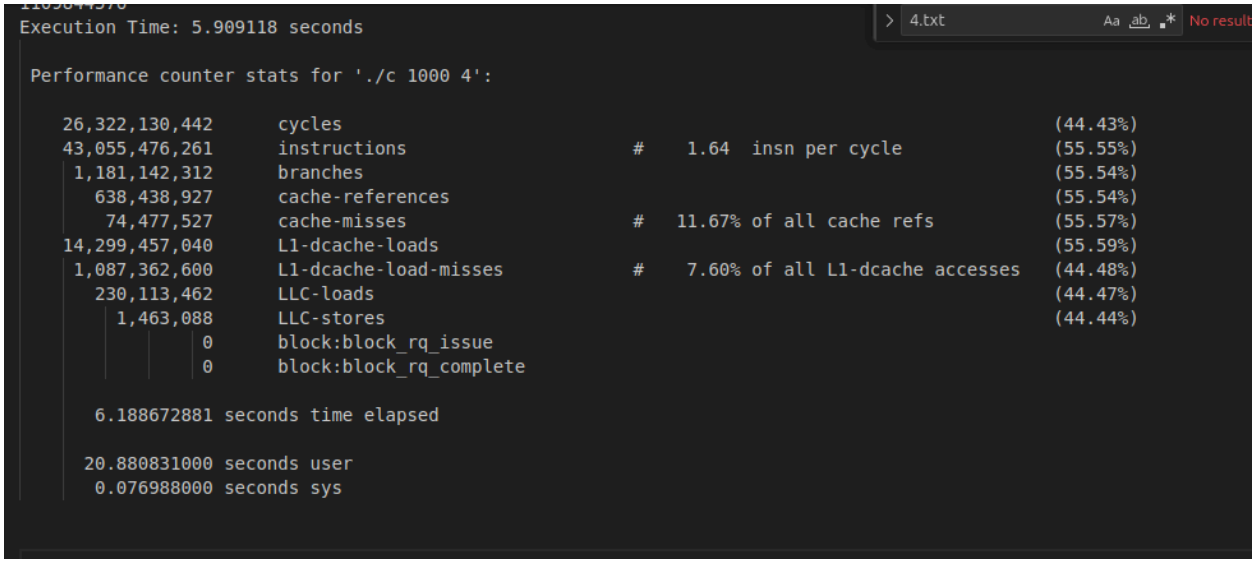
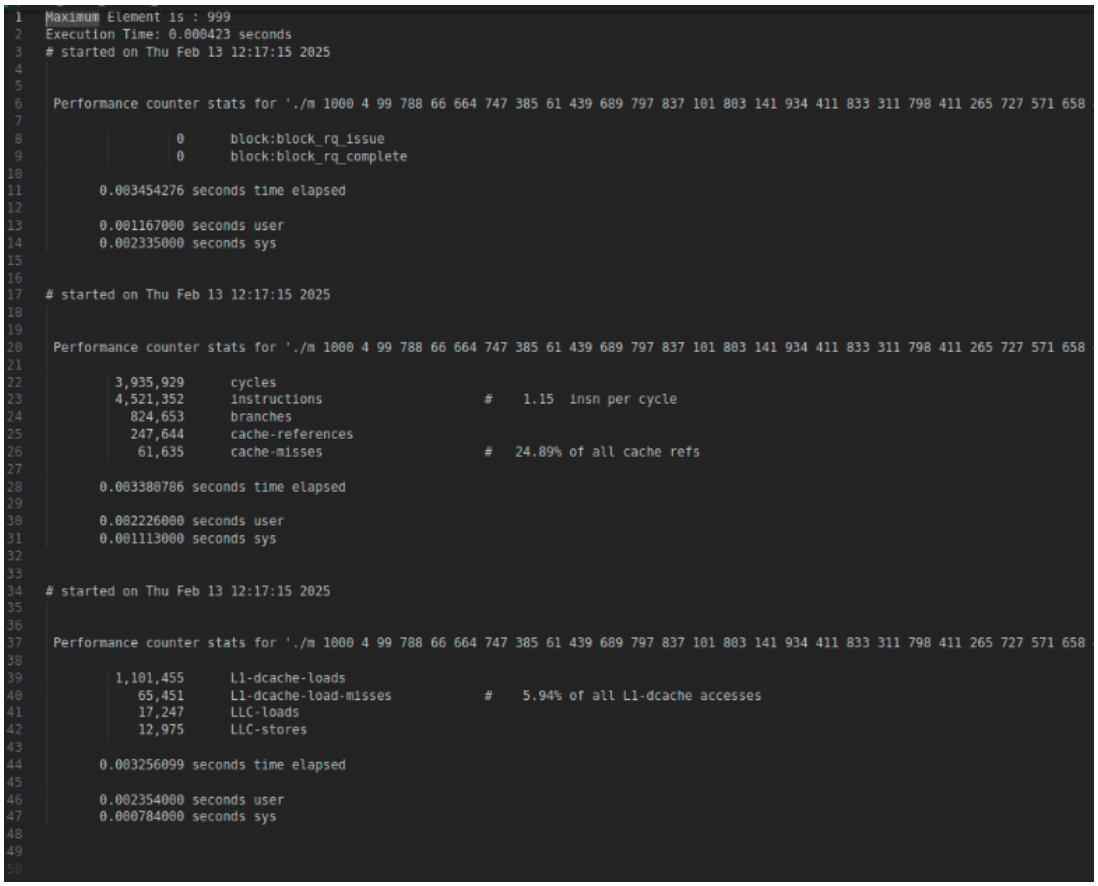
- **CPU bound task** : Matrix Multiplication , I have taken a matrix of size 1000 number of threads is 4.Perf result for it:



