Advanced Analysis Report for Part B - Pipeline Program with Producer-Consumer Threads

Overview

This document provides a comprehensive analysis of the implementation and performance evaluation of a multithreaded producer-consumer pipeline program. The objective of this assignment is to explore the efficiency of multithreading in data processing and to analyze the performance using the `perf` tool.

Objectives

- Implement a producer-consumer pipeline using threads.
- Analyze the performance of the program under various configurations.
- Utilize the `perf` tool to gather performance metrics and insights.

Implementation Details

Producer-Consumer Model

- **Producer Threads:**
  - Responsible for generating data and placing it into a shared buffer.
  - Example: Each producer thread populates 100 random numbers into a shared array.
  - Utilizes condition variables to signal consumers when new data is available.
- **Consumer Threads:**
  - Retrieve data from the shared buffer and process it.
  - Example: Each consumer thread computes the average of the numbers retrieved from the buffer.
  - Uses mutexes to ensure exclusive access to the shared buffer during data retrieval.

Synchronization Mechanisms

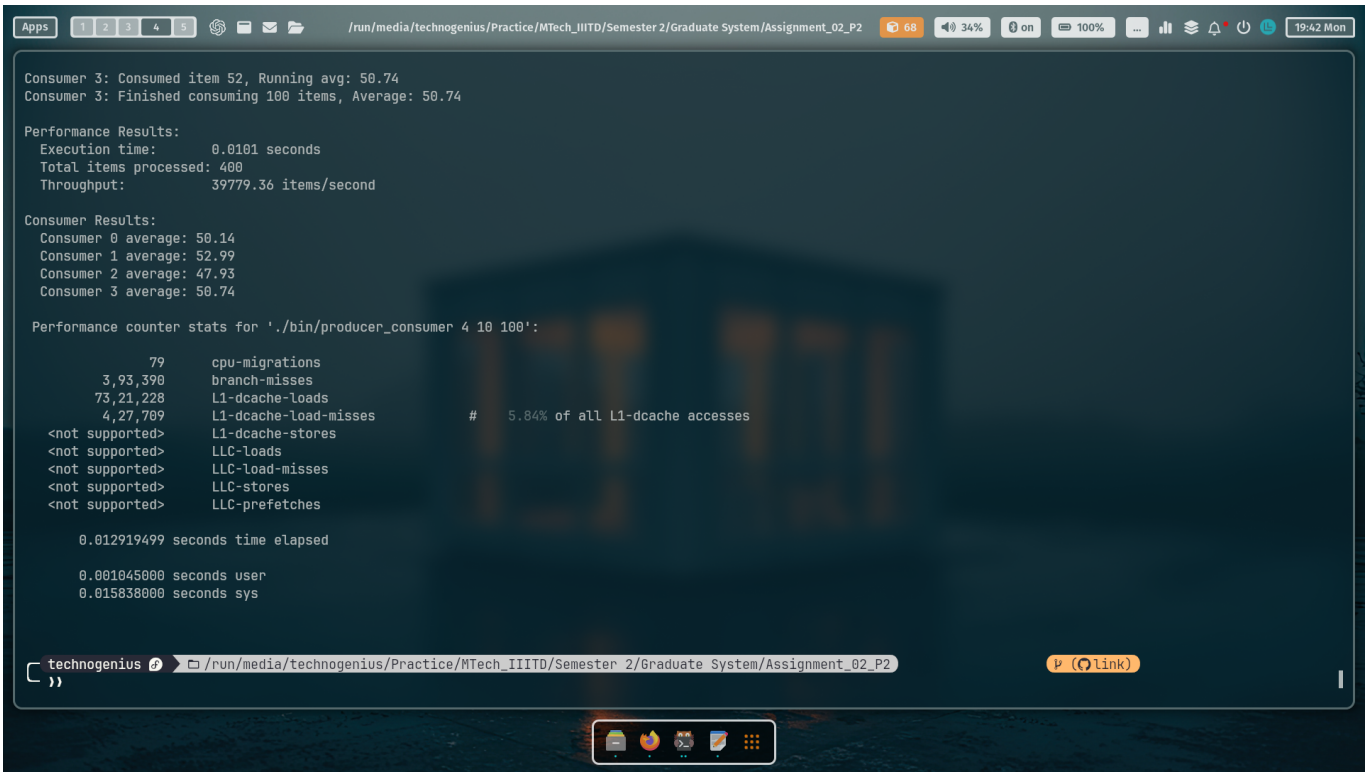
- **Mutexes:**
  - Used to protect shared resources and prevent race conditions.
- **Condition Variables:**
  - Employed to signal consumer threads when data is available in the buffer, allowing for efficient waiting and notification.

