REFERENCES

[1] Marcos K Aguilera and Svend Frølund. 2003. Strict linearizability and the power of aborting. *Technical Report HPL-2003-241* (2003). https://hpl.hp.com/techreports/2003/HPL-2003-241.html

[2] Hiroyuki Akinaga and Hisashi Shima. 2010. Resistive random access memory (ReRAM) based on metal oxides. *Proc. IEEE* 98, 12 (2010). https://doi.org/10.1109/JPROC.2010.2070830

[3] Ryan Berryhill, Wojciech Golab, and Mahesh Tripunitara. 2015. Robust shared objects for non-volatile main memory. In *OPODIS*. https://doi.org/10.4230/LIPIcs.OPODIS.2015.20

[4] Hadi Brais and Andy Rudoff. 2021. *Reply to On x86-64, is the "movnti" or "movntdq" instruction atomic when system crash?* https://stackoverflow.com/a/65587308/7289606

[5] Heng Bu, Ming-Kai Dong, Ji-Fei Yi, Bin-Yu Zang, and Hai-Bo Chen. 2021. Revisiting persistent indexing structures on Intel Optane DC persistent memory. *Journal of Computer Science and Technology* 36, 1 (2021). https://doi.org/10.1007/s11390-020-9871-0

[6] Dhruva R Chakrabarti, Hans-J Boehm, and Kumud Bhandari. 2014. Atlas: Leveraging locks for non-volatile memory consistency. *ACM SIGPLAN Notices* 49, 10 (2014). https://doi.org/10.1145/2714064.2660224

[7] Joel Coburn, Adrian M Caulfield, Ameen Akel, Laura M Grupp, Rajesh K Gupta, Ranjit Jhala, and Steven Swanson. 2011. NV-Heaps: Making persistent objects fast and safe with next-generation, non-volatile memories. In *ASPLOS*. https://doi.org/10.1145/1961295.1950380

[8] Nachshon Cohen, Michal Friedman, and James R Larus. 2017. Efficient logging in non-volatile memory by exploiting coherency protocols. *PACMPL* 1, OOPSLA (2017). https://doi.org/10.1145/3133891

[9] Nachshon Cohen, Rachid Guerraoui, and Igor Zablotchi. 2018. The inherent cost of remembering consistently. In *SPAA*. https://doi.org/10.1145/3210377.3210400

[10] Andreia Correia, Pascal Felber, and Pedro Ramalhete. 2018. Romulus: Efficient algorithms for persistent transactional memory. In *SPAA*. https://doi.org/10.1145/3210377.3210392

[11] Andreia Correia, Pascal Felber, and Pedro Ramalhete. 2020. Persistent memory and the rise of universal constructions. In *EuroSys*. https://doi.org/10.1145/3342195.3387515

[12] Tudor David, Aleksandar Dragojevic, Rachid Guerraoui, and Igor Zablotchi. 2018. Log-free concurrent data structures. In *USENIX ATC*. https://usenix.org/conference/atc18/presentation/david

[13] Tudor David, Rachid Guerraoui, and Vasileios Trigonakis. 2015. Asynchronized concurrency: The secret to scaling concurrent search data structures. In *ASPLOS*. https://doi.org/10.1145/2786763.2694359

[14] Panagiota Fatourou and Nikolaos D Kallimanis. 2012. Revisiting the combining synchronization technique. In *PPoPP*. https://doi.org/10.1145/2370036.2145849

[15] Michal Friedman, Naama Ben-David, Yuanhao Wei, Guy E Blelloch, and Erez Petrank. 2020. NVTraverse: In NVRAM data structures, the destination is more important than the journey. In *PLDI*. https://doi.org/10.1145/3385412.3386031

[16] Michal Friedman, Maurice Herlihy, Virendra Marathe, and Erez Petrank. 2018. A persistent lock-free queue for non-volatile memory. In *PPoPP*. https://doi.org/10.1145/3178487.3178490

[17] Michal Friedman, Erez Petrank, and Pedro Ramalhete. 2021. Mirror: Making lock-free data structures persistent. In *PLDI*. https://doi.org/10.1145/3453483.3454105

