[46] Weibo Liu, Yan Jin, and Mark Price. 2017. A new improved NEH heuristic for permutation flowshop scheduling problems. *International Journal of Production Economics* 193 (2017), 21–30.

[47] Yi Lu, Xiangyao Yu, Lei Cao, and Samuel Madden. 2020. Aria: A Fast and Practical Deterministic OLTP Database. *Proc. VLDB Endow.* 13, 12 (jul 2020), 2047–2060.

[48] Walther Maldonado, Patrick Marlier, Pascal Felber, Adi Suissa, Danny Hendler, Alexandra Fedorova, Julia L Lawall, and Gilles Muller. 2010. Scheduling support for transactional memory contention management. *ACM Sigplan Notices* 45, 5 (2010), 79–90.

[49] Virendra J. Marathe and Michael L. Scott. 2004. A Qualitative Survey of Modern Software Transactional Memory Systems.

[50] Muhammad Nawaz, E Emory Enscore Jr, and Inyong Ham. 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega* 11, 1 (1983), 91–95.

[51] Thomas Neumann, Tobias Mühlbauer, and Alfons Kemper. 2015. Fast serializable multi-version concurrency control for main-memory database systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 677–689.

[52] Gultekin Ozsoyoglu and Richard T Snodgrass. 1995. Temporal and real-time databases: A survey. *IEEE Transactions on Knowledge and Data Engineering* 7, 4 (1995), 513–532.

[53] Christos H. Papadimitriou. 1979. The Serializability of Concurrent Database Updates. *J. ACM* 26, 4, 631–653. https://doi.org/10.1145/322154.322158

[54] Andrew Pavlo, Carlo Curino, and Stanley Zdonik. 2012. Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 61–72.

[55] Miroslav Popovic, Branislav Kordic, and Ilija Basicevic. 2017. Transaction scheduling for Software Transactional Memory. In *2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. 191–195.

[56] Guna Prasaad, Alvin Cheung, and Dan Suciu. 2020. Handling highly contended OLTP workloads using fast dynamic partitioning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 527–542.

[57] Thamir M Qadah and Mohammad Sadoghi. 2018. Quecc: A queue-oriented, control-free concurrency architecture. In *Proceedings of the 19th International Middleware Conference*. 13–25.

[58] Dai Qin, Angela Demke Brown, and Ashvin Goel. 2021. Caracal: Contention management with deterministic concurrency control. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 180–194.

[59] Krithi Ramamritham. 1993. Real-time databases. *Distributed and parallel databases* 1 (1993), 199–226.

[60] Rubén Ruiz, Quan-Ke Pan, and Bahman Naderi. 2019. Iterated Greedy methods for the distributed permutation flowshop scheduling problem. *Omega* 83 (2019), 213–222.

[61] Rubén Ruiz and Thomas Stützle. 2007. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European journal of operational research* 177, 3 (2007), 2033–2049.

[62] Mohammad Saidi-Mehrabad and Parviz Fattahi. 2007. Flexible job shop scheduling with tabu search algorithms. *International Journal of Advanced Manufacturing Technology* 32, 5-6 (2007), 563–570.

[63] Mohammad Saidi-Mehrabad and Parviz Fattahi. 2007. Flexible job shop scheduling with tabu search algorithms. *International Journal of Advanced Manufacturing Technology* 32, 5-6 (2007), 563–570.

[64] M Saqlain, S Ali, and JY Lee. 2023. A Monte-Carlo tree search algorithm for the flexible job-shop scheduling in manufacturing systems. *Flexible Services and Manufacturing Journal* 35, 2 (2023), 548–571.

[65] William N Scherer III and Michael L Scott. 2005. Advanced contention management for dynamic software transactional memory. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 240–248.

[66] William N Scherer III and Michael L Scott. 2005. Advanced contention management for dynamic software transactional memory. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*. 240–248.

[67] DY Sha and Cheng-Yu Hsu. 2006. A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering* 51, 4 (2006), 791–808.

[68] Gaurav Sharma, Ravi R Mazumdar, and Ness B Shroff. 2006. On the complexity of scheduling in wireless networks. In *Proceedings of the 12th annual international conference on Mobile computing and networking*. 227–238.

[69] Liji Shen, Stéphane Dauzère-Pérès, and Janis S Neufeld. 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European journal of operational research* 265, 2 (2018), 503–516.

[70] Yangjun Sheng, Anthony Tomasic, Tieying Zhang, and Andrew Pavlo. 2019. Scheduling OLTP Transactions via Learned Abort Prediction. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management* (Amsterdam, Netherlands) *(aiDM '19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 8 pages.

[71] John A Stankovic, Marco Spuri, Krithi Ramamritham, and Giorgio Buttazzo. 1998. *Deadline scheduling for real-time systems: EDF and related algorithms*. Vol. 460. Springer Science & Business Media.

