Yuan Tang *Research Statement*

My research interest is *performance science*, an interdisciplinary research area at the intersection of parallel and concurrent computing, computational and data science, programming languages, systems and theory. Specifically, my research focuses on designing and implementing provably efficient and practically fast algorithms and data structures based on solid theoretical foundation for real-world applications on modern shared-memory and distributed-memory computing systems. My approach is exploiting the best possible *balance* among work, time, or equivalently parallelism in parallel computation, space, caching, and communication. My work ranges from traditional high-performance computing [YZ08; YCT07; YFD06; Yua+03; Yua+05; YS05], parallel programming model [Cho+17c; DSY16; Yua+15; Yua+14], domain-specific compilation [Cho+17c; Yua+11a; Yua+11b], and provably efficient runtime scheduling [Cho+17a; Cho+17b; DSY16], to general algorithms and data structures [TK18; YY17; Ben+18; Mic+14; Lui+14].

Motivation and Overview

Performance is the currency in the world of computing, which can always be traded for new functionalities such as security, reliability, portability, usability, modularity, maintainability, among others. Performance was "free" thanks to Moore's Law, while is drying due to "Power Wall", "Heat Wall", and "Memory Wall". Performance is bounded by both computation and communication overheads. Computation complexity counts the most relevant and expensive arithmetic operations such as floating point operations or comparisons, while communication complexity characterizes data movements. In serial setting, it is data exchange between each pair of upper and lower levels of hierarchical memory; While in parallel setting, it is data movements among different processors.

Classic approaches are inadequate in balancing computation and communication. There are two classic programming models in parallel setting, one is "processor-aware", the other is "processor-oblivious". Processor-aware model is widely used in distributed-memory setting, as well as some shared-memory setting (say Pthread model). Such model partitions and maps statically problem's computational DAG (Directed Acyclic Graph) onto a given processor grid. This approach has several concerns. Firstly, lots of processor-aware algorithms do not work for an arbitrary number, say a prime number, of processors because they require that processor count be factorized into a 2D or 3D grid. Secondly, some problem (like matrix multiplication) requires different shapes of processor grid for optimal performance, while any given processor count is not that flexible to factorize. On the other hand, a processor-oblivious algorithm divides problem's computational DAG into as small as possible pieces and relies on a processor-aware even cache-aware runtime scheduler for a dynamic load balance. This approach again has some concerns. Firstly, you can not tell an algorithm's scalability of computation and communication without assuming some runtime scheduler, thus scheduler actually becomes part of algorithm. Secondly, some runtime scheduler [**ChowdhuryRaSi13**; **BlellochChGi08**; **BlellochFiGi11**] requires hints or a small set of instructions from algorithm for provably efficiency. By this sense, algorithm per se is not that "oblivious". Finally, dynamic load balance may induce too much runtime overheads, especially when algorithm has a non-polylog (non-low-depth) critical-path length.

I have been working on balancing computation and communication for a provable and practical efficiency by Problem-Aware Cache-Oblivious algorithm, Nested Dataflow Parallel Programming Model, Cache-Oblivious Wavefront algorithm, Space-Time Adaptive and Reductive techniques , and Pochoir Stencil Compiler, as follows.

Problem-Aware Cache-Oblivious Algorithm

In this work, we extended the ideal cache model [**FrigoLePr12**] to shared-memory multi-core and many-core setting, and formulated a balanced partitioning property by the model with an emphasis on a linear or near-linear scalability in terms of both computation and communication on an arbitrary number, even a prime number, of processors within a certain range. We designed Problem-Aware Cache-Oblivious (PACO) algorithms for several important problems, including Longest Common Subsequence (LCS), general matrix multiplication (MM) on a semiring, and comparison based sorting. We also showed how to extend the approach to Strassen-like fast algorithms and to heterogeneous computing systems. All new algorithms exhibit scalability with asymptotically better bounds over best known processor-oblivious counterparts. We showed by experiments that our new algorithms significantly outperform both processor-oblivious and processor-aware counterparts. For instances, the mean and median speedup of our PACO MM algorithm for heterogeneous computing system over Intel MKL's parallel dgemm on a 72-core machine are 48.6% and 48.8%, and the mean and median speedup of PACO MM algorithm for homogeneous computing system over MKL on a 24-core machine are 11.1% and 6.4%, respectively. The mean and median speedup of PACO LCS algorithm over processor-oblivious counterpart on a 24-core machine are 71.2% and 54.4%, over processor-aware counterpart are 86.3% and 88.3%, respectively. The mean and median speedup of PACO Sort algorithm over processor-oblivious and low-depth counterpart [**BlellochGiSi10**] implemented in PBBS [**ShunBlFi12**] are 9.3% and 9.1%, respectively.

