Vol.2017-OS-141 No.3
2017/7/26

並列WALを適用したTicTocの評価

中村 泰大[1,a)]　川島 英之[2,b)]　建部 修見[2,c)]

概要：我々はこれまでに並行実行制御手法 TicToc に並列ログ先行書き込み手法 P-WAL の適用を提案し，提案手法をマイクロベンチマークで実験的に評価した．評価の結果，ログ書き込み先に不揮発性メモリを想定した環境では，データベースオブジェクトへのロックを早期解放する ELR と，ログレコードをストレージへ一括転送するグループコミットが性能劣化を招くという結果を得た．しかし，一般的なワークロードでも性能劣化が発生するのかは示していなかった．本論文では，電子商取引を模した一般的なワークロードである TPC-C を用いて提案手法のスループットを実験的に評価する．コア数の少ないマルチコア環境でのみ行ったため性能劣化はおこらなかったが，提案手法は Warehouse 数 10 で最大 250 万 transaction/sec を達成した．

1. はじめに

トランザクション処理システムは並行実行制御とログ先行書き込みから構成される．並行実行制御は複数のトランザクションを同時に実行しても一貫性を保った結果が得られることを保証することである．ログ先行書き込みはトランザクションの結果が永続化されることを保証することである．

並行実行制御手法については Silo [1] をはじめとして，MOCC [2] など多くの手法が次々に提案されている．それらの手法の 1 つに TicToc [3] がある．TicToc はタイムスタンプをベースとしたスケーラブルな並行実行制御手法である．既存のタイムスタンプベースの並行実行制御手法はタイムスタンプ発行処理に排他制御が必要なため，マルチコア環境では高い性能を示せない．TicToc はタイムスタンプ発行処理における排他制御を排除することにより，マルチコア環境で性能がスケールアップすると報告されている [3].

ログ先行書き込みについては Aether [4] をはじめとして，passive group commit [5]，P-WAL [6] が提案されている．いずれの提案においても高速化のために，データロックを早期に解放する Early Lock Release (ELR) と，複数のログレコードをまとめて永続化するグループコミットを用いている．これらの最適化手法によって，データロックの競合を緩和し，ログレコード永続化の I/O コストを削減している．

我々はログ先行書き込みの最適化手法を TicToc と組み合わせても有効か検証するため，TicToc への P-WAL の適用を提案した．そして提案手法の性能をマイクロベンチマークを用いて評価し，不揮発性メモリ・メニーコア環境においては ELR やグループコミットが性能劣化を招くことを示した [7]．本論文では一般的なワークロードである TPC-C New-Order トランザクションを用いて提案手法のスループットを評価する．TPC-C は電子商取引を模擬した OLTP ワークロードであり，多くの既存研究において性能比較に関する標準ベンチマークに使われている [8].

本論文の構成は以下の通りである．2 章では背景として TPC-C を述べる．3 章では提案手法について簡単に述べる．4 章では評価に用いたプログラムの設計と実装を述べる．5 章では TPC-C を用いて提案手法の評価を行う．6 章では関連研究を述べ，7 章では結論を述べる．

2. TPC-C

TPC-C [8] は業界団体 Transaction Processing Performance Council によって作成された OLTP ワークロードである．ワークロードは電子商取引を模して作られているため，実アプリケーションに即した性能評価のために Silo [1] や FOEDUS [9]，TicToc [3] の性能評価に用いられている．

TPC-C で用いるテーブルの概要を 図 1 に示す．データベース全体は 1 つの会社 (Company) を表している．会社はいくつかの倉庫 (Warehouse) を管理し，各倉庫は担当する地区 (District) を 10 区画，各地区は顧客 (Customer) を 30000 万人抱えている．Warehouse の数はスケールファク

1 筑波大学大学院 システム情報工学研究科
2 筑波大学 計算科学研究センター
a) nakamura@hpcs.cs.tsukuba.ac.jp
b) kawasima@cs.tsukuba.ac.jp
c) tatebe@cs.tsukuba.ac.jp

ターであり，ワークロードの実行時にパラメーターとして数を指定する．図 1 においてテーブルとしてデータベースが持つ項目は Warehouse， District， Customer である．また，図に書かれていないテーブルとして，Item， Stock，Order， Order-Line， New-Order，History がある．

