An Inquiry into Machine Learning-based Automatic Configuration Tuning Services on Real-World Database Management Systems

D. Aken , D. Yang, S. Brillard, A. Fiorino, B. Zhang, C. Bilien, and A. Pavlo, VLDB, 2021

Nov 22nd 2023
Wenxuan Li

Context & Motivations

Task: DBMS configuration tuning

- hundreds of knobs to tune for optimal performance (e.g. latency / throughput)

- binlog_cache_size
- binlog_max_flush_queue_time
- binlog_stmt_cache_size
- eq_range_index_dive_limit
- host_cache_size
- innodb_adaptive_flushing
- innodb_autoextend_increment
- innodb_buffer_pool_dump_now
- innodb_buffer_pool_load_at_startup
- innodb_buffer_pool_load_now
- innodb_change_buffer_max_size
- innodb_commit_concurrency
- innodb_compression_failure_threshold_
- innodb_compression_level
- innodb_compression_pad_pct_max
- innodb_concurrency_tickets
- innodb_flush_log_at_timeout
- innodb_flush_neighbors
- innodb_flushing_avg_loops
- innodb_ft_cache_size
- innodb_ft_result_cache_limit
- innodb_ft_sort_pll_degree
- innodb_io_capacity_max
- innodb_lock_wait_timeout

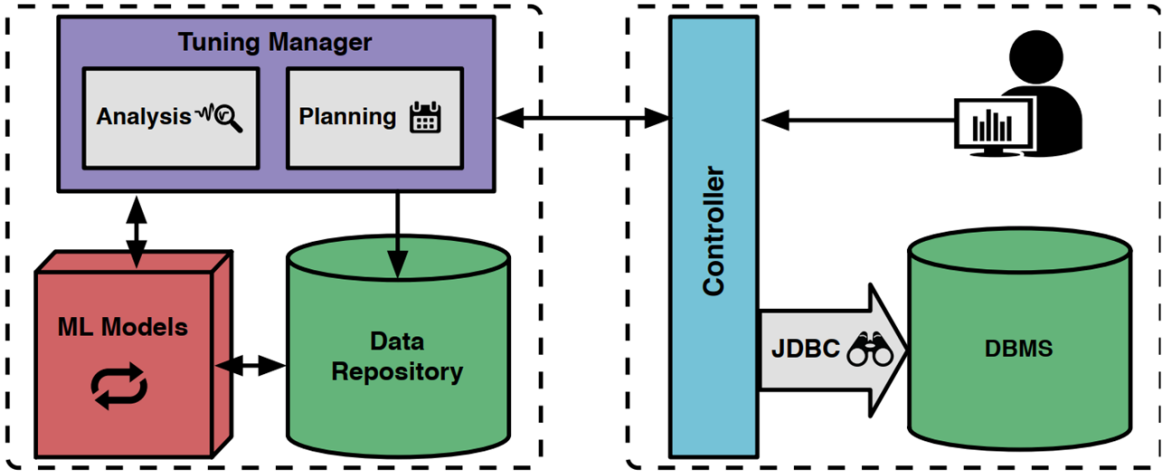
...

- autovacuum_analyze_scale_factor
- autovacuum_analyze_threshold
- autovacuum_freeze_max_age
- autovacuum_max_workers
- autovacuum_multixact_freeze_max_age
- autovacuum_naptime
- autovacuum_work_mem
- backend_flush_after
- bgwriter_flush_after
- checkpoint_flush_after
- commit_delay
- commit_siblings
- cpu_index_tuple_cost
- cpu_operator_cost
- cpu_tuple_cost
- cursor_tuple_fraction
- deadlock_timeout
- from_collapse_limit
- geqo_effort
- geqo_generations
- geqo_pool_size
- geqo_selection_bias
- geqo_threshold
- gin_fuzzy_search_limit
- gin_pending_list_limit
- join_collapse_limit
- max_locks_per_transaction

...

