Report

In order to do the experiment that determines the best fit configuration, I have following methodology for optimization:

Use $cacheesim$ to model and simulate different cache configurations with realistic workloads. Simulations can provide detailed insights into access times, miss rates, and power consumption.

Analyze access patterns of typical workloads(The given trace files) to understand the impact of B and S on hit and miss rates.

Calculate the overhead for different configurations. Highly associative caches require more tag bits per block, increasing the metadata storage.

Systematically vary B and S to find configurations that meet the optimization criteria. Pay special attention to points where further increases in B or S yield minimal improvements in access time or significantly increase metadata storage, area, or power.

The shell script I used to execute the program with different configurations is as follows:

```bash
#!/bin/bash

# Fixed parameters
c=15
C=17
trace_file="./traces/gcc.trace"
output="test.log"

# Iterate over the parameters within the specified ranges
for b in {5..7}; do
    for s in {0..5}; do
        for S in {1..6}; do # Start S greater than s to meet the restriction
            for P in {0..2}; do
                for I in "LIP" "MIP"; do
                    for r in "LFU" "LRU"; do
                        # Ensure the size of the L2 cache is strictly greater than the size of the L1 cache
                        if ((C > c)) && ((S > s)) && ((C - S > c - s)); then
                            ./run.sh -c "$c" -b "$b" -s "$s" -C "$C" -S "$S" -P "$P" -I "$I" -r "$r" -f "$trace_file" >>"$output"
                        fi
                    done
                done
            done
        done
    done
done

```

The following python script is to find the smallest Average Access Time(AAT) for L1:

```python
def find_smallest_l1_aat(file_path):
    smallest_aat = None

    with open(file_path, 'r') as file:
        for line in file:
            if line.startswith("L1 average access time (AAT):"):
                # Extract the numeric value of AAT
                aat = float(line.split(":")[1].strip())

                # Initialize smallest_aat or compare with the current smallest
                if smallest_aat is None or aat < smallest_aat:
                    smallest_aat = aat

    return smallest_aat

# Replace 'test.log' with the actual path to your log file
file_path = 'test.log'
smallest_aat = find_smallest_l1_aat(file_path)
print(f"The smallest L1 average access time (AAT) is: {smallest_aat}")
```

File $gcc.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $2.044\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,5). Replacement policy: LRU. Prefetcher disabled.
L2 (C,B,S): (17,7,6). Replacement policy: LRU. +1 prefetcher. Prefetch insertion policy: LIP.

Cache Statistics
----------------
Reads: 315500
Writes: 184500

L1 accesses: 500000
L1 hits: 495015
L1 misses: 4985
L1 hit ratio: 0.990
L1 miss ratio: 0.010
L1 average access time (AAT): 2.044

L2 reads: 4985
L2 writes: 2055
L2 read hits: 3818
L2 read misses: 1167
L2 prefetches: 840
L2 read hit ratio: 0.766
L2 read miss ratio: 0.234
L2 average access time (AAT): 29.510
```

- **L1 Cache** `(C=15, B=7, S=5)`: 14,336 bits
- **L2 Cache** `(C=17, B=7, S=6)`: 55,296 bits

File $leela.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $1.799\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,4). Replacement policy: LRU. Prefetcher disabled.
L2 (C,B,S): (17,7,5). Replacement policy: LRU. +1 prefetcher. Prefetch
insertion policy: LIP.

Cache Statistics
----------------
Reads: 352008
Writes: 147992

L1 accesses: 500000
L1 hits: 499464
L1 misses: 536
L1 hit ratio: 0.999
L1 miss ratio: 0.001
L1 average access time (AAT): 1.799

L2 reads: 536
L2 writes: 14
L2 read hits: 323
L2 read misses: 213
L2 prefetches: 123
L2 read hit ratio: 0.603
L2 read miss ratio: 0.397
L2 average access time (AAT): 45.839
```

- **L1 Cache** `(C=15, B=7, S=4)`: 14,080 bits
- **L2 Cache** `(C=17, B=7, S=5)`: 54,272 bits

File $linpack.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $4.136\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,3). Replacement policy: LRU. Prefetcher disabled.
L2 (C,B,S): (17,7,4). Replacement policy: LRU. +1 prefetcher. Prefetch
insertion policy: LIP.

