Research Statement

Yihan Sun

Introduction

Modern data-driven applications aim to process, analyze, and discover valuable information from data with increasing volumes. These applications seek high-performance in updates and queries, as well as a hybrid of both at the same time. Such requirements necessitate considering parallelism in algorithm design and engineering. Furthermore, the development of modern hardware technologies has brought parallel algorithms into practice. In addition to solutions using supercomputers or large distributed systems, over the last decade, multicore processors with terabytes of memory have become a powerful tool to address large-scale problems.

Parallelism is about improving performance, which is favored in almost all problem domains. Unfortunately, there is a gap between theory and practice in parallel computing. For the theory, many existing theoretically-efficient algorithms are much more complicated than their sequential counterparts. The delicate-designed algorithmic details make many of these algorithms hard to implement efficiently, especially considering that parallel programming is already hard. In practice, many existing parallel solutions and optimizations are tailor-made for particular platforms, input instances, or applications without high-level theoretical abstraction or provable guarantees in general cases. This issue makes those otherwise nice solutions less extendable to other settings and other platforms.

My research aims to **bridge this gap between theory and practice in parallel computing** by focusing on **solutions to large-scale problems using multicore parallelism.** My research shows that *theoretically-efficient parallel algorithms can also be simple and fast in practice*, and *highly-optimized practical parallel solutions can also be nicely modeled in theory, and be generic and extendable to different settings and applications*. My results have led to effective parallel solutions to both fundamental problems and specific applications. Many of my approaches have been (or will be) adopted by several *libraries*, integrated into existing *courses* at CMU and introduced in *keynote talks* and *tutorials* at top conferences.

On the theory side, I have designed many **simple and efficient parallel algorithms and data structures with provable guarantees, many of which have improved previous best-known asymptotical bounds**. Some examples of my work include an algorithmic framework for tree structures [2, 11, 12, 14] that unifies multiple balancing schemes and enables a series of clean and theoretically efficient parallel algorithms, the first incremental Delaunay triangulation algorithm [3, 4] with optimal work and polylogarithmic depth, and a simple parallel single-source shortest-path algorithm [6] with one of the best-known work-depth tradeoff bounds.

On the system side, I have **designed and implemented parallel programming frameworks and a parallel library, as well as applied them to various applications**, such as computational geometry, inverted indexes searching, and database systems. My approaches show efficiency on large-scale data. My library [10, 14] for parallel tree structures has been tested on hundreds of gigabytes of data, achieving almost-perfect self-speedup (60-100$\times$ on 72 cores/144 hyperthreads). I have also applied my library to database management systems (DBMS) [12] systems and tested the library on database benchmarks. My implementation outperforms state-of-the-art DBMSs on mixed workloads of OLAP (TPC-H) and OLTP (TPC-C) queries, being 4$\times$ faster for queries and 5% faster for updates.

My work involves various areas including *parallel computing, theory (algorithms/data structures), computational geometry, software engineering, databases, data mining (graph mining), and computational biology*. I will elaborate on some of my research topics in three categories: parallel data structures (Sec. 1), parallel algorithms (Sec. 2), and topics related to graph processing and graph mining (Sec. 3). I will conclude with my future work and research vision (Sec. 4).

1 Parallel Data Structures

On the data structure side, my work mainly focuses on the parallel balanced binary tree structure. As one of the most important data structures used in algorithm design and programming, balanced trees are widely used in real-world applications for organizing data. Answering the challenges thrown up by modern large-volume and ever-changing data, I proposed a class of tree structures and their implementation in a corresponding library called PAM. The tree structure demonstrates good performance both sequentially and in parallel, and is also theoretically efficient, full-featured, safe for concurrency, lock-free, multi-versioned, and support efficient garbage collection. Given these useful properties, I further applied the tree structure to a many real-world applications.

The fundament of our tree structure is an algorithmic framework based on a single primitive JOIN. A key benefit of our tree structure is that it embeds *generic methodology* for different usage scenarios. Firstly, all algorithms except JOIN are identical across four balancing schemes, which are AVL trees, red-black trees, weight-balance trees and treaps. Secondly, our methodology is easily extendable to a variety of applications in different domains. This versatility reduces users' incremental effort of adapting our tree structure simply as a black box to their own applications. Furthermore, this minimizes the required coding effort to re-create the data structure with special requirements when necessary (in another programming language, for example).

The simplicity is not at the expense of efficiency. When applied to applications, PAM helps to achieve an efficient solution to each specific application. Although designed as a general-purpose library, PAM is up to an order of magnitude faster compared to existing implementations specific to each application.