[18] Rachid Guerraoui and Ron R Levy. 2004. Robust emulations of shared memory in a crash-recovery model. In *ICDCS*. https://doi.org/10.1109/ICDCS.2004.1281605

[19] Andreas Haas, Michael Lippautz, Thomas A Henzinger, Hannes Payer, Ana Sokolova, Christoph M Kirsch, and Ali Sezgin. 2013. Distributed queues in shared memory: Multicore performance and scalability through quantitative relaxation. In *CF*. https://doi.org/10.1145/2482767.2482789

[20] Xiangpeng Hao. 2019. *Is CLWB actually implemented?* https://blog.haoxp.xyz/posts/is-clwb-implemented

[21] Maurice Herlihy. 1991. Wait-free synchronization. *TOPLAS* 13, 1 (1991). https://doi.org/10.1145/114005.102808

[22] Maurice Herlihy and Jeannette M. Wing. 1990. Linearizability: A correctness condition for concurrent objects. *TOPLAS* 12, 3 (1990). https://doi.org/10.1145/78969.78972

[23] Moshe Hoffman, Ori Shalev, and Nir Shavit. 2007. The baskets queue. In *OPODIS*. https://doi.org/10.1007/978-3-540-77096-1_29

[24] IBM. [n.d.]. *IBM MQ*. https://ibm.com/software/products/en/ibm-mq

[25] Intel. 2019. *3D XPoint™: A breakthrough in non-volatile memory technology*. https://intel.com/content/www/us/en/architecture-and-technology/intel-micron-3d-xpoint-webcast.html

[26] Intel. 2020. *Intel® 64 and IA-32 architectures software developer's manual*. https://software.intel.com/content/dam/develop/external/us/en/documents-tps/325462-sdm-vol-1-2abcd-3abcd.pdf

[27] Joseph Izraelevitz, Hammurabi Mendes, and Michael L Scott. 2016. Linearizability of persistent memory objects under a full-system-crash failure model. In *DISC*. https://doi.org/10.1007/978-3-662-53426-7_23

[28] Anuj Kalia, David Andersen, and Michael Kaminsky. 2020. Challenges and solutions for fast remote persistent memory access. In *SoCC*. https://doi.org/10.1145/3419111.3421294

[29] Christoph M Kirsch, Michael Lippautz, and Hannes Payer. 2013. Fast and scalable, lock-free k-FIFO queues. In *PaCT*. https://doi.org/10.1007/978-3-642-39958-9_18

[30] Aasheesh Kolli, Steven Pelley, Ali Saidi, Peter M Chen, and Thomas F Wenisch. 2016. High-performance transactions for persistent memories. In *ASPLOS*. https://doi.org/10.1145/2872362.2872381

[31] Edya Ladan-Mozes and Nir Shavit. 2004. An optimistic approach to lock-free FIFO queues. In *Distributed Computing*. https://doi.org/10.1007/s00446-007-0050-0

[32] Doug Lea. 2009. The Java concurrency package (JSR-166).

[33] Virendra Marathe, Achin Mishra, Amee Trivedi, Yihe Huang, Faisal Zaghloul, Sanidhya Kashyap, Margo Seltzer, Tim Harris, Steve Byan, Bill Bridge, et al. 2018. Persistent memory transactions. *arXiv preprint* (2018). arXiv:1804.00701

[34] Amirsaman Memaripour, Joseph Izraelevitz, and Steven Swanson. 2020. Pronto: Easy and fast persistence for volatile data structures. In *ASPLOS*. https://doi.org/10.1145/3373376.3378456

[35] Maged M. Michael and Michael L. Scott. 1996. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *PODC*. https://doi.org/10.1145/248052.248106

[36] Adam Morrison and Yehuda Afek. 2013. Fast concurrent queues for x86 processors. In *PPoPP*. https://doi.org/10.1145/2442516.2442527

[37] Oracle. [n.d.]. *Oracle Tuxedo Message Queue*. https://docs.oracle.com/cd/E35855_01/otmq/docs12c/overview/overview.html