Prior Work: OtterTune

D. Aken et al. 2017 Automatic Database Management System Tuning Through Large-scale Machine Learning

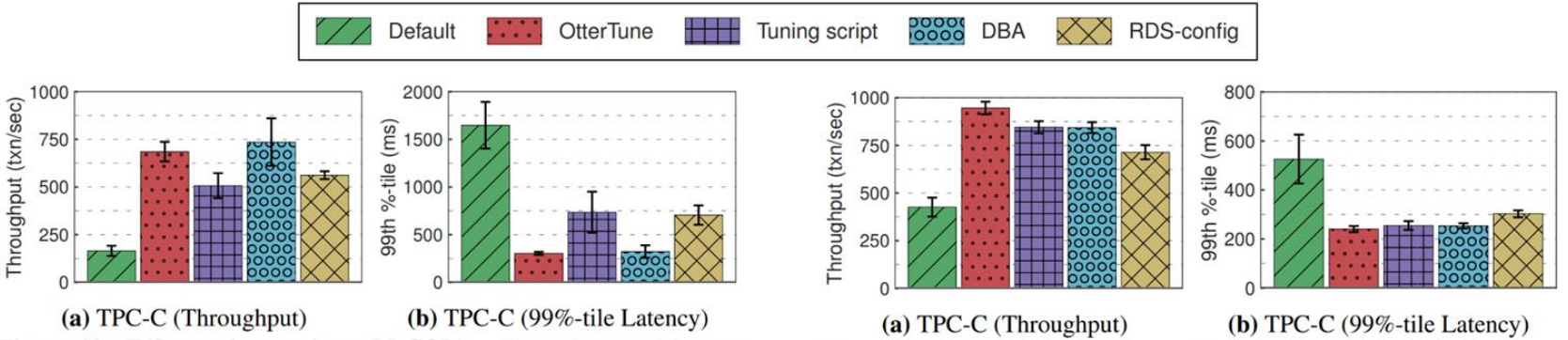


$$\theta^* = \underset{\theta \in \Theta}{argmax}\ f_w(\theta)$$

Important work: OtterTune

- An ML-based automatic DBMS configuration tuning system

Prior Work: OtterTune



(a) TPC-C (Throughput) **(b)** TPC-C (99%-tile Latency)

Figure 10: Efficacy Comparison (MySQL) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) Lithuanian DBA configuration, and (5) Amazon RDS configuration.

(a) TPC-C (Throughput) **(b)** TPC-C (99%-tile Latency)

Figure 11: Efficacy Comparison (Postgres) – Throughput and latency measurements for the TPC-C benchmark using the (1) default configuration, (2) OtterTune configuration, (3) tuning script configuration, (4) expert DBA configuration, and (5) Amazon RDS configuration.

Evaluation of OtterTune: human-comparable efficacy on configuration tuning

Problems

Problems with evaluation of prior works (not just OtterTune):

Mismatch between experimental and real-world DBMS deployments

- Workload Complexity
- System Complexity

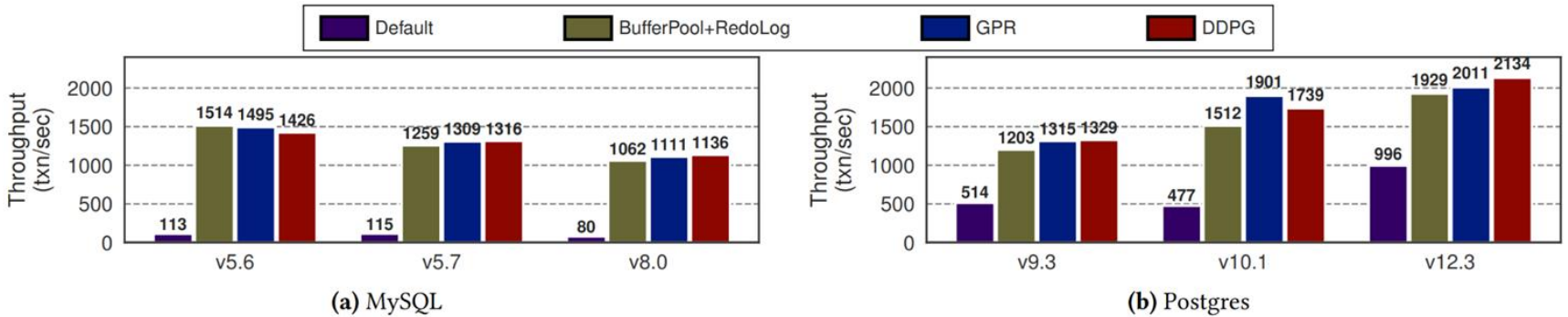


Figure 2: DBMS Tuning Comparison – Throughput measurements for the TPC-C benchmark running on three versions of MySQL (v5.6, v5.7, v8.0) and Postgres (v9.3, v10.1, v12.3) using the (1) default configuration, (2) buffer pool & redo log configuration, (3) GPR configuration, and (4) DDPG configuration.

