Research Statement

Hakim Weatherspoon

My research interests are broad, covering various aspects of information, distributed, network, and peer-to-peer systems. In these areas, I particularly focus on fault-tolerance, reliability, security, and performance of Internet-scale systems with decentralized—autonomous, federated, multi-organizational, and cooperative—control. My work generally applies principled approaches to understanding a problem, then validates the understanding gained through analysis by design, implementation, and evaluation of a system [1].

To date, I have applied this technique to several areas in systems research, including distributed wide-area storage [2, 3], Byzantine fault-tolerance [4], efficient replica maintenance and erasure codes [5, 6, 7], structured and unstructured network overlays [8, 9], content distribution networks [10, 11], enterprise-level energy consumption [12] and remote mirroring capabilities [13]. I give a brief overview of these projects below.

1 Contributions

Distributed Wide-Area On-line Archival Storage Systems. Digital information plays an increasingly mission-critical role in military systems and other enterprises, and this trend has important implications. We need data storage systems that can ensure the durability, integrity, and accessibility of digital data, and do so under potentially turbulent conditions. For example, in a large scale system, servers continuously fail all the time; data should remain durable despite constant failure. As a more extreme example, during military engagements, communication links may be disrupted, systems may be destroyed or temporarily incapacitated, and load surges may result in degraded responsiveness; however, the availability of digital data is vital.

Antiquity is a distributed storage system designed for these sorts of challenging environments [2]. It maintains data securely, consistently, and with high availability in a dynamic wide-area environment. At the core of the system is a novel secure log structure that permits Antiquity to guarantee the integrity of stored data, even under extreme stress. Data is replicated on multiple servers in a manner that ensures that data can be retrieved later even when some replicas are inaccessible. Moreover, unlike prior fault-tolerant systems, the Antiquity fault-tolerance protocols can handle high levels of server churn, regenerating data on the fly when necessary to handle faults ranging from server outages to Byzantine (malicious) attacks.

Antiquity's design has been validated via experimental evaluation that combined both global and local testbeds to realistically emulate the kinds of scenarios described above. Antiquity has been running on 400+ PlanetLab [14] servers for over two months storing nearly 20,000 logs totaling more than 84 GB of data. Despite continuous server churn, the logs used to locate and retrieve files remain durable.

Antiquity was developed in the context of OceanStore [3]. In particular, a component of OceanStore was a primary replica implemented as a Byzantine agreement process. This primary replica serialized and cryptographically signed all updates. Given this total order of all updates, the question was how to durably store and maintain the order. Antiquity's implementation of the interface and structure of a secure log assisted in durably maintaining the order over time. When data is later read from Antiquity, the secure log and repair protocols ensure that data will be returned and that returned data is the same as stored.

Efficient Replica Maintenance and Erasure Codes. Wide-area storage systems typically maintain data durability via fault tolerance and repair algorithms. Fault tolerance ensures that data is not lost due to server failure. Replication is the canonical solution for data fault tolerance. The challenge is knowing how many replicas to create and where to store them. Fault tolerance alone, however, is not sufficient to prevent data loss as the last replica will eventually fail. Thus, repair is required to replace replicas lost to failure. The system must monitor and detect server failure and create replicas in response. The problem is that not all server failure results in loss of data and the system can be tricked into creating replicas unnecessarily. The challenge is knowing when to create replicas. Both fault tolerance and repair are required to prevent the last replica from being lost, hence, maintain data durability.

My work, individually and with others, makes several contributions in this space and yields many insights on how to efficiently maintain data durability [1, 5, 6, 7]. First, I have explored the parameterization space

of fault tolerance algorithms and associated durability. This work showed that erasure-coding reduces the bandwidth costs to maintain a target level of durability when compared to replication. Alternatively, for the same storage overhead and bandwidth costs, erasure-coding can maintain a significantly higher level of durability than replication. Next, it showed durability is related to the distribution of failure bursts. Further, it showed that random placement is sufficient to increase durability via reduced repair time and avoiding many correlated failures.

My work showed how to reduce costs (number of replicas created) due to transient failures. It demonstrated a principled way to estimate the amount of extra replication required to reduce repair costs due to transient failures. In particular, when data is immutable (i.e. cannot change), it showed that the system can limit the number of unnecessary copies made due to transient failures by ensuring that recovered copies are integrated in place into the replica set. The result is that the system performs some extra work for each object early in its life, but over the long term creates new copies of the object only as fast as it suffers permanent failures.

Using these insights, we developed the Carbonite replication algorithm for keeping data durable at a low cost [7]. A simulation of Carbonite is able to maintain 100% durability over the course of a year on PlanetLab [14] despite losing over a third of the servers due to permanent failure such as disk failure. Further, in response to transient failures, it creates less than half the number of replicas when compared to previous systems.