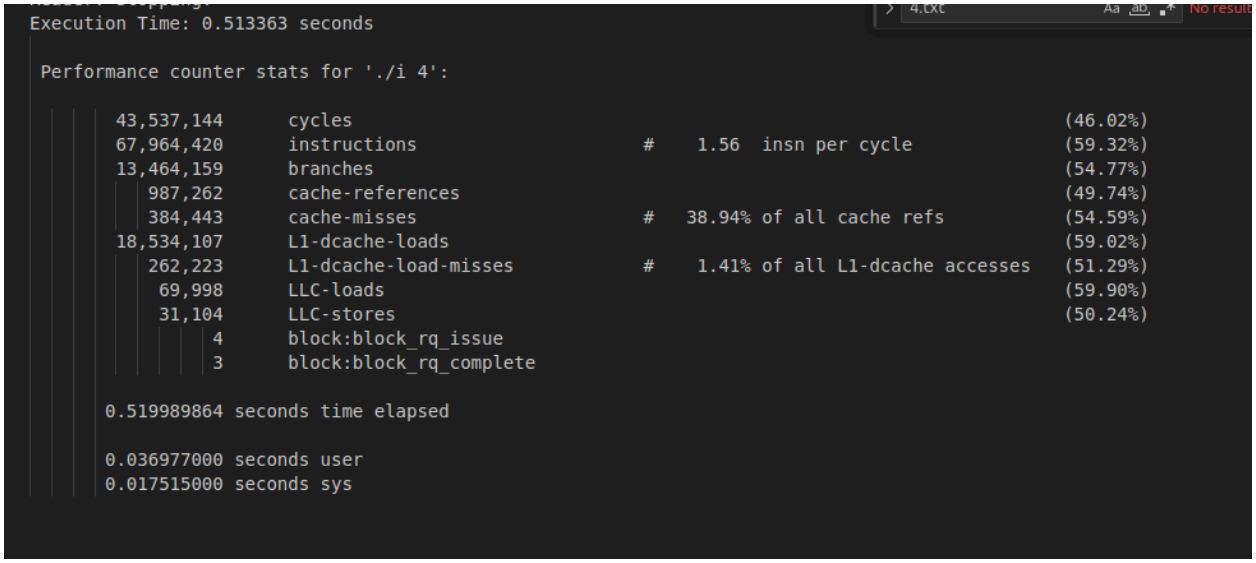
```
Execution Time: 5.909118 seconds

Performance counter stats for './c 1000 4':

    26,322,130,442      cycles                                                  (44.43%)
    43,055,476,261      instructions              #    1.64  insn per cycle     (55.55%)
     1,181,142,312      branches                                                (55.54%)
       638,438,927      cache-references                                        (55.54%)
        74,477,527      cache-misses              #   11.67% of all cache refs  (55.57%)
    14,299,457,040      L1-dcache-loads                                         (55.59%)
     1,087,362,600      L1-dcache-load-misses     #    7.60% of all L1-dcache accesses  (44.48%)
       230,113,462      LLC-loads                                               (44.47%)
         1,463,088      LLC-stores                                              (44.44%)
                 0      block:block_rq_issue
                 0      block:block_rq_complete

       6.188672881 seconds time elapsed

      20.880831000 seconds user
       0.076988000 seconds sys
```

- **Memory bound task :** Finding the largest number of the given array(here 1000),used mutex for proper synchronization.Perf result for it:

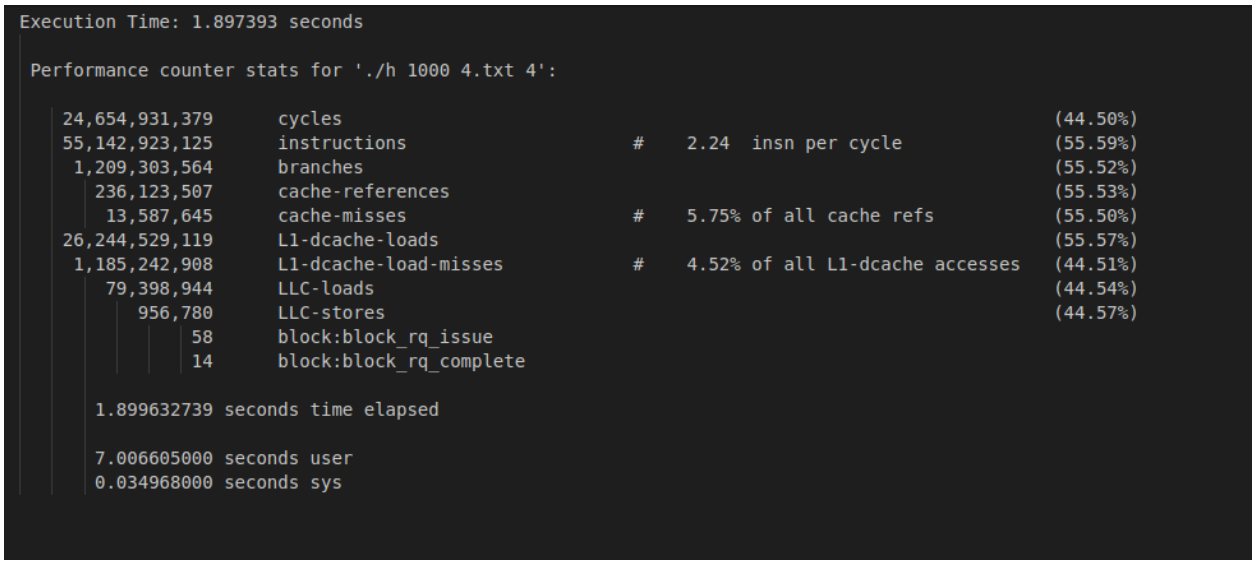


```
Maximum Element is : 999
Execution Time: 0.000423 seconds
# started on Thu Feb 13 12:17:15 2025


 Performance counter stats for './m 1000 4 99 788 66 664 747 385 61 439 689 797 837 101 803 141 934 411 833 311 798 411 265 727 571 658

                 0      block:block_rq_issue
                 0      block:block_rq_complete

       0.003454276 seconds time elapsed

       0.001167000 seconds user
       0.002335000 seconds sys


# started on Thu Feb 13 12:17:15 2025


 Performance counter stats for './m 1000 4 99 788 66 664 747 385 61 439 689 797 837 101 803 141 934 411 833 311 798 411 265 727 571 658

         3,935,929      cycles
         4,521,352      instructions              #    1.15  insn per cycle
           824,653      branches
           247,644      cache-references
            61,635      cache-misses              #   24.89% of all cache refs

       0.003380786 seconds time elapsed

       0.002226000 seconds user
       0.001113000 seconds sys


# started on Thu Feb 13 12:17:15 2025


 Performance counter stats for './m 1000 4 99 788 66 664 747 385 61 439 689 797 837 101 803 141 934 411 833 311 798 411 265 727 571 658

         1,101,455      L1-dcache-loads
            65,451      L1-dcache-load-misses     #    5.94% of all L1-dcache accesses
            17,247      LLC-loads
            12,975      LLC-stores

       0.003256099 seconds time elapsed

       0.002354000 seconds user
       0.000784000 seconds sys
```

