Multi-tier Priority Queues & 2-tier Ladder Queue for Managing Pending Events in Sequential and Optimistic Parallel Simulations

Julius Higiro, Meseret Gebre, and Dhananjai (DJ) M. Rao

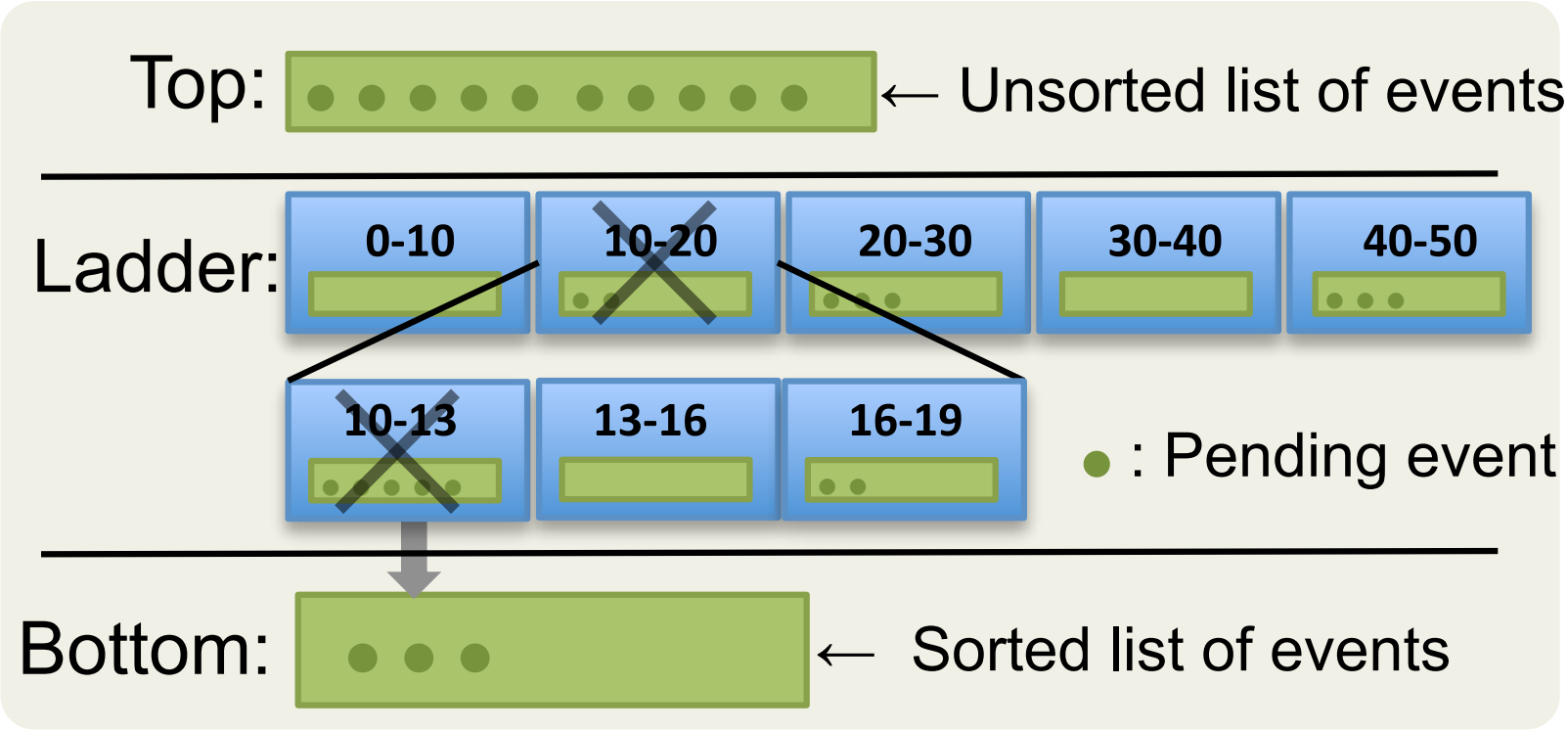
MIAMI UNIVERSITY
COLLEGE OF ENGINEERING AND COMPUTING
Computer Science and Software Engineering

Pending Event Set

- Events that are yet to be processed are called "pending events"
  - Pending events must be processed by LPs in "priority" or timestamp order to maintain causality
- data structures for managing and prioritizing pending events play a critical role in efficient simulation
  - Both sequential and parallel simulations
    - With thousands or millions of pending events
    - Fine grained simulations where the time taken to process an event is very short – i.e., LPs use only few 100s to 1000s of instructions per event.
    - Synchronization strategy used in PDES, Time Warp in particular, due to event cancelation operations

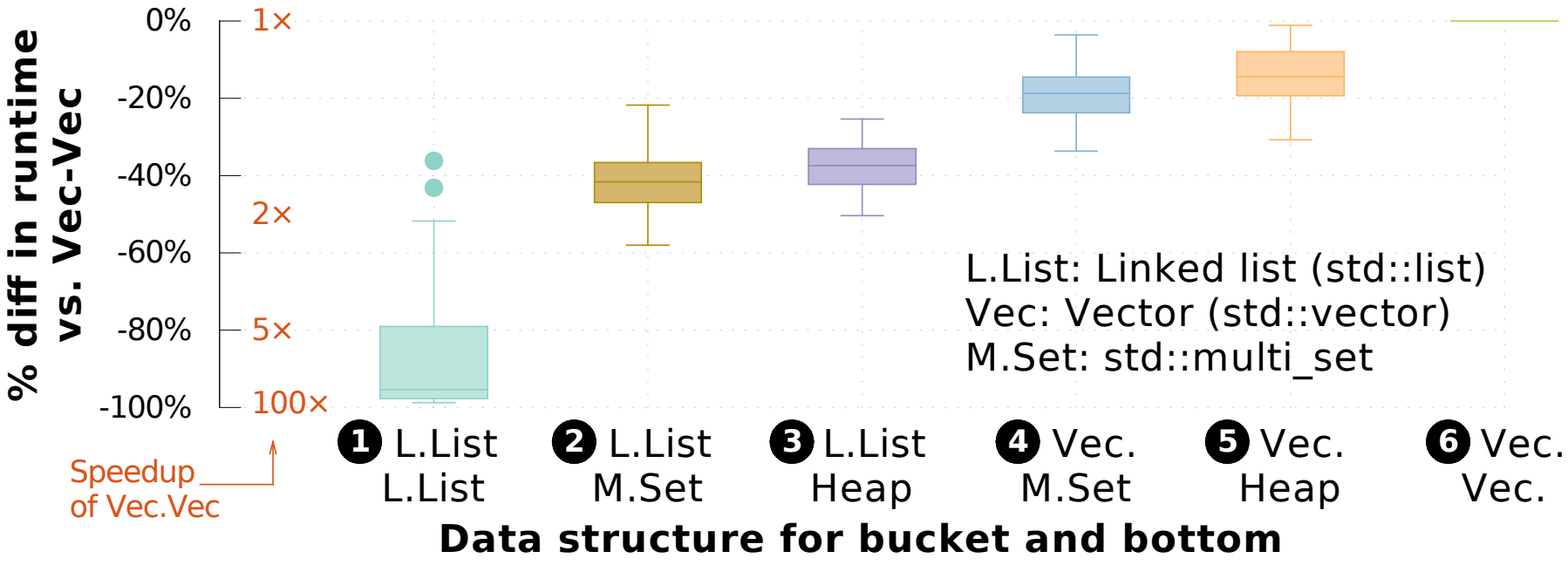
Ladder Queue Reference Data Structure

- Ladder Queue has shown to be a very effective data structure
  - Particularly for sequential simulations
  - Amortized O(1) for inserts & priority-order scheduling



Fine tuning Ladder Queue implementation

- Several data structures have been proposed for buckets & bottom
  - We found using `std::vector` to be most performant
  - Bottom uses quick sort (`std::sort`) with `std::vector`



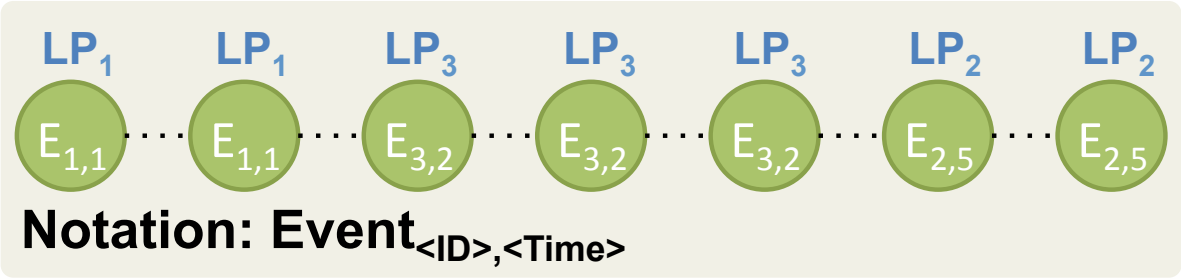
Multi-tier data structures

- Ladder Queue was substantially slower in 2 scenarios:
  - High concurrency: larger number of concurrent events (i.e., events with same timestamp) per LP
    - Long bottom increases insertion overheads
  - Time Warp synchronized parallel simulations
    - Canceling events requires scanning the whole structure
- Our multi-tier structures performed well in Time Warp PDES – i.e., simulations ran *much* faster
  - Multi-tier data structure: Separate LP scheduling vs. per-LP event management
    - Narrows events to scan for cancellation