Gossip-based Unstructured Overlays. Gossip-based mechanisms are touted for their simplicity, limited resource usage, robustness to failures, and tunable system behavior. Gossip is the periodic pairwise exchange of bounded size messages between random servers in a system, where the state of one (or both) changes to reflect the state of the other [15]. Example environments where gossip-based mechanisms perform well include wide-area networks with limited access link bandwidth, wireless sensor networks with limited battery life and communication range, and mobile ad hoc networks with limited connectivity. I refer to the above environments as *challenged networks*.

I have become increasingly interested in exploiting gossip-based mechanisms in my work: they offer robust solutions to problems that can be hard to solve in other ways, such as providing durable storage in a challenged network. In particular, gossiping allows for proactively creating new replicas at a constant low rate in anticipation of future failures. Moreover, in power-constrained environments such as wireless sensor networks and mobile ad hoc networks, predictable communication patterns, as ensured by gossiping, facilitate energy savings. In my work with gossip, up to the present, I have demonstrated how to tune the rate of gossip to achieve desirable durability levels (rate of data loss) and energy savings [8].

On the other hand, I am cautious not to apply gossip for purposes to which it is ill-suited; it is not a "one-size-fits-all" mechanism. In particular, outside of these challenged networks or others like them, gossip may not be a good mechanism if server failure is a rare occurrence or new replicas can readily be created since resources such as bandwidth or battery life are *not* limited.

Looking towards the near future, I hope to explore applications of gossip to such problems as the convergence between structured and unstructured network overlays [9] and rescue networks (i.e. sensor networks that provide durable storage until rescuers arrive).

Content Distribution Networks. Over the last few years there has been increasing usage of content distribution networks (CDNs) that deliver large volume data objects such as video and software. For example, approximately 70% of Tier 1 ISP traffic measured in Europe was peer-to-peer traffic [16]; yet, the considerable bandwidth consumption is not necessary to satisfy the download demand [17]. The challenge for CDNs like BitTorrent that transfer large objects is to minimize the download time while reducing network bandwidth consumption.

I have contributed to the design of two separate CDN projects. First, ChunkCast [10] is an anycast service that optimizes large content distribution. It uses a distributed locality-aware directory that supports an efficient query for large content. It improves the median downloading time by at least 32% compared to previous approaches and emulates multicast trees without any explicit coordination of peers.

Second, AntFarm [11] distributes content via *managed swarms*. Managed swarms differ from traditional swarming protocols in that they permit a single administrative entity, typically the content owner, to control the behavior of the swarms. This level of control exerted by a *coordinator*, in turn, can enable the swarms to operate much more efficiently than both client-server systems and existing swarming systems. The central problem faced by such a service is how to minimize the time for content distribution, given the distributor's seeding bandwidth, the number of swarms, the number of peers in each swarm, churn, and the peers' upload

and download capacities. Under the assumption that peers are selfish and potentially Byzantine, we first provide a protocol that enables the content distributor to detect misbehaving hosts, gather information on swarm dynamics, and direct the peers' allotment of upload bandwidth. We then derive the optimal strategy by which the content distributor can allocate its own seeding bandwidth and direct the upload bandwidths of peers to achieve the lowest possible delivery times for content. Extensive simulations and a PlanetLab deployment show that the system can significantly outperform BitTorrent.

2 Future Research: Optimizing Enterprise-level Remote Mirroring and Energy Consumption

A global network of datacenters is emerging as an important distributed systems paradigm—commodity clusters running high-performance applications, connected via high-speed, high-capacity, networks across hundreds of milliseconds of network latency [18]. Some key challenges in this environment are the tradeoffs between disaster tolerance and performance and energy consumption of datacenters. However challenging, this paradigm provides for many rich research opportunities into problems that have yet to be solved. Further, my skills and qualification position me as a key player in the area.

First, disaster tolerance and recovery designs confront a conundrum: the tradeoff between either the loss of performance or the potential loss of data. On the one hand, loss of performance such as application response time or throughput may not be an option. For example, it may not be desirable to slow application response time until it is assured that data will not be lost in the event of disaster. Similarly undesirable is the prospect of data loss, which can be catastrophic for many companies and organizations — 70% of small firms that experience a major loss of data go out of business within a year[Source: DTI/Price Waterhouse Cooper, 2004]. Unfortunately, there is not much of a middle ground in the design space and designers must choose one or the other.

I am beginning to explore new ideas that may offer a middle ground [13]. One idea is to *proactively* inject redundancy into the network to prevent loss of transmitted packets, and *expose* the status of out going data, so that the sender can resume activity as soon as a desired level of in-flight redundancy has been achieved for any given packet. For example, an optical link that drops one out of every trillion bits or 125 million 1 kB packets (this is the maximum error threshold beyond which current carrier-grade optical equipment shuts down) can be pushed into losing less than 1 out of every 10^{16} packets by the simple expedient of sending each packet twice — a figure that begins to approach disk reliability. Utilizing the extra bandwidth available to send more redundancy codes can effectively turn the network into a portable disk where data is stored and shipped across to the remote mirror. More importantly, the idea is independent of link latency and can run at very high absolute data rates. This may create a viable new option for developers of mission-critical datacenters on the outer edge of the performance curve.

