RESSALVA

Atendendo solicitação do
autor, o texto completo
desta dissertação será
disponibilizado somente
a partir de 28/02/2020.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Rio Claro

Rafael Pizzirani Murari

Otimização de Desempenho em Ambientes com Memória Persistente via Transações em Fase

Rio Claro - SP

2019

Rafael Pizzirani Murari

Otimização de Desempenho em Ambientes com Memória Persistente via Transações em Fase

Orientador: Prof. Dr. Alexandro José Baldassin

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Geociências e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de Rio Claro.

Financiadora: CAPES

Rio Claro - SP

2019

M972o

Murari, Rafael Pizzirani
Otimização de Desempenho em Ambientes com Memória Persistente via Transações em Fase / Rafael Pizzirani Murari. -- Rio Claro, 2019
78 f. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientador: Alexandro José Baldassin

1. Ciência da computação. 2. Programação (Computadores). 3. Memória Persistente. 4. Transações Duráveis. 5. Memória Transacional em Fases. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Rafael Pizzirani Murari

Otimização de Desempenho em Ambientes com Memória Persistente via Transações em Fase

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Geociências e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de Rio Claro.

Financiadora: CAPES

Banca Examinadora

- Prof. Dr. Alexandro José Baldassin (Orientador)
  Departamento de Estatística, Matemática Aplicada e Computação
  Universidade Estadual Paulista - UNESP
- Prof. Dr. Emilio de Camargo Francesquini
  Centro de Matemática, Computação e Cognição
  Universidade Federal do ABC - UFABC
- Prof. Dr. Orlando de Andrade Figueiredo
  Departamento de Estatística, Matemática Aplicada e Computação
  Universidade Estadual Paulista - UNESP

Rio Claro - SP

28 de fevereiro de 2019

Agradecimentos

Em primeiro lugar gostaria de agradecer a Deus por ter me guiado e auxiliado em todos os momentos de minha vida. “Tu és o meu Deus, e eu te louvarei; Tu és o meu Deus, e eu te exaltarei” - Salmos 118:28.

Agradeço a meus pais por todo suporte, pelo carinho e pela educação que me deram até este momento. Não há palavras suficientes para expressar a minha gratidão a Deus por todos os momentos que passamos juntos.

Meus sinceros agradecimentos ao meu professor e orientador, Dr. Alexandro José Baldassin pela imensa contribuição em minha vida acadêmica e profissional. Seus ensinamentos e ajuda foram de profunda valia para realização desta dissertação. Agradeço também a João Paulo Labegalini de Carvalho pela grande contribuição no desenvolvimento deste trabalho.

Agradeço a todos os professores e amigos que contribuíram em minha formação acadêmica, em especial os membros do Projeto Radiar e os integrantes da equipe do desafio de programação paralela.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, à qual agradeço.

Resumo

As emergentes tecnologias de memória persistente (PM) visam eliminar a lacuna existente entre a memória principal e a secundária. No entanto, os sistemas atuais não são capazes de usufruir totalmente dos benefícios proporcionados por estas, essencialmente devido a possíveis falhas de sistema que podem resultar em um estado inconsistente e irrecuperável. Além disto, o uso simplista da PM resulta em uma degradação de desempenho, advinda do alto custo associado às operações de escrita. Neste contexto, o uso de transações duráveis é uma das abordagens mais investigadas para facilitar a adoção da PM. Em particular, implementações de memória transacional em hardware (HTM) possibilitam a execução de transações com uma sobrecarga mínima, porém apresentam limitações de recursos. Embora as transações em software (STM) sejam flexíveis e não possuam tais limitações, estas não apresentam um bom desempenho na execução de transações curtas. Esta dissertação apresenta a solução NV-PhTM, um sistema transacional baseado em fases capaz de alterar dinamicamente o modo de execução, software ou hardware, mediante as características apresentadas pelas aplicações. A implementação do NV-PhTM foi embasada pelo sistema PhTM*, um arcabouço que provê um conjunto de heurísticas para guiar a transição e seleção do melhor modo de execução (HW/SW). O PhTM*, no entanto, foi concebido para ambientes de memória volátil. Neste contexto, o NV-PhTM propõe novas heurísticas visando contemplar as estratégias de garantia de durabilidade e as características da PM. Visando manter a corretude do sistema, estratégias foram elaboradas a fim de garantir a persistência durante a transição entre as fases. O NV-PhTM é o primeiro sistema transacional baseado em fases a prover transações duráveis. Os resultados experimentais obtidos, na execução do *benchmark* STAMP, comprovam a eficácia das novas heurísticas em guiar a transição das fases. Quando comparado ao NV-HTM (solução exclusivamente em hardware) e ao PSTM (solução exclusivamente em software), o NV-PhTM obteve os melhores resultados devido a sua natureza de seguir o sistema com melhor desempenho.