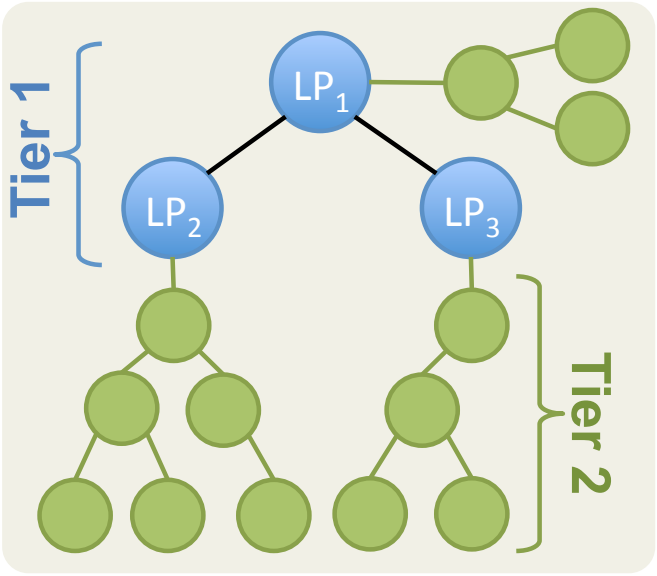
Single tier vs. multi-tier structures

Multi-tier data structure: Separate LP scheduling vs. per-LP event management

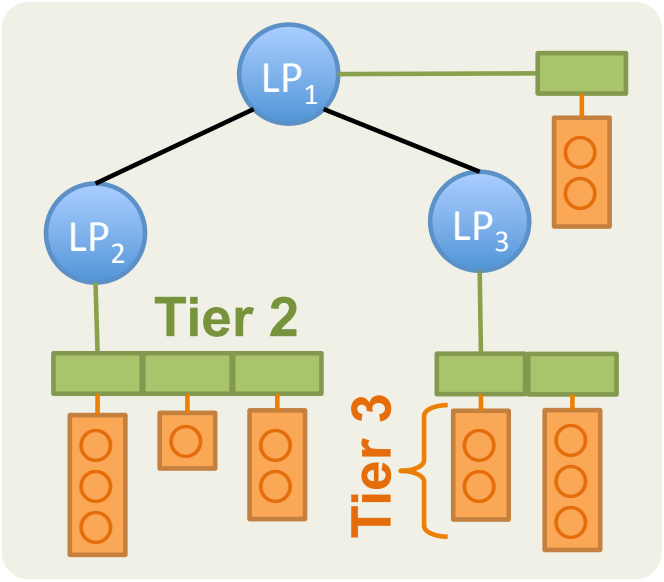
1-tier List of Events



2-tier Heap



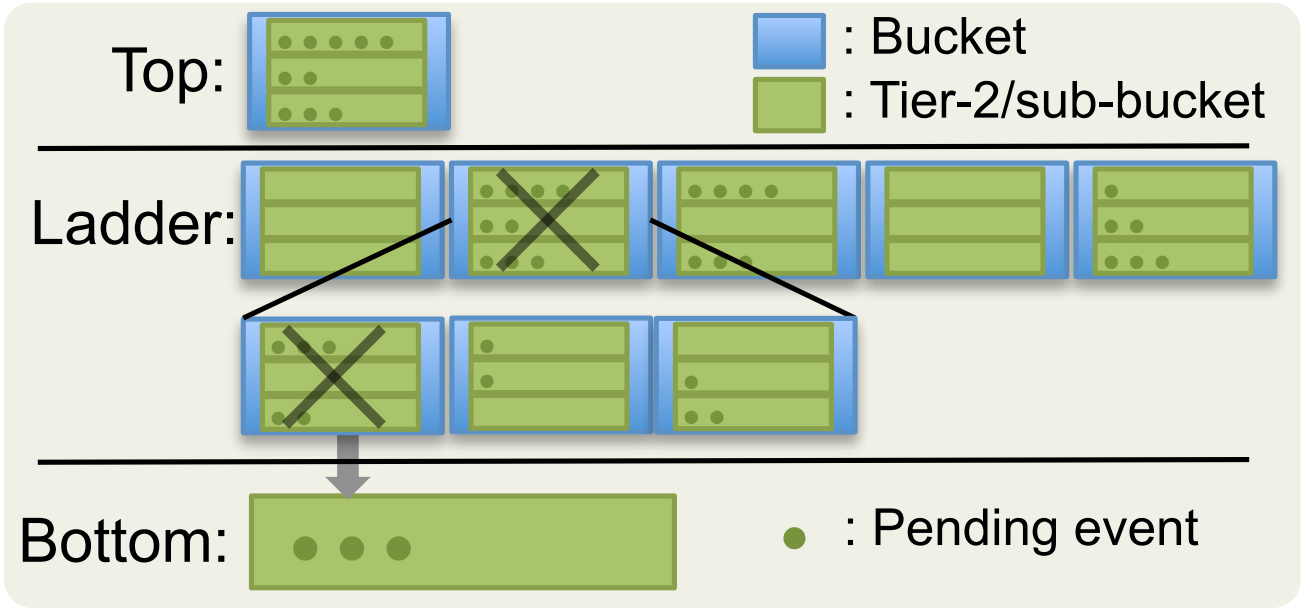
3-tier Heap



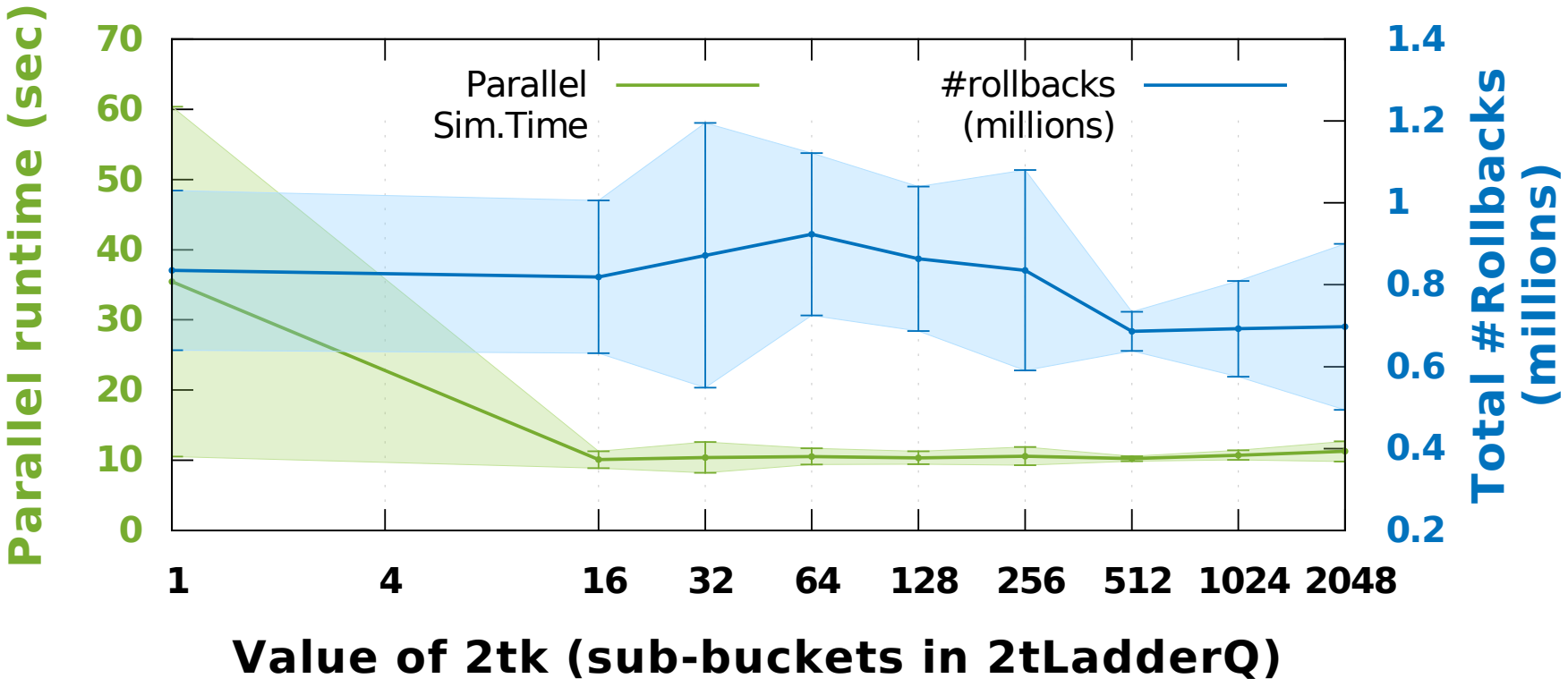
2-tier Ladder Queue (2tLadderQ)

- Aims to combine advantages of ❶ Ladder Queue and ❷ multi-tier structures
  - In `2tLadderQ` top & ladder are organized into "t2k" sub-buckets based on LP id
    - LP's ID is hashed into one of "t2k" buckets
  - Reduces number of events to scan during cancelation

If t2k=128, then number of events to scan is 1/128th



Impact of t2k on 2tLadderQ performance



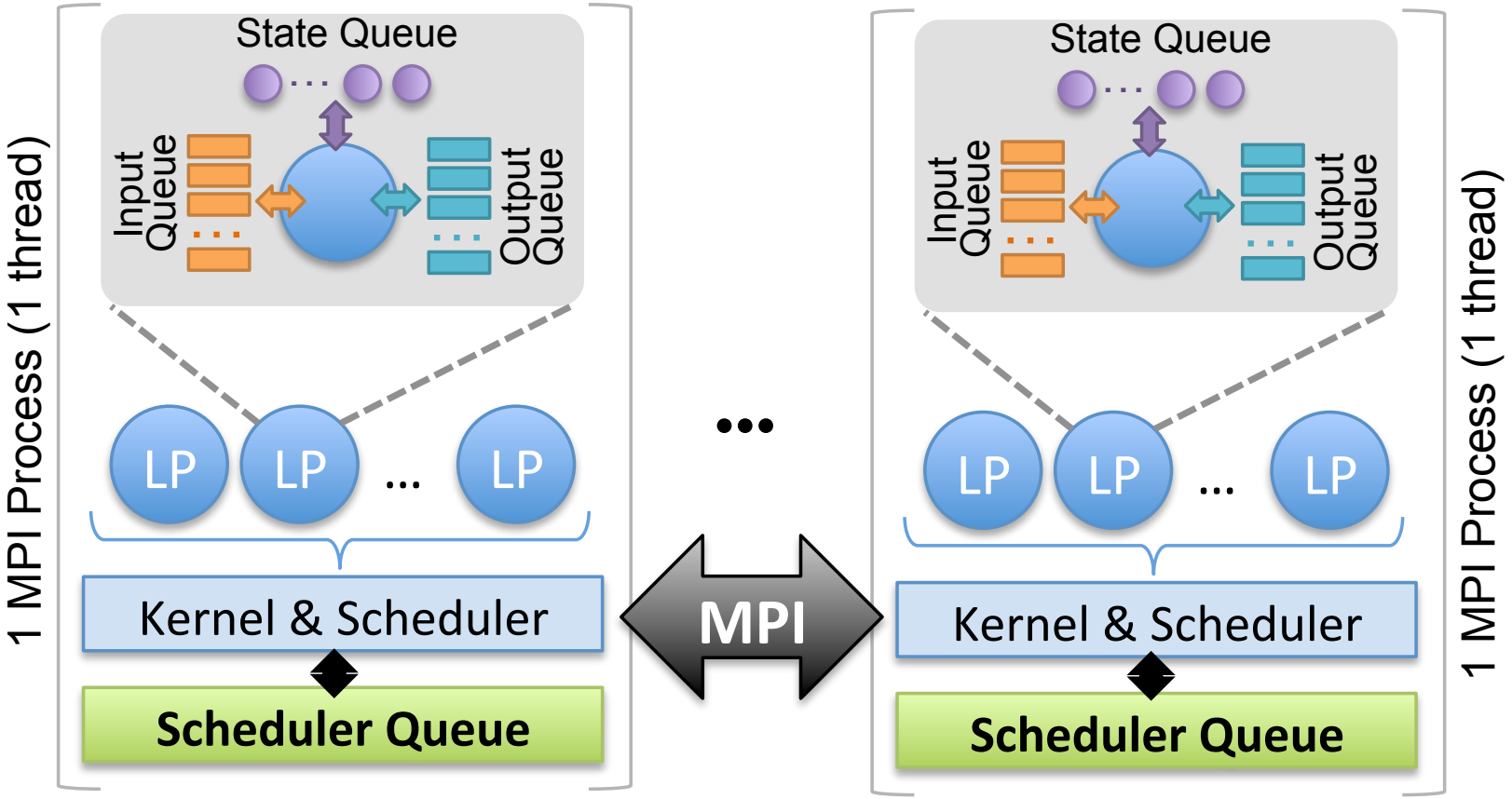
Adjectives of performance

Name	Enqueue	Dequeue	Cancel
heap	log(e•l)	log(e•l)	z•log(e•l)
2tHeap	log(e•l)	log(e•l)	z•log(e)+log(l)
fibHeap	log(e) + 1	log(e) + 1	z•log(e) + 1
3tHeap	log(e/c)+log(l)	log(l)	e+log(l)
ladderQ	1	1	e•l
2tLadderQ	1	1	e•l÷t2k

- l: #LPs
- e: #events / LP
- c: #concurrent events
- z: #canceled events

Overview of MUSE

- Miami University Simulation Environment (MUSE)
  - http://pc2lab.cec.miamiOH.edu/muse

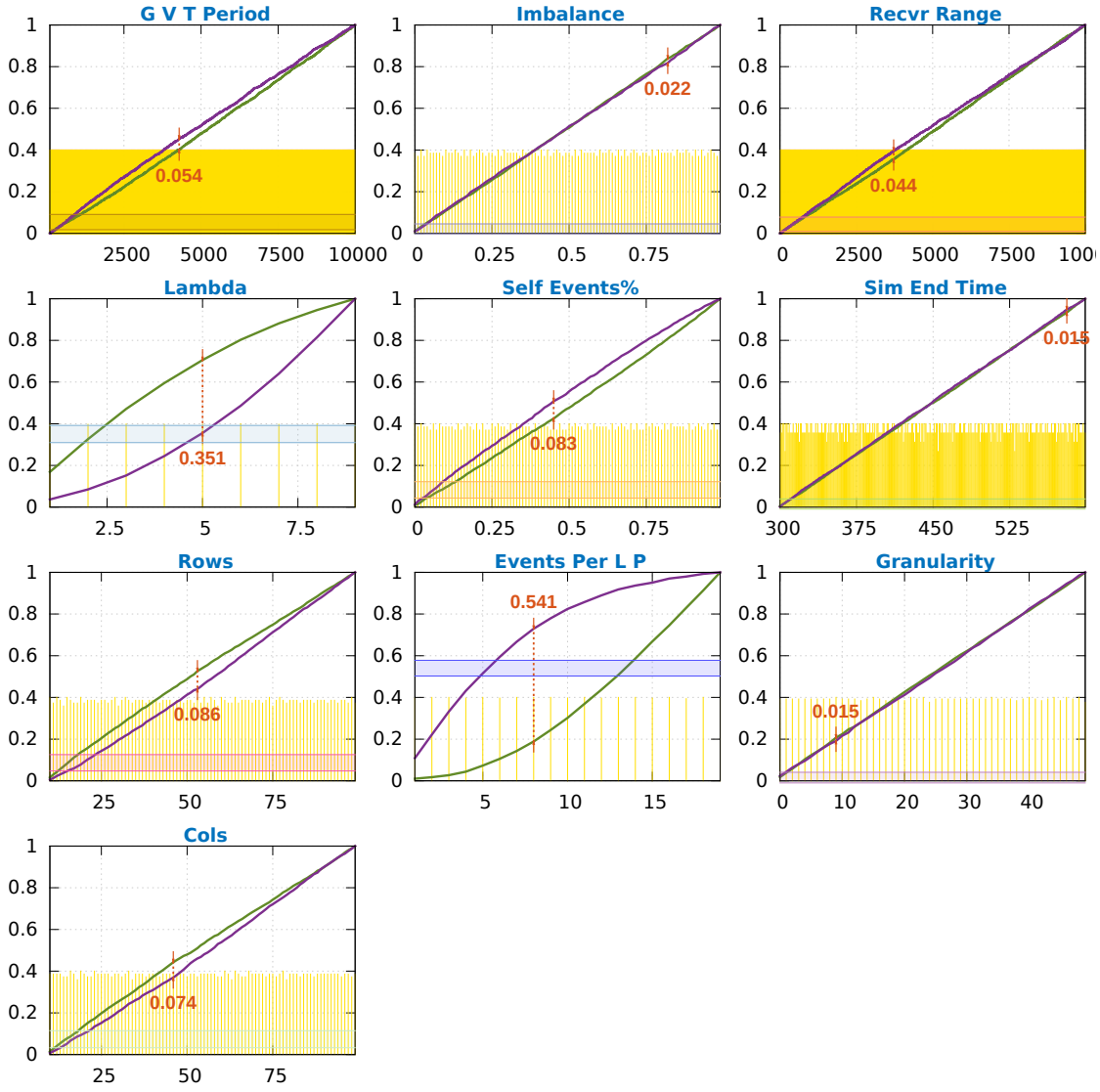


Benchmark: PHOLD Toroid grid

Parameter	Description
`rows`	Total number of rows in model
`cols`	Total number of columns in model
`eventsPerLP`	Initial number of events / LP
`delay (λ)`	Parameter for event receive times: LVT + $\lambda e^{-\lambda x}$
`%selfEvents`	Fraction of events sent to self
`granularity`	Additional compute load per event
`imbalance`	Fractional imbalance in partitioning
`simEndTime`	GVT when simulation logically ends
`recvrRange`	Destination LPs for scheduling events
`recvrDistrib`	Distribution: `uniform`, `exponential`, etc.