Problems

Problems with evaluation of prior works (not just OtterTune):

Mismatch between environmental and real-world DBMS deployments

- Workload Complexity
- System Complexity
- Operating Environment

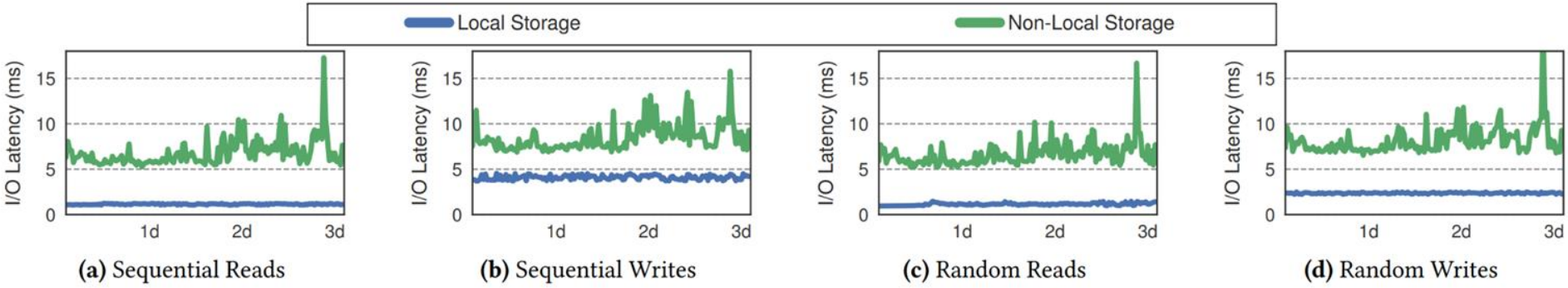


Figure 3: Operating Environment – I/O latency of local versus non-local storage for four different I/O workloads over a three-day period.

Methodology: Field Study

Field Study in real-world enterprise production environment:

Evaluating OtterTune framework at Société Générale (SG), a multi-national bank

- Target database application: TicketTracker
  - (a private issue tracking system)
  - DBMS backend: Oracle

Methodology: Environmental Settings

- database: **Snapshots**
  - using the Oracle Recovery Manager tool
- workload: trace **Collection & Replay**
  - TicketTracker workload trace collected by Oracle's Real Application Testing (RAT)
- deployment: on cloud VMs with non-local storage

Methodology: More Tuning Algorithms

- (BO) GPR and DNNs:

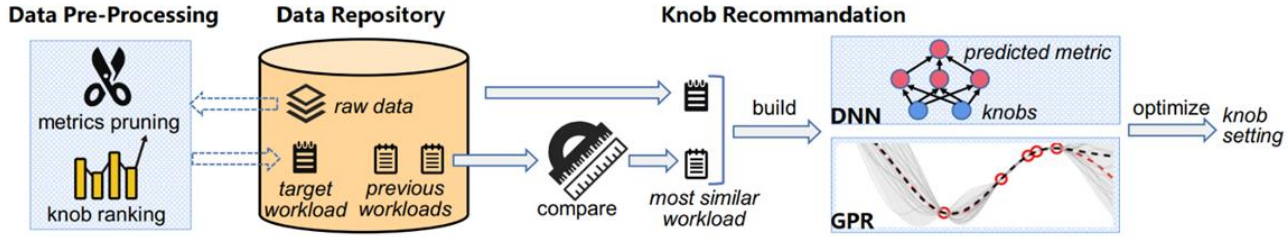


Figure 6: GPR/DNN Tuning Pipeline – The raw data for each previous workload is aggregated and compared with the target workload. Data from the most similar previous workload is then merged with the target workload data to build a machine learning model (DNN or GPR). Finally, the algorithm recommends the next configuration to run by optimizing the model.

