Bo Zhao | Postdoctoral Researcher

Huxley Building, Department of Computing, **Imperial College London**
180 Queen's Gate, London SW7 2AZ – United Kingdom
www.imperial.ac.uk/people/bo.zhao • Email: bo.zhao@imperial.ac.uk

Research Interest

My research interest is on **data-intensive computing systems** including **distributed machine learning systems**, **data stream management systems**, and **code optimization**.

Work Experience

07/2021–present: Postdoctoral Researcher, **Imperial College London, London, UK**

06/2019–09/2019: Software Development Engineer Intern at Amazon, **AWS Redshift, Berlin, Germany**

02/2016–06/2021: Research Assistant in **Humboldt-Universität zu Berlin, Berlin, Germany**

09/2015–12/2015: Research Assistant in **Technische Universität Darmstadt, Darmstadt, Germany**

10/2013–02/2015: Student Research Assistant in **RWTH-AACHEN University, Aachen, Germany**

06/2009–09/2009: Software Development Intern at **China United Network Communications Group Co.Ltd**

Education

Humboldt-Universität zu Berlin — **Berlin, Germany**
PhD in Computer Science, magna cum laude — *02/2016–12/2021*
Topic: State Management for Efficient Event Pattern Detection

Xi'an Jiaotong University — **Xi'an, China**
M.S. in Computer Science, GPA: 3.78/4 — *09/2012–07/2015*
Ranking: Top 1%, 2nd among 150 students

Wuhan Institute of Technology — **Wuhan, China**
B.S. in Network Engineering, Major GPA: 91.06/100 — *09/2008–07/2012*
Ranking: Top 1%, 2nd among 500 students

Visiting Research

University of Queensland — **Brisbane, Australia**
Visiting Researcher in Computer Science, hosted by Prof. Xiaofang Zhou — *05/2017–06/2017*
Data Stream Processing

RWTH-AACHEN University — **Aachen, Germany**
Visiting M.S. Student in Computer Science, hosted by Prof. Felix Wolf — *09/2013–02/2015*
High Performance Computing

Honors & Awards

2019-2020: Travel Grant of the Silk Road International Symposium for Distinguished Young Scholars

2017–2018: IEEE ICDE Student Travel Grant

2015–2016: EDBT Summer School Travel Grant

2014–2015: Outstanding Graduate, ACM SIGPLAN Travel Grant, ACM SIGMICRO Travel Grant

2012–2013: China National Scholarship(top 0.2%), Creative-Master Scholarship, Excellent Master Student

2011–2012: Excellent Graduation Thesis, Outstanding Graduate

2010–2011: China National Scholarship(top 0.2%), Top Grade Scholarship, Pacemaker to Merit Student, Advanced Individual in Social Practice

2009–2010: China National Scholarship(top 0.2%), Top Grade Scholarship, Pacemaker to Merit Student

2008–2009: Top Grade Scholarship, Pacemaker to Merit Student, Outstanding League Member

Publications

- Bo Zhao, Han van der Aa, Nguyen Thanh Tam, Nguyen Quoc Viet Hung, Matthias Weidlich: **EIRES: Efficient Integration of Remote Data in Event Stream Processing**, *In Proc. of the 47th ACM SIGMOD International Conference on Management of Data (**SIGMOD'21**), Xi'an, China, ACM, June 2021*

- Bo Zhao, Nguyen Quoc Viet Hung, Matthias Weidlich: **Load Shedding for Complex Event Processing: Input-based and State-based Techniques**, *In Proc. of the 36th IEEE International Conference on Data Engineering (**ICDE'20**), Dallas, TX, USA, IEEE, April 2020*

- Gururaghav Raman, Jimmy Chih-Hsien Peng, Bo Zhao, Matthias Weidlich: **Dynamic Decision Making for Demand Response through Adaptive Event Stream Monitoring**, *In Proc. of the IEEE Power & Energy Society General Meeting (**PESGM'19**), Atlanta, GA, USA. IEEE, August 2019.*

- Bo Zhao: **Complex Event Processing under Constrained Resources by State-based Load Shedding**, *In Proc. of the 34th IEEE International Conference on Data Engineering (**ICDE'18**), Paris, France, IEEE, April 2018*

- Bo Zhao, Zhen Li, Ali Jannesari, Felix Wolf, Weiguo Wu: **Dependence-Based Code Transformation for Coarse-Grained Parallelism**, *In Proc. of the International Workshop on Code Optimisation for Multi and Many Cores (**COSMIC'15**) held in conjunction with **CGO'15**, San Francisco Bay Area, CA, USA, ACM, February 2015*

