Dixin Tang

Postdoctoral Scholar
University of California, Berkeley
387 Soda Hall, Berkeley, CA 94720

(+1) 510-365-9300
totemtang@berkeley.edu
https://people.eecs.berkeley.edu/~totemtang/

Research Areas

User-Centered Data Management, Query Processing and Optimizations, Cloud Databases

Employment

2021-present Postdoctoral Scholar - University of California, Berkeley
Advisor: Aditya G. Parameswaran, Associate Professor

Education

2015-2020 Ph.D. in Computer Science - University of Chicago
Advisor: Aaron J. Elmore, Associate Professor

2011-2014 M.S. in Computer Science - Institute of Computing Technology, Chinese Academy of Sciences
Advisor: Wei Li, Associate Professor

2007-2011 B.S. in Software Engineering - Huazhong University of Science & Technology

Manuscripts

M1. Transactional Panorama: A Conceptual Framework for User Perception in Analytical Visual Interfaces
Dixin Tang, Alan Fekete, Indranil Gupta, Aditya G. Parameswaran
Under Revision at VLDB

M2. Efficient and Compact Spreadsheet Formula Graphs
Dixin Tang, Fanchao Chen, Christopher De Leon, Tana Wattanawaroon, Jeaseok Yun, Srinivasan Seshadri, Aditya G. Parameswaran
Under Revision at ICDE

M3. FormS: A Python Library for Scalable Spreadsheet Formula Execution
Dixin Tang*, Zixuan Yi*, Connor Lien, Ryan Sun, Aditya G. Parameswaran
In Preparation (*Equal contribution)

M4. ShiftXplain: A Data Shift Explanation Framework
Yong Wang, Shreya Shankar, **Dixin Tang**, Guoliang Li, Aditya G. Parameswaran
In Preparation

Recent Publications

P1. Flexible Rule-Based Decomposition and Metadata Independence in Modin: A Parallel Dataframe System
Devin Petersohn*, **Dixin Tang***, Rehan Durrani, Areg Melik-Adamyan, Joseph E. Gonzalez, Anthony D. Joseph, Aditya G. Parameswaran
VLDB 2022 (*Equal contribution)

P2. Lux: Always-on Visualization Recommendations for Exploratory Dataframe Workflows
Doris Jung-Lin Lee, **Dixin Tang**, Kunal Agarwal, Thyne Boonmark, Caitlyn Chen, Jake Kang, Ujjaini Mukhopadhyay, Jerry Song, Micah Yong, Marti A. Hearst, Aditya G. Parameswaran
VLDB 2022

P3. Enhancing the Interactivity of Dataframe Queries by Leveraging Think Time
Doris Xin, Devin Petersohn, **Dixin Tang**, Yifan Wu, Joseph E. Gonzalez, Joseph M. Hellerstein, Anthony D. Joseph, Aditya G. Parameswaran
IEEE Data Eng. Bull. 2021

P4. Resource-Efficient Shared Query Execution via Exploiting Time Slackness
Dixin Tang, Zechao Shang, William Ma, Aaron J. Elmore, Sanjay Krishnan
SIGMOD 2021

P5. CIAO: An Optimization Framework for Client-Assisted Data Loading
Cong Ding, **Dixin Tang**, Xi Liang, Aaron J. Elmore, Sanjay Krishnan
ICDE 2021, Short Paper

P6. CrocodileDB in Action: Resource-Efficient Query Execution by Exploiting Time Slackness
Dixin Tang, Zechao Shang, Aaron J. Elmore, Sanjay Krishnan, Michael J. Franklin
VLDB 2020, Demo

P7. Thrifty Query Execution via Incrementability
Dixin Tang, Zechao Shang, Aaron J. Elmore, Sanjay Krishnan, Michael J. Franklin
SIGMOD 2020

P8. CrocodileDB: Efficient Database Execution through Intelligent Deferment
Zechao Shang, Xi Liang, **Dixin Tang**, Cong Ding, Aaron J. Elmore, Sanjay Krishnan, Michael J. Franklin
CIDR 2020

P9. Intermittent Query Processing
Dixin Tang, Zechao Shang, Aaron J. Elmore, Sanjay Krishnan, Michael J. Franklin
VLDB 2019