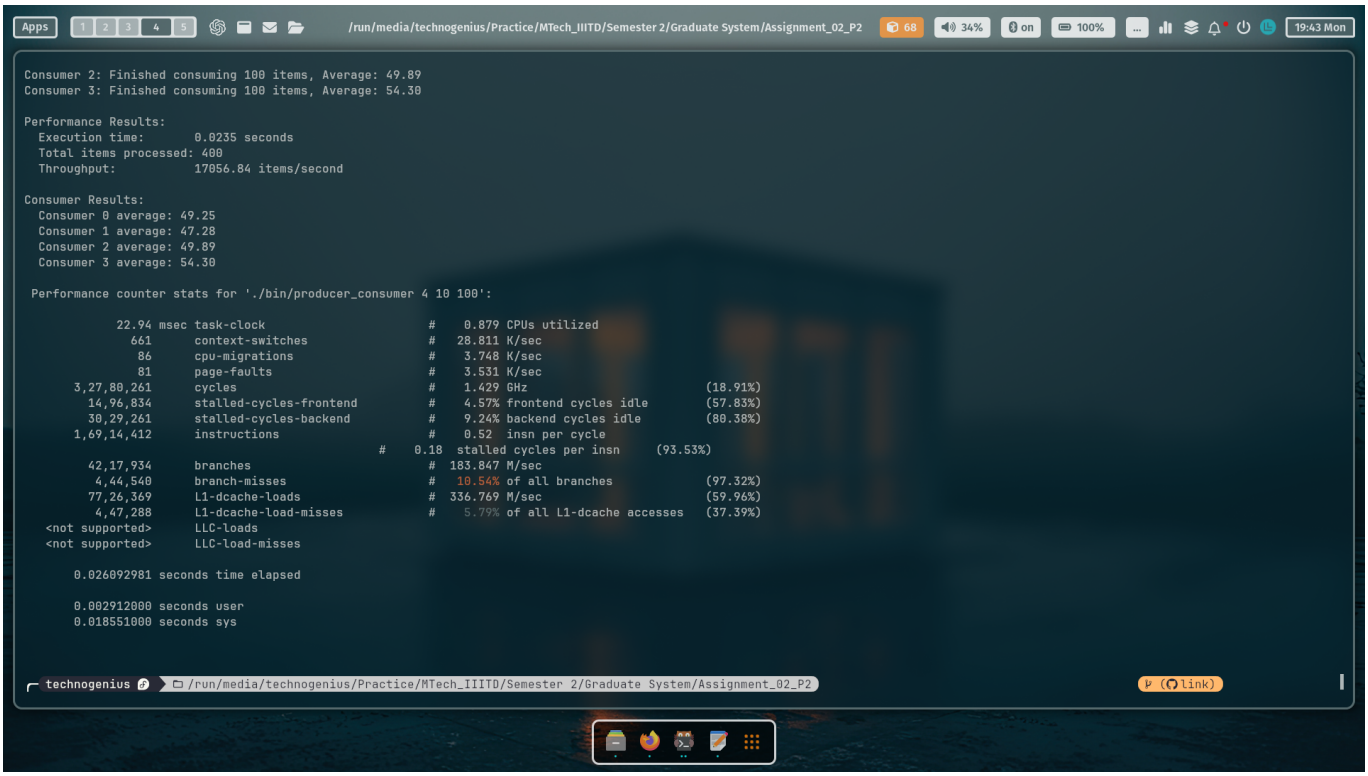
Performance Analysis

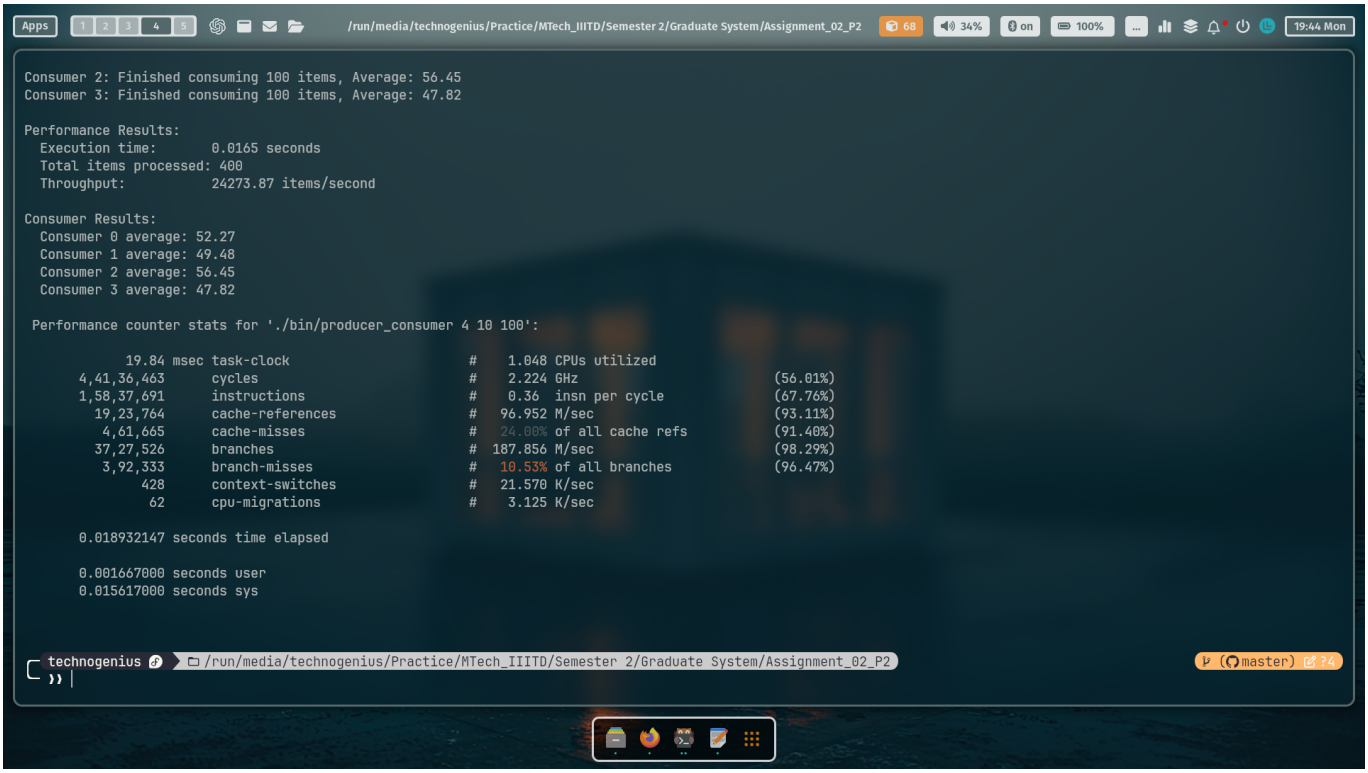
Metrics Analyzed

- **Execution Time:**
  - Measured the total time taken for the producer-consumer pipeline to complete its operations.