Cache-Oblivious Wavefront Algorithm

A Cache-Oblivious Wavefront (COW) algorithm [Yua+15; Cho+16; Cho+17a; Cho+17b] performs the same divide-and-conquer recursion as classic cache-oblivious parallel algorithm, but aligns subtasks across different levels of recursion to a proceeding wavefront. Essentially, a COW algorithm schedules only on data dependency and eliminates all control dependency, thus usually yields a better time bound than classic cache-oblivious parallel counterpart. In the meantime, a COW algorithm keeps a recursive computing order among subtasks so as to preserve communication efficiency in a cache-oblivious fashion. I have developed COW algorithm for several typical dynamic-programming problems with either constant or non-constant dependency such as stencils, Floyd-Warshall All-Pairs-Shortest-Paths, Longest Common Subsequence, GAP (computing the edit distance when allowing gaps of insertions and deletions), and Parenthesis (computing the minimum cost of parenthesizing n elements). All new algorithms are provably and practically efficient in the processor-oblivious world.

Nested Dataflow Parallel Programming Model

Since classic Nested Parallel (NP) programming model (also known as fork-join model) is unable to express ***partial dependency*** in a recursive divide-and-conquer algorithm, we extend it to

Nested Dataflow (ND) model (both models belong to the processor-oblivious class). If we view a parallel computation as a DAG , where each vertex stands for a piece of code without parallel construct and each edge stands for a data dependency. There are only two possible relations between any pair of tasks *a* and *b*. If there is no path of edges connecting *a* and *b*, these two tasks can run in parallel, or no dependency; If there exists some path connecting *a* and *b*, they have to run by some serial order, or full dependency. However, a classic nested parallel algorithm has to specify a relation for any pair of subtasks derived from its recursive divide-and-conquer partitioning, where each subtask may contain several vertices of DAG. Hence it can be a partial dependency. We generalized the linguistic constructs of NP model to ND model so that it can express clearly partial dependency, as well as how it is possibly refined throughout recursion. Moreover, we designed a provably efficient space-bounded runtime scheduler for the model.

Space-Time Adaptive and Reductive Algorithm

Besides the tradeoff between computation and communication, I also studied the tradeoff between space and time. My research [YY17; Yua17] shows that a sub-linear time bound is achievable while keeping asymptotically optimal space and cache bounds for the general matrix multiplication on a semiring, Strassen-like fast matrix multiplication on a ring, and several typical dynamic-programming problems with non-constant dependencies such as Least Weight Subsequence (LWS), GAP, and Parenthesis.

The Pochoir Stencil Compiler

I have worked at MIT since 2009 on the project of "The Pochoir (Pronounced "PO-shwar") Stencil Compiler" [Yua+14; Yua+11a; Yua+11b]. Stencil computation is a special case of dynamic-programming where each cell in a multi-dimensional table is updated by a constant number of neighboring cells. Stencil is widely used in iterative PDE (Partial Differential Equation) solvers such as Jacobi, multi-grid, and adaptive mesh refinement, as well as in image processing and geometric modeling. In this project, I worked with the Pochoir team and made following contributions:

- improving the parallelism asymptotically without losing cache efficiency for higher-dimensional stencil by inventing a "hyperspace cut" algorithm;
- unifying the handling of periodic and aperiodic boundary conditions in one algorithm;
- designing and developing an embedded DSL (Domain-Specific Language) for stencil in C++;
- designing and developing a novel two-phase compilation strategy for the stencil DSL. The new compilation strategy reduces the cost of parsing and type-checking of an embedded DSL in C++.

Summary and Future Directions

I have been exploiting performance on modern multi-core and many-core computing systems by proposing Problem-Aware Cache-Oblivious (PACO) Algorithm, Nested Dataflow (ND) Parallel Programming Model, Cache-Oblivious Wavefront (COW) algorithm, and Space-Time Adaptive and Reductive (STAR) algorithm. My work centers around the balance of computation and communication with an emphasis on linear or near-linear scalability of communication overheads on an arbitrary number, even a prime number, of processors. Going forward, my work opens up new opportunities in more areas of parallel algorithms and data structures, including but not limited to, dense / sparse linear algebra, machine learning, or more generally artificial intelligence, big data, among others.

Applications to computational biology, image processing, Big Data, and large-scale machine learning: Core algorithms behind computational biology and image processing are stencil or more generally dynamic-programming computations. For instances, "Basic RNA Secondary Structure Prediction", which is a fundamental problem in computational biology, and Viterbi algorithm , which is widely used in speech recognition, are essentially dynamic-programming problems with non-constant dependencies; the H.264 video coding algorithm has a similar data dependency pattern to LCS (Longest Common Subsequence), thus a similar algorithm is applicable. My expertise in optimizing general dynamic-programming algorithms can give me big advantages in dealing with these applications.

