Yuxin Chen

CONTACT INFORMATION

Email: yxxchen@ucdavis.edu — *Website:* YuxinxinChen.github.io
GitHub: https://github.com/YuxinxinChen — *Phone:* +1-484-790-0788

EDUCATION

UNIVERSITY OF CALIFORNIA, DAVIS, CA, USA
Ph.D., Engineering and Computer Science, October 2016–2023 (Expected)

- Thesis Topic:
  Asynchronous Partitioned Global Address Space (PGAS) Style Programming with GPUs
- Adviser: Professor John D. Owens

KING ABDULLAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, SAUDI ARABIA
M.S., Computer Science, September 2013–2015

- Thesis Topic: *GPU-Accelerated dimension-independent adaptive Metropolis*
- Adviser: Professor David E. Keyes

Xiamen University, Fujian, China
B.S., Management School, September 2009–2013
B.S., The Wang Yanan Institute for Studies in Economics, September 2009–2013

PROFESSIONAL EXPERIENCE

NVIDIA:

Extend Triton to enable device function code generation for Grafia June–September 2023

- Extended Triton to enable device function code generation for NVIDIA Grafia, which imposes several constraints on those generated functions. These constraints include:
  - Only using Grafia register pool for device functions.
  - Only using pre-allocated Grafia shared memory.
  - Generating Grafia input context and return context with the right set of registers on the fly without exposing those contexts to the Triton programming and user level.

  This effort also involved surpassing the limitation, prevailing at the time, where Triton functions accepted only global memory pointers as arguments
- At the Triton Python programming level, users have the freedom to program without constraints. After parsing the Triton Python code into Triton Multi-Level Intermediate Representation (MLIR), I decomposed and segregated the MLIR into distinct functions based on functionality (DMA, Compute, Sync, etc.) during the pipeline process to lower Triton to the LLVM level.
- In addition to my efforts to enable user-programmed Triton Python code to generate device functions that could plug into NVIDIA Grafia runtime, multiple optimizations were incorporated during the pipeline for lowering the Triton MLIR to LLVM. These optimizations include:
  - Leveraging Hopper warp specialization.
  - Extracting the producer and consumer relationship and executing them asynchronously.

UNIVERSITY OF CALIFORNIA, DAVIS:

PGAS (Partition Global Addressing Space) with Distributed GPUs June–Septem 2020

- Led the research and open-source development of novel asynchronous distributed GPU data structures and algorithms that support irregular applications in areas such as biology, graph analytics, and sparse linear algebra. These solutions leverage Partitioned Global Address Space (PGAS)-style communication between distributed GPUs, allowing for:

  - More efficient communication latency hiding
  - Higher bandwidth utilization
  - Reduced pipeline stalls through increased pipeline scheduling flexibility.
- Resulted in graph algorithms that achieved significant speedups, ranging from 1–100×, on both InfiniBand-connected and NVLink-connected distributed GPU systems, surpassing the performance of current state-of-the-art graph frameworks.
- Collaborated with researchers and software engineers to understand their needs and implement efficient solutions for GPU-based distributed systems.
- Created a highly effective programming environment that enabled end-users to efficiently leverage GPUs for irregular communication patterns, contributing to the HPC community's understanding of how to design primitives for accelerator-equipped distributed-memory machines.
- Results published at ACM Supercomputing [2] and ACM Int'l Conference on Parallel Processing [1].

GPU utilized Apache Spark 2017–2018

- Developed a GPU version of Spark's fundamental data structure, Resilient Distributed Datasets (RDDs), while adhering to the original design philosophy of Spark. The GPU component was designed to be compatible, efficient, flexible, and principled, enabling seamless integration with existing Spark workflows.
- Developed a highly efficient GPU-accelerated logistic regression, map and reduce operations, etc. Our GPU-accelerated map and reduce operations are twice as fast as its CPU-only Spark version, and the GPU-accelerated logistic regression algorithm outperforms its CPU-only Spark version by a factor of five, improving the overall performance of Spark workflows.
- Collaborated with other members of the team to integrate these enhancements into the Apache Spark codebase, enabling end-users to leverage GPU acceleration to gain deeper insights from their data, improve decision-making, and accelerate time-to-insight.

LAWRENCE LIVERMORE NATIONAL LABORATORY (LLNL)

Fast membership query hash tables April–June 2020

- I enhanced the effectiveness of hash table member queries by utilizing fingerprint tables for hash tables, where each fingerprint occupied only 8-16 bits of memory. This approach significantly reduced memory traffic and alleviated the Translation Lookaside Buffer (TLB) problem, while ensuring high query accuracy. On RMAT datasets, the solution effectively filtered out false queries with less than 5% false positive rate.
- Rather than recreating a larger hash table when an insertion failed, I devised an overflow buffer that could store failed insertions. This approach enabled GPU multi-threading searches of keys, leading to quicker query and insertion times. During testing on RMAT datasets, this approach yielded an overflow ratio of less than 0.1% for varying load factors.
- This project required advanced knowledge of data structures, GPU programming, and algorithm optimization. My contributions to this project resulted in significant performance improvements, demonstrating my ability to develop innovative and efficient solutions to complex problems.

