SVG Images Collection

Generated Script

August 13, 2024

Contents

1 Discussion **1**

1.1 Model Characteristics and Performance 1

1.2 Configuration and Platform Considerations 1

1.3 Performance Factors and Optimization 1

1.3.1 Event Processing Time 1

1.4 Performance Optimization: Rollbacks and State Saving 2

1.4.1 Introduction to Rollbacks and State Saving 2

1.4.2 State Saving and Rollback Relationship 2

1.5 Branch Implementations 2

1.5.1 Master Branch 2

1.5.2 Fossil Branch 2

1.5.3 SIMD Branch 3

1.5.4 Hashing Branch 3

1.5.5 Relaxed Compare Branch 3

1.5.6 Unified Queue Branch 3

1.6 GVT Implementation Comparison 4

1.7 Future Work 4

2 Label Definitions **5**

3 fossil 6

4 fossillocal 11

5 gvt_comparison 16

6 gvt_comparison_local 21

7 hashing 26

8 hashing_local 31

9 SIMD 36

10 SIMD_local 41

11 SIMDoptiplex 46

12 Top Performers 51

13 unified 53

14 unified_local 58

1 Discussion

In this section, we discuss the various aspects of our research on the WARPED simulation kernel, including model characteristics, configuration considerations, and the performance implications of different branch implementations.

1.1 Model Characteristics and Performance

The choice of simulation model significantly influences the performance of the WARPED kernel. We examined several models, including synthetic models like Traffic and PCS, as well as real-world models such as Epidemic-10k-ba and Epidemic-100k-ba [3]. These models vary in their computational requirements and event distribution patterns, affecting the overall simulation performance.

The Epidemic model, for instance, simulates disease outbreak phenomena using a combination of reaction and diffusion processes [2]. It incorporates complex interactions between entities, network structures (e.g., Watts-Strogatz [4] and Barabasi-Albert [1] graphs), and geographical diffusion, making it particularly challenging for parallel discrete event simulation (PDES) systems.

1.2 Configuration and Platform Considerations

While testing proposed changes to the WARPED kernel, it is crucial to determine the scenarios and platforms where they perform optimally. Factors such as instruction sets and specialized hardware can significantly impact performance. As noted earlier, the model itself plays a critical role in determining whether a particular configuration of WARPED works efficiently and quickly.

The debate between GPU and CPU computing for PDES algorithms is ongoing. While GPUs excel at performing simple tasks quickly, CPUs are better suited for handling complex, lengthy computations often required in sophisticated simulations at the Logical Process (LP) level. The choice between platforms depends on the specific requirements of the model and the nature of the computations involved.

1.3 Performance Factors and Optimization

Several factors influence the performance of WARPED simulations:

1.3.1 Event Processing Time

Models like PCS have longer processing times for events at each LP. The grid size, representing the number of LPs, determines the scope of the simulation. It's important to note that the maximum simulation time (max-sim-time) represents the timestamp of the last generated event, not the actual runtime of the simulation.

1.4 Performance Optimization: Rollbacks and State Saving

1.4.1 Introduction to Rollbacks and State Saving

In Parallel Discrete Event Simulation (PDES), the efficiency of rollbacks and state saving mechanisms plays a crucial role in overall performance. This section explores the intricate relationship between state-saving periods and rollback lengths, and presents a mathematical model to optimize these parameters.

1.4.2 State Saving and Rollback Relationship

State saving and rollbacks are interdependent processes in PDES. The frequency of state saving directly impacts the length of potential rollbacks. More frequent state saves reduce the average rollback length but increase overhead, while less frequent saves lead to potentially longer rollbacks but reduce save overhead.

1.5 Branch Implementations

We explored several branch implementations for the WARPED kernel, each focusing on different optimization strategies:

1.5.1 Master Branch

The master branch represents the original implementation of the WARPED 2 kernel. It supports both asynchronous and synchronous Global Virtual Time (GVT) implementations, uses a multiset scheduled queue, and maintains separate processed and unprocessed input queues. This serves as our baseline for comparison with other optimizations.

1.5.2 Fossil Branch

The fossil branch introduces a separate fossil collection thread, eliminating the need for locks during fossil collection. Key features include:

- Tweaks to algorithms to avoid locks during fossil collection
- Potential performance gains on systems without hyperthreading
- Similar runtime to master branch on systems with hyperthreading

Our experiments revealed that simulations with 3+1 threads (3 worker threads and 1 fossil collection thread) resulted in fewer rollbacks. Increasing to 6 threads provided some performance gain, but the improvement was limited due to the increased number of rollbacks. This behavior is likely related to cache swapping delays on our test system with 4 main cores.

1.5.3 SIMD Branch

The SIMD branch aimed to leverage SIMD instructions provided by the AVX platform. However, significant challenges were encountered:

- Requires contiguous memory allocation for members of the Event class
- Necessitates overhauling MPI serialization for multi-node simulations
- Implementation incomplete due to extensive changes required in the WARPED 2 algorithm

While not fully implemented, this branch highlights the potential for SIMD optimizations in future work.

1.5.4 Hashing Branch

The hashing branch implements a technique to reduce the cost of event comparisons:

- Uses hashing for LP names
- One-time hashing cost followed by quick comparisons
- Showed performance gains in experiments
- Potential for further improvement using advanced techniques like Google's MurmurHash

1.5.5 Relaxed Compare Branch

This branch explores a relaxed compare function based on the hypothesis that strict ordering in the scheduled queue is not always necessary:

- Relies on strict ordering at the LP level instead of the global level
- Can fail in certain scenarios
- Showed significant performance gains for some models, particularly those with high LP event processing times

1.5.6 Unified Queue Branch

The unified queue branch implements a combined approach:

- Joins processed and unprocessed events into a single ordered circular queue

- Uses synchronous GVT
- Implements a multiset schedule queue with relaxed ordering
- Utilizes a circular random access iterator for efficient sorting
- Eliminates unnecessary barriers in the simulation

This branch showed promising results by streamlining queue management and reducing synchronization overhead.

1.6 GVT Implementation Comparison

Across our branch implementations, we observed that:

- Synchronous GVT implementation showed better performance on single-node simulations
- Asynchronous GVT performed better on multi-node systems

These findings align with the general industry understanding of GVT types and their optimal use cases.

1.7 Future Work

Future research could explore several promising directions:

- Leveraging new hardware features, such as eco-cores in modern CPUs, for tasks like fossil collection
- Further optimization of the SIMD implementation to overcome current challenges
- Exploration of advanced hashing techniques to improve event comparison efficiency
- Investigating hybrid approaches that combine the strengths of different branch implementations

In conclusion, our research has highlighted the complex interplay between model characteristics, configuration choices, and branch implementations in optimizing the performance of the WARPED simulation kernel. Each branch implementation offers unique advantages and trade-offs, demonstrating the potential for significant improvements in simulation efficiency through targeted optimizations. By carefully considering these factors and applying the most suitable optimizations for specific simulation scenarios, we can continue to enhance the capabilities and performance of the WARPED simulation kernel.

2 Label Definitions

Model	**Grid Size**	**Max Sim Time**	**Worker Threads**	**State Save Period**	**Iter-ations**	**GVT Period**
Epidemic 10k-ba	10k	20000	3, 6	32	5	1000
Epidemic 100k-ba	100k	6000	3, 6	32	5	1000
PCS 10k	10k	500	3, 6	32	5	1000
Traffic 10k	10k	10000	3, 6	32	5	1000