Palavras-chave: memória persistente, transações duráveis, memória transacional.

Abstract

The emerging persistent memory technologies (PM) are aimed to eliminate the gap between main memory and storage. Nevertheless, today's programs will not readily benefit from them essentially because crash failures might render the program in an unrecoverable and inconsistent state. In addition, naive utilization of PM leads to performance degradation, mainly due to expensive writes. In this context, the usage of durable transactions is one of the main investigated approaches to ease the adoption of non-volatile memory. In particular, hardware transactional memories (HTM) provide low-overhead but are resource-constrained. Although software transactions (STM) are flexible and unbounded, they may significantly hurt the performance of short-lived transactions. This dissertation presents NV-PhTM, a transactional system for PM that delivers the best out of both HW and SW transactions by dynamically changing the execution according to the characteristics of the application. NV-PhTM is built upon the PhTM* system, a framework that provides a set of heuristics to guide phase transition and selects the best execution mode (HW/SW). PhTM*, however, did not take into account PM systems. Designing NV-PhTM required new heuristics that consider the nature of durability strategies and the characteristics of PM. In order to keep the accuracy of the system, new strategies were elaborated to guarantee persistency between phase transitions. NV-PhTM is the first phase-based system to provide durable transactions. Experimental results with the STAMP benchmark show that the proposed heuristics are efficient in guiding phase transitions with low overhead. When compared to NV-HTM (hardware-only durable transactions) and PSTM (software-only durable transactions), NV-PhTM provided the best overall results due to its nature of following the best performing system.

Keywords: persistent memory, durable transactions, transactional memory.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

API	*Application Programming Interface*
BFC	*Backward Filtering Checkpointing*
CAS	*Compare-And-Swap*
CLWB	*Cache Line Write Back*
CP	*Checkpoint Process*
DAX	*Direct Access*
DRAM	*Dynamic Random Access Memory*
HDD	*Hard Disk Drive*
HTM	*Hardware Transactional Memory*
HyTM	*Hybrid Transactional Memory*
IMDB	*In-Memory Database*
ISA	*Instruction Set Architecture*
LLC	*Last Level Cache*
MMU	*Memory Management Unit*
NV-PhTM	*Non-Volatile Phased Transactional Memory*
NVM	*Non-Volatile Memory*
PCM	*Phase-Change Memory*
PhTM	*Phased Transactional Memory*
PM	*Persistent Memory*
PS	*Persistent Snapshot*
RRAM	*Resistive Random Access Memory*
RTM	*Restricted Transactional Memory*
SGL	*Single Global Lock*
SPMD	*Single Program Multiple Data*
SRAM	*Static Random Access Memory*
SSD	*Solid-State Drive*
STAMP	*Stanford Transactional Applications for Multi-Proccessing*
STM	*Software Transactional Memory*
STT-RAM	*Spin-Transfer Torque Random Access Memory*
TLB	*Translation Lookaside Buffer*
TM	*Transactional Memory*
TSX	*Transactional Synchronization Extension*
WP	*Working Process*
WS	*Working Snapshot*

