Distributed Computing Column 73
SPAA 2018 Review

Jennifer L. Welch
Department of Computer Science and Engineering
Texas A&M University, College Station, TX 77843-3112, USA
welch@cse.tamu.edu

This issue's column contains a review of the 2018 ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) by Laxman Dhulipa. Laxman has provided an insightful overview of the keynote lectures and highlights of the contributed presentations, that I hope will inspire you to track down and read the papers, if you haven't done so already.

Many thanks to Laxman for his contribution!

Call for contributions: I welcome suggestions for material to include in this column, including news, reviews, open problems, tutorials and surveys, either exposing the community to new and interesting topics, or providing new insight on well-studied topics by organizing them in new ways.

SPAA 2018 Review

Laxman Dhulipa
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
ldhulipa@cs.cmu.edu

The 2018 ACM Symposium on Parallelism in Algorithms and Architectures was held on July 16–18 in Vienna, Austria. The conference took place on the TU Wien campus in the Engineering building, an ideal location that was walking distance to central Vienna as well as historic buildings like Karlskirche and St. Stephen's Cathedral. Thanks to the organizers, presenters and attendees for their hard work that made this conference possible!

The conference included two keynote lectures this year, by Charles E. Leiserson on *The Resurgence of Software Performance Engineering* and David A. Bader on *Massive-Scale Streaming Analytics: Models, Parallelism, and Real-World Applications*. The papers *Minimum Cuts in Near-Linear Work and Low Depth* by Barbara Geissmann and Lukas Gianinazzi and *Theoretically Efficient Parallel Graph Algorithms Can Be Fast and Scalable* by Guy E. Blelloch, Julian Shun and myself were co-awarded the best-paper award this year.

Unfortunately this report cannot cover all of the interesting results presented at the conference. However, I hope it gives a taste of what this SPAA was like, enabling the interested reader to overview some of the results presented this year, and hopefully attend SPAA in the future.

Day 1

The first session of the conference was on graph algorithms, and started with one of the best papers in the conference, presented by Lukas Gianinazzi. This work gives a nearly work-efficient parallel algorithm for computing minimum cuts, and improves the work of an RNC minimum cut algorithm from $O(n^2 \log n)$ to $O(m \log^4 n)$, which is the same as the work of the fastest sequential minimum cut algorithm, up to polylog terms [9]. The main tool is a parallel data structure that maintains the sum of weights along paths in a tree, and can process batches of weight updates and queries in parallel. Shay Solomon gave an interesting talk on dynamic representation of sparse networks; their goal is to reduce the space used by each vertex to maintain its adjacency information to something better than the maximum degree in the graph, e.g., the arboricity.

The keynote talk today was given by Charles E. Leiserson on software performance engineering [11]. Charles argued for the increasing importance of software performance engineering based on two recent trends: the emergence of cloud computing and the end of Moore's law. One of the main ideas put forth during the talk was that good performance is a currency—it can often be exchanged for other desirable features like code clarity or a convenient API. Based on this view, the common advice handed to new engineers of "premature optimization is the root of all evil" seems simplistic and fated to make writing fast code seem more like magic than science. Charles ended his talk on an uplifting note—although the steady progress of Moore's law may have ended, the widespread availability of multicores which provide reliable and consistent performance coupled with the growing science of software performance engineering shows a path forward to the next forty years of performance improvements.

The afternoon session focused on matrix algorithms. Ojas Parekh talked about constant depth threshold circuits (TC^0) for matrix multiplication that have sub-cubic size based on Strassen's algorithm. Amir Gholami talked about integrating model, batch and domain parallelism in training neural networks, with an approach inspired by communication-avoiding algorithms from numerical linear algebra. The session ended with brief announcements, including a talk on local approximations of the PageRank centrality of a vertex.

The last session of the day was on concurrent data structures. Gal Milman presented a lock-free queue that supports batching of arbitrary sequences of insertions and deletions. Their work provides a safe way to adapt concurrent data structures in the setting where the data structure can batch the incoming operations, without making assumptions on the underlying distribution of operations. Thomas Ropars presented a wait-free resizable hash table based on extendible hashing optimized for the case where the table is resized infrequently. The session ended with a talk by Dan Alistarh which introduced a relaxed notion of linearizability that relates the execution of a randomized concurrent data structure to a relaxed sequential process.

The SPAA business meeting was held after the last session today and had some lively discussions. It touched on dissertation awards, the possibility of giving increased student travel grant funding (enthusiastically supported by all of the students present), and where to host future iterations of SPAA. Despite the interest in hosting SPAA in Hawaii or Thailand, an established member of the community soberly pointed out that while warm weather and beaches are no doubt desirable, more central locations are better for increased student attendance and the health of the community.

Day 2

The second day started with a session on distributed graph algorithms. Jukka Suomela's talk introduced a non-deterministic version of the congested clique and presented analogs of classic complexity results such as the time-hierarchy theorem in this model.

David Bader gave the keynote talk today on data structures and algorithms for dynamic graph processing [3]. The talk provided an overview of David and his group's previous and current work over the past decade on parallel algorithms over graph streams, and the STINGER data structure for dynamic graphs [7]. In this setting there is a graph that is updated by a stream of updates (edge insertions, deletions, weight updates). Running static or dynamic algorithms on a STINGER data structure is usually done by phase-concurrently accessing the data structure, i.e., updates pause when a computation is running, and the computation waits for updates to be finished before running. However, real-world update streams have very high rates, which makes

phase-concurrent access unrealistic. David's talk addresses this problem by presenting a new model for graph algorithms that allows the underlying graph to be mutated while the algorithm runs. The talk also provided a great overview of an emerging field at the intersection of parallel data structures, dynamic graph algorithms, and graph processing systems.