Table 1: Modified configuration summary for different models

3 fossil

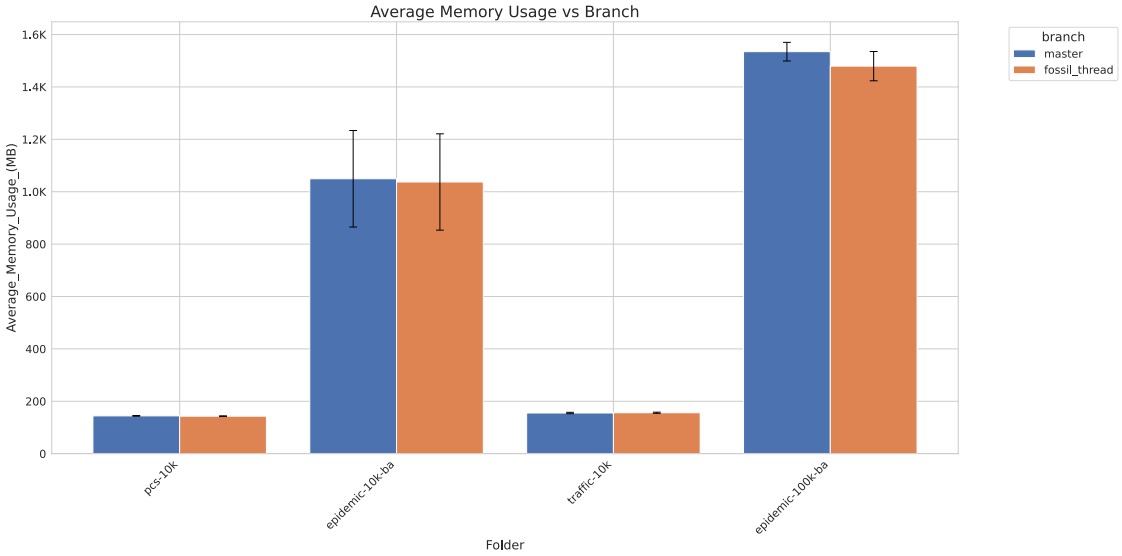

Figure 1: Average Memory Usage vs Branch

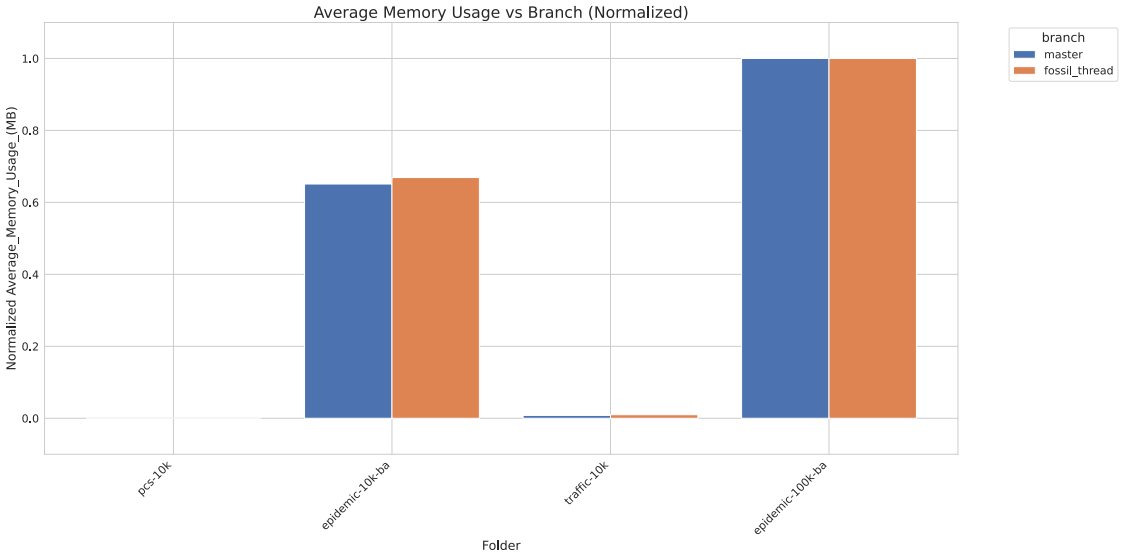

Figure 2: Average Memory Usage vs Branch_normalized

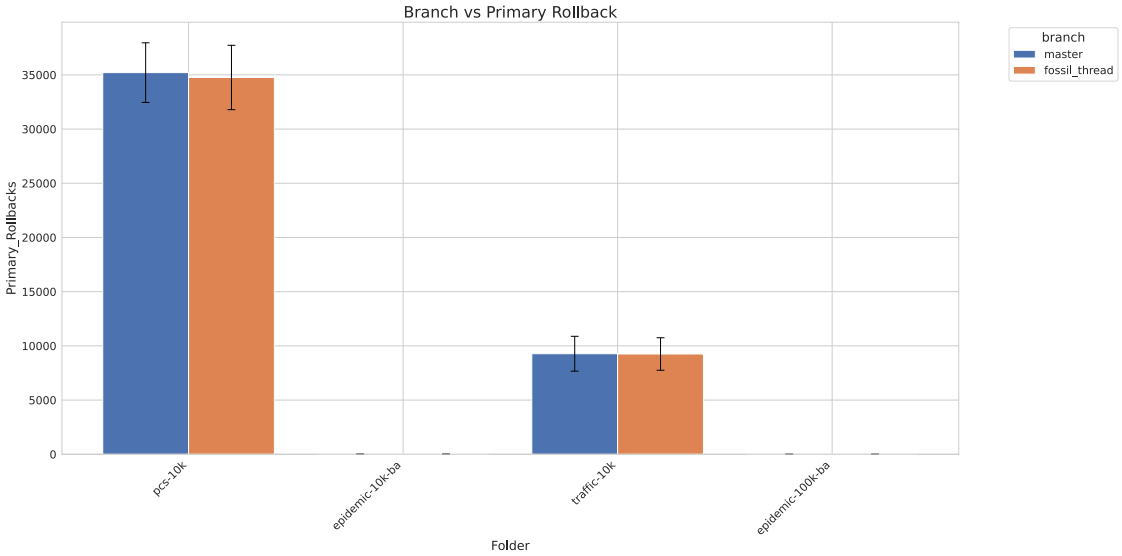

Figure 3: Branch vs Primary Rollback

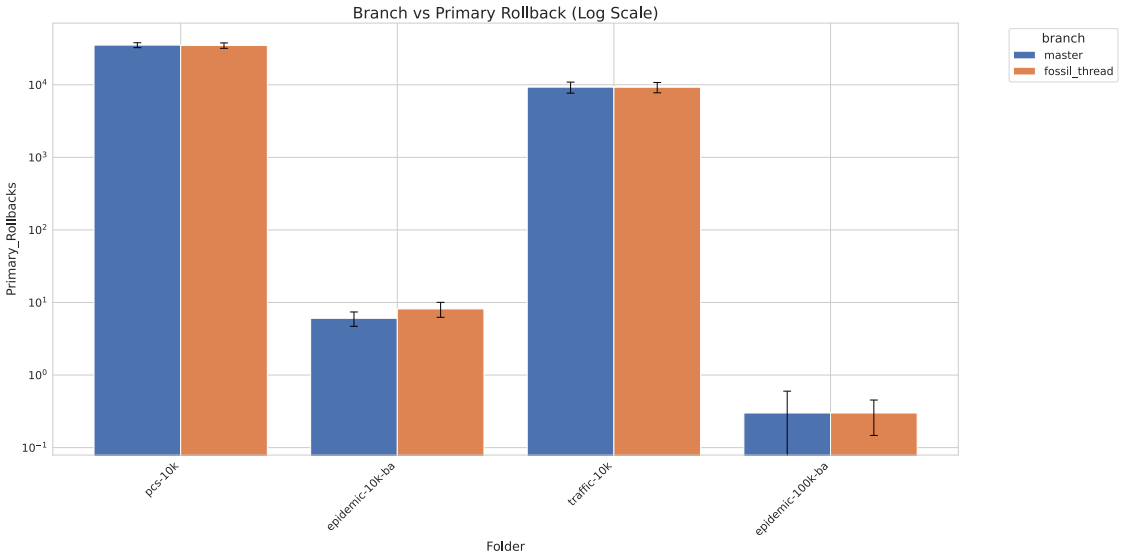

Figure 4: Branch vs Primary Rollback_log

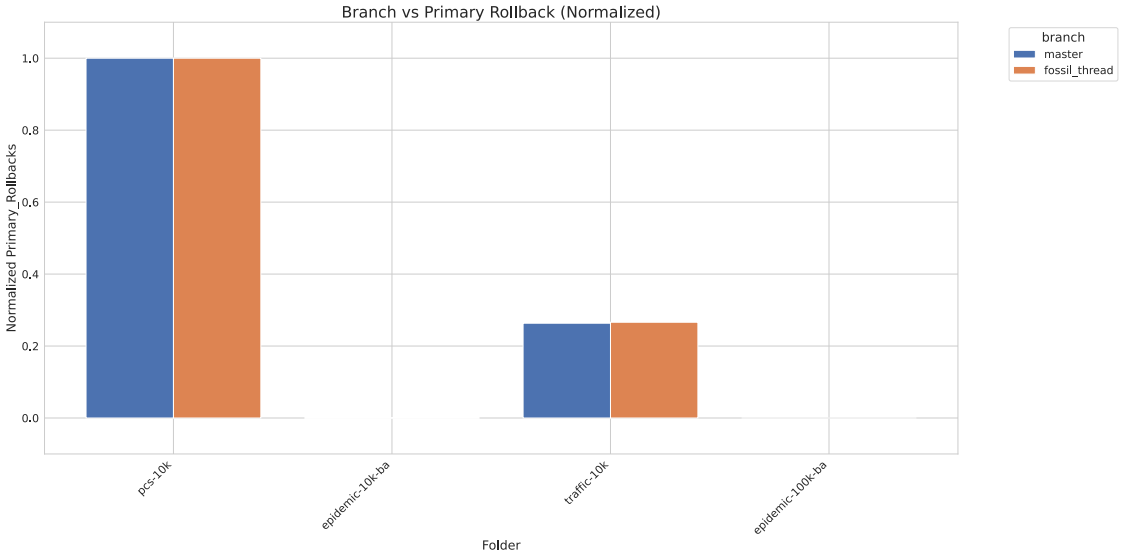

Figure 5: Branch vs Primary Rollback_normalized

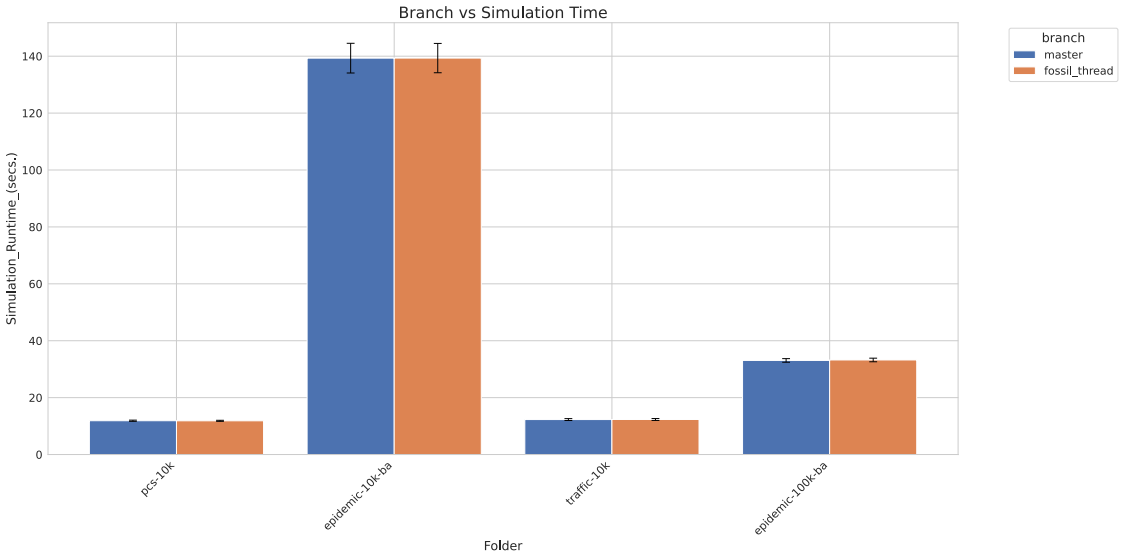

Figure 6: Branch vs Simulation Time

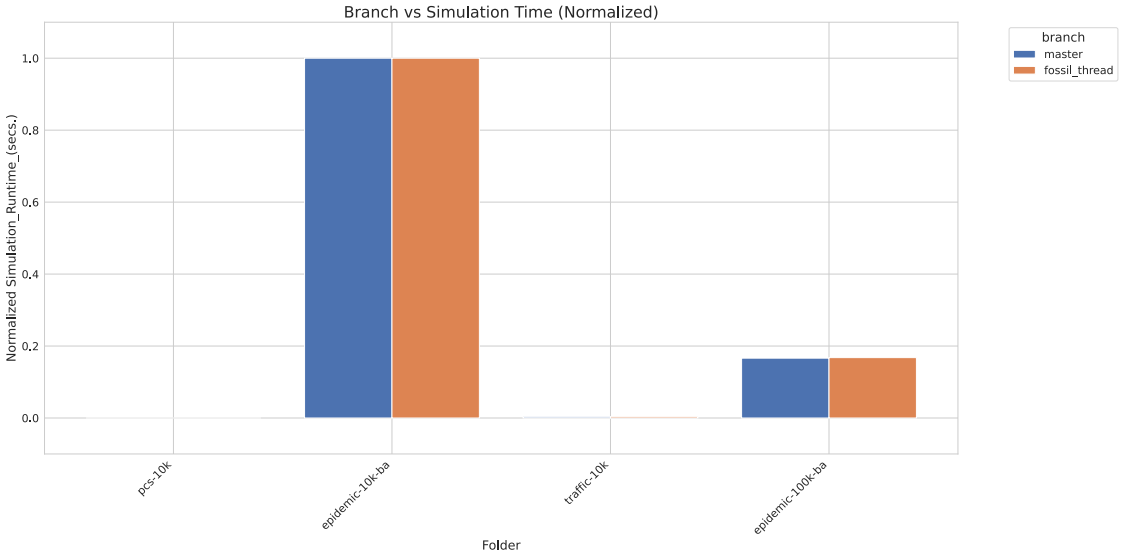

Figure 7: Branch vs Simulation Time_normalized

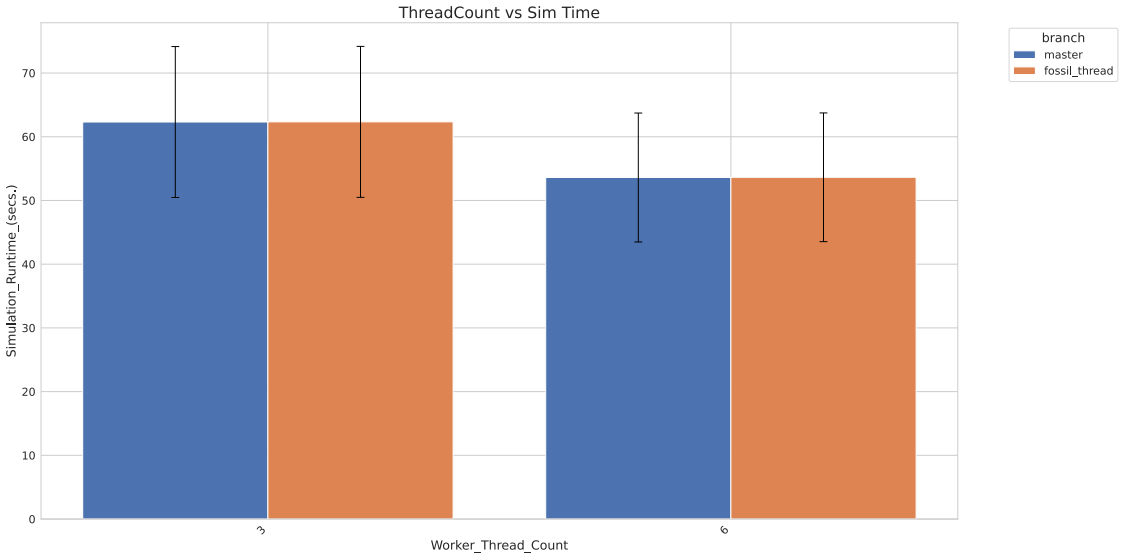

Figure 8: ThreadCount vs Sim Time

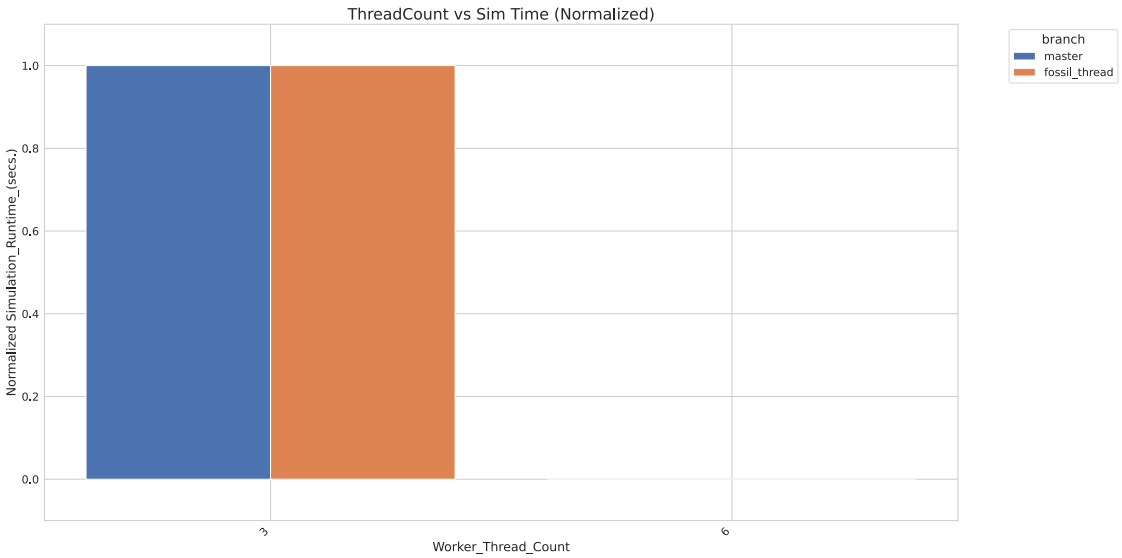

Figure 9: ThreadCount vs Sim Time_normalized

4 fossillocal

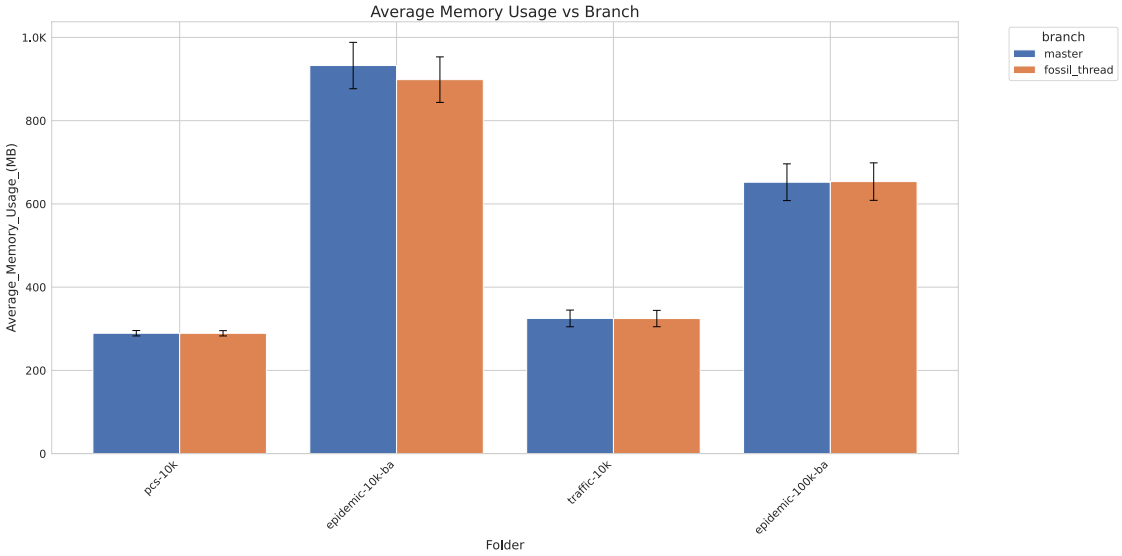

Figure 10: Average Memory Usage vs Branch

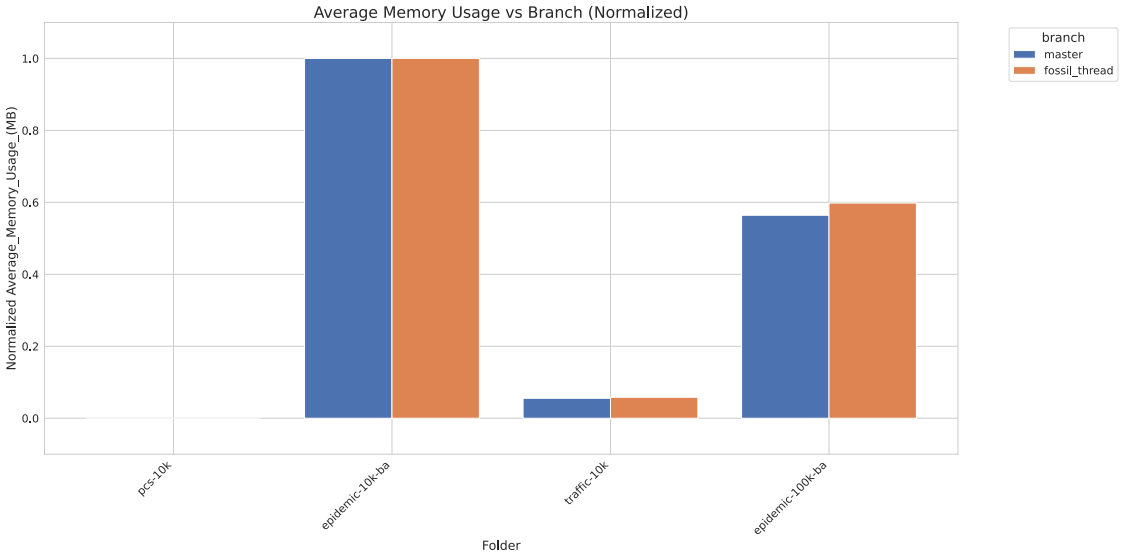

Figure 11: Average Memory Usage vs Branch_normalized

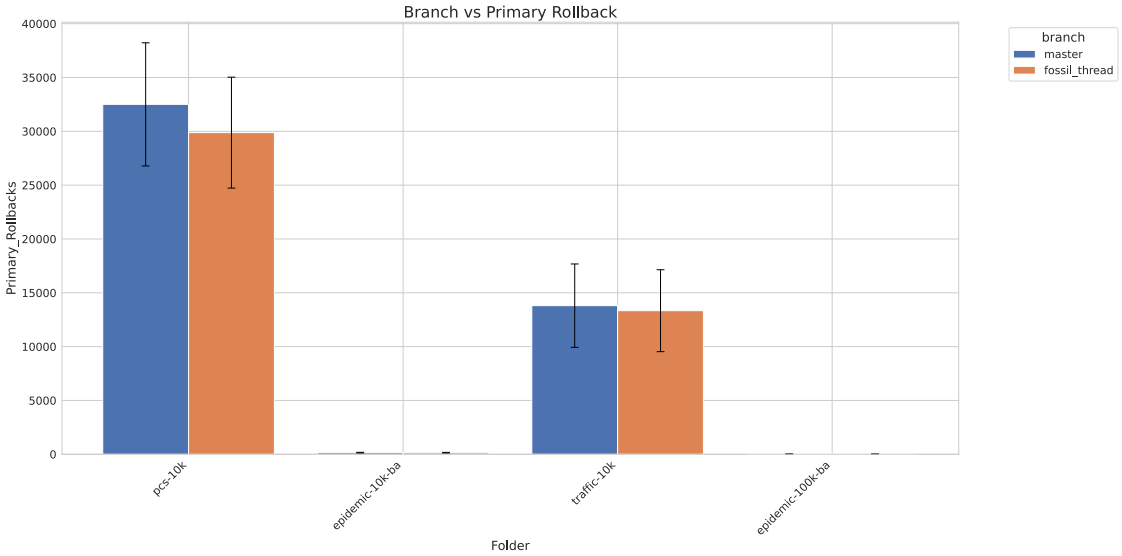

Figure 12: Branch vs Primary Rollback

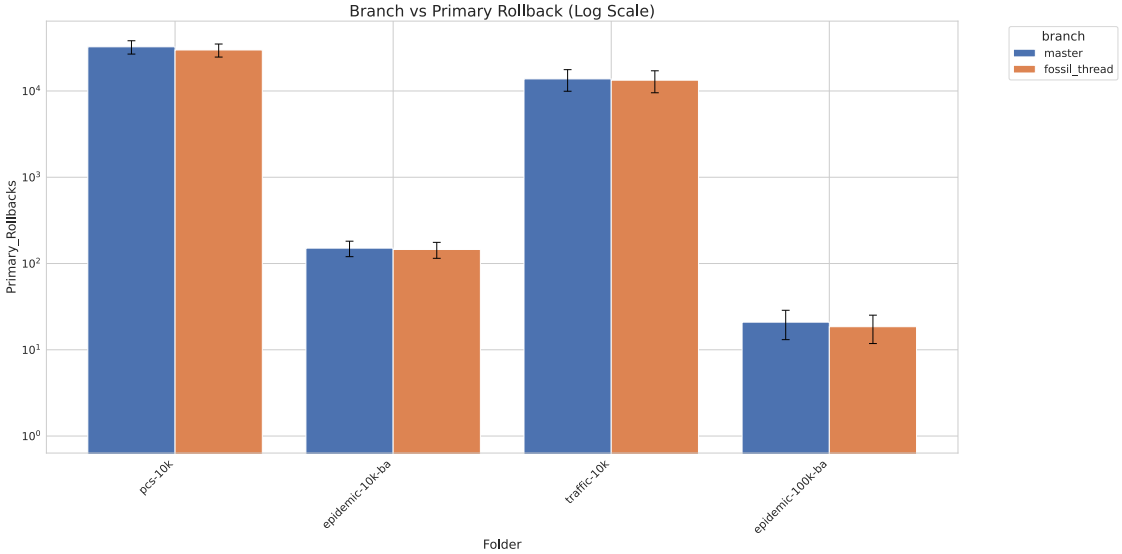

Figure 13: Branch vs Primary Rollback_log

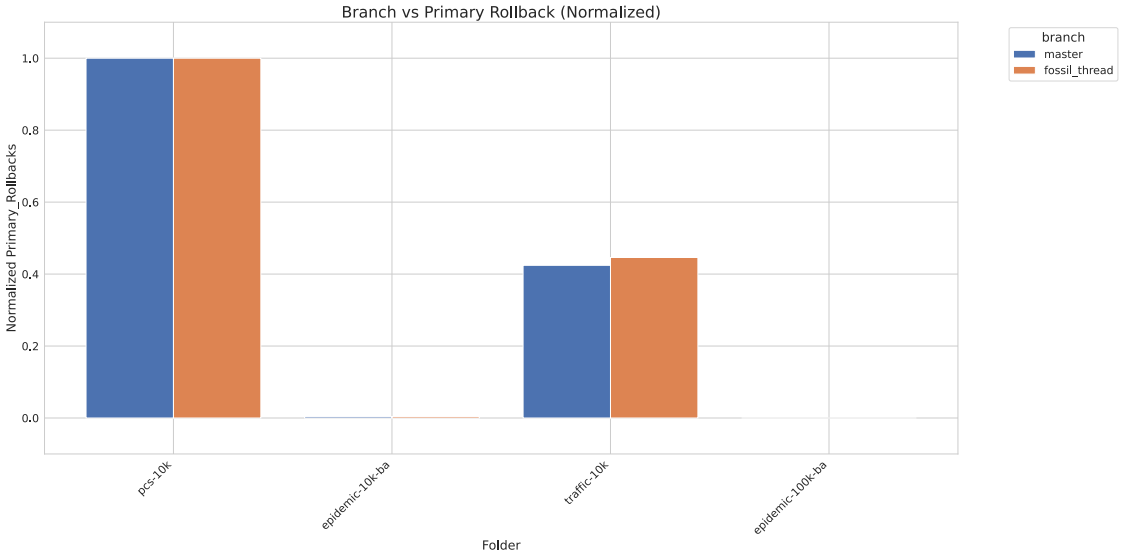

Figure 14: Branch vs Primary Rollback_normalized