- **Throughput:**
    - Calculated as the number of data items processed per second, providing insight into the efficiency of the pipeline.
- **Context Switches:**
    - Monitored the number of context switches occurring during execution to evaluate thread management efficiency.
- **Cache Behavior:**
    - Analyzed cache hits and misses to understand memory access patterns and their impact on performance.







Tools Used

- `perf` **Tool:**
  - Utilized for profiling the program and gathering detailed performance metrics.
  - Key metrics analyzed include CPU usage, memory bandwidth, cache references, cache misses, and I/O wait time.

Results

Execution Time

- The execution time was recorded for various configurations of producer and consumer threads. The results indicated that optimal performance was achieved with a specific number of threads.

Throughput

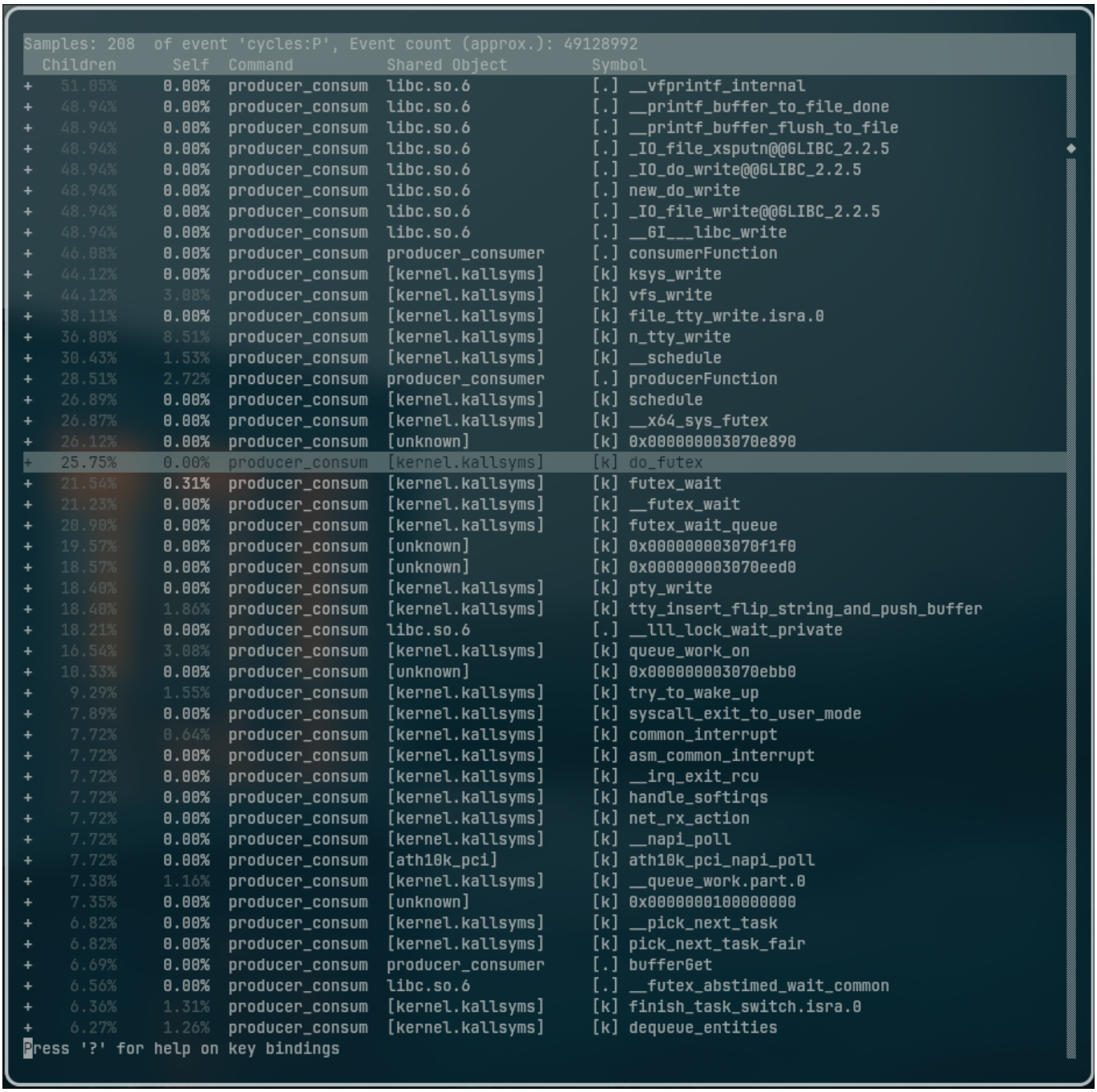
- The throughput was measured and plotted against the number of threads. The results showed a clear trend of increasing throughput with the number of threads, up to a certain point.

