# Dixin Tang


| | | | |
|---|---|---|---|
| **Address** | 5730 S Ellis Ave, JCL 299<br>Chicago, IL 60637 | **Homepage**<br>**Email** | `people.cs.uchicago.edu/~totemtang`<br>totemtang@uchicago.edu |


## Research Areas

Query Processing/Optimization, Cloud Database, Transaction Processing

## Education

| | |
|---|---|
| 2015-present | Ph.D. Candidate in Computer Science - University of Chicago |
| | Advisor: Aaron Elmore |
| 2011-2014 | M.S. in Computer Science - Institute of Computing Technology, Chinese Academy of Sciences |
| | Advisor: Wei Li |
| 2007-2011 | B.S. in Software Engineering - Huazhong University of Science & Technology |

## Publications at UChicago

- 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**

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

- 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**

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

- 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**

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

- 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

- 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**

- 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**

- 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**

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

- 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**

## Industrial 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 job is to test the new database architecture of Socrates in the industrial setting, understand its performance bottlenecks, and propose optimization opportunities.

## System Building Experience

- **Multi-Query Optimization in SparkSQL**

  I extend SparkSQL to support multi-query optimization for the full TPC-H test suite. The framework takes Dataframe queries as input and generates a query plan that shares the execution of all submitted queries. I enhance each intermediate tuple during query execution to additionally include the query IDs that the tuple is valid to and modify SparkSQL operators to leverage this information to support shared query execution.

- **Complex Materialized View Maintenance in SparkSQL**

  I extend SparkSQL to support maintaining materialized views with deletes and updates. The system supports complex materialized views that are beyond select-project-join-aggregate queries. Specifically, the supported operators include select, project, aggregate, sort, materialize, distinct, and joins (inner, outer, and anti join), and there is no ordering limit about these operators.

- **Intermediate State Management in PostgreSQL**

  I extend PostgreSQL to support materialized view maintenance with intermediate state management. The modified PostgreSQL can selectively discard some intermediate states to save memory resources. When a discarded state is required to process new data, the system can rebuild this state from saved intermediate states to quickly update the materialized view.

- **Adaptive Concurrency Control in Main-Memory Database**

  I develop a key-value main-memory database which hosts several concurrency control protocols including partition-based concurrency control, two-phase locking, and optimistic concurrency control protocols. The database uses a machine learning model to adaptively apply a protocol to a partition of the database and reconfigures protocols in response to workload changes.

## Mentoring Experience

- **Cong Ding, Undergraduate** July 2019-Present

  I mentor Cong Ding on client-assisted data loading project, which exploits partial data loading to make data quickly available, but not heavily sacrifice query performance. The research paper is under preparation.

## Research Projects at UChicago

- **Incrementability-aware Shared Query Execution** Mar. 2020-Present

  This project studies shared execution across standing queries that access the same data, but have different performance requirements. Shared execution eliminates redundant work across queries to reduce computing resources. However, the shared plan needs to meet the highest performance requirement (i.e. the lowest query latency), which forces some participating standing queries to run more eagerly. Eager incremental executions increase total work which may offset the benefits of shared execution. This project considers the two factors together and finds efficient query plans that reduce redundant work across concurrent queries and also avoid the cost of eager incremental executions.

- **Client-assisted Data Loading (mentoring project)** July 2019-Present

  Data loading is a time-consuming process. I mentor an undergraduate student to study lazily loading data to make data available earlier, but not heavily sacrifice query performance. I advise the student to redesign the data loading process by 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. These bit vectors are used 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.

- **CrocodileDB: Resource-efficient Database Execution** July 2019-Present

  CrocodileDB is a new database for processing dynamic datasets (e.g. streaming tuples). It exploits intelligent deferment to achieve low resource consumption and high query performance at the same time. In CrocodileDB, users can trade-off between resource consumption and query performance by specifying a maximally allowed time slackness between the data is ready and the corresponding query result is available to users. Our system integrates the slackness constraints along with information about new data and query structures into the query optimizer to unlock new resource-efficient query execution plans.

- **Incrementability-aware Query Processing** Mar. 2019-June 2020

  This project studies how to efficiently maintain non-positive standing queries. While starting a query early and incrementally maintain the query result can reduce the query latency, it increases total query work (i.e. CPU cycles), especially when the database eagerly maintains non-positive queries, because output tuples in earlier executions may be removed by later executions. Observing that incremental executions for different parts of a query increase different amounts of total work for the same reduced latency, I define a metric, *incrementability* to quantify the cost-effectiveness of incremental executions. Our system leverages this metric to decide the execution frequencies of different parts of a query, which is optimized to reduce total query work and also achieve low query latency.

- **Intermittent Query Processing** Dec. 2017-Sep. 2019

  This project studies how to maintain a standing query over an incomplete data set with the remaining data arriving in an intermittent yet predictable way. To process this data arrival pattern, we propose *Intermittent Query Processing*(IQP) that does not keep the query active all the time but releases some memory resources when there is no data to process. When the query has no data to process, IQP leverages the information about incoming data to selectively keeps a subset of intermediate states that are most useful for processing new data within a memory budget. Therefore, IQP achieves low latency of updating query results with limited memory consumption.

- **Adaptive Concurrency Control for Main-memory Database** Sep. 2015-Nov. 2017

  I build a main-memory database that supports adaptively mixing multiple forms of concurrency control with minimal overhead. This system decomposes the workload into partitions and selects a concurrency control protocol for each partition of workload that the protocol is optimized for, and during workload changes adaptively reconfigures the protocols online. The experiments show that this approach has higher throughput than the single best protocol under workload changes.

## Earlier Projects

- **Structured Data Shuffling for Big Data Analytical Stacks** Nov. 2013-Jan. 2015
  I 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.
- **A Fast and Space-efficient Join Method for Log Processing in MapReduce** Sep. 2012-Nov. 2013
  I 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.

## 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 Assistant

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

## Referees

**Name** Aaron Elmore
**Affiliate** University of Chicago
**Position** Assistant Professor
**Contact** aelmore@cs.uchicago.edu

**Name** Michael Franklin
**Affiliate** University of Chicago
**Position** Liew Family Chairman of Computer Science
**Contact** mjfranklin@uchicago.edu

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

**Name** Umar Farooq Minhas
**Affiliate** Microsoft Research
**Position** Principle Researcher
**Contact** ufminhas@microsoft.com

**Name** Wei Li
**Affiliate** Institute of Computing Technology
**Position** Associate Professor
**Contact** liwei@ict.ac.cn