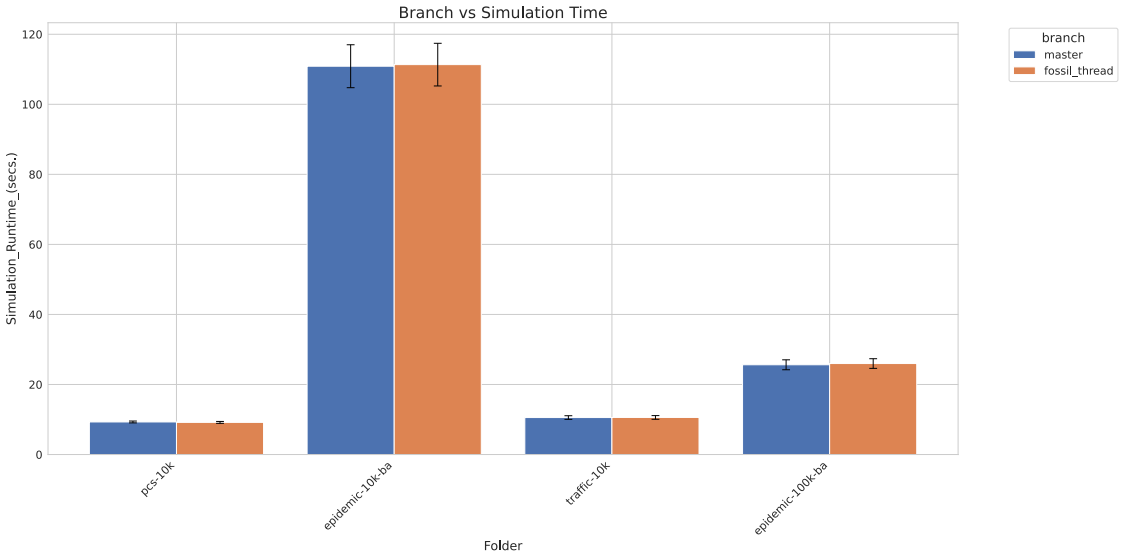

Figure 15: Branch vs Simulation Time

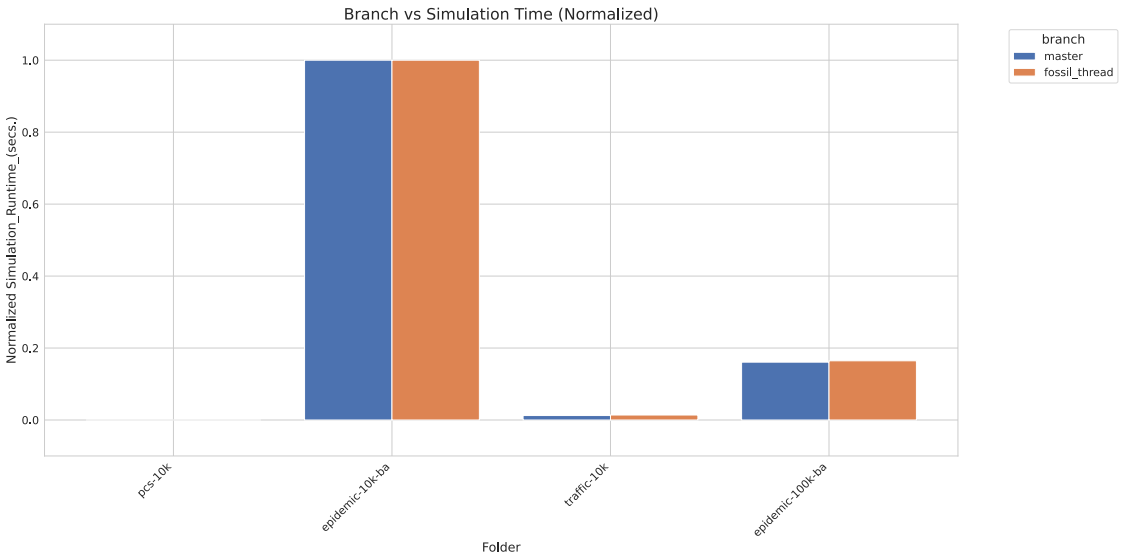

Figure 16: Branch vs Simulation Time_normalized

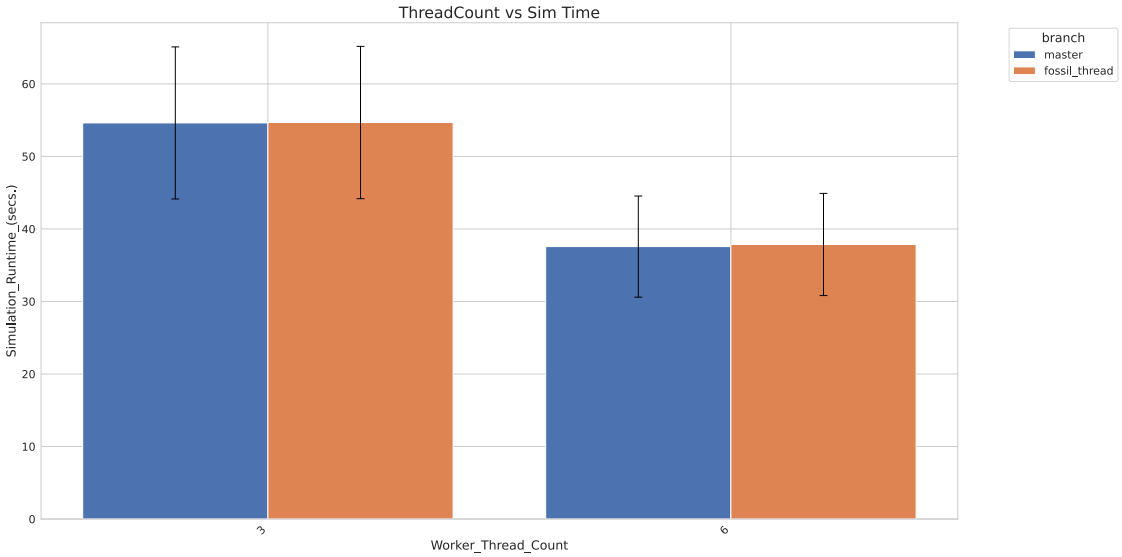

Figure 17: ThreadCount vs Sim Time

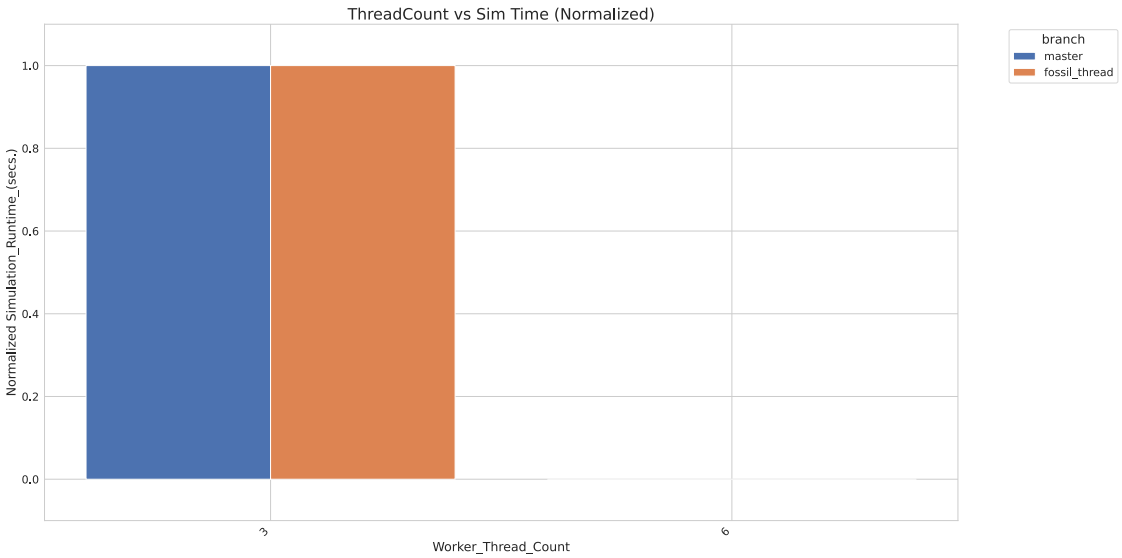

Figure 18: ThreadCount vs Sim Time_normalized

5 gvt_comparison

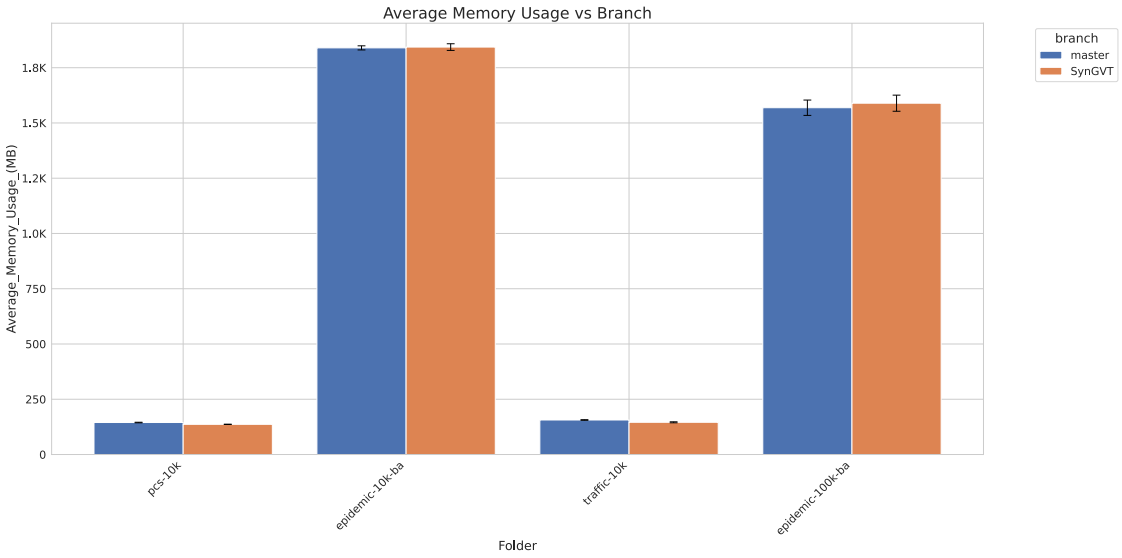

Figure 19: Average Memory Usage vs Branch

Figure 20: Average Memory Usage vs Branch_normalized

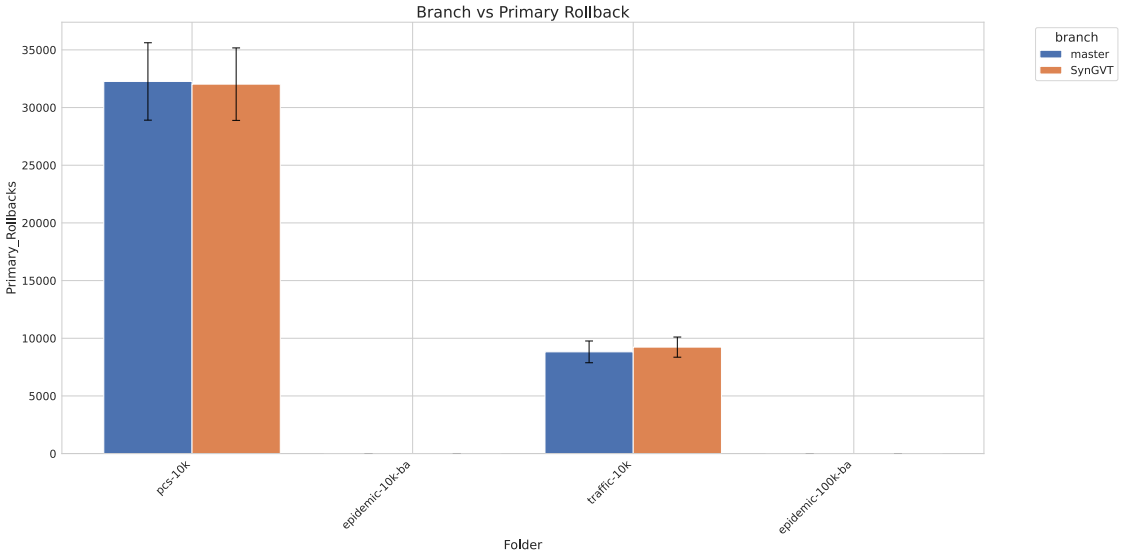

Figure 21: Branch vs Primary Rollback

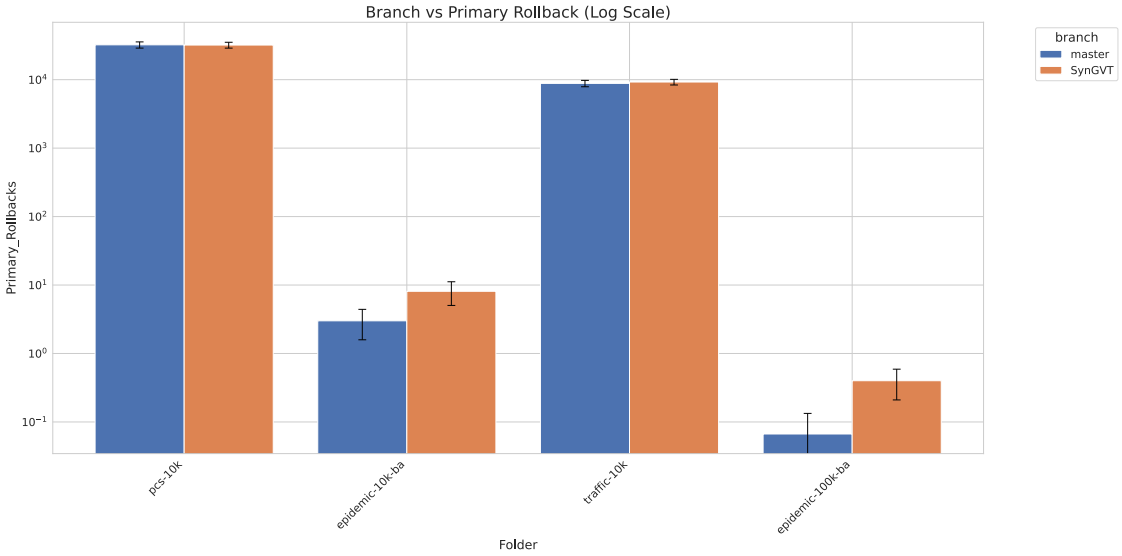

Figure 22: Branch vs Primary Rollback_log

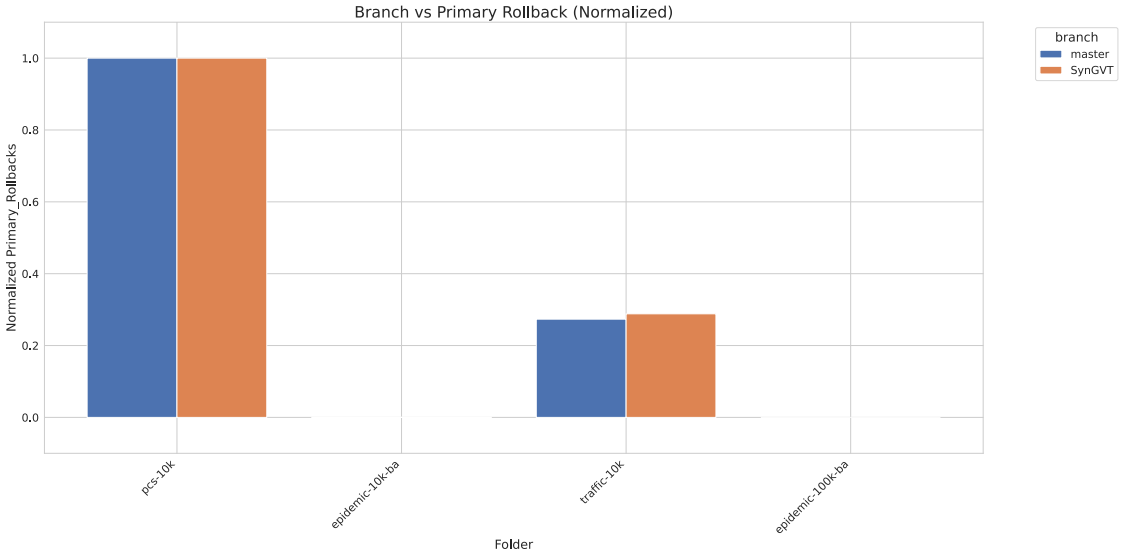

Figure 23: Branch vs Primary Rollback_normalized

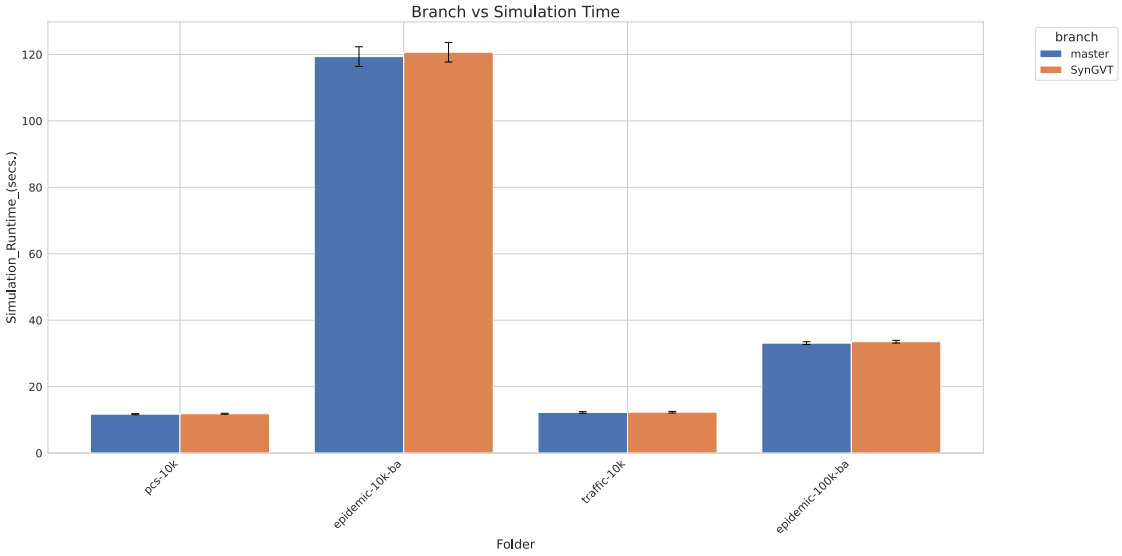

Figure 24: Branch vs Simulation Time

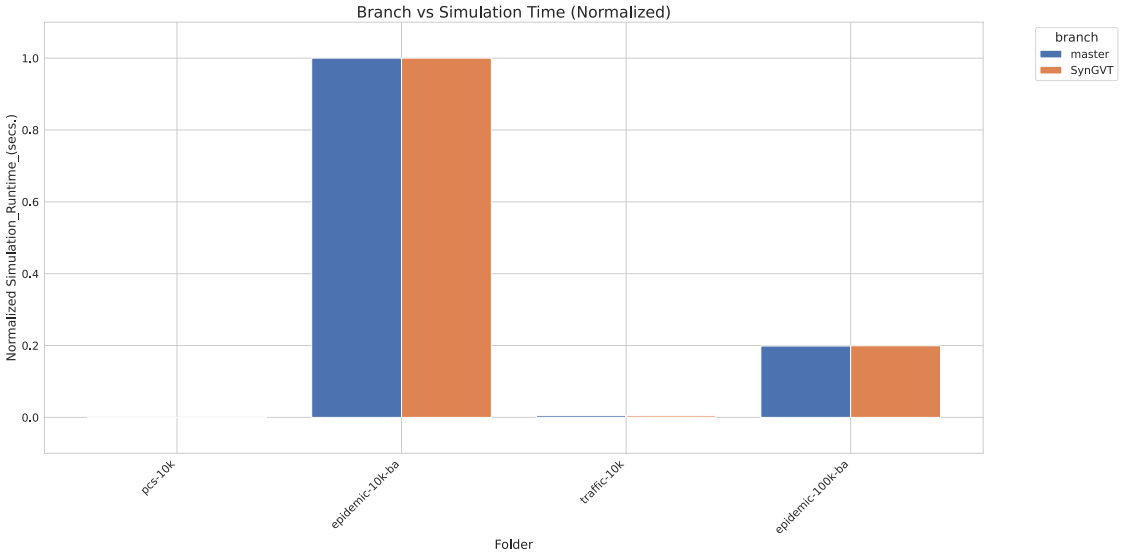

Figure 25: Branch vs Simulation Time_normalized

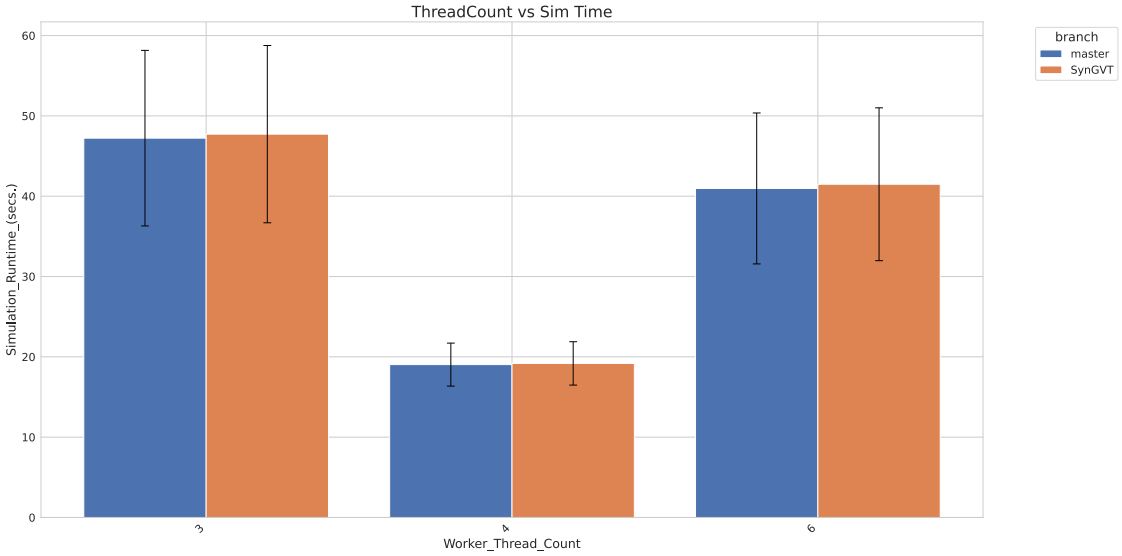

Figure 26: ThreadCount vs Sim Time

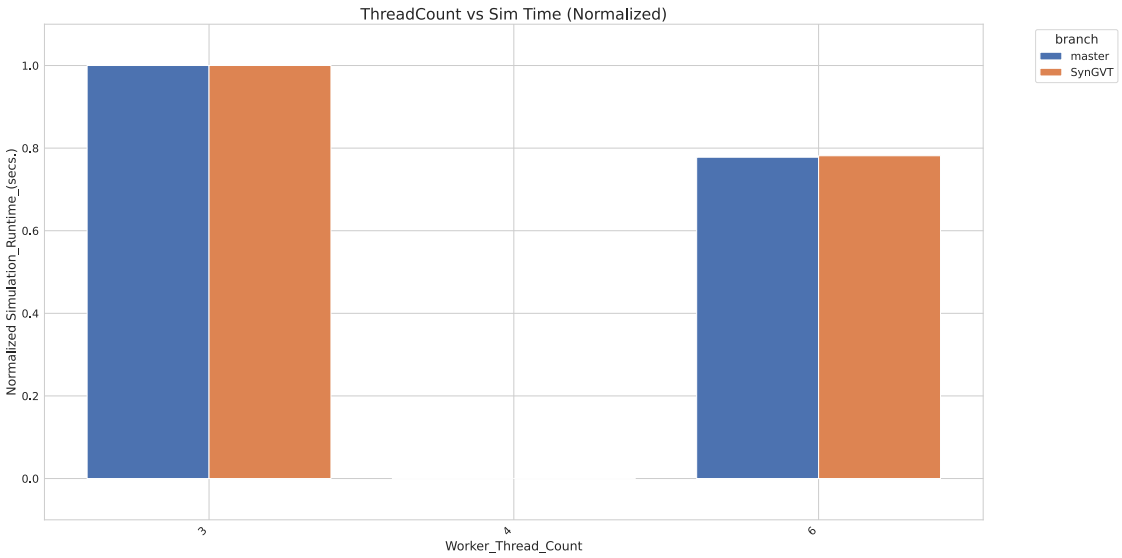

Figure 27: ThreadCount vs Sim Time_normalized

6 gvt_comparison_local

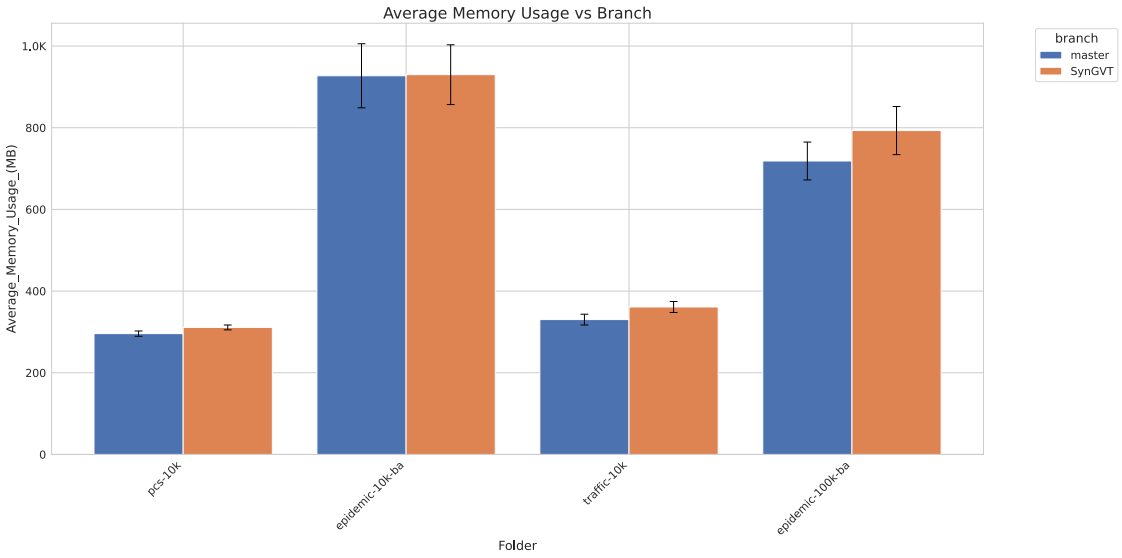

Figure 28: Average Memory Usage vs Branch

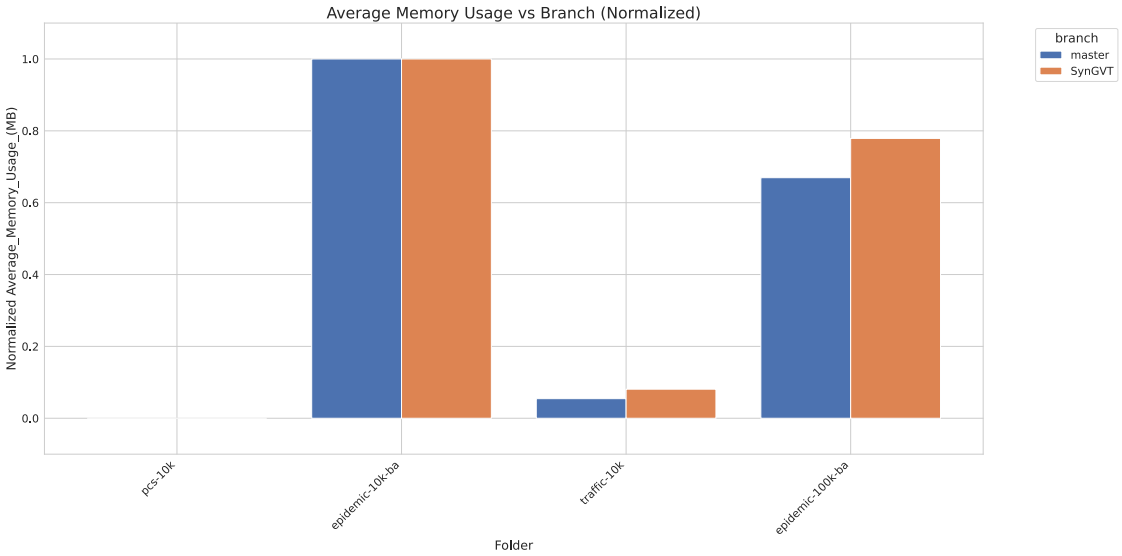

Figure 29: Average Memory Usage vs Branch_normalized

