

Report from Dagstuhl Seminar 19111

Theoretical Foundations of Storage Systems

Edited by

Martin Farach-Colton¹, Inge Li Gørtz², Rob Johnson³, and Donald E. Porter⁴

¹ Rutgers University – Piscataway, NJ, US, martin@farach-colton.com

² Technical University of Denmark – Lyngby, DK, inge@dtu.dk

³ VMware – Palo Alto, US, rtjohnsocs42@gmail.com

⁴ University of North Carolina at Chapel Hill, US, porter@cs.unc.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 19111 “Theoretical Foundations of Storage Systems.” This seminar brought together researchers from two distinct communities – algorithms researchers with an interest in external memory and systems researchers with an interest in storage – with the objective of improving the design of future storage systems.

Seminar March 10–15, 2019 – <http://www.dagstuhl.de/19111>

2012 ACM Subject Classification Information systems → Data structures, Information systems → Key-value stores, Information systems → Information storage systems, Theory of computation → Concurrency, Theory of computation → Data structures design and analysis, Theory of computation → Caching and paging algorithms, Theory of computation → Concurrent algorithms, Hardware → Non-volatile memory, Software and its engineering → File systems management, Software and its engineering → Memory management, Software and its engineering → Process management

Keywords and phrases Storage Systems, External Memory Algorithms

Digital Object Identifier 10.4230/DagRep.9.3.39

Edited in cooperation with Michael A. Bender

1 Executive Summary

Michael A. Bender

Martin Farach-Colton

Inge Li Gørtz

Rob Johnson

Donald E. Porter

License © Creative Commons BY 3.0 Unported license

© Michael A. Bender, Martin Farach-Colton, Inge Li Gørtz, Rob Johnson, and Donald E. Porter

Storage systems, including databases and file systems, are at the heart of all large data applications. Recently, some storage systems have incorporated theoretical advances in data organization techniques, with substantial improvements in performance. However, there is little contact between the systems designers who build storage systems and theoreticians who design new ways to organize data. This Seminar worked to bridge this gap, to the benefit of both communities and to improve the design of all storage systems.

External-memory algorithms are those where the data is too large to fit in memory, and hence needs to be stored on disk and accessed using I/O. Algorithmic analysis of such algorithms therefore focuses on the number of I/Os needed to complete a computation,



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 3.0 Unported license

Theoretical Foundations of Storage Systems, *Dagstuhl Reports*, Vol. 9, Issue 3, pp. 39–51

Editors: Martin Farach-Colton, Inge Li Gørtz, Rob Johnson, and Donald E. Porter



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

rather than the number of instructions. This is because an I/O can be many orders of magnitude slower than a machine instruction and therefore I/Os can be the bottleneck in such computations.

The theoretical analysis of such external-memory algorithms has produced many exciting results in the last two decades. Many of these are directly relevant to practical applications, but only a few have made the leap to deployment. This Seminar brought together theoreticians, who have an extensive understanding of the state of the art in external-memory data structures, and storage systems researchers and practitioners, who understand the details of the problems that need to be solved.

Specific questions that were addressed in the workshop include the following:

- How can we use the huge improvements in string data structures to improve storage systems that manipulate strings? Many data structures, such as LSMs and B^ϵ -trees, rely heavily on the assumption that keys are indivisible and small.
- How can we use new multi-dimensional data indexes in working systems?
- Many indexes suffer from fragmentation. Are there data structure improvements that would allow efficient storage on disks that are nearly full? Currently, disks are kept only a fraction full because the performance of existing data structures decays dramatically as the disk fills. This suggests another problem:
- How can theoretical models be improved to capture such issues as:
 - full disk: The external memory model, the disk is of unbounded size.
 - sequential access: Both hard disks and SSDs require very large blocks of sequential I/O to capture a large fraction of bandwidth. The external-memory model assumes that disks are random access.
- Concurrency: Can data structures be designed to exploit memory locality on disk while maintaining concurrency in RAM?