Sumário

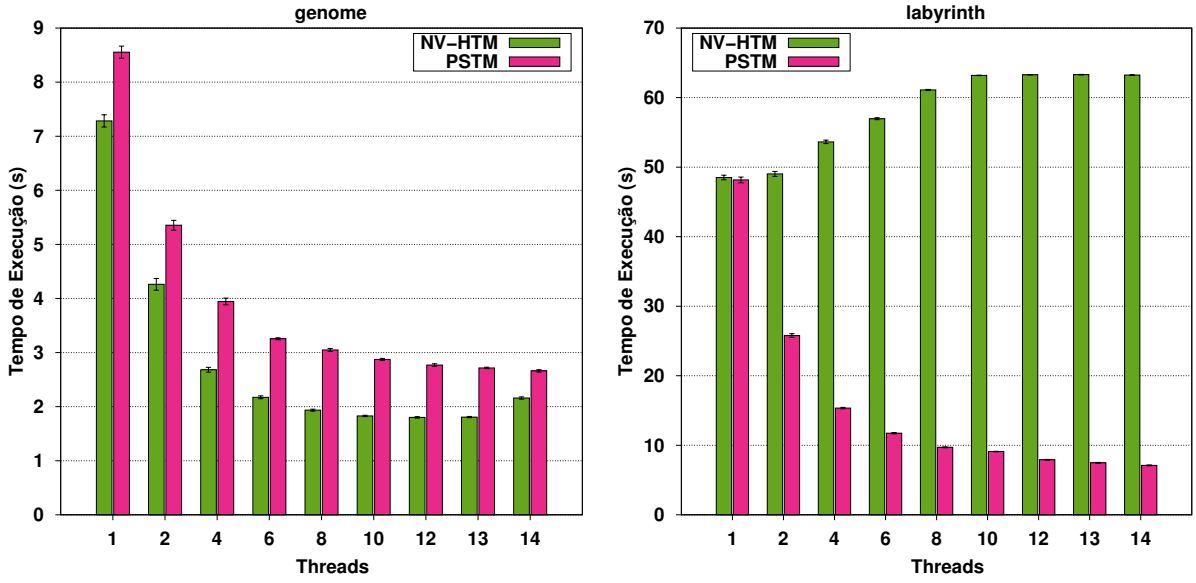
1 Introdução

A grande capacidade computacional, disponibilizada pelos processadores contemporâneos, em conjunto com a crescente demanda de dados pelas aplicações, tem exposto enfaticamente o gargalo de desempenho do subsistema de memória, resultante da alta latência de acesso aos componentes persistentes deste. Neste contexto, as emergentes tecnologias de memória não-voláteis (NVM) visam equacionar os *trade-offs* de persistência e desempenho mediante as características disruptivas apresentadas, tais como alta densidade, endereçamento por byte, acesso via instruções de memória (*load* e *store*) e latência de acesso diminuta [1].

A adoção desta tecnologia prenuncia um elevado desempenho das aplicações, no entanto certas precauções devem ser tomadas para garantia de consistência dos dados, visto que eventuais quedas de energia ou falhas de sistema podem corromper permanentemente regiões inteiras de memória. Diversas abordagens para garantia de consistência podem ser encontradas na literatura, entre estas destaca-se o uso de transações duráveis, conceito inicialmente proposto no contexto de banco de dados [2]. Esta abordagem visa delimitar as operações persistentes em blocos atômicos de execução, ou seja, as instruções presentes nestes são executadas de maneira indivisível e atômica, garantindo a transição de um estado previamente consistente para outro estado consistente ao término de sua execução.

Grande parte das soluções transacionais propostas são em software (STM) [3, 4, 5, 6, 7], possibilitando seu uso de maneira irrestrita, isto é, sem a requisição de recursos em hardware para sua execução. Entretanto, as instrumentações inseridas pelas implementações em software ocasionam custo extra, sendo este crítico em transações com *workload* reduzido. Neste âmbito, o suporte transacional em hardware (HTM), disponível em processadores da Intel e IBM, pode prover um relevante ganho de desempenho em ambientes com memória não-volátil [8]. Porém, a conjuntura NVM e HTM não é trivial, principalmente pelas especificidades apresentadas pelas implementações de HTM contemporâneas.

Diversos trabalhos propuseram modificações arquiteturais para facilitar o uso conjunto destas tecnologias [9, 10, 11, 12], todavia não é possível realizar uma avaliação real destes sistemas, visto que estes foram validados através de simulações. A solução NV-HTM, desenvolvida por Castro e colegas [8], é uma das pioneiras a viabilizar o uso de HTM sem extensões, sendo



(a) Genome (b) Labyrinth

Figura 1 – Tempo de execução das aplicações: (a) Genome e (b) Labyrinth.

capaz de usufruir do alto desempenho proporcionado por este. O único limitante desta solução é a baixa disponibilidade de recursos, advinda do pouco espaço em cache para manutenção do trabalho especulativo gerado pelas transações. Logo, a implementação transacional em hardware não é capaz de efetivar transações com amplo *workload*, recorrendo a estratégia de aquisição de um *lock* global, serializando a execução das transações, e concluindo estas em software.

