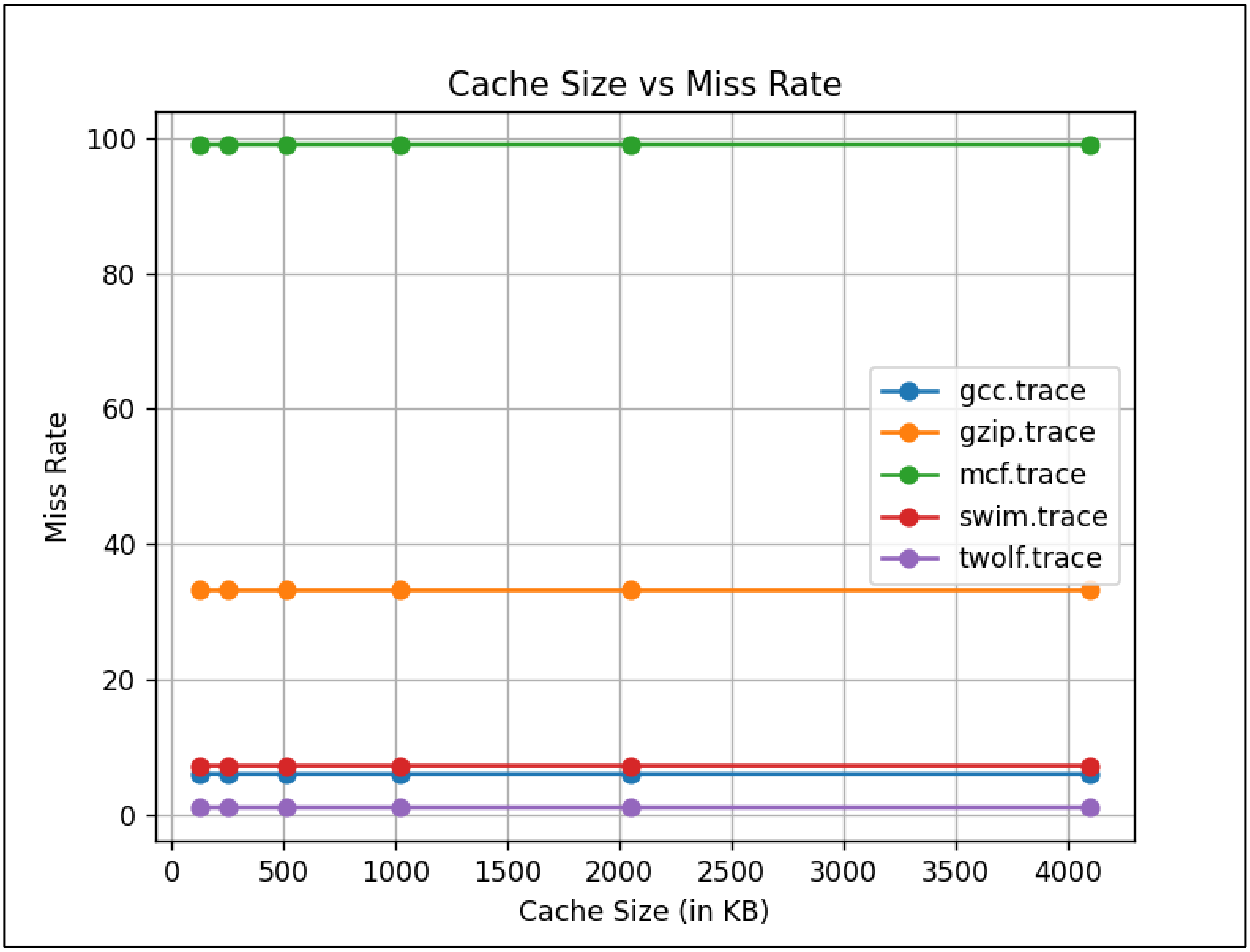
	Cache Size (in KB)	Hit count	Miss count	Hit Rate	Miss Rate
0	128	7508	719722	1.03241	98.9676
1	256	7508	719722	1.03241	98.9676
2	512	7508	719722	1.03241	98.9676
3	1024	7508	719722	1.03241	98.9676
4	2048	7508	719722	1.03241	98.9676
5	4096	7603	719627	1.04547	98.9545