[72] Chunzhi Su, Natacha Crooks, Cong Ding, Lorenzo Alvisi, and Chao Xie. 2017. Bringing modular concurrency control to the next level. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 283–297.

[73] Chunzhi Su, Natacha Crooks, Cong Ding, Lorenzo Alvisi, and Chao Xie. 2017. Bringing modular concurrency control to the next level. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 283–297.

[74] The H-Store team. 2013. SmallBank Benchmark. http://hstore.cs.brown.edu/documentation/deployment/benchmarks/smallbank/

[75] Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J. Abadi. 2012. Calvin: Fast Distributed Transactions for Partitioned Database Systems *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 1–12.

[76] Boyu Tian, Jiamin Huang, Barzan Mozafari, and Grant Schoenebeck. 2018. Contention-aware lock scheduling for transactional databases. *Proceedings of the VLDB Endowment* 11, 5 (2018), 648–662.

[77] Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. 2013. Speedy transactions in multicore in-memory databases. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. 18–32.

[78] Özgür Ulusoy and Geneva G Belford. 1993. Real-time transaction scheduling in database systems. *Information Systems* 18, 8 (1993), 559–580.

[79] Peter JM Van Laarhoven, Emile HL Aarts, and Jan Karel Lenstra. 1992. Job shop scheduling by simulated annealing. *Operations research* 40, 1 (1992), 113–125.

[80] Jia-Chen Wang, Ding Ding, Huan Wang, Conrad Christensen, Zhaoguo Wang, Haibo Chen, and Jinyang Li. 2021. Polyjuice: High-Performance Transactions via Learned Concurrency Control. In *OSDI*. 198–216.

[81] Ling Wang and Da-Zhong Zheng. 2001. An effective hybrid optimization strategy for job-shop scheduling problems. *Computers & Operations Research* 28, 6 (2001), 585–596.

[82] Zhaoguo Wang, Shuai Mu, Yang Cui, Han Yi, Haibo Chen, and Jinyang Li. 2016. Scaling multicore databases via constrained parallel execution. In *Proceedings of the 2016 International Conference on Management of Data*. 1643–1658.

[83] Xinzhou Wu, Rayadurgam Srikant, and James R Perkins. 2007. Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. *IEEE transactions on mobile computing* 6, 6 (2007), 595–605.

[84] Jin Xie, Liang Gao, Kunkun Peng, Xinyu Li, and Haoran Li. 2019. Review on flexible job shop scheduling. *IET collaborative intelligent manufacturing* 1, 3 (2019), 67–77.

[85] Ming Xiong, Qiong Wang, and Krithi Ramamritham. 2008. On earliest deadline first scheduling for temporal consistency maintenance. *Real-Time Systems* 40 (2008), 208–237.

[86] Cong Yan and Alvin Cheung. 2016. Leveraging Lock Contention to Improve OLTP Application Performance. *Proc. VLDB Endow.* 9, 5 (jan 2016), 444–455.

[87] Linguan Yang, Xinan Yan, and Bernard Wong. 2022. Natto: Providing Distributed Transaction Prioritization for High-Contention Workloads. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 715–729.

[88] Chenhao Ye, Wuh-Chwen Hwang, Keren Chen, and Xiangyao Yu. 2023. Polaris: Enabling Transaction Priority in Optimistic Concurrency Control. *Proc. ACM Manag. Data* 1, 1, Article 44 (may 2023), 24 pages.

[89] Xiangyao Yu, Andrew Pavlo, Daniel Sanchez, and Srinivas Devadas. 2016. Tictoc: Time traveling optimistic concurrency control. In *Proceedings of the 2016 International Conference on Management of Data*. 1629–1642.

[90] Erfan Zamanian, Julian Shun, Carsten Binnig, and Tim Kraska. 2020. Chiller: Contention-centric transaction execution and data partitioning for modern networks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 511–526.

[91] Rui Zhang and Cheng Wu. 2010. A hybrid immune simulated annealing algorithm for the job shop scheduling problem. *Applied Soft Computing* 10, 1 (2010), 79–89.

A APPENDIX

We model transaction scheduling by adding constraints to the job-shop scheduling optimization problem with an objective of minimizing makespan. From the original formulation [62], we have jobs as transactions, tasks as operations, and machines as data items. We assume 2PL is used to ensure serializability. The new constraints we add to model transactions are in red.