The key of Big Data is to process larger sets of higher dimensional data faster, which lies in the sweet spot of my research. Some potential application of my research on big data can be high-performance / parallel data analytics, among others.

One of the hot research on large-scale machine learning focuses on scheduling algorithm for efficient resource management, minimizing communication and caching overhead on both distributed and shared-memory systems. My expertise in provably efficient runtime scheduling algorithms and cache- / processor-oblivious wavefront algorithms, as well as parallel programming model, can be of potential help in the field.

Conditional dependency is the dependency that can be changed by runtime values. Conditional dependencies are common in irregular computation and data structures such as (social) graphs, trees and sets. There have been extensive studies on the best way to formulate either regular or irregular parallelism. I am interested in such field.

References

[Ben+18] Michael A. Bender, Rezaul Alam Chowdhury, Pramod Ganapathi, Samuel McCauley, and Yuan Tang. "The range 1 query (R1Q) problem". In: *Theor. Comput. Sci.* **(TCS)** 743 (2018), pp. 130–147 (cit. on p. 1).

[TK18] Yuan Tang and Haibin Kan. "Non-orthogonal Homothetic Range Partial-Sum Query on Integer Grids - [Extended Abstract]". In: *Frontiers in Algorithmics - 12th International Workshop, FAW 2018, Guangzhou, China, May 8-10, 2018, Proceedings.* **FAW'18**. 2018, pp. 273–285 (cit. on p. 1).

[Cho+17a] Rezaul A. Chowdhury, Pramod Ganapathi, Yuan Tang, and Jesmin Jahan Tithi. "Poster: Provably Efficient Scheduling of Cache-Oblivious Wavefront Algorithms". In: *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* **PPoPP'17**. Austin, TX, USA, 2017 (cit. on pp. 1, 2).

[Cho+17b] Rezaul A. Chowdhury, Pramod Ganapathi, Yuan Tang, and Jesmin Jahan Tithi. "Provably Efficient Scheduling of Cache-Oblivious Wavefront Algorithms". In: *Proceedings of the 19-th ACM Symposium on Parallelism in Algorithms and Architectures.* **SPAA'17**. Washington D.C. USA, 2017 (cit. on pp. 1, 2).

[Cho+17c] Rezaul Chowdhury, Pramod Ganapathi, Stephen L. Tschudi, Jesmin Jahan Tithi, Charles Bachmeier, Charles E. Leiserson, Armando Solar-Lezama, Bradley C. Kuszmaul, and Yuan Tang. “Autogen: Automatic Discovery of Efficient Recursive Divide-8-Conquer Algorithms for Solving Dynamic Programming Problems”. In: *ACM Transactions on Parallel Computing.* **(TOPC)** 4.1 (2017), 4:1–4:30 (cit. on p. 1).

[Yua17] Yuan Tang. “Brief Announcement: STAR (Space-Time Adaptive and Reductive) Algorithms for Real-World Space-Time Optimality”. In: *Proceedings of the 19-th ACM Symposium on Parallelism in Algorithms and Architectures.* **SPAA’17**. Washington D.C. USA, 2017 (cit. on p. 3).

[YY17] Yuan Tang and Ronghui You. “Poster: STAR (Space-Time Adaptive and Reductive) Algorithms for Real-World Space-Time Optimality”. In: *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* **PPoPP’17**. Austin, TX, USA, 2017 (cit. on pp. 1, 3).

[Cho+16] Rezaul A. Chowdhury, Pramod Ganapathi, Yuan Tang, and Jesmin Jahan Tithi. “Cache-Oblivious Wavefront algorithms for dynamic programming problems: efficient scheduling with optimal cache performance and high parallelism”. In: *The International Conference for High Performance Computing, Networking, Storage, and Analysis.* **SC’16** Research Poster. 2016 (cit. on p. 2).

[DSY16] David Dinh, Harsha Vardhan Simhadri, and Yuan Tang. “Extending the Nested Parallel Model to the Nested Dataflow Model with Provably Efficient Schedulers”. In: *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures.* **SPAA’16**. Pacific Grove, California, USA, 2016 (cit. on p. 1).

[Yua+15] Yuan Tang, Ronghui You, Haibin Kan, Jesmin Jahan Tithi, Pramod Ganapathi, and Rezaul A. Chowdhury. “Cache-Oblivious Wavefront: Improving Parallelism of Recursive Dynamic Programming Algorithms without Losing Cache-efficiency”. In: *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming.* **PPoPP ’15**. San Francisco, CA, USA: ACM, 2015, pp. 205–214. ISBN: 978-1-4503-3205-7 (cit. on pp. 1, 2).