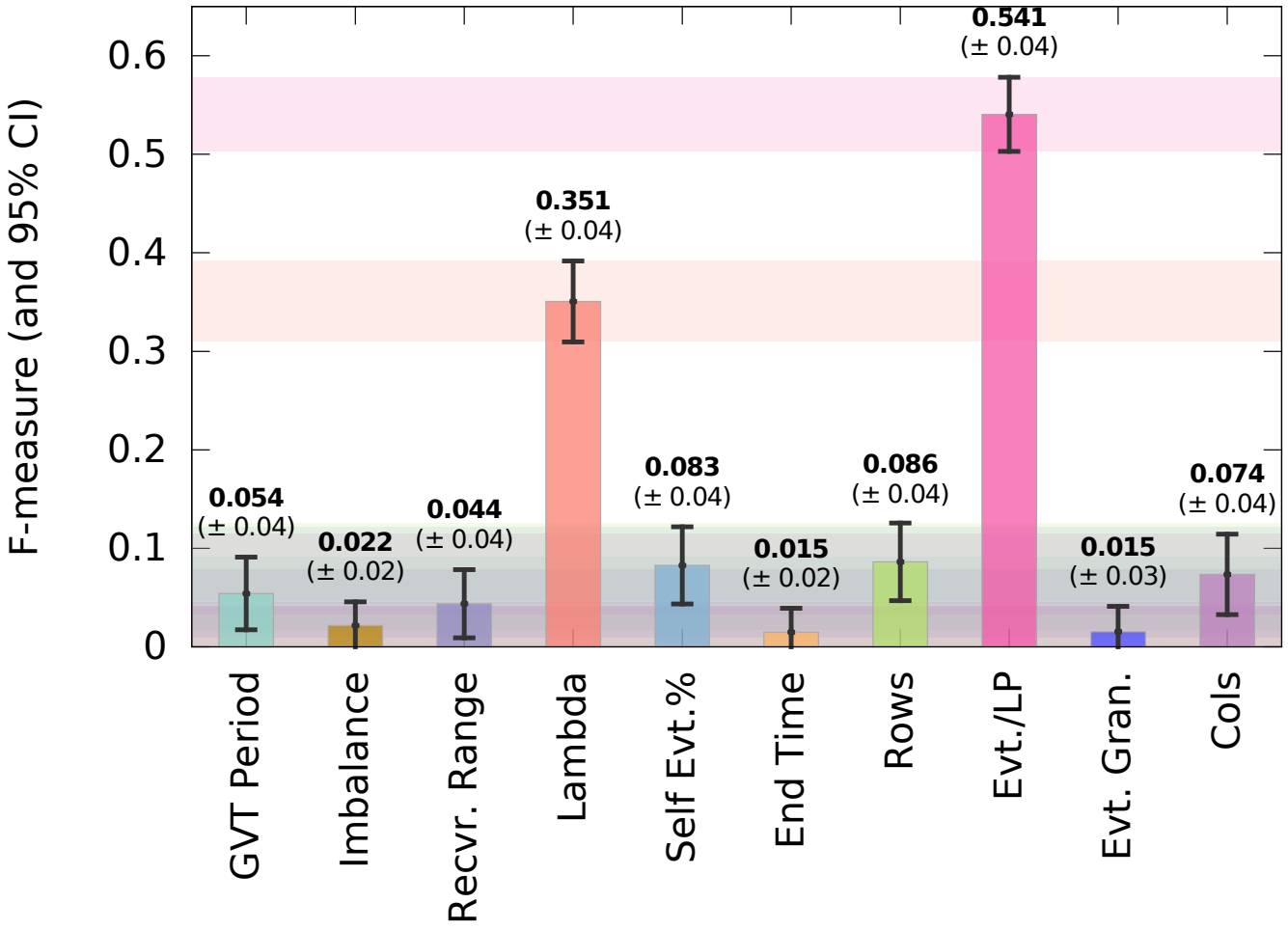
Identifying influential parameters Generalized Sensitivity Analysis

- GSA is based on two-sample Kolmogorov-Smirnov Test
  - Yields a dm,n statistic that is sensitive to differences in both central tendency and differences in the distribution functions of parameters.
  - The KS-Test is performed with data from Monte Carlo simulations involving combinations of parameter values.



GSA summary: Sequential Simulations 2tLadderQ vs. 3tHeap

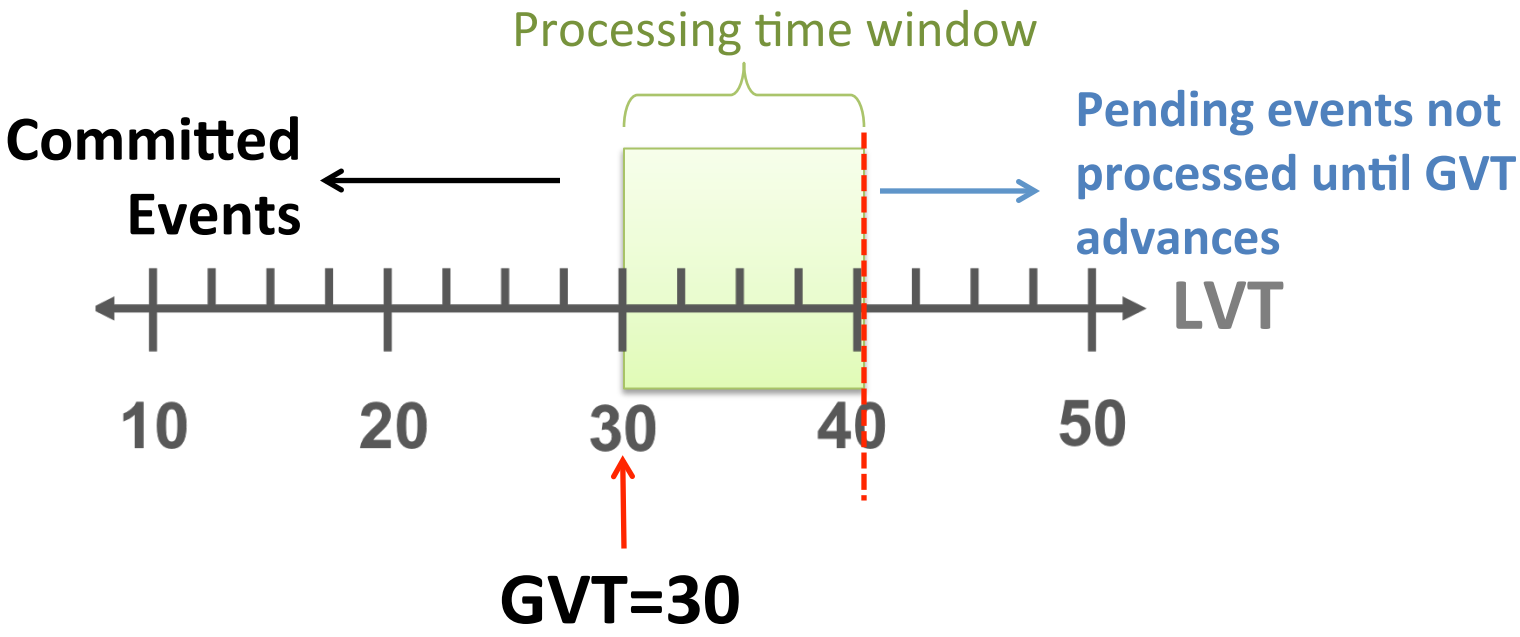
- Most influential parameters from GSA are:
  1. Events/LP
  2. Lambda



95% CI was computed using bootstrap approach using 5000 replications with 1000 samples in each.

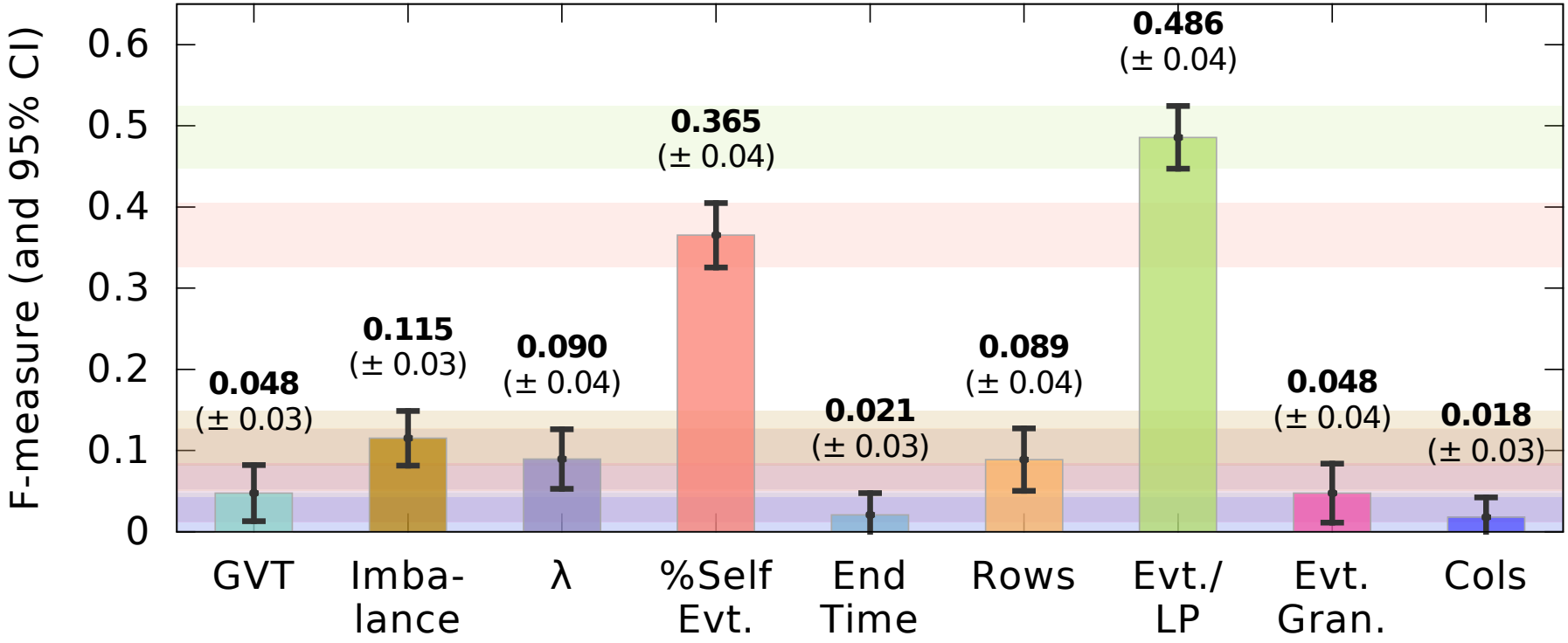
Limiting optimistic processing of events

- LadderQ experienced many cascading rollbacks
  - Possibly because rollback recovery is slow
  - We used a time window to limit optimistic advancement of LVT
    - Time window was 10 simulation time units



GSA summary: Parallel Simulations 2tLadderQ vs. 3tHeap

- Using 4 MPI processes
  - 3 independent samples per parameter combination
  - 95% CI computed using bootstrap