NVIDIA

Communication Aggregation Runtime for muti-GPU Systems June–September 2018

- Developed a communication aggregator that facilitates dynamic communication aggregation for gathering, scattering, and randomly accessing data between GPUs, resulting in significant improvements in bandwidth utilization.

- Decoupling communication granularity from computation granularity, the aggregator allows users to select the optimal communication granularity. Additionally, by decoupling the time that generated the message from the time it actually sent the data, the aggregator enables users to code in an algorithm's natural logic flow to send messages of their native size.
- The aggregator runs transparently along the application code, minimizing the burden on the user, and allows for sending data both in an eager (immediate) and in a more bandwidth-efficient manner via aggregation.

ADOBE

Auto-tuning of system configurations for deep learning training June–September 2017

- Implemented a solution for automating the tuning of hyperparameters and system configurations for training deep neural networks. This was achieved by using Bayesian Optimization, which allowed for a more efficient and effective search of the parameter space. By leveraging the auto-tuner for the Inception model on Imagenet, the search space was reduced by about half, leading to faster training of the network. This solution provided a significant improvement over traditional manual hyperparameter tuning methods, which can be time-consuming and less effective in finding optimal settings.

PUBLICATIONS

[1] Xiang Cheng, **Yuxin Chen**, Suvrit Sra. Transformers Implement Functional Gradient Descent to Learn Non-Linear Functions In Context. arXiv preprint arXiv:2312.06528

[2] **Yuxin Chen**, Benjamin Brock, Serban Porumbescu, Aydın Buluç, Katherine Yelick, and John D. Owens. Scalable irregular parallelism with GPUs: Getting CPUs out of the way. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (*SC '22*) IEEE Press, Article 50, 1–16.

[3] **Yuxin Chen**, Benjamin Brock, Serban Porumbescu, Aydın Buluç, Katherine Yelick, and John D. Owens. Atos: A Task-Parallel GPU Scheduler for Graph Analytics. In Proceedings of the International Conference on Parallel Processing (*ICPP 2022*). Association for Computing Machinery, New York, NY, USA, Article 50, 1–11. doi:10.1145/3545008.3545056.

[4] Taylor Groves, Ben Brock, **Yuxin Chen**, Khaled Z. Ibrahim, Lenny Oliker, Nicholas J. Wright, Samuel Williams, and Katherine Yelick. 2020. Performance Trade-offs in GPU Communication: A Study of Host and Device-initiated Approaches. In 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (*PMBS*). 126–137.

[5] Benjamin Brock, **Yuxin Chen**, Jiakun Yan, John D. Owens, Aydın Buluç, and Katherine Yelick. RDMA vs. RPC for Implementing Distributed Data Structures. In Proceedings of the IEEE/ACM 9th Workshop on Irregular Applications: Architectures and Algorithms, *IA3 2019*, pages 17–22, November 2019.

[6] **Yuxin Chen**, David Keyes, Kody JH Law, and Hatem Ltaief. "Accelerated dimension-independent adaptive Metropolis." *SIAM* Journal on Scientific Computing 38, no. 5 (2016): S539-S565. doi:10.1137/15M1026432.

TALKS

LET THE CHICKENS RULE THEMSELVES.

- The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC22), November 13-18, 2022, Dallas, Texas, USA.

ATOS: A TASK-PARALLEL GPU SCHEDULER FOR GRAPH ANALYTICS

- 51st International Conference on Parallel Processing (ICPP22), August 29th to Sept 1st, 2022, Online.

Scalable irregular parallelism with GPUs: Getting CPUs out of the way
- Intel, Inc, Aug 25, 2022, San Jose, CA, USA.

A Task-Parallel GPU Dynamic Scheduling Framework for Dynamic Irregular Computations
- Facebook, Inc, Dec 8, 2021, San Jose, CA, USA.

Task-Parallel GPU Dynamic Scheduling Framework for Dynamic Irregular Computations
- Women in High Performance Computing (WHPC), Nov 14, 2021, St. Louis, MO, USA.

Gunrock: GPU Graph Analytics
- Annual Industrial Affiliates Conference, May, 2017, San Jose, CA, USA.

Technical Skills

Programming: C, C++, CUDA, Java, Python, Pytorch, MATLAB
Operating Systems: Apple OS X, Linux
Other Tools: TEX, UNIX shell scripting, GNU make, CMake, Git

References

References are available upon request.