P10. Socrates: The New SQL Server in the Cloud
Panagiotis Antonopoulos, Alex Budovski, Cristian Diaconu, Alejandro Hernandez Saenz, Jack Hu, Hanuma Kodavalla, Donald Kossmann, Umar Farooq Minhas, Naveen Prakash, Hugh Qu, Chaitanya Sreenivas Ravella, Krystyna Reisteter, Sheetal Shroti, **Dixin Tang**, Vikram Wakade
SIGMOD 2019

P11. Toward Coordination-Free and Reconfigurable Mixed Concurrency Control
Dixin Tang, Aaron J. Elmore
USENIX'ATC 2018

P12. Adaptive Concurrency Control: Despite the Looking Glass, One Concurrency Control Does Not Fit All
Dixin Tang, Hao Jiang, Aaron J. Elmore
CIDR 2017

Earlier Publications (Before Ph.D.)

P13. A Case Study of Optimizing Big Data Analytical Stacks Using Structured Data Shuffling
Dixin Tang, Taoying Liu, Rubao Lee, Hong Liu, Wei Li
BigData Congress 2016

P14. SparkArray: An Array-Based Scientific Data Management System Built on Apache Spark
Wenjuan Wang, Taoying Liu, **Dixin Tang**, Hong Liu, Wei Li, Rubao Lee
NAS 2016

P15. A Case Study of Optimizing Big Data Analytical Stacks Using Structured Data Shuffling
Dixin Tang, Taoying Liu, Rubao Lee, Hong Liu, Wei Li
CLUSTER 2015, Short Paper

P16. RHJoin: A Fast and Space-Efficient Join Method for Log Processing in MapReduce
Dixin Tang, Taoying Liu, Hong Liu, Wei Li
ICPADS 2014

P17. Optimizing the Join Operation on Hive to Accelerate Cross-Matching in Astronomy
Liang Li, **Dixin Tang**, Taoying Liu, Hong Liu, Wei Li, Chenzhou Cui
IPDPS Workshops 2014

Professional Services

Conference Program Committee: SIGMOD'22, SIGMOD'22 (Demo Track), SIGMOD'23, EDBT'24

Conference Reviewer: IEEE VIS'21

Conference Session Chair: SIGMOD'22

Journal Reviewer: VLDB Journal (2022), Distributed and Parallel Databases Journal (2022)

Mentoring Experiences

Mentee	Affiliation	Period	Next Appointment	Authorship
Zixuan Yi	Undergrad at Tsinghua University	Feb. 2022-Now		Co-first author of M3
Fanchao Chen	Undergrad at Fudan University	Jun. 2022-Now		Second author of M2
Connor Lien	Undergrad at UC Berkeley	Jan. 2022-Now		Co-author of M3
Ryan Sun	Undergrad at UC Berkeley	Sep. 2022-Now		Co-author of M3
Avinash Rao	Undergrad at UC Berkeley	Jan. 2022-Now		
Joshua Wu	Master's student at UC Berkeley	Jan. 2022-Now		
Todd Yu	Master's student at UC Berkeley	Sep. 2021-Now		
Christina Fan	Undergrad at UC Berkeley	Sep. 2021-Now		
Kunal Agarwal	Master's student at UC Berkeley	Jun. 2021-Jun. 2022	SDE at Stripe	Co-author of P2
Jonathan Yun	Undergrad at UC Berkeley	Mar. 2021-Jun. 2022	SDE at Oracle	Co-author of M2
Chris De Leon	Master's student at UC Berkeley	Mar. 2021-Jun. 2022	SDE at Anchain.AI	Co-author of M2
William Ma	Master's student at UChicago	Jun. 2020-Oct. 2020	SDE at Apple	Co-author of P4
Cong Ding	Undergrad at Peking University	Jun. 2019-Oct. 2020	PhD at UW-Madison	First author of P5

Honors & Awards

2018	USENIX ATC'18 Student Travel Grant
2016	University Unrestricted (UU) Fellowship - The University of Chicago
2016	CERES 1st year Graduate Research Award - The University of Chicago

Teaching Assistants

Winter 2020	CMSC 23500 - Introduction to Database Systems
Winter 2019	CMSC 23500 - Introduction to Database Systems
Winter 2018	CMSC 23500 - Introduction to Database Systems
Winter 2017	CMSC 23500 - Introduction to Database Systems
Spring 2016	MPCS 52040 - Distributed Systems
Fall 2015	MPCS 51040 - C Programming

Industry Experience

- **Internship at Microsoft Research** June 2018-Sep. 2018
  Project: Benchmarking Socrates Mentor: Umar Farooq Minhas
  Socrates is a new cloud-native database that decouples computation from storage. My internship involved testing the new database architecture of Socrates in an industrial setting, understanding its performance bottlenecks, and proposing optimization opportunities.