swim.trace

	Cache Size (in KB)	Hit count	Miss count	Hit Rate	Miss Rate
0	128	280817	22376	92.6199	7.38012
1	256	280825	22368	92.6225	7.37748
2	512	280825	22368	92.6225	7.37748
3	1024	280825	22368	92.6225	7.37748
4	2048	280825	22368	92.6225	7.37748
5	4096	280825	22368	92.6225	7.37748

twolf.trace

	Cache Size (in KB)	Hit count	Miss count	Hit Rate	Miss Rate
0	128	476843	5981	98.7612	1.23875
1	256	476844	5980	98.7615	1.23855
2	512	476844	5980	98.7615	1.23855
3	1024	476844	5980	98.7615	1.23855
4	2048	476844	5980	98.7615	1.23855
5	4096	476844	5980	98.7615	1.23855



Observation

All the traces do not behave the same way with increasing cache size. In gcc.trace we see a decrease in miss rate up to 1024KB but no change after this, maybe because by 1024KB most of the repeating addresses have been put into the cache already. In gzip.trace there was no change in miss rate when increasing the cache size, this probably because most of the repeating addresses have already been accommodated within the smaller cache size. In mcf.trace the hit rate being 1% mostly indicates that mainly only new addresses are being accessed. There is little to no change in moss rates over the whole range of cache sizes in swim.trace and twolf.trace as well.

Part C: Varying Block Size (1B to 128B)

Increasing the block size from 1B to 128B for a 4-way set associative cache of fixed size 1024KB revealed that larger block sizes decreased miss rates for most files but beyond a certain threshold, due to reduced cache line count, the miss rate change was negligible. The sensitivity of the change in miss rate is different for each trace file.

Output for all trace file with changing Block size:

gcc.trace

	Block Size	Hit count	Miss count	Hit Rate	Miss Rate
0	1	480611	35072	93.1989	6.80108
1	2	482807	32876	93.6248	6.37523
2	4	483894	31789	93.8356	6.16445
3	8	494677	21006	95.9266	4.07343
4	16	504467	11216	97.825	2.17498
5	32	509644	6039	98.8289	1.17107
6	64	512310	3373	99.3459	0.654084
7	128	513728	1955	99.6209	0.379109

gzip.trace

	Block Size	Hit count	Miss count	Hit Rate	Miss Rate
0	1	320875	160169	66.7039	33.2961
1	2	320876	160168	66.7041	33.2959
2	4	320883	160161	66.7055	33.2945
3	8	320891	160153	66.7072	33.2928
4	16	321268	159776	66.7856	33.2144
5	32	321459	159585	66.8253	33.1747
6	64	321559	159485	66.8461	33.1539
7	128	321609	159435	66.8565	33.1435

