

Challenges Facing Tomorrow's Datacenter: Summary of the LADiS Workshop

Robbert van Renesse
Cornell University
rvr@cs.cornell.edu

Rodrigo Rodrigues
MPI-SWS
rodrigo@mpi-sws.org

Mike Spreitzer
IBM Research
mspreitz@us.ibm.com

Christopher Stewart
University of Rochester
stewart@cs.rochester.edu

Doug Terry
Microsoft Research
Doug.Terry@microsoft.com

Franco Travostino
eBay Inc.
ftravostino@ebay.com

Abstract The 2008 workshop on Large-Scale Distributed Systems and Middleware (LADiS) addressed challenges facing tomorrow's datacenter. Over the course of three days, attendees laid forth an ambitious research agenda that covered hot topics, ranging from fault-tolerance algorithms to performance management to cloud computing. This report summarizes key themes and conclusions from the workshop.

1. INTRODUCTION

Today's largest Internet services operate large distributed systems with a small staff. Reports of 1,000 servers to every 1 human operator are common.¹ However, continued progress in the management of large-scale systems faces fundamental obstacles, especially for cloud computing systems. LADiS brought together world-class researchers to address these challenges. The workshop provided a forum for presenting and discussing cutting-edge ideas. Participants offered positive feedback rather than critical review.

There were 25 technical presentations and 3 invited keynotes. Topics covered a broad spectrum from fault-tolerance algorithms to performance management to cloud computing. By the end of the workshop, attendees reached a general consensus that tomorrow's datacenters and Internet services should:

- **Monitor and aggregate data efficiently.** The health of large datacenters that host many services should be query-able across administrative domains. Customers and system administrators should have low-overhead access to consistent snapshots of the system.
- **Perform well all of the time.** Many factors affect the performance of large-scale services— some can be

¹James Hamilton, LADiS 2008 Keynote, <http://mvdirona.com/jrh/TalksAndPapers/>

understood, others can not. Tomorrow's datacenters will need a management infrastructure that delivers high, dependable, and efficient performance.

- **Use scalable protocols to communicate and synchronize data.** For datacenters with 10^6 nodes, pairwise communication between each node is unlikely to work. However, the best communication strategy for large-scale systems (*i.e.*, one that yields high performance without over-burdening programmers) is an open problem.
- **Be robust to both common and catastrophic failures.** Hardware and software failures threaten entire services. Power outages and natural disasters, threaten entire datacenters. Data integrity and service availability must persist during such calamities.
- **Tolerate Byzantine failures.** Tomorrow's datacenters will attract attackers, human errors, and unforeseen problems. Byzantine fault tolerance provides assurance that the system can operate correctly even when it has been compromised.

In the remainder of this summary, we first discuss the LADiS keynote presentations. Then, we cover each of the above themes in detail by reviewing related presentations and presenting open research challenges.

2. KEYNOTE TALKS

The keynote talks covered industrial viewpoints gathered from diverse experiences with deployed systems. Judging from the comments of attendees, these talks succeeded in stimulating a dialogue around the fit between research and practice, and how to close gaps between them.

Discussion at LADiS Jerry Cuomo of IBM shared his view of virtualization as a progressive thought that starts from *atomic virtualization*— the single-server scope which has had a long storied past— and grows into *molecular virtualization*— with a scope extended to a whole topology of components that can be effortlessly managed as an ensemble throughout their lifecycle. When embodied in an off-the-shelf middleware component like IBM WebSphere, this progression holds promise to decrease complexity and

increase confidence in the operation of virtual clusters, cloud computing, and similar exercises in horizontal scaling.

James Hamilton from Microsoft made the case that there are significant economies of scale in the operation of datacenters, albeit with a very high cost of entry. He substantiated this position with a comprehensive review of environmental factors, hardware power laws, and application requirements, which clearly point to different dynamics for the enterprise-scale operation (e.g., from 10^3 to 10^4 servers) vs. the Internet-scale operation (e.g., 10^5 and beyond). Building upon such different economies of scale, James went on to predict that Internet-scale hosted services will be a large component of many enterprise solutions.