Recent Research Projects (Detailed Descriptions)

R1. **FormS: A Python Library for Scalable Spreadsheet Formula Execution** Mar. 2022-Present
Code: `https://github.com/forms-org/forms`
FormS is a Python library for executing spreadsheet formulae in a distributed execution framework. In FormS, users write a list of formulae via a formula template based on autofill rules, as widely supported by today's spreadsheet systems. For example, applying a template `SUM(A1:A2,B$1:B2)` to a column will generate [`SUM(A1:A2,B$1:B2)`, `SUM(A2:A3,B$1:B3)`, $\cdots$], where the first range is generated similar to a sliding window while the second one is similar to an expanding window in databases. While a formula template is similar to a window operator in databases, efficiently executing a list of formulae is challenging due to two unique semantics in spreadsheets: 1) each spreadsheet function may accept multiple windows with different sizes and types; 2) spreadsheets support many functions that are not optimized by databases (e.g., `SUMIF` function) and require new optimizations to execute them efficiently. To address these challenges, I proposed novel logical and physical rewriting rules for executing a list of formulae efficiently in parallel. FormS now has support for over 50 popular spreadsheet functions; I am mentoring a few undergrads to support more functions for a public release.

R2. **Transactional Panorama: Enhancing User Perception in Analytical Interfaces** Sep. 2021-Present
Code: `https://github.com/transactional-panorama/TP`
I developed transactional panorama, a conceptual framework for user perception when the visualizations in a visual analytical interface (e.g., a dashboard or spreadsheet) are being refreshed. In such an interface, it is common for users to modify the source data or filters to explore different visualization results. With large datasets, it takes a long time to refresh the visualization results while users continue to explore the results simultaneously. In this context, existing tools either (i) hide away results that haven't been updated, hindering exploration; (ii) make the updated results immediately available to the user (on the same screen as old results), leading to confusion and incorrect insights; or (iii) present old—and therefore stale—results to the user during the update. I developed transactional panorama to discover new options for users and help users make appropriate trade-offs between the properties guaranteed for the visual results and the performance for presenting the updated results. Transactional panorama adopts database transactions to jointly model the system refreshing the visualization results and the user interacting with them, and considers three properties that are important for user perception: visibility (allowing users to continuously explore results), consistency (ensuring that results presented are from the same version of the data or filters), and monotonicity (making sure that results don't "go back in time"). I characterized all feasible property combinations, formally proved their relative orderings for various performance criteria (e.g., the total time when the user sees stale results), and discussed their use cases. With transactional panorama, the user can explore visual results with desired properties guaranteed while the visual interface is being refreshed.

R3. **Taco: Efficient and Compact Spreadsheet Formula Graphs** Jan. 2021-Present
Code: `https://github.com/taco-org/taco`
Taco is a framework for efficiently compressing, querying, and maintaining spreadsheet formula graphs to reduce the response time of spreadsheets. A formula graph is adopted to track the dependencies across spreadsheet formulae. When a spreadsheet cell is modified, the spreadsheet system needs to query the formula graph to find its dependents and calculate new formula results. Identifying dependents quickly is key to ensuring that users don't see stale or inconsistent results, and also helps spreadsheets return control early to users. Therefore, I proposed Taco to compress the formula graph by leveraging a key property, tabular locality, which means that cells close to each other are likely to have similar formula structures. Based on the analysis of real-world spreadsheets, I identified five tabular locality-based patterns and designed novel algorithms for querying the compressed representation without decompression. Our experiments on real-world spreadsheets show that the speedup of Taco over Excel on finding dependents is up to 632×.