The storage system world is in a ferment as new hardware becomes available. Now is the time to establish deep partnerships across disciplines in computer science to solve some of the most pressing big data infrastructure problems.

2 Table of Contents

Executive Summary

<i>Michael A. Bender, Martin Farach-Colton, Inge Li Gørtz, Rob Johnson, and Donald E. Porter</i>	39
--	----

Cross-Community Communication

Tutorials	43
“Say What?” Sessions	43
Talk Karaoke	43

Overview of Talks

The Design of the File Storage Service for Oracle’s Cloud <i>Bradley C. Kuszmaul</i>	43
Tutorial on Write Optimization <i>Michael A. Bender</i>	44
Formal methods and file systems <i>Thomas J. Ridge</i>	44
Tutorial on Nonvolatile Memory, including flash memory <i>Sam H. Noh</i>	44
File System Aging <i>Martin Farach-Colton</i>	45
What is a sector (and why should I care)? <i>Gala Yadgar</i>	45
Tutorial on SMR Disks <i>William Jannen</i>	45
Tales from the Algorithm Engineering Trenches: STXXL, Thrill, NVMe SSDs, and RDMA <i>Timo Bingmann</i>	46
Operating Systems Class: A Series of Algorithmic Disappointments <i>Donald E. Porter</i>	46
High-Performance Computing Storage System Challenges for Theoreticians <i>Jonathan Berry</i>	47
Traversal and generation of very large graphs in memory hierarchies <i>Ulrich Carsten Meyer</i>	47
Maintaining connected components for infinite graph streams <i>Cynthia A. Phillips and Jonathan Berry</i>	47
Introduction to Deduplication <i>Gala Yadgar and Geoff Kuenning</i>	48
Adaptive Bloom and Cuckoo Filters <i>Shikha Singh and Samuel McCauley</i>	48
Achieving Optimal Backlog in Multi-Processor Cup Emptying Games <i>William Kuszmaul</i>	49

42 **19111 – Theoretical Foundations of Storage Systems**

Online Bipartite Matching with Amortized $O(\log^2 n)$ Replacements <i>Eva Rotenberg</i>	49
A Practical Approach to Filter Adaptivity <i>Sam MacCauley</i>	50
Participants	51

3 Cross-Community Communication

3.1 Tutorials

The workshop comprised systems and theory tutorials to help participants get on the same page. These included tutorials on write-optimized data structures, deduplication, SSDs, hard drives, shingled hard drives, formal methods and file systems. All participants of the workshop—theoreticians and practitioners alike—were encouraged to target their talks for more general audiences and to make their talks tutorial-like.

3.2 “Say What?” Sessions

Any community has its own set of buzzwords and its own corpus of common knowledge. These make communication easier with a community but harder across communities. The workshop dealt with these communication challenges by having what we called *say what?* sessions—whenever any audience member heard an unfamiliar term from a speaker, they would call out “say what?” and the speaker wrote the term on the chalkboard. One side of the board was devoted to system terms that were foreign to most theoreticians, and the other side was devoted to theory terms that were foreign to most system researchers. During the Say What? sessions, which took place in the evenings, these terms were demystified. These sessions lead to what was described by many participants as a warm and collegial atmosphere throughout the conference.


3.3 Talk Karaoke

One of the cross-community activities of the conference was an activity called talk karaoke. In this activity, speakers were given a slide deck that they had never seen before, and needed to give a three-minute talk based on the slide deck. Systems speakers were given theory slides and theory speakers were given systems slides. This was a wonderful bonding activity full of laughter even after several hours, and several participants commented that this was one of the most enjoyable activities that they had seen at any conference or workshop.