Franco Travostino and Randy Shoup of eBay gave a backstage tour of one of the world's largest Web sites (eBay). In the first part of the talk, they reviewed the architectural patterns that have come to underlie their IT infrastructure and its massive scale-out. In the second part, they described some "big rocks" that will shape their agenda for the next five years. From emergent behaviors to harnessing the power of clouds, the same deeply-held architectural patterns used to date will continue to rule, with an even greater purpose than before. For an organization to succeed at that scale of operation, it must embrace scaling as a lifestyle choice rather than a point-in-time exercise.

Conclusions From different vantage points, the keynotes surfaced a number of opportunities to meaningfully improve the state of practice in large-scale distributed systems. Increasingly, these opportunities are a must-have in order to scale to 10^5 servers. Opportunities include:

- Tools to organize and manage meta data in real time;
- Machine-learning toolkits to manage convergence, perturbation, and dampening, thereby improving our confidence in the behavior of large-scale systems.
- Methods to ascertain whether lower-quality components in large numbers are more likely to meet a service level objective in aggregate than an equivalent set of fewer, higher-quality components;
- Data management techniques incorporating relaxed consistency models (the speakers often drew upon Brewer's CAP conjecture);
- Algorithms and tools that assist in the placement of components and/or data in the core vs. the edge of a topology;
- Clean-sheet redesign of system layers in light of the widening gap in relative performance of hardware components (e.g., CPU outpacing all means to feed it) and the emerging new layer in the storage hierarchy, *i.e.* solid-state storage devices;
- Graceful degradation and admission control that maximize accrued value in the presence of operational constraints;
- Active power management tailored to workloads, software/hardware integration, and operational policies therein;
- Practices for experience-driven innovation which benefits from a large live user community and is equipped to either succeed or fail fast, fail cheap.

3. DATA COLLECTION AND DISSEMINATION

Datacenters are able to deal with tremendous amounts of data and computation by extensive partitioning and specialization. This gives rise to deploying thousands to even millions of compute nodes, each with modest capabilities, but together creating a formidable platform. The management of such a platform is non-trivial. The load on nodes will be highly dynamic. Node and communication failures will be the rule, and no longer exceptional. "N-squared" communication will not scale. We need to come up with new ways to collect, aggregate, and disseminate information among the nodes that scale to millions of nodes, adapt to dynamics in the datacenters, and are tolerant of node and network failures.

Discussion at LADiS Anne-Marie Kermarrec presented a peer-to-peer system that divides the collection of nodes into a set of equal-sized slices according to some attribute of the nodes such as CPU capacity. Doing so is useful for structuring an overlay mesh, with high capacity nodes forming the backbone of the mesh, and low capacity nodes moved toward the edges of the network.

Robbert van Renesse presented techniques to improve performance of gossip-based communication in large systems. The amount of information that can be gossiped is limited by the amount of bandwidth that is available. Using careful reconciliation techniques and flow control, the predictable performance of gossip can be guaranteed.

Maxime Monod presented work on overcoming the problems with gossip's homogeneity. Normally gossip imposes a uniform load on the nodes of the system. Yet in many real-world architectures the nodes are very heterogeneous in their capacities. Different nodes can also derive very different amounts of benefit from the gossip. Maxime told us about HEAP, a protocol that adapts its load on each node according to how the node's capacity compares to the system-wide average; that average is determined by a gossip-based aggregation protocol. HEAP also has a sub-protocol named LIFT for discovering nodes that declare high capacity but do not provide correspondingly high service.

Rolf Stadler discussed scalable real-time aggregation of attributes of nodes. Such a service is useful for monitoring of a datacenter or individual applications within the datacenter, and may also be used for load balancing. Using gossip, it is possible to monitor a large number of nodes continuously, with low latency of aggregation and robustness in the face of failures.

Gregory Chockler talked about scalable pub/sub-style dissemination of information. In emerging applications, sub-

scription patterns are highly dynamic and the number of participants is large. By clustering subscribers with similar interests, an overlay network can be constructed that carefully balances node degree with per-topic graph diameter.

Joe Hoffert discussed issues related to timely event notification. Various protocols have been developed that provide various guarantees. Using modeling, it is possible to reason about composing such protocols and optimizing utility for applications.