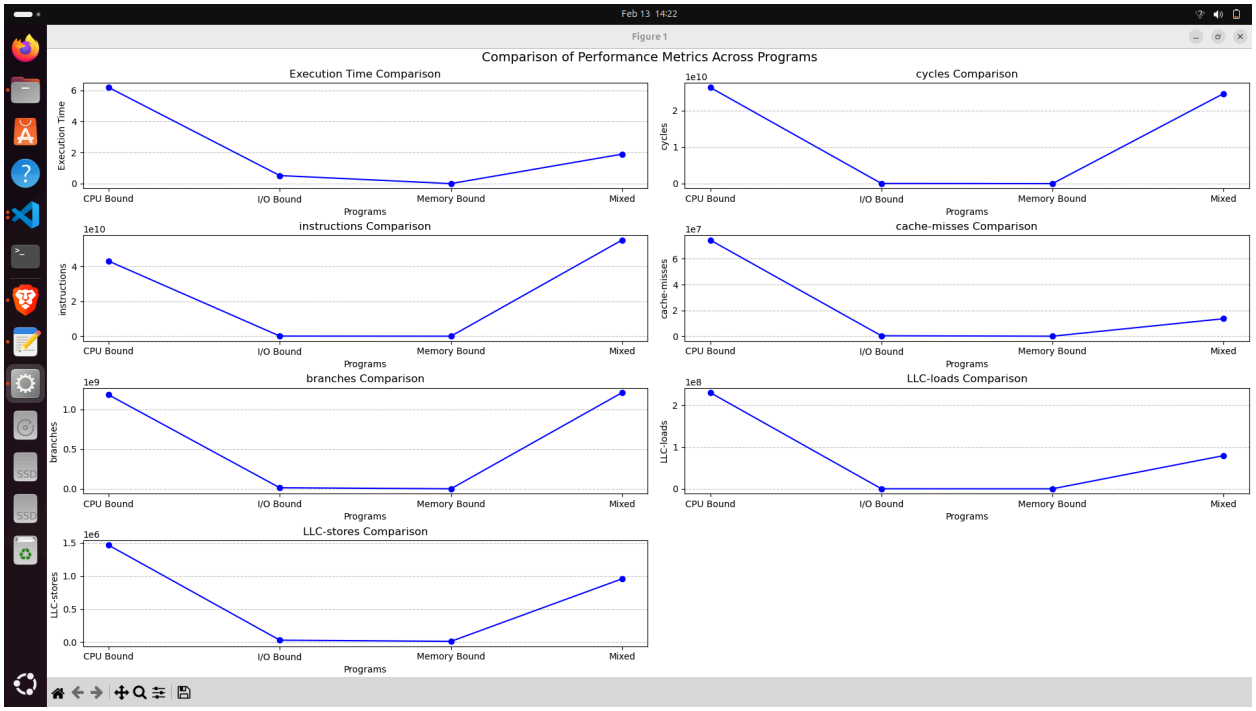
- **IO_bound task :** I have a file numbers.txt with numbers 1 to 10000 , the program reads numbers from from there and multiplies each number by 2 and writes it again to same file.Perf results :



- **Mixed Workload:** It is first doing matrix multiplication for CPU task, then find largest number from there , for Memory bound task an writing all this to a .txt file , then reading everything from the file n writing it on console.Perf Result:



COMPARING THE RESULT OF ALL PROGRAMS:



1.Execution Time:

- CPU Bound program exhibits the highest execution time compared to the other programs. This is because the program is computation-heavy and depends on the CPU's processing power, which is inefficient due to memory access inefficiencies (such as cache misses).
- I/O Bound has a lower execution time, as expected, since this program's bottleneck is related to I/O operations, and it doesn't spend as much time on CPU-bound computations.
- Memory Bound has slightly higher execution time than I/O Bound due to the significant memory access overhead, leading to more cache misses.
- Mixed program exhibits the lowest execution time, indicating it balances both CPU and memory operations efficiently.

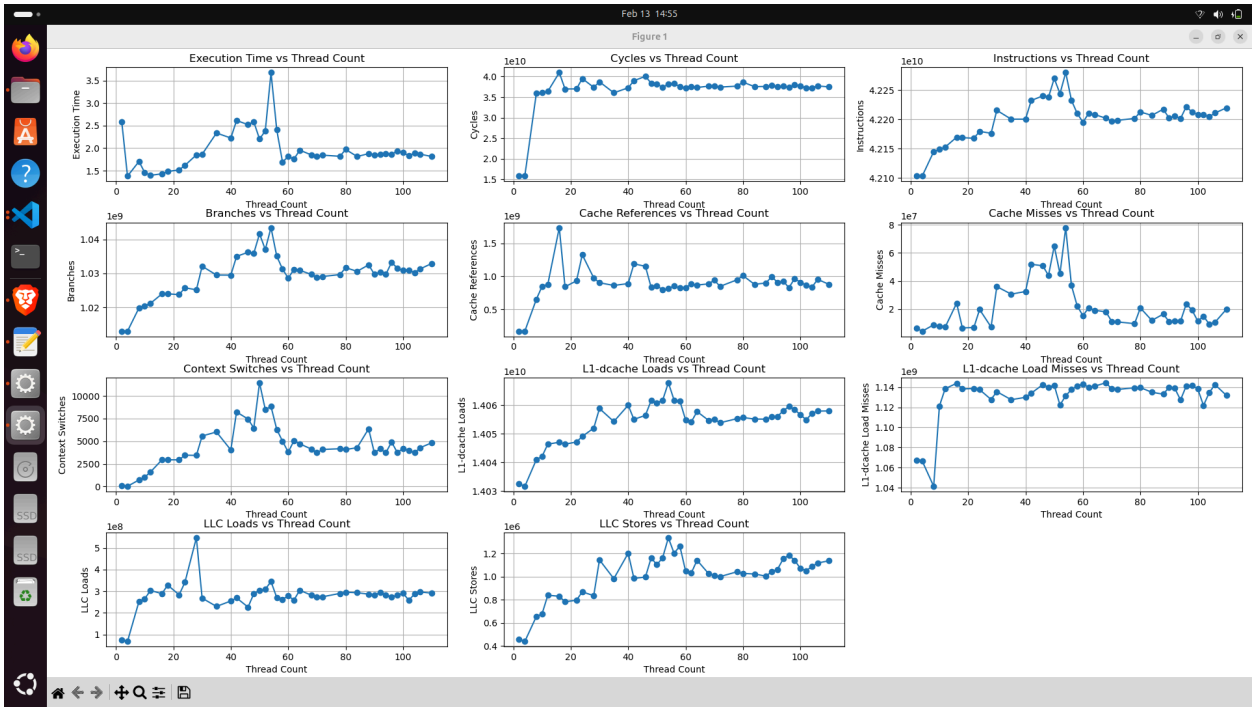
2. Cycles:
   - CPU Bound program consumes the highest number of cycles, as it is heavily CPU-bound with intensive computations.
   - I/O Bound and Memory Bound programs show significantly fewer cycles. I/O Bound spends more time waiting on I/O, and Memory Bound spends more time dealing with memory accesses rather than performing computational tasks.
   - Mixed program shows intermediate cycle usage, indicating balanced operations.
3. Instructions:
   - CPU Bound program has the lowest instruction count among all, which indicates that the program may be spending more time on each instruction, likely due to inefficient memory access.
   - I/O Bound and Memory Bound have higher instruction counts because their workloads involve more data processing.
   - Mixed has a higher instruction count as it involves both memory and CPU operations.
4. Cache Misses:
   - Memory Bound and Mixed programs have higher cache misses, indicating inefficiency in memory accesses. Memory Bound has a particularly high rate of cache misses due to heavy reliance on memory operations.
   - CPU Bound shows fewer cache misses, but its performance is still constrained by the CPU and memory bandwidth.