4 Overview of Talks

4.1 The Design of the File Storage Service for Oracle’s Cloud

Bradley C. Kuszmaul (Oracle Corporation – Burlington, US)


License  Creative Commons BY 3.0 Unported license
© Bradley C. Kuszmaul

File Storage Service is an elastic filesystem provided as a managed NFS service in Oracle Cloud Infrastructure. Using a pipelined Paxos implementation, we implemented a scalable block store that provides linearizable multipage limited-size transactions. On top of the block store, we built a scalable B-tree that provides linearizable multikey limited-size transactions. By using self-validating B-tree nodes and performing all B-tree housekeeping operations as separate transactions, each key in a B-tree transaction requires only one page in the

underlying block transaction. The B-tree holds the filesystem metadata. The filesystem provides snapshots by using versioned key-value pairs. The entire system is programmed using a nonblocking lock-free programming style. The presentation servers maintain no persistent local state, with any state kept in the B-tree, making it easy to scale up and failover the presentation servers. We use a non-scalable Paxos-replicated hash table to store configuration information required to bootstrap the system. The system throughput can be predicted by comparing an estimate of the network bandwidth needed for replication to the network bandwidth provided by the hardware. Latency on an unloaded system is about 4 times higher than a Linux NFS server backed by NVMe, reflecting the cost of replication.

4.2 Tutorial on Write Optimization

Michael A. Bender (Stony Brook University, US)

License  Creative Commons BY 3.0 Unported license
© Michael A. Bender

Write-optimized dictionaries (WODs), such as LSM trees and B^ε-trees, are increasingly used in databases and file systems. Such data structures support very fast insertions without sacrificing lookup performance.

This talk explains how WODs can substantially reduce the I/O cost of many workloads, enabling some applications to scale by orders of magnitude. In contrast, traditional data structures, such as B-trees, are often I/O bound on these workloads. The talk explores write-optimization from the perspective of foundational theory, parallelization, and applications.

4.3 Formal methods and file systems


Thomas J. Ridge (University of Leicester, UK)

License  Creative Commons BY 3.0 Unported license
© Thomas J. Ridge

This is a short talk giving an overview of formal methods (in particular, theorem provers such as Isabelle), and their application to file systems and related technologies (key-value stores, databases).

4.4 Tutorial on Nonvolatile Memory, including flash memory

Sam H. Noh (Ulsan National Institute of Science and Technology, KR)

License  Creative Commons BY 3.0 Unported license
© Sam H. Noh

This talk was on the introduction of nonvolatile memory (including flash memory) and the issues it raises in regards to storage systems. For flash memory based storage, in particular, we describe how Ultra Low Latency SSDs are changing the view of secondary storage. For byte addressable nonvolatile memory, we describe issues such as consistency and persistency that rises when using this new media. Such issues forces us to revisit data structure that have traditionally been used in storage systems.

4.5 File System Aging

Martin Farach-Colton (Rutgers University – Piscataway, US)

License © Creative Commons BY 3.0 Unported license
© Martin Farach-Colton

Joint work of Alexander Conway, Ainesh Bakshi, Yizheng Jiao, William Jannen, Yang Zhan, Jun Yuan, Michael A. Bender, Rob Johnson, Bradley C. Kuszmaul, Donald E. Porter, Martin Farach-Colton

Main reference Alexander Conway, Ainesh Bakshi, Yizheng Jiao, William Jannen, Yang Zhan, Jun Yuan, Michael A. Bender, Rob Johnson, Bradley C. Kuszmaul, Donald E. Porter, Martin Farach-Colton: “File Systems Fated for Senescence? Nonsense, Says Science!”, in Proc. of the 15th USENIX Conference on File and Storage Technologies, FAST 2017, Santa Clara, CA, USA, February 27 – March 2, 2017, pp. 45–58, USENIX Association, 2017.

URL <https://www.usenix.org/conference/fast17/technical-sessions/presentation/conway>

File systems have heuristics to avoid fragmentation and aging, where aging is the degradation of performance due to fragmentation. However, a theoretical analysis suggests that modern file systems are unable to stay ahead of fragmentation under normal workloads. On the other hand, Be-tree file systems should be able to avoid fragmentation and aging by using large block sizes that amortize the cost of a seek.