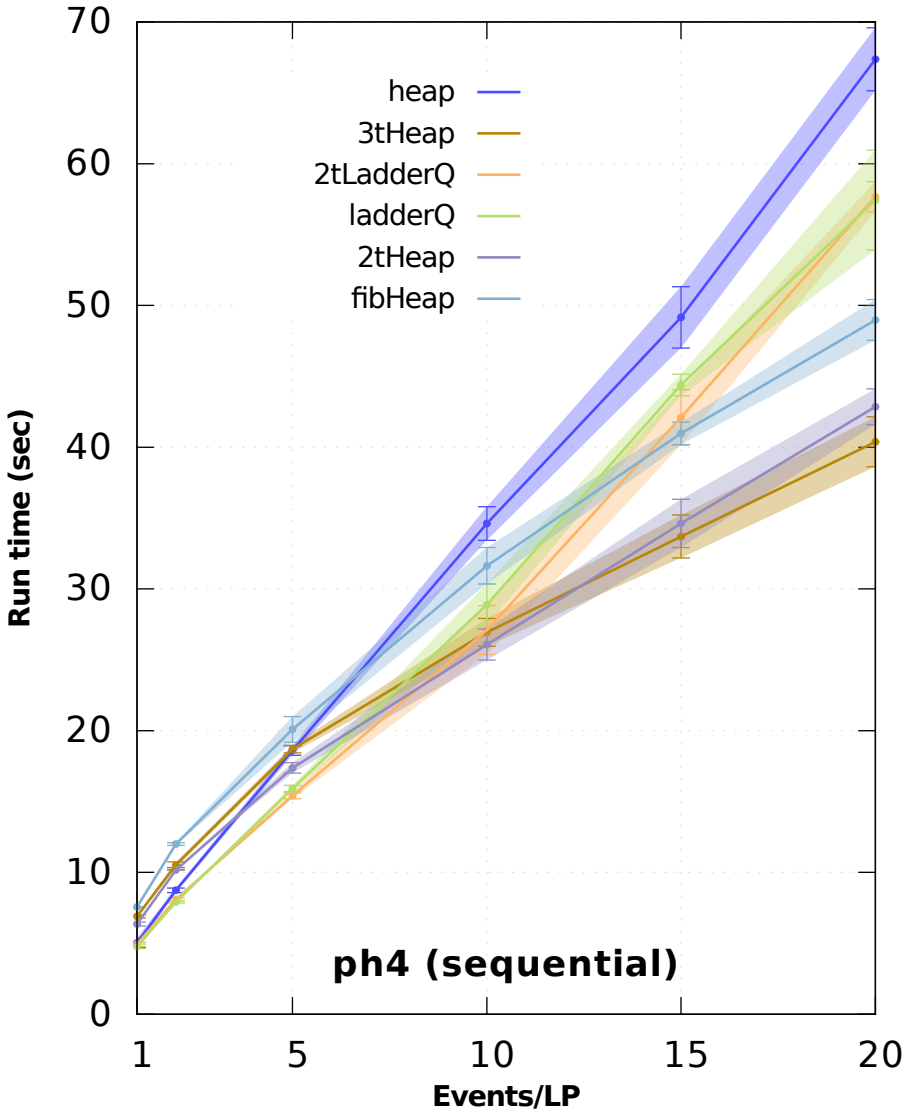


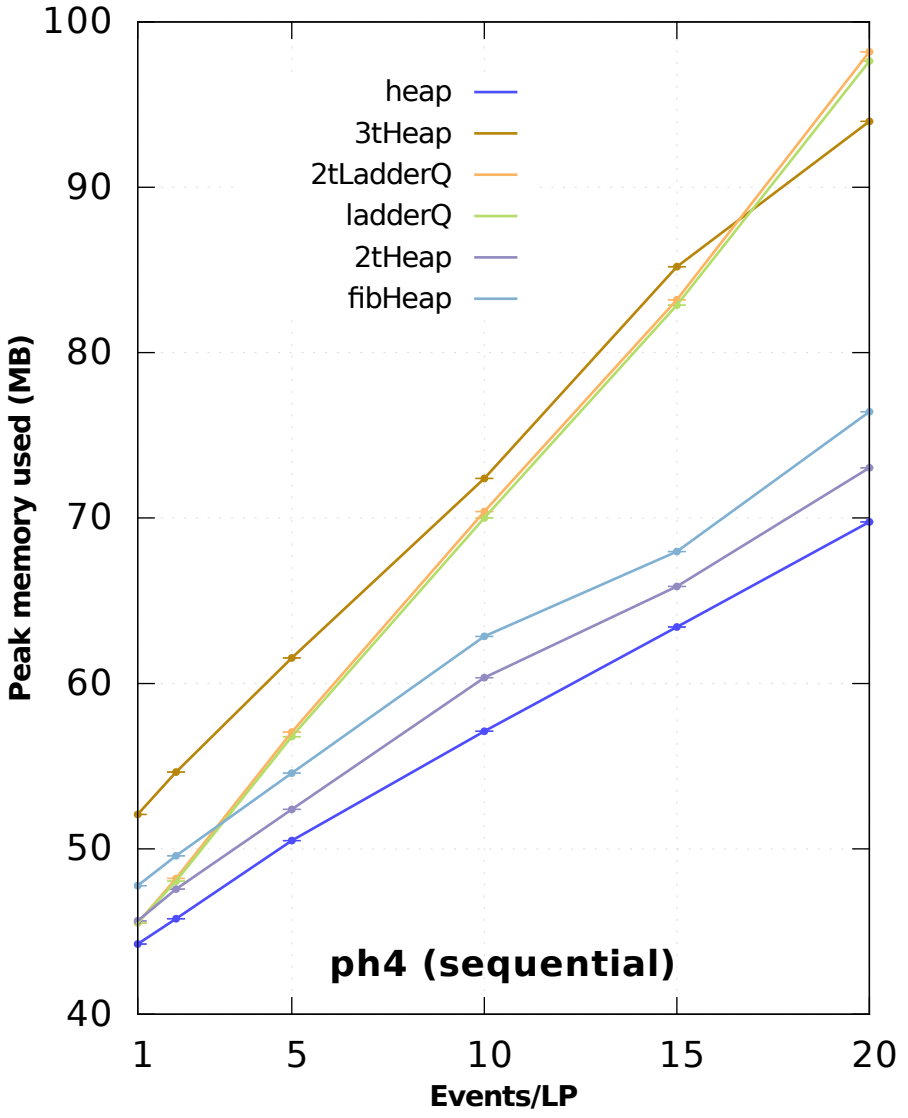
PHOLD Configurations used for experiments

Name	Rows × Cols	#LPs	Sim. End Time	
			Seq.	Parallel
ph3	100x10	1000	5000	2000
ph4	100x100	10,000	500	5000
ph5	1,000x100	100,000	100	1000

- Events/LP was varied: 1, 2, 5, 10, 20
- Lambda was varied: 1 (wider range), 10 (narrow range)
- Number of MPI processes was varied for parallel simulation

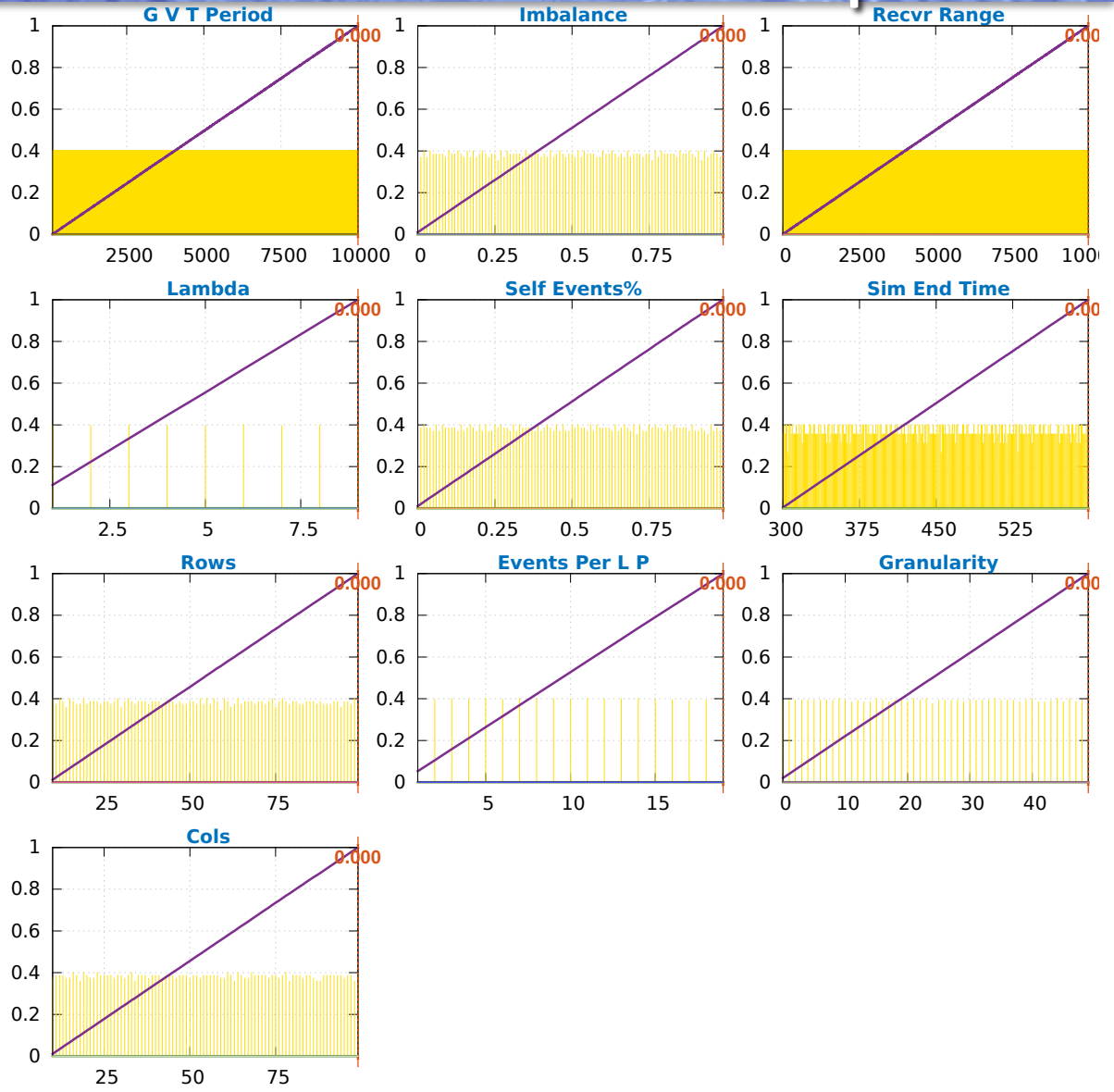
ph4: Sequential Timings λ=1 (best case for ladderQ)





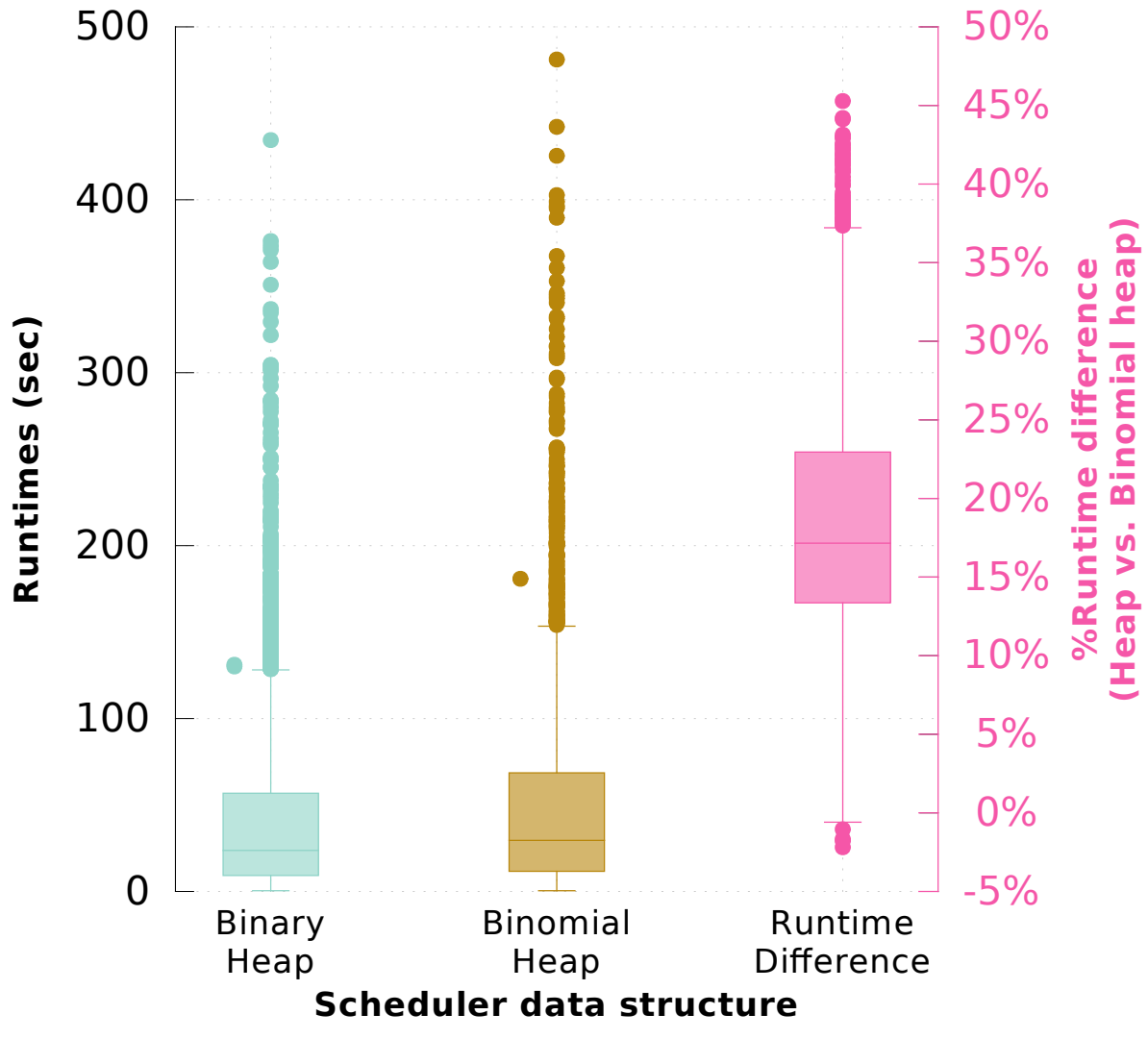
Binary Heap vs. Binomial Heap for scheduler queue

