Supplementary Materials (PADS 2017)

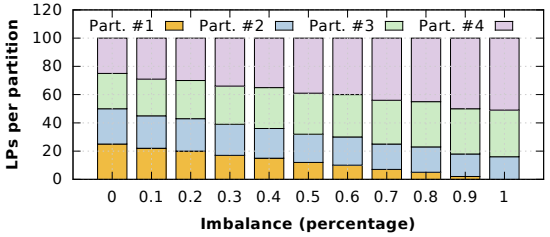
Multi-tier Priority Queues & 2-tier Ladder Queue for Managing Pending Events in Sequential & Optimistic Parallel Simulations

Julius Higiro, Meseret Gebre, and Dhananjai M. Rao

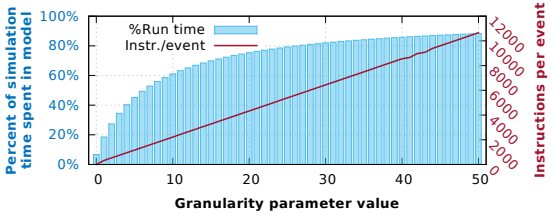
Table of Contents

Reducing the parameter space for analysis

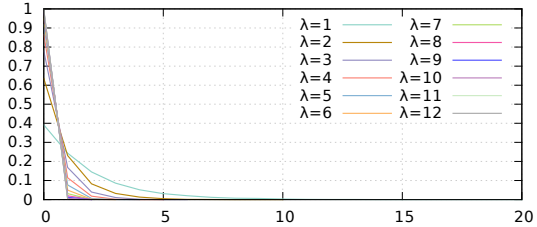
In this study we have used the standard PHOLD benchmark (also used by other investigators) to assess the comparative performance of the 6 different scheduler queues used in this study. However, that required exploring 9 different parameters with a broad range of parameter settings. The large parameter space makes it impractical for experimental analysis of 6 different queues. Consequently, we have used Generalized Sensitivity Analysis (GSA) to reduce the parameter space for analysis. It is important to ensure that GSA covers the necessary range of parameters pertinent to this study. So first we have characterized the behavior of key parameters in PHOLD. The impact of 3 key parameters in PHOLD is shown in the figures below (same figures from PADS paper):



(a) impact of `imbalance`



(b) impact of `granularity`



(c) impact of λ (`delay`)

For other parameters we have explored a broad range of values, namely: ❶ `rows`: 1 to 100, ❷ `cols`: 1 to 100, ❸ `eventsPerLP`: 1 to 20, and ❹ `GVT` rate/period (i.e., number of scheduler cycles after which a GVT computation is triggered if one is not already underway): 1000 to 10,000. These are the range of values in the x-axes of the GSA sub-charts. For `rows` and `cols` we have limited them to 100×100 so that at `eventsPerLP` = 20, the peak memory usage of the simulation will not exceed the NUMA memory limit (4 GB/core) of the compute nodes. The NUMA hardware configuration of the compute nodes is shown in the adjacent figure. Exceeding the NUMA threshold caused the sequential simulation times to vary a lot and 3 simulation replications were no longer sufficient to get a good estimate.

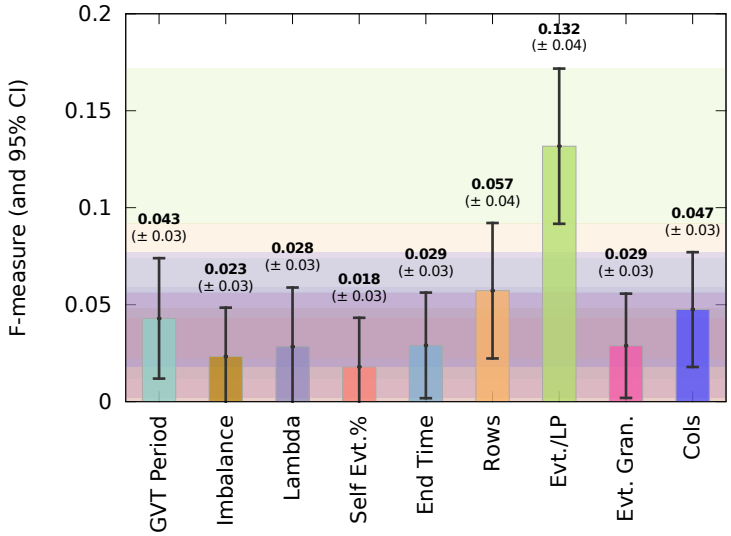
```
$ numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 2 4 6
node 0 size: 16384 MB
node 0 free: 15383 MB
node 1 cpus: 1 3 5 7
node 1 size: 16371 MB
node 1 free: 15304 MB
node distances:
node   0   1
  0:  10  20
  1:  20  10
```

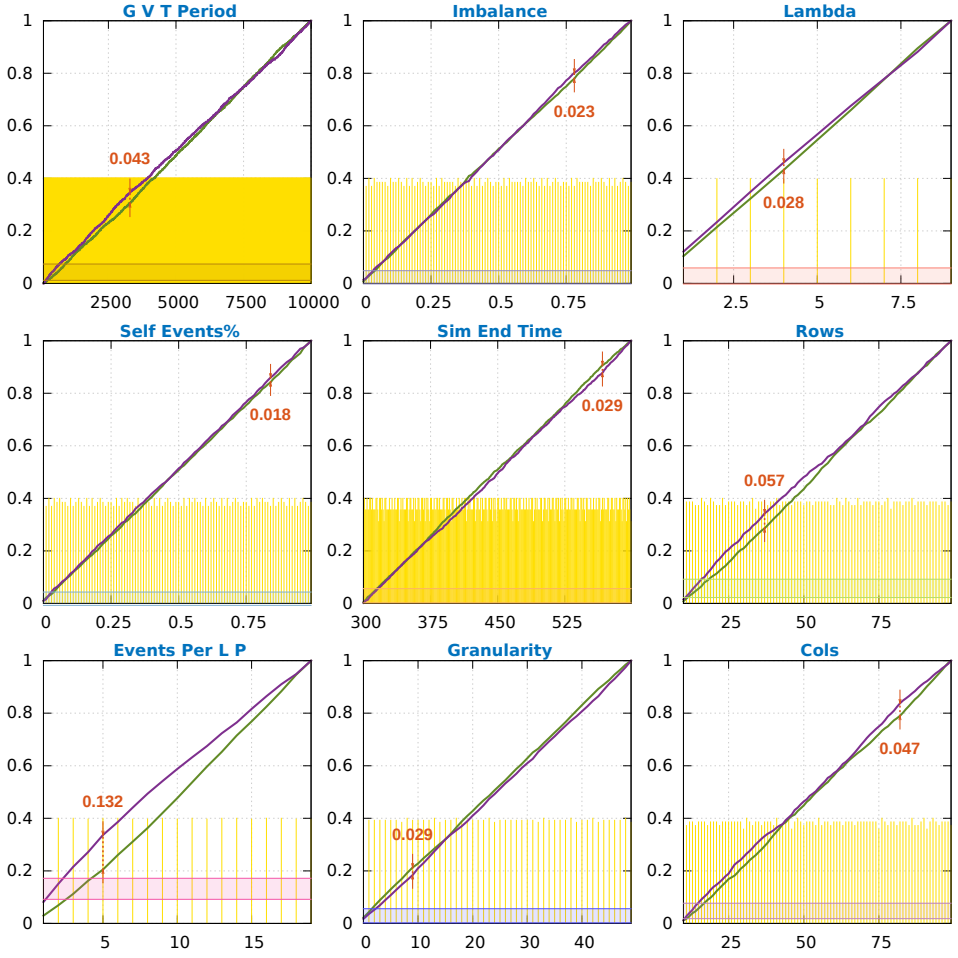
Generalized Sensitivity Analysis (GSA) for Sequential Simulations

This section includes results from the various GSA that was conducted to assess the impact of various PHOLD parameters on relative performance of the six scheduler queues. The 2-tier Ladder Queue (2tLadderQ) is used as the reference for comparison in all cases. The data in each figure is from 2500 (parameter combination) × 3 (reps/combo) = 7500 simulations.

Ladder Queue (`ladderQ`) vs. 2-tier Ladder Queue (`2tLadderQ`)

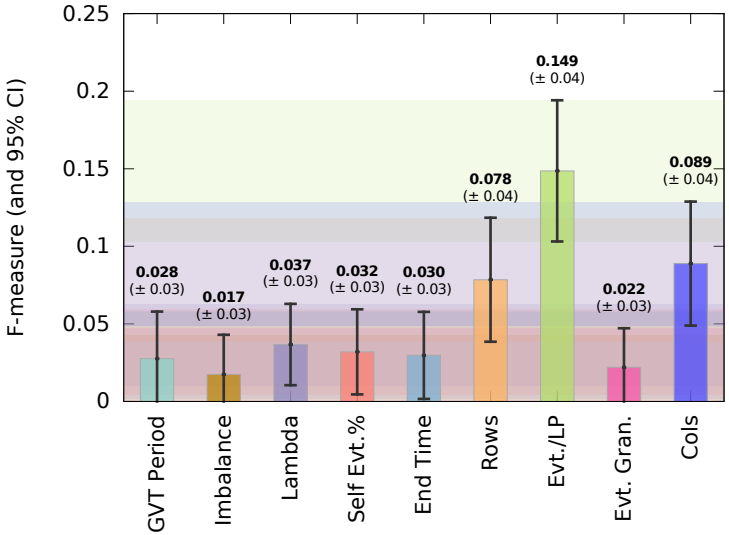
The GSA data shows that in sequential simulations, the performance of `2tLadderQ` (with $_{2t}$k=1) is very close to that of `ladderQ` immaterial of the parameter settings. The `eventsPerLP` parameter slightly increases the performance of `2tLadderQ`.

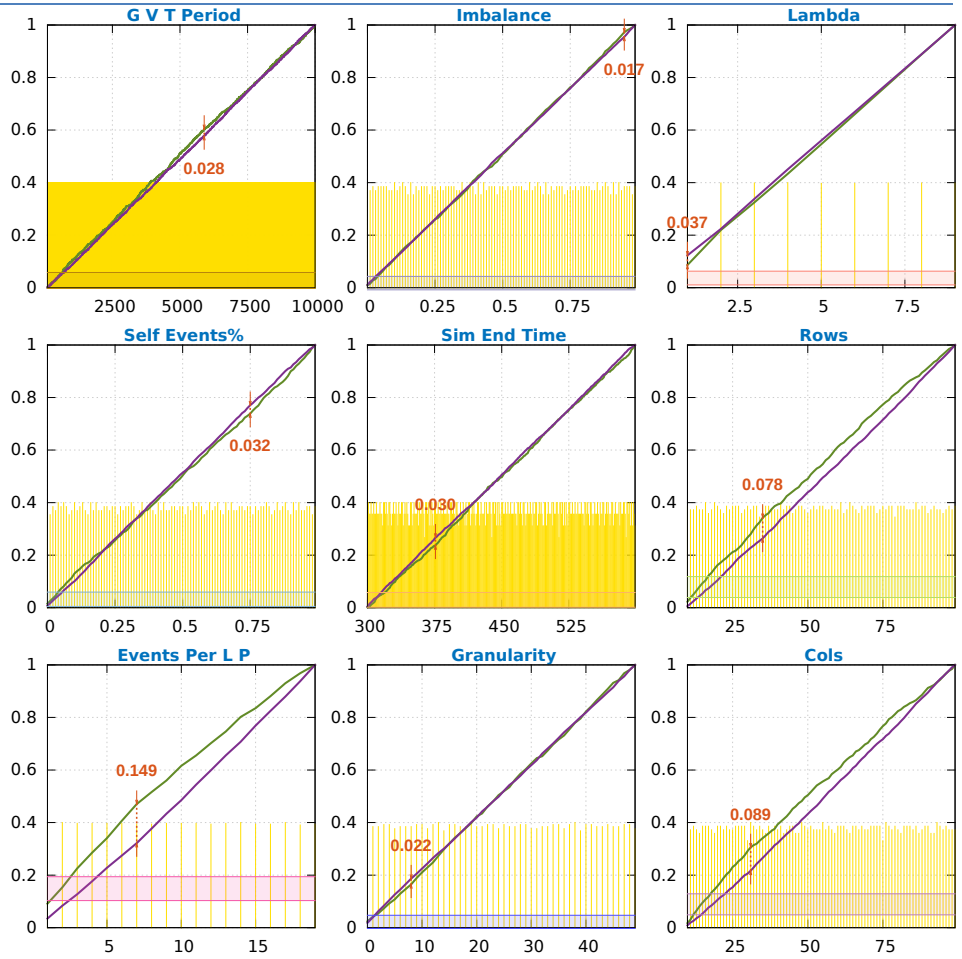




2-tier Ladder Queue (`2tLadderQ`) vs. 1-tier binary Heap (`heap`)

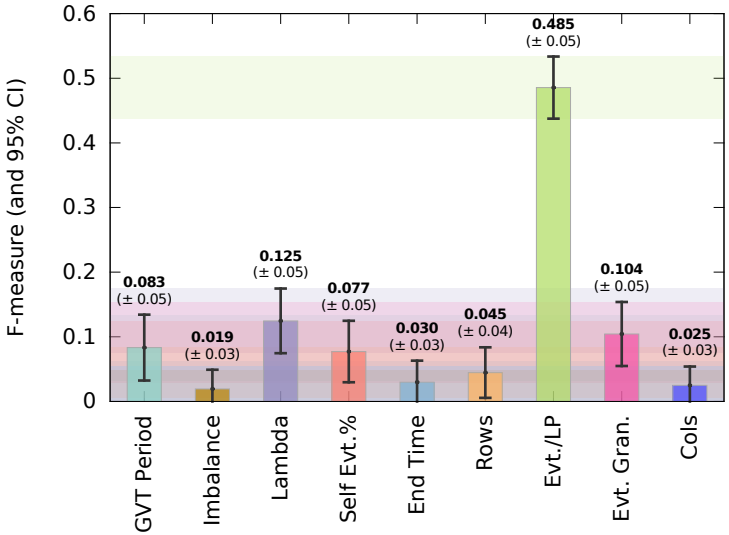
The GSA data shows that in sequential simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with $_{2t}$k=1) and `heap` with $d_{m,n}$ << 0.1. Experiments show that `2tLadderQ` generally performs better than heap. Higher values of `Evt./Agent` (`eventsPerLP`) parameter cause the `2tLadderQ` to perform much better than the `heap`.