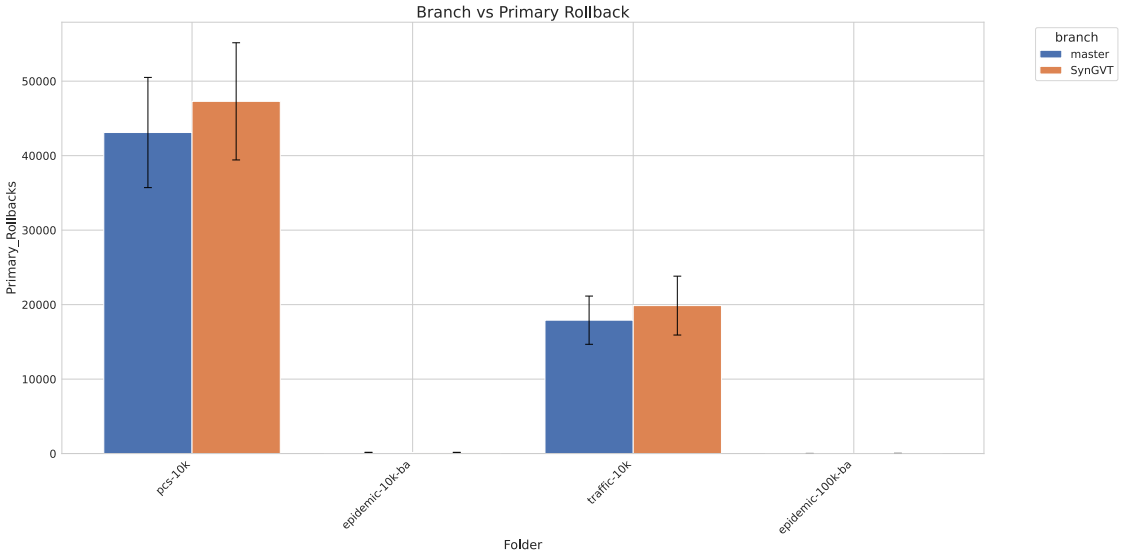

Figure 30: Branch vs Primary Rollback

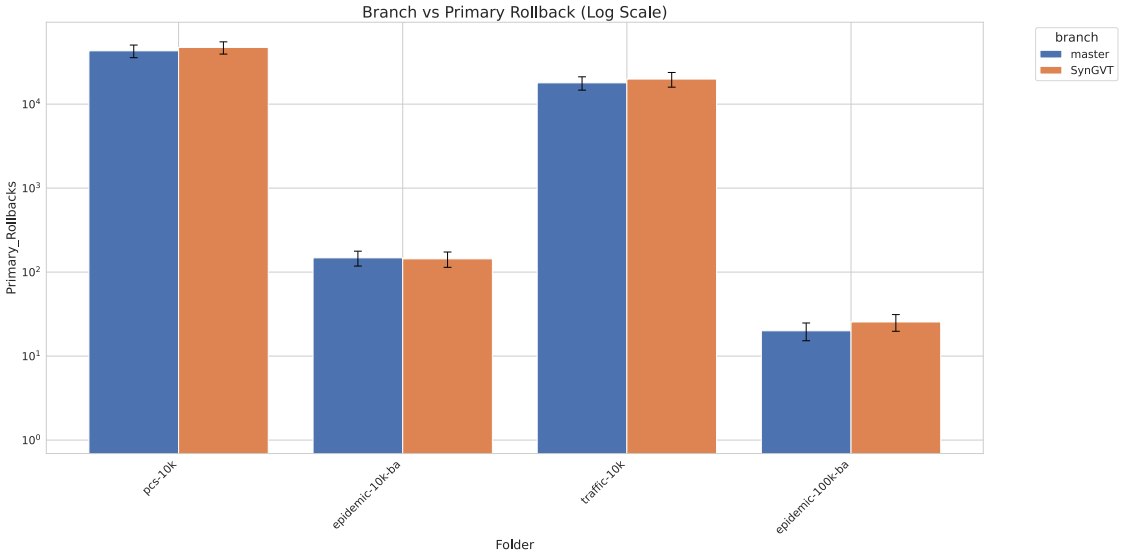

Figure 31: Branch vs Primary Rollback_log

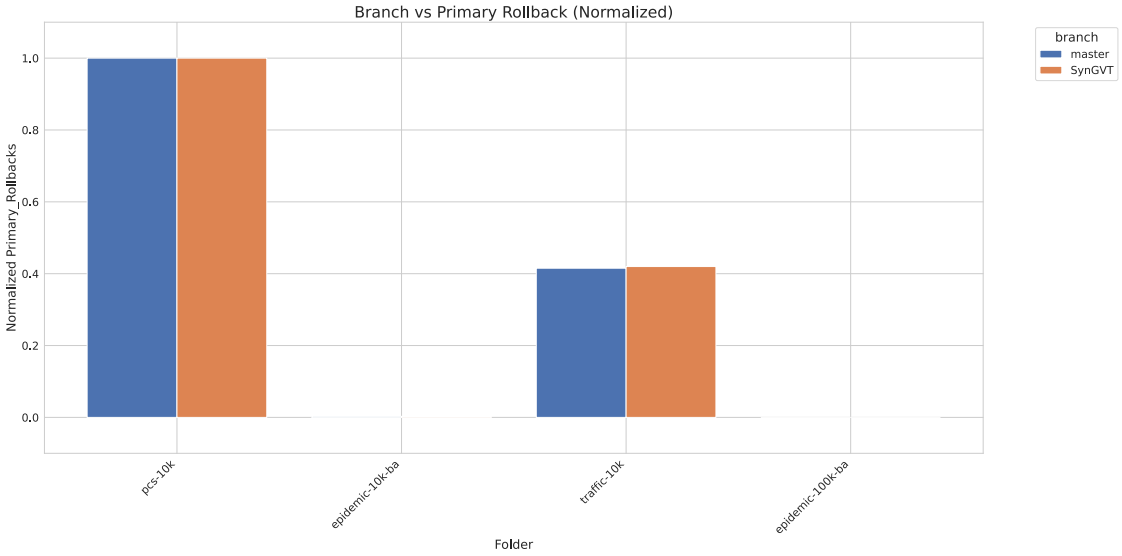

Figure 32: Branch vs Primary Rollback_normalized

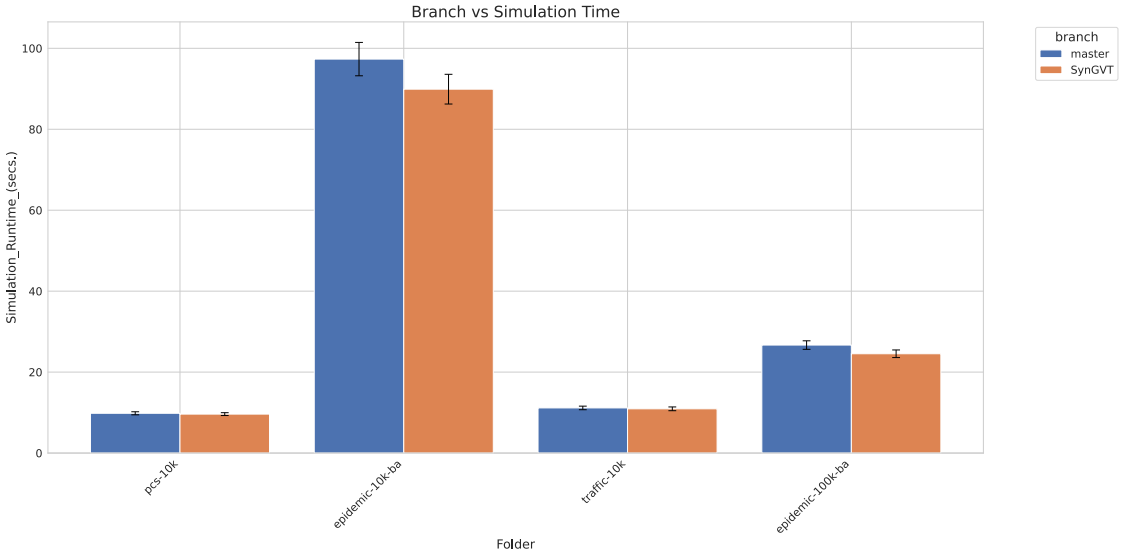

Figure 33: Branch vs Simulation Time

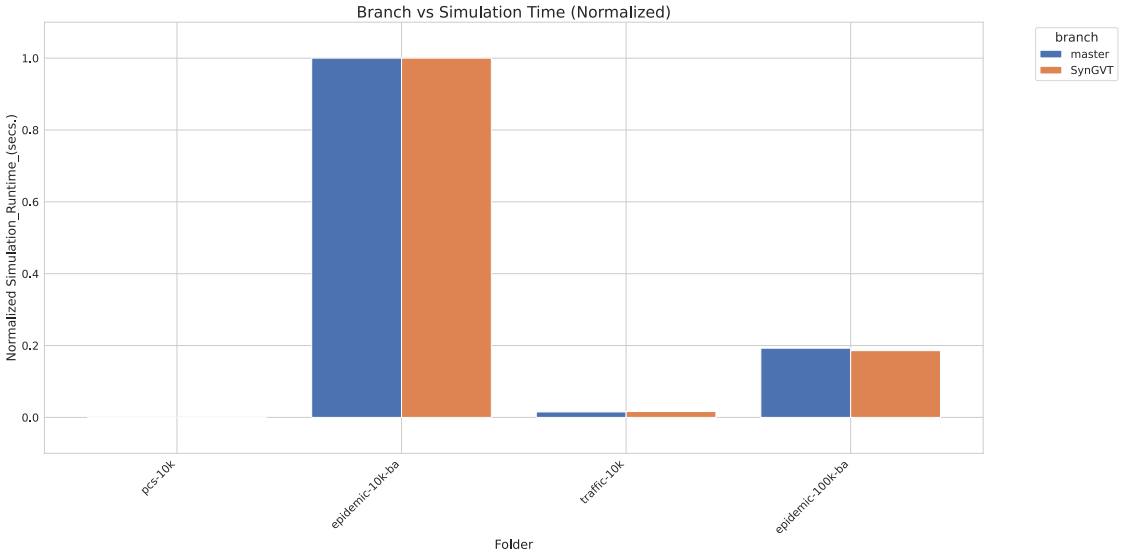

Figure 34: Branch vs Simulation Time_normalized

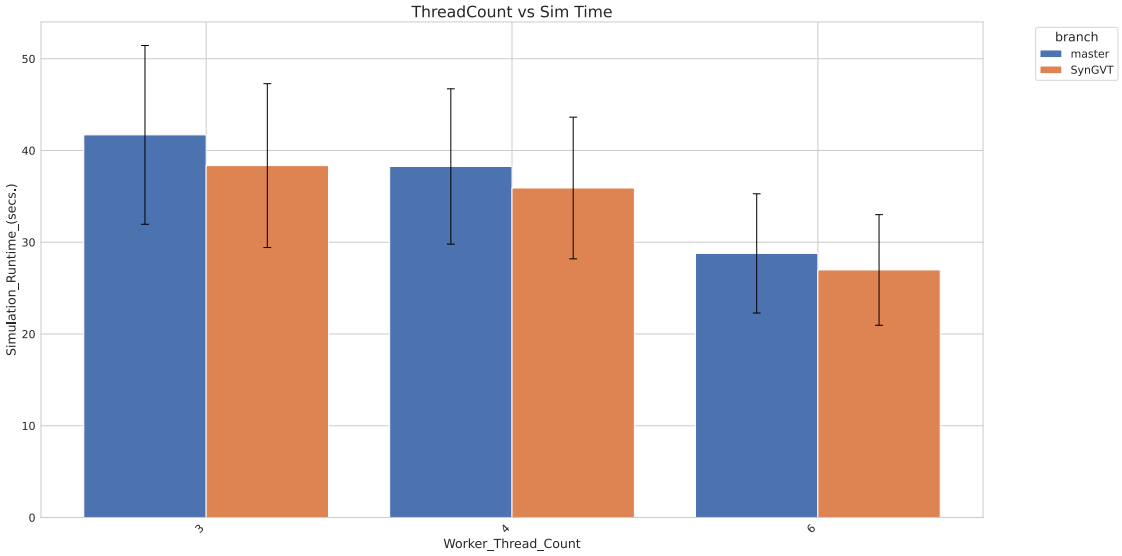

Figure 35: ThreadCount vs Sim Time

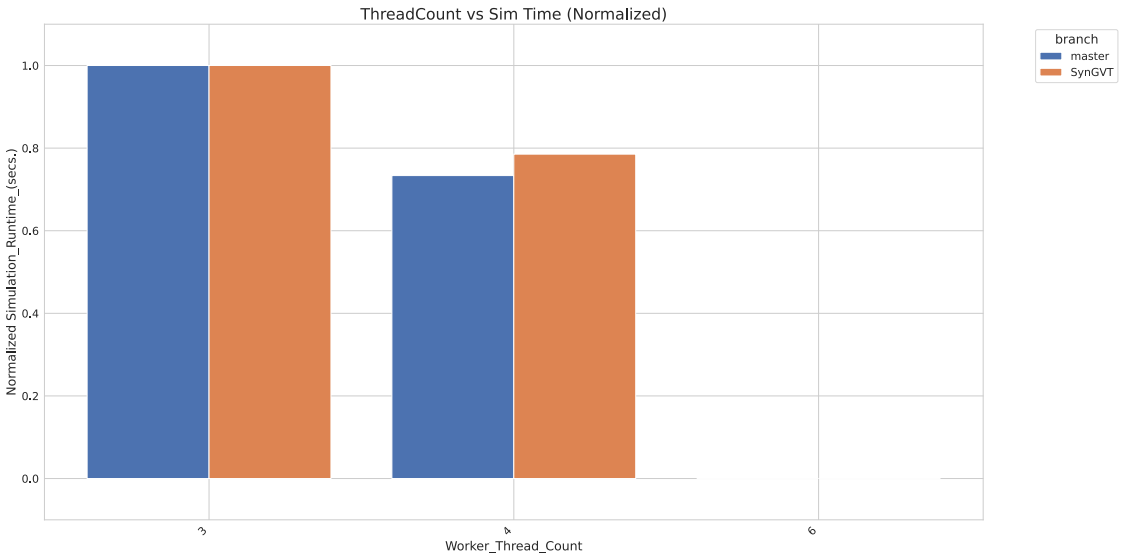

Figure 36: ThreadCount vs Sim Time_normalized

7 hashing

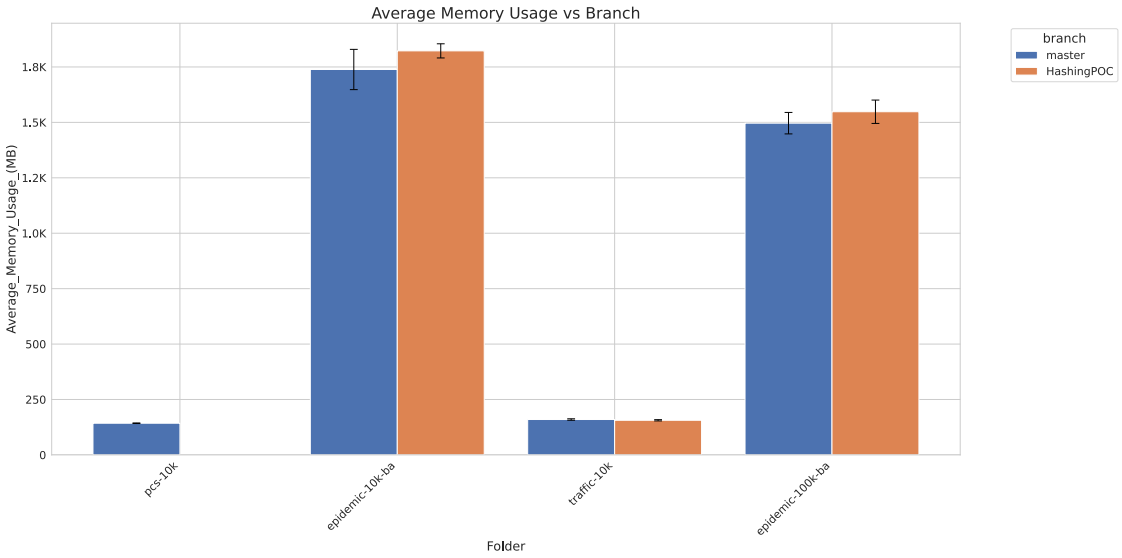

Figure 37: Average Memory Usage vs Branch

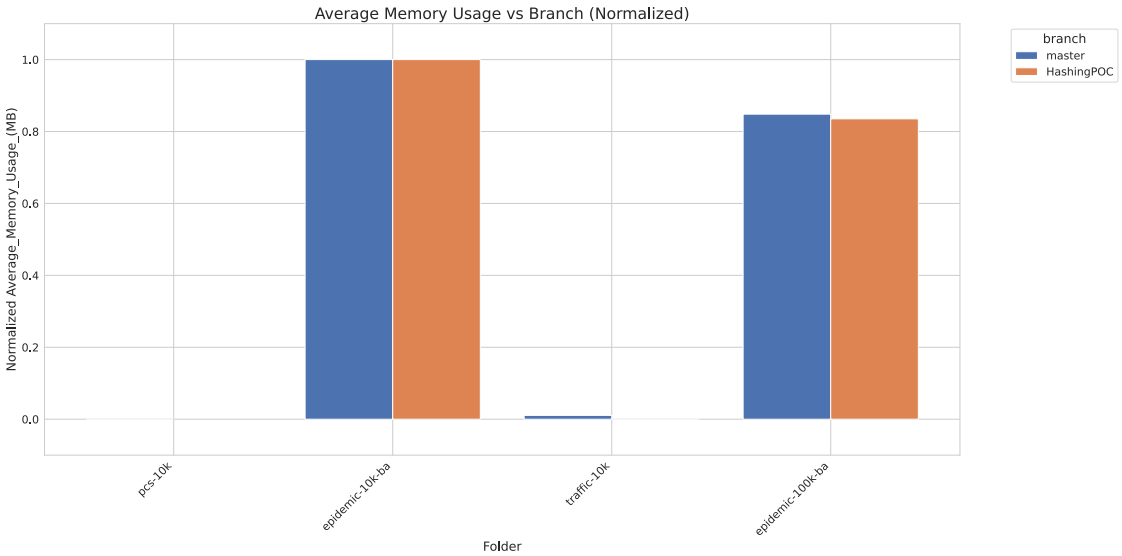

Figure 38: Average Memory Usage vs Branch_normalized

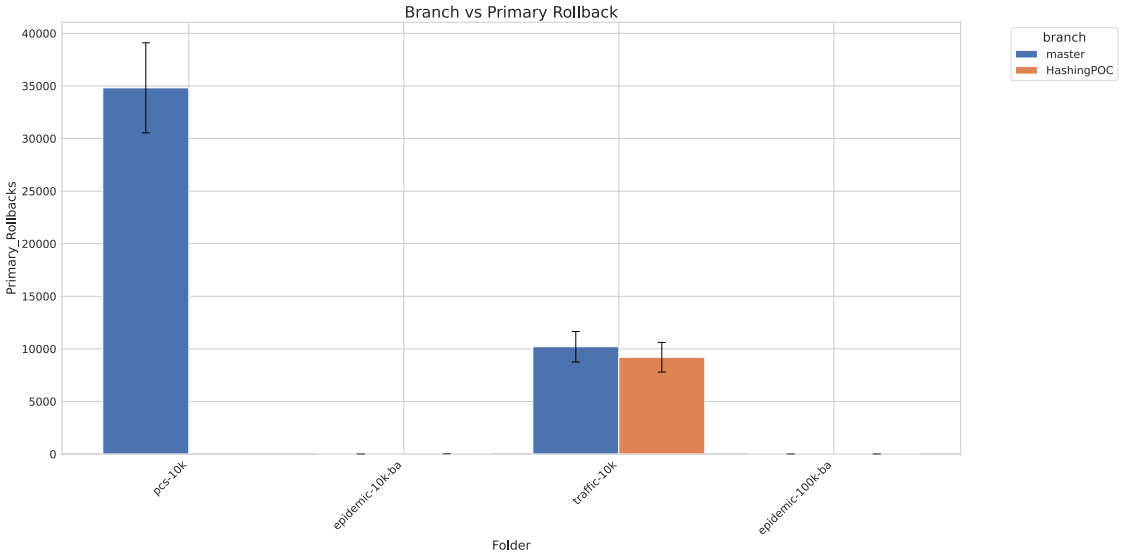

Figure 39: Branch vs Primary Rollback

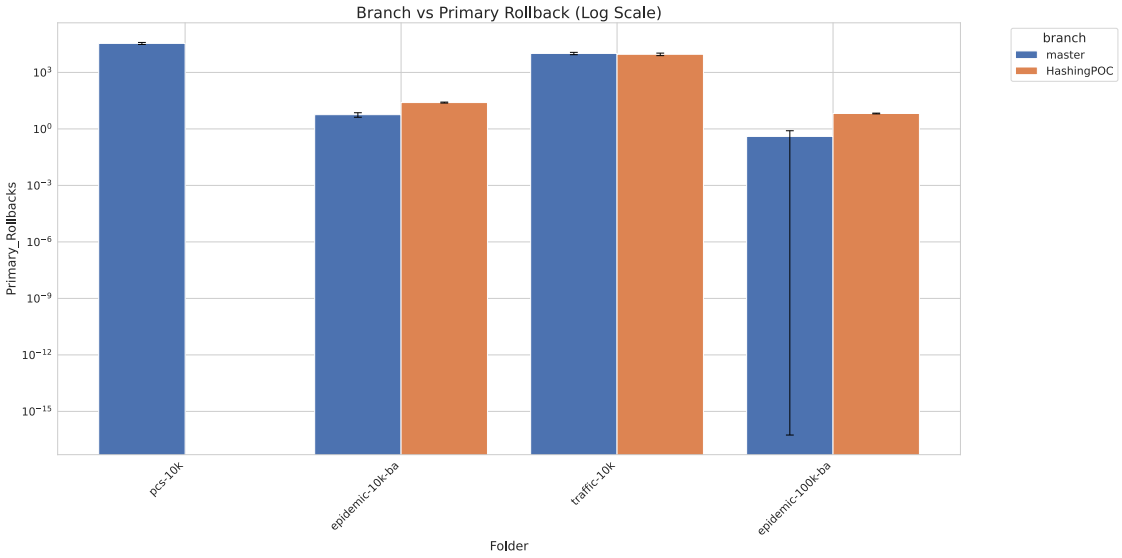

Figure 40: Branch vs Primary Rollback_log

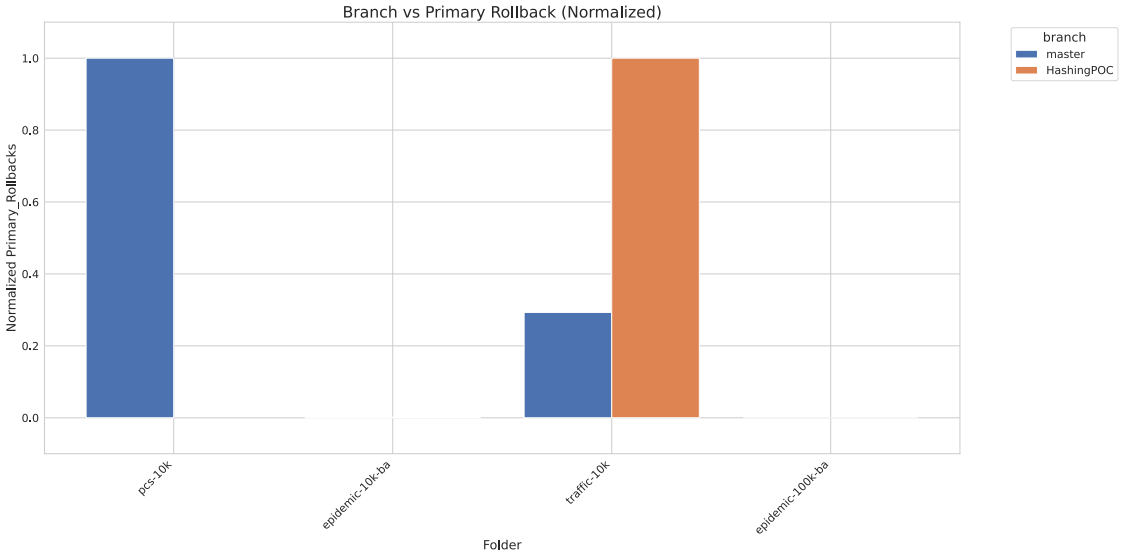

Figure 41: Branch vs Primary Rollback_normalized

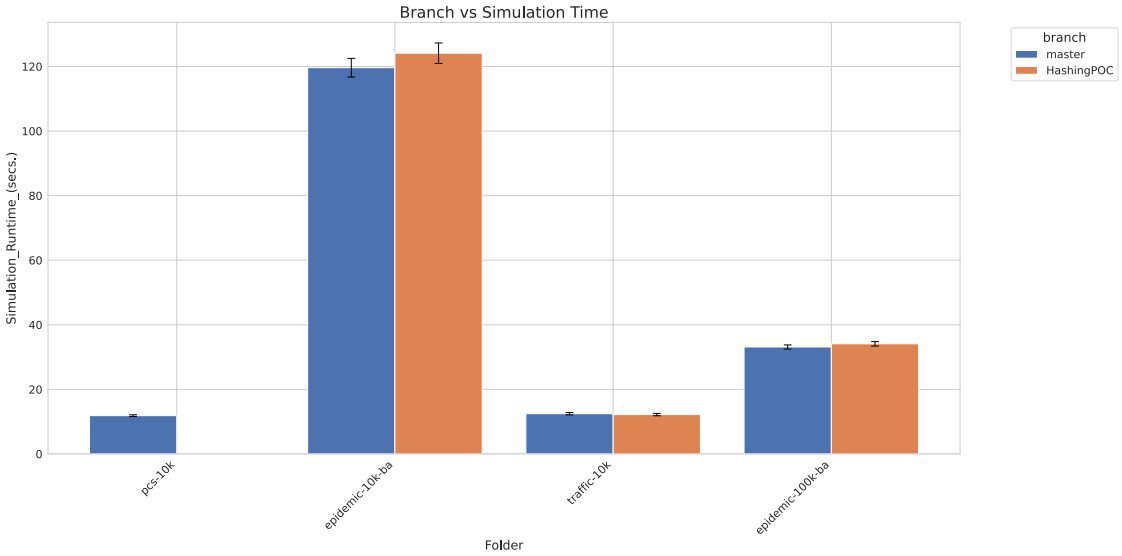

Figure 42: Branch vs Simulation Time

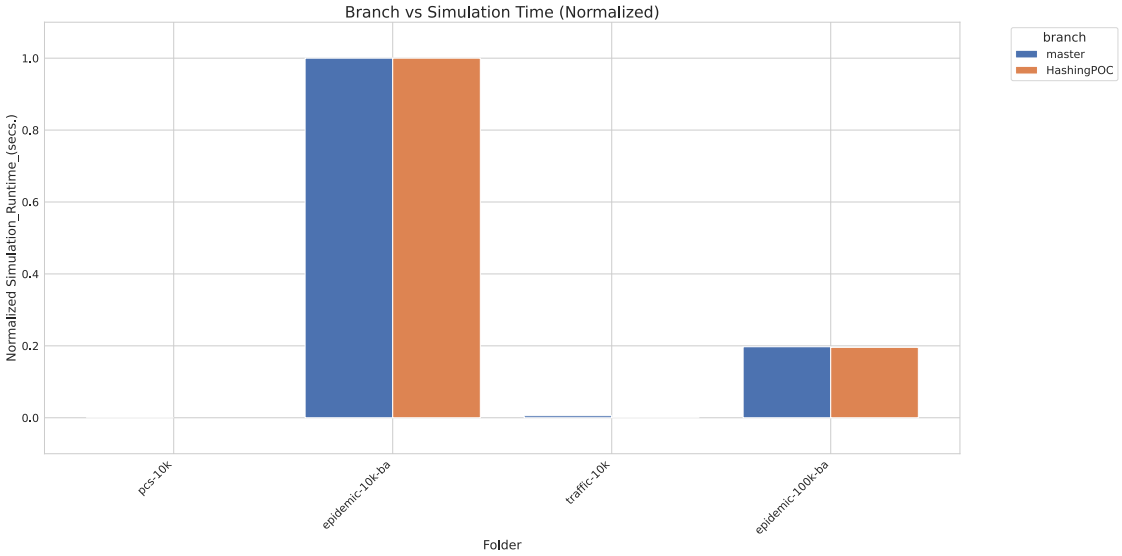

Figure 43: Branch vs Simulation Time_normalized

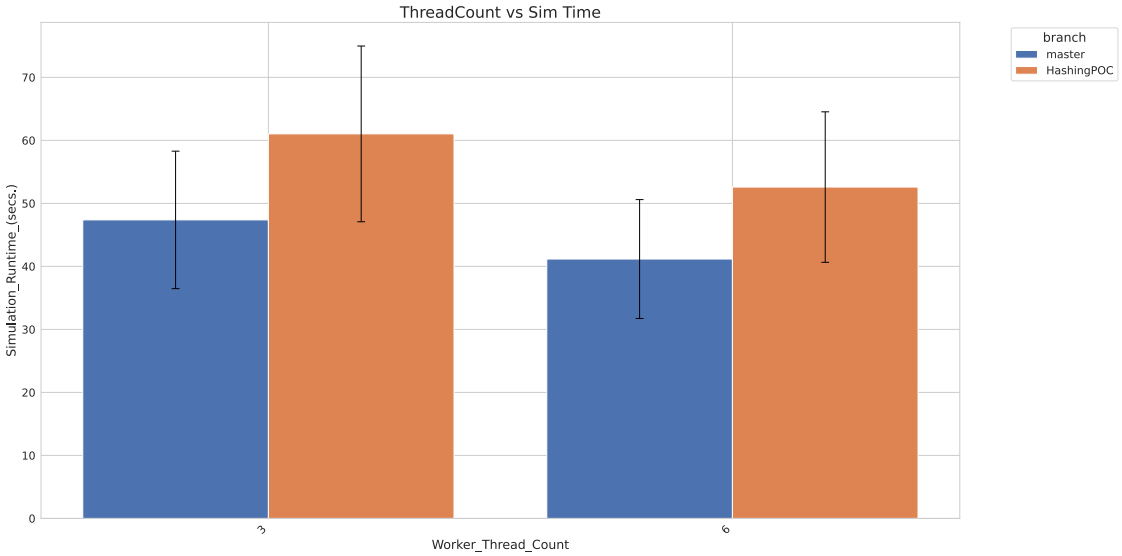

Figure 44: ThreadCount vs Sim Time