    - I/O Bound has fewer cache misses due to less reliance on memory operations, but its performance is hindered by the I/O bottleneck.
5. LLC Loads:
    - Memory Bound and Mixed programs exhibit the highest number of LLC loads, indicating significant memory access. These programs rely on efficient memory bandwidth, and optimizing memory access patterns could help reduce these counts.
    - CPU Bound program has fewer LLC loads, as it spends more time performing calculations than accessing memory.
    - I/O Bound has fewer LLC loads due to the nature of I/O operations, where the program isn't processing much data in memory.

Discussion of Results:

- CPU Bound Program:
    - Bottleneck: The CPU Bound program is constrained primarily by computation-intensive tasks. The high cycle and instruction counts indicate that it is heavily utilizing CPU resources, but the high cache misses suggest that memory access inefficiencies are also a significant issue.
    - Optimization Opportunity: Improving memory access patterns, optimizing the algorithm, and better utilizing CPU caches could help improve performance. It might benefit from using more efficient matrix operations (e.g., blocking or using optimized libraries).
- I/O Bound Program:
    - Bottleneck: This program is constrained by I/O wait times rather than computational tasks. The lower cycle count indicates that the CPU isn't as active, but the program still shows a moderate level of cache misses and LLC loads.
    - Optimization Opportunity: I/O optimizations could yield substantial performance improvements. Asynchronous I/O or reducing the amount of I/O-bound waiting could improve performance.
- Memory Bound Program:
    - Bottleneck: This program is limited by inefficient memory access patterns, as reflected in the high cache misses, LLC loads, and LLC stores.
    - Optimization Opportunity: Optimizing memory access patterns through improved cache locality or utilizing better data structures could reduce cache misses. Using more efficient algorithms for memory-bound tasks might also help.
- Mixed Program:
    - Bottleneck: The Mixed program balances both memory and computation, resulting in moderate cycle usage and cache misses. However, the performance could still be impacted by the inefficiencies in both memory and CPU operations.
    - Optimization Opportunity: The Mixed program could benefit from better distributing the workload between CPU and memory-bound tasks. Optimizing both memory access patterns and CPU instructions could lead to better performance.

SCALABILITY :
1.CPU - BOUND:



Analysis of Performance for Matrix Size 1000 with different number of Threads:

Key Observations:

1. Execution Time:
    - Spike at certain thread count: Execution time drops initially but spikes at around 50 threads, indicating resource contention or saturation after a certain threshold. Beyond this point, adding more threads doesn't provide performance benefits and even degrades performance.
2. Cache Misses:
    - Significant increase with threads: As the number of threads increases, cache misses rise sharply. This points to inefficient cache utilization, likely due to cache contention where multiple threads are competing for the same cache resources.
3. LLC Loads and Stores:
    - Increased LLC accesses: The growing number of LLC loads and stores with more threads suggests that data is being fetched from higher-level caches (L2/L3) instead of being efficiently retrieved from the L1 cache. This indicates that memory access patterns need optimization, as the data isn't local to the processor's first-level cache.

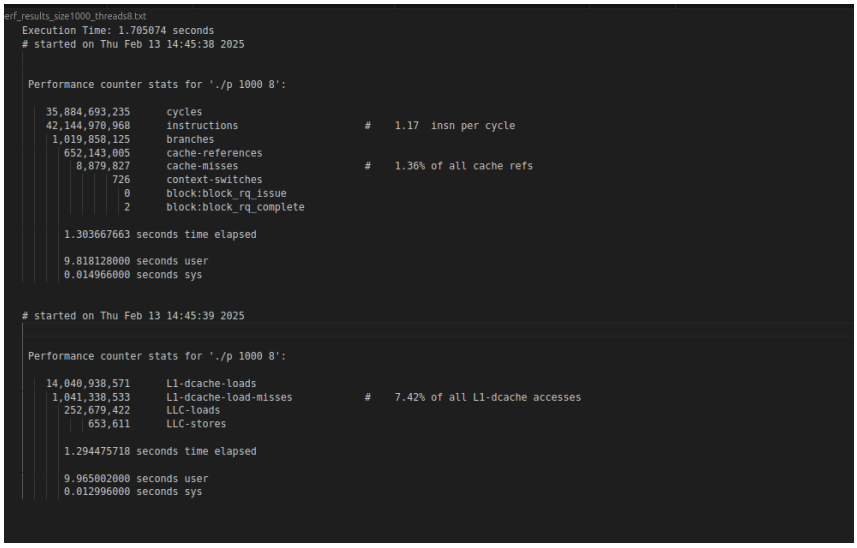
Importance and Key Takeaways:

- Threading Efficiency: Beyond 50 threads, the program shows a performance drop due to CPU resource contention and thread management overhead.
- Memory and Cache Optimization: High cache misses and LLC accesses indicate that memory locality is not being optimized, which is a bottleneck. Optimizing data access patterns (e.g., by improving cache-friendly access patterns) would help in reducing these memory access penalties.