Conclusions Information collection and dissemination remains an interesting topic of research because of ever changing scale of infrastructure, communication patterns and requirements, and failure models. Mechanisms must support service level agreements, while minimizing unnecessary overheads. Today's challenges for research in this area include:

- **Can information monitoring and distribution services degrade gracefully in the face of large-scale failures?** Meltdowns are incredibly costly to datacenters. Information collection and dissemination has to continue to work in the face of highly adverse conditions, and never contribute to these conditions.
- **Can information collection and dissemination scale to geographically distributed datacenters across high-latency network links?** In order to deal with efficient access from anywhere in the world, as well as surviving the failure of an entire datacenter, it is increasingly important to deploy multiple geographically distributed datacenters. The high latency that separates those datacenters leads to new challenges in data collection and dissemination.
- **How can researchers obtain access to actual experience and realistic communication requirements in large datacenters, as well as experimental facilities?** Datacenters are not eager to share horror stories with the public, as this might diminish their perceived value. Also, data like subscription patterns might reveal privacy sensitive information. We need to find ways for researchers to have access to these kinds of data. A secondary concern is how researchers can perform experiments in realistic datacenter environments.

4. RESOURCE MANAGEMENT

Large-scale datacenters are replete with resources, such as processors, storage, and internal network bandwidth. *Resource management* seeks to distribute resources in a way that meets user demands and minimizes costs. However, resource management for Internet services is hard, because 1) their workloads and configurations change over time and 2) the services themselves are volatile due to software updates, security patches, and new features.

Discussion at LADiS Franco Travostino described management techniques used in eBay's datacenters. For instance, minimum performance requirements are codified in internal service-level agreements. When the service level is

too high or too low, the service's configuration is adapted—*e.g.*, by changing the number of processors. Travostino also discussed the importance of model-driven management to maintain cross-tier relationships and to maximize performance metrics like throughput-per-watt.

Steven Ko spoke about operations that require a partial snapshot of the system state—*e.g.*, CPU utilization at 5:00pm for all nodes running application X. Such on-demand operations will be especially important for datacenters hosting many administrative domains. Ko described HoneyAdapt, a scheduler for large-scale distributed systems that uses on-demand operations.

Mike Spreitzer described decentralized approaches to resource management. Decentralized approaches are robust to node failures. Also, they may scale better than centralized solutions. As one example, Spreitzer presented an algorithm for component placement that relied only on pair-wise communication between components.

Kai Shen discussed performance anomalies— performance degradations that are not intended by the system design. Performance anomalies are often caused by poorly implemented system components or emergent behaviors. They are hard to identify because performance degradations can be subtle— unlike system crash bugs. Kai proposed the use of performance models and decision trees to identify anomalies and predict the conditions under which they are likely to occur.

Conclusions Proper resource management will be essential for the next generation of large-scale distributed systems. Cloud systems must efficiently distribute resources to maximize profits. Services that continue to operate their own IT infrastructure will undoubtedly look to maximize customer satisfaction, meet service level expectations, and lower costs. However, there are many open questions for resource management.

- **What will the management infrastructure for tomorrow's datacenters look like?**

As larger datacenters support more services, practical issues about data collection and privileges across administrative domains will affect resource management goals and tools.

- **Which management decisions can be driven by performance modeling, algorithms, and automated testing?**

Large-scale systems are too complex for manual resource management. Researchers must continue to identify opportunities to substantially reduce the management burden through automation.

- **Can resource management be effective for services that occasionally exhibit poor performance?**

Recovery-oriented computing (ROC) led to highly available services from faulty components. The next step is to create services that are still well behaved and dependable in the presence of partial failures and performance anomalies in their components.

5. COMMUNICATION AND AGREEMENT PRIMITIVES

Many datacenter architectures include generic, low-level services for system-wide communication, agreement, or coherency. The services discussed at LADiS span a range of semantics and qualities of service. The service might be as simple (semantically, anyway!) as IP multicast: a best-effort datagram service that is, nonetheless, difficult to implement in an acceptable way in the large-scale datacenter. Moving up the scale a bit, content-based publish/subscribe provides higher function but still no coherence or synchrony guarantees between nodes. Adding guarantees such as reliable delivery, total ordering, or causal ordering provides a stronger platform on which to build higher layers — and is more challenging. Gossip provides dissemination with eventual consistency. Paxos is a valued building block, but utilizing it in the design of an agreement service for a datacenter setting has interesting issues. Another way of communicating is via a shared memory abstraction. Transactional memory is known in the non-distributed setting, and can be extended to be a distributed service in the datacenter. Replicating data brings additional issues due to the difficulty of achieving consistency, availability, and performance. In some cases designers choose to tolerate Byzantine faults.