We show a simple workload that elicits aging in a variety of file systems. Indeed, it is easy to make file systems slow down by a factor of 20 under normal workloads. In contrast, BetrFS, a Be-tree file system, avoids aging, as predicted.

4.6 What is a sector (and why should I care)?

Gala Yadgar (Technion – Haifa, IL)

License © Creative Commons BY 3.0 Unported license
© Gala Yadgar

This short tutorial explained the basic concepts of hard-disk drives: arm movements, rotation delays, and logical organization of data on disk. It was constructed to highlight the difference between the current theoretical model used to measure the I/O efficiency of algorithms and data structures (DAM), and the real costs incurred by these mechanisms in practical systems. A better understanding of this gap by the theoretical community will hopefully motivate new theoretical models and constructions.

4.7 Tutorial on SMR Disks

William Jannen (Williams College – Williamstown, MA, US)

License © Creative Commons BY 3.0 Unported license
© William Jannen

This talk gives an introduction to SMR disks.

4.8 Tales from the Algorithm Engineering Trenches: STXXL, Thrill, NVMe SSDs, and RDMA

Timo Bingmann (KIT – Karlsruher Institut für Technologie, DE)

License © Creative Commons BY 3.0 Unported license

© Timo Bingmann

Joint work of Roman Dementiev, Lutz Kettner, Peter Sanders, Timo Bingmann, Michael Axtmann, Emanuel Jöbstl, Sebastian Lamm, Huyen Chau Nguyen, Alexander Noe, Sebastian Schlag, Matthias Stumpp, Tobias Sturm

Main reference Timo Bingmann, Michael Axtmann, Emanuel Jöbstl, Sebastian Lamm, Huyen Chau Nguyen, Alexander Noe, Sebastian Schlag, Matthias Stumpp, Tobias Sturm, Peter Sanders: “Thrill: High-performance algorithmic distributed batch data processing with C++”, in Proc. of the 2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016, pp. 172–183, IEEE Computer Society, 2016.

URL <http://dx.doi.org/10.1109/BigData.2016.7840603>

This talk starts with a retrospective on STXXL [1], which is a library of external memory algorithms and data structures started back in 2003. STXXL contains multiple efficient external containers such as a sorter, a B-tree, and a priority queue, which were integral for developing many larger external algorithms in later years. STXXL also pioneered pipelining of algorithm stages to save constant factors due to round trips of data to disks.

Thrill [2] is our new parallel distributed computation framework in C++ which draws on the experiences from STXXL. Using Thrill one can pipeline scalable parallel primitives such as Map, Sort, Reduce, and PrefixSum to construct larger complex algorithms.

In the talk we motivate why parallelism in algorithms, as used in Thrill, is absolutely necessary to fully utilize the current generation of fast external memory storage devices. To underline this, we presented fresh experiments of sequential scanning speed and random access latency on rotational disks, NAND SSDs, and NVMe SSDs.

References

- 1 Roman Dementiev, Lutz Kettner, and Peter Sanders. “STXXL: Standard Template Library for XXL Data Sets”. In: *Software & Practice and Experience* 38.6 (2008), pages 589–637. ISSN: 1097-024X. John Wiley & Sons, Ltd.
- 2 Timo Bingmann, Michael Axtmann, Emanuel Jöbstl, Sebastian Lamm, Huyen Chau Nguyen, Alexander Noe, Sebastian Schlag, Matthias Stumpp, Tobias Sturm, and Peter Sanders. “Thrill: High-Performance Algorithmic Distributed Batch Data Processing with C++”. In: *IEEE International Conference on Big Data*. preprint arXiv:1608.05634, pages 172–183. ISBN: 978-1-4673-9005-7. IEEE. Dec. 2016.

4.9 Operating Systems Class: A Series of Algorithmic Disappointments

Donald E. Porter (University of North Carolina at Chapel Hill, US)

This talk walks through several examples of practical OS challenges where the textbook solutions are unsatisfying, and perhaps standard theory models do not capture essential requirements for a practical solution. These examples include CPU rebalancing, physical page allocation, and directory caching.