mcf.trace

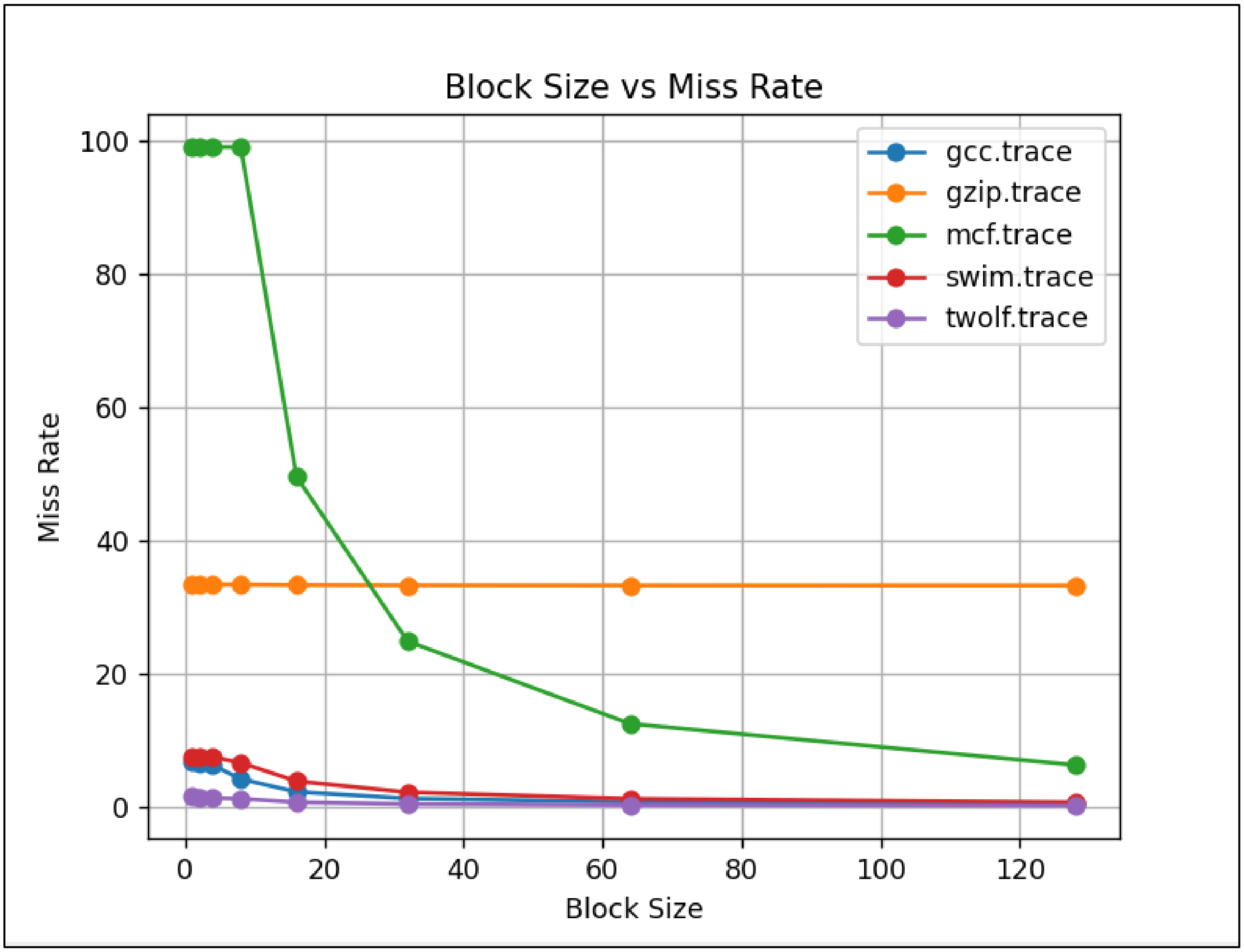
	Block Size	Hit count	Miss count	Hit Rate	Miss Rate
0	1	7451	719779	1.02457	98.9754
1	2	7481	719749	1.0287	98.9713
2	4	7508	719722	1.03241	98.9676
3	8	7551	719679	1.03832	98.9617
4	16	367273	359957	50.503	49.497
5	32	547152	180078	75.2378	24.7622
6	64	637112	90118	87.608	12.392
7	128	682109	45121	93.7955	6.2045

swim.trace

	Block Size	Hit count	Miss count	Hit Rate	Miss Rate
0	1	280588	22605	92.5444	7.45565
1	2	280737	22456	92.5935	7.4065
2	4	280825	22368	92.6225	7.37748
3	8	283377	19816	93.4642	6.53577
4	16	291770	11423	96.2324	3.76757
5	32	296797	6396	97.8905	2.10955
6	64	299740	3453	98.8611	1.13888
7	128	301367	1826	99.3977	0.602257

twolf.trace

	Block Size	Hit count	Miss count	Hit Rate	Miss Rate
0	1	475470	7354	98.4769	1.52312
1	2	476358	6466	98.6608	1.3392
2	4	476844	5980	98.7615	1.23855
3	8	477319	5505	98.8598	1.14017
4	16	479869	2955	99.388	0.612024
5	32	481182	1642	99.6599	0.340083
6	64	481870	954	99.8024	0.197588
7	128	482249	575	99.8809	0.119091