GSA analysis using sequential simulations shows that none of the 10 model parameter settings influence the performance difference between binary & binomial heaps

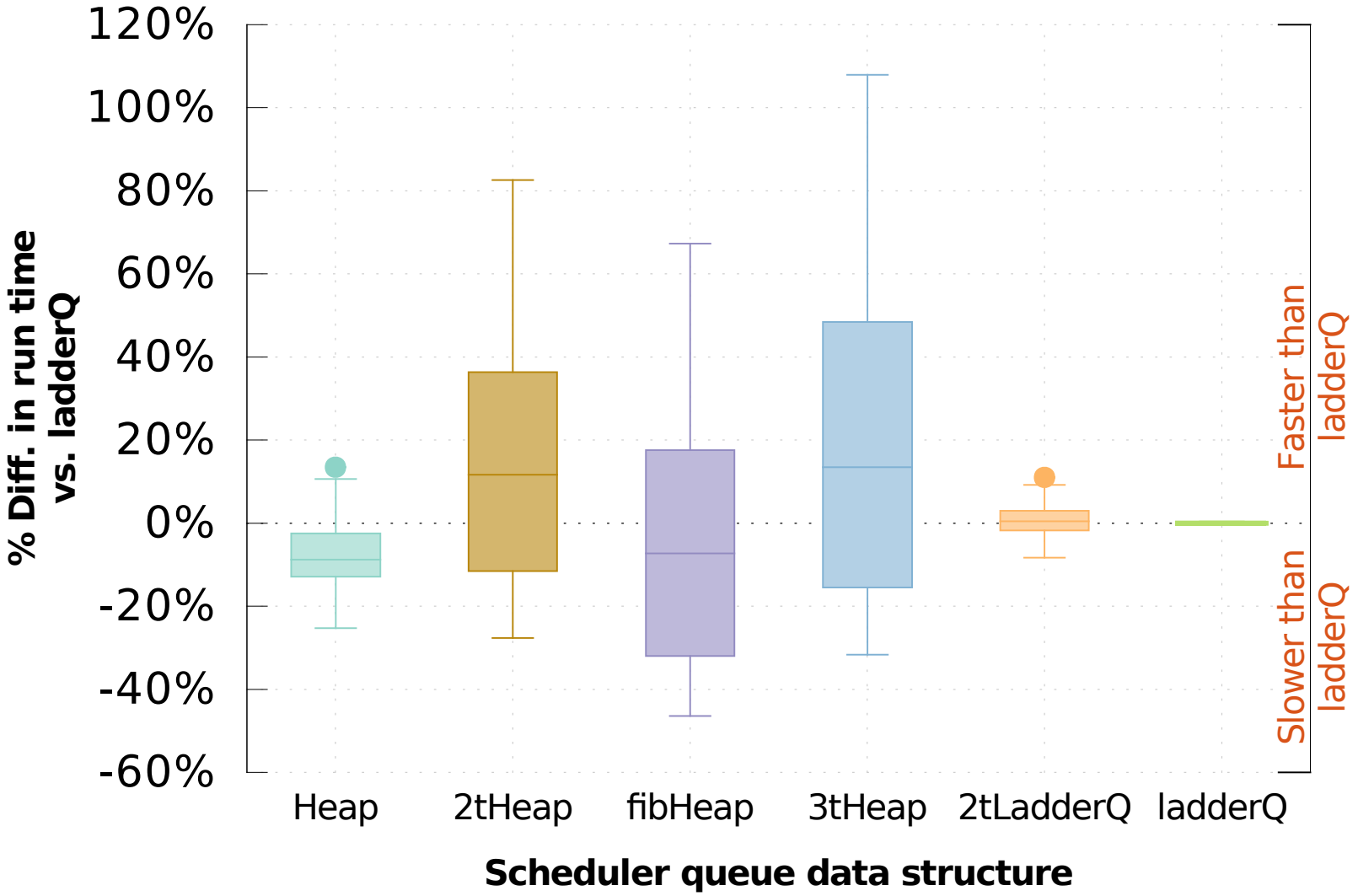


Binary Heap vs. Binomial Heap

In most cases the binary heap was faster than binomial heap

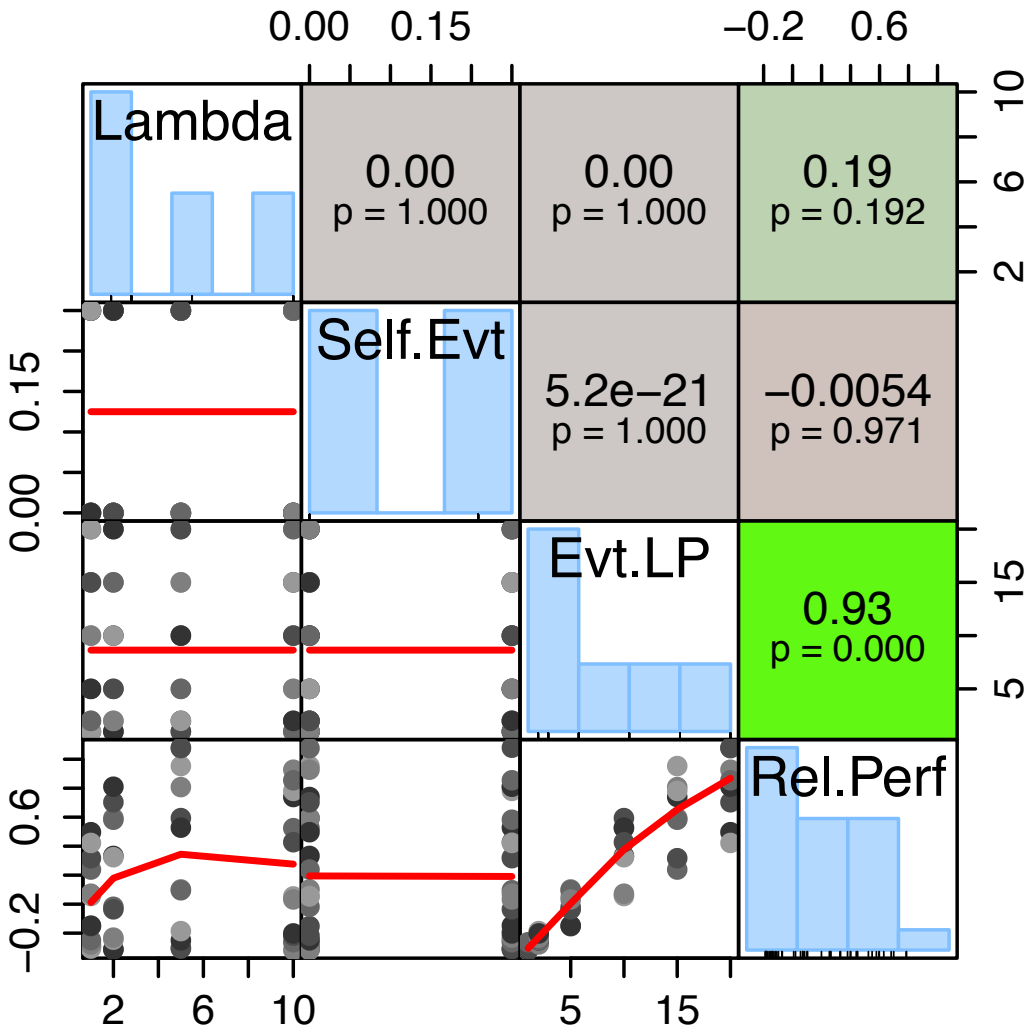


Sequential timings summary
1500 runs of PH3, PH4, & PH5

