Chapter Eight

Conclusion

This thesis proposed Distributed Streaming Virtual Machine Introspection (DS-VMI), a fundamentally new way of accessing persistent state—agentlessly—without the cooperation of the writer. Although this thread of research was born in cloud computing, it is applicable to any writer sending structured data to persistent storage. For example, a stream of writes to a network attached storage system could be introspected at the level of network packets as a form of deep packet inspection. The goal of this thesis, which guided the design of its interfaces and optimizations, was to find an agentless, application-agnostic, near-real-time view into persistent cloud-wide file-level state. With DS-VMI, we achieved this goal. As a mechanism, DS-VMI is only as useful as the interfaces layered on top of it. We took a holistic approach by designing three interfaces covering the various types of persistent state—live and archival—and the various types of workloads—batch and event-driven.

The first interface, `cloud-inotify` as described in Chapter 3, implements a publish-subscribe, eventually consistent, selective view over persistent state designed for event-driven workloads. Although it requires rewriting legacy applications to its channel-based subscriptions, it is an application and OS agnostic interface. External monitoring agents using `cloud-inotify` benefit by not needing completely separate implementations per monitored writer. Again, this is similar to deep packet inspecting firewalls which are not dependent on the operating system environments of the network hosts they monitor. The second interface, `/cloud` as described in Chapter 4, implements a strongly consistent view over persistent state with a read-only POSIX file system designed for batch workloads. Such a simple interface—a file system—enables compatibility with legacy applications without any extra work.

`cloud-inotify` and `/cloud` both interface with live persistent state. However, significant effort is also spent on archived information kept in the form of backups. Historic data holds answers to many key questions that a business must answer. For example, to fulfill its responsibility to customers, a business needs to sift through historic logs and file-level clues in the event of an information security breach. Tracing the flow of information, systems accessed, and steps taken by attackers is critical information in determining which customers are affected. In order to get a better understanding of the backup solution space, we performed an in-depth research study

of over 30 backup systems in Chapter 7. We found a dearth of backup systems which could support quick random-access querying and indexing with agentless capture of state. Yet trends in archival storage such as Google's Nearline show that the future of backup includes the heretofore unthinkable feature of low-latency, random access to backup data. Services such as Dropbox and Backblaze have begun indexing historic file-level state for consumers for some applications such as fulltext document search, but the vision of this thesis is application agnostic interfaces. Thus, we designed an interface leveraging DS-VMI to provide agentless backup to monitored systems with optimizations maximizing the utilization of storage space, while minimizing the time to index at a file-level.

`/cloud-history` as described in Chapter 5, implements an agentless backup system designed to capture versions of files. File versioning occurs on file-level update streams obtained via DS-VMI with versions derived by a timeout from the last modification. This timeout heuristic tries to model user open-close, and ideally modify-save, semantics. `/cloud-history` assumes fast access to archival storage. In the case of Google Nearline, this is now a publicly available reality. Our study, presented in Section 5.2, of a research backup system confirms that although backup data quickly balloons in scale due to the frequency of scheduled backups, it contains significant amounts of duplicate files. Previous backup studies [71, 123] find significant duplication at a block- or file-level, but report in numbers of bytes saved—not in numbers of duplicate files.

In this dissertation, we took a further step in studying the practical implications of such file-level duplication. We realized, experimentally, that general indexing of backup data at a file-level becomes practical with file-level deduplication. We found indexing every backed up file without file-level deduplication is prohibitively costly. File-level deduplication—an application agnostic optimization—reduces the file space by two orders of magnitude making indexing not only tractable, but reasonable enough to repeat with some regularity. In other words, it should be possible to iteratively and interactively explore backup data even when pre-existing indexes do not exist.

At a high level, this dissertation posed the question: is it possible to completely manage and monitor persistent state agentlessly without any writer support whatsoever? This idea goes against the two traditional models of either a single reader-writer with absolute control over local file systems, or distributed file systems. Although distributed file systems allow for multi-readers and multi-writers, they require the participation, and therefore configuration, of the aforementioned readers and writers. This dissertation developed a core technology abbreviated DS-VMI, which formed the foundation for three interfaces: `cloud-inotify`, `/cloud`, and `/cloud-history`. The unifying theme of finding solutions which are agnostic to the OS or application being monitored was a difficult goal, but realized in DS-VMI. This dissertation also contributes a prototype implementation including integration into real world cloud software called OpenStack. The answer developed in this dissertation leads to more questions.

What will the clouds of tomorrow look like? They will be more managed then today, deploying technologies similar to DS-VMI. By observing the DevOps movement, we see the immediate immense popularity of agentless tooling such as Ansible. This is because agentless solutions require less configuration of monitored systems, have no moving—breakable—parts within the monitored

system, and are impossible to turn off or corrupt by the entity being monitored. It is difficult to argue against the value proposition of agentless technologies, except for their overheads. Could DS-VMI be made more efficient? We believe that it can, and it is only a matter of investing engineering effort. DS-VMI is efficient for most workloads except high bandwidth writes over extended time periods. To address this overhead, one low-hanging engineering fruit is implementing a zero-copy optimization between DS-VMI and the monitored writers.

In conclusion, the future looks bright for agentless monitoring and management tooling. The agentless value proposition is constructed from foundational guarantees that are unobtainable with agents. Put bluntly, agentless technologies are technically superior to agent-based monitoring. For example, we can guarantee zero impact on quality of service via agentless solutions, which is impossible to guarantee with agent-based solutions. Even if you took drastic measures, such as turning off the monitoring agent, agents could disobey, or may have already introduced a malicious entity into the system. DS-VMI demonstrates a path towards agentless monitoring of persistent storage across the cloud landscape. It is the first description of a unifying framework for addressing persistent storage at a file-level. DS-VMI embraces the now prevalent paradigm of software defined storage. By embracing modern storage's reality rather than clinging to an anachronistic, monolithic system design, this dissertation enables a new generation of outsourced monitoring applications.