Jiakun Yan

jiakuny3@illinois.edu • jiakunyan.github.io

Research Interests

My research interest lies in **parallel computing**. Currently, I am interested in co-designing high-level task-based programming models and low-level communication systems to better utilize modern parallel architectures and improve the performance, scalability, and programmability of modern parallel applications.

Education

University of Illinois at Urbana-Champaign **Illinois, USA**
Aug. 2020 -

- Computer Science PhD student, advised by Marc Snir .
- Pursue research in Parallel Computing.
- GPA: 4.0/4.0 | Relevant Courses: CS555 Numerical Methods for PDEs (ongoing), CS425 Distributed Systems, CS523 Advanced Operating System, CS533 Parallel Computer Architecture, CS526 Advance Compiler Construction, CS483 Applied Parallel Programming.

Shanghai Jiao Tong University **Shanghai, China**
Sep. 2016 - Jun. 2020

- Bachelor's Degree of Engineering, Dept. of Computer Science.
- Zhiyuan Honors Program of Engineering (an elite program for top 5% talented students)
- GPA: 91.88/100 | Ranking: 4^{th}/151.

University of California, Berkeley **California, USA**
Jan. 2019 - May 2019

- Exchange student, Berkeley Global Access Discover Program, GPA: 4.0/4.0.

Experience

Programming Systems and Applications Research Group **NVIDIA Research**
Research Intern, advised by Michael Bauer and Michael Garland *May. 2022 - Aug. 2022*

- Realm Collective: design and implement collective communication operations in Realm.

PASSION Lab **Lawrence Berkeley Laboratory**
Research Assistant, advised by Aydın Buluç and Katherine Yelick *Aug. 2019 - Jan. 2020*

- Asynchronous RPC Library (ARL): a high-throughput RPC system with node-level aggregation and single-node work-stealing.
- RDMA vs. RPC for Implementing Distributed Data Structures

Publication

- Benjamin Brock, Yuxin Chen, **Jiakun Yan**, John Owens, Aydın Buluç, and Katherine Yelick. "RDMA vs. RPC for Implementing Distributed Data Structures", Workshop on Irregular Applications: Architectures and Algorithms (IA3), 2019.

Project

HPX over LCI **UIUC**
Advised by Marc Snir and Hartmut Kaiser , WAMTA23 Poster *Aug. 2021 - Present*

- HPX is a runtime system known for its support for the asynchronous task programming model. Previously, HPX uses MPI as its major communication backend. In this project, we added an LCI parcelport for HPX, using LCI features including (a) dedicated progress threads to improve cache locality (b) one-sided "put iovec" primitive to minimize message number and memory copies (c) completion queues to reduce probing and testing.
- The first version of LCI parcelport has been completed and shipped with HPX releases (1.8.0 and later). We evaluated the performance using a real-world application, Octo-Tiger : a star system simulator based on the fast multipole method on adaptive Octrees. We achieved more than 40% performance improvement compared to the MPI parcelport on 32 nodes/4096 cores.

Lightweight Communication Interface **UIUC**

Advised by Marc Snir *Aug. 2020 - Present*

- The Lightweight Communication Interface (LCI) is designed to be a low-level communication library used by high-level libraries and frameworks. It aims to support irregular and asynchronous applications such as graph analysis, sparse linear algebra, and task-based runtime on modern parallel architectures. Major features include (a) support for more communication primitives such as two-sided send/recv and one-sided remote put (b) better multi-threaded performance (c) explicit user control of communication resource (d) flexible signaling mechanisms such as synchronizer, completion queue, and active message handler.
- I am one of the major developers of LCI. Main contributions include developing the Libfabric backend of LCI and designing/implementing LCI v1.7 along with a parameterized testing framework and performance counters. I am working on evaluating the multi-threaded performance of LCI and exploring ways, such as utilizing multiple hardware contexts, to improve its multi-threaded performance.

Collective Communication Operations in Realm **NVIDIA Research**

Advised by Michael Bauer and Michael Garland *May 2021 - Aug. 2021*

- Realm is an event-based low-level runtime system providing a high-performance asynchronous task execution model for the higher-level data-centric parallel programming system Legion . It offers the ability to perform memory copies across different data buffers, regardless of their physical location. Originally, Realm only supports point-to-point data copy operations. In this project, we extended the copy operation to handle collective broadcast communication.
- We designed and implemented a hierarchical path planning algorithm that includes inter-node radix tree broadcast and intra-node path aggregation. We used a set of synthetic benchmarks to evaluate the broadcast operations and found it achieved significant improvement compared to the original point-to-point copies. (The actual speedup number depends on the benchmark setup.)

TaskFlow: Task-based Runtime on Distributed-memory System **UIUC**

Advised by Josep Torrellas and Marc Snir , CS533 course project *Jan. 2021 - May 2021*

- TaskFlow is a simple but efficient task-based runtime for distributed-memory systems. It adopts the PTG-based task programming model that enables reduced time/memory overhead and fine-grained synchronization. It executes tasks according to an explicit task dependency graph and uses active messages to proactively signal remote tasks.
- We implemented TaskFlow based on Argobots and MPI. We performed a collection of micro-benchmarks and mini-applications to evaluate the performance of its various configurations and compare it with two established PTG-based task systems, TaskTorrent and PaRSEC. The benchmark results showed that TaskFlow generally achieved the best performance under various circumstances.

Asynchronous RPC Library (ARL) **LBNL**

Advised by Aydın Buluç and Katherine Yelick *Aug. 2019 - Jul. 2020*

- Data-driven HPC applications suffer significant overheads for their fine-grained communication patterns. ARL is a thread-based RPC system that targets data-driven applications. It uses Remote Procedure Call (RPC) to provide powerful expressive ability. It achieves high performance through node-level aggregation, single-node work-stealing, and innovative concurrent data structures. It also provides a flexible programming interface for users.
- Node-level aggregation is the primary idea underlying the ARL, which aggregates RPC requests sharing the same source and target node and sends them together as one large message. Using this methodology, ARL is able to utilize high bandwidth across cores on the same node to achieve low overhead and high throughput.
- I am the main developer of the ARL. ARL is developed as a C++ header-only library based on the GASNet_EX communication library.

Skills

- **Programming Language:** C (proficient), C++, Python, Java, Rust, Go
- **Library & Framework:** libibverbs, libfabric, MPI, Argobots, PAPI, CUDA, GASNet-EX, UPC++, OpenSHMEM, Pytorch, Android, Qt