Discussion at LADiS Ymir Vigfusson presented Dr. Multicast (MCMD), which makes IP multicast a good citizen in the datacenter. The normal, un-tamed IP multicast is problematic if a large number of multicast groups is actually used; even a few hundred is enough to overload some routers and/or NICs. Dr. Multicast is interposed at the IP multicast API and solves the problems with scalability in number of groups by a combination of two tactics. One tactic is to use a few network-level groups to carry the traffic of many API-level groups when there is great overlap in the receiver sets. The other tactic is to do some group communication via unicasts instead of true network-level multicast. A central server collects statistics and periodically updates the decisions about which tactics to employ for which groups. Administrators can impose limits on the number of network-level groups in the system and the number used by a given NIC; the server optimizes the use of the allowed numbers of groups.

Antonio Carzaniga presented work on scalable content-based publish/subscribe. Maintaining 1) a spanning tree per source node and, 2) a predicate per (source node, outward link) pair at all nodes delivers optimal latency and decent load balancing — at a cost of $O(N^2)$ storage in each node. Re-organizing the routing table to simply have a predicate per outward link (the predicate can test the source node) can yield significant savings because those predicates can often be expressed more compactly than a disjunction of the predicates used in the baseline design. We presume here that each message is tagged with its source node. In many cases several source nodes are treated identically, and the routing predicates can be further simplified if the message is tagged with an agreed representative of its set of equivalent sources. Further reductions are possible if the tag is updated at each step of forwarding to be the most recent representative. The preceding ideas allow space savings with no loss of routing accuracy. Antonio also presented a particular protocol that

goes further: it uses Bloom filters to compress the predicates with some loss of accuracy (only “safe” loss — messages will go to at least the nodes that want them) and the source computes the list of receivers for a given message.

Benjamin Reed presented the atomic broadcast service used inside Yahoo!’s ZooKeeper service. The atomic broadcast service delivers a given message to all correct servers or none of them, and delivers the messages in the same total and causal order to all those servers. This service can be understood in terms of its few differences from Paxos. This service provides causal ordering while Paxos does not. Paxos tolerates message loss and re-ordering, while the Yahoo! service is built on TCP and relies on the fact that TCP provides a FIFO channel; this makes it easy for Yahoo!’s service to achieve causal ordering. Yahoo!’s service uses epochs to short-cut parts of the recovery procedures; the particular leader election technique used also eases the recovery job by selecting as leader a node that already holds the most in-flight data.

Yair Amir presented work on a leader election algorithm for Paxos that is good for the datacenter setting. The liveness of Paxos depends critically on the liveness of the leader — which depends on the leader election technique. Yair presented a leader election technique that produces a very stable selection of leader when there is churn in the system — which is the norm in the large datacenter. Also, the leader adjusts the pace of proposals so that all of the followers can keep up. Experiments were done with and without batching of proposals and acceptances, and show that batching both increases throughput and decreases average latency.

Paolo Romano discussed the idea of extending Software Transactional Memory (STM) to the distributed setting. Paolo compared a non-distributed STM benchmark with a database benchmark, and noted that the STM benchmark workload exhibited much greater heterogeneity in transaction size, the difference between read set size and write set size was much less for STM, and most of the STM transactions completed in less time than it takes to do an atomic broadcast. These differences make the techniques used to build distributed databases less effective for distributed STM. Paolo is working on the following ideas to improve the distribution of STM: speculative transaction execution, using Bloom filters to encode read and write sets, and adaptive selection of replication strategy.

Marcos Aguilera showed how to use Sinfonia minitransactions to build a distributed B-tree. A Sinfonia minitransaction, introduced at SOSP 2007, provides the ACID properties and consists of a set of equality tests and a set of reads and a set of writes; the reads and writes are done if all the equality tests are met. The data is a simple linear space distributed across “memory nodes”. The Sinfonia library provides configurable fault tolerance, memory node replication, transactional backups, and debugging facilities. The basic idea of building a B-tree using these minitransactions is to do a series of reads, to find the data sought or the location where an insertion will be done, and finally one minitransaction checks that none of the relevant tree nodes on the path from the root have changed and, if so, makes the

ultimate read or change. Marcos mentioned two optimizations, lazy caching of inner tree nodes in the clients (based on tree node version numbers) and eager replication of tree node version numbers at servers.