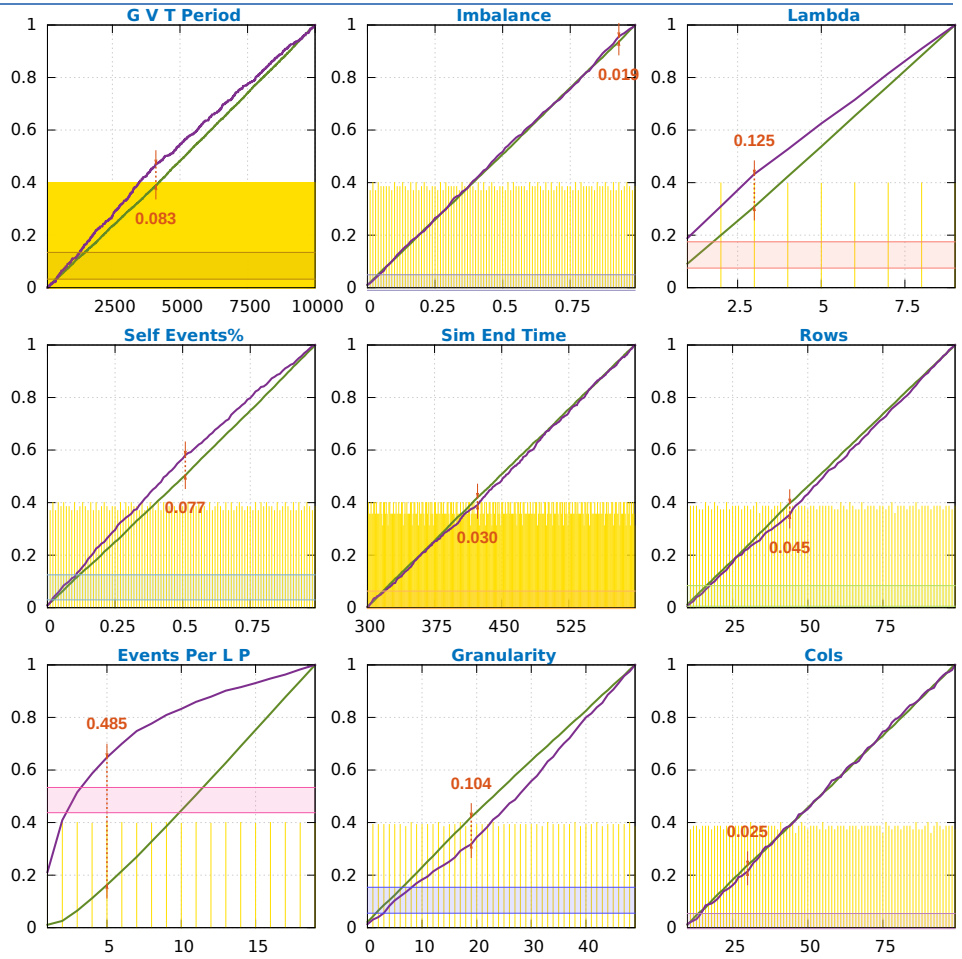




2-tier Ladder Queue (`2tLadderQ`) vs. 2-tier Heap (`2tHeap`)

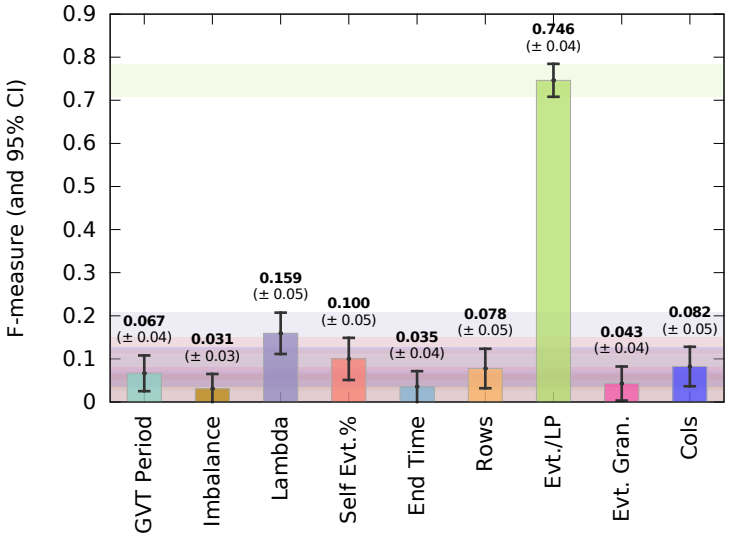
The GSA data shows that in sequential simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with ${}_{2t}k=1$) and `2tHeap` with $d_{m,n} << 0.1$. Experiments show that `2tLadderQ` generally performs better than heap. However, higher values of `Evt./LP` (`eventsPerLP`) parameter cause the `2tHeap` to start performing better than the `2tLadderQ`.

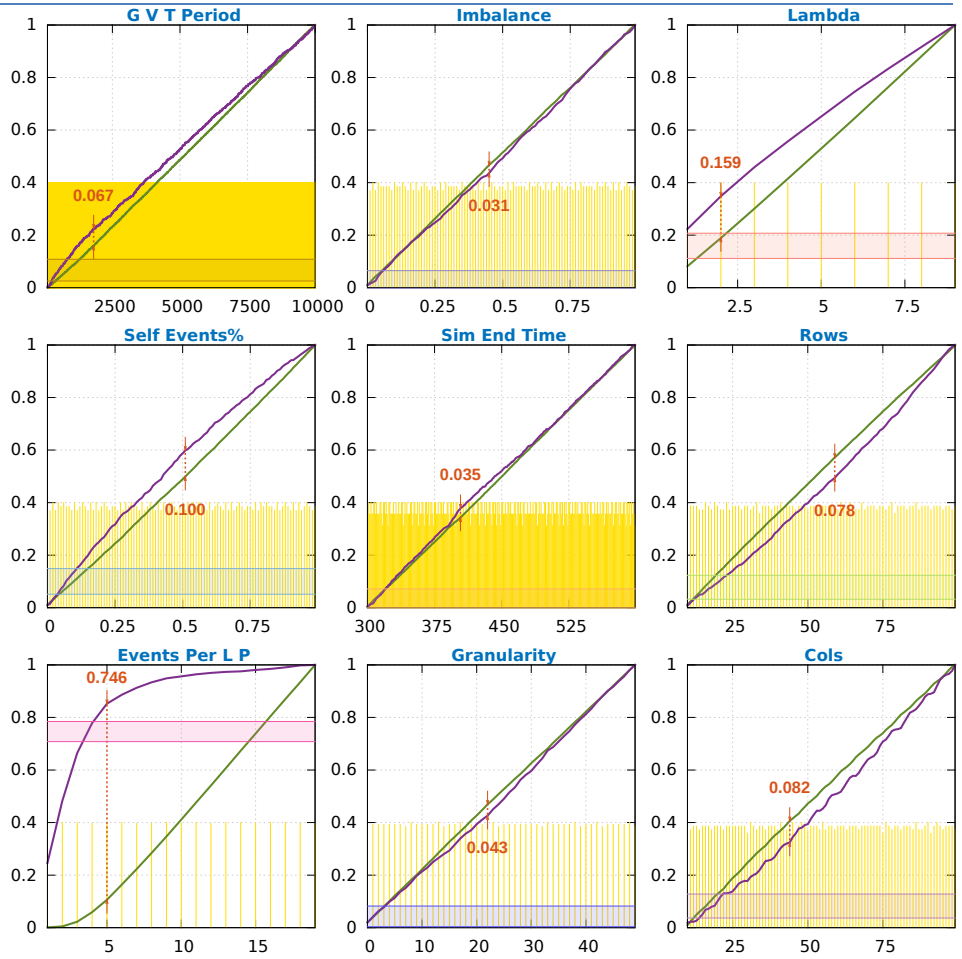




2-tier Ladder Queue (`2tLadderQ`) vs. 2-tier Fibonacci Heap (`fibHeap`)

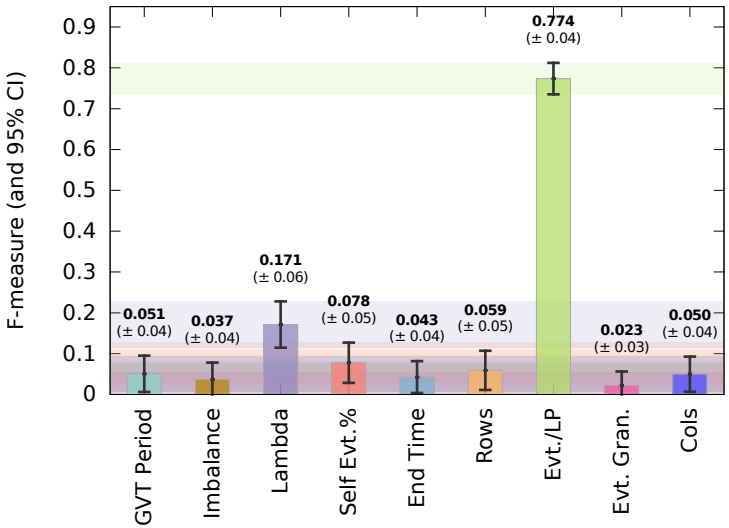
The GSA data shows that in sequential simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with $_{2t}$k=1) and `fibHeap` with $d_{m,n}$ << 0.1. Experiments show that `2tLadderQ` generally performs better than `fibHeap`. However, higher values of `Evt./LP` (`eventsPerLP`) parameter cause the `fibHeap` to start performing better and catches up with the performance of the `2tLadderQ`.

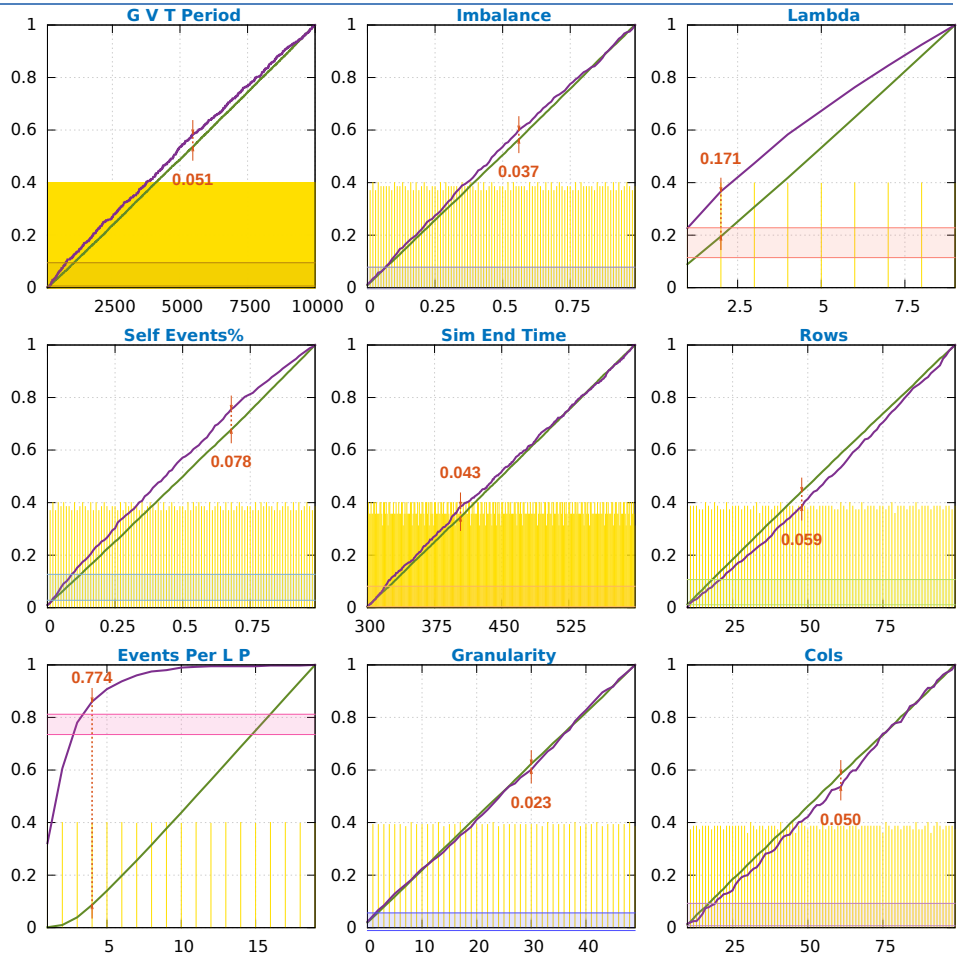




2-tier Ladder Queue (`2tLadderQ`) vs. 3-tier Heap (`3tHeap`)

This is the same data shown in the PADS paper. The GSA data shows that in sequential simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with $_{2t}k=1$) and `3tHeap` with $d_{m,n} << 0.1$. Experiments show that 2tLadderQ generally performs better than heap. However, higher values of `Evt./LP` (`eventsPerLP`) parameter cause the `3tHeap` to start performing better than the `2tLadderQ`.