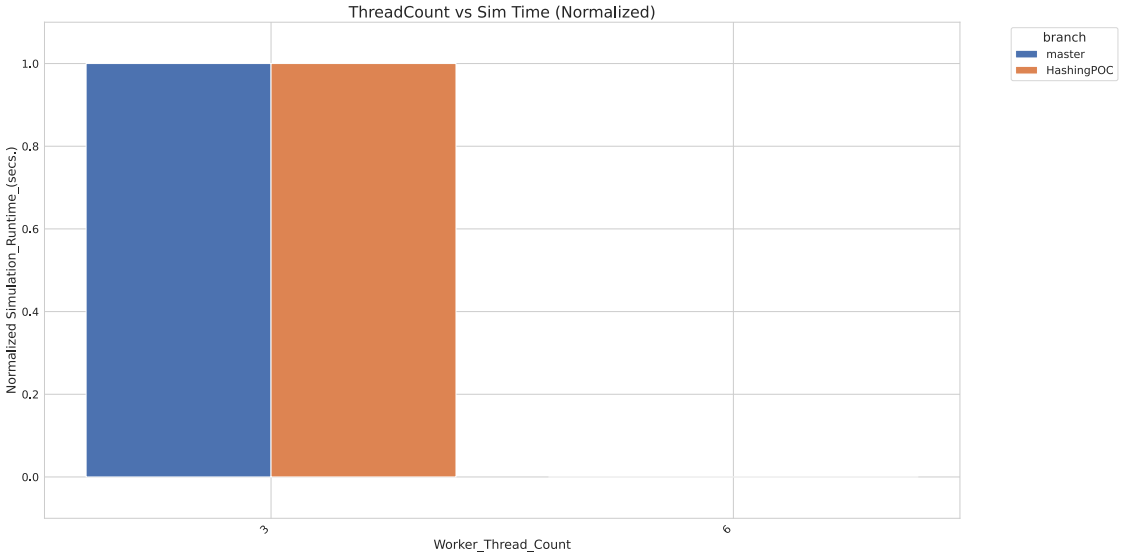

Figure 45: ThreadCount vs Sim Time_normalized

8 hashing_local

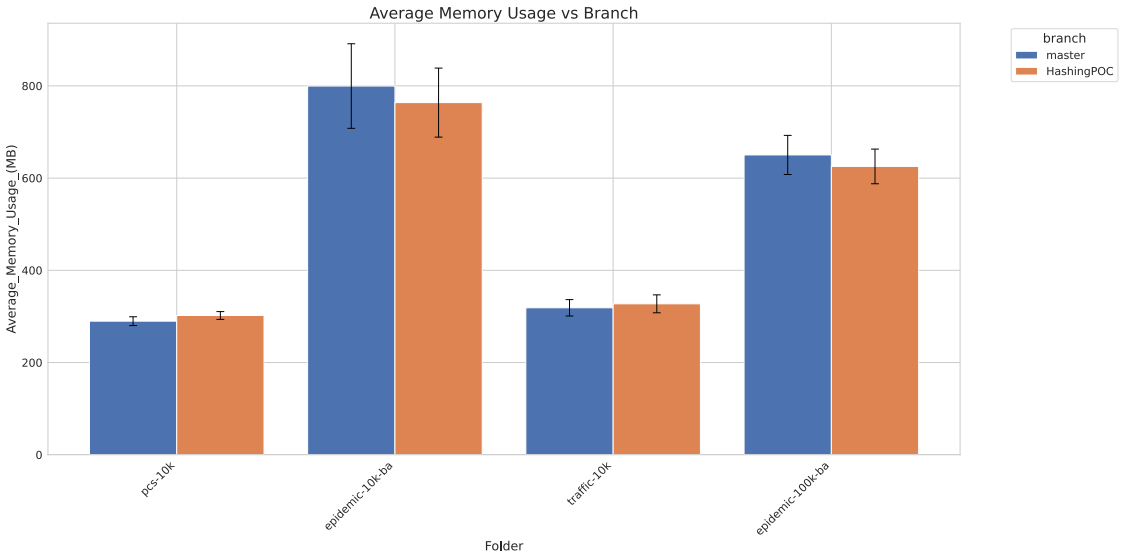

Figure 46: Average Memory Usage vs Branch

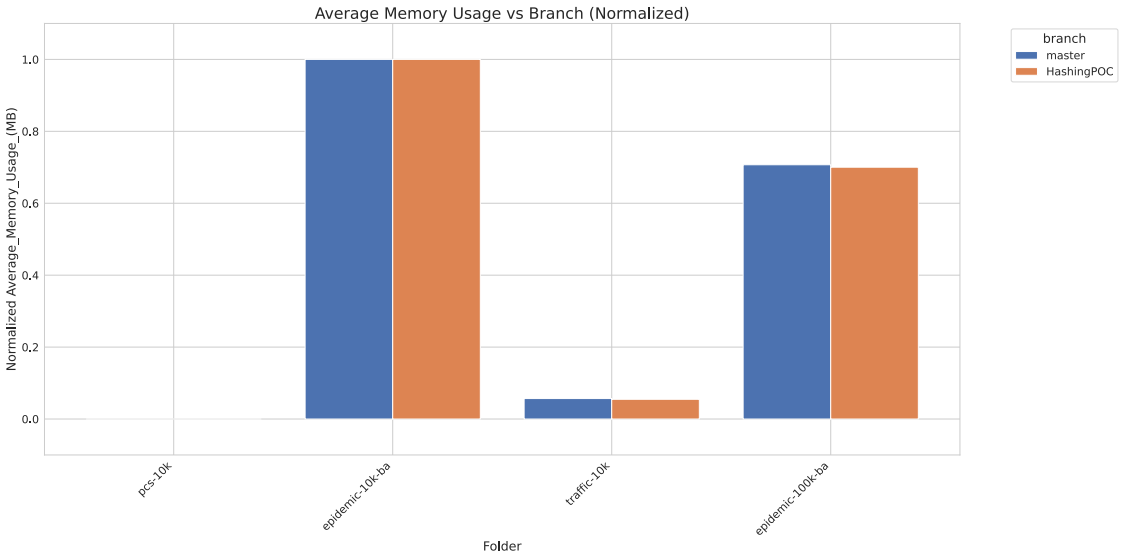

Figure 47: Average Memory Usage vs Branch_normalized

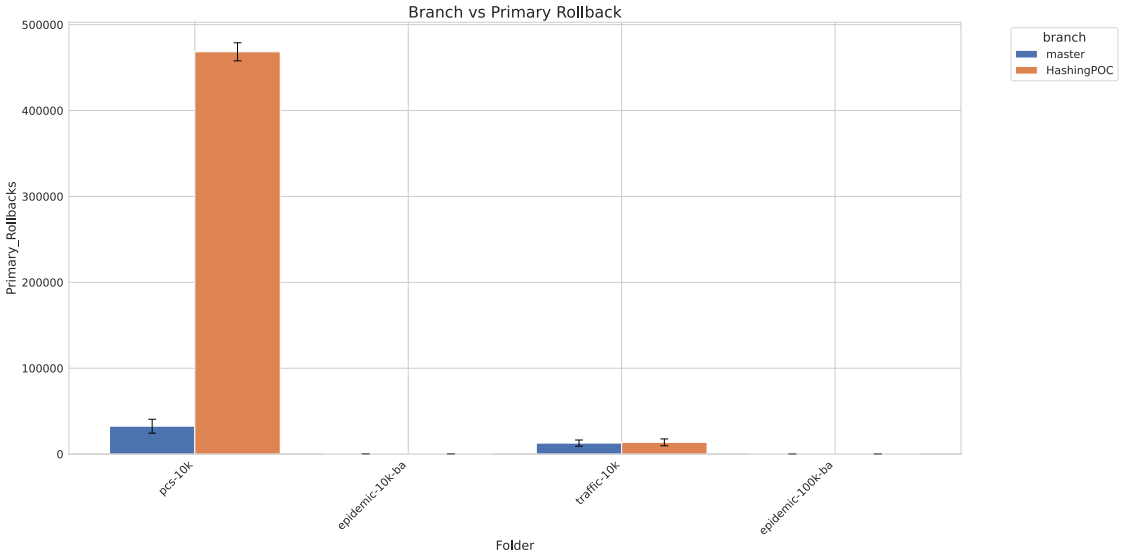

Figure 48: Branch vs Primary Rollback

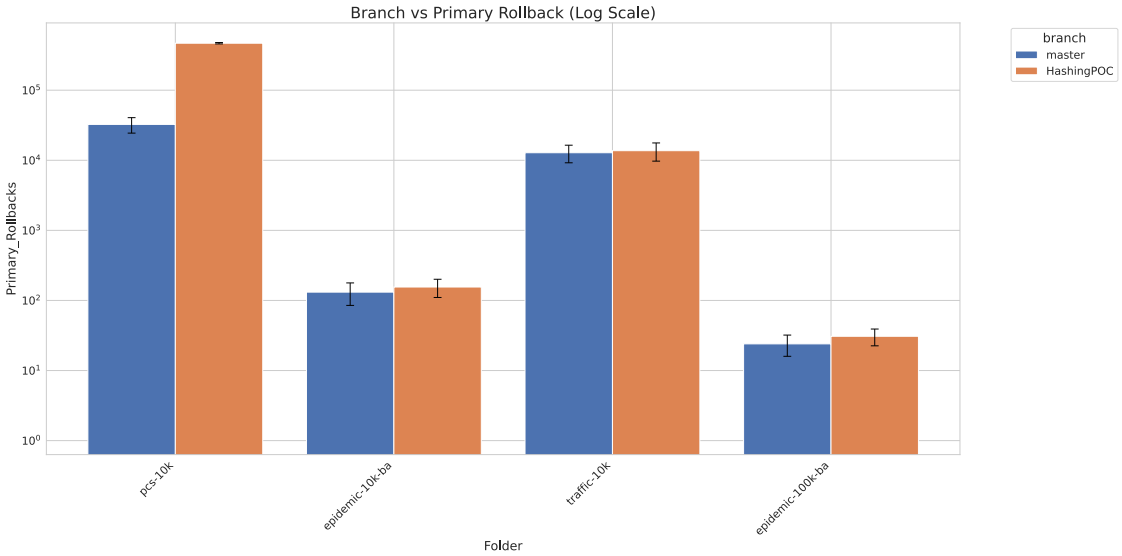

Figure 49: Branch vs Primary Rollback_log

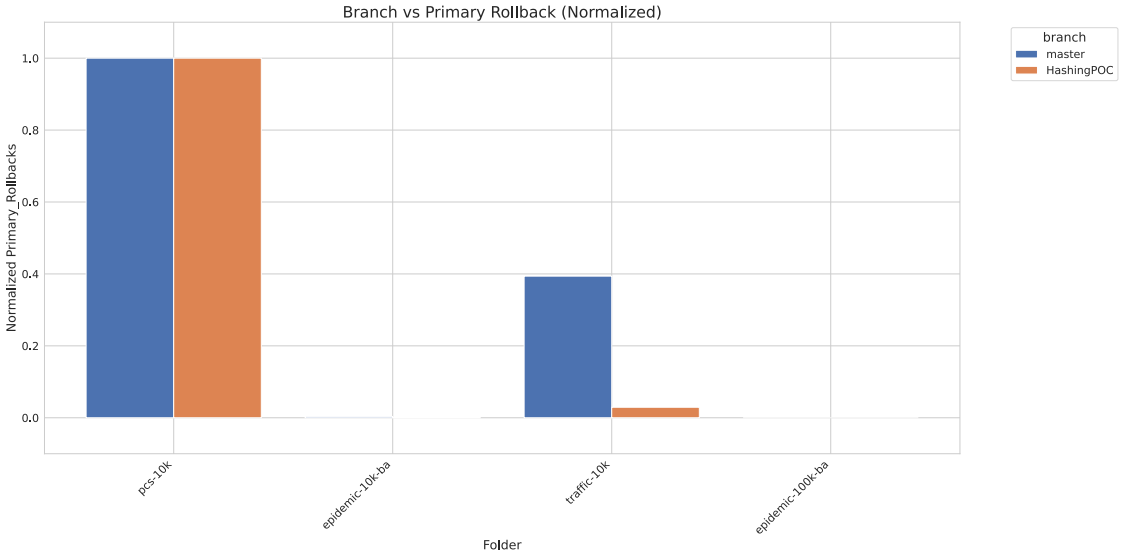

Figure 50: Branch vs Primary Rollback_normalized

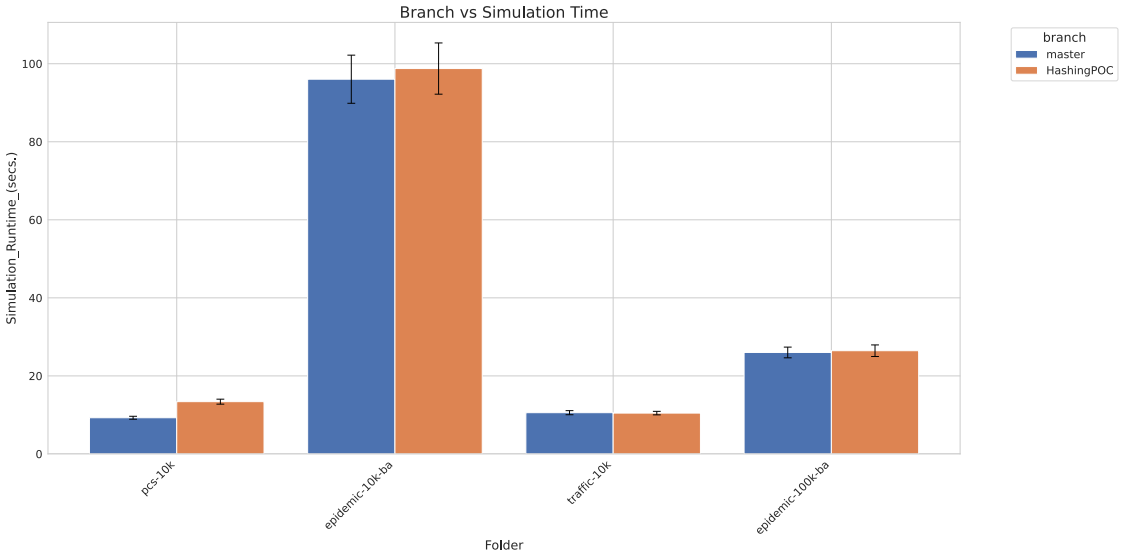

Figure 51: Branch vs Simulation Time

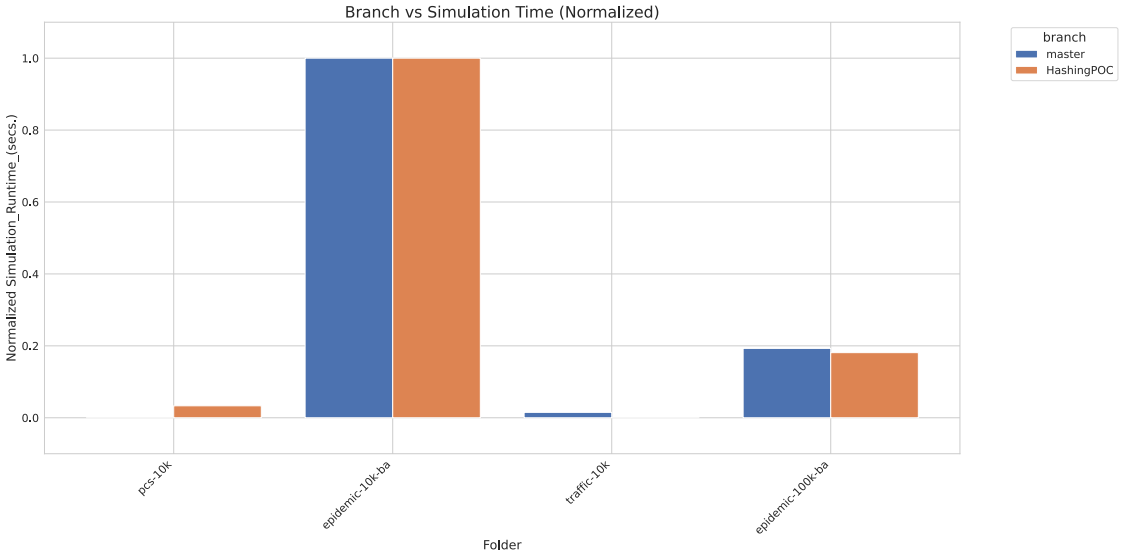

Figure 52: Branch vs Simulation Time_normalized

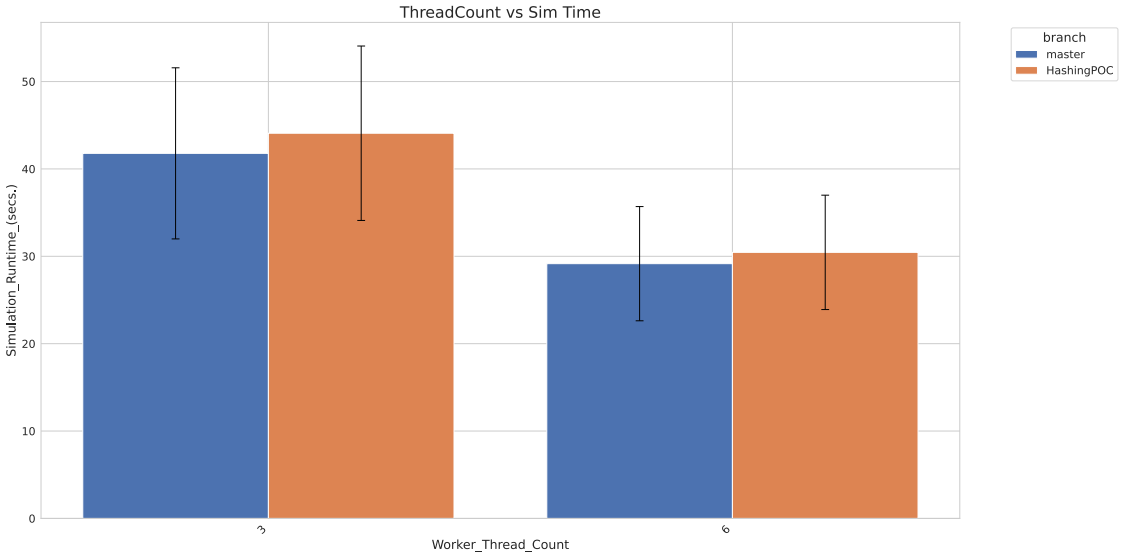

Figure 53: ThreadCount vs Sim Time

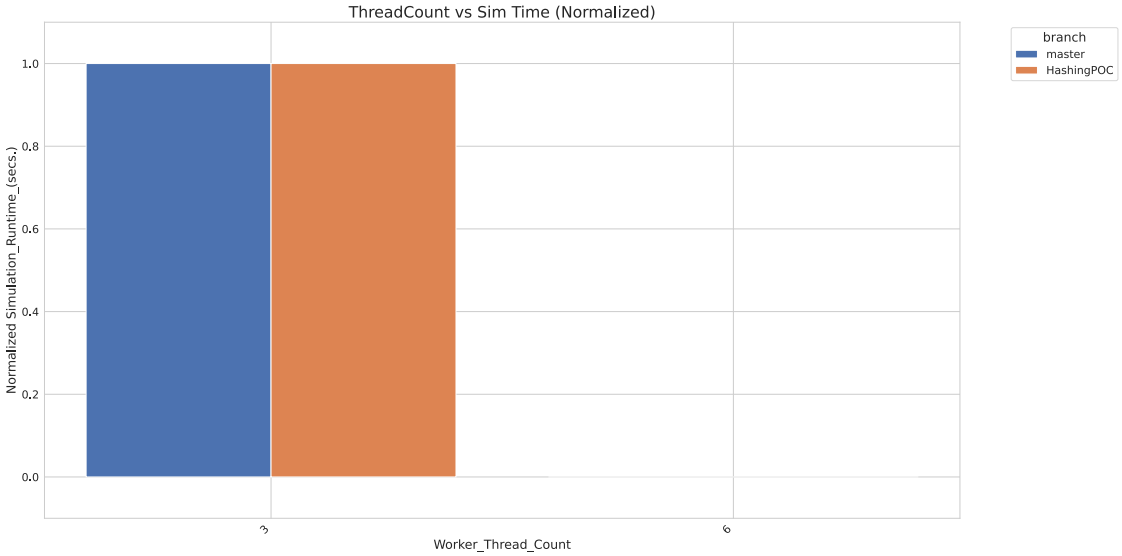

Figure 54: ThreadCount vs Sim Time_normalized

9 SIMD

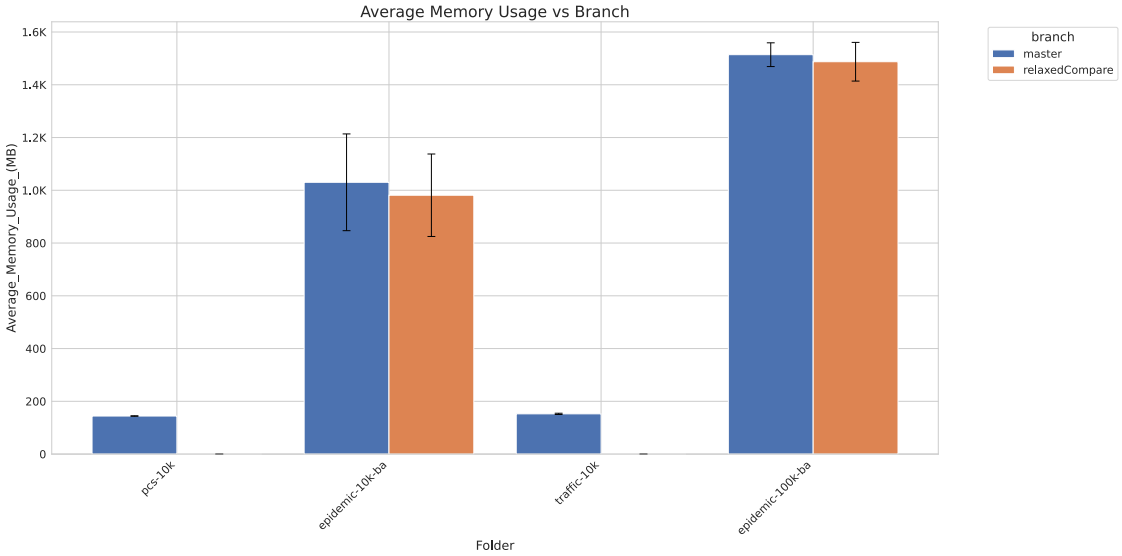

Figure 55: Average Memory Usage vs Branch

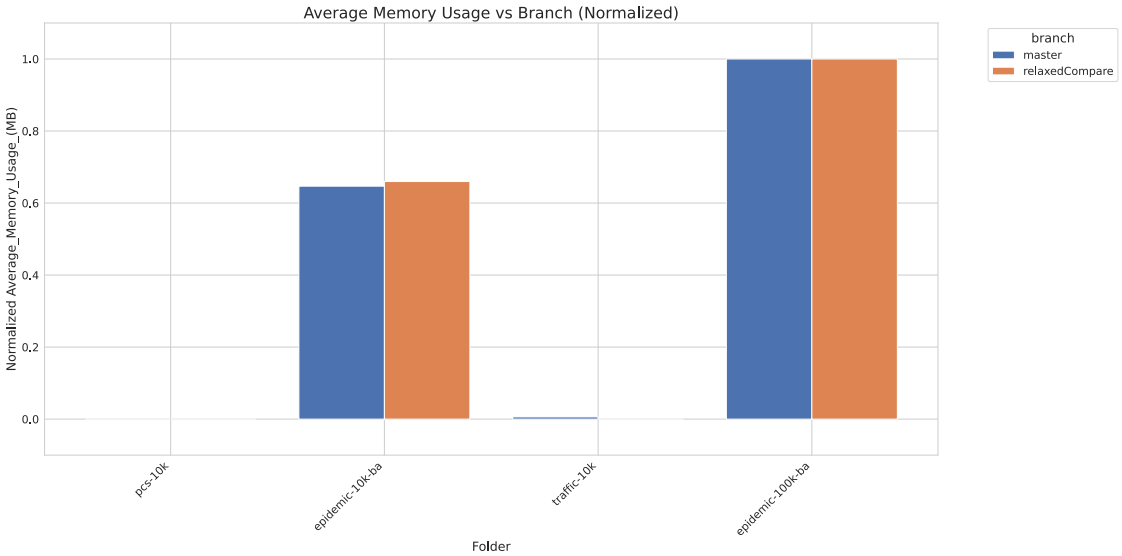

Figure 56: Average Memory Usage vs Branch_normalized

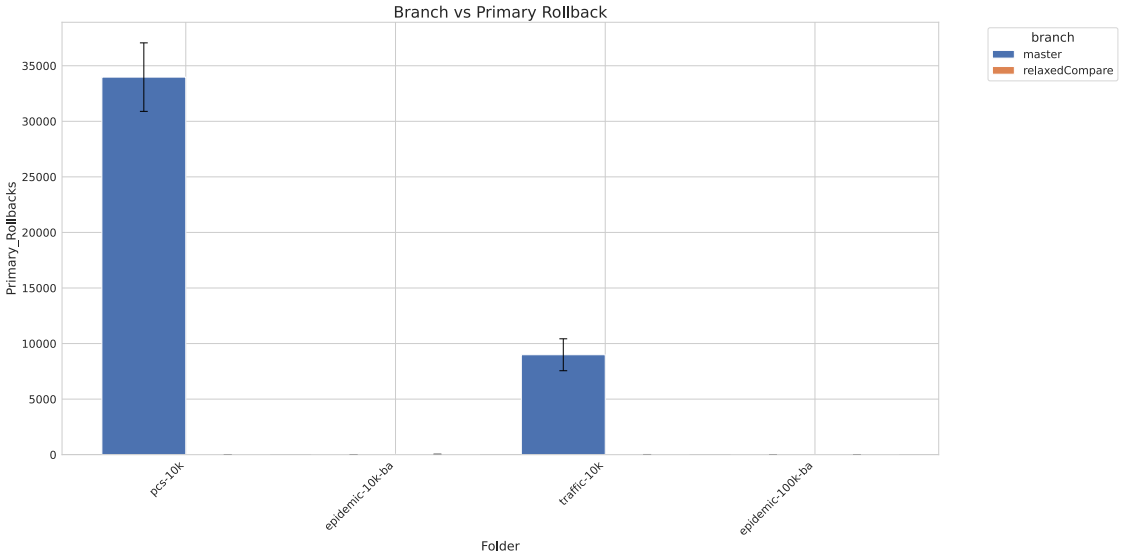

Figure 57: Branch vs Primary Rollback

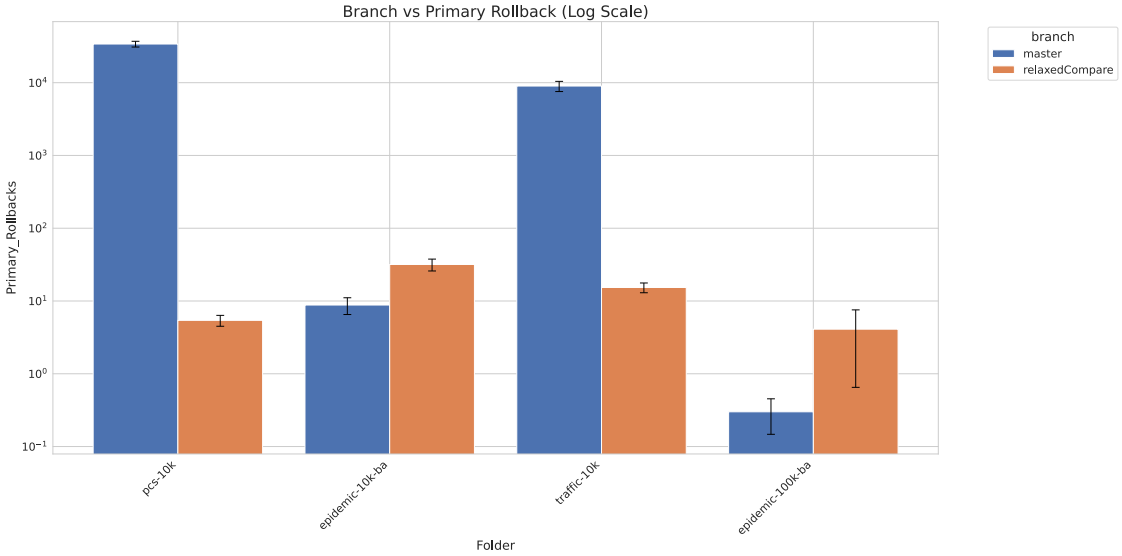

Figure 58: Branch vs Primary Rollback_log

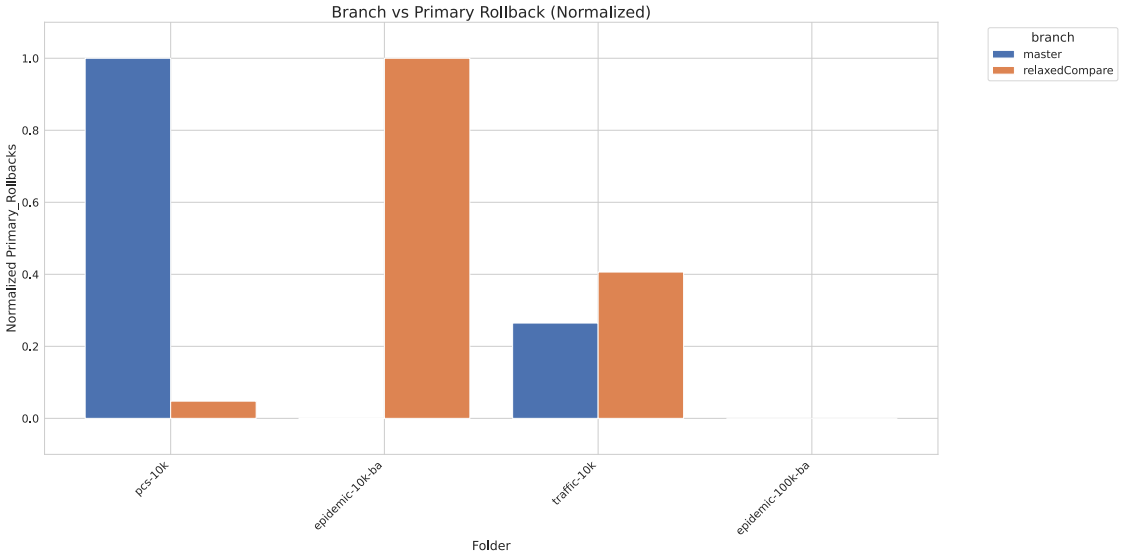