Cache Statistics
----------------
Reads: 332993
Writes: 167007

L1 accesses: 500000
L1 hits: 478869
L1 misses: 21131
L1 hit ratio: 0.958
L1 miss ratio: 0.042
L1 average access time (AAT): 4.136

L2 reads: 21131
L2 writes: 20875
L2 read hits: 10490
```

```
L2 read misses: 10641
L2 prefetches: 10494
L2 read hit ratio: 0.496
L2 read miss ratio: 0.504
L2 average access time (AAT): 56.457
```

- **L1 Cache** `(C=15, B=7, S=3)`: 13,824 bits
- **L2 Cache** `(C=17, B=7, S=4)`: 53,248 bits

File $matmul\_naive.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $3.321\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,5). Replacement policy: LFU. Prefetcher disabled.
L2 (C,B,S): (17,7,6). Replacement policy: LFU. +1 prefetcher. Prefetch
insertion policy: LIP.

Cache Statistics
----------------
Reads: 494452
Writes: 5548

L1 accesses: 500000
L1 hits: 385860
L1 misses: 114140
L1 hit ratio: 0.772
L1 miss ratio: 0.228
L1 average access time (AAT): 3.321

L2 reads: 114140
L2 writes: 981
L2 read hits: 113249
L2 read misses: 891
L2 prefetches: 808
L2 read hit ratio: 0.992
L2 read miss ratio: 0.008
L2 average access time (AAT): 6.881
```

- **L1 Cache** `(C=15, B=7, S=5)`: 14,336 bits
- **L2 Cache** `(C=17, B=7, S=6)`: 55,296 bits

File $matmul\_tiled.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $1.909\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,3). Replacement policy: LRU. Prefetcher disabled.
L2 (C,B,S): (17,7,4). Replacement policy: LRU. +1 prefetcher. Prefetch
insertion policy: MIP.

Cache Statistics
----------------
Reads: 482548
```

```
Writes: 17452

L1 accesses: 500000
L1 hits: 498182
L1 misses: 1818
L1 hit ratio: 0.996
L1 miss ratio: 0.004
L1 average access time (AAT): 1.909

L2 reads: 1818
L2 writes: 311
L2 read hits: 1135
L2 read misses: 683
L2 prefetches: 610
L2 read hit ratio: 0.624
L2 read miss ratio: 0.376
L2 average access time (AAT): 43.669
```

- **L1 Cache** `(C=15, B=7, S=3)`: 13,824 bits
- **L2 Cache** `(C=17, B=7, S=4)`: 53,248 bits

File $mcf.trace$: The smallest average access time (AAT) for L1 among the provided configurations is $1.918\ ns$, which is achieved with the cache settings of:

```
Cache Settings
--------------
L1 (C,B,S): (15,7,4). Replacement policy: LRU. Prefetcher disabled.
L2 (C,B,S): (17,7,5). Replacement policy: LRU. +1 prefetcher. Prefetch
insertion policy: MIP.

Cache Statistics
----------------
Reads: 304587
Writes: 195413

L1 accesses: 500000
L1 hits: 498969
L1 misses: 1031
L1 hit ratio: 0.998
L1 miss ratio: 0.002
L1 average access time (AAT): 1.918

L2 reads: 1031
L2 writes: 709
L2 read hits: 256
L2 read misses: 775
L2 prefetches: 766
L2 read hit ratio: 0.248
L2 read miss ratio: 0.752
L2 average access time (AAT): 81.270
```

- **L1 Cache** `(C=15, B=7, S=4)`: 14,080 bits
- **L2 Cache** `(C=17, B=7, S=5)`: 54,272 bits