I present below a list of my work related to the parallel tree structure.

Design and Analysis for JOIN-based algorithms [2]. I propose the JOIN-based algorithmic framework for parallel tree algorithms, covering most commonly-used functions and unifying multiple balancing schemes. Despite generic and simple, the efficiency of these algorithms is non-trivial, especially for the set-set functions (e.g., UNION). An initial (sequential) version of JOIN-based set-set algorithms was proposed in 1992 on weight-balanced trees, but the work bound was open until 2016, when I presented a generic proof for multiple balancing schemes. My results prove the optimality of the join-based set-set algorithms under the comparison model, and show that these algorithms are also highly-parallelized. The join-based algorithms have been widely applied in several libraries and frameworks, including Hackage (the Haskell standard library), Aspen (a graph processing framework extending Ligra) and my own C++ library PAM [10, 14]. My work has helped in developing these libraries by both improving the algorithms and providing theoretical support.

The formalization of *augmented map* and its applications [11, 14]. I propose an abstract data type (ADT) called *augmented map* based on ordered maps (key-value stores), which serves as a high-level abstraction that elegantly models many applications. I propose two underlying representations: JOIN-based augmented trees, and prefix structures [11]. For both I develop parallel algorithms to implement them. Using PAM, many applications can be implemented easily with good performance. I will elaborate on some examples in the following sections.

A parallel library PAM [14]. I implement the tree structure in a parallel library called PAM. The interface covers a wide range of functions.It supports fully-persistent (and functional) tree structures using path-copying. The library is also safe for concurrency, and supports efficient garbage collection.

With the theories and the library, I applied the tree structure to many areas. As a fundamental data structure, our tree structure is beneficial in improving the performance of a variety of applications. I list a few below.

A class of geometric data structures [11]. I applied the augmented map to solve a list of 2D range-related geometric problems, including *2D range query, segment query and rectangle query*. Such problems are fundamental in computational geometry. However, in terms of parallel algorithms, most theoretically efficient solutions have not been evaluated in practice, and most data structures in use do not provide a worst-case theoretical guarantee. Using augmented maps, I show theoretically-efficient parallel solutions to these problems, and provide an in-depth experimental study of these data structures. My solution allows simple implementation and efficient performance. Even sequentially, my implementation is up to $2\times$ faster than existing sequential libraries (CGAL and Boost) in construction and up to $1000\times$ faster in queries. In parallel, my implementation achieves 30-$60\times$ self-speedup in construction and up to $100\times$ self-speedup in queries on 72 cores with hyperthreading (144 threads).

Multi-version concurrency control (MVCC) with strong properties [1]. With my co-authors, I have developed algorithms for MVCC with provable strong properties [1], which are achieved by using a functional tree structure and a global single writer.Such a setting is motivated by scenarios such as online data services, where the heavy workload lies in concurrent analytic queries on data requiring fast respond, while light streaming updates happen continuously. Our system guarantees no-aborts, serializability, wait-free, (provable) low delay, (provable amortized) low contention, and safe and precise garbage collection (GC), where safe GC means that any freed tuples will not be used any more, and precise GC means that any tuples that will not be used any more are freed immediately. Such strong properties were never achieved simultaneously in previous work even under the single-writer assumption. We also extend our approach to multi-writers using combining (batching) methods. In addition to the theoretical results, our solution also shows efficiency in experiments. Experiments verify that our system conducts GC fairly timely and hardly suffers from contention among threads. Also, even using a single writer for batched updates, PAM outperforms state-of-the-art concurrent data structures by 1.2-1.8$\times$ for concurrent reads and updates.

An HTAP database system with snapshot isolation [12]. Using PAM, I developed a parallel HTAP (Hybrid Transactional and Analytical Processing) database management system (DBMS) supporting snapshot isolation (SI). This work is to answer the requirements of modern DBMSs, where ever-changing data sets and the necessity of analyzing the latest data exist simultaneously. Our system focuses on fast responses to queries, while allowing updates to be visible in time. The fast queries are made possible by two key features. First, PAM uses path-copying to support SI, which only requires constant time for acquiring a snapshot. Secondly, our tree structure enables denormalization and materialization view on logically hierarchical data by table partitioning with minimal extra cost. As queries being processed, concurrent updates are batched and committed by a global write transaction. This not only yields serializability, but also avoids contention caused by concurrent writes. Combining these techniques, our system outperforms existing systems in different workloads: a pure OLAP (online analytical processing) workload, a pure OLTP workload (online transaction processing), and a mixed workload of both (an HTAP workload). Especially, on the HTAP workload, our system is compared with two existing real-world DBMS HyPer and MemSQL, and is more than 4$\times$ faster than the compared systems in queries, and is at least 5% faster in updates.