Observation:

Not all traces exhibit the same behavior. In gcc.trace the increase in block size affects the miss rate substantially, reducing it from 6.8% to 0.3%. This is mainly because of the increased use of spatial locality. The gzip.trace file showed no significant change in hit rates with varying block sizes. In contrast, the mcf.trace file had a very high miss rate of 98.9% with block sizes up to 8B, but the miss rate decreased rapidly from 8B to 128B, reaching 6.2% at a block size of 128B. This shows that mcf.trace utilizes spatial locality to its fullest after 8B. Swim.trace behaved similarly to gcc.trace whereas twolf.trace behaved similarly to gzip.trace.

Part D: Varying Associativity (1-Way to 64-Way)

Increasing the associativity generally decreased the miss rate, but beyond 8-way associativity the trace files exhibited diminishing returns.

Output for all trace file with changing Associativity:

gcc.trace

	Associativity	Hit count	Miss count	Hit Rate	Miss Rate
0	1	483868	31815	93.8305	6.16949
1	2	483890	31793	93.8348	6.16522
2	4	483894	31789	93.8356	6.16445
3	8	483894	31789	93.8356	6.16445
4	16	483895	31788	93.8357	6.16425
5	32	483896	31787	93.8359	6.16406
6	64	483896	31787	93.8359	6.16406

gzip.trace

	Associativity	Hit count	Miss count	Hit Rate	Miss Rate
0	1	320883	160161	66.7055	33.2945
1	2	320883	160161	66.7055	33.2945
2	4	320883	160161	66.7055	33.2945
3	8	320883	160161	66.7055	33.2945
4	16	320883	160161	66.7055	33.2945
5	32	320883	160161	66.7055	33.2945
6	64	320883	160161	66.7055	33.2945