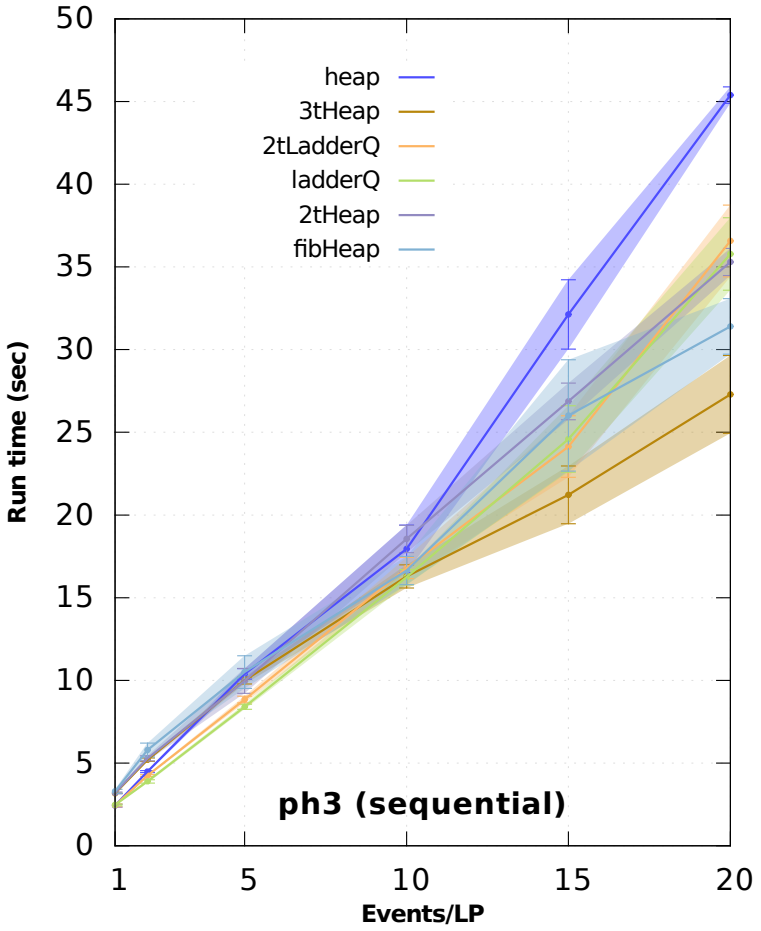
Additional sequential simulation results

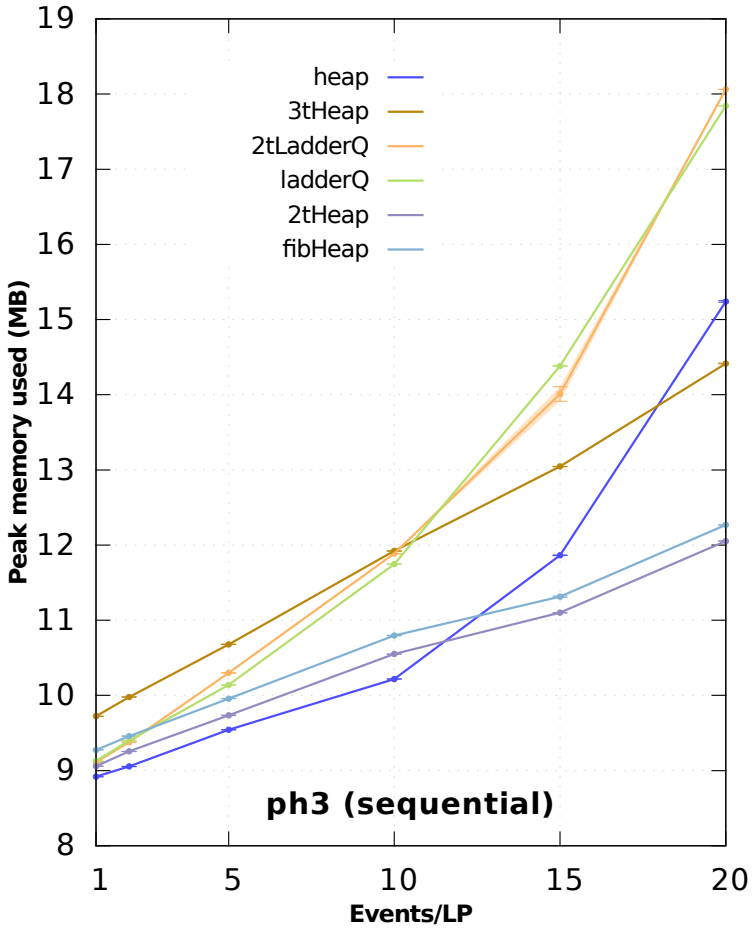
The sequential simulation experiments reported in the paper systematically explore the parameter space of the most influential parameters identified via GSA, namely:

1. `eventsPerLP` = {1, 2, 5, 10, 15, 20}
2. `%Self-Events` = {0.0, 0.25}
3. λ (`delay`) = {1, 2, 5, 10}

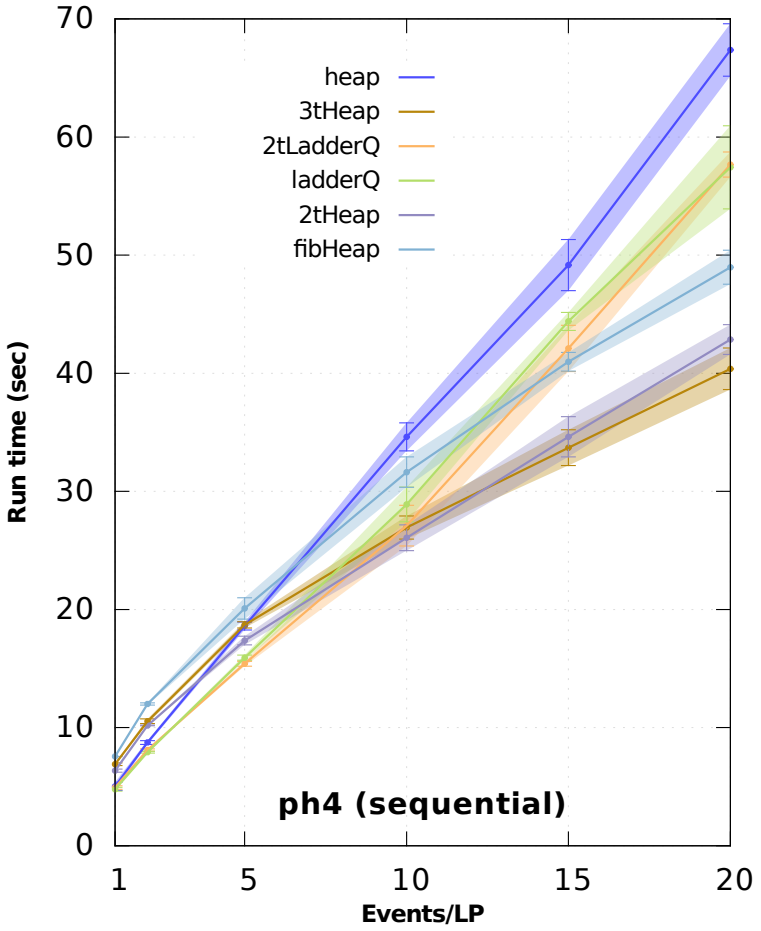
The combination of parameters resulted in 48 different combination of parameters. Ten simulation replications for each configuration (total of 480 simulations) have been used to report the simulation results in the paper. The paper included data for the 4 queues, namely: `heap`, `ladderQ`, `2tLadderQ`, and `3tHeap`. These queues performed best in at least one configuration explored. The charts below show the full data set for simulation execution times and the peak memory used for all 6 queues using the same configurations show in the paper. The data is average values from 10 independent simulation replications along with 95% Confidence Intervals (CI)

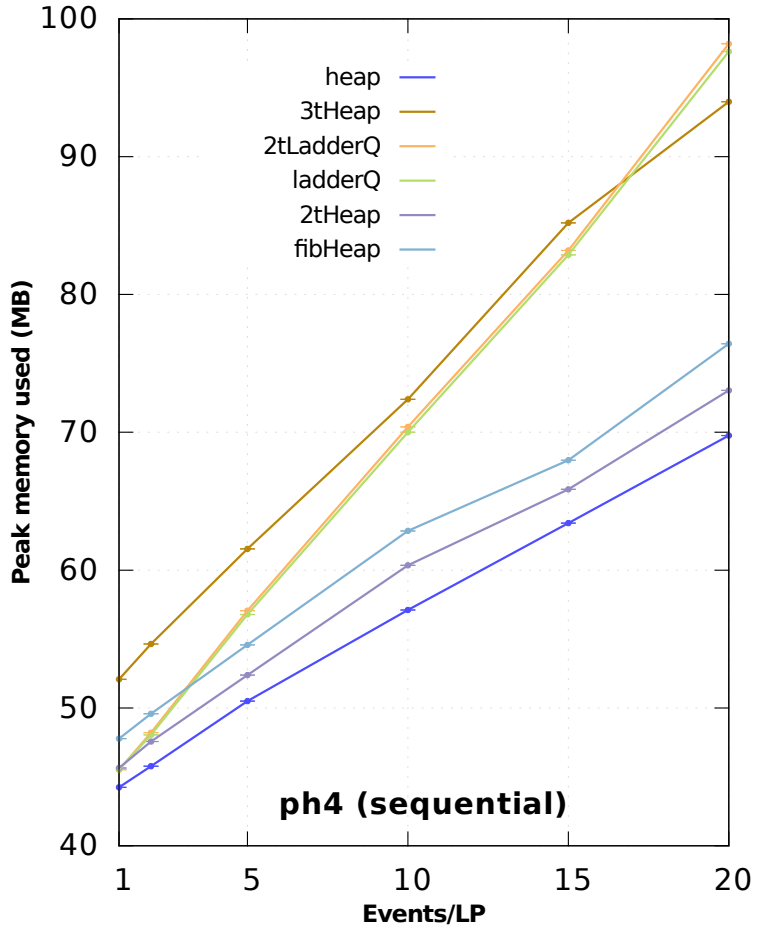
Sequential simulation runtime and peak memory use for the `ph3` PHOLD configuration with 1,000 (10^3) agents. The data is for λ=1, %Self-Events = 25%. The full set of data for all the other configurations will be in raw form as a table online.