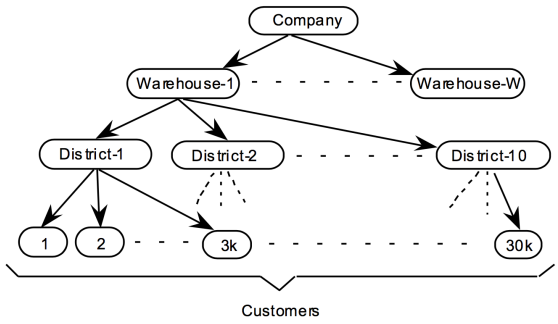


図 1: TPC-C のテーブル構成の概念図 ([8] より引用)

このモデル上で商品の注文や在庫の管理，商品発送の管理を行うために 5 つのトランザクションを用いる．1 つ目は New-Order トランザクションで，顧客からの注文を受けて New-Order テーブルに未発送の注文を生成する．加えて，発送に必要な分の在庫の消費や，顧客の割引額や倉庫と地区の税率を加味した合計金額の計算を行う．2 つ目は Payment トランザクションで，支払いに関する処理を行う．3 つ目は Order-Status トランザクションで，テーブルの状態は変更せずに注文の状態を参照する．4 つ目は Delivery トランザクションで，New-Order テーブルにある未発送商品の発送手続きを行う．5 つ目は Stock-Level トランザクションで，商品の在庫量を確認する．ワークロードにおいて，これらのトランザクションの実行比率は一定の値に設定される．また，一部のトランザクションのみをベンチマークに用いるという使われ方もある．

3. 提案手法

トランザクション処理システムは並行実行制御とログ先行書き込みから構成される．TicToc の原論文 [3] では Silo のような並列ログ先行書き込みが容易に実装可能だと記述されているが，具体的な検討はされていない．そこで我々はこれまでに TicToc と P-WAL を結合し，TicToc に永続化処理を施す 4 つの手法を提案した [7]．提案手法は，NLR/ELR とグループコミットをそれぞれ組み合わせた TT (NLR) + PWAL (NoGroup)， TT (NLR) + PWAL (Group)， TT (ELR) + PWAL (NoGroup)， TT (ELR) + PWAL (Group) である．

P-WAL は ELR の利用を前提としているため，共有カウンタを用いて Log Sequence Number (LSN) を各ログレコードに発行し，通知制御に利用する．ELR はログを永続化する前にデータロックを解放するため，トランザクションが依存するデータ全てが永続化されている保証はない．そこで，LSN でログレコードに全順序をつけ，通知したいトランザクションが生成したログよりも前のログレコードが永続化されているか確認する．依存する全てのログの永続化されていること確認すると通知制御によりコミットをクライアントに通知する．TT (ELR) + PWAL (NoGroup)，TT (ELR) + PWAL (Group) は ELR を用いているため，Sequence Number を発行している．TT (NLR) + PWAL (NoGroup)， TT (NLR) + PWAL (Group) は Nomal Lock Release (NLR) を用いている．これら 2 手法はログの永続化を行ってからデータロックを解放する．TicToc ではロックされているタプルを Read フェーズで読み込まない．したがって，ワーカースレッドが Read フェーズで読み込むデータベースオブジェクトの値は，永続化されていることを保証できる．よって，データロックを解放した時に，トランザクションが依存する全てのログが永続化されていることを保証できるため，通知制御と Sequence Number の発行は不要となる．

TT (NLR) + PWAL (Group) と TT (ELR) + PWAL (Group) ではグループコミットを行う．NLR にグループコミットを適用すると全順序に従ってデータロックを獲得できないため，容易にデッドロックしてしまう．そこで，TT (NLR) + PWAL (Group) はデッドロックを回避するために，グループ数に達していなくてもログの永続化とデータロックの解放を行う．この手法でログの永続化を行うのは，グループ数に達した時，データロックの獲得に失敗した時，Read フェーズ時にデータロックの解放待ちが発生した時である．

4. 設計と実装

設計したプロトタイプシステムのモジュール構成を図 2 に示す．プロトタイムシステムはストアドプロシージャー方式を採用した．Worker はトランザクションを実行するモジュールで, 各 Worker は実行する命令列 Procedure を持つ．Procedure は，Transaction manager を介して Database とのやりとりすることで実行される．Transaction manager はトランザクション処理全体の制御を行う．各 Worker は個別に WalBuffer を持つ．WalBuffer は Transaction manager からログレコードを受け取って蓄積し，Transaction manager からの制御命令によって蓄積したログレコードを対応する WAL File に書き込んでログレコードを永続化する．WalBuffer はそれぞれ別の WAL File に対応づけられている．