R4. **Data Shift Explaination** Sep. 2021-Present
ShiftXplain is a framework for explaining data shift. Data shift is ubiquitous in real-world datasets due to the natural evolution in the underlying data relationships and patterns, often leading to a degradation of the performance of data-dependent applications (e.g., an ML model). ShiftXplain explains the data shift between two datasets using a conjunction of predicates and proposes a novel metric, shiftIndex, to capture both general and unique shift patterns. More importantly, shiftIndex is partially monotonic and bounded, which is leveraged by our search algorithm to efficiently prune the search space without sacrificing the explanation quality.

R5. **Modin: A Scalable Dataframe System** Jan. 2021-Present

Code: `https://github.com/modin-project/modin`

I led the effort in developing two key techniques for efficient parallel execution and metadata management in Modin. Pandas is a popular dataframe library widely embraced by data scientists and has been the defacto tool for doing exploratory data analysis in Python. However, building a scalable dataframe system that maintains the unique semantics of dataframes (e.g., supporting mixed types of data in a column and requiring a specific row order for the output dataframe of a function) and supports a large number of pandas functions (i.e., over 600) is challenging. Modin addresses the challenges by mapping pandas functions to 15 core operators, and efficiently parallelizing the core operators and managing the associated metadata (e.g., the type information per column). Specifically, I formally developed decomposition rules that decompose the execution of a core operator in row-wise, column-wise, and cell-wise ways to parallelize the core operator while maintaining its unique semantics. I then developed rewriting rules to choose the decomposition rules for efficient parallel execution. For efficient metadata management, I proposed metadata independence, a technique that always logically maintains the metadata and lazily materializes the metadata when necessary. Modin has been adopted by many data scientists to accelerate their pandas execution. It has more than 7K daily downloads and 8K stars on GitHub.

R6. **Lux: Always-on Visualization Recommendations** Jan. 2021-Present

Code: `https://github.com/lux-org/lux`

Lux is a visualization recommendation tool to reduce the programming overhead for generating visualizations. It proposes a novel data-centered intent language that allows users to use attributes and filters to specify the portion of data of interest without having to consider the visualization encodings, which are inferred by Lux automatically. In addition, Lux models the recommendation process as a state machine, where the state space is defined by the possible combinations of attributes and filters and each recommendation moves users from the current state to an adjacent one. For example, if a user specifies the intent on two attributes and we consider the attribute space, Lux will visualize the relationship between the two attributes (e.g., using a scatter plot) and additionally consider the visualizations that remove, add, or swap one of the original two attributes as candidates. For the candidate visualizations to be recommended, Lux ranks them based on pre-defined interestingness scores and choosees the top k visualizations, where k is configurable. Similar recommendations can be applied to the filter space. Lux is integrated into the pandas dataframe workflow, where users visualize their dataframe by simply printing the dataframe. Lux has 400K downloads and 4.2K stars on Github, and is adopted in many domains, including medicine, education, finance, and more.

R7. **Client-Assisted Data Loading** July 2019-Dec. 2020

Data loading is time-consuming due to type parsing, integrity checking, and maintaining data structures. I mentored an undergraduate to leverage lazy data loading to make data quickly available without heavily sacrificing query performance. The idea is actively pushing predicates of prospective queries into the clients (e.g. edge devices) to generate bit-vectors that indicate whether a tuple is valid for a predicate. The system leverages the bit vectors to partially load data that is most frequently accessed by prospective queries. When queries access unloaded data, the system uses the bit-vectors as an index to accelerate the query execution by skipping irrelevant tuples.

R8. **CrocodileDB: Resource-Efficient Database Execution** Nov. 2017-Mar. 2021

Code: `https://github.com/orgs/crocodiledb/repositories`

Scalable data systems, while performant, consume many computing and memory resources, introducing a high monetary cost if run on the cloud. I built a resource-efficient database, CrocodileDB, to reduce resource usage while meeting a performance goal. CrocodileDB supports scheduled queries over a stream of tuples (e.g., ETL jobs or recurring dashboard reports) and the performance goal is the maximally allowed query latency, which is defined as the time between when the full data is ready for one scheduled query (i.e., the last tuple arrives) and the query result is computed. The core idea for reducing resource consumption is regarding the performance goal as a time slackness and designing novel system strategies to exploit the slackness, including 1) selectively executing parts of a scheduled query lazily, 2) judiciously deciding the queries to share, and 3) intelligently choosing the intermediate states (e.g., a hash table for a hash join) to maintain in memory. Parts of the techniques in CrocodileDB are adopted in a system in Alibaba for reducing the resource consumption of recurring jobs for daily reports.