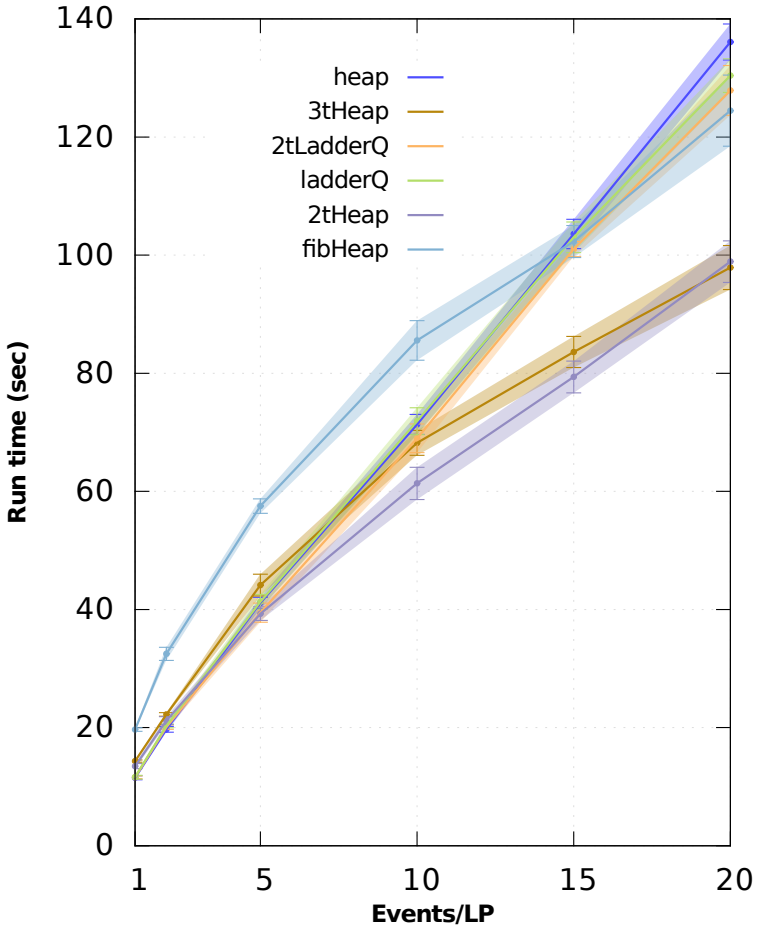


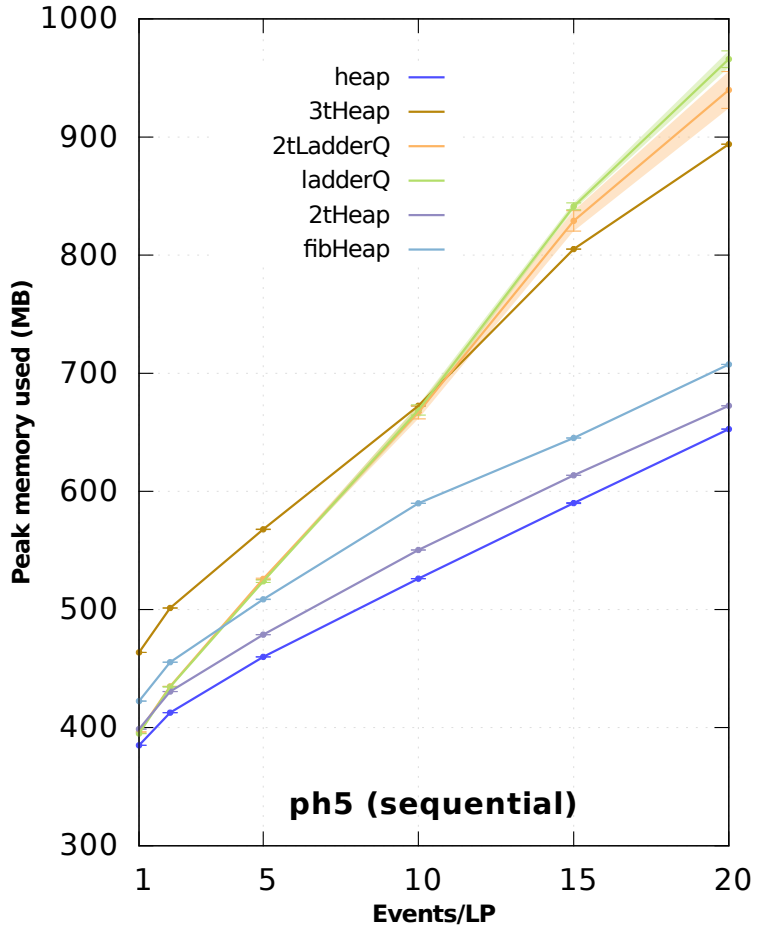
Sequential simulation runtime and peak memory use for the `ph4` PHOLD configuration with 10,000 (10^4) agents. The data is for λ=1, %Self-Events = 25%. The full set of data for all the other configurations will be in raw form as a table online.





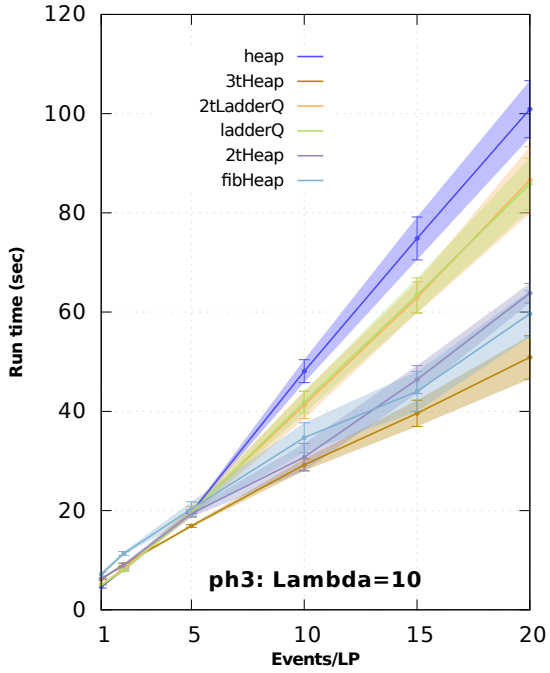
Sequential simulation runtime and peak memory use for the `ph5` PHOLD configuration with 100,000 (10^5) agents. The data is for λ=1, %Self-Events = 25%. The full set of data for all the other configurations will be in raw form as a table online.

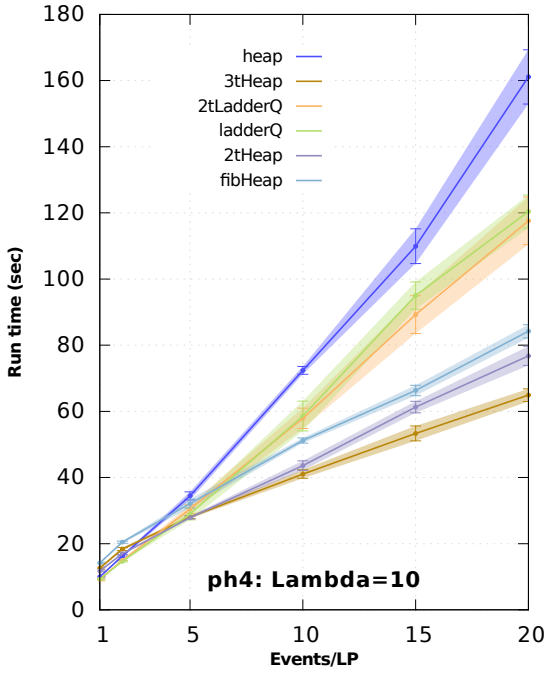


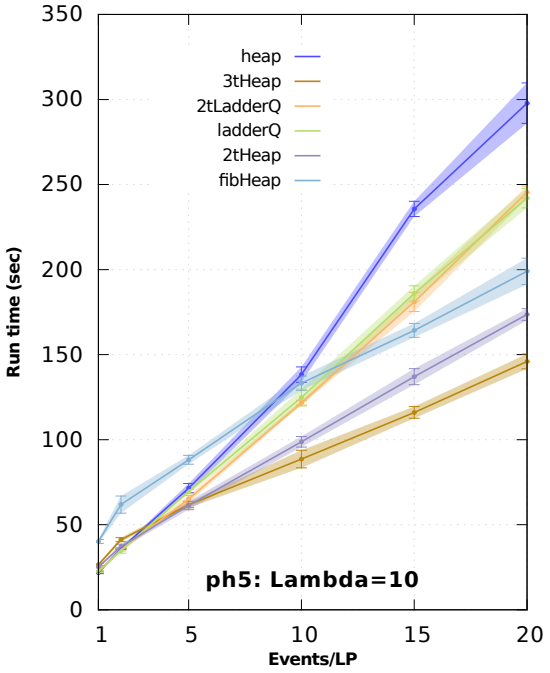


Sequential simulation: λ =10 (best-case for `3tHeap` in our data)

The simulation runtime for the 3 PHOLD configurations with λ =10 is shown below. Recollect that with λ =10, majority of the events are scheduled close to the current epoch and consequently the number of concurrent event with the same timestamp value increases per LP. This is generally the best-case timing for `3tHeap` in our experiments. In this situation the `3tHeap` consistently outperforms all the other queues.

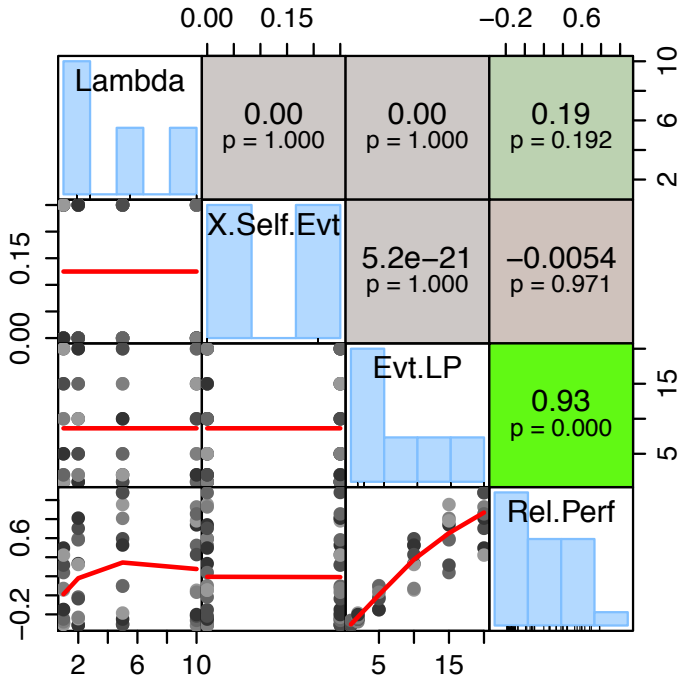






Cross-verification of GSA results between `ladderQ` and `3tHeap`

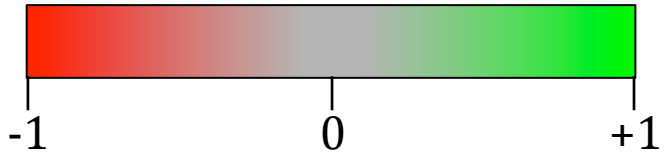
The simulation results from 10 replications of 48 different configurations provide an additional opportunity to cross-verify the GSA results by exploring correlations between the 3 parameters and the performance difference observed between `2tLadderQ` and `3tHeap`. For this we have converted the raw runtimes to percentage difference in runtime between `2tLadderQ` and `3tHeap`. We chose the `ph4` configuration, which was the mid-sized configuration. The corellogram was plotted using R and the `PerformanceAnalytics` package.



The diagonal shows the parameters being analyzed and the bottom-right corner corresponds to the relative performance between the queues.

The lower-triangle of the corellogram show the scatter plots between the pair of parameters on the diagonals. For example, the bottom-right most scatterplot corresponds to the data `Evt.LP` vs `Rel.Perf`. The red lines in the scatter plot show the LOESS smoothed fitted curve that eases observing potential trends in the data. Flat/horizontal lines correspond to low correlation while diagonal lines show high correlation.

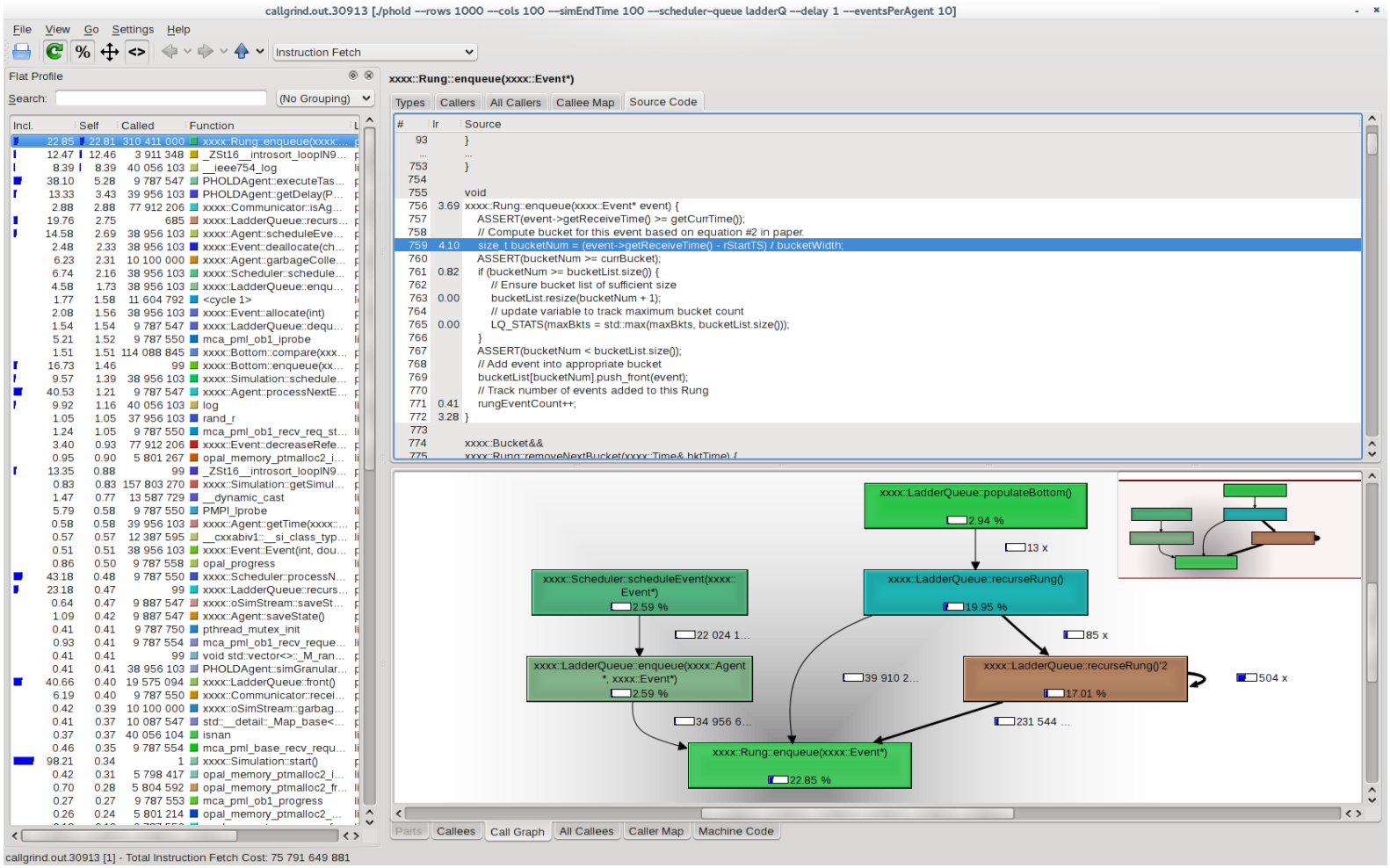
The upper-triangle of the corellogram show the Pearson correlation coefficient and p-value between each pair of parameters on the diagonal. The boxes are also color coded as shown in the scale below:



