Tutorial: Adaptive Replication and Partitioning in Data Systems

Brad Glasbergen, Michael Abebe, Khuzaima Daudjee
Cheriton School of Computer Science
University of Waterloo
{bjglasbe,mtabebe,kdaudjee}@uwaterloo.ca

ABSTRACT

To meet growing application demands, distributed data systems replicate and partition data across multiple machines. Replication increases the resource and request processing capabilities of a system by spreading copies of the data across multiple machines, while partitioning splits data across machines to achieve the same objectives. Replication and partitioning present different trade-offs in the form of replication maintenance and multi-machine coordination costs, which system administrators must carefully evaluate. Traditionally, administrators made replication and partitioning decisions based on their understanding of the application workload, which results in suboptimal performance if the system is misconfigured or if the workload changes. However, systems that adaptively employ replication and partitioning can adjust these decisions based on workload observations and predictions, which improves performance and reduces complexity for administrators. In this tutorial, we present an overview of techniques used by systems to adaptively partition and replicate data and services. We focus on the decision-making strategies employed by these systems, and how these decisions are executed in an online environment. Finally, we identify opportunities for research in the area.

ACM Reference Format:
Brad Glasbergen, Michael Abebe, Khuzaima Daudjee. 2018. Tutorial: Adaptive Replication and Partitioning in Data Systems. In *19th International Middleware Conference Tutorials (Middleware '18 Tutorials), December 10–14, 2018, Rennes, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3279945.3279946

1 INTRODUCTION

Driven by enterprise and individual needs, big data applications are placing ever-increasing demands on data systems [3, 29, 43]. For example, it is now common for systems to receive hundreds-of-thousands of requests per second from clients around the world, all of whom expect fast response times [8, 10]. These stringent demands exceed the capabilities of a single machine and therefore require data systems to scale and operate in a distributed environment.

There are two primary means of scaling out systems that manage data: *replication* and *partitioning*. Replication creates copies of data and spreads them across a pool of machines. A client may access any node hosting its desired data to perform computations and retrieve results, which increases the amount of computing power available and may bring data closer to clients. By contrast, partitioning splits data across multiple machines, which enables a request to be processed by any machine that holds the data the request needs, and increases the amount of available storage and compute power.

Although data replication and partitioning can improve the performance of data systems, they also pose many challenges. Administrators must decide which data items to replicate and on which machines these replicas should be placed. Similarly, they must select a partitioning scheme for the data that maximizes parallelism without degrading performance due to excessive communication and coordination among machines. Choosing such a replication and partitioning scheme can be extremely challenging. A poor choice leads to low levels of performance [12], and a good choice for one workload may be poor for another. Compounding this problem, a partitioning or replication scheme may initially perform well but then suffer due to changes in the workload.

Data system applications experience variation in workloads because of changes in external factors, such as human behaviour. For example, popular data items result in hot-spots in systems and cause contention. These hot-spots may change over time as the popularity of items changes or as a result of "follow-the-sun" cycles from geo-distribution of clients [46]. Finally, data systems may suffer from short-term load spikes that occur because of a short burst of increased demand. These dynamic workload effects make it very difficult for a system administrator to select a single data replication or partitioning strategy for their system that works well for a variety of workloads.

Consequently, data systems are increasingly employing adaptive data replication and partitioning techniques. Such systems modify their partitioning and replication schemes *while the system is running* to ensure good performance even when the workload changes. Systems that adaptively replicate and partition data continually decide which data items to replicate and how partitions and replicas should be placed based on the workload. These systems monitor the running workload, extract salient characteristics, predict future requests, and adjust data placements using the discovered information. Given the personnel costs for administrators to determine and perform these actions manually [42], the ubiquitous nature of dynamic workloads [45], and the desire to minimize reaction time to workload change, the time is ripe for both research and industrial communities to investigate and develop truly autonomous, adaptive data systems.