4.10 High-Performance Computing Storage System Challenges for Theoreticians

Jonathan Berry (Sandia National Laboratories – Albuquerque, US)

License © Creative Commons BY 3.0 Unported license
© Jonathan Berry

Joint work of Jonathan Beryy, Cynthia Phillips, Steven Plimpton, Alexandra Porter, Timothy Shead

I present some I/O challenges associated with high-performance computing. For scientific computing, a primary challenge is doing checkpoint/restart. This differs from checkpointing in transactional database systems since the constraints are different. For data analysis, I outline some challenges with streaming computation. Limited storage decreases accuracy of analytics, motivating work that combines high-speed streaming with external-memory algorithms.

4.11 Traversal and generation of very large graphs in memory hierarchies

Ulrich Carsten Meyer (Goethe-Universität – Frankfurt a. M., DE)

License © Creative Commons BY 3.0 Unported license
© Ulrich Carsten Meyer

Very large graphs arise naturally in such diverse domains as social networks, web search, computational biology, machine learning and more. Even simple tasks like traversing these graphs become challenging once they cannot be stored in the main memory of one machine. If the graph is stored in external memory, then many algorithms that perform very well on graphs that are stored in internal memory, become inefficient because of the large number of I/Os they incur. In order to alleviate the I/O bottleneck, a number of external memory graph algorithms have been designed with provable worst-case guarantees. In the talk I highlight some techniques used in the design and engineering of such algorithms and survey the state-of-the-art in I/O-efficient graph traversal algorithms. In the engineering process mentioned above, artificially generated input graphs play an important role for systematic testing and tuning. In big data settings, however, not only processing huge graphs but also the efficient generation of appropriate test instances itself becomes challenging. Hence, I will also discuss two recent results for large-scale graph generation under resource constraints.

4.12 Maintaining connected components for infinite graph streams

Cynthia A. Phillips (Sandia National Labs – Albuquerque, US) and Jonathan Berry (Sandia National Labs – Albuquerque, US)

License © Creative Commons BY 3.0 Unported license
© Cynthia A. Phillips and Jonathan Berry

Joint work of Cynthia A. Phillips, Jonathan Berry, Steven Plimpton, Alexandra Porter, Timothy Shead

We present an algorithm to maintain the connected components of a graph that arrives as an infinite stream of edges. We show how to use distributed parallel processors to avoid file read/writes for a multi-pass streaming algorithm. This then allows us to run a version of the algorithm in a setting where there is no end to the input, as is the case when monitoring event streams. We describe our design decisions/tradeoff choices when trying to manage large storage and provide fast, accurate query response.

4.13 Introduction to Deduplication

Gala Yadgar (Technion – Haifa, IL) and Geoff Kuenning (Harvey Mudd College – Claremont, US)


License  Creative Commons BY 3.0 Unported license
 © Gala Yadgar and Geoff Kuenning

Deduplication has become an essential method for reducing storage requirements. This talk covered the fundamentals of how deduplication works and some of the challenges involved in making it efficient.

It turns out that many of the challenges posed by deduplication are theoretical problems, including, chunking, fingerprinting, approximate lookups, and indexing in general. The purpose of this talk was to place these theoretical challenges in their system-level context, thus promoting joint (theory-systems) research on the subject

4.14 Adaptive Bloom and Cuckoo Filters

Shikha Singh (Williams College – Williamstown, US) and Samuel McCauley (Williams College – Williamstown, US)

License  Creative Commons BY 3.0 Unported license
 © Shikha Singh and Samuel McCauley
Main reference Michael A. Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, Shikha Singh: “Bloom Filters, Adaptivity, and the Dictionary Problem”, CoRR, Vol. abs/1711.01616, 2017.
URL <https://arxiv.org/abs/1711.01616>

When a large set is stored remotely, membership queries (that is, checking if an element is present in the set) can be expensive. Instead of querying the remote set each time, usually a small in-memory sketch is consulted first to determine “approximate membership.” Bloom filters are one such widely-used approximate-membership query data structures (AMQs). A Bloom filter maintains a compact probabilistic representation of a set S . On a query for an element in S , it guarantees a correct response of “present” but on a query for an element not in S , it may return “present” with a small false-positive probability ε .

