Analyzing Persistent State Interactions to Improve State Management

Chad Verbowski, Emre Kıcıman, Brad Daniels, Shan Lu, Roussi Roussev, Yi-Min Wang, Juhan Lee

Categories and Subject Descriptors

D.4.7 **[Software]**: Operating Systems
H.4.m **[Information Systems Applications]**: Miscellaneous;

General Terms

Experimentation, Management, Measurement

Keywords

System Management, Persistent State, Registry, File, Trace, State Management

1. Introduction

A primary challenge to building reliable and secure computer systems is managing the persistent state (PS) of the system: all the executable files, configuration settings and other data that govern how a system functions. The difficulty comes from the sheer volume of this persistent state, the frequency of changes to it, and the variety of workloads and requirements that require customization of persistent state. The cost of not managing a system's persistent state effectively is high: configuration errors are the leading cause of downtime at Internet services, troubleshooting configuration problems is a leading component of total cost of ownership in corporate environments, and malware – effectively, unwanted persistent state – is a serious privacy and security concern on personal computers [1, 6,8,14].

The first step to building better PS management tools is gaining a better understanding and characterization of how computer systems interact with their PS – how and when this state is created, read, written and deleted by the programs and users of a computer system. To this end, we collected over 3648 machine days of these *PS interactions* over an 8 month period from March to November, 2005. We monitored the PS interactions of 193 machines operating under real workloads in a variety of environments, including Internet services, corporate desktops, experimental lab machines and home machines.

There have been many studies of file system workload traces with the goal of improving I/O system performance by optimizing disk layout, replication, etc. [2,3,4,5,7,9,10,11,13]. To our knowledge, we are the first to study file system accesses and registry accesses with the goal of characterizing and improving the management of PS.

2. Trace Collection

In this section, we describe our instrumentation package for monitoring and collecting PS interactions. We also describe the machines and environments from which we collected our traces. We use *PS* to refer to both the file system and the Windows Registry. PS *entries* refer to files and folders as well as their registry equivalents. A PS *interaction* is any kind of access, such as a read or write, to an entry.

Table 1 Summary information about our collected traces

Environment	Number of Machines	Total Observed Machine-Days
Internet service machines	76	841
Research lab machines	72	1703
Corporate desktops	35	849
Home machines	7	169
Idle machines	3	86
Total	**193**	**3648**

To collect our traces of PS interactions, we built a black box instrumentation tool for the Windows operating system. It consists of (1) a kernel mode driver that intercepts all PS interactions with the file system and the Windows Registry, along with process creation and binary load activity; and (2) a user mode daemon that manages the trace files and uploads them to a central server. Neither the kernel mode driver nor the user mode daemon requires any changes to the core operating system or the applications running atop it.

Table 1 summarizes the traces we collected from machines across several environments. Our deployment of the data collector was gradual, so many machines were not monitored for the full 8 month period. The largest collections of traces come from research lab machines managed by our labs' own IT staff; and five different services at MSN. Depending on the individual service, the Internet service machines were subject to different workloads and management styles, including one service administered under a "follow the sun model," with 3 operations teams around the world monitoring the system. Our other traces were captured from corporate desktops and laptops, home machines and, as a control, idle systems running within virtual machines. Together, these environments represent a broad sample of current systems management styles, from the "intensely managed" Internet services to unmanaged home machines.

3. Some Survey Highlights

One of the major challenges to effectively managing persistent is the sheer volume of state and how frequently it changes. Our experiments agree with previous studies of file system contents [3,9,12] and the Windows registry [13] which find that modern computer systems contain on average 70k files and approximately 200k registry settings. In the rest of this short paper, we highlight our measurements of the volume of persistent state interactions.

Table 2 presents the average volume of daily interactions, divided by the category of application generating the interaction and type of activity: (1) an OS-level process, or a surrogate process such as

Table 2 Summary of daily PS activity per machine across environments. All units are in millions.

Environment	Category			Type		Total
	Workload	Mgmt.	OS	Read	Write	
Svc. 1	39.75	26.52	3.59	68.31	1.55	**69.86**
Svc. 4	19.16	37.10	4.37	57.77	2.86	**60.63**
Svc. 5	2.29	23.10	3.67	28.31	0.76	**29.07**
Svc. 2	0.005	21.00	1.45	21.26	1.20	**22.46**
Svc. 3	1.63	14.93	2.18	16.90	1.83	**18.73**
Home	4.27	8.89	4.17	16.70	0.62	**17.33**
Desktop	2.74	5.02	1.62	8.94	0.44	**9.38**
Lab	2.52	5.99	0.74	8.73	0.51	**9.25**
Idle	0.005	0.25	0.10	0.34	0.02	**0.36**

cmd.exe which primarily spawns other processes; (2) a state management tool, such as an installer, configuration panel or antivirus program; (3) a workload application such as a word processor on a home machine or a web service on a server. One surprising result is that existing management applications, such as hardware configuration tools and antivirus scanners, generate 38%-98% of the interactions across our environments. Overall, we see that server machines generate considerably more interactions than desktop, home, and lab machines.