2 Parallel Algorithms

2.1 Write-efficient algorithms

Emerging non-volatile memory (NVM) is projected to become the dominant type of main memory in the near future. Despite its advantages of significantly lower energy and higher density (bits per area) than DRAM, there is one important distinction: *writes are significantly more expensive than reads*, suffering from higher latency, lower per-chip bandwidth, higher energy costs, and so on. This challenges classic algorithm design where, over fifty years, read and write were considered to have similar cost. As a result, the study of *write-efficient* algorithms, which aim at reducing the number of writes relative to existing algorithms, is of significant and long-lasting importance. My research particularly considers the read-write asymmetry in tree algorithms [9] and geometric algorithms [4].

Write-efficient Balanced Binary Trees. My research has conducted empirical study of write-efficient tree algorithms [9]. In this work, I study factors that affect the overall I/O cost on tree algorithms, including the different properties that different balancing schemes exhibit under the read-write asymmetry, the batch size relative to the cache size, and different read-write asymmetric ratios. These evaluations lead to many interesting findings that are beneficial to implementing write-efficient algorithms.

Write-efficient Geometric Algorithms. My research has studied parallel write-efficient geometric algorithms and data structures, e.g., the first parallel write-efficient Delaunay triangulation algorithm, a construction algorithm for k-d trees, and a sorting algorithm. These algorithms all achieve linear write, optimal work and logarithmic depth. We also show write-efficient solutions for both initial construction and future dynamic updates on augmented trees, including interval trees, range trees and priority search trees, which were not achieved by any previous algorithms.

2.2 Parallel Algorithms in Computational Geometry

I have worked on both theoretical and experimental work in parallel computational geometric algorithms. They are either combined with my work on tree algorithms [11, 14], which has been introduced in Section 1, or using an analysis framework of parallel randomized incremental algorithms [3, 4] that was proposed by our paper [3].

Parallel Incremental Algorithms for Computational Geometry. My research has developed a class of parallel randomized incremental algorithms for some geometric problems. In our work, we propose the first parallel Delaunay triangulation algorithm with optimal work and logarithmic depth [3], and later made it write-efficient [4]. We also give simple and efficient solutions to closest pair [3], smallest enclosing disk [3], and some data structures for geometric problems considering read-write asymmetry [3], as introduced in Section 2.1.

2.3 Other Parallel Algorithms

I also worked on other parallel algorithms. For example, my work designed the first parallel semi-sort [8] algorithm with linear work and poly-logarithmic depth. The semi-sort problem requires reordering an array of key-value pairs, such that entries with equal keys are contiguous. This is a fundamental problem with a wide range of applications, such as the shuffle step in MapReduce paradigm and the `groupBy` operation in database languages. In addition to the theoretical bound, our algorithm also show efficiency in practice, with a good parallel speedup and performance better than other parallel integer sorting algorithms.

3 Graph Processing and Graph Mining

My work on graph algorithms can roughly be divided into two categories. The first aims at *efficiency* in processing large-scale graphs, and thus focuses on performance and parallelism, usually yielding *better theoretical cost bounds*. The second category aims at *effectiveness* in graph mining in various applications, and thus focuses on understanding data smartly, usually giving effective heuristics and solutions for optimizing certain objectives.

Parallel Algorithm for Single Source Shortest Path (SSSP) [6]. My work proposes the *radius-stepping* algorithm for the parallel SSSP problem. The parallel solution to the SSSP problem is notoriously hard because of the transitive-closure bottleneck. As a result, most of the theoretical work aims at achieving better tradeoff between work and depth. On the empirical side, δ-stepping is believed to work well in practice. However, it does not have theoretical guarantee for general cases. Our algorithm improves δ-stepping and achieves one of the best-known work-depth tradeoffs theoretically. Experiments also show that our algorithm is highly-parallelized.

Parallel Algorithm for Strong Connected Component (SCC) [3]. My research proposes a parallel SCC algorithm with improved bounds. Considering the hardness of the parallel SCC problem, most existing parallel solutions (as well as ours), both in theory and in practice, are based on efficient parallel reachability query. Our solution achieves the best-known work and depth bounds among the reachability-based SCC algorithms. A variant of this algorithm was later implemented in the parallel library Ligra and shown to be efficient in practice.