As implementações atuais, em hardware e software, não contemplam o amplo espectro de características apresentadas pelas aplicações, culminando em perda de desempenho. Para ilustrar tal cenário, a Figura 1 apresenta o tempo de execução obtido pelas aplicações Genome e Labyrinth, integrantes do *benchmark* STAMP [13], em dois sistemas distintos: NV-HTM (implementação em hardware) e PSTM [10] (implementação em software). As baixas taxas de cancelamentos e de contenção das transações, apresentadas pela aplicação Genome, possibilitam um melhor desempenho em implementações em hardware, visto que o software introduz um elevado *overhead* para garantia de consistência. No entanto, aplicações com transações longas, isto é, grande quantidade de instruções presentes no bloco atômico, e com amplo *workload*, conforme apresentado pela aplicação Labyrinth, não obtêm êxito em hardware, forçando constantes serializações para garantia de progresso. Neste cenário, a implementação em software proporciona desempenho amplamente superior.

1.1 Objetivos e Contribuições

Este trabalho apresenta a solução NV-PhTM, sistema transacional baseado em fases capaz de alterar dinamicamente o modo de execução, software ou hardware, mediante as características apresentadas pelas aplicações. A implementação do NV-PhTM foi embasada pelo PhTM* [14], sistema em fases, concebido para ambientes de memória volátil, coordenado por um conjunto de heurísticas responsável pela detecção do modo mais eficiente de execução (hardware/software) e pela respectiva transição para tal modo.

As principais contribuições do NV-PhTM são: (i) a adição do conceito de durabilidade em sistemas baseados em fase; (ii) a elaboração de novas heurísticas, contemplando a natureza das estratégias de durabilidade empregadas pelo NV-HTM e as características da NVM; (iii) o desenvolvimento de estratégias para garantia de consistência durante o processo de transição entre os distintos modos do sistema.

A avaliação experimental realizada comprova a eficácia das novas heurísticas em guiar a transição das fases, bem como o baixo *overhead* adicionado neste processo. A comparação realizada com sistemas exclusivamente em software ou hardware, demonstra o êxito do NV-PhTM em direcionar o sistema a acompanhar a melhor abordagem nos mais distintos cenários.

1.2 Organização do Texto

Para melhor compreensão deste trabalho, o texto foi organizado da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica dos diversos conceitos envolvidos no desenvolvimento desta dissertação. O Capítulo 3 apresenta os principais trabalhos propostos na literatura para uso conjunto de HTM e NVM. Em seguida, o Capítulo 4 apresenta o sistema NV-PhTM e seus respectivos detalhes de implementação. Como motivação da proposta, apresenta-se uma análise que contrasta o desempenho das implementações em software e hardware. Por fim, os Capítulos 5 e 6 apresentam a avaliação experimental realizada e as conclusões obtidas, respectivamente.

6 Conclusão

Neste trabalho foi apresentado o potencial desempenho proporcionado pelas emergentes tecnologias de memória persistente. A análise feita revelou que a maior exposição da memória para as aplicações permite eliminar vários *overheads* e maximizar o desempenho das aplicações. Porém, é necessário a utilização de estratégias para garantia de consistência dos dados em eventuais falhas de sistemas ou quedas de energia. Entre as estratégias foi destacado o uso de Memória Transacional em suas distintas implementações, software e hardware. Mais especificamente, foram analisadas as principais virtudes e limitações apresentadas pelas diversas soluções presentes na literatura, culminando na constatação de que os sistemas atuais não contemplam o amplo espectro de características das aplicações, por consequência obtendo um desempenho subótimo.