While millions of interactions occur every day on a typical machine, there are several factors that we can use to easily reduce the volume of events and state that we care about, while still improving PS management. First, we see that read activity is consistently an order of magnitude (or more) greater than write activity across all environments, consistent with prior findings [2]. While read interactions are important when trying to debug or explain software behavior, we do not have to worry about these interactions corrupting PS.

Furthermore, we find that the number of distinct non-temporary files and registry entries read or written every day is much smaller than the total number of interactions ð up to 2 orders of magnitude smaller. However, this is still a large number of entries: 10-15% of the 70k files and 5-10% of the 200k registry entries on a system are used on any given day. Although server machines have more total interactions per day, our traces show that they use an order of magnitude fewer distinct PS entries compared to the desktop and home machines.

Finally, we find that while there are tens of millions of daily accesses to files and registry settings on both server and desktop systems, these file system accesses show a large degree of structure and repetition, in the form of activity bursts. Recognizing this structure enables the volume of events to be reduced by several orders of magnitude from $O(10^7)$ daily events to $O(10^3)$ distinct daily activity bursts. We believe this reduction makes the on-line analysis of PS interactions feasible as a building block to improve systems management practice.

Our full-length survey paper [11] presents our analysis of activity bursts in more detail, as well as delving into many other issues not mentioned here, such as analyses of how processes use the state on a system and the implications for PS management; and how frequently software is installed and upgraded, and by whom (e.g., auto-update software, remote administrators, etc). We also present case studies of how monitoring PS interactions can help address current persistent state management problems. One case study shows how PS interaction monitoring can expose security-sensitive configuration settings that might allow unwanted software (malware) to attach themselves as plug-ins to a system. Not only can PS interaction monitoring discover these critical configuration settings, but also quantify their importance in terms of the privileges exposed and the duration of the system exposure.

4. REFERENCES

[1] W. Arbaugh, W. Fithen, and J. McHugh, ñWindows of Vulnerability: A Case Study Analysis,ò In *IEEE Computer*, Vol. 33, No. 12, Dec. 2000.

[2] M. Baker, J. Hartman, M. Kupfer, K. Shirriff, and J. Ousterhout, ñMeasurements of a Distributed File System,ò In *Proc. Of SOSP* (1991).

[3] J. Douceur and B. Bolosky, ñA Large-Scale Study of File-System Contents,ò in *Proc. SIGMETRICS* (1999).

[4] D. Ellard, J. Ledlie, P. Malkani, and M. Seltzer, ñPassive NFS Tracing of Email and Research Workloads,ò in *Proc. of FAST* (2003).

[5] W. H. Hau and A. J. Smith, ñCharacteristics of I/O Traffic in personal computer and server workloads,ò in *IBM Systems Journal*, Vol. 42, No. 2, pp. 347 ï372, 2003.

[6] D. Oppenheimer, A. Ganapathi, and D. Patterson, ñWhy do Internet services fail, and what can be done about it? ò in *Proc. of USITS* (2003).

[7] K. K. Ramakrishnan, P. Biswas and R. Karedla, ñAnalysis of File I/O Traces in Commercial Computing Environments ,ò in *Proc. of SIGMETRICS* (1992).

[8] E. Rescorla, ñSecurity Holesé Who Cares?ò in *Proc. of the 12th USENIX Security Symp.*, Aug. 2003.

[9] D. Roselli, J. Lorch, and T. Anderson, ñA Comparison of File System Workloads,ò in *Proc. of USENIX* (2000).

[10] C. Ruemmler and J. Wilkes, ñUNIX Disk Access Patterns,ò in *Proc. of USENIX* (1993).

[11] C. Verbowski, E. Kiciman, B. Daniels, S. Lu, R. Roussev, Y.-M. Wang, ñAnalyzing Persistent State Interactions to Improve State Management,ò Tech. Rep. MSR-TR-2006-39, Microsoft Research, Redmond, WA (April 2006).

[12] W. Vogels, ñFile system usage in Windows NT 4.0,ò in *Proc. of SOSP* (1999).

[13] Y.-M. Wang, C. Verbowski, J. Dunagan, Y. Chen, H. Wang, C. Yuan, and Z. Zhang, ñSTRIDER: A Black-box, State-based Approach to Change and Configuration Management and Support,ò in *Proc. of LISA* (2003).