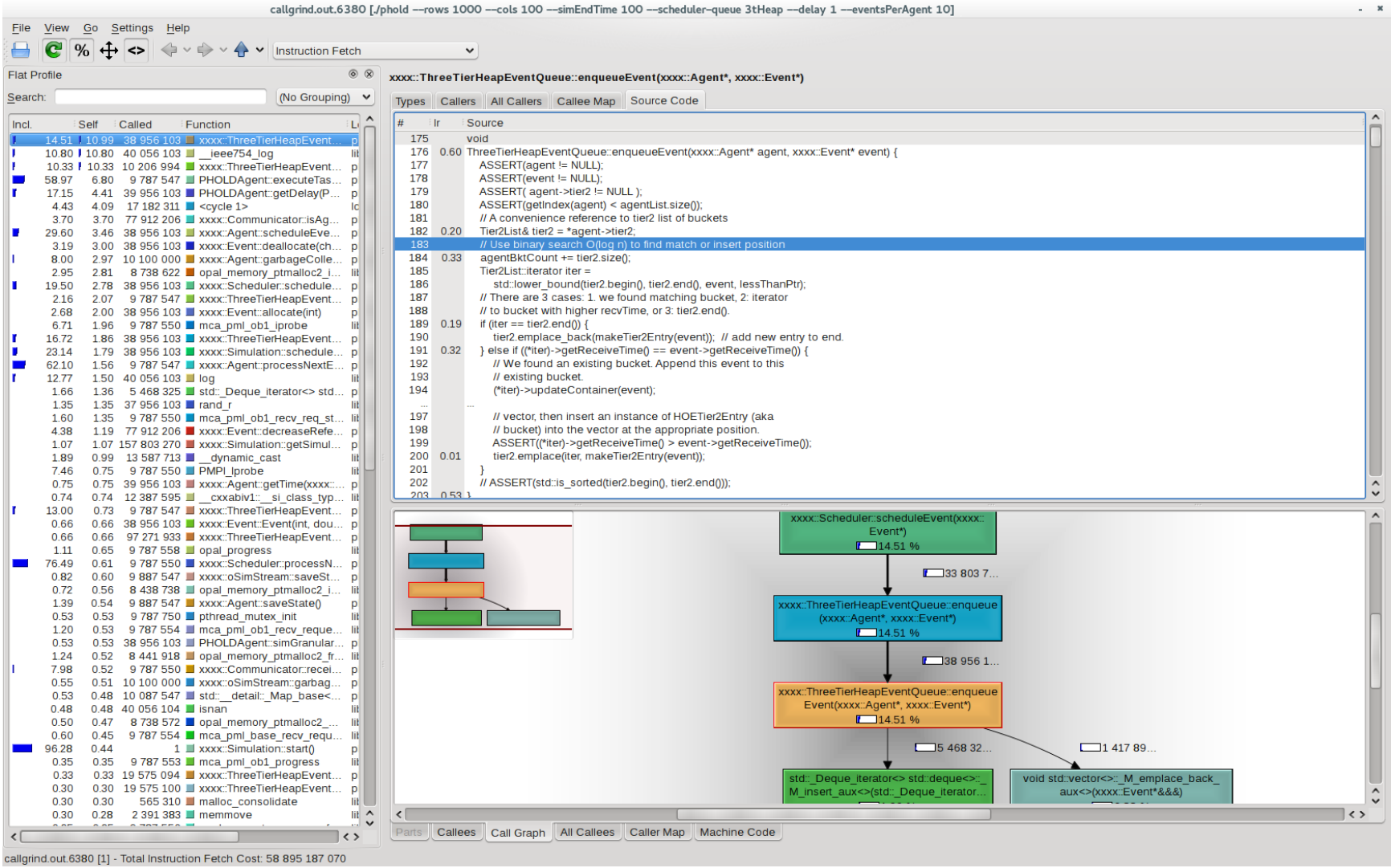
The corellogram verifies the results from GSA analysis by confirming that only Evt/LP has the strongest correlation with relative performance between `ladderQ` and `3tHeap`. Lambda (λ) also has a small influence on the relative performance, consistent with GSA results.

Profiler data for Ladder Queue and 3-tier Heap

The runtime profile was collected using `valgrind`'s `callgrind` tool on a sequential simulation using the `ph4` (medium size) model with λ=1, `eventsPerAgent`=10, and %Self Events=0.0. The profiler data shows that the overhead of re-bucketing events from rung-to-rung of the ladder takes time, even though each operation is very light weight coming in at less than 22 CPU cycles per call.



Profile data for Ladder Queue showing increasing overhead of rung insertions and/or rung-to-rung re-bucketing



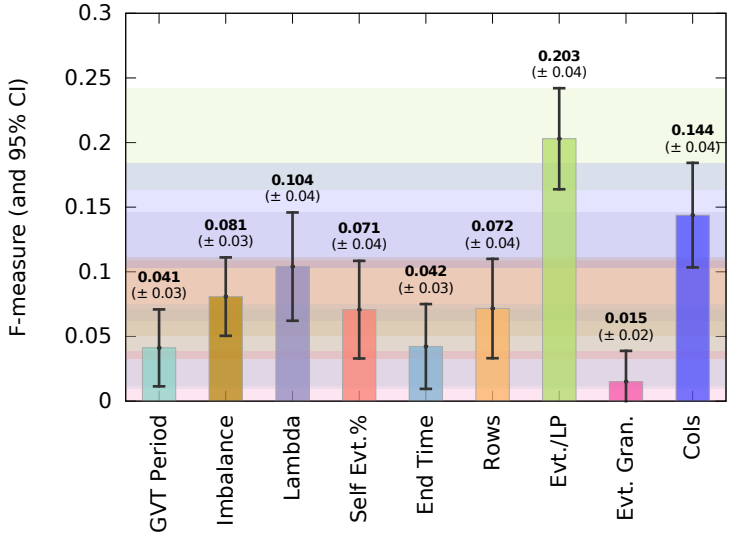
Profile data for 3tHeap for the same configuration of Ladder Queue. Most time is spent in inserting in 2nd/3rd tier

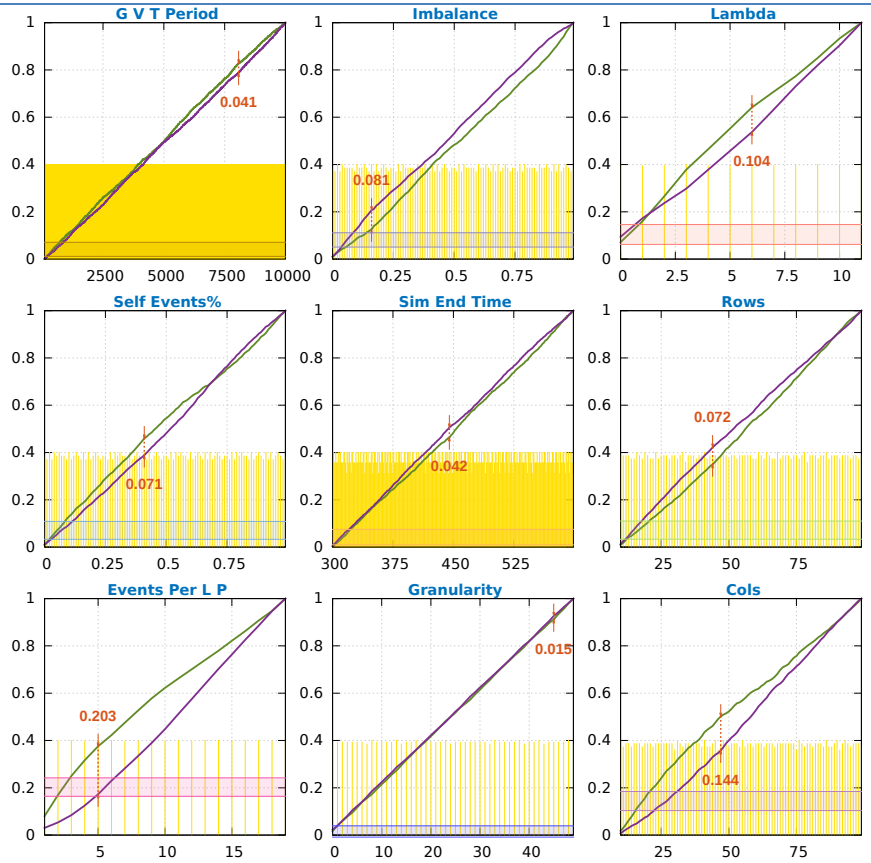
Generalized Sensitivity Analysis (GSA) for Parallel Simulations

This section includes results from the GSA that was conducted to assess the impact of various PHOLD parameters on relative performance of `ladderQ`, `2tLadderQ`, and `3tHeap` in parallel simulations. The parallel simulations were conducted using 4 MPI-processes and with a time-window of 10 simulation units. The time-window is consistently used in all parallel simulations to ensure that cascading rollbacks thereby providing runtimes with lower variance do not bog down the `ladderQ`. The runtimes for each GSA configuration are average of 3 runs. The data is from 2500 × 3 = 7500 simulations. The 2-tier Ladder Queue (2tLadderQ) is used as the reference for comparison in all cases

Ladder Queue (`ladderQ`) vs. 2-tier Ladder Queue (`2tLadderQ`)

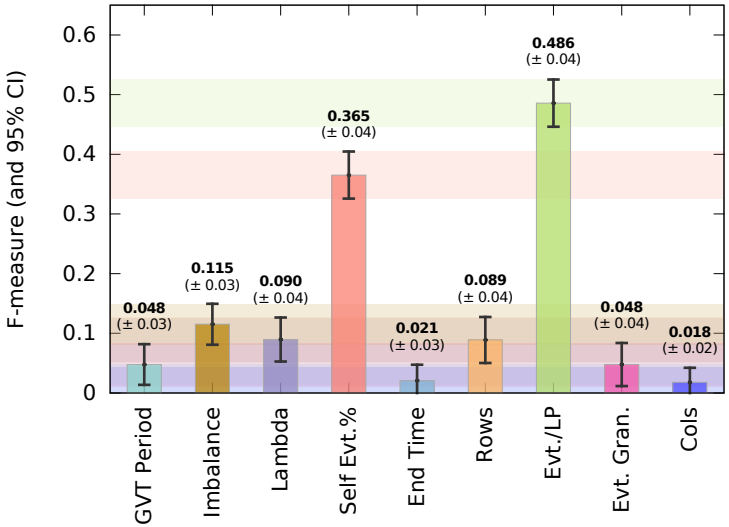
The GSA data shows that in parallel simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with $_{2t}k$=128) and `ladderQ` with $d_{m,n} << 0.1$. Experiments show that `2tLadderQ` generally performs as well or better than `ladderQ`. However, higher values of `Evt./LP` (`eventsPerLP`) parameter cause the `2tLadderQ` to perform better.

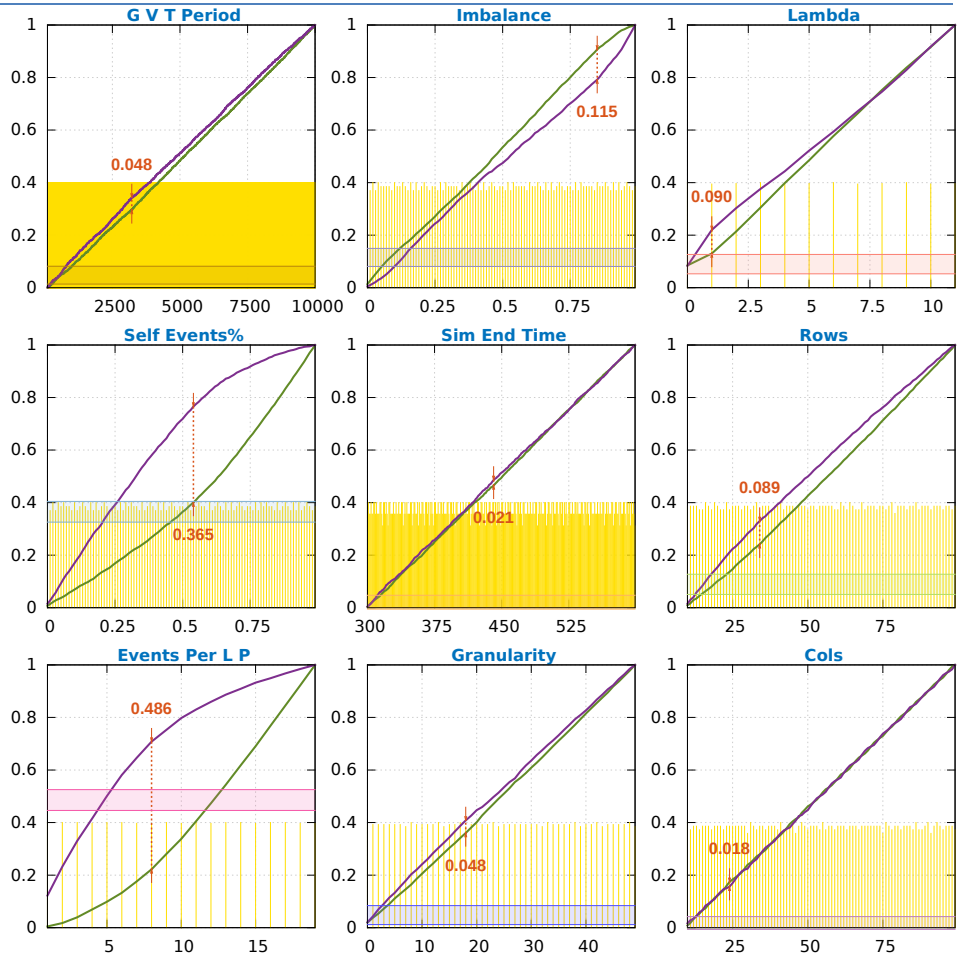




2-tier Ladder Queue (`2tLadderQ`) vs. 3-tier Heap (`3tHeap`)