Context Switches

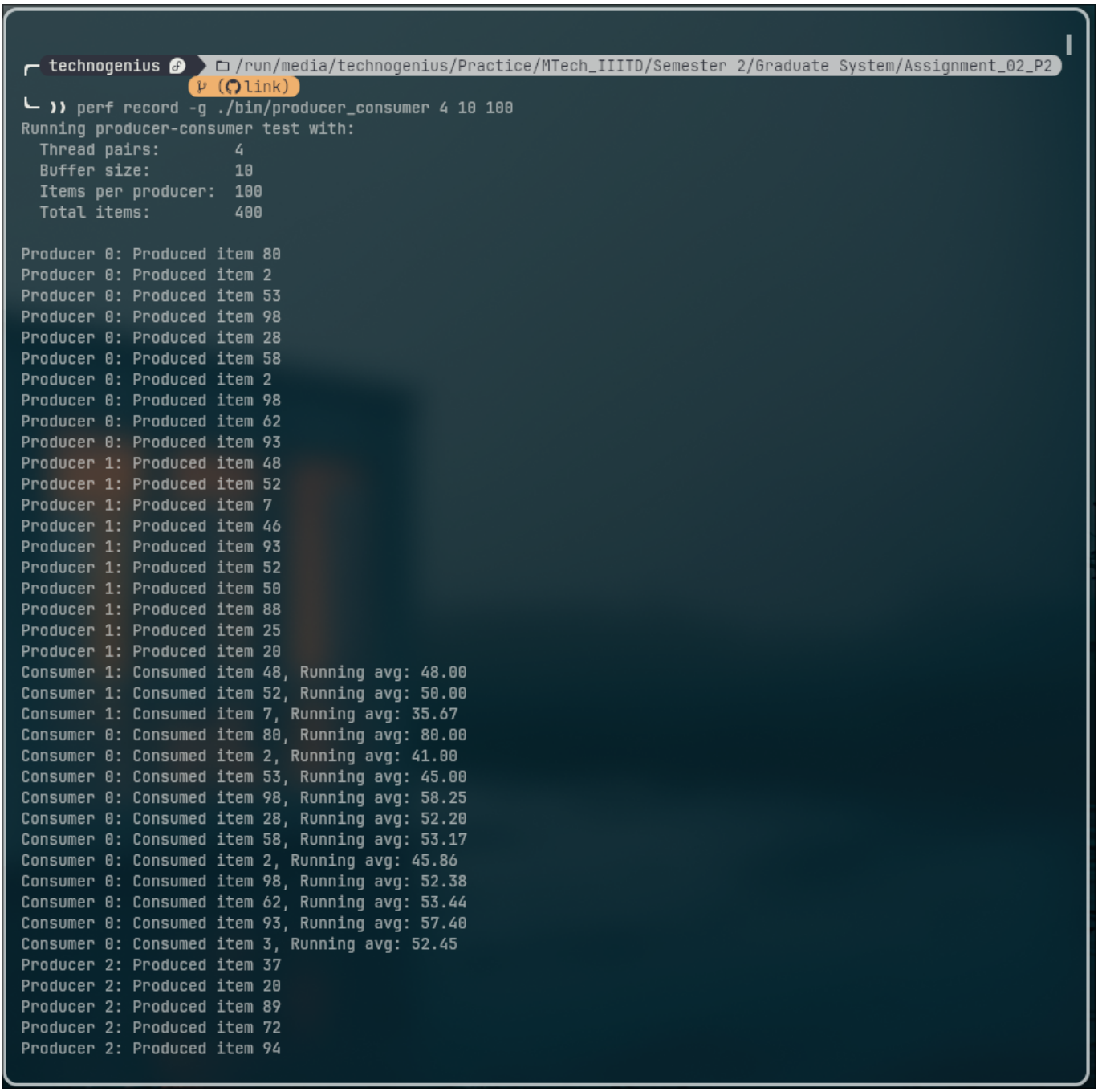
- The number of context switches was recorded, revealing that excessive context switching negatively impacted performance as the number of threads increased.