Building adaptive replication and partitioning into data systems presents several challenges from both decision-making and decision-execution perspectives. In this tutorial, we survey the recent work on systems that adaptively replicate and partition data, focusing on the techniques that are used in these systems and highlighting how they address these challenges. We describe the key

Middleware '18 Tutorials, December 10–14, 2018, Rennes, France

ACM ISBN 978-1-4503-6073-9/18/12.
https://doi.org/10.1145/3279945.3279946

features of these systems and discuss research opportunities in the area.

2 TUTORIAL OVERVIEW

The target audience of this tutorial is systems researchers and practitioners that have a basic knowledge of distributed data management. Some knowledge of data replication and partitioning is helpful, but not necessary. For newcomers to the area, this tutorial introduces replication, partitioning and the motivation for applying these techniques adaptively. For researchers that are experienced with replication and partitioning, this tutorial provides a broader view of the common approaches to providing adaptivity in data systems and offers suggestions on how current techniques can be improved. In general, we hope that this tutorial inspires our colleagues from the middleware and systems communities to research and implement adaptivity as a core component in data systems across a variety of data management domains including relational databases, graph data management, stream processing, and distributed storage systems.

We present replication and partitioning techniques from a variety of data management domains and emphasize one or two representative systems from each domain that employ these techniques adaptively. For these systems, we discuss the decision-control framework for determining new replication and partitioning strategies and the underlying mechanisms used in each system. Finally, we discuss trends within this area and present open research challenges and opportunities for adaptive data systems.

The tutorial is divided into four modules: (i) The motivation behind data replication and partitioning, and why adaptivity matters (Section 1) (ii) Adaptive replication (Section 3.1), (iii) Adaptive partitioning (Section 3.2), (iv) Outlook for adaptive data management systems (Section 4).

3 TUTORIAL OUTLINE

3.1 Adaptive Replication

Data replication improves system performance by enabling read operations to be performed on replicas. The replicas may be physically close to a client or may be under less load than the primary copy of data, both of which decrease access latency. Consequently, replication is a popular method used in numerous data management domains [6, 9, 19, 20, 36, 49]. A key challenge is keeping replicas synchronized with the primary data copy. As requests update the primary copy, they must be propagated and applied to the replicas. Deciding which set of data items should be replicated, where the replicas should be placed, which updates should be propagated, and when they should be applied comprise the fundamental design decisions of any replicated data system. By leveraging adaptive replication techniques, systems can perform these decisions on-the-fly with respect to the workload, thereby improving performance and reducing system configuration complexity.

The adaptive data replication (ADR) algorithm [50] is among early work in adaptive replication. ADR formulates its cost functions based on system requirements and the environment, which optimize the number of replicas, their placement, and which updates are propagated to each replica.

At a global scale, distributed systems must place replicas of data around the world such that both access and update latencies are minimized. Data placement is a challenging problem as clients can be spread globally, and as more copies of data are created, the cost of keeping data consistent in the face of updates increases. Sharov and colleagues [41] propose an optimization formulation for leader replica placement. The GPlacer system [51] also addresses replica placement, but generalizes beyond leader placement to other non-leader replica protocols such as Paxos.

Data caching is a particular form of adaptive replication, in the sense that what is replicated, or cached, can adapt based on access history. Predictive caching [7, 34] aims to further improve the performance of cache-based systems by placing data items that will be accessed in the cache ahead of time. An example of such a system is Apollo [14], a middleware cache for database systems that learns database query patterns to predictively cache query results to improve system performance.

Finally, data replication is often used to ensure high-availability and fault tolerance. In distributed storage systems, erasure-coding is often used as a replacement for replication because it can provide lower storage overhead [18, 31]. However, these systems face many of the same design decisions as replicated storage systems. EC-Store [2] is an adaptive erasure-coded storage system that dynamically places and moves data to reduce access latencies. We contrast the data-placement decisions made in EC-Store with those made in replicated storage systems such as C3 [33, 37, 44].