Figure 59: Branch vs Primary Rollback_normalized

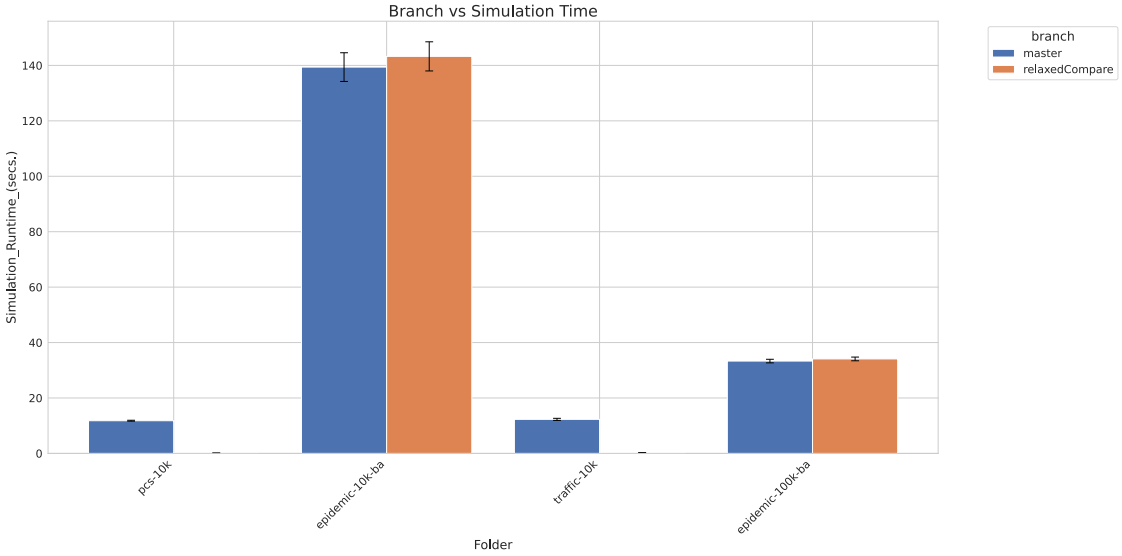

Figure 60: Branch vs Simulation Time

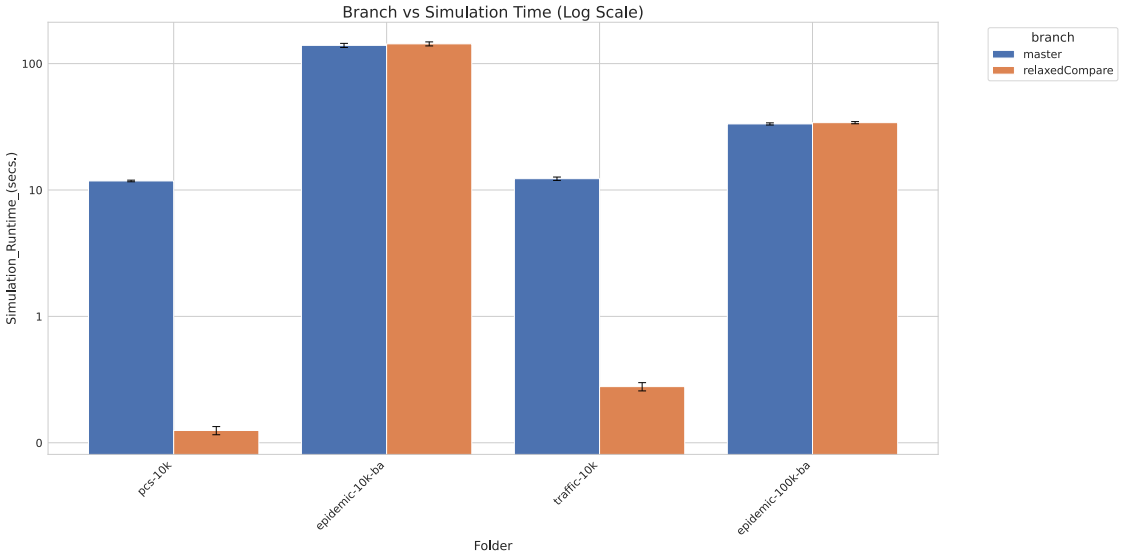

Figure 61: Branch vs Simulation Time_log

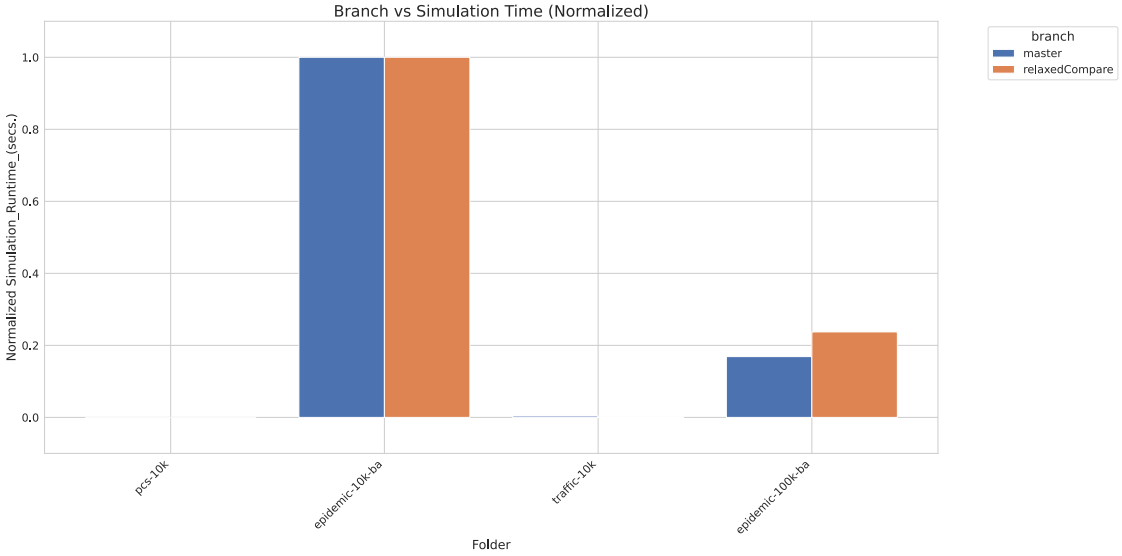

Figure 62: Branch vs Simulation Time_normalized

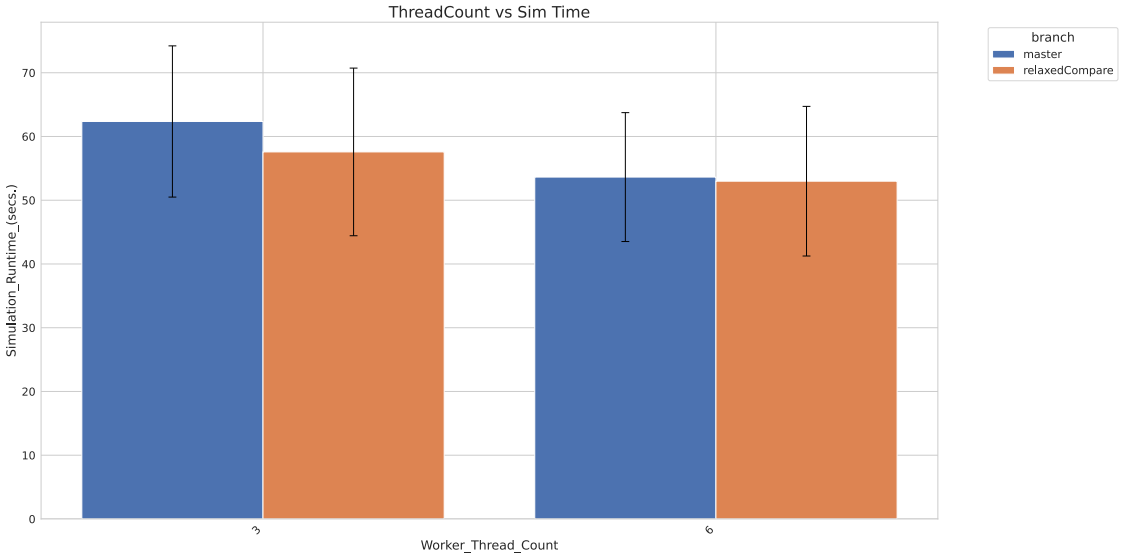

Figure 63: ThreadCount vs Sim Time

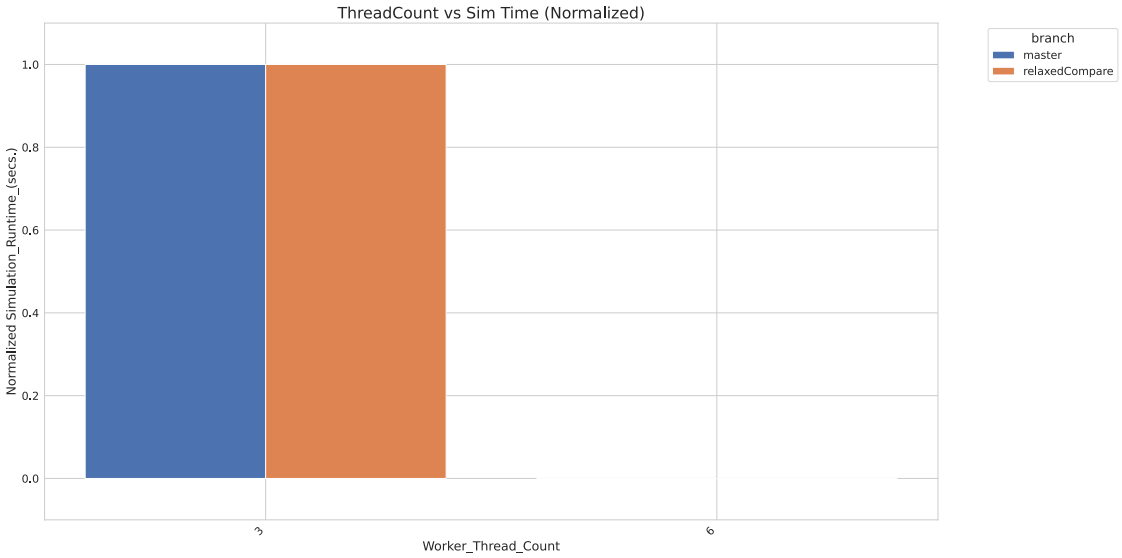

Figure 64: ThreadCount vs Sim Time_normalized

10 SIMD_local

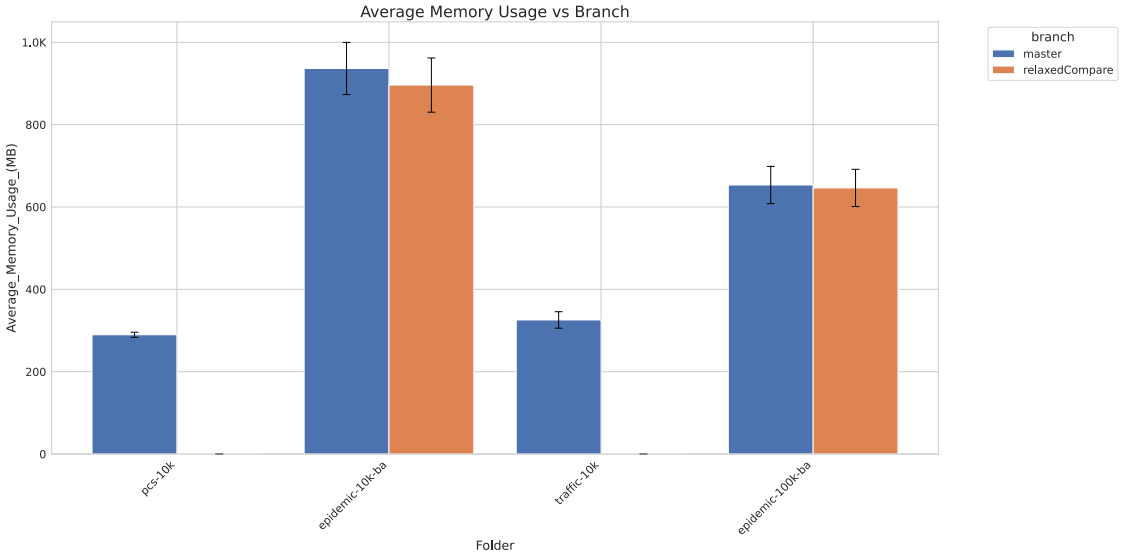

Figure 65: Average Memory Usage vs Branch

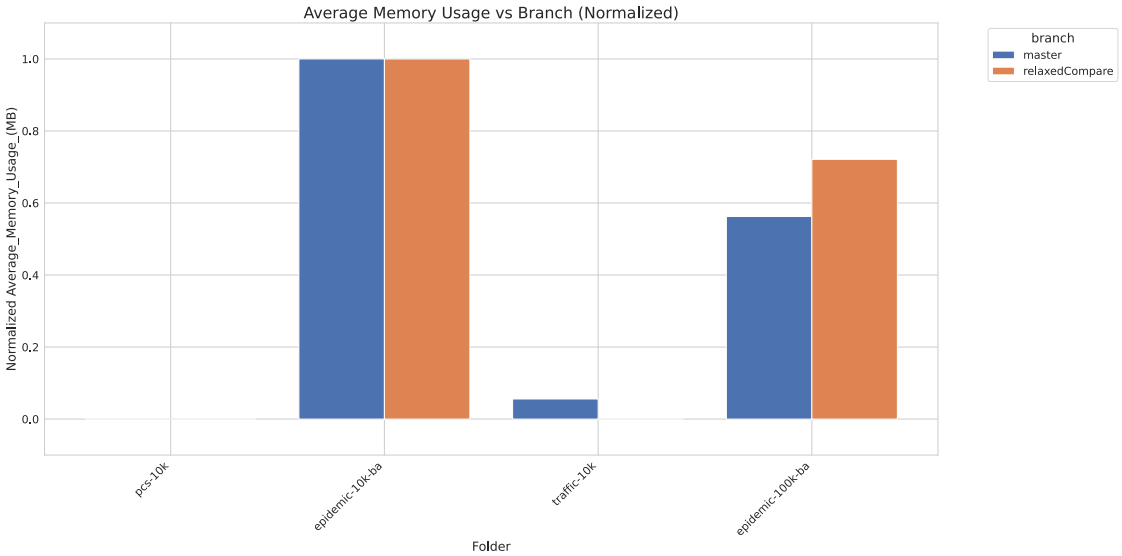

Figure 66: Average Memory Usage vs Branch_normalized

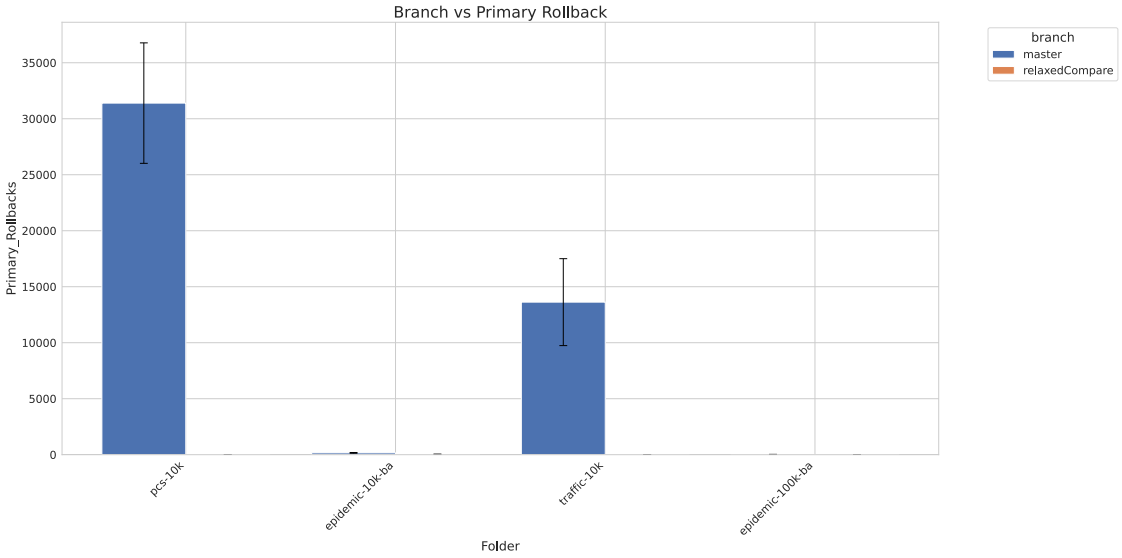

Figure 67: Branch vs Primary Rollback

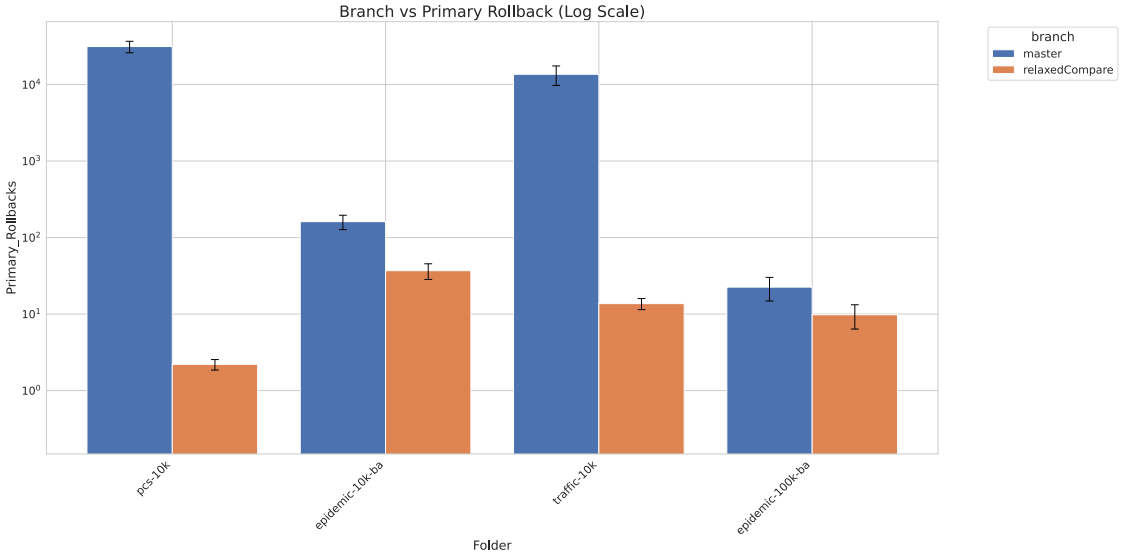

Figure 68: Branch vs Primary Rollback_log

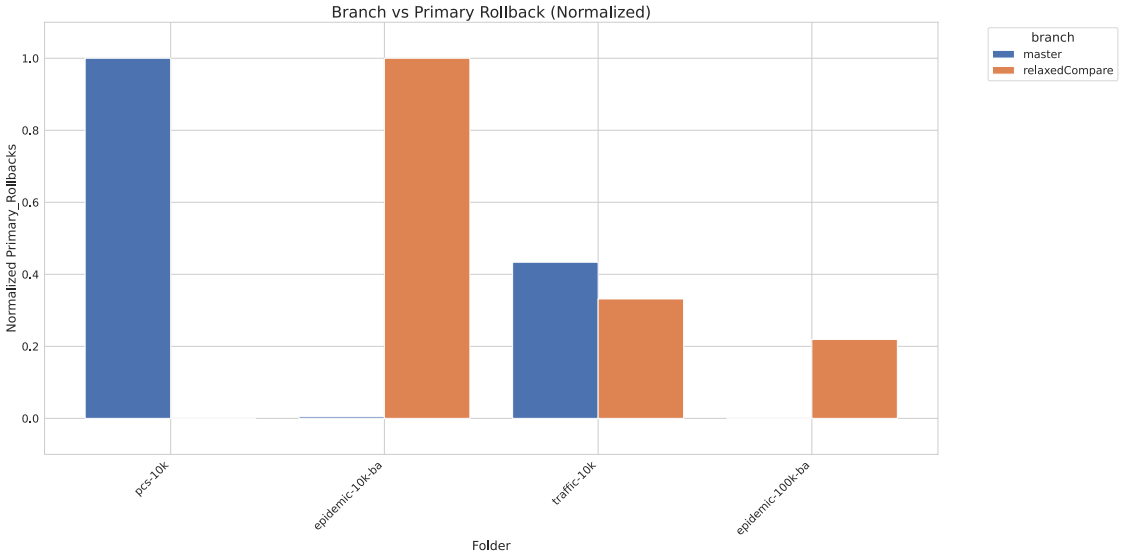

Figure 69: Branch vs Primary Rollback_normalized

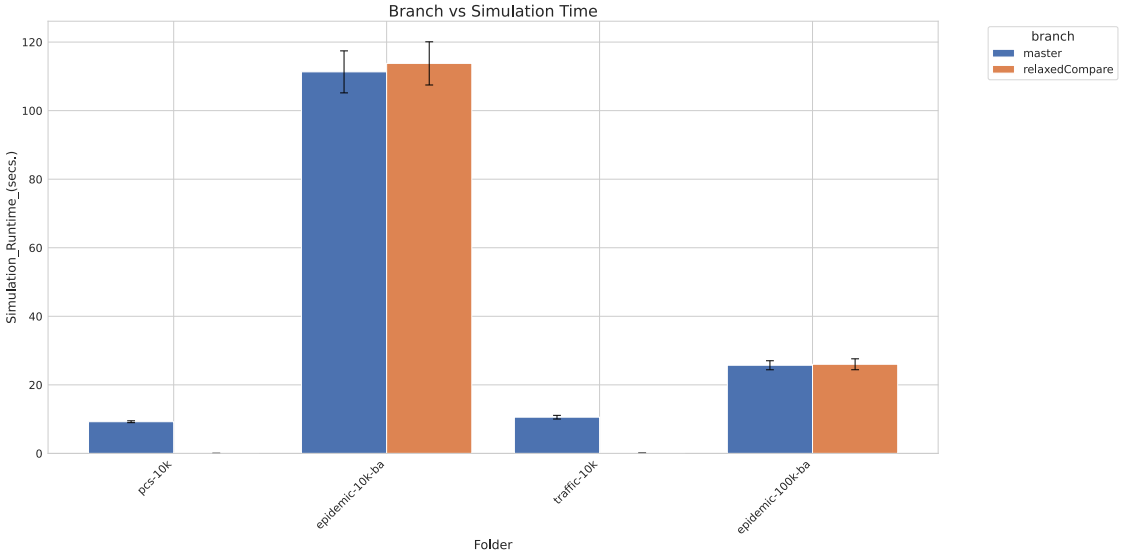

Figure 70: Branch vs Simulation Time

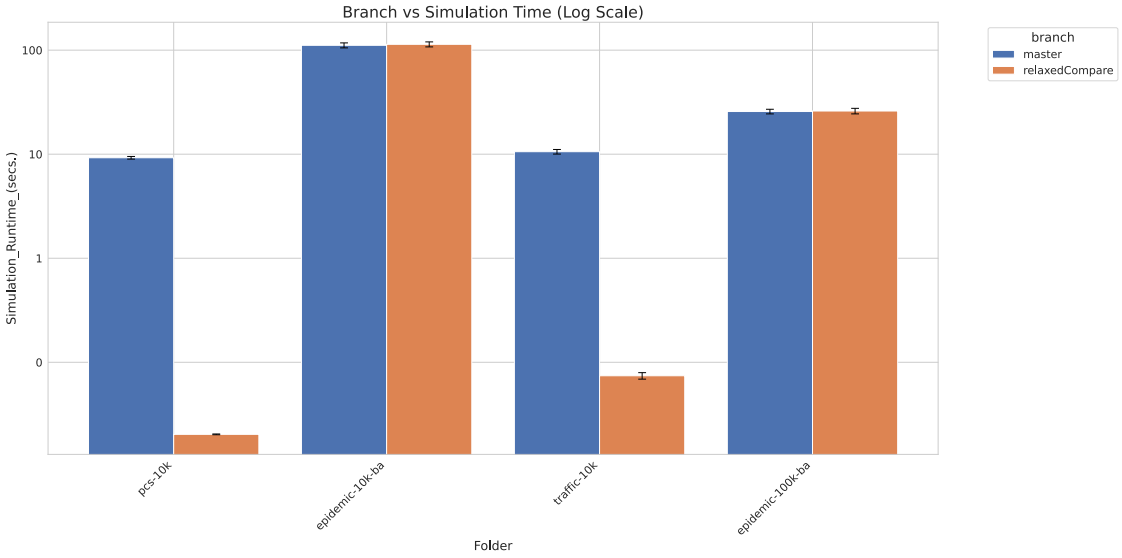

Figure 71: Branch vs Simulation Time_log

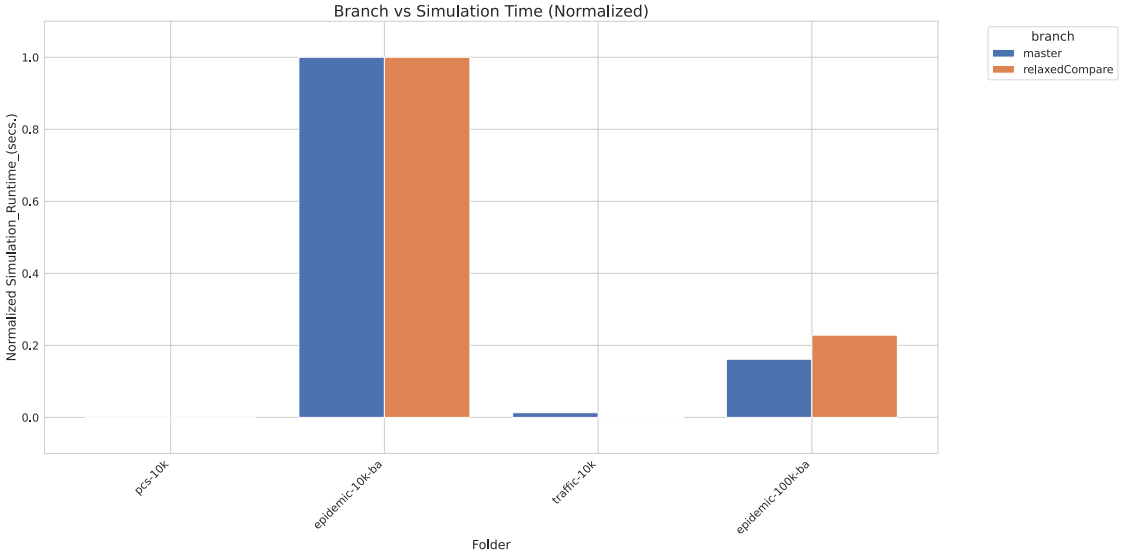

Figure 72: Branch vs Simulation Time_normalized

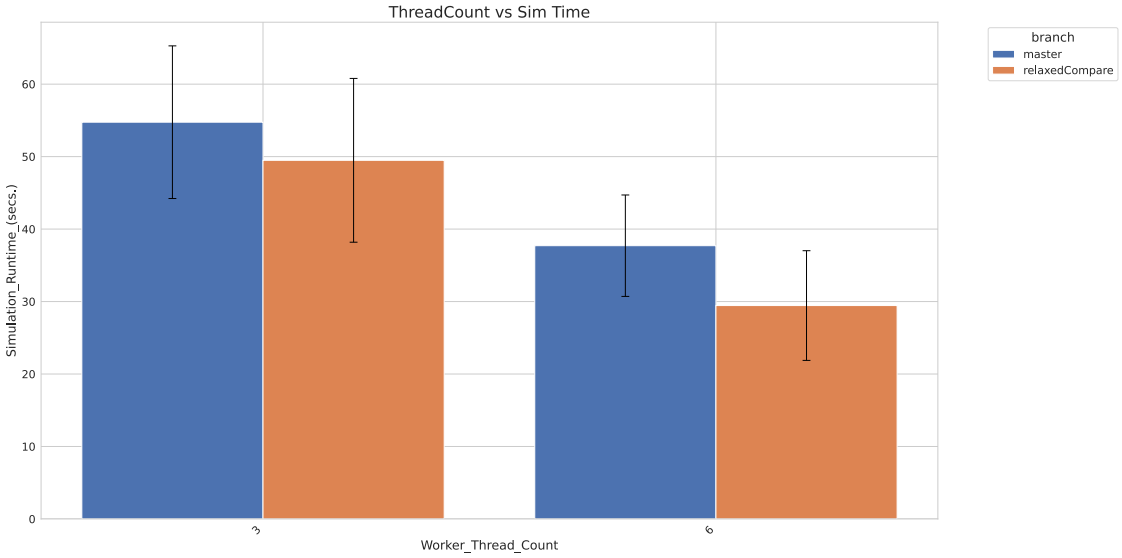

Figure 73: ThreadCount vs Sim Time

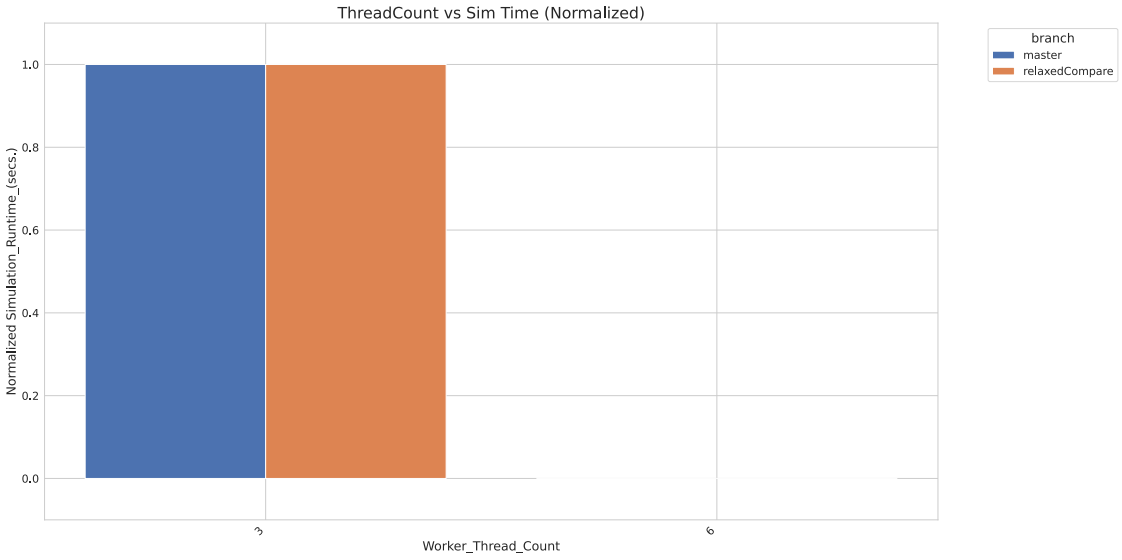