[Lui+14] Luis Barba, Otfried Cheong, Jean-Lou De Carufel, Michael Gene Dobbins, Rudolf Fleischer, Akitoshi Kawamura, Matias Korman, Yoshio Okamoto, János Pach, Yuan Tang, Takeshi Tokuyama, Sander Verdonschot, and Tianhao Wang. “Weight Balancing on Boundaries and Skeletons”. In: *Proceedings of the 30th Annual Symposium on Computational Geometry.* **SoCG ’14**. Kyoto, Japan: ACM, 2014, 436:436–436:443. ISBN: 978-1-4503-2594-3 (cit. on p. 1).

[Mic+14] Michael A. Bender, Rezaul Alam Chowdhury, Pramod Ganapathi, Samuel McCauley, and Yuan Tang. “The Range 1 Query (R1Q) Problem”. In: *Proceedings 20th International Conference on Computing and Combinatorics.* **COCOON ’14**. 2014, pp. 116–128 (cit. on p. 1).

[Yua+14] Yuan Tang, Ronghui You, Haibin Kan, Jesmin Jahan Tithi, Pramod Ganapathi, and Rezaul A. Chowdhury. “Improving Parallelism of Recursive Stencil Computations without Sacrificing Cache Performance”. In: *Proceedings of the 2nd Workshop on Optimizing Stencil Computations*. **WOSC ’14**. Portland, Oregon, USA: ACM, 2014, pp. 1–7. ISBN: 978-1-4503-2308-6 (cit. on pp. 1, 3).

[Yua+11a] Yuan Tang, Rezaul Alam Chowdhury, Bradley C. Kuszmaul, Chi-Keung Luk, and Charles E. Leiserson. “The Pochoir Stencil Compiler”. In: *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures*. **SPAA ’11**. San Jose, California, USA: ACM, 2011, pp. 117–128. ISBN: 978-1-4503-0743-7 (cit. on pp. 1, 3).

[Yua+11b] Yuan Tang, Rezaul Chowdhury, Chi-keung Luk, and Charles E. Leiserson. “Coding Stencil Computations Using the Pochoir Stencil-Specification Language”. In: *3rd USENIX Workshop on Hot Topics in Parallelism*. **HotPar ’11**. USENIX, 2011 (cit. on pp. 1, 3).

[YZ08] Yuan Tang and Yunquan Zhang. “Utilizing the Multi-threading Techniques to Improve the Two-Level Checkpoint/Rollback System for MPI Applications.” In: *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications*. **HPCC ’08**. IEEE, 2008, pp. 864–869. ISBN: 978-0-7695-3352-0 (cit. on p. 1).

[YCT07] Yunquan Zhang, Ying Chen, and Yuan Tang. “Block Size Selection of Parallel LU and QR on PVP-based and RISC-based Supercomputers”. In: *Proceedings of the Asian Technology Information Program’s (ATIP’s) 3rd Workshop on High Performance Computing in China: Solution Approaches to Impediments for High Performance Computing*. **HPC CHINA ’07**. Reno, Nevada: ACM, 2007, pp. 115–125. ISBN: 978-1-59593-903-6 (cit. on p. 1).

[YFD06] Yuan Tang, G.E. Fagg, and J.J. Dongarra. “Proposal of MPI Operation Level Checkpoint/Rollback and One Implementation”. In: *Proceedings of the 6th IEEE International Symposium on Cluster Computing and Grid*. Vol. 1. **CCGrid ’06**. May 2006, pp. 27–34 (cit. on p. 1).

[YS05] Yuan Tang and Jiachang Sun. “Research of NASA Parallel Benchmark’s communication pattern on Myrinet 2000 (in Chinese)”. In: *Journal on Numerical Methods and Computer Applications* (2005). ISSN: 1000-3266 (cit. on p. 1).

[Yua+05] Yuan Tang, Jiachang Sun, Yunquan Zhang, and Linbo Zhang. “New Consideration on the Evaluation Model of Cluster Area Network (in Chinese)”. In: *Journal of Software* 16(6) (2005), pp. 1131–1139. ISSN: 1000-9825 (cit. on p. 1).

[Yua+03] Yuan Tang, Yunquan Zhang, Jiachang Sun, and Yucheng Li. “Hardware Impact on Communication Performance of Beowulf LINUX Cluster”. In: *Proceedings of the 21st IASTED International Multi-Conference on Applied Informatics*. Innsbruck, Austria, Feb. 2003, pp. 495–500 (cit. on p. 1).