プロトタイプシステムの実装にはプログラミング言語 C++を用いた．プロシージャはプロトタイプシステムにハードコーディングすることで登録した．テーブルのカラム定義は C++の構造体で記述し，各テーブルは構造体の配列で表現した．なお，配列は固定長とし，新たな要素の

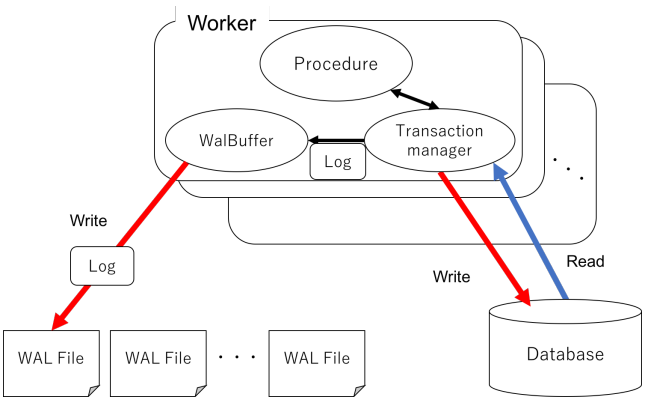


図 2: プロトタイプシステムのモジュール構成

挿入は行われないこととした．つまり，各要素へはキーを用いて $O(1)$ でアクセス可能である．プロシージャに与える引数は乱数を用いて実行した．乱数には高速な擬似乱数である xorshift128+ [10] を用いた．TicToc と P-WAL を結合した 4 つの手法はこれまでに提案したプロトコル [7] に則って実装した．ELR を適用する場合の通知制御は，本実装では行っていない．

WAL File への書き込みが行われる時には，実際のストレージに書き込みを行うのではなく，不揮発性メモリへの書き込みを想定した固定長のレイテンシを挿入する．この待機処理には CPU の Time Stamp Counter (TSC) を用いた．TSC を監視し，指定した待機時間分だけ CPU クロックが進むのを待つ．他にレイテンシを挿入する方法として nanosleep() が考えられるが，コンテキストスイッチに時間がかかってしまうため利用しなかった．

5. 評価

設計したトランザクション処理手法のスループットを評価する．評価はマルチコア環境で行う．NLR/ELR とグループコミットをそれぞれ組み合わせた TT (NLR) + PWAL (NoGroup)， TT (NLR) + PWAL (Group)， TT (ELR) + PWAL (NoGroup)， TT (ELR) + PWAL (Group) の 4 手法を実験により性能の測定を行い，評価する．

この実験では，各ワーカースレッドを 10 秒間動作させ何回トランザクションを実行したかを計測し，スループットを算出する．ワーカースレッドの開始及び終了のタイミングは，ワーカースレッドとは別のスレッドで管理される．WAL ファイルの書き込み先は不揮発性メモリを想定し，WAL ファイルへの書き込みが行われるタイミングで固定長のレイテンシを挿入する．プロトタイプシステムでグループコミット数を 16 とした．ワークロードには TPC-C New-Order トランザクションを用いる．また，トランザクションはインサートが発生しないように変更を行った．あくまでインサートをしていないだけであり，インサートに必要な情報の収集と生成は行なっている．

5.1 マルチコア環境

マルチコア環境の実験では，Warehouse および WAL の書き込みレイテンシ (NVM latency) を変化させ，スループットを測定した．Warehouse は 1, 4, 10 と変化させ，WAL 書き込みレイテンシは 500 ns, 5 μs, 50 μs と変化させた．実験環境を表 1 に示す．この計算機はプロセッサを 2 ソケット搭載し，計 24 コアを有する．