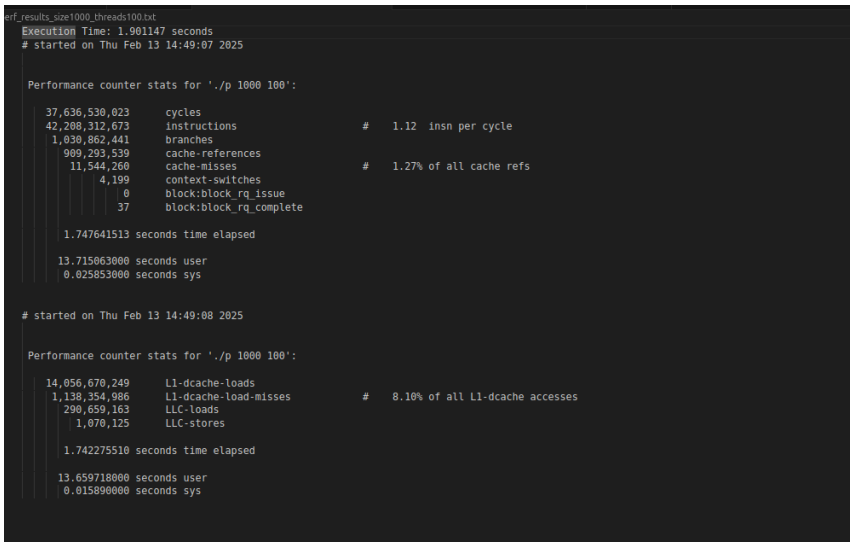
Conclusion:

The performance bottleneck seems to be related to memory access and CPU thread saturation. To improve performance, the focus should be on optimizing thread usage and memory access patterns to better utilize cache, especially L1 cache.

PERF RESULT FOR 8 THREADS VS 100 THREADS:

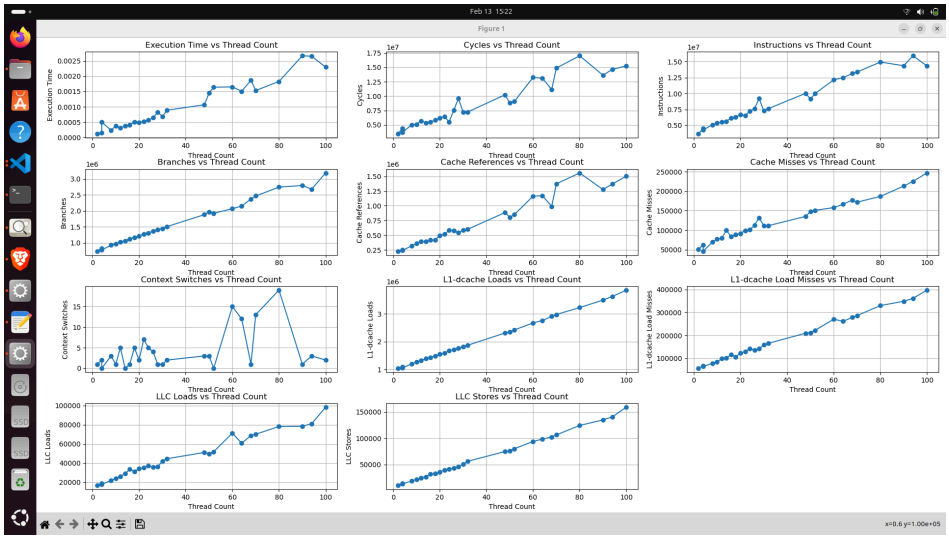


```
erf_results_size1000_threads8.txt
Execution Time: 1.705074 seconds
# started on Thu Feb 13 14:45:38 2025


 Performance counter stats for './p 1000 8':

    35,884,693,235      cycles
    42,144,970,968      instructions              #    1.17  insn per cycle
     1,019,058,125      branches
       652,143,805      cache-references
         8,879,827      cache-misses              #    1.36% of all cache refs
               726      context-switches
                 0      block:block_rq_issue
                 2      block:block_rq_complete

       1.303667663 seconds time elapsed

       9.818128000 seconds user
       0.014966000 seconds sys


# started on Thu Feb 13 14:45:39 2025


 Performance counter stats for './p 1000 8':

    14,040,938,571      L1-dcache-loads
     1,041,338,533      L1-dcache-load-misses     #    7.42% of all L1-dcache accesses
       252,679,422      LLC-loads
           653,611      LLC-stores

       1.294475718 seconds time elapsed

       9.965002000 seconds user
       0.012996000 seconds sys
```



```
erf_results_size1000_threads100.txt
Execution Time: 1.901147 seconds
# started on Thu Feb 13 14:49:07 2025


 Performance counter stats for './p 1000 100':

    37,636,530,023      cycles
    42,208,312,673      instructions              #    1.12  insn per cycle
     1,030,862,441      branches
       909,293,539      cache-references
        11,544,260      cache-misses              #    1.27% of all cache refs
             4,199      context-switches
                 0      block:block_rq_issue
                37      block:block_rq_complete

       1.747641513 seconds time elapsed

      13.715063000 seconds user
       0.025853000 seconds sys


# started on Thu Feb 13 14:49:08 2025


 Performance counter stats for './p 1000 100':

    14,056,670,249      L1-dcache-loads
     1,138,354,986      L1-dcache-load-misses     #    8.10% of all L1-dcache accesses
       290,659,163      LLC-loads
         1,070,125      LLC-stores

       1.742275510 seconds time elapsed

      13.659718000 seconds user
       0.015890000 seconds sys
```

2.MEMORY BOUND TASK



Analysis of Performance for Array Size 1000 with different number of Threads:

Key Observations:

1. Execution Time:
    - Increasing trend with threads: The execution time increases with the number of threads, which is unusual and suggests that the memory-bound program is encountering issues as it scales up with additional threads. This could be due to memory contention or inefficient thread synchronization (e.g., mutex locks), which cause threads to wait for each other, thus diminishing the performance benefits of parallelism.
2. Cache Misses:
    - Steady increase: Cache misses steadily increase as the number of threads grows. This suggests that the memory access pattern may not be optimized for cache efficiency. Higher cache misses result in increased memory access time, leading to performance degradation as the number of threads increases. This could be exacerbated by mutex contention, which blocks threads and may prevent the efficient use of CPU caches.
3. LLC Loads and Stores:
    - Increasing LLC accesses: With more threads, LLC (Last Level Cache) loads and stores also increase. This could be due to data being fetched from higher-level caches (L2, L3) as the L1 cache is not able to hold all the required data for multiple threads. This is a clear indication of inefficient memory locality. Additionally, mutex contention could cause more frequent cache access beyond the L1 cache, further deteriorating the performance.
4. L1-dcache Load Misses:
    - Rapid growth: The L1-dcache load misses grow rapidly as the thread count increases, further highlighting that the program is not using cache effectively. This could be due to poor data locality in memory access patterns, too many memory accesses to a larger dataset that exceeds the available cache, or mutex-induced delays where threads cannot keep the cache occupied efficiently. The mutex overhead increases contention for memory resources, thus worsening cache miss rates.

- The use of mutexes in the program has led to additional overhead in the form of thread blocking and increased context switches. Mutex contention limits parallel efficiency by preventing threads from accessing memory concurrently, causing delays in execution and increasing overall execution time. The mutex overhead could be a significant factor contributing to the observed increase in execution time and other performance metrics.