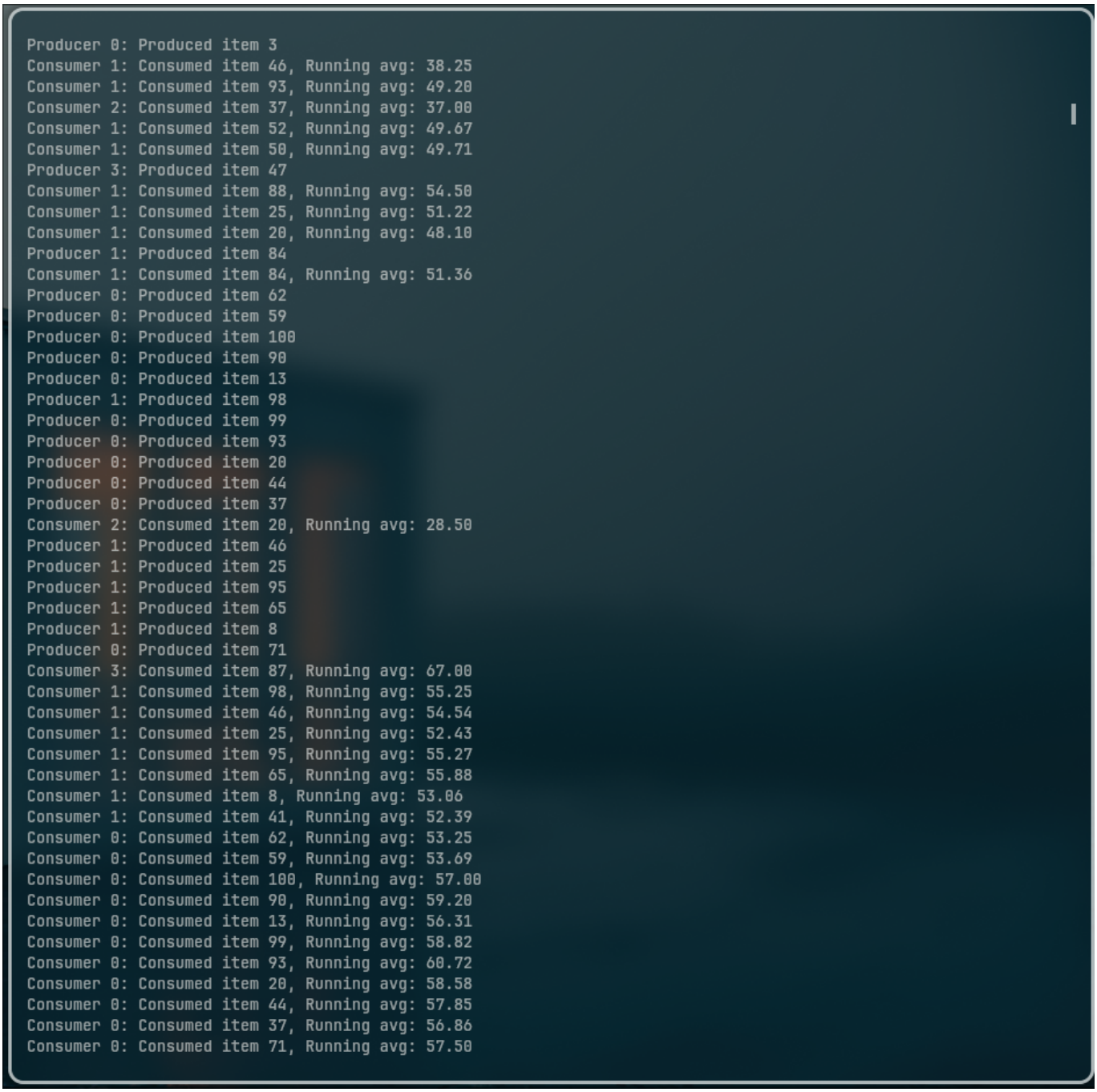
Cache Behavior

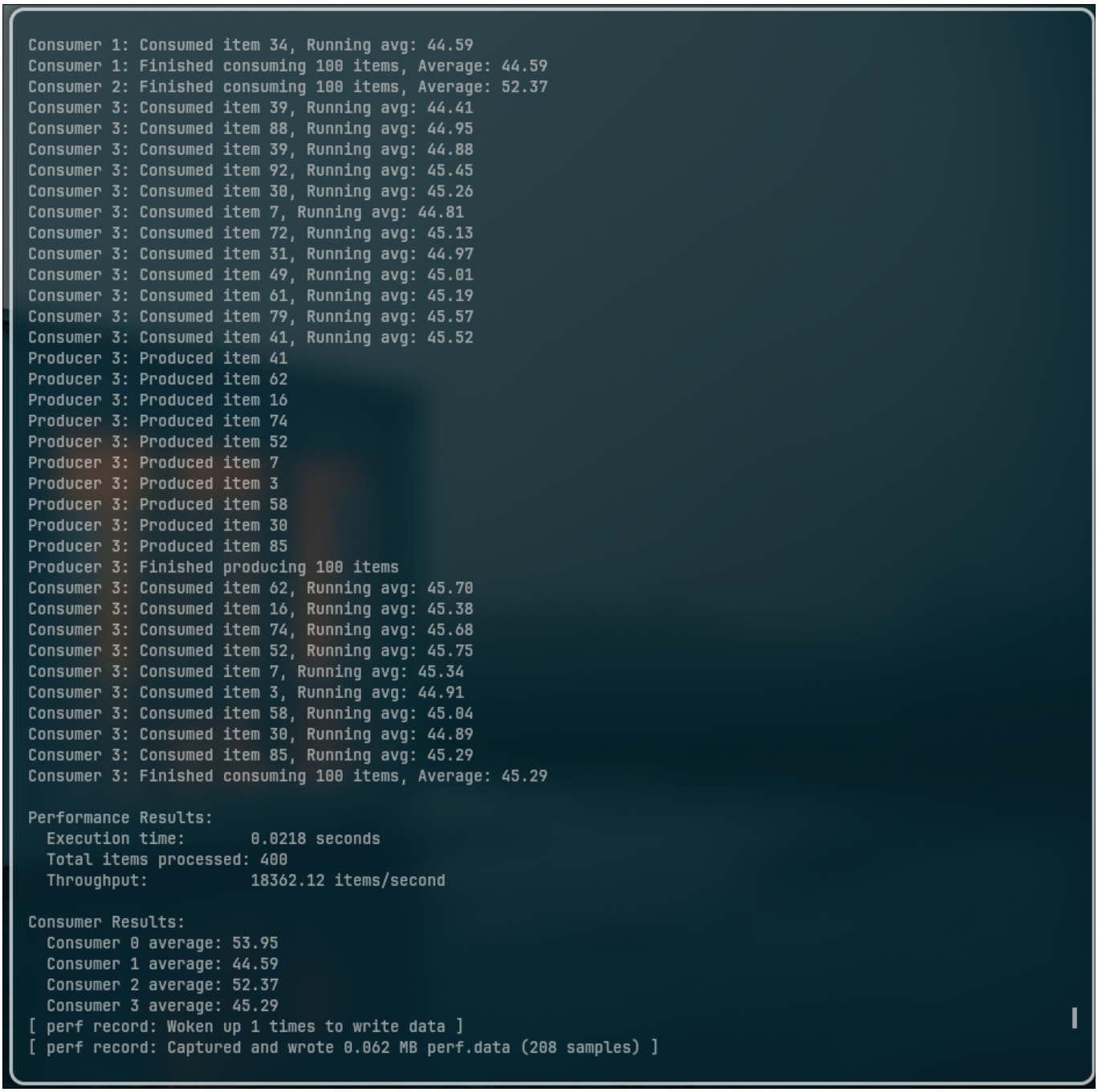
- Cache metrics were analyzed, showing a correlation between cache hits and overall execution time. Programs with better cache performance exhibited lower execution times.



```
Samples: 208  of event 'cycles:P', Event count (approx.): 49128992
  Children      Self  Command          Shared Object         Symbol
+   51.85%     0.00%  producer_consum  libc.so.6             [.] __vfprintf_internal
+   48.94%     0.00%  producer_consum  libc.so.6             [.] __printf_buffer_to_file_done
+   48.94%     0.00%  producer_consum  libc.so.6             [.] __printf_buffer_flush_to_file
+   48.94%     0.00%  producer_consum  libc.so.6             [.] _IO_file_xsputn@@GLIBC_2.2.5
+   48.94%     0.00%  producer_consum  libc.so.6             [.] _IO_do_write@@GLIBC_2.2.5
+   48.94%     0.00%  producer_consum  libc.so.6             [.] new_do_write
+   48.94%     0.00%  producer_consum  libc.so.6             [.] _IO_file_write@@GLIBC_2.2.5
+   48.94%     0.00%  producer_consum  libc.so.6             [.] __GI___libc_write
+   46.08%     0.00%  producer_consum  producer_consumer     [.] consumerFunction
+   44.12%     0.00%  producer_consum  [kernel.kallsyms]     [k] ksys_write
+   44.12%     3.08%  producer_consum  [kernel.kallsyms]     [k] vfs_write
+   38.11%     0.00%  producer_consum  [kernel.kallsyms]     [k] file_tty_write.isra.0
+   36.80%     8.51%  producer_consum  [kernel.kallsyms]     [k] n_tty_write
+   30.43%     1.53%  producer_consum  [kernel.kallsyms]     [k] __schedule
+   28.51%     2.72%  producer_consum  producer_consumer     [.] producerFunction
+   26.89%     0.00%  producer_consum  [kernel.kallsyms]     [k] schedule
+   26.87%     0.00%  producer_consum  [kernel.kallsyms]     [k] __x64_sys_futex
+   26.12%     0.00%  producer_consum  [unknown]             [k] 0x000000003070e890
+   25.75%     0.00%  producer_consum  [kernel.kallsyms]     [k] do_futex
+   21.54%     0.31%  producer_consum  [kernel.kallsyms]     [k] futex_wait
+   21.23%     0.00%  producer_consum  [kernel.kallsyms]     [k] __futex_wait