Specifically, I developed incrementability-aware query processing, or InQP, for reducing the CPU consumption for one scheduled query. To reduce the query latency and meet the performance goal, incremental execution is employed to compute the partial results early and incrementally maintain the

previous results. The more eagerly we incrementally execute a query (e.g., perform one incremental execution for each new tuple), the higher query work and subsequent CPU consumption there will be, mainly because output tuples in earlier executions may be removed by later executions, increasing the query work. Interestingly, eager incremental execution of different parts of a query will increase different amounts of query work for the same amount of reduced latency. Therefore, I proposed incrementability, a novel metric for quantifying the cost-effectiveness of incremental execution for different parts of a query, and developed an algorithm for incrementally executing different sub-queries at different levels of eagerness based on their incrementability to reduce total query work while meeting the performance goal.

Next, I developed iShare, which selectively shares queries that execute on the same data and have different performance goals. Shared execution eliminates redundant computation to save CPU cycles. However, shared execution for different performance goals requires the shared plan to meet the hardest goal (i.e., the lowest latency constraint) and forces some participating queries to run more eagerly. Eager incremental execution increases total query work, which may offset the benefits of shared execution. iShare considers the two factors together and finds efficient query plans that reduce redundant work across concurrent queries and avoid the cost of eager incremental execution.

Finally, I developed intermittent query processing, or IQP, for reducing memory consumption. Our observation is that the new data for a scheduled query may arrive intermittently, rather than continuously in some cases. Therefore, there may be a long time when the query does not incorporate new data. During this period, IQP selectively discards parts of the intermediate states to reduce memory consumption with respect to a memory budget and recomputes them from other saved intermediate states when necessary. I proposed an algorithm to decide which intermediate states to discard to best reduce the query latency based on the predictive information about the new data, such as the estimated size and distribution of the base tables having new data.

R9. **Adaptive Concurrency Control for Main-Memory Databases** Sep. 2015-Nov. 2017
Code: `https://github.com/totemtang/cc-testbed`

I built a main-memory database that adaptively mixes multiple concurrency control protocols, where each protocol is optimized for a different type of workload. I implemented multiple protocols, developed a machine learning model to predict the ideal protocol for a workload, and developed a mediated protocol for switching protocols online. Our experimental results show that this approach has much higher transaction throughput than the best single protocol under varied workloads.

Earlier Projects (Before Ph.D.)

R10. **Structured Data Shuffling for Big Data Analytical Stacks** Nov. 2013-Jan. 2015

We build a structured data shuffling procedure that can leverage the semantics of SQL queries to apply efficient compression algorithms and discard unnecessary data during data shuffling.

R11. **A Fast and Space-Efficient Join Method for Log Processing in MapReduce** Sep. 2012-Nov. 2013

We design a join method that achieves high query performance with a small extra storage cost for log processing. It shuffles the log table to avoid huge storage consumption and optimizes the shuffle procedure to achieve high query performance.

Referees

Name Aditya G. Parameswaran
Affiliation University of California, Berkeley
Position Associate Professor
Contact adityagp@eecs.berkeley.edu

Name Aaron J. Elmore
Affiliation University of Chicago
Position Associate Professor
Contact aelmore@cs.uchicago.edu

Name Michael J. Franklin
Affiliation University of Chicago
Position Full Professor, Liew Family Chairman of Computer Science
Contact mjfranklin@uchicago.edu

Name Sanjay Krishnan
Affiliation University of Chicago
Position Assistant Professor
Contact skr@cs.uchicago.edu

Name Indranil Gupta
Affiliation University of Illinois Urbana-Champaign
Position Full Professor
Contact indy@illinois.edu

Name Umar Farooq Minhas
Affiliation Apple Knowledge Platform
Position Engineering Leader
Contact umarfm13@gmail.com

Name Wei Li
Affiliation Institute of Computing Technology, Chinese Academy of Sciences
Position Associate Professor
Contact liwei@ict.ac.cn