3.2 Adaptive Partitioning

Data partitioning enables systems to scale by splitting data across multiple machines. Traditional partitioning techniques such as hash or range partitioning result in even distributions of the number of data items across machines. However, skew in data item popularity can result in load imbalance among machines. To mitigate this effect, systems aim to adaptively repartition data to ensure load balance while maintaining locality among co-accessed data items.

The L-Store system [26] aims to minimize the number of distributed transactions that are performed in a partitioned database system. L-Store guarantees that transactions are performed at a single site by localizing data items to that site during transaction execution. AdaptCache [5] also focuses on localized processing, partitioning both requests and data so that an application server can find the objects needed to handle a request in their local cache most of the time.

Dynamic workloads change the demand they make on data systems over time. Consequently, elastic systems dynamically adjust their physical resources according to demand. When machines are added or removed from the system according to current workload demand, it is important to repartition data accordingly. The E-Store system [46] incorporates repartitioning into its elastic operations via the Squall system [13]. While E-Store *reacts* to load changes, P-Store [45] *predicts* load demands, elastically scaling the database according to its predictions.

While data systems frequently partition horizontally, that is, on the individual data item level, data can also be partitioned vertically by data item attributes. Database cracking [16, 21, 22, 38, 39] automatically vertically partitions databases based on the presence of

predicate queries. Database cracking highlights a consequence of data partitioning: changes need to be made to data access operators, for example, pushing down *cracked* database operators that operate on vertically partitioned data.

Finally, adaptive data partitioning is applied in many other data management domains to improve system performance. As an example, graph data systems aim to partition data items (vertices and edges) such that the number of machines over which graph traversal operations are executed is minimized [19, 23]. Hermes [32] uses a lightweight dynamic partitioning algorithm that aims to keep load balanced while minimizing the number of edge-cuts and consequently network I/O. Additionally, many systems rely on graph partitioning algorithms to generate partitions by modelling their data as a graph [12, 17, 37]. In stream processing engines, data items can be repartitioned to balance load across the streaming engines [1, 15, 40]. Adaptive partitioning techniques for streaming systems has become an emerging area of research [24, 27, 48].

4 OUTLOOK

Many current systems use data replication and partitioning to improve system performance. However, systems are becoming increasingly adaptive, automatically determining partitioning and replication strategies. These systems adapt by monitoring the system and client workload behaviour, such as the observed load or patterns in data access, and then make decisions based on these observations. In particular, systems adapt by *reacting* to changes in behaviour. However, many of these systems could offer improved performance by making decisions based on *predictions* of future system and client behaviour [4, 35], such as in P-Store [45]. While advances in machine learning have been applied to parameter selection and tuning [30, 47, 52] as well as data structures such as indexes [25], they have not yet been broadly applied to the areas of partitioning and replication. We believe that data systems should leverage advances in workload forecasting [11, 28] to make decisions about partitioning and replication. Put broadly, we pose two questions to the research community: "How should we model workloads in your domain of expertise?" and "If you had an accurate model of your workload in terms of both data accesses and load, how would you design a replication and partitioning scheme for it?". We expect that these questions will spark interest, discussion and debate over the design of future data systems.

5 BIOGRAPHICAL SKETCHES

Brad Glasbergen is a doctoral student at the University of Waterloo. His research interests focus on intelligent data management techniques. Recently, he has worked on predictive caching for relational databases and intelligent transaction routing in distributed data systems. In the past, he has worked for SAP Waterloo on the Database Query Engine team. (Homepage)

Michael Abebe is a doctoral student at the University of Waterloo. He is interested in distributed data management as well as adaptive data systems and their infrastructure. Previously, he has worked at Facebook on data and service infrastructure teams. (Homepage)

Khuzaima Daudjee is a faculty member in the Cheriton School of Computer Science at the University of Waterloo. His research interests are in designing and building large-scale data systems. He is an Associate Editor for Information Systems and IEEE Transactions on Knowledge and Data Engineering. He has been a Visiting Research Scientist at Japan National Institute of Informatics and Visiting Professor at Sapienza University of Rome. He is the recipient of a best paper award at the ACM Symposium on Cloud Computing. (Homepage)

ACKNOWLEDGMENTS

Funding for this project was provided by the Natural Sciences and Engineering Research Council of Canada and the Province of Ontario.