Como alternativa às soluções existentes, este trabalho apresenta uma implementação transacional baseada em fases (NV-PhTM), a qual utiliza o *feedback* retornado pelas transações em conjunto com heurísticas para guiar a transição entre as distintas implementações transacionais. As principais contribuições deste trabalho são: (i) a adição do conceito de durabilidade em sistemas baseados em fase; (ii) a concepção de novas heurísticas que contemplam as especificidades apresentadas pela NVM e pelas soluções integrantes do sistema; (iii) a elaboração de estratégias para garantia de consistência durante a transição entre os distintos modos.

A avaliação experimental realizada comprova o êxito do NV-PhTM em detectar e acompanhar a melhor abordagem de execução nos mais distintos cenários apresentados. As novas heurísticas foram determinantes para evitar o processo de estagnação, originado pela ausência de espaço nos *logs* e pela demora na execução do processo de *checkpoint*. Por fim, as estratégias empregadas para consolidação das transições introduziram um *overhead* diminuto, impactando minimamente o desempenho do sistema.

6.1 Trabalhos Futuros

Apesar dos bons resultados obtidos, ainda assim pode-se destacar uma possível melhoria a ser feita: a elaboração de novas métricas que contemplem os cenários adversos apresentados pelas aplicações Intruder e Yada. Em ambos os casos o sistema transita para software, permanecendo neste modo independentemente do melhor desempenho proporcionado pelo hardware. Neste

contexto, a inserção de transições esporádicas para o modo HW poderia contribuir na atualização de estatísticas, tais como a média de ciclos empregadas pelas transações, utilizadas para detecção da melhor implementação a ser executada.

Considerando a modularidade oferecida pelo NV-PhTM, diversas propostas de trabalhos futuros podem ser elencadas:

- A avaliação de distintas implementações transacionais em software, encontradas na literatura, visando encontrar a solução que melhor se adequa às especificidades apresentadas pela implementação em hardware (NV-HTM), e também pela etapa de consolidação, realizada durante as transições para garantia de consistência;
- A implementação de uma abordagem de STM baseada nos mesmos conceitos apresentados pelo NV-HTM (*checkpoint*, *snapshot* persistente, *snapshot* temporário, ...). A inserção deste novo módulo em SW no sistema possibilita a exclusão do processo de drenagem dos *logs*, visto que ambas implementações (HW e SW) utilizam as mesmas estratégias para garantia de consistência do estado persistente da aplicação. Esta nova conjuntura apresenta duas possíveis configurações: (i) o uso compartilhado do *log* pelos distintos modos, possibilitando o reuso de memória; (ii) a disponibilização de uma região persistente distinta de memória para manutenção do *log* da implementação em SW, proporcionando um certo isolamento dos metadados das soluções;
- A ampliação do conjunto de aplicações utilizadas para avaliação do sistema, possibilitando uma melhor validação e uma possível expansão do conjunto de heurísticas previamente definido.

Referências

1 BADAM, A. How persistent memory will change software systems. *Computer*, IEEE, v. 46, n. 8, p. 45–51, 2013.

2 HARRIS, T.; LARUS, J.; RAJWAR, R. *Transactional Memory*. 2. ed. [S.l.]: Morgan & Claypool Publishers, 2010.

3 VOLOS, H.; TACK, A. J.; SWIFT, M. M. Mnemosyne: Lightweight persistent memory. In: *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2011. (ASPLOS XVI), p. 91–104.

4 COBURN, J.; CAULFIELD, A. M.; AKEL, A.; GRUPP, L. M.; GUPTA, R. K.; JHALA, R.; SWANSON, S. Nv-heaps: Making persistent objects fast and safe with next-generation, non-volatile memories. In: *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2011. (ASPLOS XVI), p. 105–118.

5 MEMARIPOUR, A.; BADAM, A.; PHANISHAYEE, A.; ZHOU, Y.; ALAGAPPAN, R.; STRAUSS, K.; SWANSON, S. Atomic in-place updates for non-volatile main memories with kamino-tx. In: *Proceedings of the Twelfth European Conference on Computer Systems*. New York, NY, USA: ACM, 2017. (EuroSys '17), p. 499–512.

6 TEAM, T. N. L. *Pmem.io: Persistent Memory Programming*. 2014. Disponível em: <http://pmem.io/>. Acessado em: 09 de jan. 2019.

7 DENNY, J. E.; LEE, S.; VETTER, J. S. Nvl-c: Static analysis techniques for efficient, correct programming of non-volatile main memory systems. In: *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*. New York, NY, USA: ACM, 2016. (HPDC '16), p. 125–136.