Graph Metric Embedding [5]. My research proposes an approximate graph metric embedding algorithm. This algorithm runs in $O(m \log n)$ time with high probability, which improves the previous best-known bound of $O(m \log^3 n)$. We then proved that FRT trees (the generated tree embedding) are Ramsey partitions with asymptotically tight bound, such that our algorithm can be used to accelerate a series of distance oracles.

Data Mining on Social Networks [15]. My research has formalized the problem of *influence maximization in dynamic social networks*, and propose algorithms addressing it. Influence maximization aims at selecting a small set of users in the social network to maximize the propagated influence to a cascade in their neighborhood. However, it is usually hard to keep up with the new network on rapidly-evolving social networks. Our method probes a small set of nodes at each timestamp to update their neighborhood, expecting them to reveal maximum information related to influence maximization. Our probing algorithm achieves up to 4x more activated nodes than a random probing strategy, and approaches the best possible solution by 47%-99% on different types of networks.

Protein-Protein Interaction (PPI) Network Alignment in Computational Biology [7, 13]. Biological network alignment identifies conserved regions between networks of different species, which helps to transfer information from well- to poorly-annotated species. My research conducted a comprehensive empirical study [7] of algorithms on PPI network alignment, and also proposed a novel alignment algorithm *WAVE* [13], which is shown to be effective in practice. We compare WAVE with two existing alignment algorithms MI-GRAAL and GHOST. Among all test inputs and evaluation indicators we tested, WAVE is better than both MI-GRAAL and GHOST in 72% of the cases.

4 Future Work and Research Vision

Through decades of development of computer science, classic (sequential) algorithm design and programming have been promoted and popularized, benefiting humans in many fields. With the evolution of hardware and growth in the volume of data, it has become imperative to consider the aspect of parallelism. I argue that with the development and popularization of hardware technologies, in the future, **"solving problems in parallel"** (also known as *parallel thinking*) will become the **default way of thinking for people to learn, design, and implement algorithms**.

However, parallel algorithm design and programming, for now at least, are considered notoriously hard, and less accessible to both engineers and researchers. My research goal thus lies in **enhancing the future of parallel computing, making parallelism accessible to everyone and every domain**, making it no harder than, and as popular as the manner in which sequential algorithms benefit us today. This goal requires the consideration of simplified and efficient algorithms, effective computational models, handy programming tools and frameworks, adaptability to the development of architectures, the requirements raised by applications, as well as the interplay among them. These aspects require expertise in many fields, and once acquired, would benefit many fields. I look forward to working closely with researchers in different fields to identify and solve interesting research problems, as well as applying parallelism to applications and problems in specific or interdisciplinary areas, including theory, systems, data science and data mining, computational biology, database and data warehouses, computer graphics, human-computer interaction, programming languages and machine learning. I present several important goals that my research intend to achieve for the better future of parallelism, and list a few of my ongoing works below.

Scalability and Efficiency. My past research has already presented solutions to many applications using multicore machines that can scale to hundreds of gigabytes of data with almost-perfect speedup. My long-term goal is to make it possible to use multicore machines to manipulate most of the large datasets of interest in different areas with high performance. Firstly, parallelism allows high performance, which is crucial for providing real-time responses to queries in big data management. Secondly, improving efficiency is crucial to developing more effective solutions in areas such as numerical computation, computer graphics, and deep learning. As a result, in addition to studying fundamental parallel algorithmic building blocks, in the future, I also would like to consider more specific applications in more areas such as databases, machine learning, data mining, and computational biology to address real-world problems. With expertise in parallel computing and experience of working on multiple areas, I am interested and confident in studying high-performance parallel solutions to the problems in these areas.

Accessibility to Everyone. My past research has already developed many algorithms that are simple and efficient. My long-time goal is to make parallelism as accessible as sequential algorithm design and engineering, even to novices. To achieve this goal, in theory, I hope to study and design theoretical models that can elegantly abstract the high-level idea in parallel computing, and to design simplified and efficient algorithms. In practice, I am also eager to collaborate with people working in software engineering, human-computer interaction, and programming languages to build efficient and user-friendly systems for parallel computing. I am also keen to develop courses, textbooks, and tutorials to popularize parallelism.