REFERENCES

[1] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. 2003. Aurora: A New Model and Architecture for Data Stream Management. *The VLDB Journal* 12, 2 (Aug. 2003), 120–139. https://doi.org/10.1007/s00778-003-0095-z

[2] Michael Abebe, Khuzaima Daudjee, Brad Glasbergen, and Yuanfeng Tian. 2018. EC-Store: Bridging the Gap Between Storage and Latency in Distributed Erasure Coded Systems. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-5, 2018.*

[3] Divyakant Agrawal, Sudipto Das, and Amr El Abbadi. 2011. Big data and cloud computing: current state and future opportunities. In *EDBT 2011, 14th International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011, Proceedings.* 530–533. https://doi.org/10.1145/1951365.1951432

[4] Mert Akdere, Ugur Çetintemel, Matteo Riondato, Eli Upfal, and Stanley B. Zdonik. 2011. The Case for Predictive Database Systems: Opportunities and Challenges. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings.* 167–174. http://cidrdb.org/cidr2011/Papers/CIDR11_Paper20.pdf

[5] Omar Asad and Bettina Kemme. 2016. AdaptCache: Adaptive Data Partitioning and Migration for Distributed Object Caches. In *Proceedings of the 17th International Middleware Conference (Middleware '16).* ACM, New York, NY, USA, Article 7, 13 pages. https://doi.org/10.1145/2988336.2988343

[6] Magdalena Balazinska, Hari Balakrishnan, Samuel Madden, and Michael Stonebraker. 2005. Fault-tolerance in the Borealis Distributed Stream Processing System. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05).* ACM, New York, NY, USA, 13–24. https://doi.org/10.1145/1066157.1066160

[7] Ivan T. Bowman and Kenneth Salem. 2005. Optimization of Query Streams Using Semantic Prefetching. *ACM Trans. Database Syst.* 30, 4 (Dec. 2005), 1056–1101. https://doi.org/10.1145/1114244.1114250

[8] Nathan Bronson, Zach Amsden, George Cabrera, Prasad Chakka, Peter Dimov, Hui Ding, Jack Ferris, Anthony Giardullo, Sachin Kulkarni, Harry Li, Mark Marchukov, Dmitri Petrov, Lovro Puzar, Yee Jiun Song, and Venkat Venkataramani. 2013. TAO: Facebook's Distributed Data Store for the Social Graph. In *Proceedings of the 2013 USENIX Conference on Annual Technical Conference (USENIX ATC'13).* USENIX Association, Berkeley, CA, USA, 49–60. http://dl.acm.org/citation.cfm?id=2535461.2535468

[9] Emmanuel Cecchet, George Candea, and Anastasia Ailamaki. 2008. Middleware-based Database Replication: The Gaps Between Theory and Practice. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08).* ACM, New York, NY, USA, 739–752. https://doi.org/10.1145/1376616.1376691

[10] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, et al. 2013. Spanner: Google&Rsquo;s Globally Distributed Database. *ACM Trans. Comput. Syst.* 31, 3, Article 8 (Aug. 2013), 22 pages. https://doi.org/10.1145/2491245

[11] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17).* ACM, New York, NY, USA, 153–167. https://doi.org/10.1145/3132747.3132772

[12] Carlo Curino, Evan Jones, Yang Zhang, and Sam Madden. 2010. Schism: A Workload-driven Approach to Database Replication and Partitioning. *Proc. VLDB Endow.* 3, 1-2 (Sept. 2010), 48–57. https://doi.org/10.14778/1920841.1920853