Conclusions There was a lot of discussion of gossip, pub/sub, and Paxos, resulting in several research questions:

- **How can Paxos be incorporated into a complete datacenter design?** Paxos leaves many important things unsaid, and the best choices for the datacenter are still not settled. It seems the leader election technique is a particularly important part, and we have seen how to use it to solve two different sets of problems (good liveness vs. efficiently providing causal ordering and fast recovery); is there a leader election technique that does both?
- **Can pub/sub protocols be extended to meet the needs of the datacenter?** While parts of the pub/sub problem have been addressed well, we still do not have a solution that does a good job on all aspects: efficiently supporting a large number of groups, of various sizes and throughputs, over heterogeneous nodes with content-based subscriptions — in a self-assembling way when the pubs and subs arrive on-line in a system with dynamic membership and flow control is needed.
- **What is the role of transactional memory in the datacenter?** While transactional memory has become a phenom in parallel programming, its relevance in large-scale distributed datacenters is still disputed.

6. ROBUSTNESS

Given the massive numbers of processors, disks, and other components in modern datacenters, transient and permanent failures happen routinely and must be dealt with as a matter of course. Increasingly, as computations and data are routed and shared across geographically distributed datacenters, services are affected not only by local failures but also by disruptions in remote locations. Cloud-based services must tolerate a wide variety of failures to achieve the levels of robustness and availability that users expect. Redundancy is the key to building reliable systems out of unreliable hardware and software components.

Discussion at LADiS Hakim Weatherspoon, in a paper with Lakshmi Ganesh, Tudor Marian, Mahesh Balakrishnan, and Ken Birman, presented a technique for replicating files at remote locations while maintaining good performance. The goal is to reduce the chance of data loss by mirroring local data in a remote datacenter. Writers want to get assurances that their data is safely stored in multiple sites, but do not want to wait for acknowledgments from the remote mirror. The proposed "network-sync" scheme uses network level redundancy so that network failures do not result in data loss. The Smoke and Mirrors File System was built using network-sync.

Venugopalan Ramasubramanian (aka Rama) discussed joint work with John MacCormick, Nick Murphy, Kunal Talwar, Udi Wieder, Junfeng Yang, and Lidong Zhou on placing replicated data within a datacenter storage system. Their approach tries to balance resource usage across servers and also takes into account which servers share hardware components and, hence, are likely to fail together. It relies on hashing to determine a set of potential storage sites that readers and writers can choose to access. The multi-choice paradigm combines the benefits of hashing with those of directory-based systems.

Conclusions Although techniques for building robust, highly available systems have been widely studied in the research community, the applicability of such techniques in the context of large-scale, geographically distributed datacenters remains in doubt. Technologies such as two-phase commit and quorum consensus, for example, may be acceptable for clusters within a datacenter but incur unacceptably high overheads when used on large numbers of machines or across datacenters. The discussions at LADiS raised a number of open issues regarding robustness, including:

- **Should the same approach for achieving robustness be used across datacenters as within a datacenter?** The failure and network characteristics within a datacenter often drastically differ from those across datacenters. For instance, high communication latency between datacenters in different parts of the world may dictate new types of replication and coordination protocols. However, datacenters would like to avoid the cost of supporting different protocols, and so techniques that adapt to different situations are desired.
- **What robustness can be achieved with edge-based solutions?** Hardware and software systems at the ingress/egress points of datacenters can help mask various kinds of failures and performance problems using techniques such as rerouting traffic or masquerading. Achieving robustness this way can be simpler than incorporating fault-tolerance mechanisms into the datacenter service software itself, but we do not understand the limitations of edge-based solutions.
- **What consistency guarantees are acceptable to users of cloud-based services, and how can services provide such guarantees?** One-copy serializability has long been viewed as the preferred correctness criteria for replicated systems. However, users have shown a willingness to tolerate weaker forms of consistency in exchange for increased availability or performance, and essentially all existing services discussed at the workshop have given up on strong consistency. A research challenge remains in precisely formulating the inherent tradeoffs between consistency and other system properties, choosing the right guarantee for particular cloud-based services, and providing tools for service developers to deal with variable consistency.