Parameters:

n: Number of txns

m: Number of keys

$O_{j,h}$ for $h = 1, ..., h_j$: hth operation of txn j

$O_{i,j,h}$: Operation $O_{j,h}$ on key i

$p_{i,j,h}$: Processing time of $O_{i,j,h}$
M: A large number

$$y_{i,j,h} = \begin{cases} 1 & \text{if } O_{j,h} \text{ can be performed on key } i \\ 0 & \text{otherwise} \end{cases}$$

$$r_{j,h} = \begin{cases} 1 & \text{if } O_{j,h} \text{ is a read} \\ 0 & \text{otherwise} \end{cases}$$

$$u_{j,h} = \begin{cases} 1 & \text{if } O_{j,h} \text{ is a read in a txn with a write to the same key} \\ 0 & \text{otherwise} \end{cases}$$

Decision variables:

$$x_{i,j,h,k,l} = \begin{cases} 1 & \text{if } O_{i,j,h} \text{ immediately precedes } O_{i,k,l} \\ 0 & \text{otherwise} \end{cases}$$

$t_{j,h}$: Start time of the processing of operation $O_{j,h}$
$f_{j,h}$: Finish time of the processing of operation $O_{j,h}$
$s_{i,j,h}$: Release time of shared lock for key i based on ongoing txns
C_{max}: Makespan of a schedule

Optimization problem:
Min C_{max}
s.t.
(1) $t_{j,h} + y_{i,j,h} * p_{i,j,h} \leq f_{j,h}, \forall i = 1, ..., m; j = 1, ..., n; h = 1, ..., h_j$
(2) $t_{j,h} + p_{i,j,h} \leq t_{j,h+1}, \forall i = 1, ..., m; j = 1, ..., n; h = 1, ..., h_j$
(3) $f_{j,h} \leq f_j, \forall h = 1, ..., h_j$
(4) $f_j \leq C_{max}$
(5) $f_j - r_{j,h} * r_{k,l} * M \leq t_{k,l} + (1 - x_{i,j,h,k,l}) * M, \forall i = 1, ..., m; j = 0, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k; j \neq k$
(6) $t_{j,h} + p_{i,j,h} - r_{j,h} * r_{k,l} * M \leq t_{k,l} + (1 - x_{i,j,h,k,l}) * M, \forall i = 1, ..., m; j = 0, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k$
(7) $f_j - (1 - r_{j,h}) * M \leq s_{i,k,l} + (1 - x_{i,j,h,k,l}) * M, \forall i = 1, ..., m; j = 0, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k; j \neq k$
(8) $s_{i,j,h} - (1 - r_{j,h}) * (1 - r_{k,l}) * M \leq s_{i,k,l} + (1 - x_{i,j,h,k,l}) * M, \forall i = 1, ..., m; j = 0, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k; j \neq k$
(9) $s_{i,j,h} - r_{k,l} * M \leq t_{k,l} + (1 - x_{i,j,h,k,l}) * M, \forall i = 1, ..., m; j = 0, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k; j \neq k$
(10) $\sum_i y_{i,j,h} = 1, \forall j = 0, 1, ..., n; h = 1, ..., h_j$
(11) $\sum_j \sum_h x_{i,j,h,k,l} = y_{i,k,l}, \forall i = 1, ..., m; k = 1, ..., n; l = 1, ..., h_k$
(12) $\sum_k \sum_l x_{i,j,h,k,l} = y_{i,j,h}, \forall i = 1, ..., m; j = 0, 1, ..., n; h = 1, ..., h_j$
(13) $x_{i,j,h,j,h} = 0, \forall i = 1, ..., m; j = 0, 1, ..., n; h = 1, ..., h_j$
(14) $t_{j,h} \geq 0, \forall j = 0, 1, ..., n; h = 1, ..., h_j$
(15) $f_{j,h} \geq 0, \forall j = 0, 1, ..., n; h = 1, ..., h_j$
(16) $y_{i,j,h} \in \{0, 1\}, \forall i = 1, ..., m; j = 0, 1, ..., n; h = 1, ..., h_j$
(17) $x_{i,j,h,k,l} \in \{0, 1\}, \forall i = 1, ..., m; j = 0, 1, ..., n; h = 1, ..., h_j; k = 1, ..., n; l = 1, ..., h_k$

We describe the purpose of each constraint. (1) defines the processing time of each operation. (2) requires that operations in each transaction must follow a specified partial order. (3) ensures that no locks are released until the end of each transaction. (4) defines the makespan. (5) enables shared and read-for-update locks. (6) ensures only one operation can execute at a time on a key unless both are reads. (7) defines the release time of if this operation is a read. (8) defines the release time of a lock if the next operation is a read. (9) ensures a write can only occur after all read locks are released. (10) requires each operation to execute on only one key. (11) and (12) define circular permutations of operations on each machine. They eliminate alternative operations that are excluded in a final schedule. (11) selects one operation $O_{i,j,h}$ that immediately precedes a scheduled alternative operation $O_{i,k,l}$. (12) selects one operation $O_{i,k,l}$that immediately follows a scheduled alternative operation$O_{i,j,h}$. A circular permutation of operations on a machine yields a scheduled sequence of the operations on the same machine. (13) requires that each operation only occurs once. (14) and (15) ensure that operation and transaction execution times cannot be negative. (16) and (17) ensure all operations in all transactions are scheduled.