mcf.trace

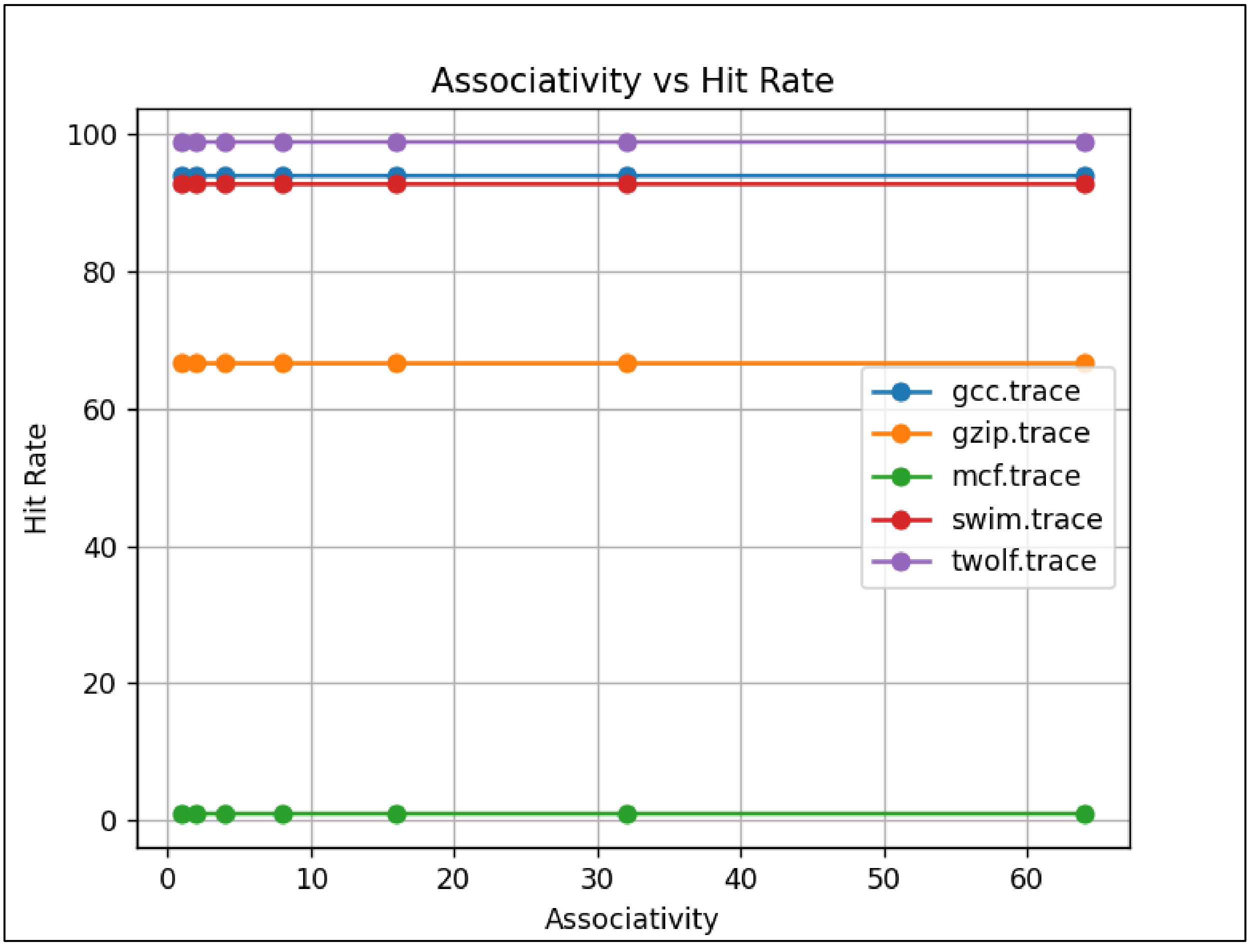
	Associativity	Hit count	Miss count	Hit Rate	Miss Rate
0	1	7505	719725	1.032	98.968
1	2	7507	719723	1.03227	98.9677
2	4	7508	719722	1.03241	98.9676
3	8	7508	719722	1.03241	98.9676
4	16	7508	719722	1.03241	98.9676
5	32	7508	719722	1.03241	98.9676
6	64	7508	719722	1.03241	98.9676

swim.trace

	Associativity	Hit count	Miss count	Hit Rate	Miss Rate
0	1	280819	22374	92.6205	7.37946
1	2	280825	22368	92.6225	7.37748
2	4	280825	22368	92.6225	7.37748
3	8	280825	22368	92.6225	7.37748
4	16	280825	22368	92.6225	7.37748
5	32	280825	22368	92.6225	7.37748
6	64	280825	22368	92.6225	7.37748

twolf.trace

	Associativity	Hit count	Miss count	Hit Rate	Miss Rate
0	1	476771	6053	98.7463	1.25367
1	2	476841	5983	98.7608	1.23917
2	4	476844	5980	98.7615	1.23855
3	8	476844	5980	98.7615	1.23855
4	16	476844	5980	98.7615	1.23855
5	32	476844	5980	98.7615	1.23855
6	64	476844	5980	98.7615	1.23855



Observation:

In general, higher associativity reduced conflicts, but further increases beyond 8-way yielded minimal improvements in hit rate. In gcc.trace the miss rate does change slightly when changed from 1-way to a 4-way SA cache but beyond that there is little to no change. In gzip.trace there is no change at all in the miss rate over the whole range of associativities. Mcf.trace behaves similarly to gcc.trace. In swim.trace there is a change only when going from 1-way to 2-way. In twolf.trace there is some difference when going from a 1-way to a 2-way SA cache but no change after that.

Conclusion

The experiments highlight the trade-offs involved in cache design. Increasing cache size and associativity generally improves hit rates, but with diminishing returns. Block size also plays a crucial role, with optimal configurations depending on the specific memory access patterns of the program.