8 CASTRO, D.; ROMANO, P.; BARRETO, J. Hardware transactional memory meets memory persistency. In: *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Vancouver, BC, Canada: IEEE, 2018. p. 368–377.

9 WANG, Z.; YI, H.; LIU, R.; DONG, M.; CHEN, H. Persistent transactional memory. *IEEE Computer Architecture Letters*, IEEE, v. 14, n. 1, p. 58–61, Jan 2015.

10 AVNI, H.; LEVY, E.; MENDELSON, A. Hardware transactions in nonvolatile memory. In: *Proceedings of the 29th International Symposium on Distributed Computing - Volume 9363*. Berlin, Heidelberg: Springer-Verlag, 2015. (DISC 2015), p. 617–630.

11 AVNI, H.; BROWN, T. Persistent hybrid transactional memory for databases. *Proc. VLDB Endow.*, VLDB Endowment, v. 10, n. 4, p. 409–420, nov. 2016.

12 JOSHI, A.; NAGARAJAN, V.; CINTRA, M.; VIGLAS, S. Dhtm: Durable hardware transactional memory. In: *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. Los Angeles, CA, USA: IEEE, 2018. p. 452–465.

13 MINH, C. C.; CHUNG, J.; KOZYRAKIS, C.; OLUKOTUN, K. Stamp: Stanford transactional applications for multi-processing. In: *2008 IEEE International Symposium on Workload Characterization*. Seattle, WA, USA: IEEE, 2008. p. 35–46.

14 CARVALHO, J. P. D.; ARAUJO, G.; BALDASSIN, A. The case for phase-based transactional memory. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, 2018.

15 HENNESSY, J. L.; PATTERSON, D. A. *Computer Architecture, Fifth Edition: A Quantitative Approach*. 5. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

16 MOORE, G. E. Progress in digital integrated electronics. In: *Electron Devices Meeting*. Santa Clara, California: IEEE, 1975. v. 21, p. 11–13.

17 BURKS, A. W.; GOLDSTINE, H. H.; NEUMANN, J. V. Preliminary discussion of the logical design of an electronic computing instrument. In: *The Origins of Digital Computers*. Berlin Heidelberg: Springer, 1982. p. 399–413.

18 ÅKERMAN, J. Toward a universal memory. *Science*, American Association for the Advancement of Science, v. 308, n. 5721, p. 508–510, 2005.

19 YANG, J.; WEI, Q.; CHEN, C.; WANG, C.; YONG, K. L.; HE, B. Nv-tree: Reducing consistency cost for nvm-based single level systems. In: *Proceedings of the 13th USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2015. (FAST'15), p. 167–181.

20 ZHANG, Z.; FENG, D.; CHEN, J.; YU, Y.; ZENG, J. The design and implementation of a lightweight management framework for non-volatile memory. In: *2016 IEEE Trustcom/BigDataSE/ISPA*. Tianjin, China: IEEE, 2016. p. 1551–1558.

21 WULF, W. A.; MCKEE, S. A. Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, ACM, New York, NY, USA, v. 23, n. 1, p. 20–24, mar. 1995.

22 PAPAMARCOS, M. S.; PATEL, J. H. A low-overhead coherence solution for multiprocessors with private cache memories. In: *Proceedings of the 11th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 1984. (ISCA '84), p. 348–354.

23 BAER, J.-L. *Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors*. 1. ed. New York, NY, USA: Cambridge University Press, 2009.

24 JACOB, B.; NG, S.; WANG, D. *Memory Systems: Cache, DRAM, Disk*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

25 OUKID, I.; LEHNER, W. Data structure engineering for byte-addressable non-volatile memory. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. New York, NY, USA: ACM, 2017. (SIGMOD '17), p. 1759–1764.

26 MUTLU, O.; SUBRAMANIAN, L. Research problems and opportunities in memory systems. *Supercomputing frontiers and innovations*, South Ural State University, v. 1, n. 3, p. 19, 2014.

27 KüLTüRSAY, E.; KANDEMIR, M.; SIVASUBRAMANIAM, A.; MUTLU, O. Evaluating stt-ram as an energy-efficient main memory alternative. In: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Austin, TX, USA: IEEE, 2013. p. 256–267.