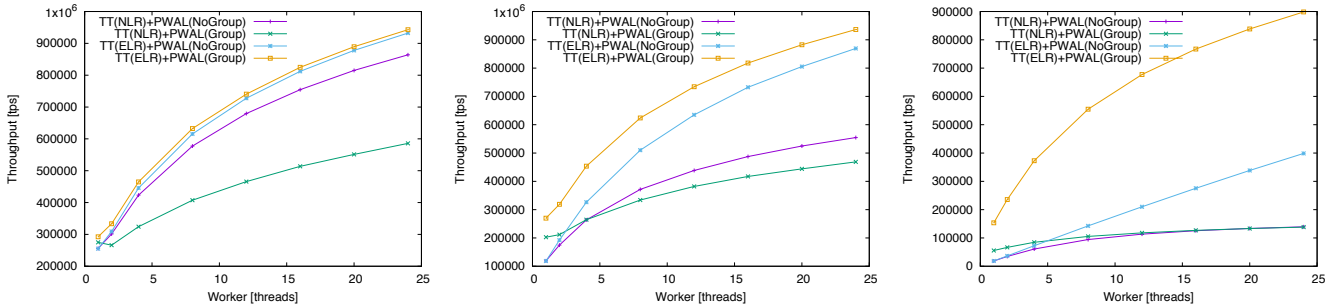
表 1: マルチコアの評価環境

プロセッサ	Intel(R) Xeon(R) CPU E5-2695 v2 @ 2.40GHz
コア	24 cores
メモリ	64GB
OS	CentOS release 6.8 (Final)

1 Warehouse での実験結果を図 3 に示す．全てのケースで TT (ELR) + PWAL (Group) が最も良い性能を示している．これは，4 手法の中で TT (ELR) + PWAL (Group) のロック保有期間およびログ書き込み時間が最も短いことが原因であると考えられる．ELR の 2 手法を比較すると，NVM latency が増加するにつれて性能差が大きくなることがわかる．NVM latency が 500 ns の場合では TT (ELR) + PWAL (Group) と TT (ELR) + PWAL (NoGroup) の性能差はほとんどないが，NVM latency が 50 μs の場合では TT (ELR) + PWAL (Group) は TT (ELR) + PWAL (NoGroup) と比べて 2.25 倍スループットが良い．よって，WAL ファイルへの書き込みレイテンシが大きいほどグループコミットによる高性能化が達成され，500 ns ほどの小さなレイテンシではグループコミットではほとんど高性能化を得られないということがわかる．

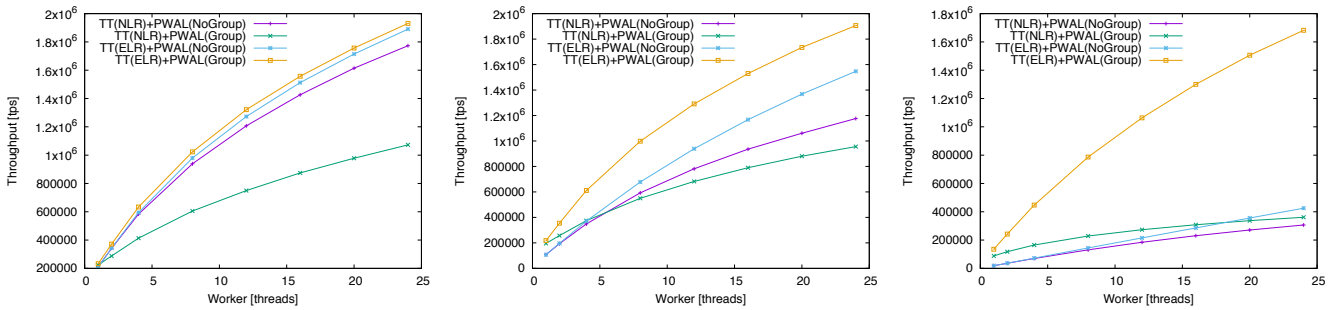
4 Warehouses および 10 Warehouses での実験結果をそれぞれ図 4 と図 5 に示す．これらの結果と 1 Warehouse とを比較すると，ワーカー数の変化によるスループットの変化の傾向はあまり変わっていない．しかし，Warehouse が増加するに従って分散度が高まるため全体的にスループットが向上している．NVM latency が 500 ns である場合に注目すると，Warehouse 数が増加するに従って TT (NLR) + PWAL (NoGroup) のスループットが ELR のスループットに近づいている．これは分散度が高まったことでデータロックの衝突が減少したことが原因であると考えられる．

マルチコア環境でマイクロベンチマークを実行した時に ELR による性能劣化が発生していた [7] が，電子商取引を模したワークロードである TPC-C を実行しても ELR による性能劣化は起こらなかった．よって，一般的なトランザクションを実行する範囲において ELR による性能劣化は発生しないと思われる．