- (RL) DDPG and DDPG++

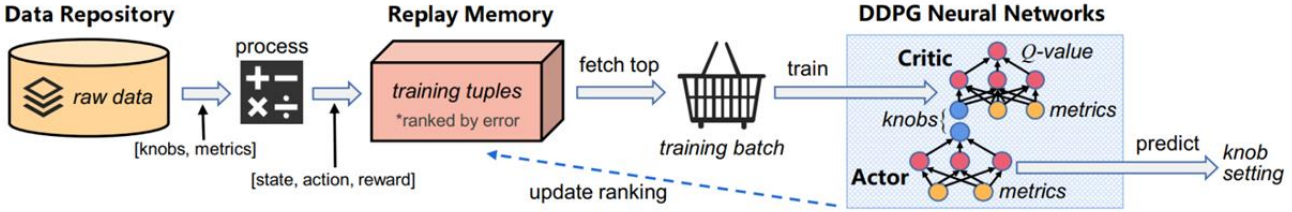


Figure 7: DDPG Tuning Pipeline – The raw data is converted to states, actions, and rewards and then inserted into the replay memory. The tuples in the replay memory are ranked by the error of the predicted Q-value. In the training process, the critic and actor are updated with a batch of the top tuples. After training, the prediction error in the replay memory is updated, and the actor recommends the next configuration to run.

Evaluation: Basic Strategies

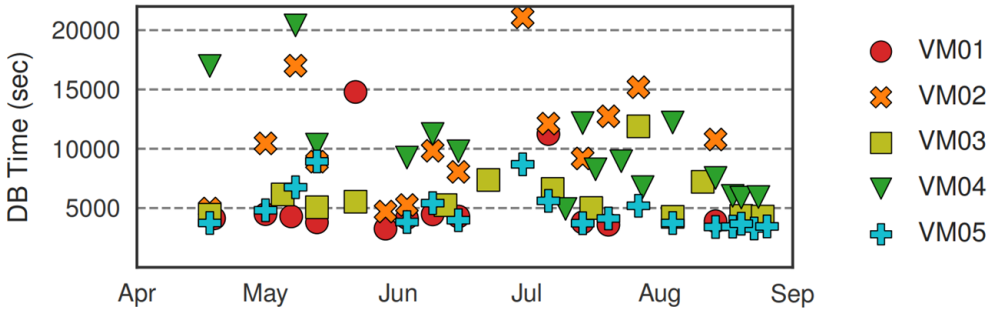


Figure 8: Performance Variability – Performance for the TicketTracker workload using the default configuration on multiple VMs over six months.

- Baseline: Latin Hypercude Sampling (LHS)
- Target metric: DB Time
- Initial evaluation of the variability in the performance measurements for SG's environment:
  - Oracle DBMS on multiple VMs on the same physical machine (shared storage)
  - measured the performance of multiple VMs once a week over six months

Evaluation: Basic Strategies

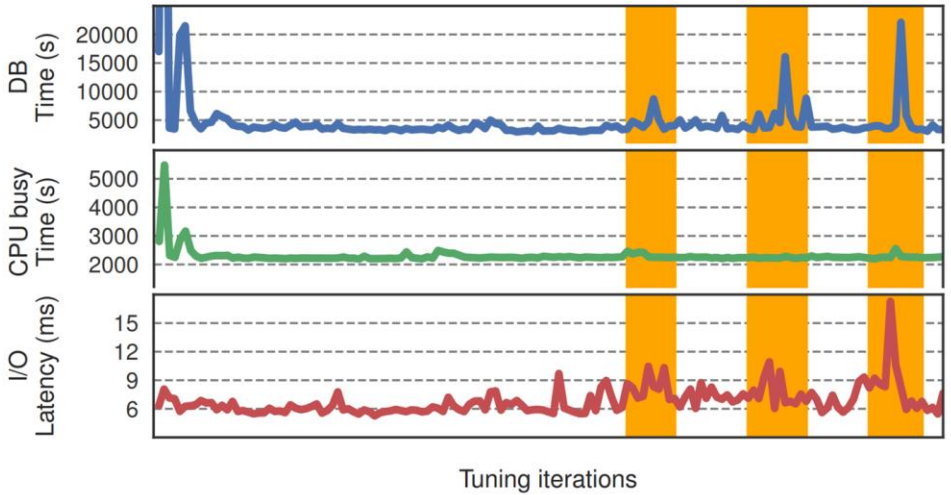


Figure 9: Effect of I/O Latency Spikes – Runtime measurements of DBMS performance with CPU utilization and I/O latency.

