Research Statement

Prashant Pandey

I design and build theoretically well-founded data structures for big data problems in computational biology, databases, and file systems. My work is having a transformative impact on computational biology, because I am essentially rebuilding the entire genomic and transcriptomic analysis tool chain around data structures of my design, and I am obtaining significant gains in speed, accuracy, space-efficiency, and simplicity. This work is described in four flagship conference papers (IMSB, RECOMB, SIGMOD, WABI) and two top journal papers (Cell Systems, Bioinformatics). It is in use in at least ten top labs around the world. In file systems, I have two conference papers (FAST) and two journal papers (ToS), including a Best Paper Award. These papers show how to use modern data structures to overcome decades-old trade-offs in file-system design, yielding orders-of-magnitude performance gains.

My research has pursued two basic strategies for managing big data: shrink it, and organize it. For applications in computational biology and databases that can afford a small number of errors in results, I shrink data in favor of space-efficiency and speed. On the other hand, for file systems, where accuracy is critical, I organize data in a disk friendly way to efficiently perform disk accesses and achieve better performance.

In computational biology, we showed how to use recent theoretical advances in the theory of compact and succinct data structures to shrink genomic and transcriptomic data down to the size where it can be processed on a laptop. I first developed Squeakr [9] to solve k-mer counting. k-mer (a length-k substring) counting is one of the most common and arguably one of the simplest preliminary step in many bioinformatics algorithms. The number of existing papers on this problem suggest, however, that efficient execution of this task, with reasonable memory use, is far from trivial. Squeakr uses the counting quotient filter [9] and being smaller in size achieved an order-of-magnitude faster queries and at the same time several times faster construction compared to the state-of-the-art. I then developed deBGR [7], which efficiently represents de Bruijn graphs which are at the heart of many sequence analyses. Despite the computational benefits that the de Bruijn graph provides, the graph still tends to require a substantial amount of memory for large data sets making many analyses slow or even impossible on a single machine. deBGR uses an error-correction algorithm that exploits invariants of de Bruijn graphs to iteratively correct all the approximation errors and store them space-efficiently, making it practical to perform many de Bruijn graph analyses in RAM. I also developed a large-scale search index Mantis [5], which attempts to solve the fundamental problem of large-scale sequence-search over a sequence database. It is simpler, smaller, and faster than production systems. This now enables people to analyze and assemble those fundamentally large genomes which were not possible before on a single machine.

Much of my computational biology work builds on my research into Approximate Membership Query (AMQ) data structures. AMQs such as the Bloom filter [2], are almost five decades old and have been a workhorse in numerous applications in databases, storage systems, networks, computational biology, and other domains. However, many applications must work around limitations in the capabilities or performance of current AMQs, making these applications more complex and less performant. I developed the Counting Quotient Filter (CQF) [8] which supports approximate membership testing and counting the occurrences of items in a data set. The CQF is small and fast, supports counting (even on skewed data sets), and has features that other AMQs lack. The CQF is a feature rich and practical data structure which is backed by strong theoretical results and offers an order-of-magnitude faster operations than Bloom filters. Based on my research, several (at least ten) top teams around the world have ripped out Bloom filters in their code and replaced them with counting quotient filters.

Similar to succinct data structures [6] the CQF is also based on rank-and-select operations. Though, asymptotically optimal, many succinct data structures have poor performance due to the constants involved in bit-vector rank-and-select implementations [10]. However, we use new x86 bit-manipulation instructions to speed up rank-and-select operations and our implementation can also be useful for other rank-and-select-based data structures which were not practical before.

In file systems, we showed how we can better organize data on disk using write-optimized dictionaries to speed up file system operations. We developed BetrFS [3, 11], the first in-kernel file system that uses B$^{\varepsilon}$-tree, an asymptotically optimal write-optimized dictionary. BetrFS can match or outperform traditional file systems on almost every operation, some by an order of magnitude. I implemented zone trees in BetrFS that were a significant part of the BetrFS paper at FAST 2016 that got the Best paper award. Our paper was Runners-up at FAST 2015.

I have a fair bit of industry experience through internships. During my internships I worked on problems different from my own research. This helped me in two ways. First, in getting a better understanding of domains other than my own research. Second, learning on how to apply principles from one domain to solve problems in a different one. While interning at Intel Labs, I worked on an encrypted FUSE file system using Intel SGX [4]. I also developed new SGX instructions to fork processes securely from a process already running inside an SGX environment. At Google, I designed and implemented an extension to the ext4 file system for cryptographically ensuring file integrity. Google is currently working to integrate this extension into Android and the mainline Linux kernel. While at Google, I also worked on the core data structures of Spanner, Googles geo-distributed big database.

For future work, I hope to scale Mantis to petabytes of data and deploy it as a public service available to scientists around the world. Prior genomic search tools, such as BLAST [1], has played an instrumental role in advancing fundamental research in bioinformatics and evidently been cited over 70K times. I hope to have a similar or greater impact with Mantis, since Mantis can search through even more data than BLAST. I also hope to continue shrinking computational biology data so that we can start performing analyses on phones or other devices that can be carried into the field, where internet or even laptops may not be available or practical. I also plan to show how my work on AMQs can be used to accelerate write-optimized file and storage systems. I hope to apply the basic strategy behind deBGR – using invariants from the underlying graph to create a compact representation – to other big data graph problems, such as in social networks, natural languages, etc.

I believe that we need to be at the intersection of theory and practice in order to solve next generation problems.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[3] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. BetrFS: A right-optimized write-optimized file system. In J. Schindler and E. Zadok, editors, *Proc. 13th USENIX Conference on File and Storage Technologies (FAST)*, pages 301–315, February 2015.

[4] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. *HASP@ ISCA*, 10, 2013.

[5] P. Pandey, F. Almodaresi, M. A. Bender, M. Ferdman, R. Johnson, and R. Patro. Mantis: A fast, small, and exact large-scale sequence search index. *bioRxiv*, page 217372, 2017.

[6] P. Pandey, M. A. Bender, and R. Johnson. A fast x86 implementation of select. *CoRR*, abs/1706.00990, 2017.

[7] P. Pandey, M. A. Bender, R. Johnson, and R. Patro. debgr: an efficient and near-exact representation of the weighted de bruijn graph. *Bioinformatics*, 33(14):i133–i141, 2017.

[8] P. Pandey, M. A. Bender, R. Johnson, and R. Patro. A general-purpose counting filter: Making every bit count. In S. Salihoglu, W. Zhou, R. Chirkova, J. Yang, and D. Suciu, editors, *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 775–787. ACM, 2017.

[9] P. Pandey, M. A. Bender, R. Johnson, and R. Patro. Squeakr: An exact and approximate k-mer counting system. *Bioinformatics*, page btx636, 2017.

[10] S. Vigna. Broadword implementation of rank/select queries. In *International Workshop on Experimental and Efficient Algorithms*, pages 154–168. Springer, 2008.

[11] J. Yuan, Y. Zhan, W. Jannen, P. Pandey, A. Akshintala, K. Chandnani, P. Deo, Z. Kasheff, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Optimizing every operation in a write-optimized file system. In *Proc. 14th USENIX Conference on File and Storage Technologies (FAST)*, 2016.