[13] Aaron J. Elmore, Vaibhav Arora, Rebecca Taft, Andrew Pavlo, Divyakant Agrawal, and Amr El Abbadi. 2015. Squall: Fine-Grained Live Reconfiguration for Partitioned Main Memory Databases. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15).* ACM, New York, NY, USA, 299–313. https://doi.org/10.1145/2723372.2723726

[14] Brad Glasbergen, Michael Abebe, Khuzaima Daudjee, Scott Foggo, and Anil Pacaci. 2018. Apollo: Learning Query Correlations for Predictive Caching in Geo-Distributed Systems. In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.* 253–264. https://doi.org/10.5441/002/edbt.2018.23

[15] Lukasz Golab and M. Tamer Özsu. 2003. Issues in Data Stream Management. *SIGMOD Rec.* 32, 2 (June 2003), 5–14. https://doi.org/10.1145/776985.776986

[16] Felix Halim, Stratos Idreos, Panagiotis Karras, and Roland H. C. Yap. 2012. Stochastic Database Cracking: Towards Robust Adaptive Indexing in Main-memory Column-stores. *Proc. VLDB Endow.* 5, 6 (Feb. 2012), 502–513. https://doi.org/10.14778/2168651.2168652

[17] Katja Hose and Ralf Schenkel. 2013. WARP: Workload-aware replication and partitioning for RDF. In *Workshops Proceedings of the 29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013.* 1–6. https://doi.org/10.1109/ICDEW.2013.6547414

[18] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. 2012. Erasure coding in windows azure storage. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12).* 15–26.

[19] Jiewen Huang and Daniel J. Abadi. 2016. Leopard: Lightweight Edge-oriented Partitioning and Replication for Dynamic Graphs. *Proc. VLDB Endow.* 9, 7 (March 2016), 540–551. https://doi.org/10.14778/2904483.2904486

[20] Yixiu Huang, A. Prasad Sistla, and Ouri Wolfson. 1994. Data Replication for Mobile Computers. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, USA, May 24-27, 1994.* 13–24. https://doi.org/10.1145/191839.191845

[21] Stratos Idreos, Martin L. Kersten, and Stefan Manegold. 2007. Database Cracking. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings.* 68–78. http://cidrdb.org/cidr2007/papers/cidr07p07.pdf

[22] Stratos Idreos, Stefan Manegold, Harumi Kuno, and Goetz Graefe. 2011. Merging What's Cracked, Cracking What's Merged: Adaptive Indexing in Main-memory Column-stores. *Proc. VLDB Endow.* 4, 9 (June 2011), 586–597. https://doi.org/10.14778/2002938.2002944

[23] George Karypis and Vipin Kumar. 1998. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* 20, 1 (Dec. 1998), 359–392. https://doi.org/10.1137/S1064827595287997

[24] Nikos R. Katsipoulakis, Alexandros Labrinidis, and Panos K. Chrysanthis. 2017. A Holistic View of Stream Partitioning Costs. *Proc. VLDB Endow.* 10, 11 (Aug. 2017), 1286–1297. https://doi.org/10.14778/3137628.3137639

[25] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The Case for Learned Index Structures. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18).* ACM, New York, NY, USA, 489–504. https://doi.org/10.1145/3183713.3196909

[26] Qian Lin, Pengfei Chang, Gang Chen, Beng Chin Ooi, Kian-Lee Tan, and Zhengkui Wang. 2016. Towards a Non-2PC Transaction Management in Distributed Database Systems. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16).* ACM, New York, NY, USA, 1659–1674. https://doi.org/10.1145/2882903.2882923

[27] Federico Lombardi, Leonardo Aniello, Silvia Bonomi, and Leonardo Querzoni. 2018. Elastic symbiotic scaling of operators and resources in stream processing systems. *IEEE Transactions on Parallel and Distributed Systems* 29, 3 (2018), 572–585.