- Baseline: Latin Hypercude Sampling (LHS)
- Target metric: DB Time
- Initial evaluation of the variability in the performance measurements for SG's environment:
  - Measure DBMS's performance with CPU time and I/O latency for one VM during a tuning session

Evaluation: Basic Strategies

- Baseline: Latin Hypercude Sampling (LHS)
- Target metric: DB Time
- Initial evaluation of the variability in the performance measurements for SG's environment

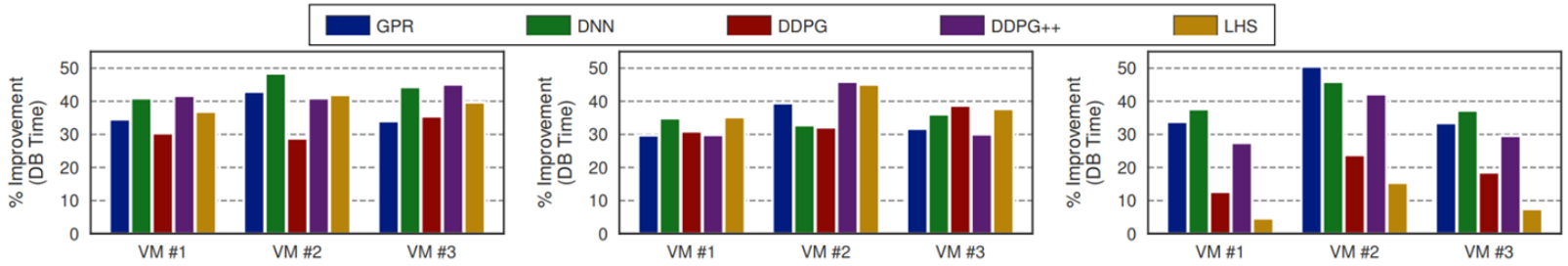
Given performance variability :

=> 3 tuning sessions on 3 VMs

=> for evaluating a tuned configuration from each tuning session, run the workload with it on DBMSs on 3 VMs for 3 times (9 times in total)

- overall performance of the configuration => average across all executions

Evaluation: Tuning Knobs Selected by human DBAs



(a) 10 Knobs (b) 20 Knobs (c) 40 Knobs

Figure 10: Tuning Knobs Selected by DBA (Per VM) – The performance improvement of the best configuration per algorithm running on separate VMs relative to the performance of the SG default configuration measured at the beginning of the tuning session.

- ⇧ performances of best configuration **per algorithm** on different VMs
- ⇨ best and worst overall performance of best configurations from 3 tuning sessions

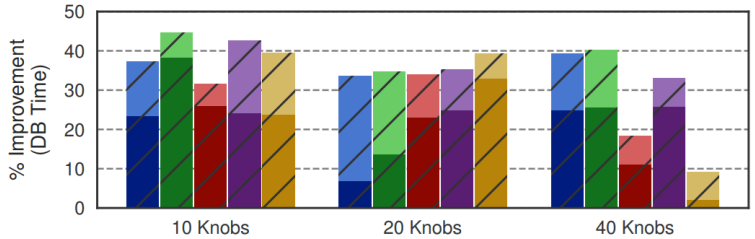


Figure 11: Tuning Knobs Selected by DBA – Performance measurements for 10, 20, and 40 knob configurations for the TicketTracker workload. The shading on each bar indicates the minimum and maximum performance of the optimized configurations from three tuning sessions.