Figure 74: ThreadCount vs Sim Time_normalized

11 SIMDoptiplex

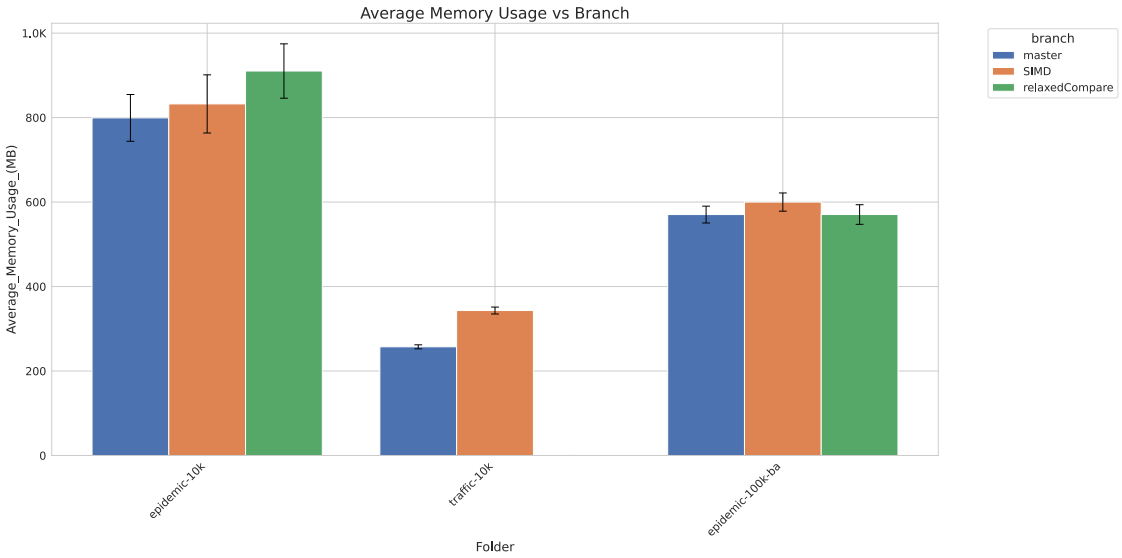

Figure 75: Average Memory Usage vs Branch

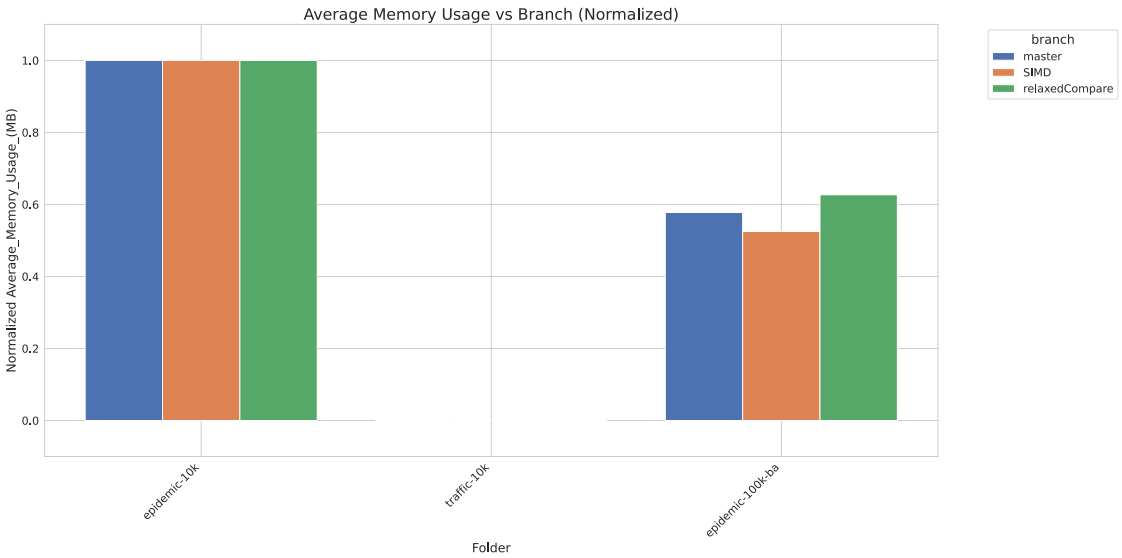

Figure 76: Average Memory Usage vs Branch_normalized

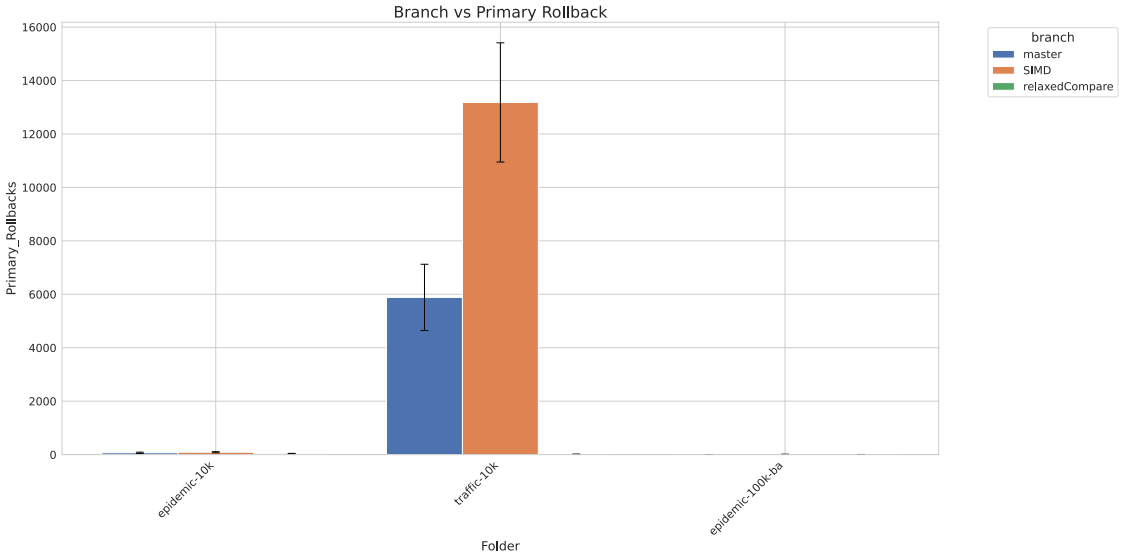

Figure 77: Branch vs Primary Rollback

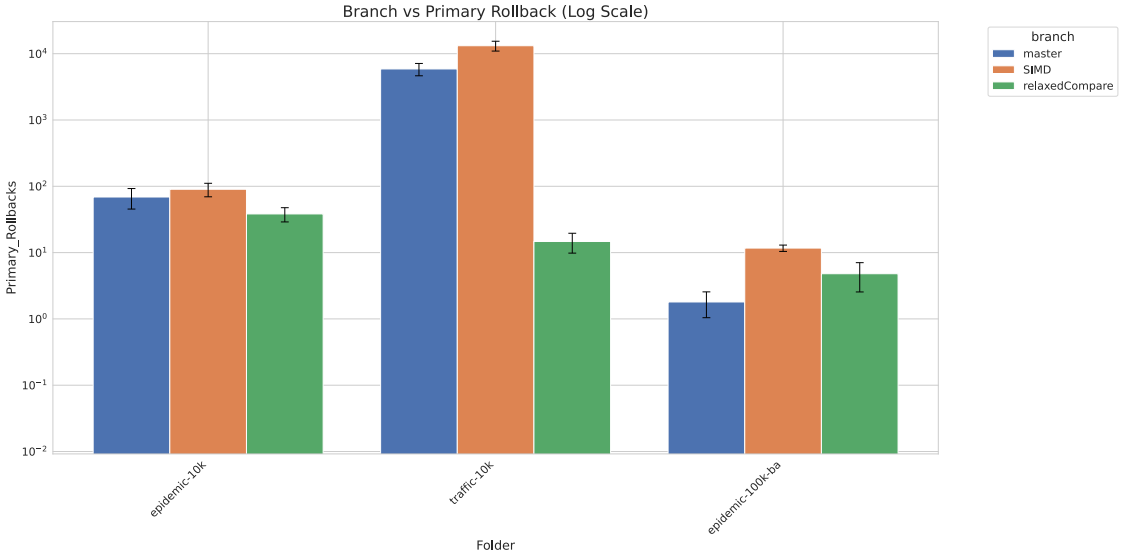

Figure 78: Branch vs Primary Rollback_log

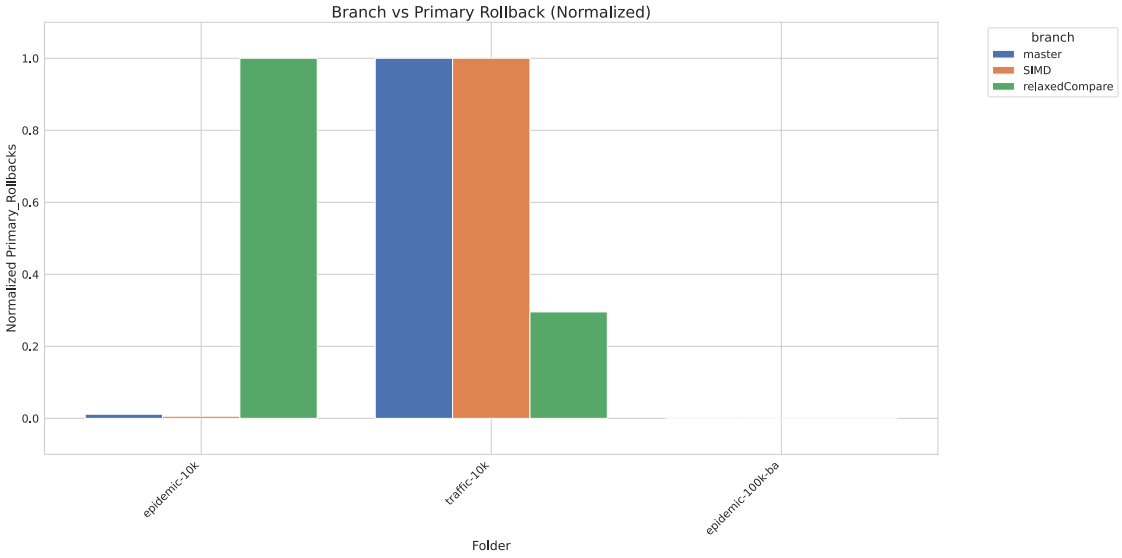

Figure 79: Branch vs Primary Rollback_normalized

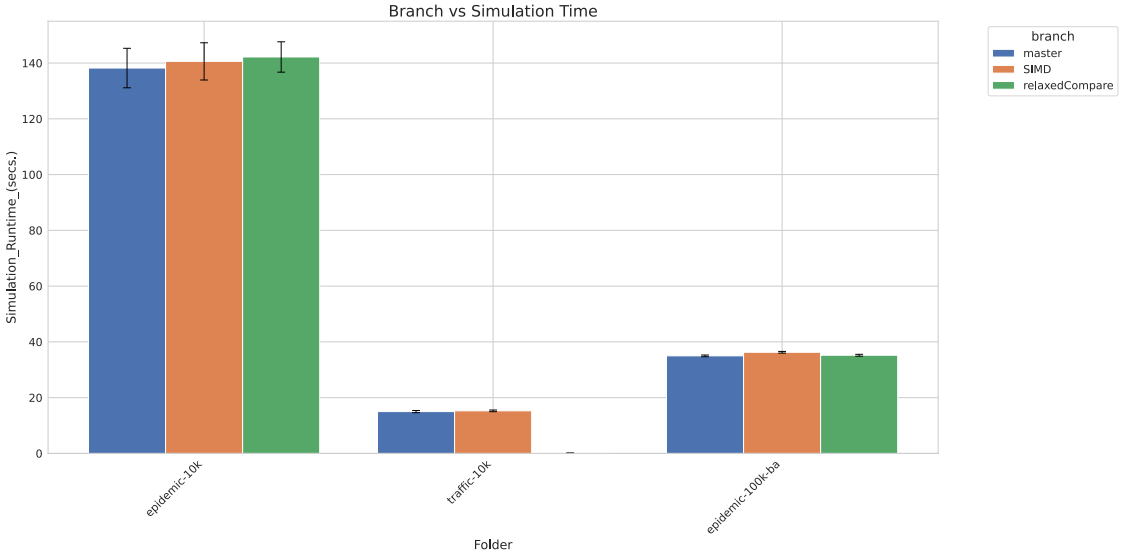

Figure 80: Branch vs Simulation Time

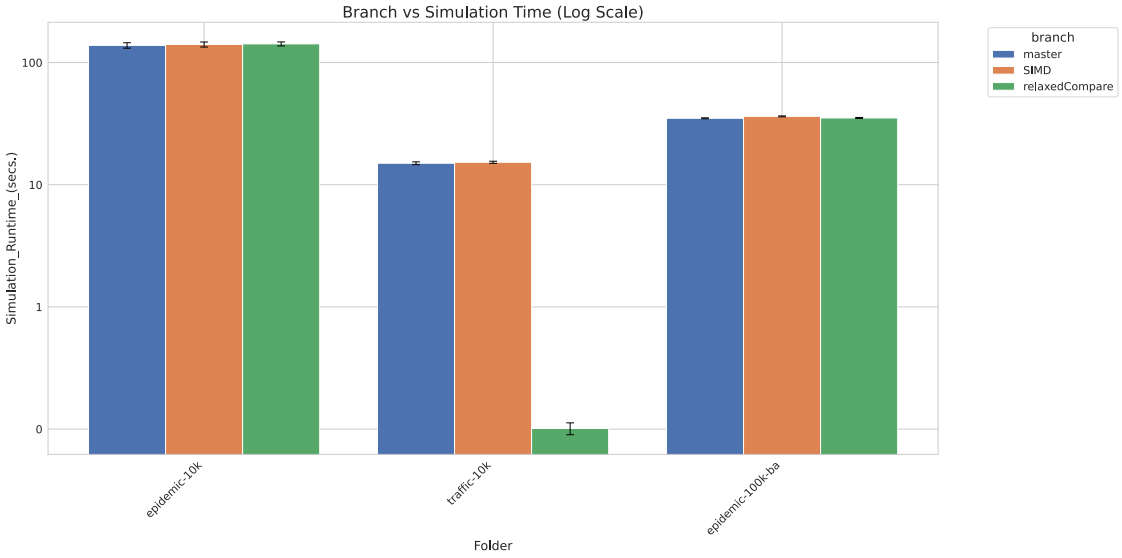

Figure 81: Branch vs Simulation Time_log

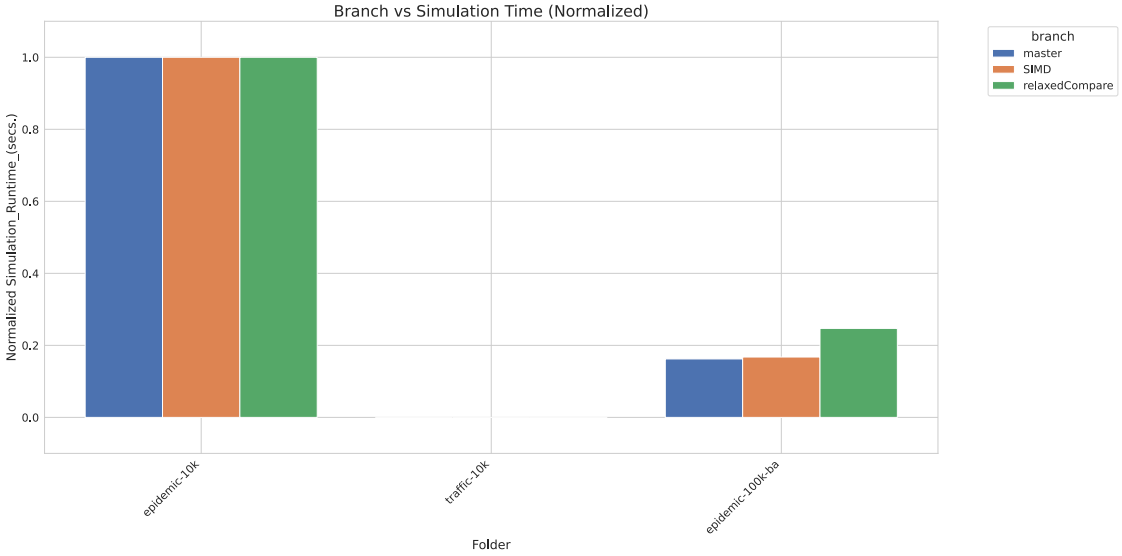

Figure 82: Branch vs Simulation Time_normalized

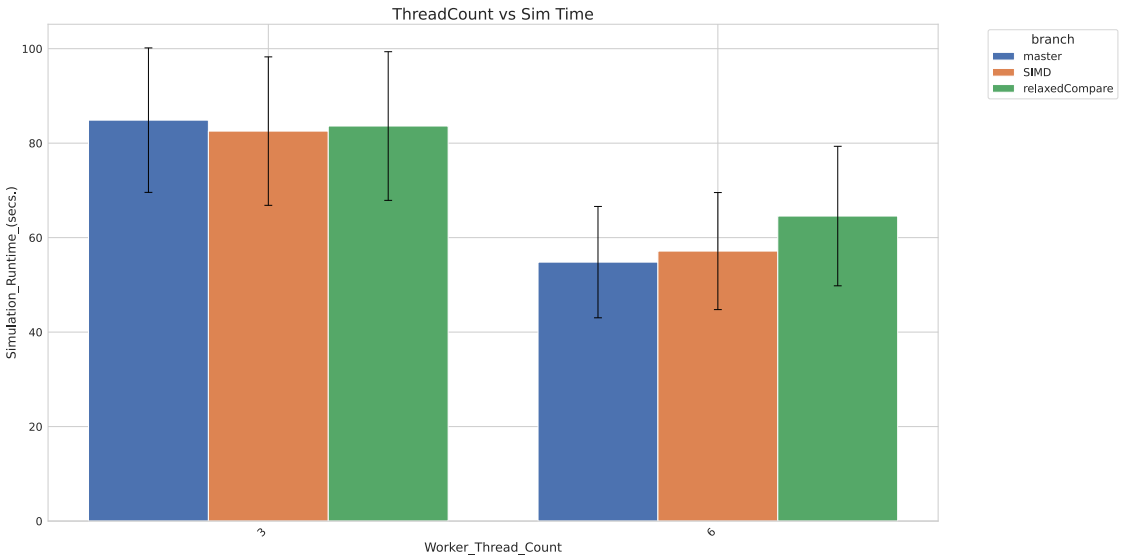

Figure 83: ThreadCount vs Sim Time

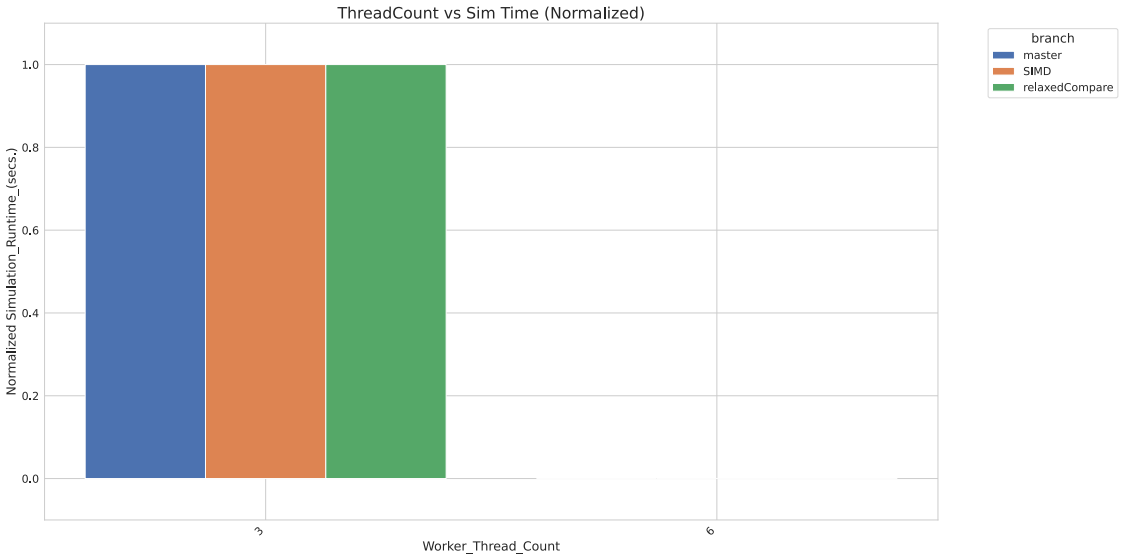

Figure 84: ThreadCount vs Sim Time_normalized

12 Top Performers

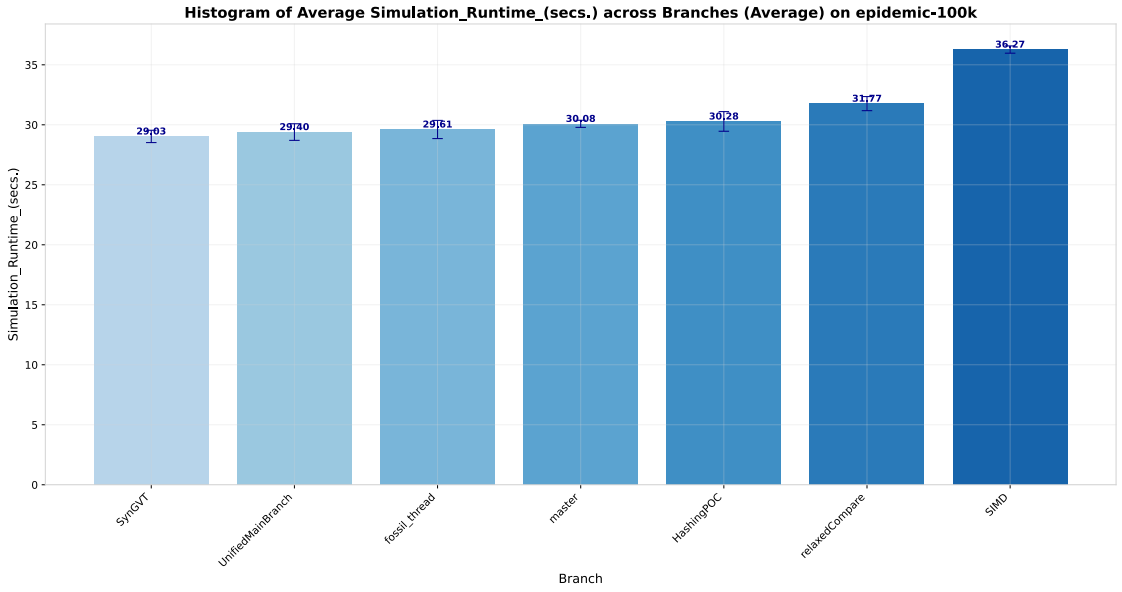

Figure 85: Epidemic100kPerformers_(Average)

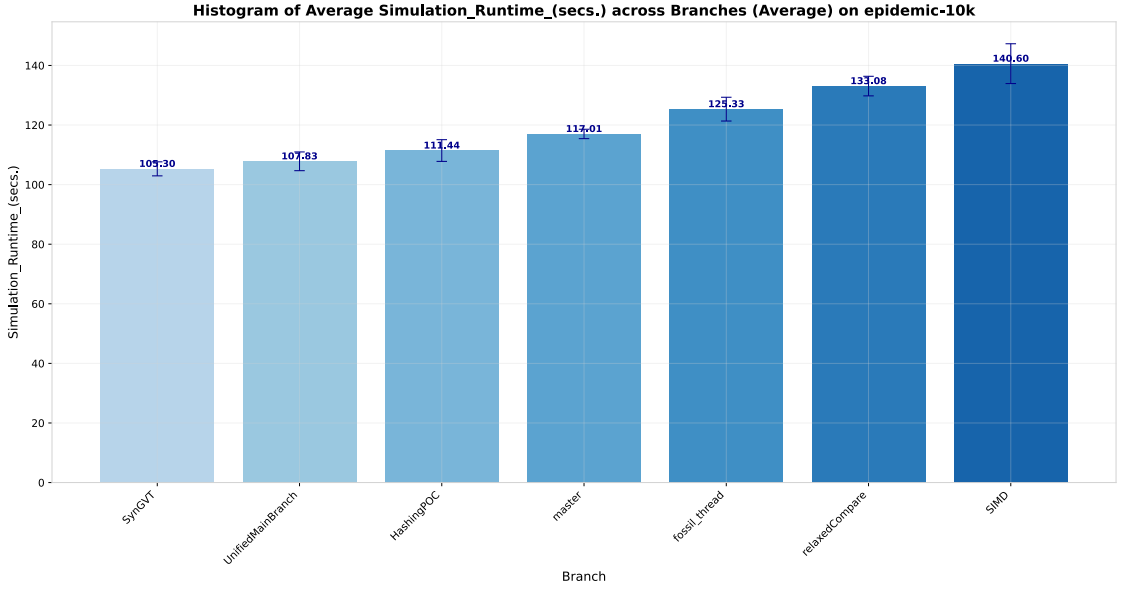

Figure 86: EpidemicPerformers_(Average)

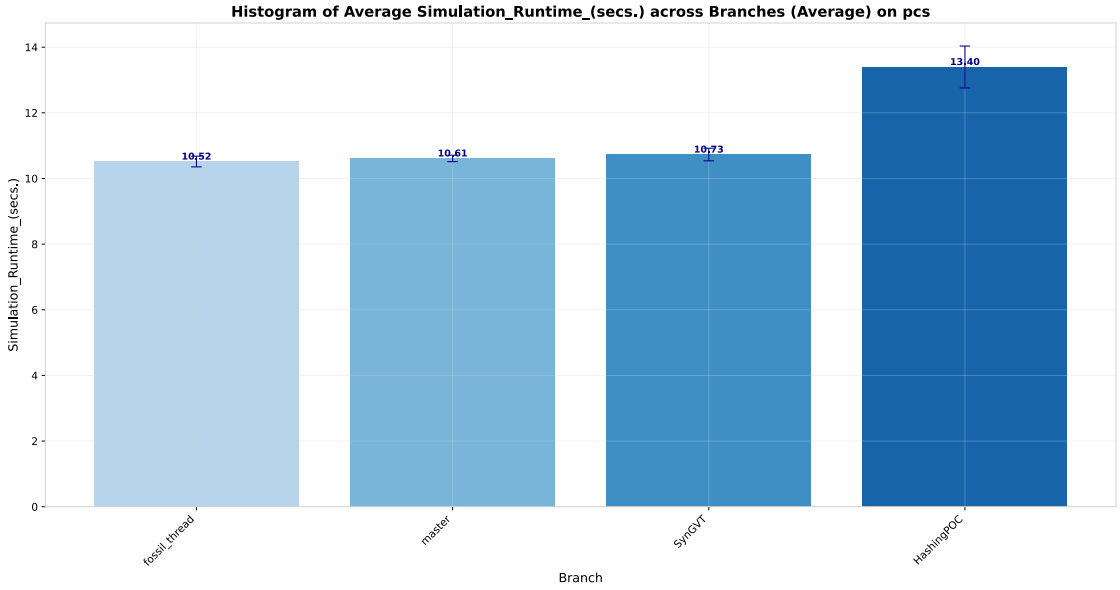

Figure 87: PCSPerformers_(Average)

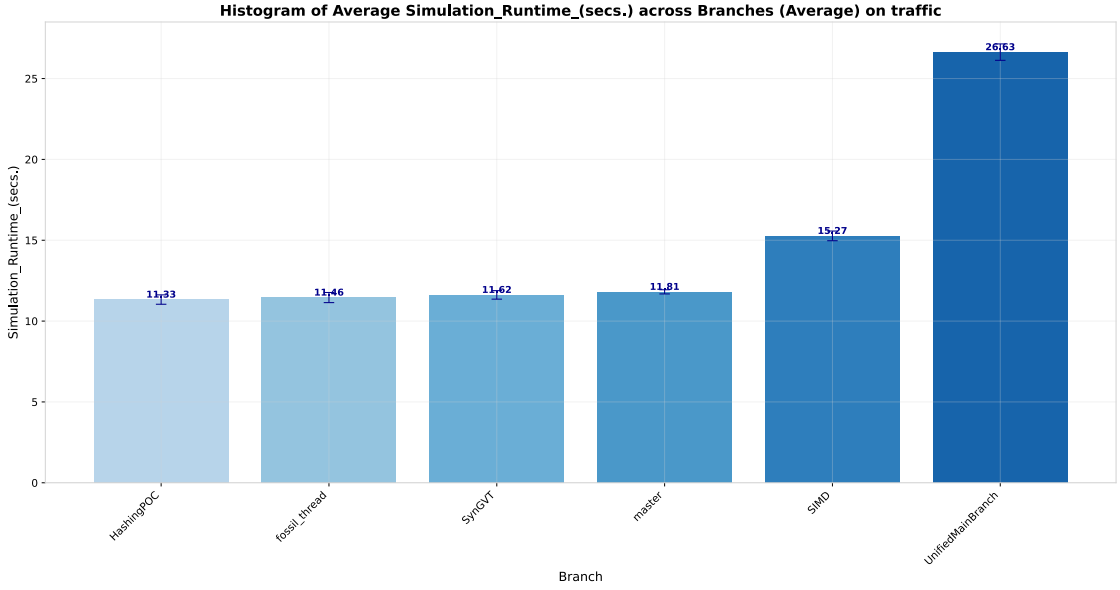

Figure 88: TrafficPerformers_(Average)