Second, energy consumption in modern datacenters is an issue receiving increasing attention because of its growing importance — an article in The New York Times describes one Google's datacenters: "... a computing center as big as two football fields, with twin cooling plants protruding four stories into the sky" [19]. At issue is the financial and environmental cost of keeping hundreds of thousands of disks spinning.

One of many possible opportunities for saving energy in large datacenters is within the file system. The idea is elegant in its simplicity: log structured file systems write only to the log head; as a result, if read accesses are served by the cache, then write accesses touch only the log head disk, potentially allowing us to power down all the other disks [12]. Existing solutions like disk management solutions and caching solutions are typically application-specific; our solution, on the other hand, is applicable to any cacheable dataset. Since existing solutions are typically layered on top of the file-system, they could be used in conjunction with our solution to take advantage of application-specific optimizations

Looking further into the future, I am starting work on the conceptual design a new kind of file system that would link datacenters over very long distance optical network links. This raises many technical challenges, some of which are described above. Once finished, the file system will give users a very simple normal view of one file system, but should be able to automatically move data around for performance reasons, and also adapt itself to power and other environmental considerations. It will capitalize on opportunities to save power such as trading off which disks are spinning in which datacenters against the costs of communicating over fast, wide-area links.

References

[1] **H. Weatherspoon**. *Design and Evaluation of Distributed Wide-Area On-line Archival Storage Systems*. PhD thesis, EECS Department, University of California, Berkeley, October 13 2006.

[2] **H. Weatherspoon**, P. Eaton, B. G. Chun, and J. Kubiatowicz. Antiquity: Exploiting a secure log for wide-area distributed storage. In *Proc. of ACM European Conference on Computer Systems (EuroSys)*, March 2007.

[3] S. Rhea, P. Eaton, D. Geels, **H. Weatherspoon**, B. Zhao, and J. Kubiatowicz. Pond: the OceanStore prototype. In *Proc. of USENIX Conference on File and Storage Technologies (FAST)*, 2003.

[4] **H. Weatherspoon**, R. van Renesse, P. Eaton, B. G. Chun, and J. Kubiatowicz. Reactively reconfiguring byzantine fault-tolerant systems. Submitted for publication to IEEE Conference on Dependable Systems and Networks (DSN), December 2007.

[5] **H. Weatherspoon** and J. Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. of International Workshop of Peer-to-Peer Systems (IPTPS)*, March 2002.

[6] **H. Weatherspoon**, B. Chun, C. W. So, and J. Kubiatowicz. Long-term data maintenance in wide-area storage systems: A quantitative approach. Technical Report UCB//CSD-05-1404, U. C. Berkeley, July 2005.

[7] B. Chun, F. Dabek, A. Haeberlen, E. Sit, **H. Weatherspoon**, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Efficient replica maintenance for distributed storage systems. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, May 2006.

[8] **H. Weatherspoon**, H. Miranda, K. Iwanicki, A. Ghodsi, and Y. Busnel. GOSSIP: Gossiping over storage systems is practical. *ACM SIGOPS Operating Systems Review*, 41(5):75–81, October 2007.

[9] A. Ghodsi, S. Haridi, and **H. Weatherspoon**. Exploiting the synergy between gossiping and structured overlays. *ACM SIGOPS Operating Systems Review*, 41(5):61–66, October 2007.

[10] B. Chun, P. Wu, **H. Weatherspoon**, and J. Kubiatowicz. An anycast service for large content distribution. In *Proc. of International Workshop of Peer-to-Peer Systems (IPTPS)*, Santa Barbara, CA, February 2006.

[11] R. Peterson, **H. Weatherspoon**, and E. G. Sirer. Antfarm: A content distribution service based on optimally managed swarms. Submitted for publication, October 2007.

[12] L. Ganesh, **H. Weatherspoon**, M. Balakrishnan, and K. Birman. Optimizing power consumption in large scale storage systems. In *Proc. of USENIX Workshop on Hot Topics in Operating Systems (HotOS)*, May 2007.

[13] **H. Weatherspoon**, L. Ganesh, T. Marian, M. Balakrishnan, and K. Birman. Smoke and mirrors: Shadowing files at a geographically remote location without loss of performance. Submitted for publication to IEEE Conference on Dependable Systems and Networks (DSN), December 2007.

[14] L. Peterson, A. Bavier, E. Fiuczynski, and S. Muir. Experiences building planetlab. In *Proc. of USENIX Symposium on Operating System Design and Implementation (OSDI)*, November 2006.

[15] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 1–12, 1987.

[16] Cachelogic. http://www.cachelogic.com.

[17] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[18] M. Balakrishnan, T. Marian, K. Birman, **H. Weatherspoon**, and E. Vollset. Maelstrom: Transparent error correction for lambda networks. In *Proc. of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, April 2008.

[19] J. Markoff and S. Hansell. Hiding in Plain Sight, Google Seeks More Power. In *The New York Times, Online Ed.*, June 14 2006. `http://www.nytimes.com/2006/06/14/technology/14search.html?ex=1307937600&en=d96a72b3c5f91c47&ei=5090`.