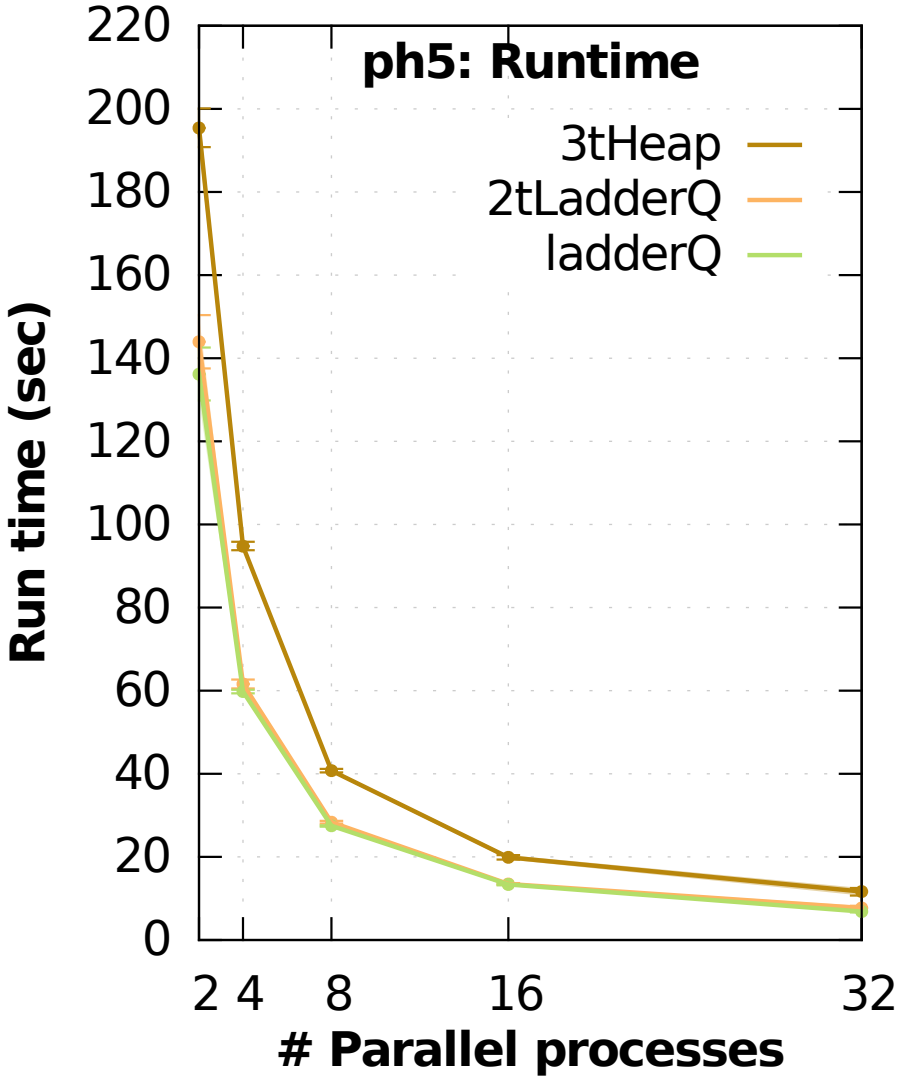


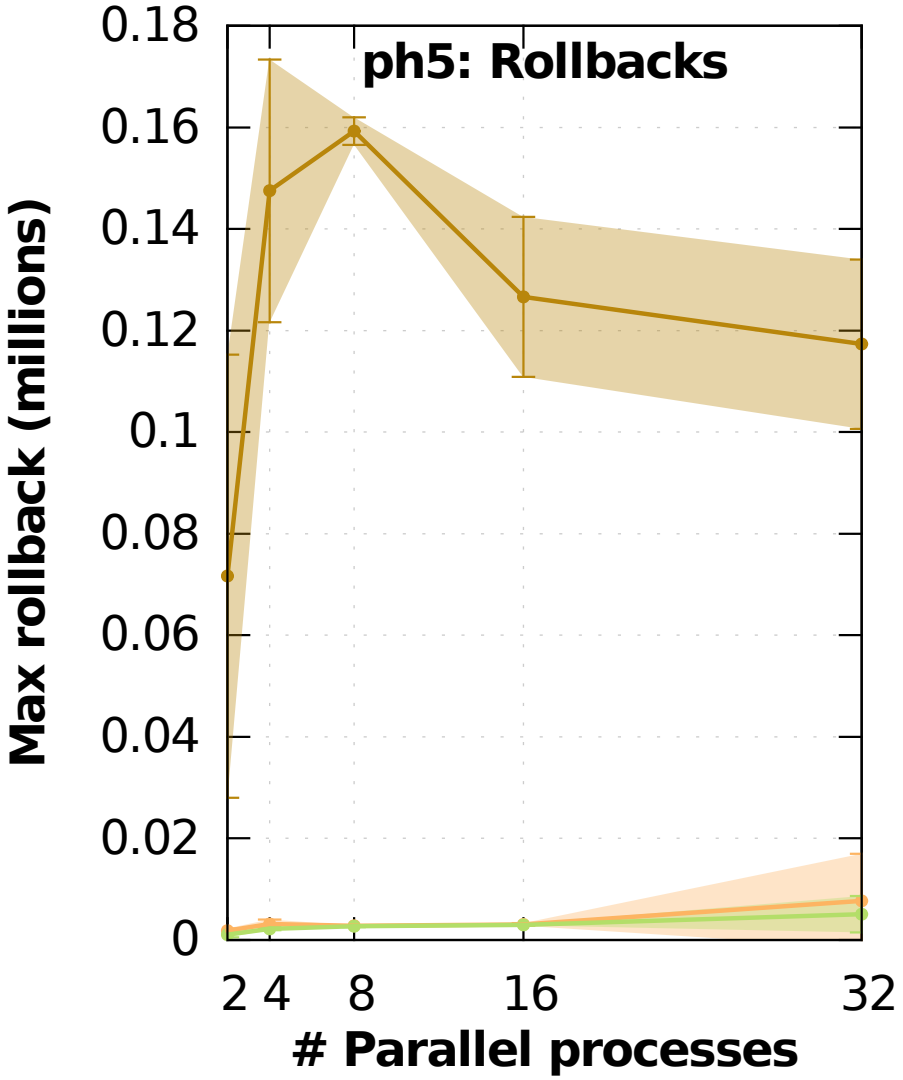
Interplay between parameters

- Corellogram to illustrate potential interplay between PHOLD model parameters
  - 3tHeap vs. ladderQ
- Confirms results from GSA

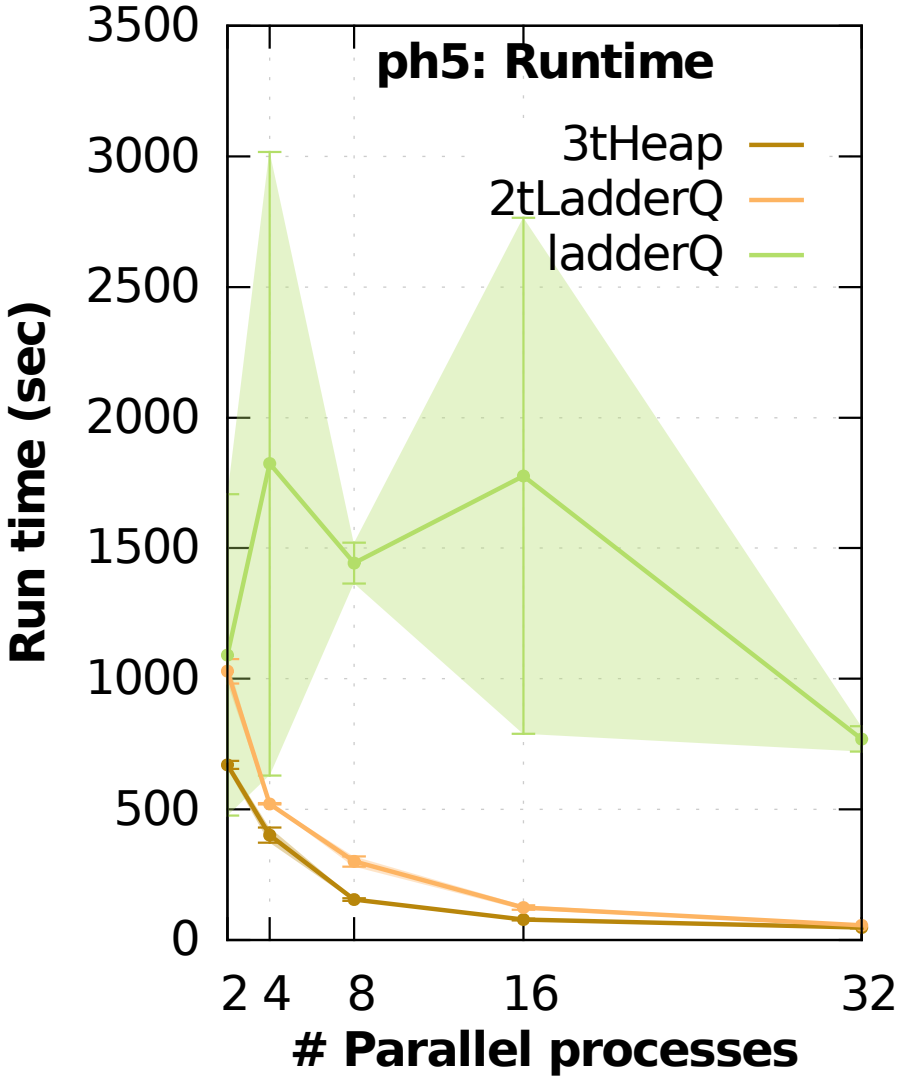


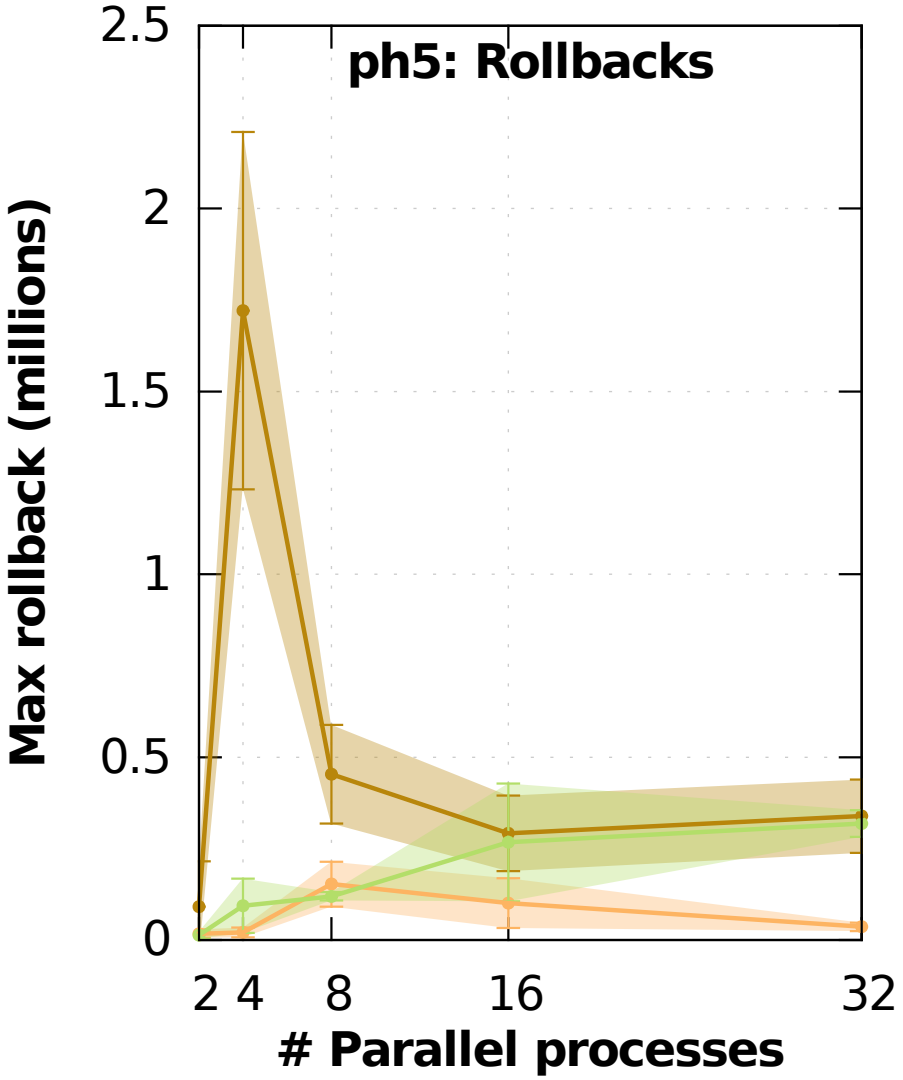
Low concurrency Parallel Simulation
λ=1, Events/LP = 2, Time window = 10