28 LEE, B. C.; IPEK, E.; MUTLU, O.; BURGER, D. Architecting phase change memory as a scalable dram alternative. In: *Proceedings of the 36th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2009. (ISCA '09), p. 2–13.

29 DHIMAN, G.; AYOUB, R.; ROSING, T. Pdram: A hybrid pram and dram main memory system. In: *2009 46th ACM/IEEE Design Automation Conference*. San Francisco, CA, USA: IEEE, 2009. p. 664–669.

30 YOON, H. Row buffer locality aware caching policies for hybrid memories. In: *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD 2012)*. Washington, DC, USA: IEEE Computer Society, 2012. (ICCD '12), p. 337–344.

31 WONG, H.-S. P.; RAOUX, S.; KIM, S.; LIANG, J.; REIFENBERG, J. P.; RAJENDRAN, B.; ASHEGHI, M.; GOODSON, K. E. Phase change memory. *Proceedings of the IEEE*, IEEE, v. 98, n. 12, p. 2201–2227, 2010.

32 CHEN, E.; APALKOV, D.; DIAO, Z.; DRISKILL-SMITH, A.; DRUIST, D.; LOTTIS, D.; NIKITIN, V.; TANG, X.; WATTS, S.; WANG, S.; WOLF, S. A.; GHOSH, A. W.; LU, J. W.; POON, S. J.; STAN, M.; BUTLER, W. H.; GUPTA, S.; MEWES, C. K. A.; MEWES, T.; VISSCHER, P. B. Advances and future prospects of spin-transfer torque random access memory. *IEEE Transactions on Magnetics*, IEEE, v. 46, n. 6, p. 1873–1878, 2010.

33 CHANG, T.-C.; CHANG, K.-C.; TSAI, T.-M.; CHU, T.-J.; SZE, S. M. Resistance random access memory. *Materials Today*, v. 19, n. 5, p. 254 – 264, 2016.

34 POSITION, S. T. *NVM Programming Model (NPM) Version 1.2*. 2017. Disponível em: <https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf>. Acessado em: 09 de jan. 2019.

35 VENKATARAMAN, S.; TOLIA, N.; RANGANATHAN, P.; CAMPBELL, R. H. Consistent and durable data structures for non-volatile byte-addressable memory. In: *Proceedings of the 9th USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2011. (FAST'11), p. 61–75.

36 CHAKRABARTI, D. R.; BOEHM, H.-J.; BHANDARI, K. Atlas: Leveraging locks for non-volatile memory consistency. In: *Proceedings of the 2014 ACM International Conference on Object Oriented Programming Systems Languages & Applications*. New York, NY, USA: ACM, 2014. (OOPSLA '14), p. 433–452.

37 IZRAELEVITZ, J.; KELLY, T.; KOLLI, A. Failure-atomic persistent memory updates via justdo logging. In: *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2016. (ASPLOS '16), p. 427–442.

38 HSU, T. C.-H.; BRüGNER, H.; ROY, I.; KEETON, K.; EUGSTER, P. Nvthreads: Practical persistence for multi-threaded applications. In: *Proceedings of the Twelfth European Conference on Computer Systems*. New York, NY, USA: ACM, 2017. (EuroSys '17), p. 468–482.

39 ESWARAN, K. P.; GRAY, J. N.; LORIE, R. A.; TRAIGER, I. L. The notions of consistency and predicate locks in a database system. *Commun. ACM*, ACM, New York, NY, USA, v. 19, n. 11, p. 624–633, nov. 1976.

40 BALDASSIN, A. Tudo o que você sempre quis saber sobre memória transacional mas tinha medo de perguntar. In: MARTINS, C. A. P. da S. (Ed.). *Sistemas Computacionais*. Porto Alegre: Sociedade Brasileira de Computação, 2014.

41 CARVALHO, J. P. L. de. *PhTM*: uma implementação eficiente de transações em fases*. Tese (Mestrado em Ciência da Computação) - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista. Rio Claro, São Paulo. 2016.

42 HERLIHY, M.; MOSS, J. E. B. Transactional memory: Architectural support for lock-free data structures. In: *Proceedings of the 20th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 1993. (ISCA '93), p. 289–300.