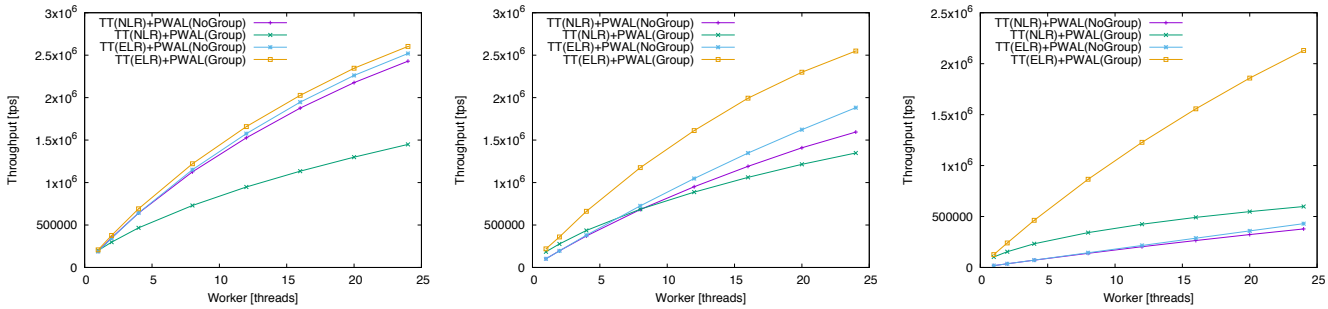
(a) NVM latency = 500 ns **(b)** NVM latency = 5 μs **(c)** NVM latency = 50 μs

図 3: TPC-C (1 Warehouse)：各提案方式のスループット



(a) NVM latency = 500 ns **(b)** NVM latency = 5 μs **(c)** NVM latency = 50 μs

図 4: TPC-C (4 Warehouses)：各提案方式のスループット



(a) NVM latency = 500 ns **(b)** NVM latency = 5 μs **(c)** NVM latency = 50 μs

図 5: TPC-C (10 Warehouses)：各提案方式のスループット

6. 関連研究

6.1 共有メモリを想定した研究

FOEDUS [9] は不揮発性メモリを前提として設計された高性能トランザクション処理システムである．範囲検索を高性能化するために Masstree [11] を不揮発性メモリに対応させた MasterTree が新規に提案されており，Silo [1] に基づくコミットプロトコルならびに並行実行制御プロトコルが提案されている．FOEDUS の応用としてグラフ処理システム Janus [12] が考案されている．また，温度制御などを用いて楽観的並行実行制御と悲観的並行実行制御を組み合わせた高性能並行実行制御方式 MOCC [13] が FOEDUS 上で実装されており，優れた性能を示している．

Cicada [14] は楽観的，複数版，複数時計に基づく高性能トランザクション処理方式である．これは外部一貫性を破る緩和同期を用いることで，MOCC を上回る性能を示す．

本研究で取り上げた TicToc [3] はタイムスタンプをベースにした並行実行制御であり，Silo に比べて高い性能を示すことが報告されている．一方，本研究で述べたようにメニーコア環境での性能が不明であり，さらにログ先行書き込みとの結合方式については検討が未熟だった．

6.2 ログ先行書き込みに関する研究

Wang ら [5] は複数の WAL バッファを NVM 内に用意し，その中へログレコードを並列的に転送する手法である passive group commit を提案している．Passive group commit は NVM 上に WAL バッファを置き, ログを DRAM を経由せずに直接 NVM に書き込む．Passive group commit は ELR であり，各トランザクションがコミット条件を満たしたかどうかをまとめて確認する専用のデーモンを用意する．また，passive group commit には GSN(Global Sequence Number) と呼ばれるシーケンス番号が必要である．

P-WAL [6] は並列にログ先行書き込みを実行するプロトコルであり，本論文でも詳述した．P-WAL は ioDrive のようにストレージデバイスが並列性を内包する際に優れたスループットを示す．

7. 結論

本論文はタイムスタンプベースの並行実行制御手法 TicToc に着目した．TicToc に P-WAL を適用した提案手法がマイクロベンチマークを用いた評価で，ELR とグループコミットが不揮発性メモリ・メニーコア環境では性能劣化を招くことは我々のこれまでの研究で示している [7] .