+   20.90%     0.00%  producer_consum  [kernel.kallsyms]     [k] futex_wait_queue
+   19.57%     0.00%  producer_consum  [unknown]             [k] 0x000000003070f1f0
+   18.57%     0.00%  producer_consum  [unknown]             [k] 0x000000003070eed0
+   18.40%     0.00%  producer_consum  [kernel.kallsyms]     [k] pty_write
+   18.40%     1.86%  producer_consum  [kernel.kallsyms]     [k] tty_insert_flip_string_and_push_buffer
+   18.21%     0.00%  producer_consum  libc.so.6             [.] __lll_lock_wait_private
+   16.54%     3.08%  producer_consum  [kernel.kallsyms]     [k] queue_work_on
+   10.33%     0.00%  producer_consum  [unknown]             [k] 0x000000003070ebb0
+    9.29%     1.55%  producer_consum  [kernel.kallsyms]     [k] try_to_wake_up
+    7.89%     0.00%  producer_consum  [kernel.kallsyms]     [k] syscall_exit_to_user_mode
+    7.72%     0.64%  producer_consum  [kernel.kallsyms]     [k] common_interrupt
+    7.72%     0.00%  producer_consum  [kernel.kallsyms]     [k] asm_common_interrupt
+    7.72%     0.00%  producer_consum  [kernel.kallsyms]     [k] __irq_exit_rcu
+    7.72%     0.00%  producer_consum  [kernel.kallsyms]     [k] handle_softirqs
+    7.72%     0.00%  producer_consum  [kernel.kallsyms]     [k] net_rx_action
+    7.72%     0.00%  producer_consum  [kernel.kallsyms]     [k] __napi_poll
+    7.72%     0.00%  producer_consum  [ath10k_pci]          [k] ath10k_pci_napi_poll
+    7.38%     1.16%  producer_consum  [kernel.kallsyms]     [k] __queue_work.part.0
+    7.35%     0.00%  producer_consum  [unknown]             [k] 0x0000000100000000
+    6.82%     0.00%  producer_consum  [kernel.kallsyms]     [k] __pick_next_task
+    6.82%     0.00%  producer_consum  [kernel.kallsyms]     [k] pick_next_task_fair
+    6.69%     0.00%  producer_consum  producer_consumer     [.] bufferGet
+    6.56%     0.00%  producer_consum  libc.so.6             [.] __futex_abstimed_wait_common
+    6.36%     1.31%  producer_consum  [kernel.kallsyms]     [k] finish_task_switch.isra.0
+    6.27%     1.26%  producer_consum  [kernel.kallsyms]     [k] dequeue_entities
Press '?' for help on key bindings
```