43 SHAVIT, N.; TOUITOU, D. Software transactional memory. In: *Proceedings of the Fourteenth Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM, 1995. (PODC '95), p. 204–213.

44 DALESSANDRO, L.; SPEAR, M. F.; SCOTT, M. L. Norec: Streamlining stm by abolishing ownership records. In: *Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. New York, NY, USA: ACM, 2010. (PPoPP '10), p. 67–78.

45 HERLIHY, M.; SHAVIT, N. *The Art of Multiprocessor Programming*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

46 INTEL CORPORATION. *Intel® Architecture Instruction Set Extensions Programming Reference*. [S.l.], 2012.

47 JACOBI, C.; SLEGEL, T.; GREINER, D. Transactional memory architecture and implementation for IBM system z. In: *MICRO12*. Vancouver, BC, Canada: IEEE, 2012. p. 25–36.

48 JIANG, J.; PHILIPPEN, P.; KNOBLOCH, M.; MOHR, B. Performance measurement and analysis of transactional memory and speculative execution on IBM blue Gene/Q. In: *EUROPAR14*. Porto, Portugal: Springer, 2014. p. 26–37.

49 LEV, Y.; MOIR, M.; NUSSBAUM, D. PhTM: Phased Transactional Memory. In: *Workshop on Transactional Computing (Transact)*. Portland, Oregon, USA: [s.n.], 2007.

50 GOEL, B.; TITOS-GIL, R.; NEGI, A.; MCKEE, S. A.; STENSTROM, P. Performance and energy analysis of the restricted transactional memory implementation on haswell. In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. Phoenix, AZ, USA: IEEE, 2014. p. 615–624.

51 YOO, R. M.; LEE, H.-H. S. Adaptive transaction scheduling for transactional memory systems. In: *Proceedings of the Twentieth Annual Symposium on Parallelism in Algorithms and Architectures*. New York, NY, USA: ACM, 2008. (SPAA '08), p. 169–178.

52 KERNEL LINUX DOCUMENTATION. *DAX - Direct Access for Files*. Disponível em: <https://www.kernel.org/doc/Documentation/filesystems/dax.txt>. Acessado em: 09 de jan. 2019.

53 CORREIA, A.; FELBER, P.; RAMALHETE, P. Romulus: Efficient algorithms for persistent transactional memory. In: *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*. New York, NY, USA: ACM, 2018. (SPAA '18), p. 271–282.

54 LIU, M.; ZHANG, M.; CHEN, K.; QIAN, X.; WU, Y.; ZHENG, W.; REN, J. Dudetm: Building durable transactions with decoupling for persistent memory. In: *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2017. (ASPLOS '17), p. 329–343.

55 GILES, E.; DOSHI, K.; VARMAN, P. Continuous checkpointing of htm transactions in nvm. In: *Proceedings of the 2017 ACM SIGPLAN International Symposium on Memory Management*. New York, NY, USA: ACM, 2017. (ISMM 2017), p. 70–81.

56 FELBER, P.; FETZER, C.; RIEGEL, T. Dynamic performance tuning of word-based software transactional memory. In: *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. New York, NY, USA: ACM, 2008. p. 237–246.

57 FELBER, P.; FETZER, C.; MARLIER, P.; RIEGEL, T. Time-based software transactional memory. *IEEE Transactions on Parallel & Distributed Systems*, IEEE Computer Society, Los Alamitos, CA, USA, v. 21, p. 1793–1807, 2010.

58 BALDASSIN, A.; BORIN, E.; ARAUJO, G. Performance implications of dynamic memory allocators on transactional memory systems. In: *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. New York, NY, USA: ACM, 2015. (PPoPP 2015), p. 87–96.

59 DICE, D.; HARRIS, T.; KOGAN, A.; LEV, Y. The influence of malloc placement on TSX hardware transactional memory. *CoRR*, abs/1504.04640, 2015.

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

TERMO DE REPRODUÇÃO XEROGRÁFICA

Autorizo a reprodução xerográfica do presente Trabalho de Conclusão, na íntegra ou em partes, para fins de pesquisa.

São José do Rio Preto, 11/03/2019

Rafael Pizzirani Murari

Assinatura do autor