PERF RESULT FOR 8 THREADS VS 100 THREADS:

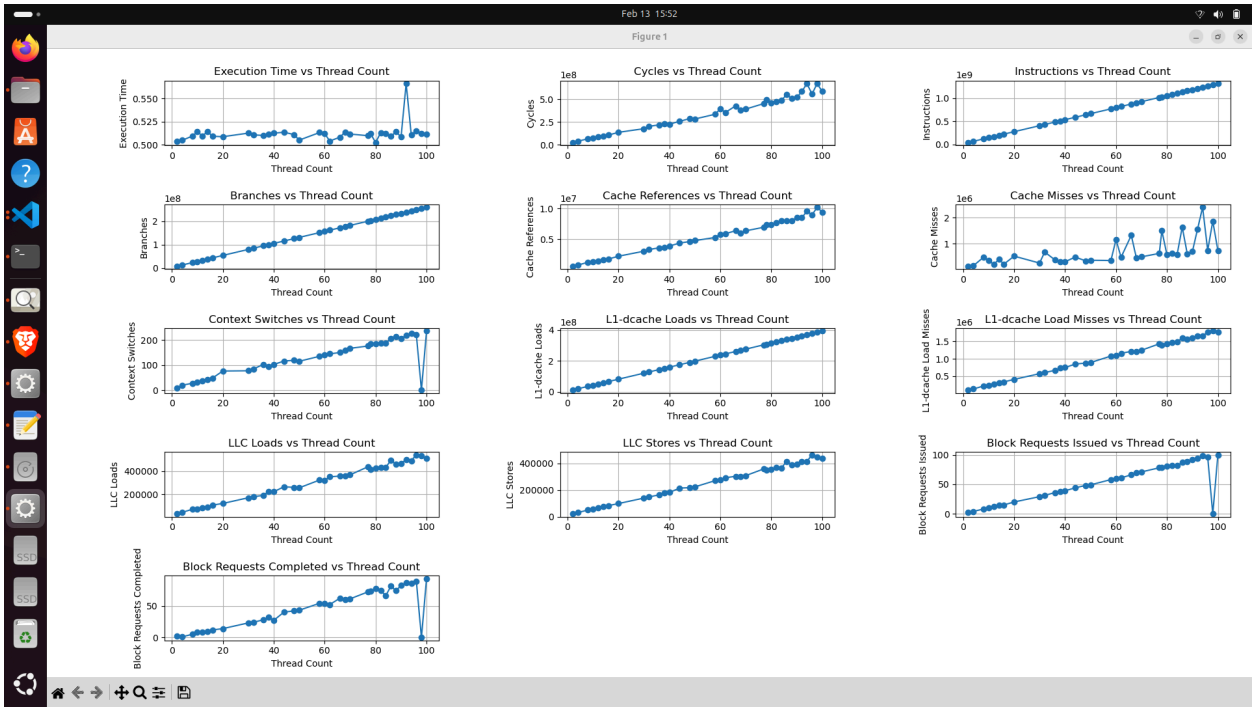
perfMB_results_size1000_threads8.txt

```
Maximum Element is : 998
Execution Time: 0.000226 seconds
# started on Thu Feb 13 15:04:08 2025


 Performance counter stats for './m 1000 8 134 955 74 372 595 960 961 575 659 74 869 822 381 275 504 311 854 521 346 236 757 710 901 899 424 696 837 368 481 437 426 943 800 780 677 277 548 747 681 374 96 417 711 595 6

                 0      block:block_rq_issue
                 0      block:block_rq_complete

       0.001770102 seconds time elapsed

       0.000000000 seconds user
       0.001774000 seconds sys


# started on Thu Feb 13 15:04:08 2025


 Performance counter stats for './m 1000 8 134 955 74 372 595 960 961 575 659 74 869 822 381 275 504 311 854 521 346 236 757 710 901 899 424 696 837 368 481 437 426 943 800 780 677 277 548 747 681 374 96 417 711 595 6

         5,031,702      cycles                                                  (41.52%)
         5,105,808      instructions              #    1.01  insn per cycle
           930,144      branches
           319,720      cache-references
            69,502      cache-misses              #   21.74% of all cache refs
            20,065      branch-misses             #    2.16% of all branches        (62.44%)
                 3      context-switches

       0.001715987 seconds time elapsed

       0.001832000 seconds user
       0.000000000 seconds sys


# started on Thu Feb 13 15:04:08 2025


 Performance counter stats for './m 1000 8 134 955 74 372 595 960 961 575 659 74 869 822 381 275 504 311 854 521 346 236 757 710 901 899 424 696 837 368 481 437 426 943 800 780 677 277 548 747 681 374 96 417 711 595 6

         1,200,762      L1-dcache-loads
            78,651      L1-dcache-load-misses     #    6.55% of all L1-dcache accesses
            21,911      LLC-loads
            18,849      LLC-stores

       0.001681393 seconds time elapsed

       0.000000000 seconds user
       0.001673000 seconds sys
```

perfMB_results_size1000_threads100.txt

```
Maximum Element is : 999
Execution Time: 0.002299 seconds
# started on Thu Feb 13 15:04:20 2025


 Performance counter stats for './m 1000 100 306 139 577 806 845 903 320 797 510 650 156 321 96 862 596 753 431 667 41 835 906 521 245 307 977 371 743 263 814 664 475 129 933 829 170 841 197 952 554 792 998 348 272 15

                 0      block:block_rq_issue
                 0      block:block_rq_complete

       0.004950339 seconds time elapsed

       0.000000000 seconds user
       0.005010000 seconds sys


# started on Thu Feb 13 15:04:20 2025


 Performance counter stats for './m 1000 100 306 139 577 806 845 903 320 797 510 650 156 321 96 862 596 753 431 667 41 835 906 521 245 307 977 371 743 263 814 664 475 129 933 829 170 841 197 952 554 792 998 348 272 15

        15,255,827      cycles                                                  (5.29%)
        14,328,060      instructions              #    0.94  insn per cycle      (59.49%)
         3,187,358      branches
         1,504,774      cache-references
           247,204      cache-misses              #   16.43% of all cache refs
            41,071      branch-misses             #    1.29% of all branches        (95.79%)
                 2      context-switches

       0.004680939 seconds time elapsed

       0.000000000 seconds user
       0.004853000 seconds sys


# started on Thu Feb 13 15:04:20 2025


 Performance counter stats for './m 1000 100 306 139 577 806 845 903 320 797 510 650 156 321 96 862 596 753 431 667 41 835 906 521 245 307 977 371 743 263 814 664 475 129 933 829 170 841 197 952 554 792 998 348 272 15

         3,851,550      L1-dcache-loads
           397,785      L1-dcache-load-misses     #   10.33% of all L1-dcache accesses
            98,214      LLC-loads
           159,277      LLC-stores

       0.006635657 seconds time elapsed

       0.000768000 seconds user
       0.006145000 seconds sys
```