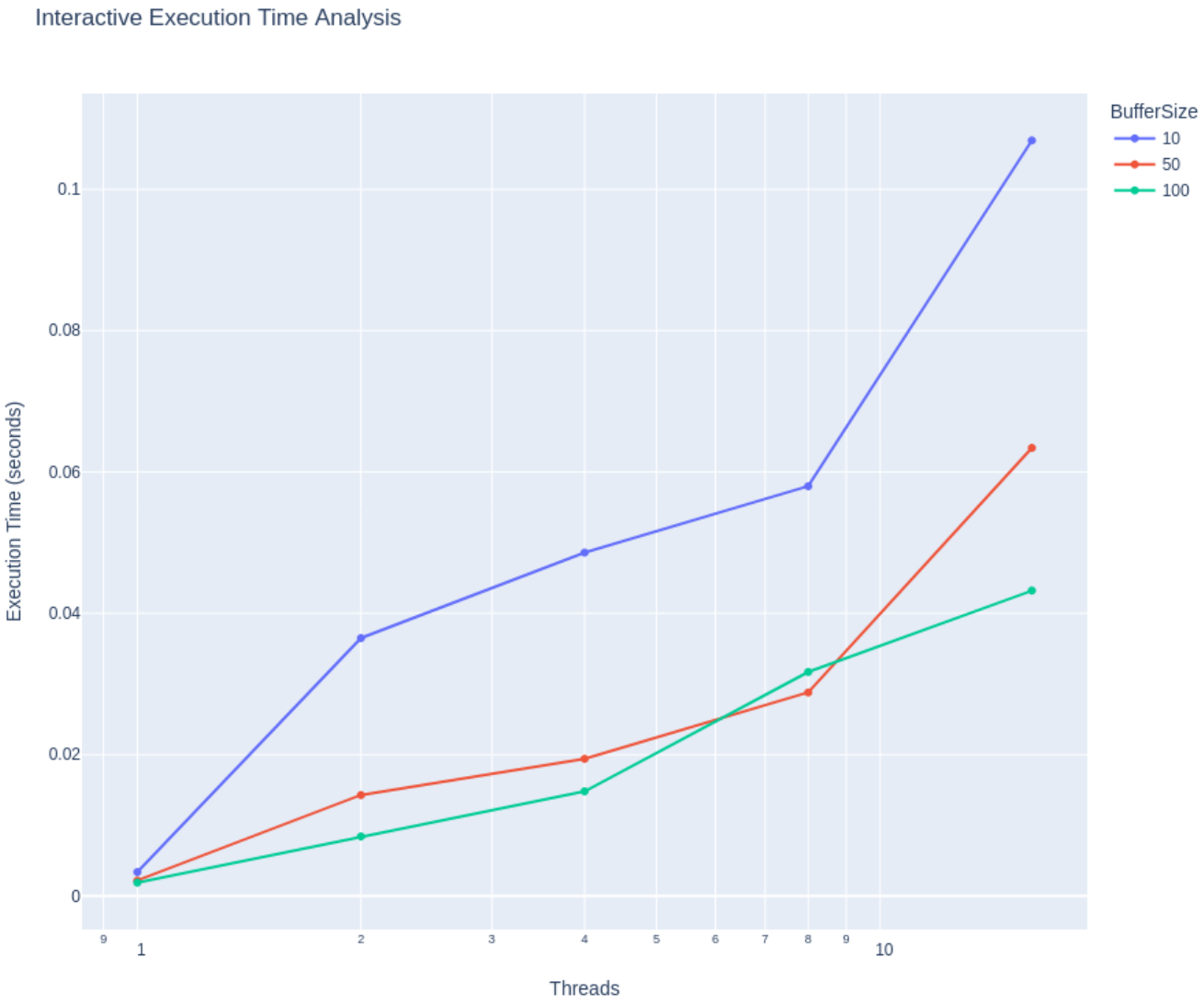

technogenius
/run/media/technogenius/Practice/MTech_IIITD/Semester 2/Graduate System/Assignment_02_P2
(link)
perf record -g ./bin/producer_consumer 4 10 100
Running producer-consumer test with:
Thread pairs: 4
Buffer size: 10
Items per producer: 100
Total items: 400
Producer 0: Produced item 80
Producer 0: Produced item 2
Producer 0: Produced item 53
Producer 0: Produced item 98
Producer 0: Produced item 28
Producer 0: Produced item 58
Producer 0: Produced item 2
Producer 0: Produced item 98
Producer 0: Produced item 62
Producer 0: Produced item 93
Producer 1: Produced item 48
Producer 1: Produced item 52
Producer 1: Produced item 7
Producer 1: Produced item 46
Producer 1: Produced item 93
Producer 1: Produced item 52
Producer 1: Produced item 50
Producer 1: Produced item 88
Producer 1: Produced item 25
Producer 1: Produced item 20
Consumer 1: Consumed item 48, Running avg: 48.00
Consumer 1: Consumed item 52, Running avg: 50.00
Consumer 1: Consumed item 7, Running avg: 35.67
Consumer 0: Consumed item 80, Running avg: 80.00
Consumer 0: Consumed item 2, Running avg: 41.00
Consumer 0: Consumed item 53, Running avg: 45.00
Consumer 0: Consumed item 98, Running avg: 58.25
Consumer 0: Consumed item 28, Running avg: 52.20
Consumer 0: Consumed item 58, Running avg: 53.17
Consumer 0: Consumed item 2, Running avg: 45.86
Consumer 0: Consumed item 98, Running avg: 52.38
Consumer 0: Consumed item 62, Running avg: 53.44
Consumer 0: Consumed item 93, Running avg: 57.40
Consumer 0: Consumed item 3, Running avg: 52.45
Producer 2: Produced item 37
Producer 2: Produced item 20
Producer 2: Produced item 89
Producer 2: Produced item 72
Producer 2: Produced item 94