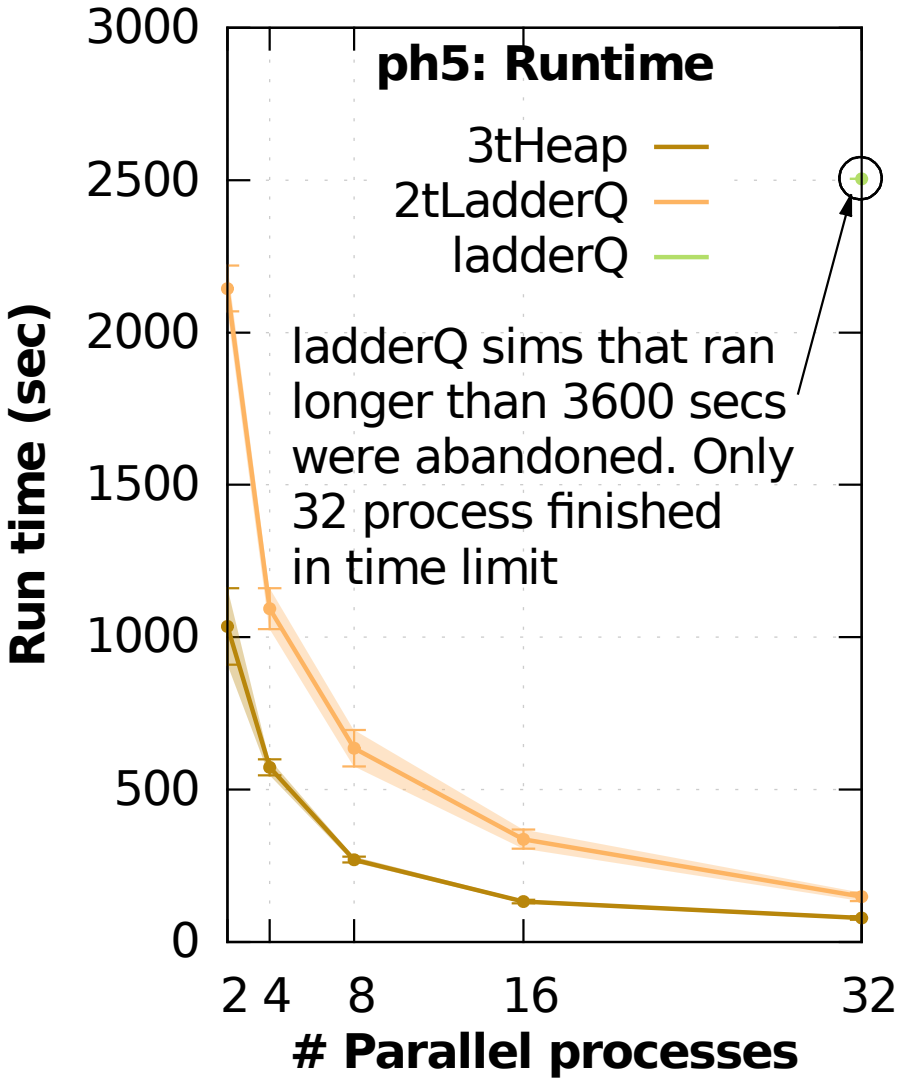


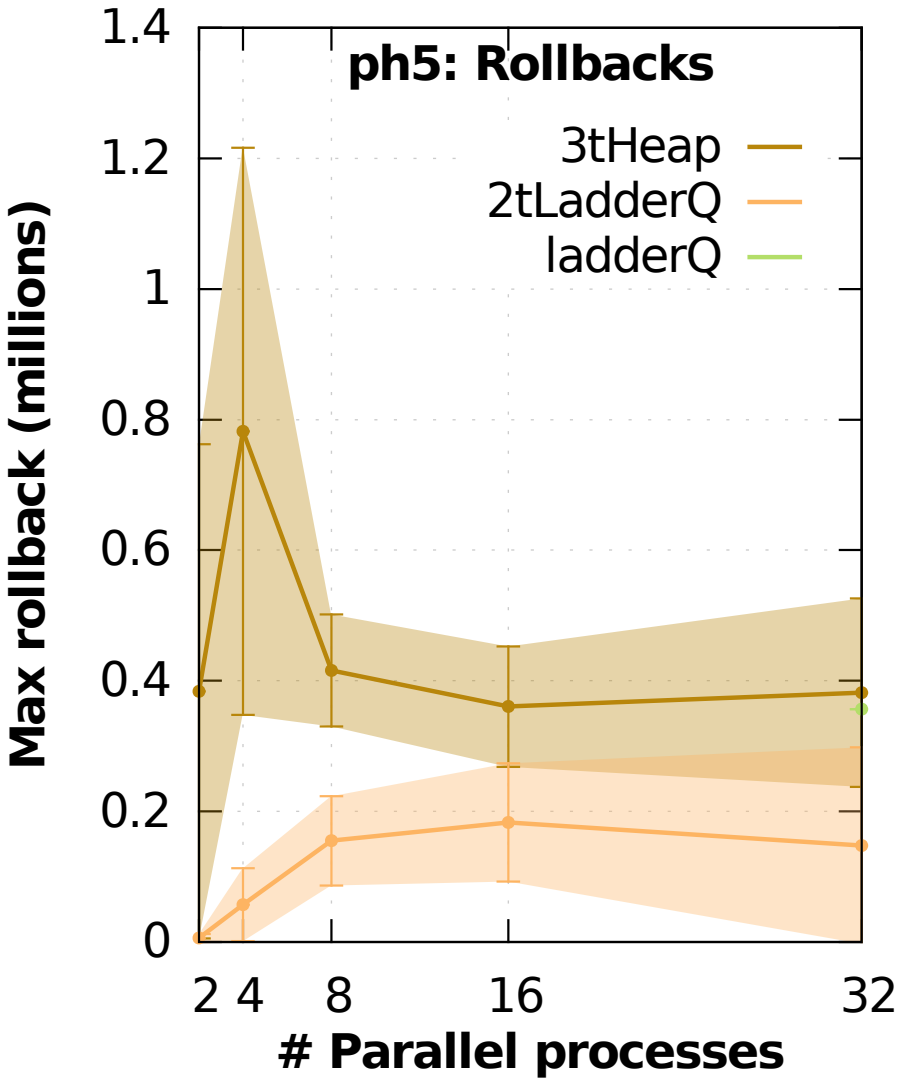
Medium concurrency parallel Simulation
λ=10, Events/LP = 10, Time window = 10



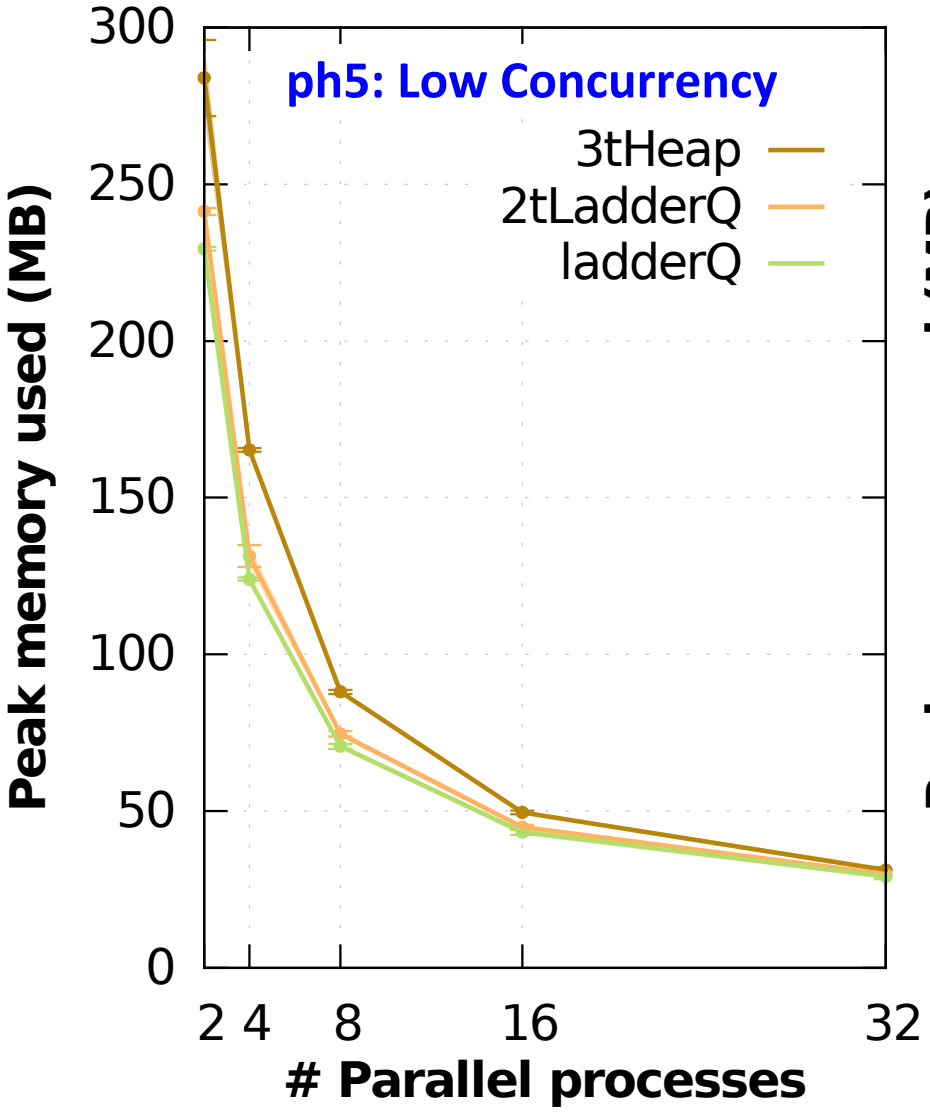


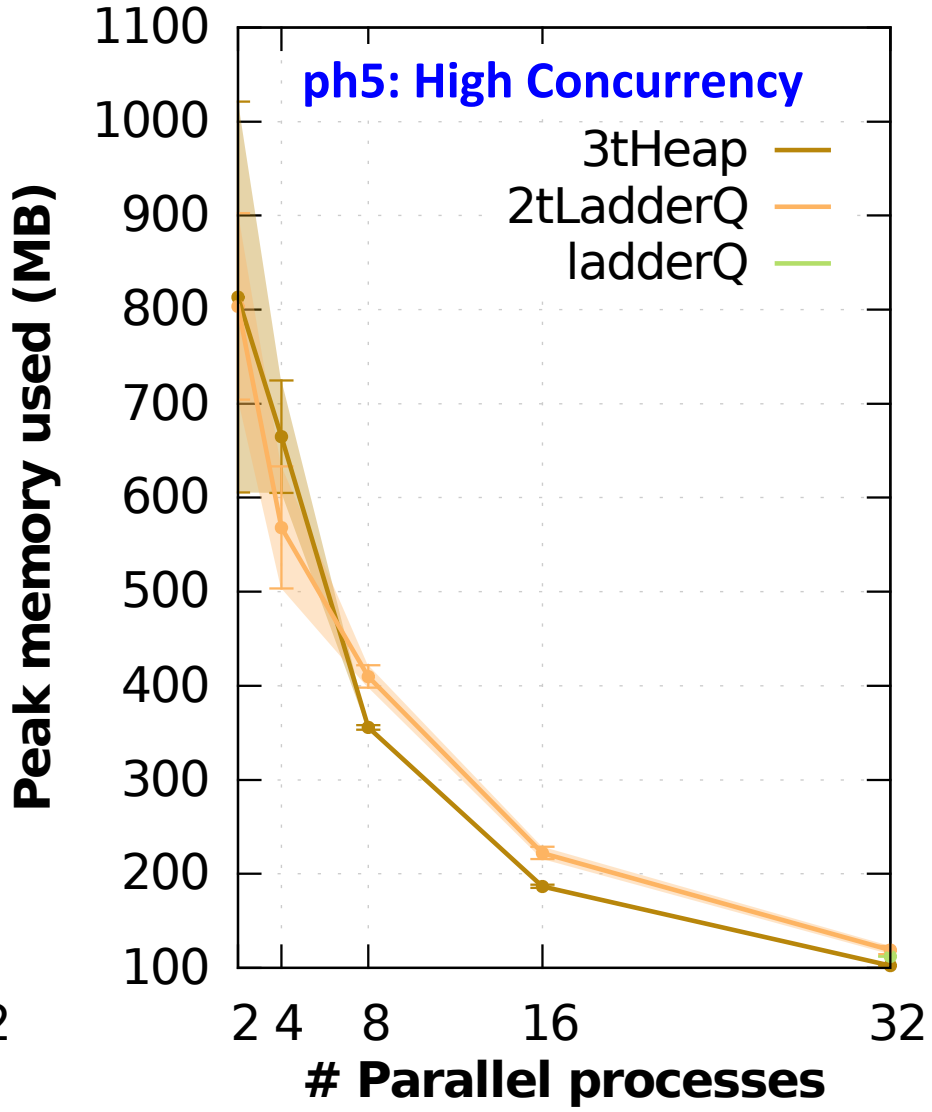
High concurrency Parallel Simulation
λ=10, Events/LP=20, Time window=10



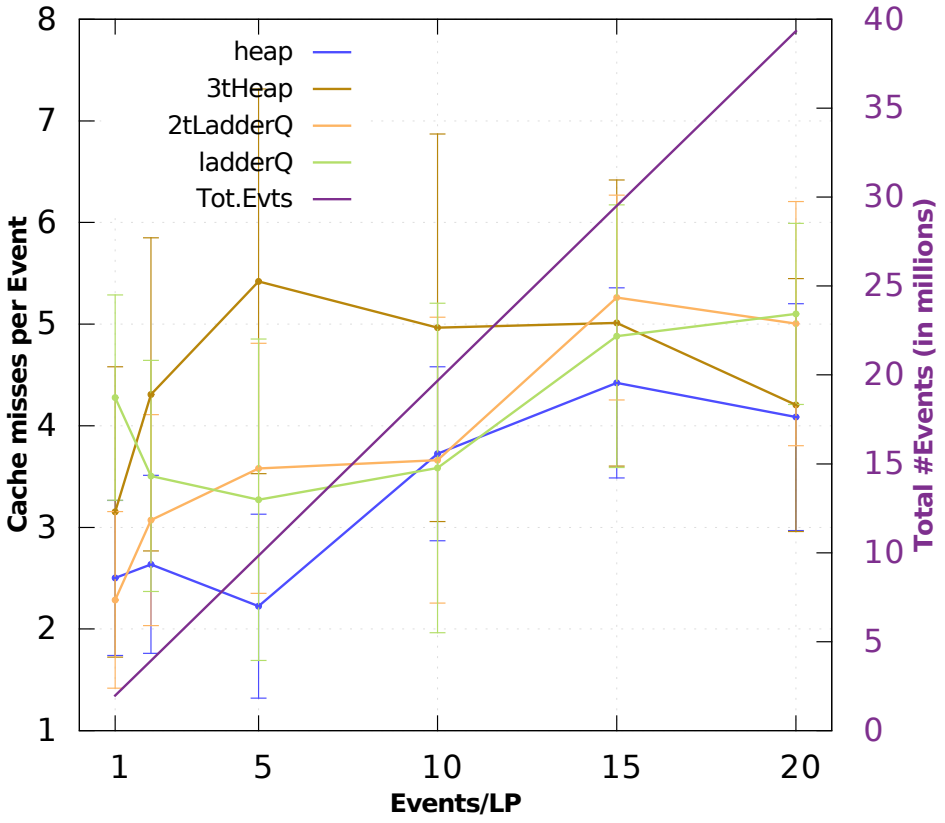


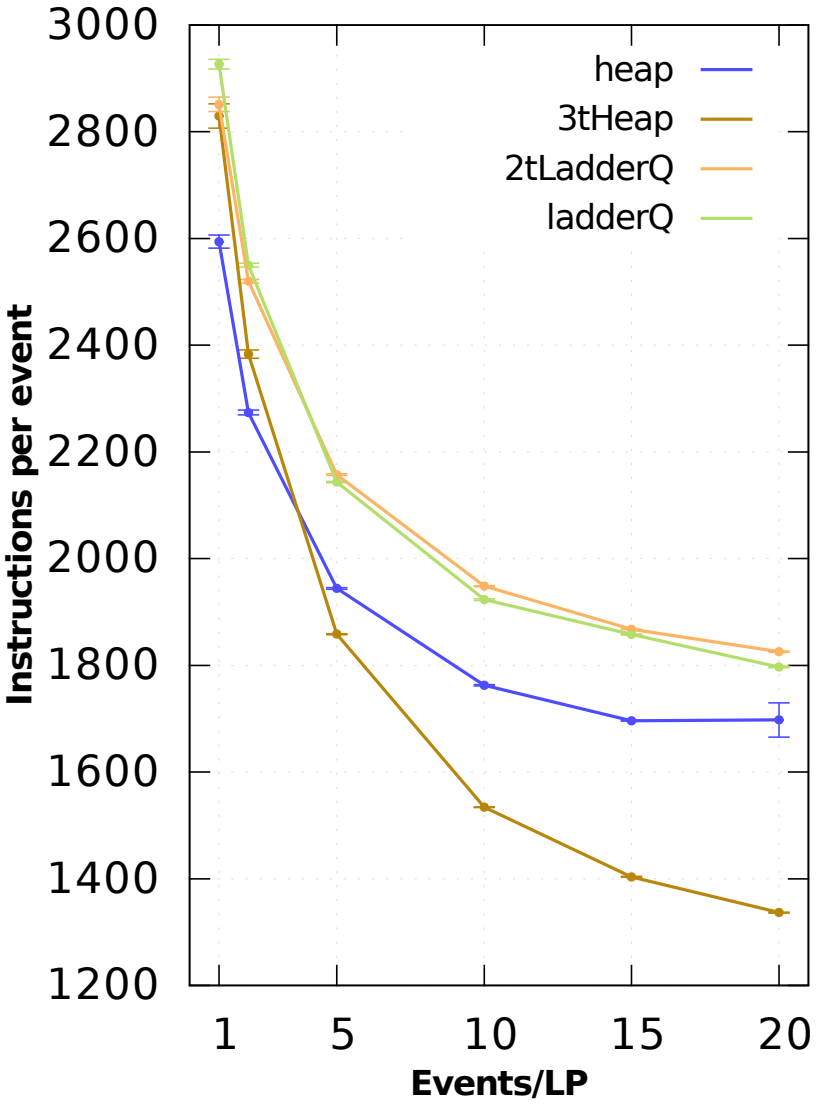
Memory comparison





Cache usage comparisons Sequential simulations





ph3 Configuration (1,000 agents)
λ = 1 (wide range of receive time)
Events/Agent = 20