7. BYZANTINE FAULT TOLERANCE

The increased value and importance of Internet services deployed in large-scale datacenters causes a pressing need for

these services to work correctly and continuously. To address this challenge in an environment where faults are inevitable, and even commonplace, existing services deployed on datacenters rely on replication to tolerate crash faults. However, such techniques do not protect against failures such as memory mutations that cause a server to provide incorrect replies to certain requests. Such failure modes have been observed in the wild.

Byzantine fault tolerance (BFT) addresses this issue by making no assumptions about how faulty components behave. In other words, a service that uses Byzantine-fault-tolerant replication is able to work correctly despite arbitrary faults of a subset of its machines.

Discussion at LADiS Atul Singh presented techniques for improving the availability and performance of BFT replication in the presence of network partitions and heterogeneous inter-node latency. To achieve this, instead of aiming for strong semantics that mimic the behavior of a single server, he proposes a specification for a BFT service that offers eventual consistency. He also discussed a protocol that meets this specification and makes progress when only a small subset of the replicas are reachable.

Dan Dobre presented a new replication protocol for BFT storage with a read/write interface that combines strong consistency and liveness properties with optimal latency and space efficiency. These are called amnesic algorithms since values previously stored are not permanently kept, but erased after a number of subsequent write operations.

Allen Clement pointed out that existing BFT replication protocols have optimized for the common case of a fault-free execution (or, at most, one where only crash faults occur). As a consequence, the performance of BFT replication degrades significantly in the presence of Byzantine faults. He discussed a new protocol called Aardvark that achieves better performance than existing protocols under the actual occurrence of arbitrary faults.

Marco Serafini discussed how to lower the number of replicas in BFT replication protocols to reduce the cost of deploying them. While recent proposals offer a tradeoff between performance in the common case where a few replicas might fail (even by crashing) and the number of replicas they use, this research proposes an improvement on this tradeoff by presenting a new protocol that lowers the number of replicas while still providing good performance in the common case.

Rachid Guerraoui presented a new building block for designing BFT replication protocols called Abstract. This proposal consists of an abstraction that can be composed, possibly multiple times, to build BFT protocols. This simplifies the design, implementation, and correctness proof of these complex protocols. They exemplify the utility of Abstract by building two new protocols with better latency and higher throughput, respectively, than existing BFT protocols.

Michael G. Merideth presented a way to use write markers to improve quorum-based BFT tolerant replication. A write

marker identifies the quorum for a write in a way that is computationally hard to forge. By including these in the protocol, non-faulty parties can reject votes from faulty servers that are not in the quorum. This both increases the fraction of the system that can be faulty and decreases the load.

Conclusions Given the societal importance of services running on modern datacenters, it is crucial that these services execute correctly and continuously despite any kind of faults. BFT is a promising technique for achieving this goal. As one of the papers in this workshop pointed out, "the time is now" for BFT systems to be adopted in this context, given that it is becoming advantageous to trade the extra hardware needed by BFT systems for the additional guarantees they provide. But a number of challenges still need to be addressed to increase the chances of adopting these techniques.

- **Are BFT protocols too slow to meet the strict SLAs required by most of the services running in datacenters?** How can we improve their common case performance when a few machines are unreachable? How to guarantee that SLAs are also met when the system is under attack and some machines are misbehaving? How to provide continuous service when a network partition divides the replica set into mutually unreachable groups of replicas?
- **What are the right semantics for BFT services?** Systems like Amazon's Dynamo have shown that performance and availability can matter more than strong semantics, but how can we weaken the semantics enough to provide high availability and good performance and still be able to reason about the services we are building?
- **How to improve the way that we build BFT protocols?** These are very complex protocols and implementations, which have to be correct for the replicated service to provide any guarantees. How can we simplify or improve the process of designing and building BFT protocols?
- **Are there meaningful failure models somewhere between crash and Byzantine failures that are realistic and allow building more efficient protocols?** In a trusted and managed environment like a data center it is unlikely that attackers will gain control over machines, so it seems like BFT techniques might be overkill for dealing with software bugs or hardware faults. Thus developing new failure models that tolerate such faults yet enable the construction of more efficient protocols than BFT replication is an area of future work.
- **What can Byzantine failure detection do to help?** Recent advances in this field have enabled the efficient detection of deviations from the expected behavior after they occurred. More research is needed to understand how to use these techniques in the context of complex systems that run in datacenters.