- Bo Zhao, Ali Jannesari: **Dependence-Based Parallel Code Generation Using Intel CnC**, *In Proc. of the 24th International Conference on Parallel Architectures and Compilation Techniques (**PACT'15**),San Francisco Bay Area, CA, USA, October 2015* (ACM SRC poster)

- Zhen Li, Bo Zhao, Ali Jannesari, Felix Wolf: **Beyond Data Parallelism: Identifying Parallel Tasks in Sequential Programs**, *In Proc. of the 15th International Conference on Algorithms and Architectures for Parallel Processing (**ICA3PP'15**), Springer, November 2015*

- Song Liu, Bo Zhao, Qing Jiang, Weiguo Wu: **A Semi-Automatic Coarse-Grained Parallelization Approach for Loop Optimization And Irregular Code Sections** (in Chinese). *Chinese Journal of Computers, 2017*

- Song Liu, Weiguo Wu, Bo Zhao, Qing Jiang: **Loop Tiling for Optimization of Locality and Parallelism** (in Chinese). *Journal of Computer Research and Development, 2015*

Teaching Experience

Summer semester 2020: Oral exam examiner on *process mining*

Winter semester 2019: Oral exam examiner on *event process*

Summer semester 2018: Oral exam examiner on *process mining*

Summer semester 2018: Seminar on *event stream processing*

Talks

December 2021: Invited talk at Xi'an Jiaotong University, Virtual Event, China;

November 2021: Invited talk at Nanjing University, Virtual Event, China;

June 2021: The 47th ACM International Conference on Management of Data (*SIGMOD'21*), Virtual Event, China;

March 2021: Invited talk at EPFL, Lausanne, Switzerland;

December 2020: Invited talk at Hasso Plattner Institute, Potsdam, Germany;

November 2020: Invited talk at ETH Zürich, Zürich, Switzerland;

November 2020: Invited talk at Imperial College London, London, UK;

November 2020: Invited talk at Technical University of Berlin, Berlin Germany;

April 2020: The 36th IEEE International Conference on Data Engineering (*ICDE'20*), Dallas, TX, USA;

April 2019: Invited talk at Xi'an Jiaotong University, Xi'an, China;

April 2018: The 34th IEEE International Conference on Data Engineering (*ICDE'18*), Paris, France;

September 2015: The 7th Annual Concurrent Collections Workshop (*with LCPC'15*), Raleigh, NC, USA;

September 2015: The 44th International Conference on Parallel Processing (*ICPP'15*), Beijing, China;

February 2015: The 2nd International Workshop on Code Optimisation for Multi and Many Cores (*COSMIC'15*), San Francisco Bay Area, CA, USA;

September 2014: The Sixth Annual Concurrent Collections Workshop, Intel Corp in Hillsboro, OR, USA;

Academic Services

PC Member: CIKM'21, CIKM'22

Availability Committee: SIGMOD'22

Projects

Distributed Reinforcement Learning Systems — **Imperial College London**
07/2021–present

I am working on a reinforcement learning (RL) framework that supports the distributed training of agents using various RL algorithms. The framework offers a clean API abstraction for writing RL algorithms, which decouples the algorithm from deployment and execution considerations, including the use of accelerators, the level of parallelism and the distribution of computation across a cluster of workers. It translates the RL algorithm into a series of compiled computational graphs, which run efficiently on CPUs, GPUs, and other AI processors. We released an initial open source version (https://www.mindspore.cn/reinforcement/en). The detailed publications will come soon. Stay tuned.😎

State Management for Efficient Event Stream Processing — **Humboldt-Universität zu Berlin**
02/2016–05/2021