The false-positive probability guarantee ε of most AMQs holds for a single query or a random query workload. In particular, an adversary who discovers a false positive of a Bloom filter can query it repeatedly driving its false-positive rate to 1.

We say an AMQ is adaptive if it guarantees a false-positive probability of ε for every query, regardless of answers to previous queries. In this talk, we will present adaptive variants of popular filters like Bloom filters and Cuckoo filters.

4.15 Achieving Optimal Backlog in Multi-Processor Cup Emptying Games

William Kuszmaul (MIT – Cambridge, US)

License © Creative Commons BY 3.0 Unported license
© William Kuszmaul

Joint work of Michael A. Bender, Martin Farach-Colton, William Kuszmaul

Main reference Michael A. Bender, Martin Farach-Colton, William Kuszmaul: “Achieving Optimal Backlog in Multi-Processor Cup Games”, CoRR, Vol. abs/1904.02861, 2019.

URL <http://arxiv.org/abs/1904.02861>

Many problems in processor scheduling, deamortization, and buffer management can be modeled as single- and multi-processor cup games.

At the beginning of the single-processor n -cup game, all cups are empty. In each step of the game, d distributes $1 - \varepsilon$ units of water among the cups, and then a selects a cup and removes up to 1 unit of water from it. The goal of the emptier is to minimize the amount of water in the fullest cup, also known as the backlog. The greedy algorithm (i.e., empty from the fullest cup) is known to achieve backlog $O(\log n)$, and no deterministic algorithm can do better.

We show that the performance of the greedy algorithm can be exponentially improved with a small amount of randomization: After each step and for any $k \geq \Omega(\log \varepsilon^{-1})$, the emptier achieves backlog at most $O(k)$ with probability at least $1 - O(2^{-2^k})$. We call our algorithm the *randomized greedy* because it follows from a smoothed analysis of the (standard) greedy algorithm.

In each step of the p -processor n -cup game, the filler distributes $p(1 - \varepsilon)$ unit of water among the cups, with no cup receiving more than $1 - \delta$ units of water, and then the emptier selects p cups and removes 1 unit of water from each. Proving nontrivial bounds on the backlog for the multi-processor cup game has remained open for decades. We present a simple analysis of the greedy algorithm for the multi-processor cup game, establishing a backlog of $O(\varepsilon^{-1} \log n)$, as long as $\delta > 1/\text{poly}(n)$.

Turning to randomized algorithms, we find that the backlog drops to constant. Specifically, we show that if ε and δ satisfy reasonable constraints, then there exists an algorithm that bounds the backlog after a given step by 3 with probability at least $1 - O(\exp(-\Omega(\varepsilon^2 p)))$.

We prove that our results are asymptotically optimal for constant ε , in the sense that no algorithms can achieve better bounds, up to constant factors in the backlog and in p . Moreover, we prove robustness results, demonstrating that our randomized algorithms continue to behave well even when placed in bad starting states.

4.16 Online Bipartite Matching with Amortized $O(\log^2 n)$ Replacements

Eva Rotenberg (DTU – Copenhagen, DK)

License © Creative Commons BY 3.0 Unported license
© Eva Rotenberg

Joint work of Aaron Bernstein, Jacob Holmm, Eva Rotenberg

In this talk, we introduce the online with recourse model, and give an example: online bipartite matchings. Here, all the vertices on one side of the bipartition are given, and the vertices on the other side arrive one by one with all their incident edges. The goal is to maintain a maximum matching while minimizing the number of changes (replacements) to the matching. With Aaron Bernstein and Jacob Holm, we recently showed that the greedy

algorithm that always takes the shortest augmenting path from the newly inserted vertex (denoted the SAP protocol) uses at most amortized $O(\log^2 n)$ replacements per insertion, where n is the total number of vertices inserted, where the previous best strategy achieved amortized $O(\sqrt{n})$.