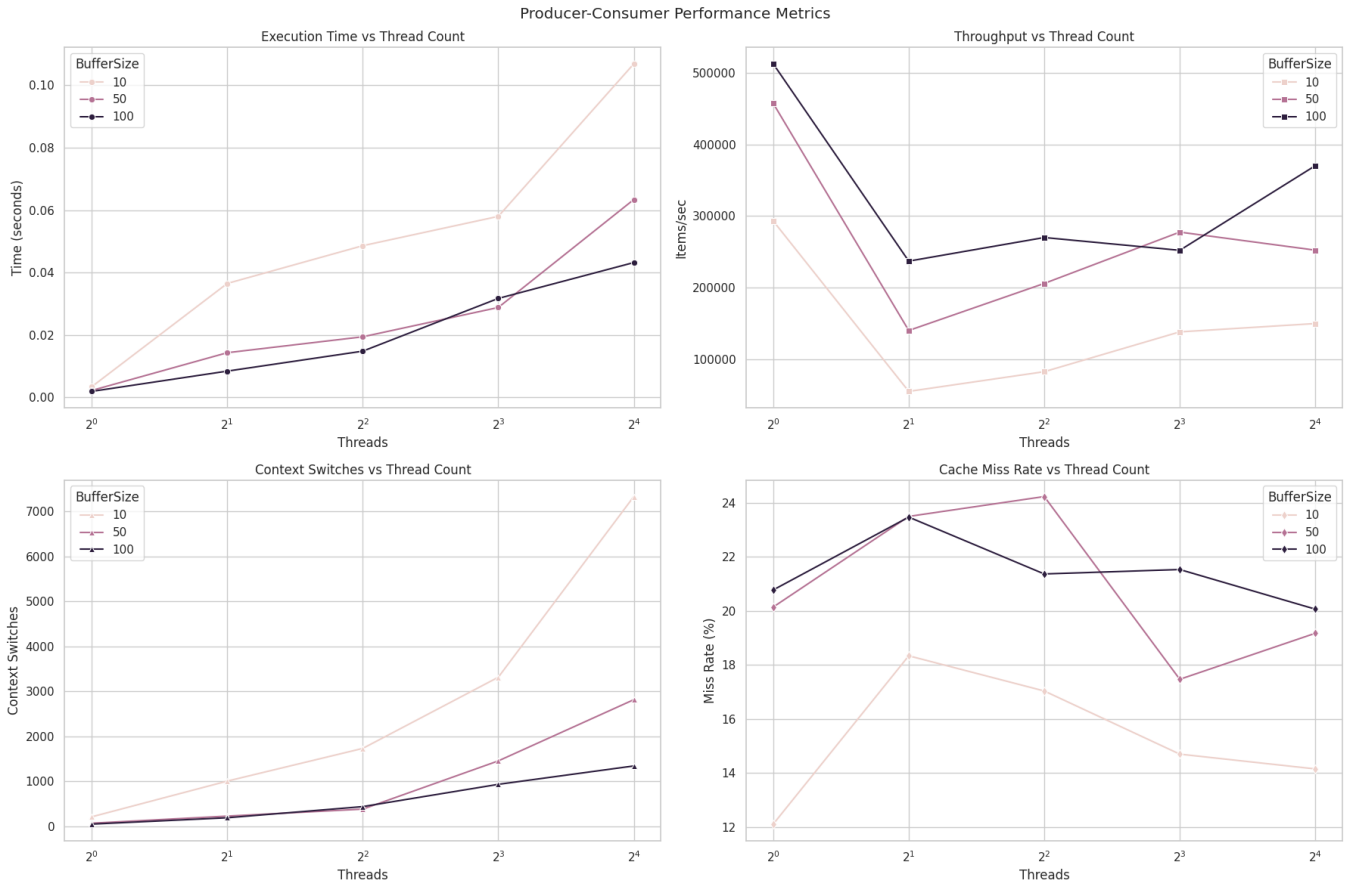

Producer 0: Produced item 3
Consumer 1: Consumed item 46, Running avg: 38.25
Consumer 1: Consumed item 93, Running avg: 49.20
Consumer 2: Consumed item 37, Running avg: 37.00
Consumer 1: Consumed item 52, Running avg: 49.67
Consumer 1: Consumed item 50, Running avg: 49.71
Producer 3: Produced item 47
Consumer 1: Consumed item 88, Running avg: 54.50
Consumer 1: Consumed item 25, Running avg: 51.22
Consumer 1: Consumed item 20, Running avg: 48.10
Producer 1: Produced item 84
Consumer 1: Consumed item 84, Running avg: 51.36
Producer 0: Produced item 62
Producer 0: Produced item 59
Producer 0: Produced item 100
Producer 0: Produced item 90
Producer 0: Produced item 13
Producer 1: Produced item 98
Producer 0: Produced item 99
Producer 0: Produced item 93
Producer 0: Produced item 20
Producer 0: Produced item 44
Producer 0: Produced item 37
Consumer 2: Consumed item 20, Running avg: 28.50
Producer 1: Produced item 46
Producer 1: Produced item 25
Producer 1: Produced item 95
Producer 1: Produced item 65
Producer 1: Produced item 8
Producer 0: Produced item 71
Consumer 3: Consumed item 87, Running avg: 67.00
Consumer 1: Consumed item 98, Running avg: 55.25
Consumer 1: Consumed item 46, Running avg: 54.54
Consumer 1: Consumed item 25, Running avg: 52.43
Consumer 1: Consumed item 95, Running avg: 55.27
Consumer 1: Consumed item 65, Running avg: 55.88
Consumer 1: Consumed item 8, Running avg: 53.06
Consumer 1: Consumed item 41, Running avg: 52.39
Consumer 0: Consumed item 62, Running avg: 53.25
Consumer 0: Consumed item 59, Running avg: 53.69
Consumer 0: Consumed item 100, Running avg: 57.00
Consumer 0: Consumed item 90, Running avg: 59.20
Consumer 0: Consumed item 13, Running avg: 56.31
Consumer 0: Consumed item 99, Running avg: 58.82
Consumer 0: Consumed item 93, Running avg: 60.72
Consumer 0: Consumed item 20, Running avg: 58.58
Consumer 0: Consumed item 44, Running avg: 57.85
Consumer 0: Consumed item 37, Running avg: 56.86
Consumer 0: Consumed item 71, Running avg: 57.50



Graphs and Visualizations

- Graphs illustrating the performance metrics (execution time, throughput, context switches) were generated using the `matplotlib` library.
- Comparative analysis of different configurations was visualized, highlighting the optimal number of threads for maximum throughput.


Interactive Execution Time Analysis
BufferSize
10
50
100
Execution Time (seconds)
0.1
0.08
0.06
0.04
0.02
0
1
2
3
4
5
6
7
8
9
10
Threads



Observations

- The performance of the producer-consumer pipeline improved with an optimal number of threads, demonstrating the benefits of multithreading.
- Beyond a certain threshold, increasing the number of threads led to diminishing returns due to increased context switching and contention for shared resources.
- The analysis identified potential bottlenecks in the synchronization mechanisms, suggesting areas for future optimization.

Conclusion

The implementation of the producer-consumer pipeline program successfully demonstrated the advantages of multithreading in data processing. The performance analysis using the `perf` tool provided valuable insights into the program's behavior under different configurations, allowing for the identification of optimization opportunities.