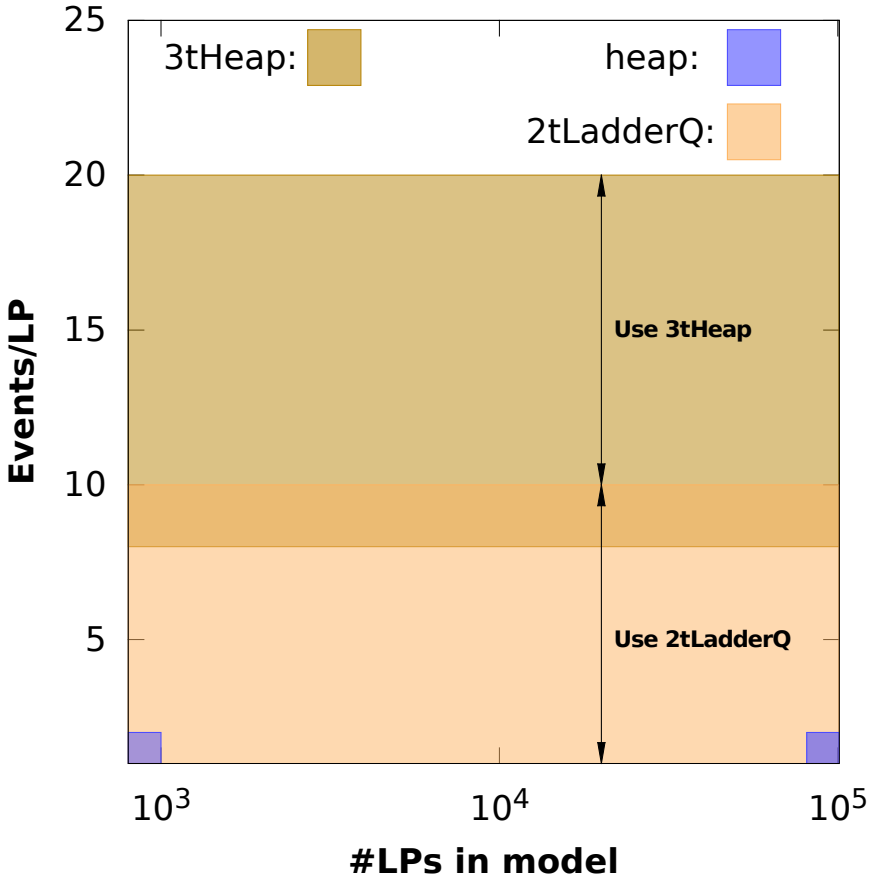
Conclusions

- Generalized Sensitivity Analysis (GSA) seems like a good strategy to narrow down solution space
  - Concurrent events/LP was the most dominant factor
    - Distribution & events/LP are parameters that influence it
  - Maybe use GSA to characterize/standardize benchmarks?
  - Care needs to be taken to cover parameter space
- LadderQ fine-tuning experience suggests that
    - Reduction in runtime constants was primarily realized by minimizing memory management overheads

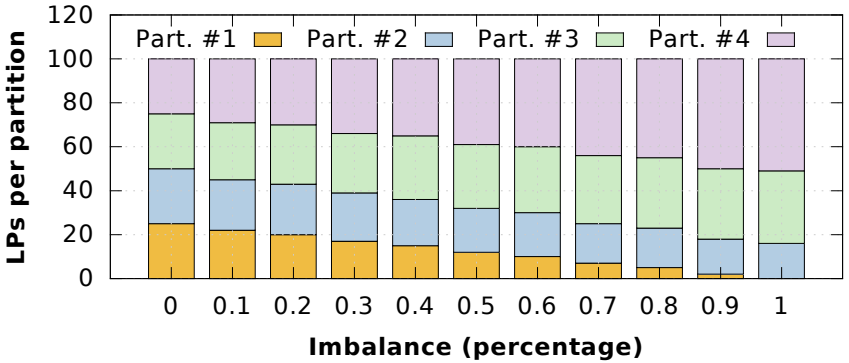
1. Favor few bulk operations via `std::vector` over linked-lists
2. Recycle memory / substructures rather than reallocating them.

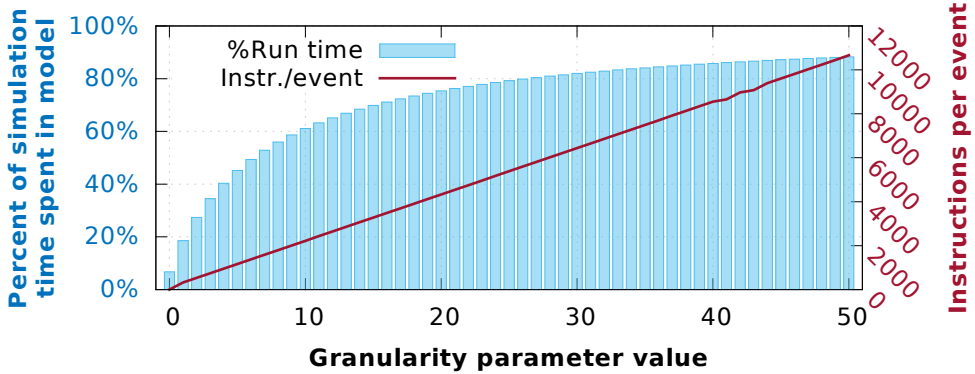
Conclusions (Contd.)

- Runtime constants may play a more dominant role than asymptotic time complexity
  - E.g. 3tHeap with log(n) time complexity outperformed Fibonacci Heap with O(1) time complexity
- 2tLadderQ performs better than ladderQ in parallel simulations
  - And in sequential simulation configurations with high concurrency
- 3tLadderQ outperforms 2tLadderQ in high concurrency scenarios



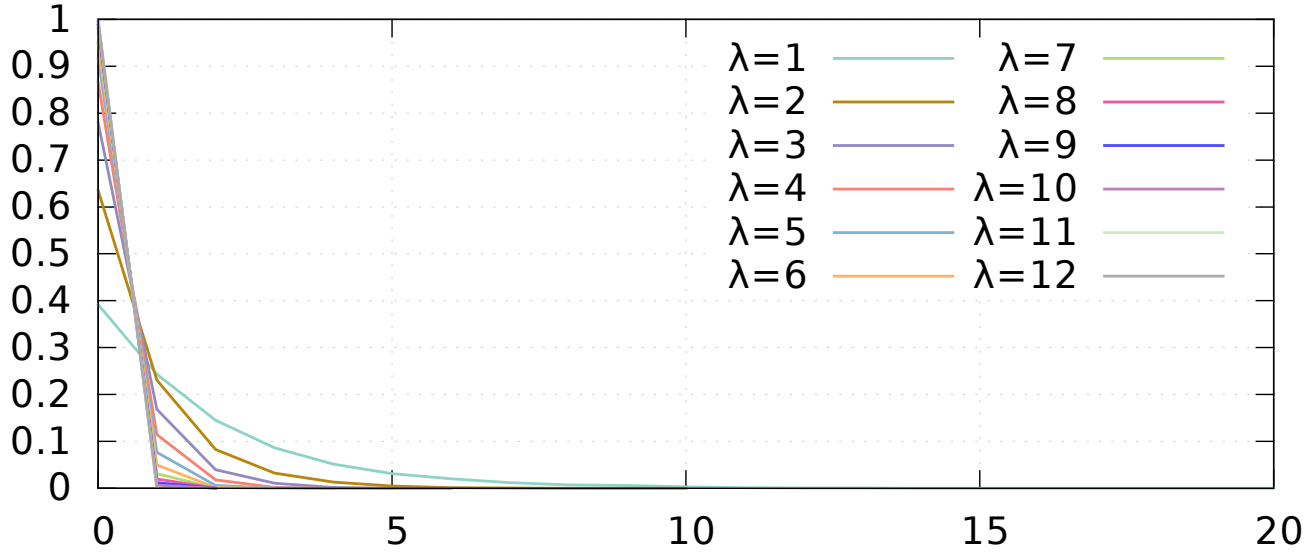
Impact of some PHOLD parameters





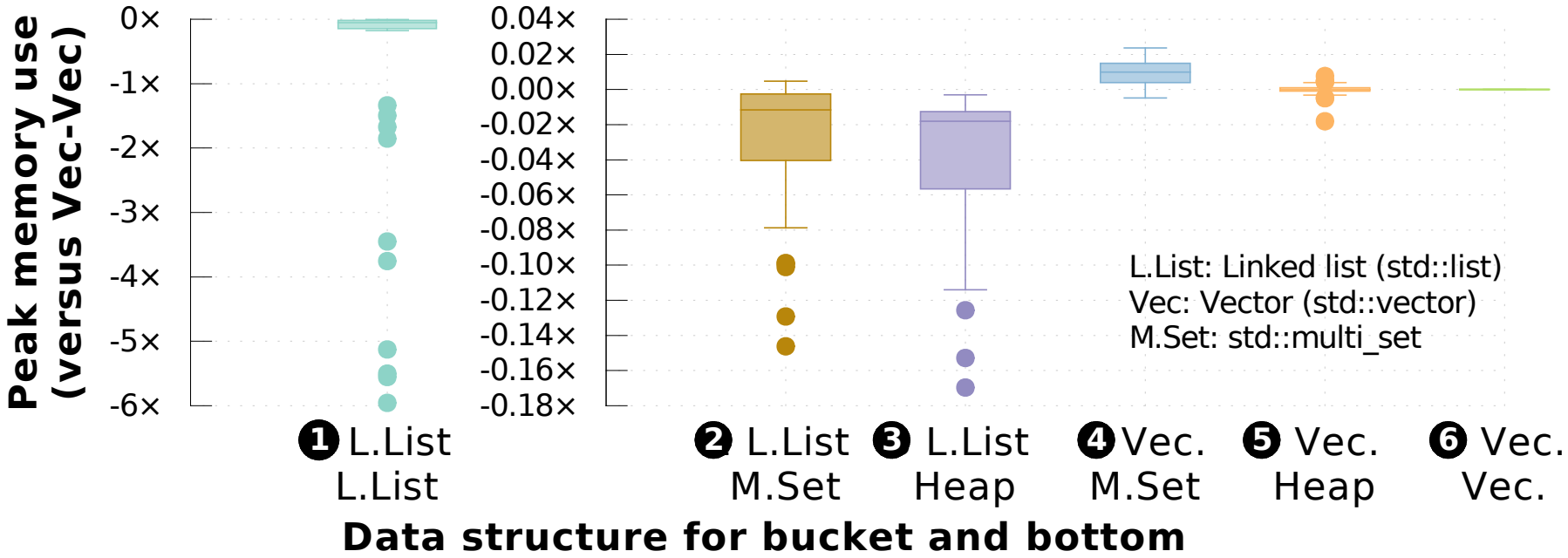
Impact of Lambda on receive time of events

Receive times: LVT + $\lambda e^{-\lambda x}$



Fine tuning Ladder Queue implementation: Memory Comparison

- The Vec-Vec solution consumes more memory
  - We found using `std::vector` to be most performant
  - Bottom uses quick sort (`std::sort`) with `std::vector`



Ladder Queue: Number of rungs in the ladder