一般的なワークロードでも性能劣化が起こるのかは不明であった．そこで，我々は TPC-C NewOrder トランザクションを一部改変したベンチマークを作成し，提案手法のスループットを実験的に評価した．

マルチコア環境においては，従来の最適化手法である ELR とグループコミットを用いた手法 (TT (ELR) + PWAL (Group)) がどの条件においても最も良い性能を示した．また，ログの書き込みレイテンシが大きくなるほど，ELR とグループコミットの最適化により，TT (ELR) + PWAL (Group) が他の手法に比べて良い性能を示した．よって，マルチコア環境においては従来の最適化手法が最も良い性能を発揮すると本論文は結論する．メニーコア環境においての性能評価はまだ行っていないため，今後の課題とする．

謝辞 本研究の一部は，JST CREST「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」，JST CREST「広域撮像探査観測のビッグデータ分析による統計計算宇宙物理学」，JSPS 科研費 16K00150 および JSPS 科研費 JP17H01748 の助成を受けたものです．

参考文献

[1] Tu, S., Zheng, W., Kohler, E., Liskov, B. and Madden, S.: Speedy Transactions in Multicore In-memory Databases, *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, ACM, pp. 18–32 (2013).

[2] Wang, T. and Kimura, H.: Mostly-optimistic concurrency control for highly contended dynamic workloads on a thousand cores, *Proceedings of the VLDB Endowment*, Vol. 10, No. 2, pp. 49–60 (2016).

[3] Yu, X., Pavlo, A., Sanchez, D. and Devadas, S.: TicToc: Time Traveling Optimistic Concurrency Control, *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, New York, NY, USA, ACM, pp. 1629–1642 (online), DOI: 10.1145/2882903.2882935 (2016).

[4] Johnson, R., Pandis, I., Stoica, R., Athanassoulis, M. and Ailamaki, A.: Aether: a scalable approach to logging, *Proceedings of the VLDB Endowment*, Vol. 3, No. 1-2, pp. 681–692 (2010).

[5] Wang, T. and Johnson, R.: Scalable Logging Through Emerging Non-volatile Memory, *Proc. VLDB Endow.*, Vol. 7, No. 10, pp. 865–876 (online), DOI: 10.14778/2732951.2732960 (2014).

[6] 神谷孝明，川島英之， 星野喬，建部修見：並列ログ先行書き込み手法 P-WAL，情報処理学会論文誌データベース（TOD），Vol. 10, No. 1, pp. 24–39 (2017).

[7] 中村泰大，川島英之，神谷孝明，建部修見：並行実行制御手法 TicToc と並列ログ先行書き込み手法 P-WAL の結合，研究報告システムソフトウェアとオペレーティング・システム（OS），Vol. 2017-OS-139, No. 16, pp. 1–11 (2017).

[8] Transaction Processing Performance Council: TPC benchmark C Revision 5.11, http://www.tpc.org/tpcc/ (2010).

[9] Kimura, H.: FOEDUS: OLTP Engine for a Thousand Cores and NVRAM, *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, New York, NY, USA, ACM, pp. 691–706 (online), DOI: 10.1145/2723372.2746480 (2015).

[10] Blackman, D. and Vigna, S.: xorshift128+,

http://xoroshiro.di.unimi.it/xoroshiro128plus.c (2016).

[11] Mao, Y., Kohler, E. and Morris, R. T.: Cache craftiness for fast multicore key-value storage, *Proceedings of the 7th ACM european conference on Computer Systems*, ACM, pp. 183–196 (2012).

[12] Kimura, H., Simitsis, A. and Wilkinson, K.: Janus: Transaction Processing of Navigation and Analytic Graph Queries on Many-core Servers, *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings*, (online), available from ⟨http://cidrdb.org/cidr2017/papers/p104-kimura-cidr17.pdf⟩ (2017).

[13] Wang, T. and Kimura, H.: Mostly-optimistic Concurrency Control for Highly Contended Dynamic Workloads on a Thousand Cores, *Proc. VLDB Endow.*, Vol. 10, No. 2, pp. 49–60 (online), DOI: 10.14778/3015274.3015276 (2016).

[14] Lim, H., Kaminsky, M. and Andersen, D. G.: Cicada: Dependably Fast Multi-Core In-Memory Transactions, *Proceedings of the 2017 ACM International Conference on Management of Data*, pp. 21–35 (2017).