[28] Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J. Gordon. 2018. Query-based Workload Forecasting for Self-Driving Database Management Systems. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18).* ACM, New York, NY, USA, 631–645. https://doi.org/10.1145/3183713.3196908

[29] Sam Madden. 2012. From Databases to Big Data. *IEEE Internet Computing* 16, 3 (May 2012), 4–6. https://doi.org/10.1109/MIC.2012.50

[30] Ashraf Mahgoub, Paul Wood, Sachandhan Ganesh, Subrata Mitra, Wolfgang Gerlach, Travis Harrison, Folker Meyer, Ananth Grama, Saurabh Bagchi, and Somali Chaterji. 2017. Rafiki: A Middleware for Parameter Tuning of NoSQL Datastores for Dynamic Metagenomics Workloads. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference (Middleware '17).* ACM, New York, NY, USA, 28–40. https://doi.org/10.1145/3135974.3135991

[31] Subramanian Muralidhar, Wyatt Lloyd, Sabyasachi Roy, Cory Hill, Ernest Lin, Weiwen Liu, Satadru Pan, Shiva Shankar, Viswanath Sivakumar, Linpeng Tang, et al. 2014. f4: Facebook's warm BLOB storage system. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14).* 383–398.

[32] Daniel Nicoara, Shahin Kamali, Khuzaima Daudjee, and Lei Chen. 2015. Hermes: Dynamic Partitioning for Distributed Social Network Graph Databases. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.* 25–36. https://doi.org/10.5441/002/edbt.2015.04

[33] João Paiva, Pedro Ruivo, Paolo Romano, and Luís Rodrigues. 2014. AutoPlacer: Scalable Self-Tuning Data Placement in Distributed Key-Value Stores. *ACM Trans. Auton. Adapt. Syst.* 9, 4, Article 19 (Dec. 2014), 30 pages. https://doi.org/10.1145/2641573

[34] Mark Palmer and Stanley B. Zdonik. 1991. *Fido: A Cache That Learns To Fetch.* Brown University, Providence, RI, USA.

[35] Andrew Pavlo, Gustavo Angulo, Joy Arulraj, Haibin Lin, Jiexi Lin, Lin Ma, Prashanth Menon, Todd C. Mowry, Matthew Perron, Ian Quah, Siddharth Santurkar, Anthony Tomasic, Skye Toor, Dana Van Aken, Ziqi Wang, Yingjun Wu, Ran Xian, and Tieying Zhang. 2017. Self-Driving Database Management Systems. In *CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings.* http://cidrdb.org/cidr2017/papers/p42-pavlo-cidr17.pdf

[36] Christian Plattner and Gustavo Alonso. 2004. Ganymed: Scalable Replication for Transactional Web Applications. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware (Middleware '04).* Springer-Verlag, Berlin, Heidelberg, 155–174. http://dl.acm.org/citation.cfm?id=1045658.1045671

[37] Abdul Quamar, K. Ashwin Kumar, and Amol Deshpande. 2013. SWORD: Scalable Workload-aware Data Placement for Transactional Workloads. In *Proceedings of the 16th International Conference on Extending Database Technology (EDBT '13).* ACM, New York, NY, USA, 430–441. https://doi.org/10.1145/2452376.2452427

[38] Felix Martin Schuhknecht, Alekh Jindal, and Jens Dittrich. 2013. The Uncracked Pieces in Database Cracking. *Proc. VLDB Endow.* 7, 2 (Oct. 2013), 97–108. https://doi.org/10.14778/2732228.2732229

[39] Felix Martin Schuhknecht, Alekh Jindal, and Jens Dittrich. 2016. An Experimental Evaluation and Analysis of Database Cracking. *The VLDB Journal* 25, 1 (Feb. 2016), 27–52. https://doi.org/10.1007/s00778-015-0397-y

[40] Mehul A. Shah, Joseph M. Hellerstein, Sirish Chandrasekaran, and Michael J. Franklin. 2003. Flux: An Adaptive Partitioning Operator for Continuous Query Systems. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India.* 25–36. https://doi.org/10.1109/ICDE.2003.1260779