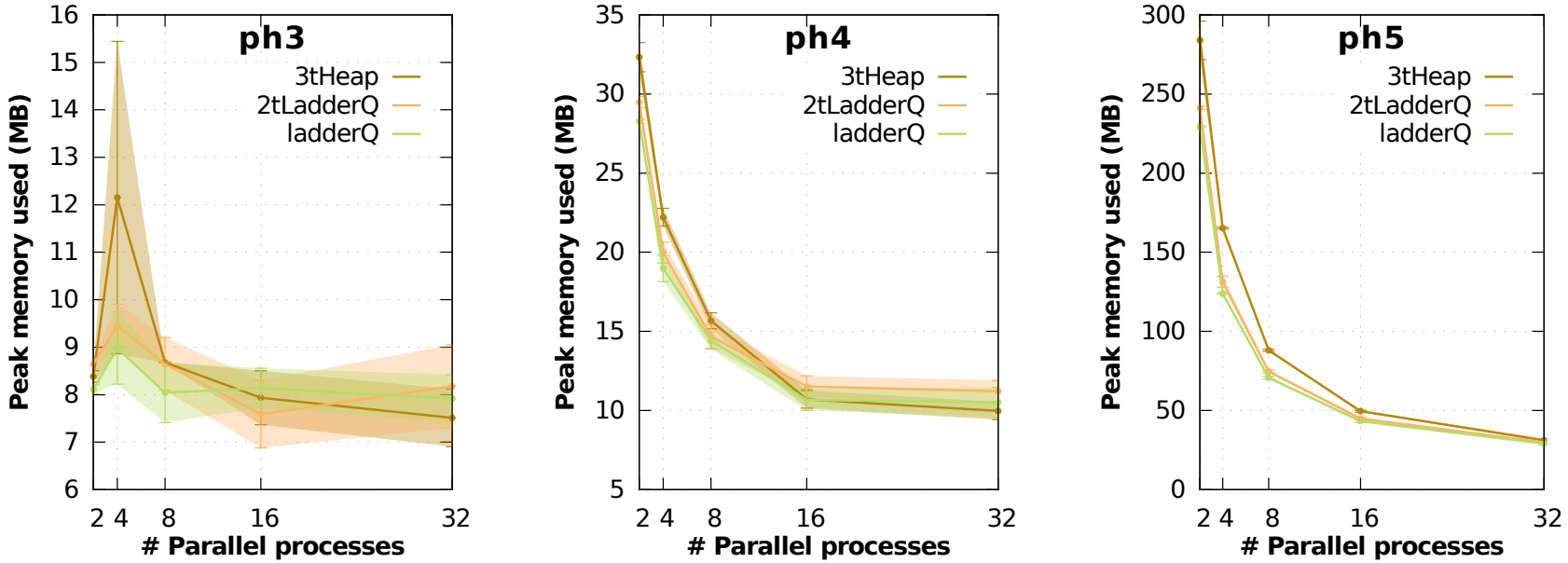
This is the same data shown in the PADS paper. The GSA data shows that in parallel simulations, most of the parameters do not influence performance difference between `2tLadderQ` (with $_{2t}k=1$) and `3tHeap` with $d_{m,n} << 0.1$. Experiments show that `3tHeap` generally performs better for higher values of values of `Evt./Agent` (`eventsPerAgent`). The `%Self Events` parameter also has some influence but the overall influence of this parameter was not conspicuous in experimental data.



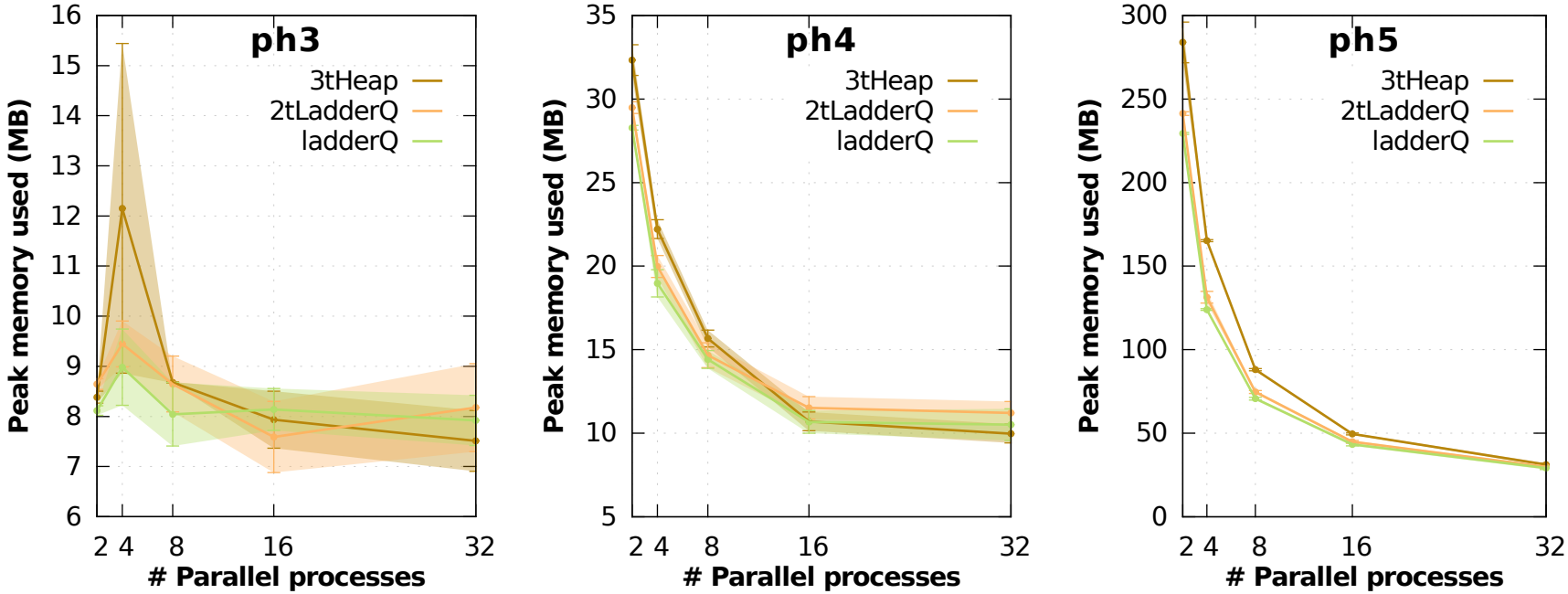


Additional parallel simulation results

The sequential simulation assessments clearly show that `ladderQ`, `2tLadderQ`, and `3tHeap` performed the best for broad range of PHOLD parameter settings. Consequently, we focused on assessing the effectiveness of these 3 queues for Time Warp synchronized parallel simulations. The experiments were conducted on our compute cluster using a varying number of MPI-processes, with one process per CPU-core. In order to ensure sufficiently long runtimes with 32-cores, we increased `simEndTime` for parallel simulations. The main paper showed simulation time and rollback data for two key configurations for `ph3`, `ph4`, and `ph5`. The memory usage for the first two configuration in the paper are shown below:



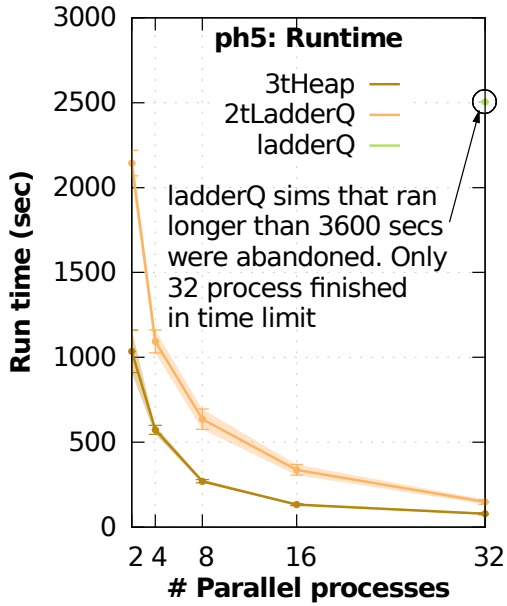
The peak memory (value reported for "`Maximum resident set size`" by `/usr/bin/time`) usage for the 3 different PHOLD configuration with `eventsPerLP`=2, λ= 1, and `%Self-Events`=0.25. This data corresponds to the best-case configuration for the `ladderQ`. This is the best case for `ladderQ` because λ= 1 produces the widest range of timestamps thereby significantly reducing the effective number of `eventsPerLP` at any given time. Consequently, buckets are smallest in this configuration.

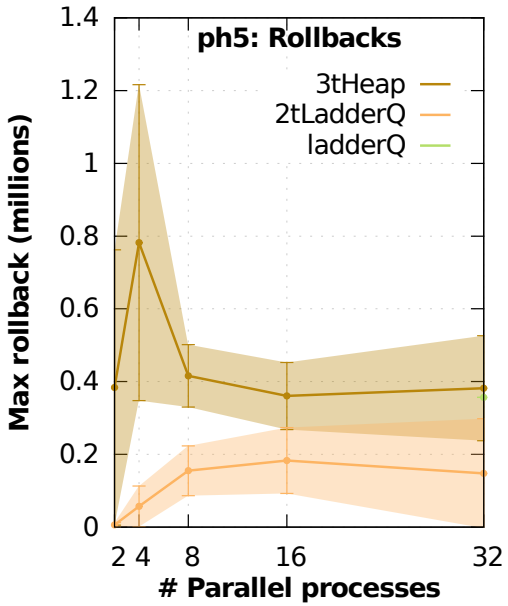


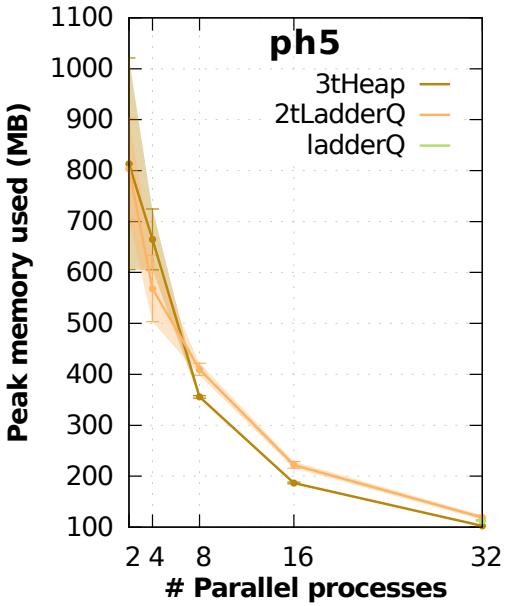
The peak memory (value reported for "`Maximum resident set size`" by `/usr/bin/time`) usage for the 3 different PHOLD configuration with `eventsPerLP`=10, λ= 10, and `%Self-Events`=0.25. This data corresponds to the knee-case where the performance of the ladder queues and `3tHeap` were similar in sequential simulations. This is the scenario in which the 3tHeap just starts to outperform the `ladderQ`. Note that this is not the best case for `3tHeap` which generally performs best for `eventsPerLP` > 10.

PH5: Best case for `3tHeap` (Evts./LP=20, λ=10, %SelfEvents=0.25)

The best case for `3tHeap` arises with high concurrency where a larger batch of events is scheduled per LP in each simulation cycle. In this configuration the `ladderQ` simulations did not complete in 3600 seconds, which was the time limit. This time limit is about 3× -- 30× more than the runtime of `3tHeap` in this configuration. The `ladderQ` simulation for 32 processes successfully completed and that is the only data point plotted in the charts below. This is the worst-case scenario for the ladder queue structures. However, the `2tLadderQ` performs well. The `3tHeap` considerably outperforms the other queues despite having a higher number of rollbacks.