13 unified

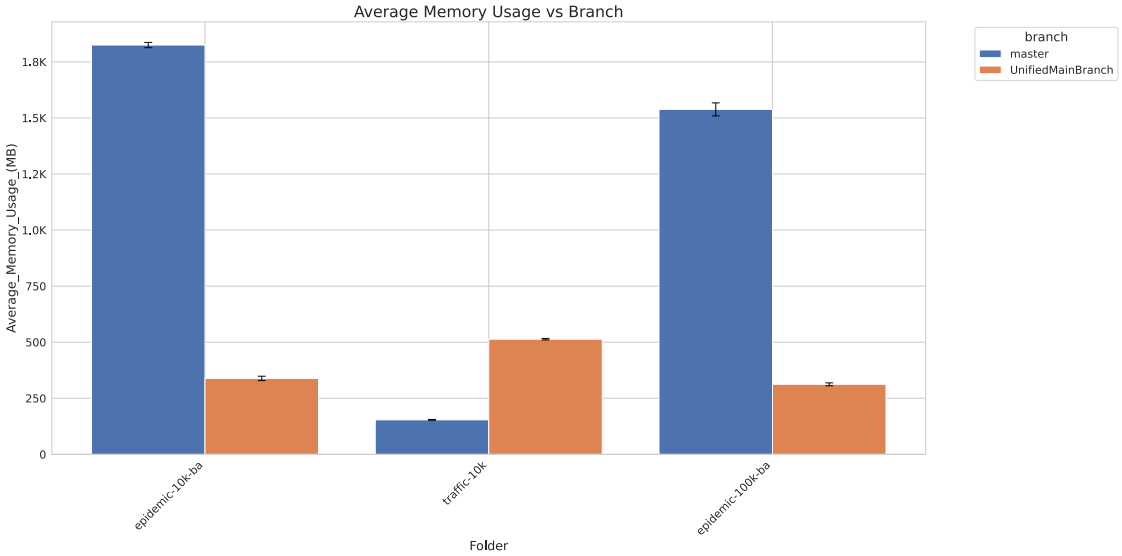

Figure 89: Average Memory Usage vs Branch

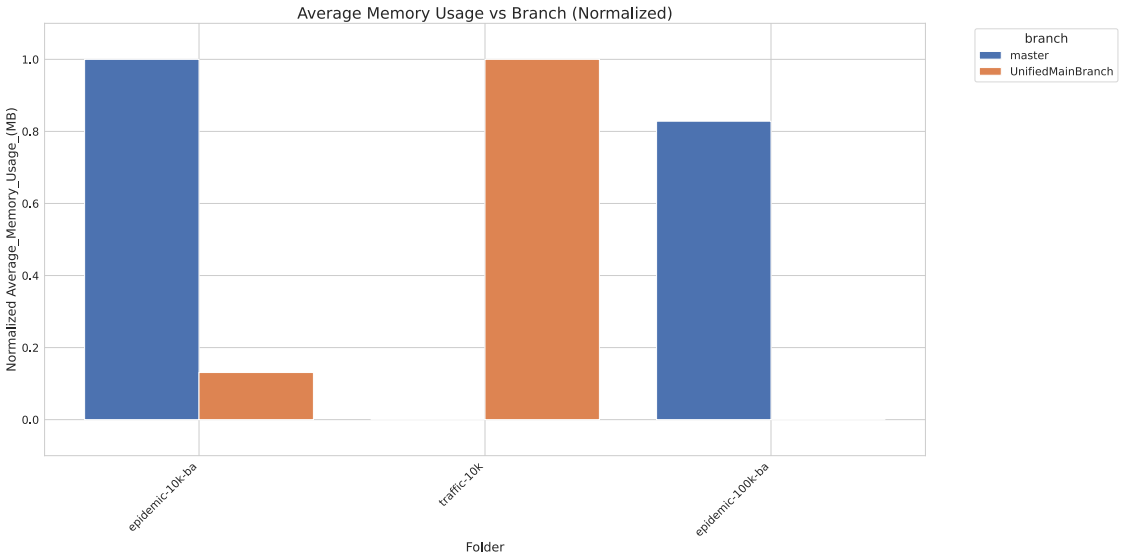

Figure 90: Average Memory Usage vs Branch_normalized

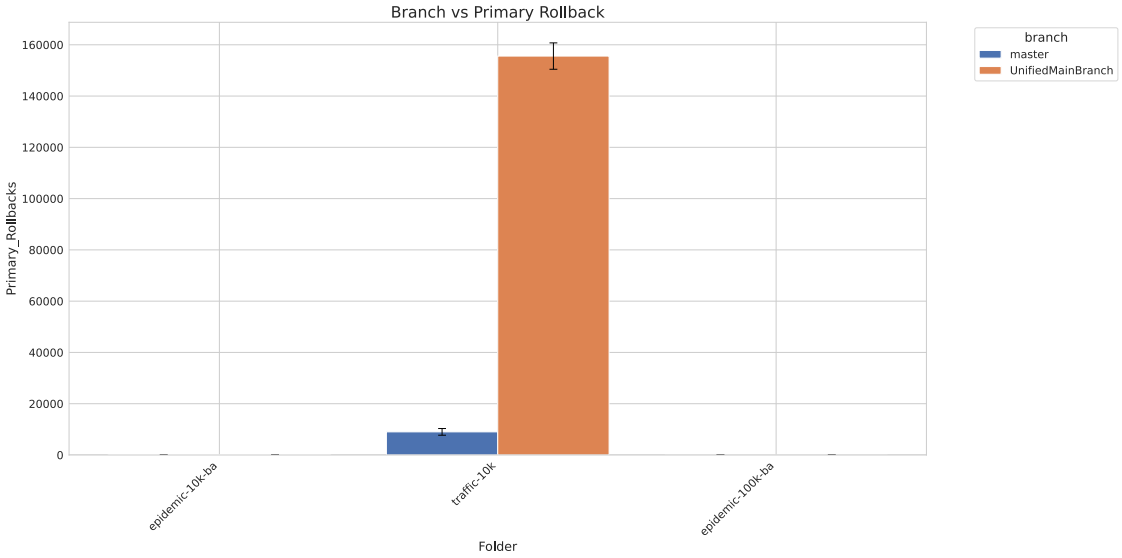

Figure 91: Branch vs Primary Rollback

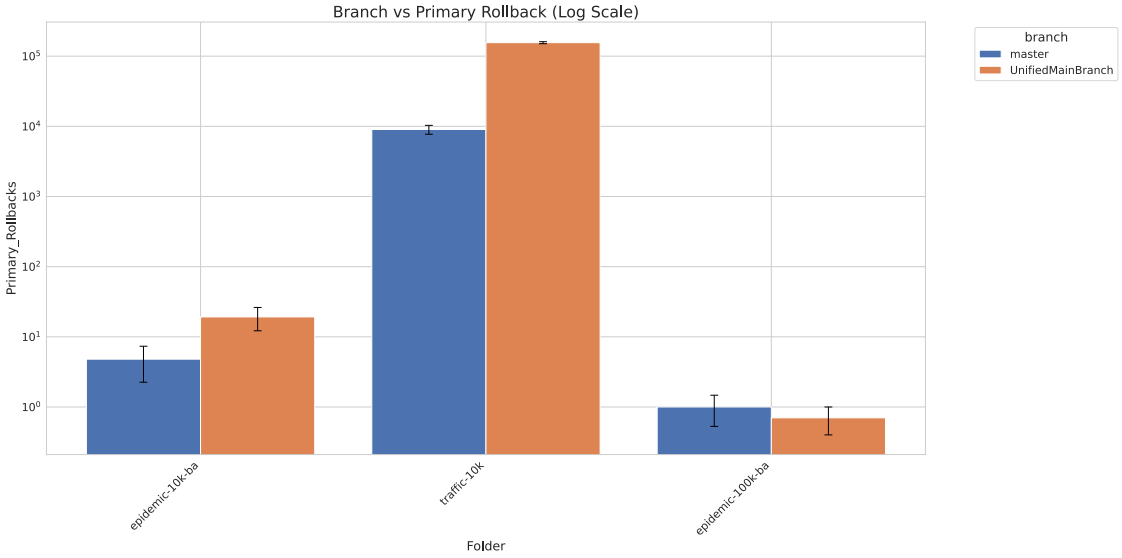

Figure 92: Branch vs Primary Rollback_log

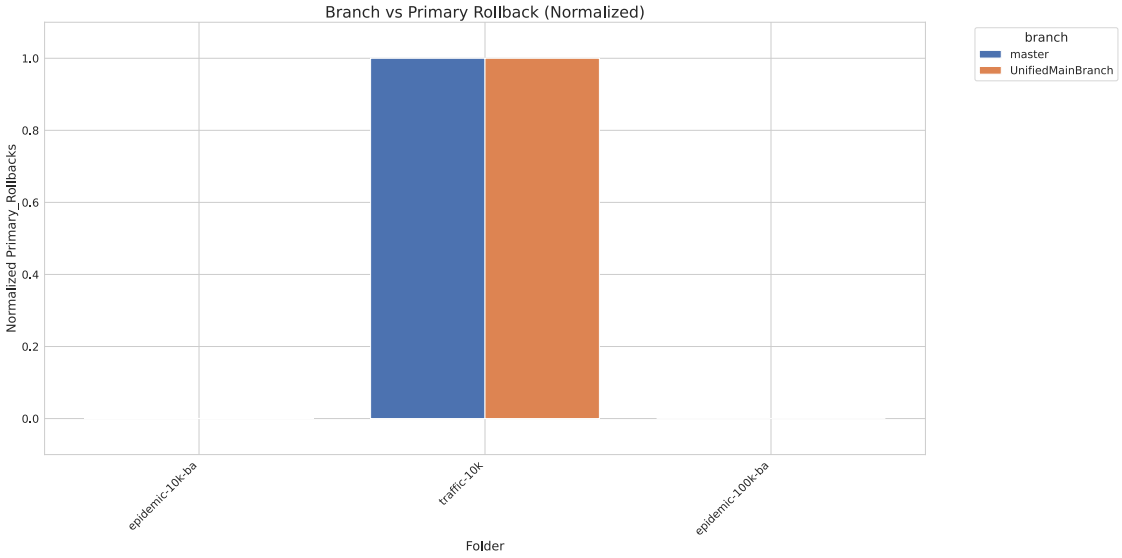

Figure 93: Branch vs Primary Rollback_normalized

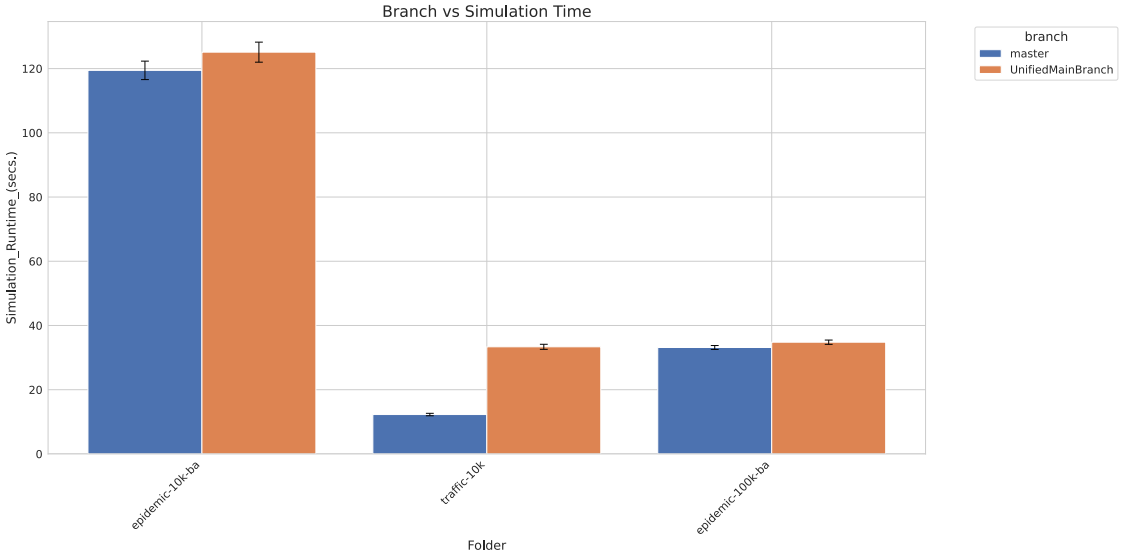

Figure 94: Branch vs Simulation Time

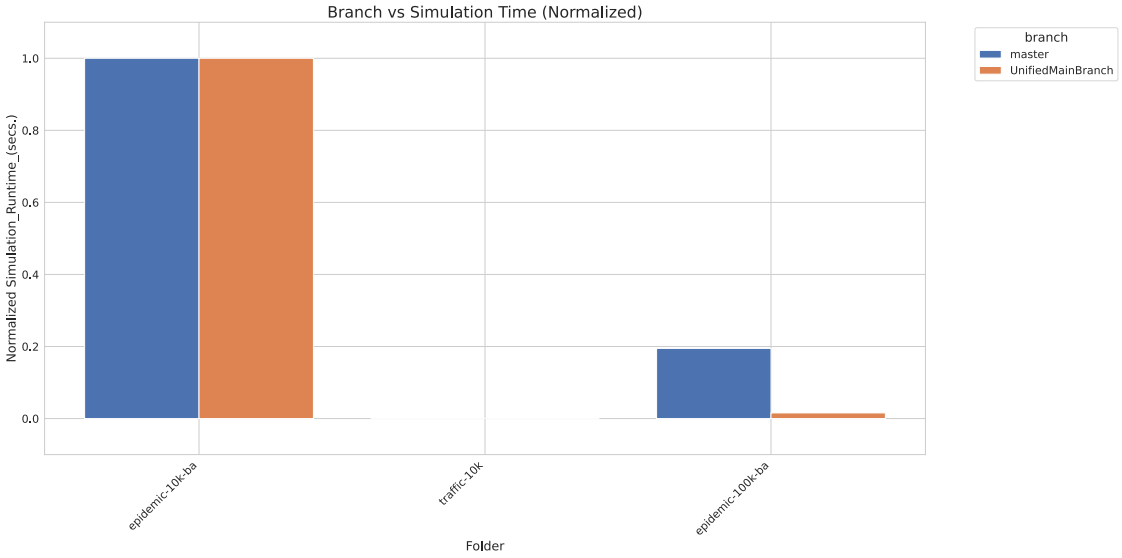

Figure 95: Branch vs Simulation Time_normalized

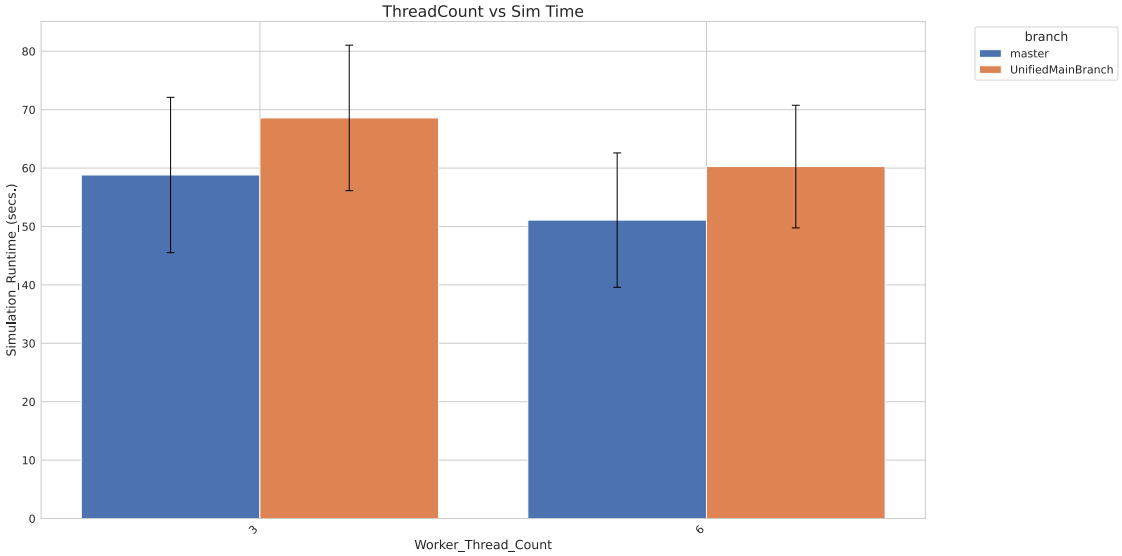

Figure 96: ThreadCount vs Sim Time

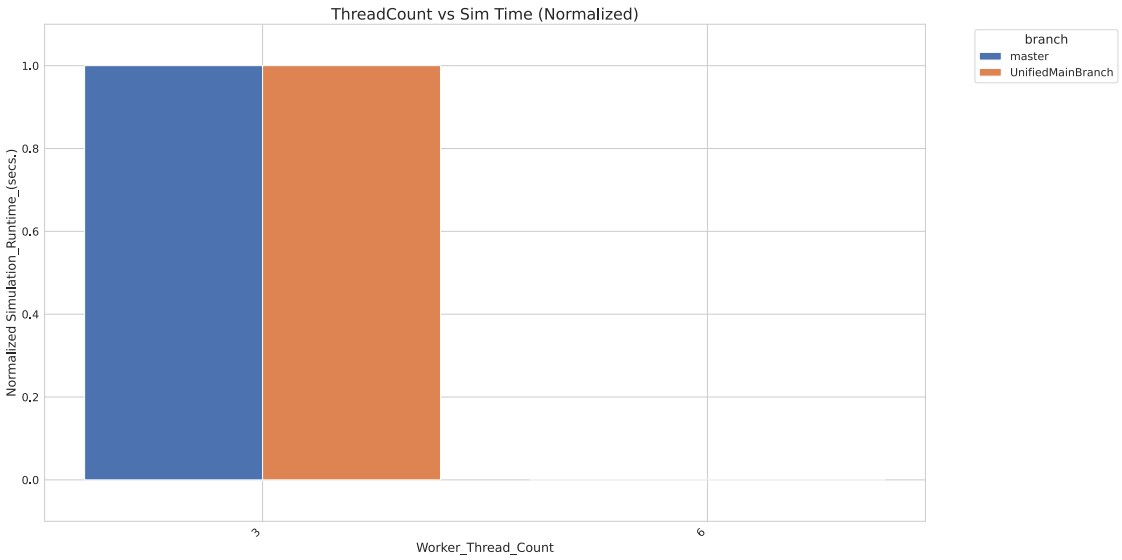

Figure 97: ThreadCount vs Sim Time_normalized

14 unified_local

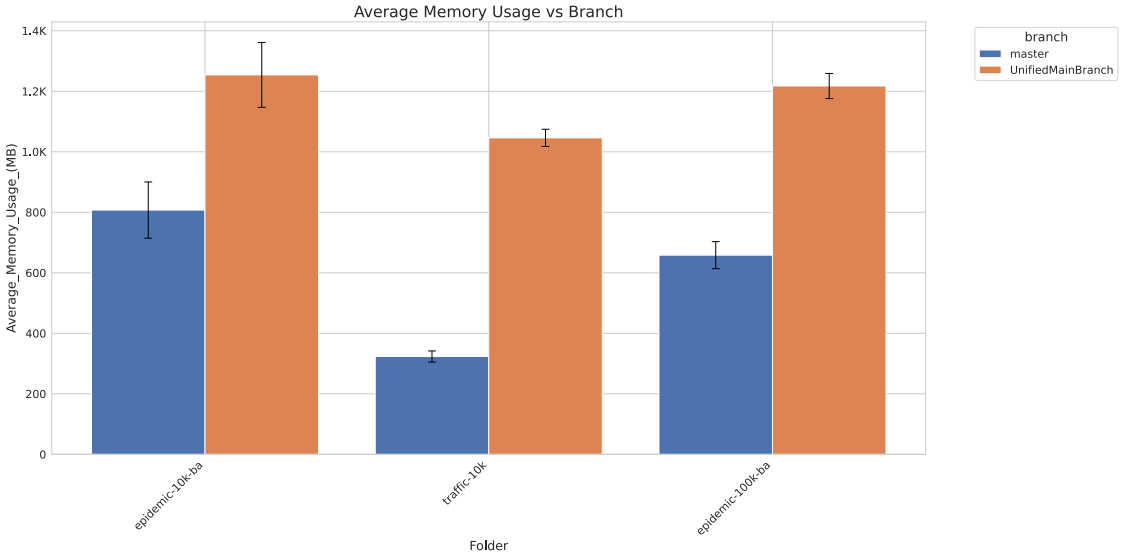

Figure 98: Average Memory Usage vs Branch

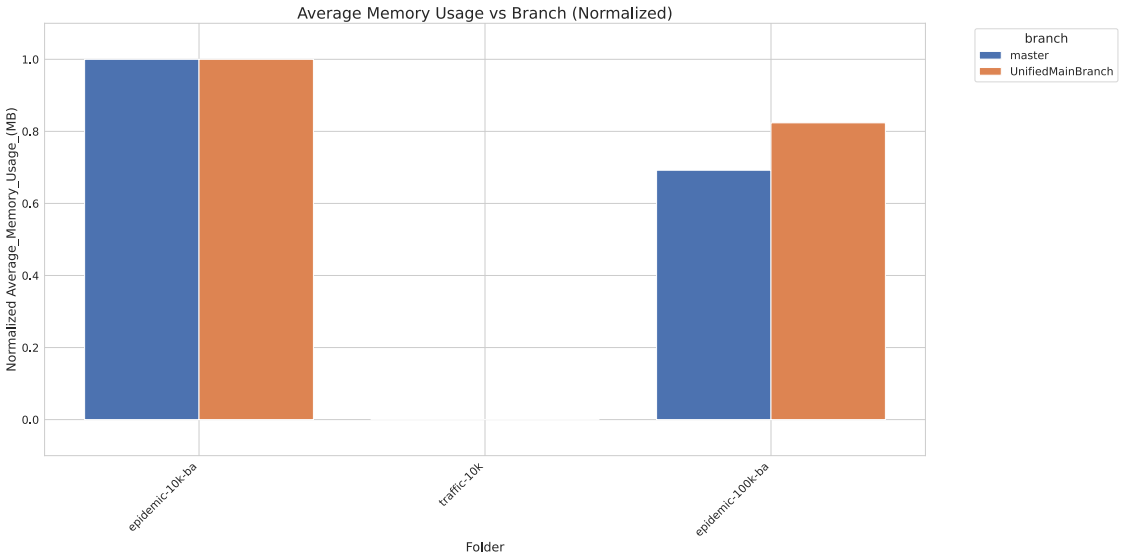

Figure 99: Average Memory Usage vs Branch_normalized

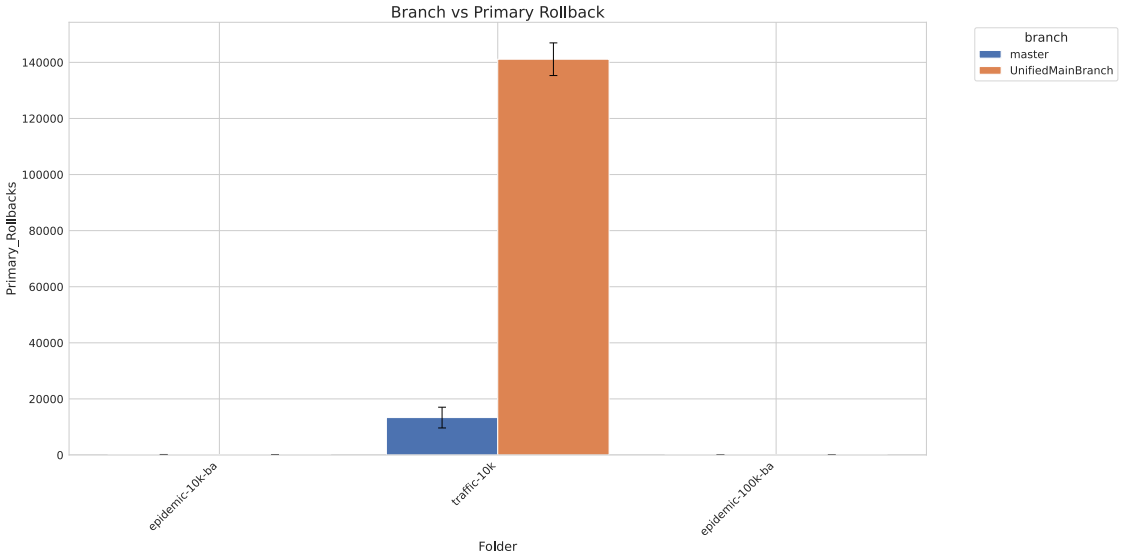

Figure 100: Branch vs Primary Rollback

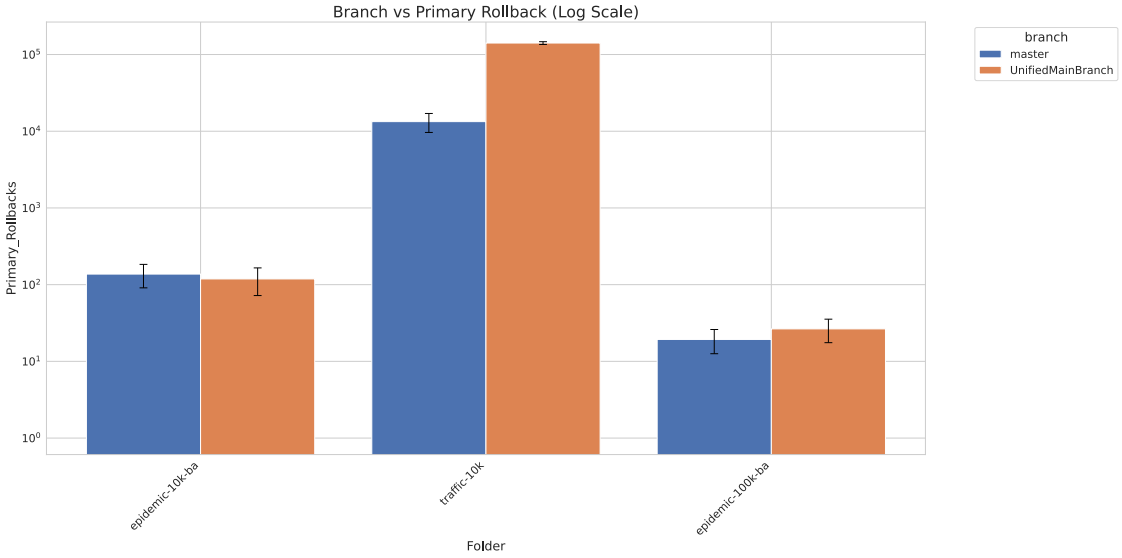

Figure 101: Branch vs Primary Rollback_log

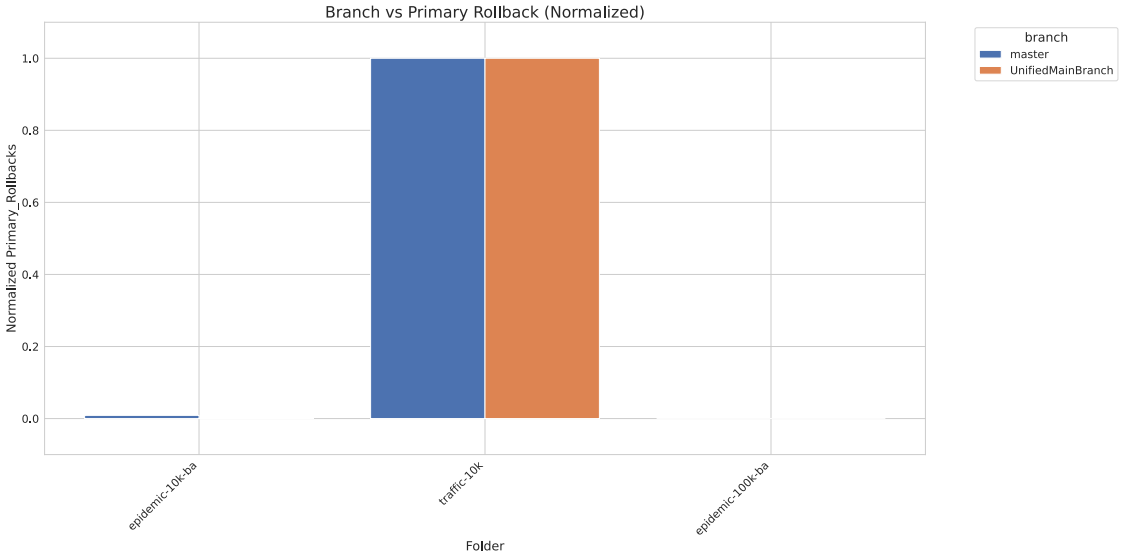

Figure 102: Branch vs Primary Rollback_normalized

Figure 103: Branch vs Simulation Time

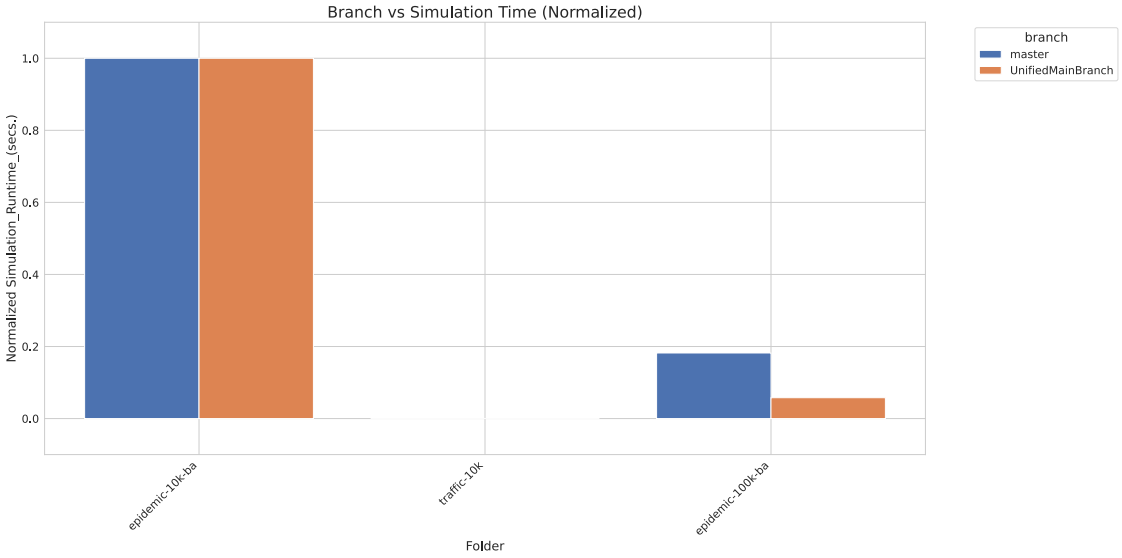

Figure 104: Branch vs Simulation Time_normalized

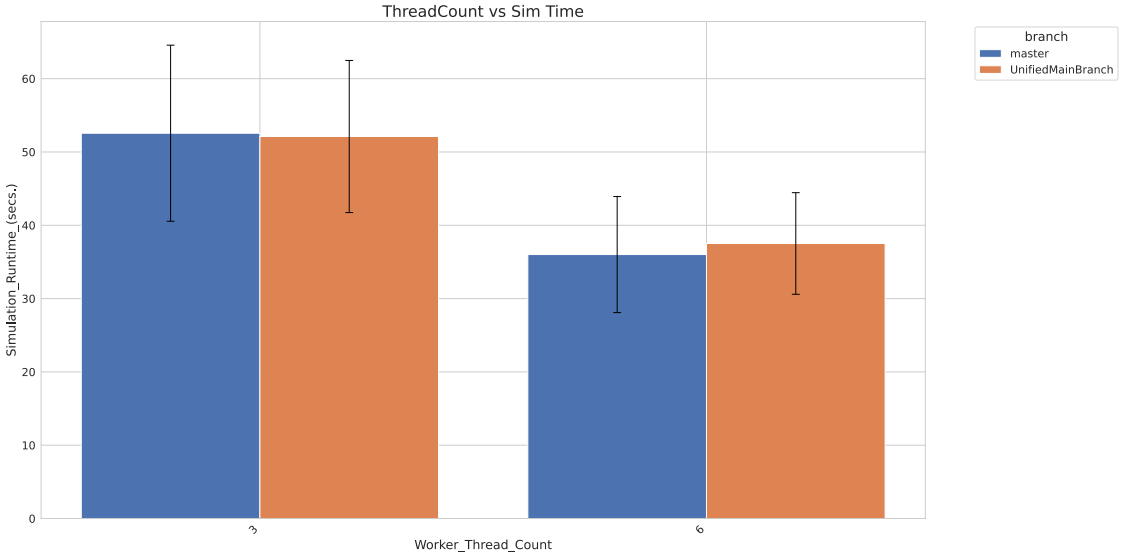

Figure 105: ThreadCount vs Sim Time

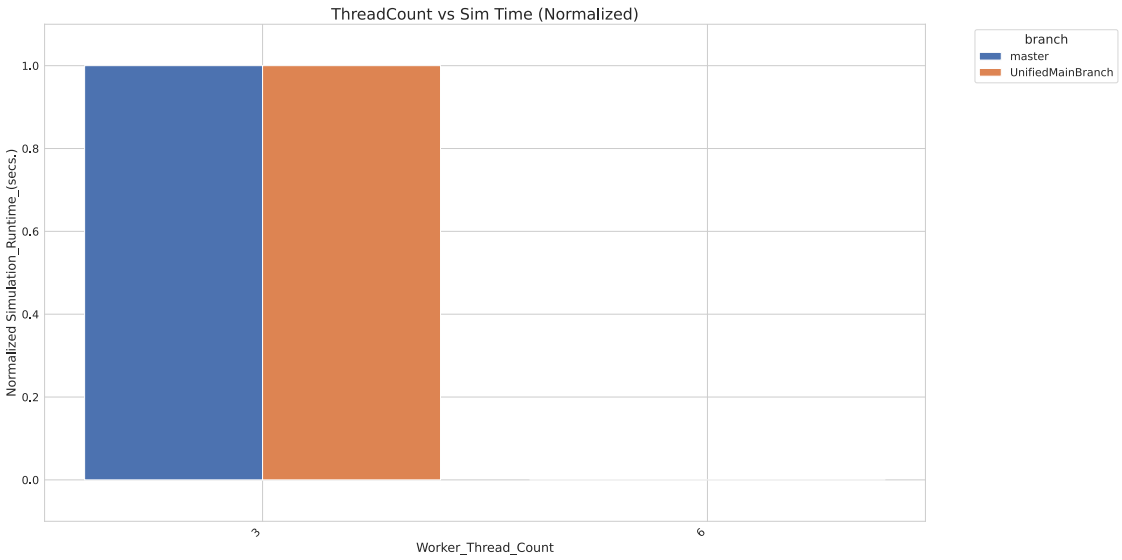

Figure 106: ThreadCount vs Sim Time_normalized

References

[1] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[2] Christopher L Barrett, Keith R Bisset, Stephen G Eubank, Xizhou Feng, and Madhav V Marathe. Episimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–12, 2008.

[3] Kalyan S Perumalla and Sudip K Seal. Discrete event modeling and massively parallel execution of epidemic outbreak phenomena. *Simulation*, 88(7):768–783, 2012.

[4] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.