4.17 A Practical Approach to Filter Adaptivity

Sam MacCauley (Williams College – Williamstown, US)

License  Creative Commons BY 3.0 Unported license
© Sam MacCauley

Filters (such as Bloom Filters) are a fundamental data structure that speed up dictionary accesses by storing a compressed representation of a set. Filters are very space efficient, but can make one-sided errors: they may report that a query element is stored in the filter when it is not (a false positive). Recent research has focused on designing methods to dynamically adapt filters. Dynamically adapting to queries reduces the number of false positives in the case that elements are queried over and over again. Such adaptations have the potential to greatly improve performance in many practical use cases.

In this talk I will discuss a new Adaptive Cuckoo Filter with theoretical guarantees on its performance. Specifically, I will show that the new Adaptive Cuckoo Filter is support sensitive—the number of false positives it incurs depends on the number of unique queries performed. This captures the notion of fixing previously-seen false positives, and gives greatly improved performance guarantees on skewed datasets. I will also show lower bounds on the performance of a large family of filters against a worst-case adversary. In particular, I will show that by carefully tailoring the query sequence based on the results of previous queries, the adversary can cause most queries to be a false positive.

Participants

- Michael A. Bender
Stony Brook University, US
- Ioana Bercea
Tel Aviv University, IL
- Jonathan Berry
Sandia National Labs –
Albuquerque, US
- Philip Bille
Technical University of Denmark
– Lyngby, DK
- Timo Bingmann
KIT – Karlsruhe Institut für
Technologie, DE
- Alexander Conway
Rutgers University –
Piscataway, US
- Guy Even
Tel Aviv University, IL
- Martin Farach-Colton
Rutgers University –
Piscataway, US
- Jeremy Fineman
Georgetown University –
Washington, US
- Johannes Fischer
TU Dortmund, DE
- Pawel Gawrychowski
University of Wrocław, PL
- Seth Gilbert
National University of
Singapore, SG
- Inge Li Gørtz
Technical University of Denmark
– Lyngby, DK
- Magnús M. Halldórsson
Reykjavik University, IS
- William Jannen
Williams College –
Williamstown, US
- Rob Johnson
VMware – Palo Alto, US
- Tomasz Kociumaka
Univ. of Warsaw, PL & Bar-Ilan
Univ. Ramat-Gan, IL
- Geoff Kuenning
Harvey Mudd College –
Claremont, US
- Bradley C. Kuszmaul
Oracle Labs – Burlington, US
- William Kuszmaul
MIT – Cambridge, US
- Simon Mauras
University Paris-Diderot, FR
- Samuel McCauley
Williams College –
Williamstown, US
- Ulrich Carsten Meyer
Goethe-Universität –
Frankfurt a. M., DE
- Miguel A. Mosteiro
Pace University – New York, US
- Ian Munro
University of Waterloo, CA
- Sam H. Noh
Ulsan National Institute of
Science and Technology, KR
- Prashant Pandey
Carnegie Mellon University –
Pittsburgh, US
- Nikos Parotsidis
University of Copenhagen, DK
- Cynthia A. Phillips
Sandia National Labs –
Albuquerque, US
- Solon Pissis
CWI – Amsterdam, NL
- Donald E. Porter
University of North Carolina at
Chapel Hill, US
- Simon J. Puglisi
University of Helsinki, FI
- Tom Ridge
University of Leicester, GB
- Eva Rotenberg
Technical University of Denmark
– Lyngby, DK
- Siddhartha Sen
Microsoft – New York, US
- Francesco Silvestri
University of Padova, IT
- Shikha Singh
Wellesley College, US
- Meng-Tsung Tsai
National Chiao-Tung University –
Hsinchu, TW
- Przemyslaw Uznanski
University of Wrocław, PL
- Janet Vorobyeva
Stony Brook University, US
- Gala Yadgar
Technion – Haifa, IL