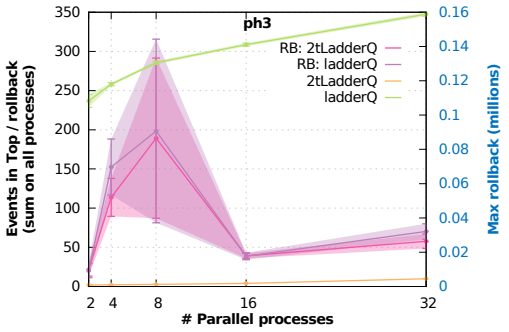


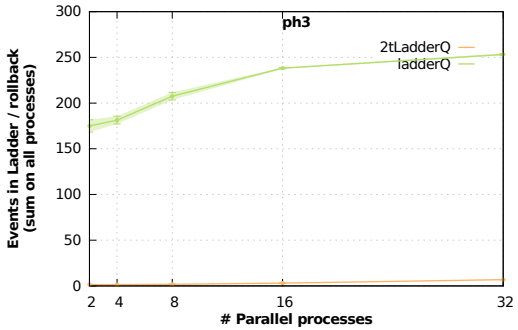


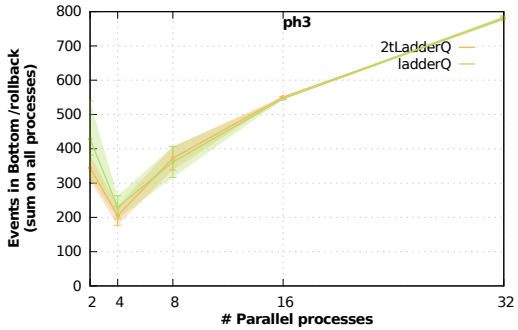


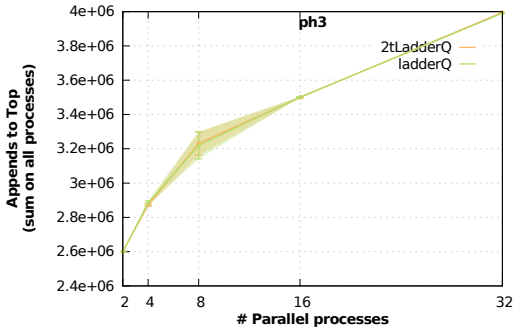
PH3: 2-tier ladder queue operations (Evts./LP=2, λ=1, %SelfEvents=0.25)

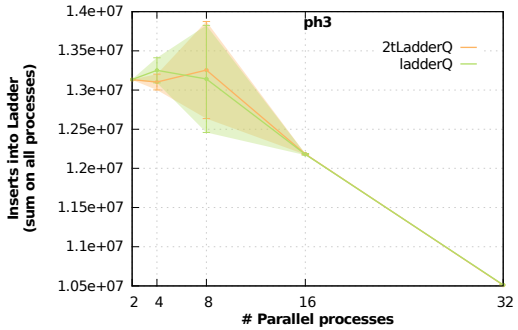
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Most of the data between the 2 queues are similar except for the number of events scanned per rollback in Top, Ladder rungs, and bottom. The difference arises from breaking the bucket into 128 sub-buckets. This essentially reduces the number scans by 1/128th as evidenced in the first 3 charts. In the case of `ph3` the total number of Bottom→Ladder operations were high but the number of events being moved were very few and that did not significantly degrade their performance. Furthermore as shown by the rollbacks in the first figure, these operations do not show strong correlation with rollbacks but with partitioning differences.

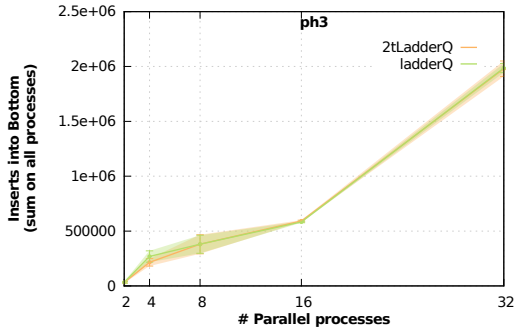


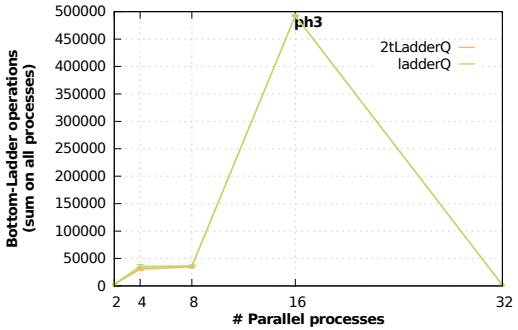






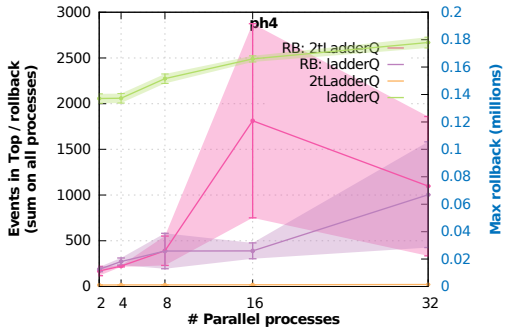


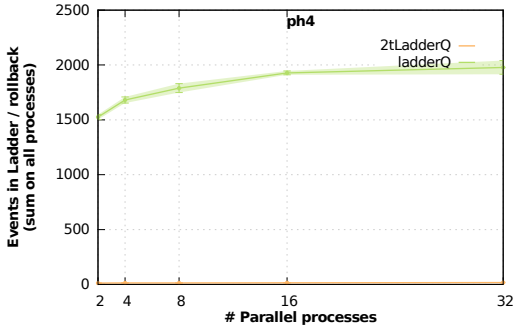


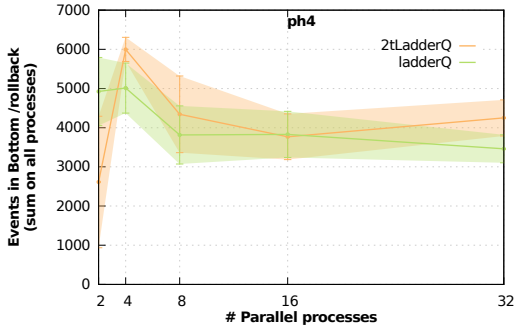


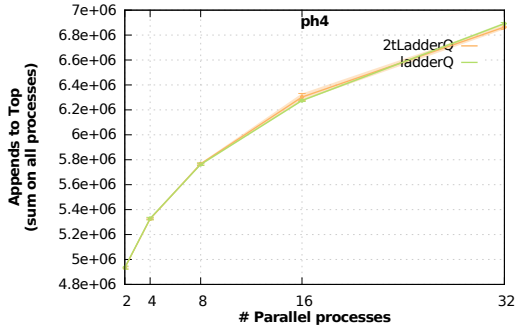
PH4: 2-tier ladder queue operations (Evts./LP=2, λ=1, %SelfEvents=0.25)

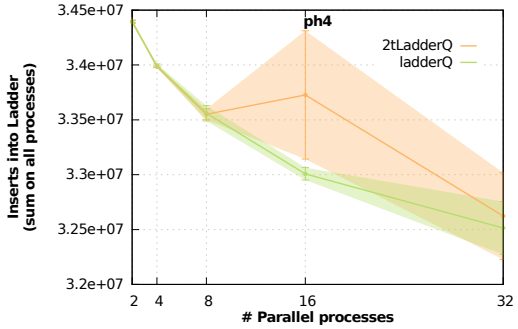
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Most of the data between the 2 queues are similar except for the number of events scanned per rollback in Top, Ladder rungs, and bottom. The difference arises from breaking the bucket into 128 sub-buckets. This essentially reduces the number scans by 1/128th as evidenced in the first 3 charts where the curve for `2tLadderQ` is close to the x-axis. The Bottom→Ladder operations are much lower in this configuration but increase as the events get spread out between the processes. However, the number of events decrease and do not hurt performance.

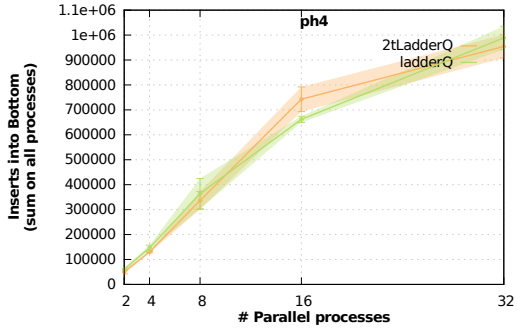


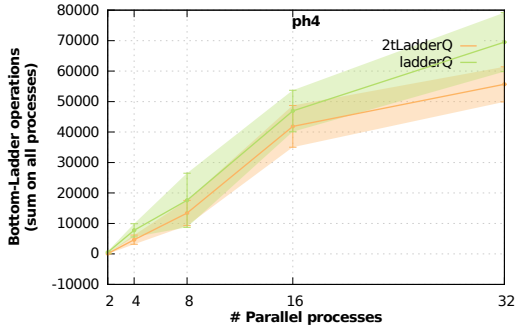






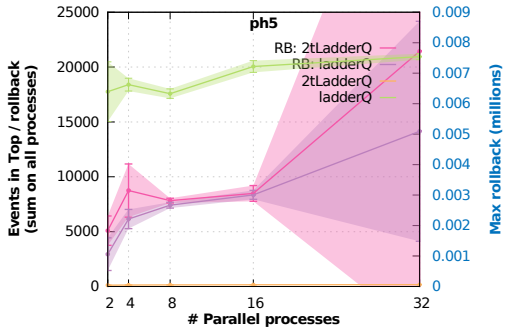


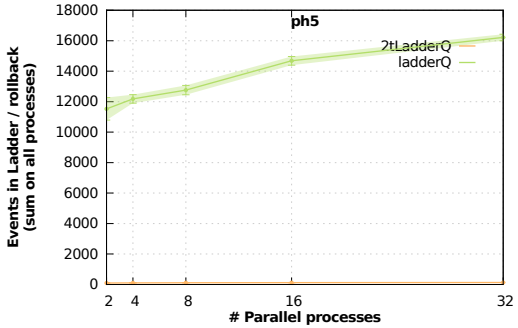


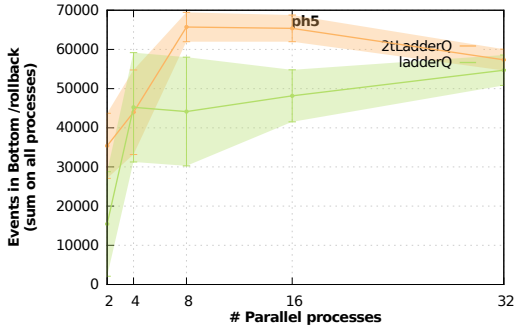


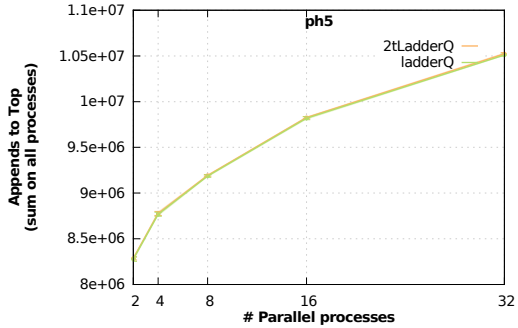
PH5: 2-tier ladder queue operations (Evts./LP=2, λ=1, %SelfEvents=0.25)

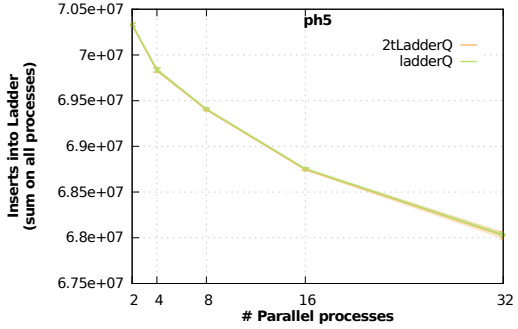
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Most of the data between the 2 queues are similar except for the number of events scanned per rollback in Top, Ladder rungs, and bottom. The difference arises from breaking the bucket into 128 sub-buckets. The number Bottom→Ladder operations were considerably lower in this case but the bottom lengths are typically much longer due to the larger number of LPs. This is the best configuration for both ladder queues.