3.IO BOUND TASK:

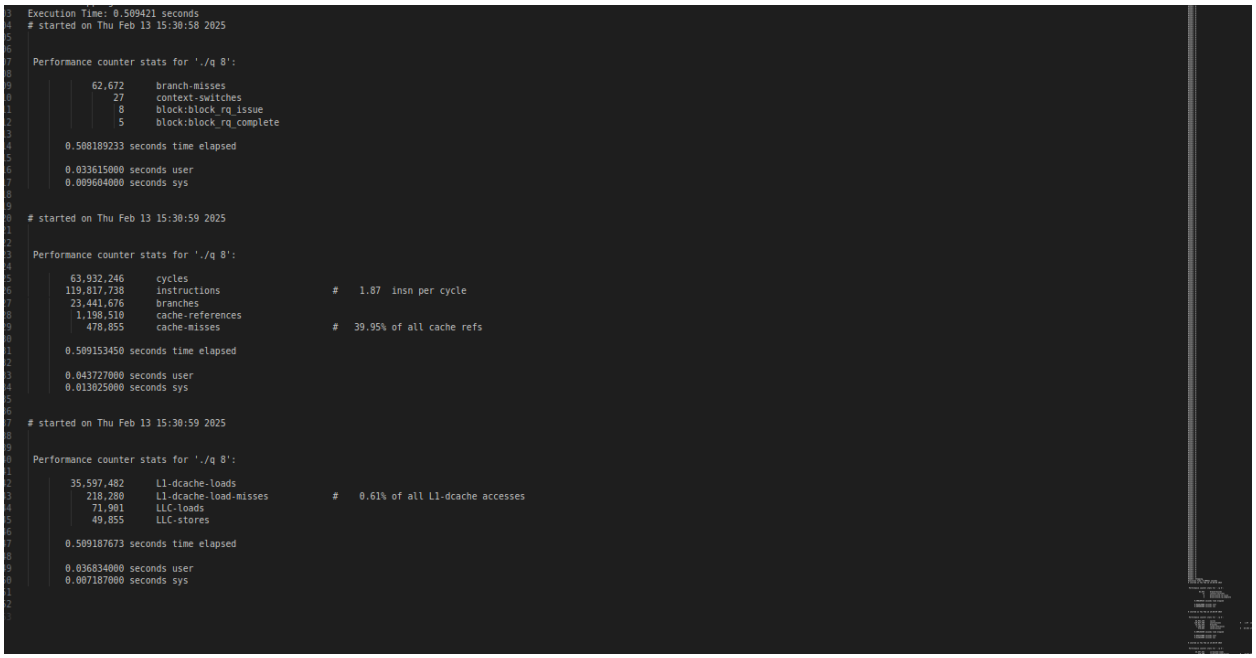


Key Observations:

1. Execution Time:
    - Increase with threads: The execution time increases significantly as the number of threads increases. This suggests that the program is experiencing thread contention, particularly with respect to waiting for I/O operations. When more threads are introduced, they are likely waiting on I/O, causing delays and making the program slower.
2. Cache Misses:
    - Moderate increase: While cache misses are typically more of a concern in CPU-bound programs, the cache miss rate in an I/O-bound program does increase slightly. This increase may still be due to memory contention, but the overall bottleneck here is not memory access speed, but rather the I/O latency and waiting time.
3. Context Switches:
    - Sharp increase: The increase in context switches as the number of threads rises further supports the idea that the program is experiencing thread contention. When many threads are trying to execute, the system performs more context switches, adding overhead. This often happens when threads are waiting on I/O, which leads to more scheduling work by the operating system.
4. LLC Loads and LLC Stores:
    - Linear increase: These metrics increase linearly with the number of threads, suggesting that as more threads are introduced, they are all accessing and interacting with memory, but the effect on the overall program is minimal since the main issue lies in I/O operations.
5. Block Requests Issued and Completed:
    - Increase but with fluctuations: Both block requests issued and block requests completed show an increasing trend with the number of threads. This is expected as more threads will make more I/O requests. The fluctuations may indicate inefficiencies in I/O handling or synchronization issues, especially since mutexes are used in the program to control thread access to shared resources.

- This program is clearly I/O-bound, and the increase in execution time with more threads is indicative of the program's struggle to handle more concurrent I/O requests. The increase in context switches and mutex usage suggests that the program is slowing down as more threads are introduced due to thread contention.