[41] Artyom Sharov, Alexander Shraer, Arif Merchant, and Murray Stokely. 2015. Take Me to Your Leader!: Online Optimization of Distributed Storage Configurations. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 1490–1501. https://doi.org/10.14778/2824032.2824047

[42] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. 2007. The End of an Architectural Era: (It's Time for a Complete Rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07).* VLDB Endowment, 1150–1160. http://dl.acm.org/citation.cfm?id=1325851.1325981

[43] Michael Stonebraker, Sam Madden, and Pradeep Dubey. 2013. Intel "Big Data" Science and Technology Center Vision and Execution Plan. *SIGMOD Rec.* 42, 1 (May 2013), 44–49. https://doi.org/10.1145/2481528.2481537

[44] Lalith Suresh, Marco Canini, Stefan Schmid, and Anja Feldmann. 2015. C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection. In *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15).* USENIX Association, Berkeley, CA, USA, 513–527. http://dl.acm.org/citation.cfm?id=2789770.2789806

[45] Rebecca Taft, Nosayba El-Sayed, Marco Serafini, Yu Lu, Ashraf Aboulnaga, Michael Stonebraker, Ricardo Mayerhofer, and Francisco Andrade. 2018. P-Store: An Elastic Database System with Predictive Provisioning. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18).* ACM, New York, NY, USA, 205–219. https://doi.org/10.1145/3183713.3190650

[46] Rebecca Taft, Essam Mansour, Marco Serafini, Jennie Duggan, Aaron J. Elmore, Ashraf Aboulnaga, Andrew Pavlo, and Michael Stonebraker. 2014. E-store: Fine-grained Elastic Partitioning for Distributed Transaction Processing Systems. *Proc. VLDB Endow.* 8, 3 (Nov. 2014), 245–256. https://doi.org/10.14778/2735508.2735514

[47] Dana Van Aken, Andrew Pavlo, Geoffrey J. Gordon, and Bohan Zhang. 2017. Automatic Database Management System Tuning Through Large-scale Machine Learning. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17).* ACM, New York, NY, USA, 1009–1024. https://doi.org/10.1145/3035918.3064029

[48] Shivaram Venkataraman, Aurojit Panda, Kay Ousterhout, Michael Armbrust, Ali Ghodsi, Michael J. Franklin, Benjamin Recht, and Ion Stoica. 2017. Drizzle: Fast and Adaptable Stream Processing at Scale. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17).* ACM, New York, NY, USA, 374–389. https://doi.org/10.1145/3132747.3132750

[49] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, and G. Alonso. 2000. Understanding Replication in Databases and Distributed Systems. In *Proceedings of the The 20th International Conference on Distributed Computing Systems (ICDCS 2000) (ICDCS '00).* IEEE Computer Society, Washington, DC, USA, 464–. http://dl.acm.org/citation.cfm?id=850927.851782

[50] Ouri Wolfson, Sushil Jajodia, and Yixiu Huang. 1997. An Adaptive Data Replication Algorithm. *ACM Trans. Database Syst.* 22, 2 (June 1997), 255–314. https://doi.org/10.1145/249978.249982

[51] Victor Zakhary, Faisal Nawab, Divy Agrawal, and Amr El Abbadi. 2018. Global-Scale Placement of Transactional Data Stores. In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.* 385–396. https://doi.org/10.5441/002/edbt.2018.34

[52] Yuqing Zhu, Jianxun Liu, Mengying Guo, Yungang Bao, Wenlong Ma, Zhuoyue Liu, Kunpeng Song, and Yingchun Yang. 2017. BestConfig: tapping the performance potential of systems via automatic configuration tuning. In *Proceedings of the 2017 Symposium on Cloud Computing, SoCC 2017, Santa Clara, CA, USA, September 24 - 27, 2017*. 338–350. https://doi.org/10.1145/3127479.3128605