Adaptability to Architectures. The advent and development of new hardware pose new challenges in algorithm design. Such architecture-level considerations have also impacted the theoretical study of algorithms, leading to new theoretical models and algorithms such as I/O-efficient algorithms. Recently, many new-developed techniques have gained attention, such as NVM technologies (which motivates the study of write-efficient algorithms, see Sec. 2.1) and near-memory/in-memory computing. I believe that the systematic study of these new technologies requires effort on both theoretical can practical sides. Theoretically, models and cost analysis can help us understand the advantages and bottleneck of computations on such new hardware technologies. Practically, the experimental evaluation reflects the real performance of new algorithms, and would lead to interesting findings hidden by the theoretical analysis. I will pay close attention to the requirement and development of technologies in the industry, and work on the problems at the forefront of parallel computing.

Ongoing Work. My previous work has led to many interesting future directions, some of which I am currently working on. For example, through collaboration with other researchers, I am seeking solutions of efficient compressed representation of our parallel tree structure. I am also working on efficient batching algorithms in the single-writer database that I developed. Theoretically, we are interested in formalizing the problem in a queueing model, and developing batching strategies to guarantee stability and other properties of the system. In practice, we are interested in applying it to our database system. My collaborators and I are also studying new parallel computational models, and we are interested in showing upper- and lower-bounds on the new models.

I am eager to take part in the effort to address the important challenges in parallel computing, both in theory and in practice, as well as to apply the power of parallelism to benefit more domains. My previous work involves various areas in computer science, and therefore I am open to and confident about collaborating with other researchers. I am confident that my research results will have high impacts, and I am excited to take part in the popularization and education of parallel computing through both my research work and teaching/mentoring efforts.

References

(Authors of papers [1]–[6] are listed alphabetically, following the convention in mathematics and theoretical computer science. Others are listed by contribution.)

[1] *(by alphabetical order)* Naama Ben-David, Guy Blelloch, **Yihan Sun**, and Yuanhao Wei. Efficient single writer concurrency. *Manuscript. arXiv preprint arXiv:1803.08617*, 2018.

[2] *(by alphabetical order)* Guy Blelloch, Daniel Ferizovic, and **Yihan Sun**. Just join for parallel ordered sets. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[3] *(by alphabetical order)* Guy Blelloch, Yan Gu, Julian Shun, and **Yihan Sun**. Parallelism in randomized incremental algorithms. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[4] *(by alphabetical order)* Guy Blelloch, Yan Gu, Julian Shun, and **Yihan Sun**. Parallel write-efficient algorithms and data structures for computational geometry. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[5] *(by alphabetical order)* Guy Blelloch, Yan Gu, and **Yihan Sun**. Efficient construction of probabilistic tree embeddings. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017.

[6] *(by alphabetical order)* Guy Blelloch, Yan Gu, **Yihan Sun**, and Kanat Tangwongsan. Parallel shortest paths using radius stepping. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[7] Joseph Crawford, **Yihan Sun**, and Tijana Milenković. Fair evaluation of global network aligners. In *Algorithms for Molecular Biology*, 10:19, 2015.

[8] Yan Gu, Julian Shun, **Yihan Sun**, and Guy Blelloch. A top-down parallel semisort. In *Proceedings of Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2015.

[9] Yan Gu, **Yihan Sun**, and Guy Blelloch. Algorithmic building blocks for asymmetric memories. In *European Symposium on Algorithms (ESA)*, 2018. *arXiv preprint arXiv:1806.10370*.

[10] **Yihan Sun**, Guy Blelloch, and Daniel Ferizovic. The PAM library, 2018. `https://github.com/cmuparlay/PAM`

[11] **Yihan Sun** and Guy Blelloch. Parallel range and segment queries with augmented maps. In *Algorithm Engineering and Experiments (ALENEX)*, 2019.

[12] **Yihan Sun**, Guy Blelloch, Andrew Pavlo, and Wan Shen Lim. On supporting efficient snapshot isolation for hybrid workloads with multi-versioned indexes. *Manuscript*, 2018.

[13] **Yihan Sun**, Joseph Crawford, Jie Tang, and Tijana Milenković. Simultaneous optimization of both node and edge conservation in network alignment via WAVE. In *International Workshop on Algorithms in Bioinformatics (WABI)*, 2015.

[14] **Yihan Sun**, Daniel Ferizovic, and Guy Blelloch. PAM: parallel augmented maps. In *Proceedings of Symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2018.

[15] Honglei Zhuang, **Yihan Sun**, Jie Tang, Jialin Zhang and Xiaoming Sun. Influence maximization in dynamic social networks. In *International Conference on Data Mining (ICDM)*, 2013.