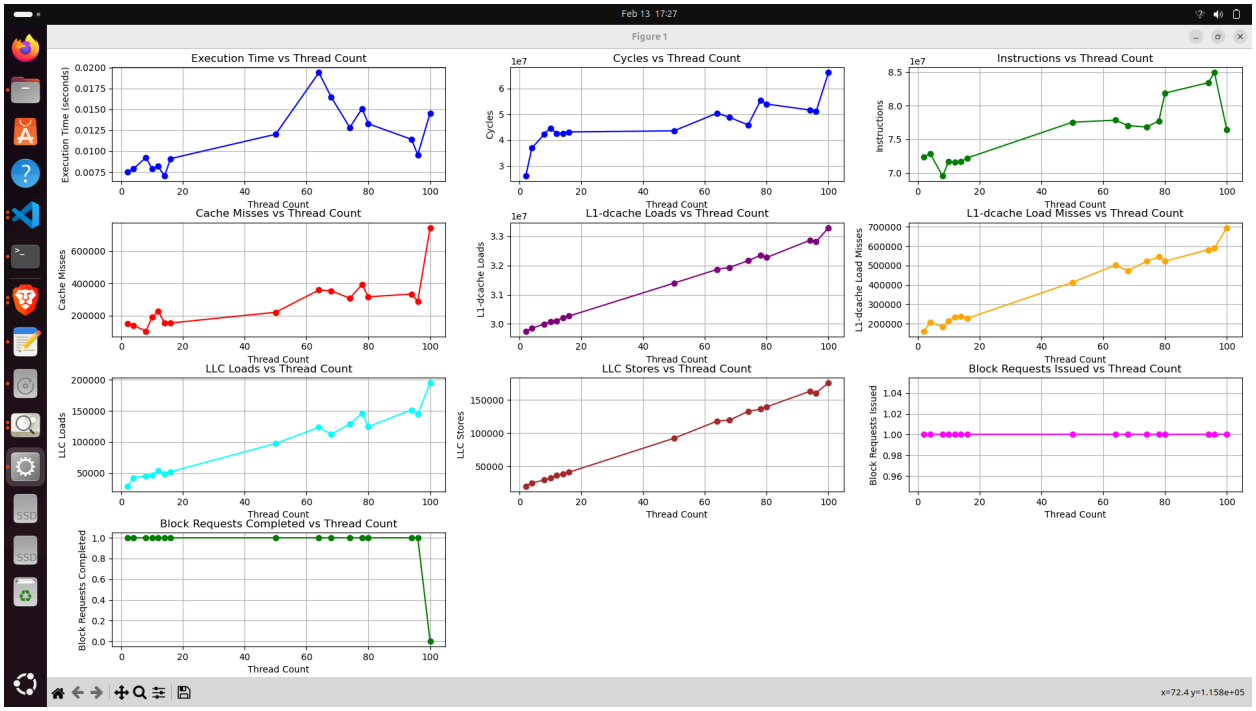
PERF RESULT FOR 8 THREADS VS 100 THREADS:



```
Execution Time: 0.509421 seconds
# started on Thu Feb 13 15:30:58 2025


 Performance counter stats for './q 8':

            62,672      branch-misses
               27      context-switches
                8      block:block_rq_issue
                5      block:block_rq_complete

       0.508189233 seconds time elapsed

       0.033615000 seconds user
       0.009604000 seconds sys


 # started on Thu Feb 13 15:30:59 2025


 Performance counter stats for './q 8':

        63,932,246      cycles
       119,817,738      instructions              #    1.87  insn per cycle
        23,441,676      branches
         1,198,510      cache-references
           478,855      cache-misses              #   39.95% of all cache refs

       0.509153450 seconds time elapsed

       0.043727000 seconds user
       0.013025000 seconds sys


 # started on Thu Feb 13 15:30:59 2025


 Performance counter stats for './q 8':

        35,597,482      L1-dcache-loads
           218,280      L1-dcache-load-misses     #    0.61% of all L1-dcache accesses
            71,901      LLC-loads
            49,855      LLC-stores

       0.509187673 seconds time elapsed

       0.036834000 seconds user
       0.007187000 seconds sys
```

```
Execution Time: 0.511631 seconds
# started on Thu Feb 13 15:32:25 2025


 Performance counter stats for './q 100':

           459,350      branch-misses
               236      context-switches
               100      block:block_rq_issue
                93      block:block_rq_complete

       0.505669999 seconds time elapsed

       0.147842000 seconds user
       0.023717000 seconds sys


 # started on Thu Feb 13 15:32:25 2025


 Performance counter stats for './q 100':

       583,700,472      cycles
     1,315,543,386      instructions              #    2.25  insn per cycle
       258,920,231      branches
         9,450,816      cache-references
           739,591      cache-misses              #    7.83% of all cache refs

       0.508996311 seconds time elapsed

       0.170450000 seconds user
       0.026349000 seconds sys


 # started on Thu Feb 13 15:32:26 2025


 Performance counter stats for './q 100':

       392,557,609      L1-dcache-loads
         1,770,873      L1-dcache-load-misses     #    0.45% of all L1-dcache accesses
           510,780      LLC-loads
           438,436      LLC-stores

       0.506709403 seconds time elapsed

       0.127871000 seconds user
       0.020696000 seconds sys
```

4.MIXED WORKLOAD:



Analysis of Performance for Matrix Size 1000 with different number of Threads:

Improved performance with threads: The decrease in execution time with more threads is a strong indication that parallelism is beneficial in the case of this mixed workload. The program can split its CPU-bound, memory-bound, and I/O-bound tasks across multiple threads, which leads to performance improvements.

Despite the performance improvements with more threads, the increasing cache misses and L1-dcache load misses signal that the program's memory access patterns could be further optimized for better cache utilization. It seems that the program is experiencing memory contention as more threads are added.

The sharp drop in block requests completed indicates that the program may face mutex contention or other synchronization-related issues when more threads are introduced. Proper synchronization techniques could mitigate this bottleneck and improve performance at higher thread counts.

The program scales well with increased threads initially but starts to show diminishing returns as more threads are added. This suggests that the system might not have enough resources to handle a large number of threads efficiently, especially if memory contention and synchronization overhead are not addressed.

PERF RESULT FOR 8 THREADS VS 100 THREADS:

```
Execution Time: 0.009231 seconds
# started on Thu Feb 13 16:52:24 2025


 Performance counter stats for './k 100 4.txt 8':

        42,274,582      cycles                                                  (78.81%)
        69,529,637      instructions              #    1.64  insn per cycle     (84.97%)
         5,458,278      branches                                                (65.75%)
           791,889      cache-references                                        (80.43%)
           103,378      cache-misses              #   13.05% of all cache refs  (95.08%)
            42,021      branch-misses             #    0.77% of all branches    (91.87%)
                14      context-switches
                 1      block:block_rq_issue
                 1      block:block_rq_complete

       0.008335058 seconds time elapsed

       0.018318000 seconds user
       0.002497000 seconds sys


# started on Thu Feb 13 16:52:24 2025


 Performance counter stats for './k 100 4.txt 8':

        29,997,736      L1-dcache-loads
           183,034      L1-dcache-load-misses     #    0.61% of all L1-dcache accesses
            44,841      LLC-loads
            29,400      LLC-stores

       0.010716379 seconds time elapsed

       0.016146000 seconds user
       0.002201000 seconds sys
```

```
Execution Time: 0.014560 seconds
# started on Thu Feb 13 16:52:26 2025


 Performance counter stats for './k 100 4.txt 100':

        66,076,591      cycles                                                  (74.27%)
        76,415,106      instructions              #    1.16  insn per cycle     (91.75%)
         6,817,998      branches                                                (89.05%)
         3,972,691      cache-references                                        (91.08%)
           745,812      cache-misses              #   18.77% of all cache refs  (87.26%)
           110,240      branch-misses             #    1.62% of all branches    (85.19%)
                72      context-switches
                 1      block:block_rq_issue
                 0      block:block_rq_complete

       0.046535722 seconds time elapsed

       0.027758000 seconds user
       0.033174000 seconds sys


# started on Thu Feb 13 16:52:26 2025


 Performance counter stats for './k 100 4.txt 100':

        33,292,351      L1-dcache-loads
           695,397      L1-dcache-load-misses     #    2.09% of all L1-dcache accesses
           195,376      LLC-loads
           175,668      LLC-stores

       0.034067992 seconds time elapsed

       0.027879000 seconds user
       0.025618000 seconds sys
```