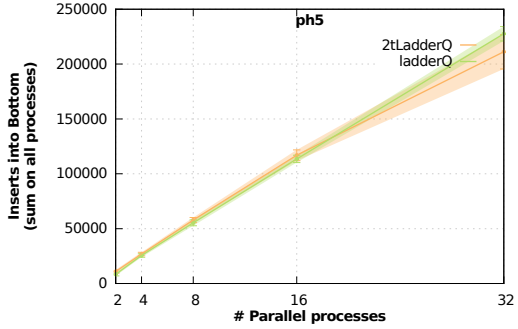


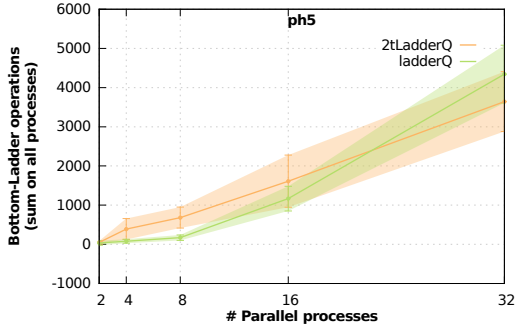






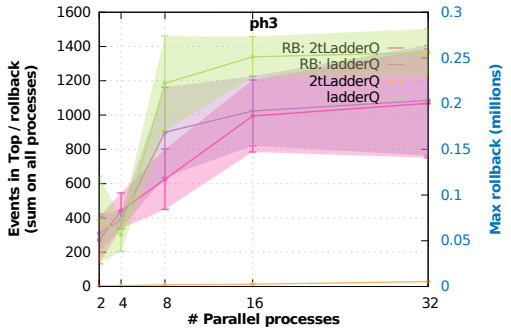


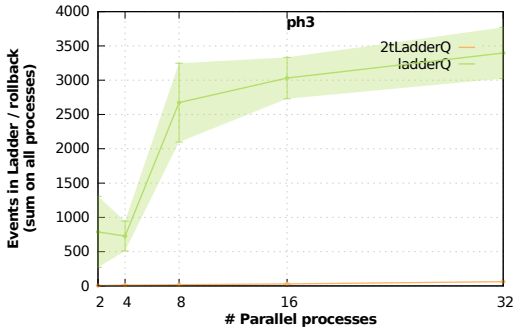


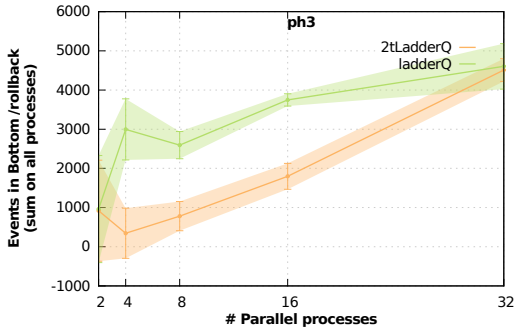


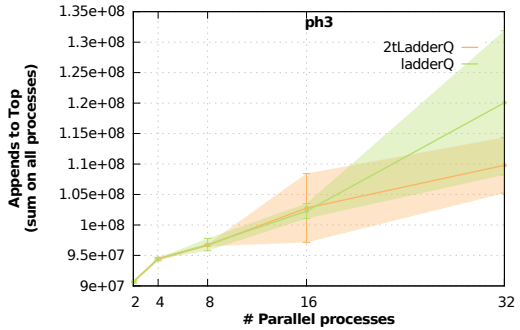
PH3: 2-tier ladder queue operations (Evts./LP=10, λ=10, %SelfEvents=0.25)

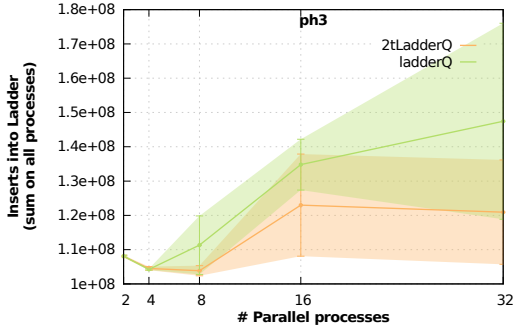
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Most of the data between the 2 queues are similar except for the number of events scanned per rollback in Top, Ladder rungs, and bottom. The difference arises from breaking the bucket into 128 sub-buckets. This essentially reduces the number scans by 1/128th as evidenced in the first 3 charts. In the case of `ph3` the total number of Bottom→Ladder operations were high but the number of events being moved were very few and that did not significantly degrade their performance. Furthermore as shown by the rollbacks in the first figure, these operations do not show strong correlation with rollbacks but with partitioning differences.

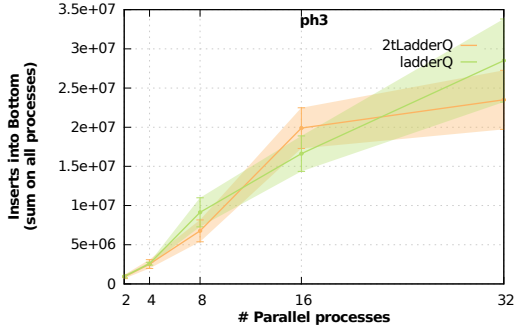


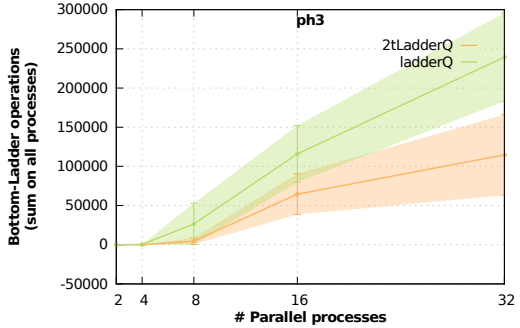






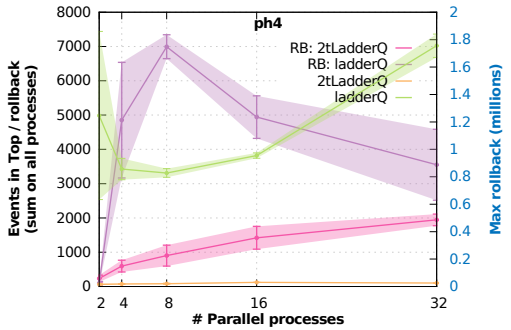


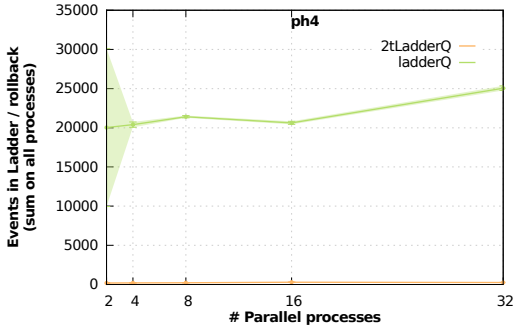


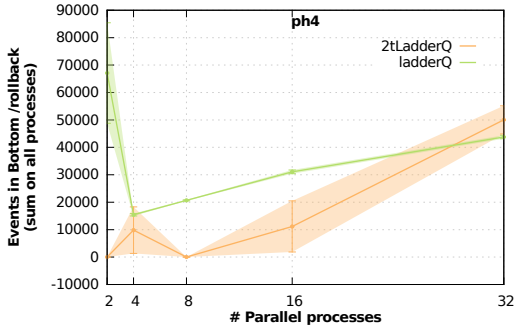


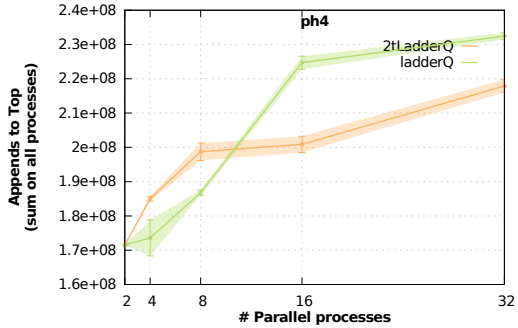
PH4: 2-tier ladder queue operations (Evts./LP=10, λ=10, %SelfEvents=0.25)

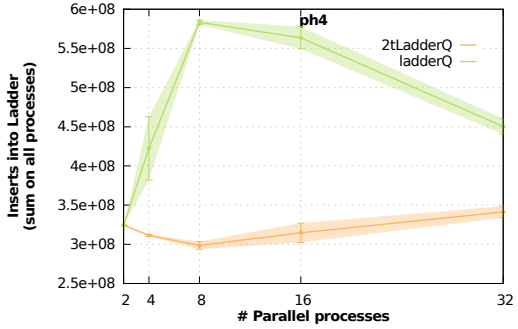
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Unlike the earlier configurations that were the efficient configuration for `ladderQ`. However, the ratio of number of events scanned per rollback in Top, Ladder rungs, and bottom remain consistently 1/128th. However, the number of Bottom→Ladder operations for `ladderQ` were much higher, possibly due to higher rollbacks for `ladderQ`.

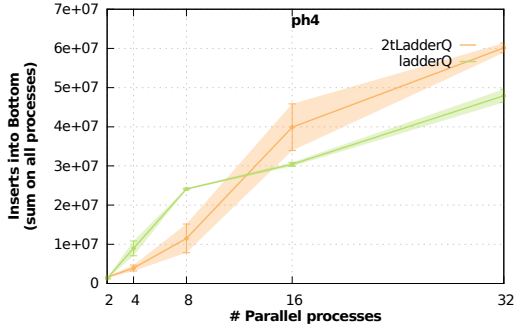


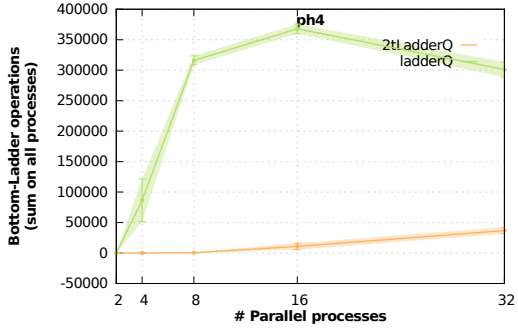






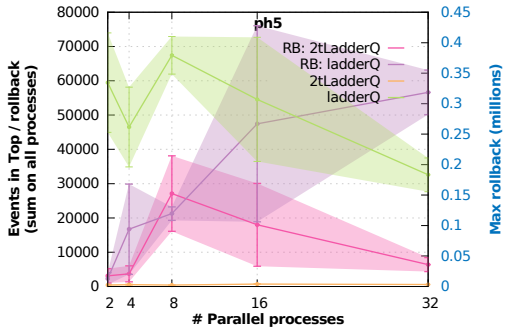


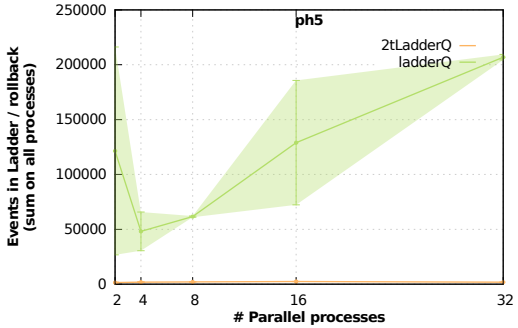


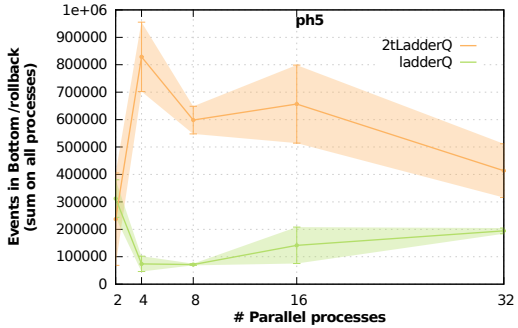


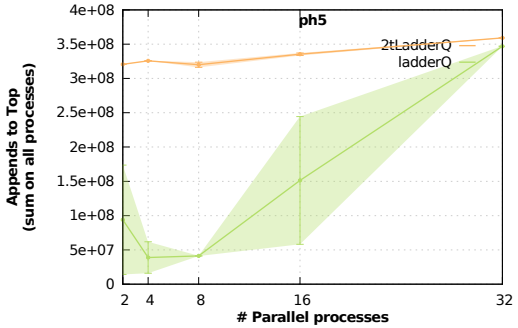
PH5: 2-tier ladder queue operations (Evts./LP=10, λ=10, %SelfEvents=0.25)

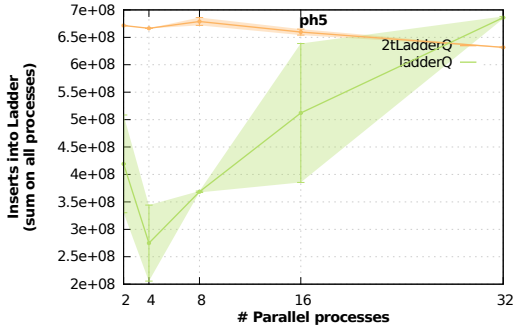
The charts show comparison of key operations between `ladderQ` and `2tLadderQ`. Unlike the earlier configurations that were the efficient configuration for `ladderQ`. However, the ratio of number of events scanned per rollback in Top, Ladder rungs, and bottom remain consistently 1/128th. However, the number of Bottom→Ladder operations for `ladderQ` were much higher, possibly due to higher rollbacks for `ladderQ`. The increased rollbacks also occur because of the increased overhead of canceling events. This causes processes to optimistically simulate ahead only to get rolledback and requiring Bottom→Ladder operations as events are scheduled for reprocessing.

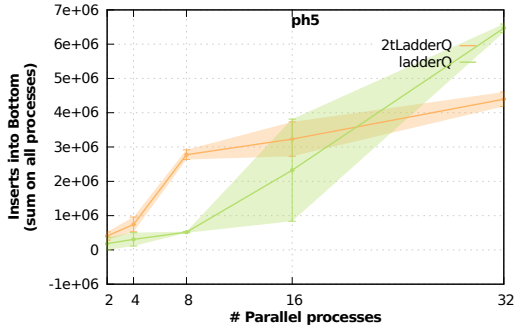


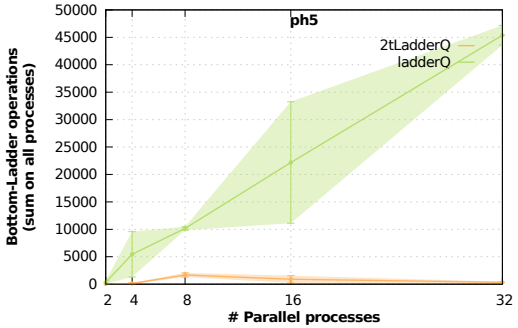












PH5: Ladder queue statistics (Best case for 3tHeap, Evts./LP=20, λ=10, %SelfEvents=0.25)

The charts show the key operations occurring in the `2tLadderQ`. The statistics suggest that the 2tLadderQ is operating in optimal modes with very few Bottom→Ladder operations. However, the length of bottom is long as indicated by the number of events checked per rollback. However, the number of events scanned in top and in the rungs is rather few about 1/128th of the events in these structures. The data for `ladderQ` is not seen because the simulations did not complete in the 3600 seconds allotted for the jobs.