The third session today was on the topic of caching. Quanquan Liu gave a talk on the complexity of measuring the tradeoff between the size of the cache and number of memory transfers using the red-blue pebble game [10]. Guy Even discussed algorithms for an extension of the generalized caching problem, where an adversary can change the weight of an arbitrary page in cache. Helen Xu ended the session with a talk on cache-adaptive algorithms, a topic introduced in SPAA'14 that measures how well an algorithm can make use of changing cache sizes [4]. One of the interesting results in the new paper is a general technique that transforms a class of non-cache-adaptive algorithms (including Strassen's algorithm) to cache-adaptive algorithms.

The last session of the day was on algorithms for Non-Volatile Memories (NVM). Many of the talks in this session were motivated by the fact that writes are much more costly than writes in emerging NVM technologies. Yan Gu discussed computational geometry algorithms for planar Delaunay triangulation, k-d trees, and other problems on trees that perform asymptotically fewer writes than standard algorithms for these problems. Charles McGuffey gave the next talk on the Parallel Persistent Memory Model, which captures programming parallel algorithms on persistent memory when processors can fail. One interesting result in their work is that the bounds achieved by the work-stealing scheduler of Arora, Blumofe, and Plaxton [2] can be recovered in the parallel persistent memory model, with a term that depends on the probability of a processor failure.

The second day ended with a banquet held at the Vienna Rathaus, a beautiful neo-Gothic building within walking distance of the conference venue. Jeremy Fineman, the PC chair this year, presented the best-paper awards to the recipients. Phil Gibbons gave a great talk on SPAA's history, with trivia about past SPAA locations, influential previous papers presented at SPAA, and frequent contributers to SPAA.

Day 3

The third day started with a session on scheduling and load balancing. Unfortunately I was unable to attend this session, and so cannot summarize the interesting results presented here.

The morning session continued with on parallel data structures, and brief announcements. Wei Quan Lim talked about parallel working-set structures, which extend a recent idea of implicit batching [1] to self-adjusting structures where accesses to different elements may have drastically different costs. Tsiv Kopelowitz gave an interesting talk on parallelizing Frederickson's classic deterministic minimum spanning forest algorithm [8], which has a worst case complexity of $O(\sqrt{n})$. The authors obtain a nearly work-efficient deterministic parallel algorithm, which processes a single update in $O(\sqrt{n}\log n)$ work and $O(\log n)$ depth. It would be interesting to see whether the data structure can handle batches of updates and queries in low-depth while preserving work-efficiency. TB Schardl presented a brief announcement on the *Open Cilk* project[1], an open-source implementation of the Cilk concurrency platform which is leading the development and maintenance of Cilk after GCC dropped Cilk Plus support in 2017.

The third session continued after lunch on online algorithms. Giorgi Nadiradze talked about the transactional conflict problem. They consider two back-off policies used when conflicts are detected

[1] `http://cilk.mit.edu/`

by real systems, and showed that the analysis of the running time penalty incurred by the policies is equivalent to variants of the ski-rental problem.

The final session of the conference was on graph and mesh computations. The session started with the other winner of the best paper award, presented by myself. I spoke about our benchmark of theoretically-efficient shared-memory algorithms[2], and graph-processing optimizations used in our work. The results show that classic work-efficient and low-depth parallel algorithms designed for the PRAM, like the Tarjan-Vishkin biconnectivity algorithm, Borůvka's MSF algorithm, as well as recently developed parallel strongly-connected component algorithms [5] are practical and can process graphs with hundreds of billions of edges in minutes on a single machine with a terabyte of RAM [6]. Peter Robinson gave an interesting talk on improved upper and lower bounds for PageRank and triangle enumeration in the k-machine model. The session ended with William Hasenplaugh, who gave an engaging talk on Laika, a scheduler for graph computations on meshes that achieves asymptotically fewer cache misses by reordering the mesh based on space-filing curves.

References

[1] K. Agrawal, J. T. Fineman, K. Lu, B. Sheridan, J. Sukha, and R. Utterback. Provably good scheduling for parallel programs that use data structures through implicit batching. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.

[2] N. S. Arora, R. D. Blumofe, and C. G. Plaxton. Thread scheduling for multiprogrammed multiprocessors. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 1998.

[3] D. A. Bader. Massive-scale streaming analytics: Models, parallelism, and real-world applications. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[4] M. A. Bender, R. Ebrahimi, J. T. Fineman, G. Ghasemiesfeh, R. Johnson, and S. McCauley. Cache-adaptive algorithms. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

[5] G. E. Blelloch, Y. Gu, J. Shun, and Y. Sun. Parallelism in randomized incremental algorithms. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016.

[6] L. Dhulipala, G. E. Blelloch, and J. Shun. Theoretically efficient parallel graph algorithms can be fast and scalable. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[7] D. Ediger, R. McColl, J. Riedy, and D. A. Bader. Stinger: High performance data structure for streaming graphs. In *IEEE Conference on High Performance Extreme Computing (HPEC)*, 2012.

[8] G. N. Frederickson. Data structures for on-line updating of minimum spanning trees. In *ACM Symposium on Theory of Computing (STOC)*, 1983.

[9] B. Geissmann and L. Gianinazzi. Parallel minimum cuts in near-linear work and low depth. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.

[2] `https://github.com/ldhulipala/gbbs`

[10] H. Jia-Wei and H. T. Kung. I/O complexity: The red-blue pebble game. In *ACM Symposium on Theory of Computing (STOC)*, 1981.

[11] C. E. Leiserson. The resurgence of software performance engineering. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2018.