[38] Or Ostrovsky and Adam Morrison. 2020. Scaling concurrent queues by using HTM to profit from failed atomic operations. In *PPoPP*. https://doi.org/10.1145/3332466.3374511

[39] Azalea Raad, John Wickerson, Gil Neiger, and Viktor Vafeiadis. 2020. Persistency semantics of the Intel-x86 architecture. *PACMPL* 4, POPL (2020). https://doi.org/10.1145/3371079

[40] Pedro Ramalhete, Andreia Correia, Pascal Felber, and Nachshon Cohen. 2019. OneFile: A wait-free persistent transactional memory. In *DSN*. https://doi.org/10.1109/DSN.2019.00028

[41] Simone Raoux, Geoffrey W Burr, Matthew J Breitwisch, Charles T Rettner, Y-C Chen, Robert M Shelby, Martin Salinga, Daniel Krebs, S-H Chen, H-L Lung, et al. 2008. Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development* 52, 4.5 (2008). https://doi.org/10.1147/rd.524.0465

[42] Andy Rudoff. 2019. *Reply to How to use CLWB instructions*. https://groups.google.com/g/pmem/c/R8H3sKq9sLQ/m/ltL7Kng4BAAJ

[43] Andy Rudoff. 2020. *Reply to 8 byte atomicity & larger store operations*. https://groups.google.com/g/pmem/c/6_5daOuEI00/m/nY_mtKd0CAAJ

[44] Steve Scargall. 2020. *Programming persistent memory: A comprehensive guide for developers*. Springer Nature. https://doi.org/10.1007/978-1-4842-4932-1

[45] Gal Sela, Maurice Herlihy, and Erez Petrank. 2021. Brief announcement: Linearizability: A typo. In *PODC*. https://doi.org/10.1145/3465084.3467944

[46] Gal Sela and Erez Petrank. 2021. Durable queues: The second amendment. *arXiv preprint* (2021). arXiv:2105.08706

[47] SNIA. 2017. *NVM Programming Model (NPM)*. https://snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf

[48] Pivotal Software. [n.d.]. *RabbitMQ*. https://rabbitmq.com

[49] Intel PMDK team. [n.d.]. *Persistent memory programming*. https://pmem.io

[50] Alexander van Renen, Lukas Vogel, Viktor Leis, Thomas Neumann, and Alfons Kemper. 2019. Persistent memory I/O primitives. In *DaMoN*. https://doi.org/10.1145/3329785.3329930

[51] Haris Volos, Andres Jaan Tack, and Michael M Swift. 2011. Mnemosyne: Lightweight persistent memory. In *ASPLOS*. https://doi.org/10.1145/1950365.1950379

[52] Haosen Wen, Wentao Cai, Mingzhe Du, Louis Jenkins, Benjamin Valpey, and Michael L Scott. 2020. Montage: A general system for buffered durably linearizable data structures. *arXiv preprint* (2020). arXiv:2009.13701

[53] Kai Wu, Jie Ren, and Dong Li. 2019. Architecture-aware, high performance transaction for persistent memory. *arXiv preprint* (2019). arXiv:1903.06226

[54] Chaoran Yang and John Mellor-Crummey. 2016. A wait-free queue as fast as fetch-and-add. In *PPoPP*. https://doi.org/10.1145/2851141.2851168

[55] Jian Yang, Juno Kim, Morteza Hoseinzadeh, Joseph Izraelevitz, and Steve Swanson. 2020. An empirical guide to the behavior and use of scalable persistent memory. In *FAST*.

[56] Pantea Zardoshti, Tingzhe Zhou, Yujie Liu, and Michael Spear. 2019. Optimizing persistent memory transactions. In *PACT*. https://doi.org/10.1109/PACT.2019.00025

[57] Yoav Zuriel, Michal Friedman, Gali Sheffi, Nachshon Cohen, and Erez Petrank. 2019. Efficient lock-free durable sets. *PACMPL* 3, OOPSLA (2019). https://doi.org/10.1145/3360554