This is my PhD project. I focus on optimized state management for complex event processing (CEP). CEP evaluates queries over streams of events for low-latency detection of user-specified patterns which correlate event data within certain time windows. Some complex queries detect sequence patterns or perform sophisticated mathematical algorithms (*eg*, machine learning) on top of value predicates. Such queries are stateful and therefore, the CEP engine needs to maintain a set of partial results. When combined with kleene closure operators, the size of partial results grows exponentially in the number of processed events. High input rates of streams amplify such a problem. This makes low-latency data analysis challenging. What's worse, when integrating with remote data sources, CEP's performance deteriorates even further by data transmission latency. Massive parallelization is not trivial because of irregular data dependences among partial results and events. To tackle this problem, I propose optimizations of load shedding and efficient remote data fetching and caching techniques. To this end, I built a prototype CEP engine in C++ around 7k lines of code and implemented the load shedding optimization and remote data fetching and caching techniques. For development efficiency, I use Python library for offline machine learning such as training classifiers for load shedding decisions. Whereas, online classifiers are implemented in C++ for low latency analysis. Initial results have been published in ICDE'18 PhD Symposium. More detailed evaluations have been published in ICDE'20 (source code https://github.com/zbjob/AthenaCEP). The other work of remote data fetching and caching optimizations has been published in SIGMOD'21 (source code https://github.com/zbjob/EIRES). I also cooperated with the Department of Electronic Engineering, National University of Singapore, to apply our complex event processing optimizations to smart grid management using Esper stream engine (source code https://github.com/zbjob/SmartGrid). The result is published in the IEEE PES-GM'19 conference paper. More comprehensive evaluations are presented in a journal article under review for IEEE Transactions on Power Systems.

Lightweight Profiling for AWS Redsfhit Queries — **Amazon Web Services**
06/2019–09/2019

I was a software development intern at AWS Redshift team. During this internship, I developed a light-weight performance analysis tool to trace and analyse detailed runtime query execution behaviours of the cloud-based distributed database, Redshift, deployed on AWS. The tool is able to target the performance bottleneck and pave the way for further improvements. However, profiling at the extremely fine granular incurs unacceptably high computational overhead. My goal was to close this gap. The performance analysis tool is based on eBPF (extended Berkeley Packet Filter) and its front end BCC (BPF Compiler Collection). In order to reduce the computational overhead, the eBPF code monitors statistics in kernel space and store them in BPF maps. In user space, I fetch the BPF maps via BCC APIs (through Python scripts) and perform sophisticated analysis asynchronously. To further reduce the computational overhead, I employ the approximation technique (e.g. sketching), sampling, to monitor execution time for different code sections. I integrate this profiler into the code generator of the query plan. Therefore, the probes have been automatically inserted to the generated C++ code that is compiled from PostgreSQL. When the queries are executed, the profiling is automatically done. I evaluated the performance analysis tool against the TPC-DS benchmark at 3TB and 10TB scale on clusters of AWS 5-node DC2.8xlarge instances. The tool is able to reduce the computational overhead to 1.0% and obtain accurate profiling information. I also found some insights for performance improvements on real-world business workloads in AWS. For instance, one query of a Redshift customer has been improved by 1.75 × faster.

Auto-Parallelization on Multicores — **RWTH-Aachen University**
10/2013–02/2015

I was a member of the Multicore Programming Research group in German Research School for Simulation Sciences, RWTH-AACHEN University, where I did my master thesis. I proposed and developed a novel auto-parallelization framework, which integrates data-dependence profiling, coarse-grained task parallelism extraction and source-to-source code transformation. Our group identified sections of code called computational units (CUs), which follow a read-compute-write pattern and can be used as building blocks to construct parallel tasks. First, I used our own profiling tool to analyse the data and control dependence among the source code and therefore, generated the dependence graph of CUs. Then I merged CUs in the CU graph and generated a more coarse-grained task graph. Second, I used LLVM front-end Clang to parse the source code, extract the AST, traverse the AST to locate DOALL loops and the code sections targeted by the task graph. Finally, I transformed the sequential source code to parallel one using Intel Threading Building Blocks (TBB) and Intel Concurrent Collections (CnC). My work has been published in two conference papers. Project link: https://www.discopop.tu-darmstadt.de

Loop Optimization on Multicore Systems — **Xi'an Jiaotong University**
09/2012–06/2015

This project was supported by the National Natural Science Foundation of China. A PhD student and I mainly focused on data-intensive parallel loop optimizations on shared memory architectures, especially NUMA machines. We proposed a loop tiling strategy for imperfectly-nested loop nests and leveraged machine learning technologies to select the appropriate tile size to keep the cache hot. We successfully exploited fine-grained parallelism and data locality. We evaluated our approach on several benchmarks, achieving speedups up to 200 ×. We published our results in two journal articles.

References

Name	Affiliation	Email address
Prof. Peter Pietzuch	Imperial College London	prp@imperial.ac.uk
Prof. Matthias Weidlich	Humboldt-Universität zu Berlin	matthias.weidlich@hu-berlin.de
Dr. Nguyen Quoc Viet Hung	Griffith University	quocviethung.nguyen@griffith.edu.au
Prof. Han van der Aa	Universität Mannheim	han@informatik.uni-mannheim.de