Evaluation: Tuning Knobs Ranked by OtterTune

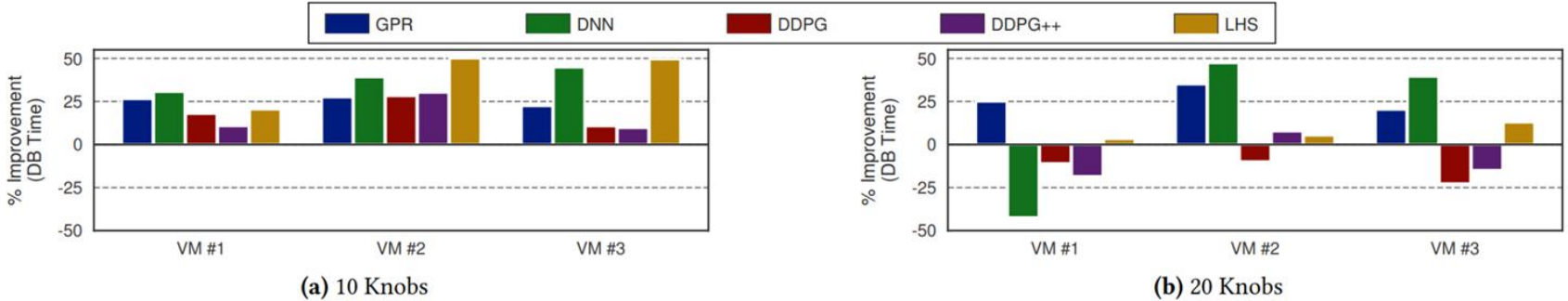


Figure 12: Tuning Knobs Ranked by OtterTune (Per VM) – The performance improvement of the best configuration per algorithm running on separate VMs relative to the performance of the SG default configuration measured at the beginning of the tuning session.

- ⇧ performances of best configuration **per algorithm** on different VMs
- ⇨ best and worst overall performance of best configuration **per tuning session**

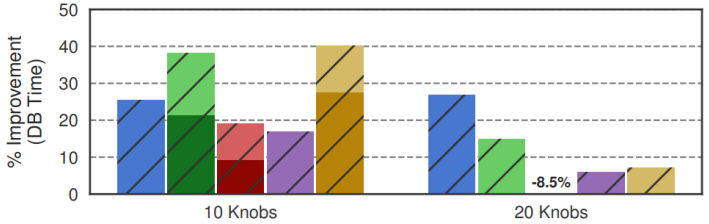
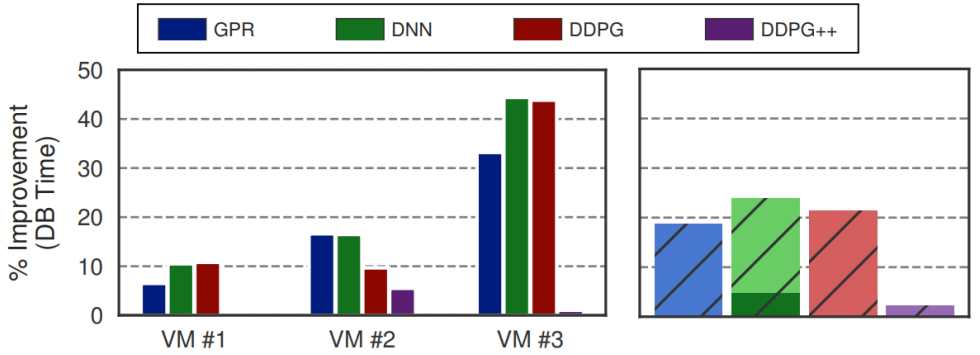


Figure 13: Tuning Knobs Ranked by OtterTune – Performance measurements for the ML algorithm configurations using 10 and 20 knobs selected by OtterTune's Lasso ranking algorithm. The shading on each bar indicates the minimum and maximum performance of the optimized configurations from three tuning sessions.

Evaluation: Adaptability to Different Workload

Firstly tune for TPC-C workload, and then use trained model for tuning towards TicketTracker workload trace segment



(a) Performance per VM **(b)** Performance Summary

Figure 14: Adaptability to Different Workloads – Comparison when applying the model trained on TPC-C data to the TicketTracker workload.

Knob Name	Default	Best Observed
DB_CACHE_SIZE	4 GB	20–30 GB
DB_32K_CACHE_SIZE	10 GB	15 GB
OPTIMIZER_FEATURES_ENABLE	v11.2.0.4	v12.2.0.1

Table 2: Most Important Knobs – The three most important knobs for the TicketTracker workload with their default and best observed values.

Criticism

Novelty/Advantages

- Production-level environments for evaluating SOTA methods
- Provide valuable environment results for reference in later works

Limitations

- Poor Reproducibility (almost infeasible for small research groups)
- Limited Scope / Not sufficient diversity of workloads,
- Many trivial observations in experiment part consisting of a hard-to-read long text (information scattered) and no good summary of single parts of experiments
- Lack in-depth analysis for certain algorithms

Discussion