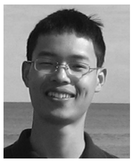
WEIHANG JIANG, CHONGFENG HU, SHANKAR PASUPATHY, ARKADY KANEVSKY, ZHENMIN LI, AND YUANYUAN ZHOU

storage system problem troubleshooting and system logs

Weihang Jiang is a senior research engineer at Pattern Insight. He received a Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign.

Weihang.jiang@gmail.com


Chongfeng Hu did his graduate study at the University of Illinois at Urbana-Champaign. He is interested in operating systems and software reliability. He is now a software engineer at Microsoft.

chu7@cs.uiuc.edu

Shankar is a member of NetApp's advanced technology group, where he leads the indexing project. He is involved in research related to indexing petabyte-scale storage, as well as building large content repositories.

Shankar.Pasupathy@netapp.org

Arkady Kanevsky is a senior research engineer at NetApp Advanced Technology Group. Arkady has done extensive research on RDMA technology, storage resiliency, scalable storage systems, and parallel and distributed computing. He received a Ph.D. in computer science from the University of Illinois in 1987. He was a faculty member at Dartmouth College and Texas A&M University prior to joining the industry world. Arkady has written or co-authored over 60 publications and is a chair of DAT Collaborative and MPI-RT standards.

arkady@netapp.com

Yuanyuan Zhou is an associate professor at the University of Illinois, Urbana-Champaign. Her research spans the areas of operating system, storage systems, software reliability, and power management.

yyzhou@uiuc.edu

CUSTOMER PROBLEM TROUBLEshooting has been a critically important issue for both customers and system providers. A recent study indicates that problem-diagnosis-related activity is 36–43% of TCO (total cost of ownership) in terms of support costs [4]. Additionally, downtime can cost a customer 18–35% of TCO [4]. The system vendor pays a price as well. A survey showed that vendors devote more than 8% of total revenue and 15% of total employee costs on technical support for customers [10]. In this article, we explain how our FAST '09 paper made two major contributions to better understanding how logs pertain to solving problems.

We provided a characteristic study of customer problem troubleshooting using a large set (636,108) of real-world customer cases reported from 100,000 commercially deployed storage systems in the last two years. We studied the characteristics of customer problem troubleshooting from various dimensions as well as correlation among them. Our results show that while some failures are either benign or resolved automatically, many others can take hours or days of manual diagnosis to fix. For modern storage systems, hardware failures and misconfigurations dominate customer cases, but software failures take a longer time to resolve. Interestingly, a relatively significant percentage of cases occur because customers lack sufficient knowledge about the system. We observed that customer problem reports with attached system logs are invariably resolved much faster than those without logs.

We also evaluated the potential of using storage system logs to resolve these problems. Our analysis shows that a failure message alone is a poor indicator of root cause, and that combining failure messages with multiple log events can improve low-level root cause prediction by a factor of three. We then discuss the challenges in log analysis and possible solutions.

Data Selection

We used two primary databases in selecting customer case data for analysis in our research: a *Customer Support Database*, which contains details on every customer case that was human-generated or auto-generated, and an *Engineering Case Database* for problems that cannot be resolved by customer support staff, which are escalated to engineering teams.

We analyzed 636,108 NetApp customer cases from the Customer Support Database over the period 1/1/2006 to 1/1/2008. Of these, 329,484 were human-generated and 306,624 were auto-generated. Overall, these represent about 100,000 storage systems.

For each of these 636,108 customer cases, *problem category* and *resolution time* were retrieved from the Customer Support Database. For each of the 306,624 auto-generated customer cases, we also retrieved the critical event that led to the creation of the case. However, the human-generated cases do not include such information.

The goal for resolving any customer case is to determine the problem root case as soon as possible. Since such information in the Customer Support Database is unstructured, it was difficult to identify problem root cause for solved cases. However, the Engineering Case Database includes the problem root cause at a fine level. We used 4,769 such cases that were present in both the Customer Support and Engineering Case databases to analyze problem root cause and its correlation with critical events.

To study the correlation between problem root cause and storage system logs, we retrieved the AutoSupport logs from the NetApp AutoSupport Database. Since not all customer systems send AutoSupport logs to NetApp, 4,535 out of 4,769 customer cases have corresponding AutoSupport log information.

Problem Resolution

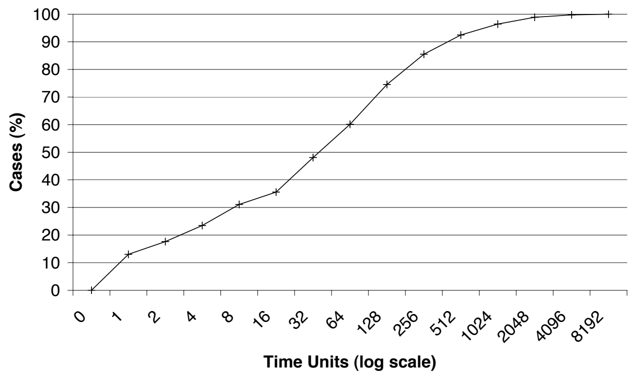


FIGURE 1: CUMULATIVE DISTRIBUTION FUNCTION (CDF) OF RESOLUTION TIME FOR ALL CUSTOMER CASES. WE ANONYMIZE RESULTS TO PRESERVE CONFIDENTIALITY.

One of the most important metrics of customer support is problem resolution time, the time between when a case is opened and when the resolution or a workaround is available for a customer. The distribution of problem resolution times is the key to understanding the complexity of a specific problem or problem class, since it mostly reflects the amount of time spent on troubleshooting problems. This time should not be directly used to calculate MTTR (Mean Time To Recovery), since it does not capture the amount of time to completely solve the problems (e.g., for hardware-related problems, it does not include hardware replacement.)

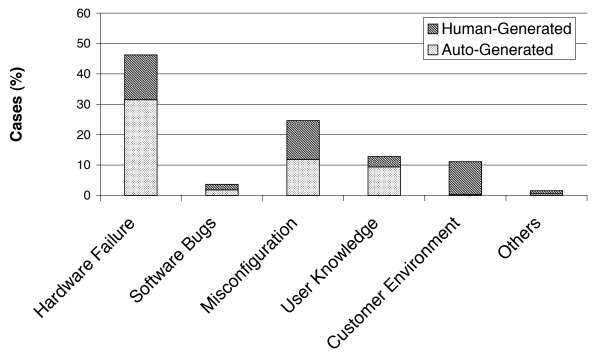
Figure 1 shows the Cumulative Distribution Function (CDF) of resolution time for all customer cases selected from the Customer Support Database. Troubleshooting can take many hours. For a small fraction of cases, resolution time can be even longer. Since the x-axis of the figure is logarithmic, the graph shows that doubling the amount of time spent on problem resolution does not double the number of cases resolved. While the AutoSupport logging system is an important step in helping troubleshoot problems, this figure makes the case that better tools and techniques are needed to reduce problem resolution time.

PROBLEM ROOT CAUSE CATEGORIES

Analyzing the distribution of problem root causes is useful in understanding where one should spend effort when troubleshooting customer cases or designing more robust systems. While a problem root cause is precise, such as a SCSI bus failure, in this section we lump root causes into categories such as hardware, software, misconfiguration, etc. For all the customer cases, we study resolution time for each category, relative frequency of cases in each category, and the cost, which is the average resolution time multiplied by the number of cases for that category.

(a) Categorization of Problem Root Causes

(b) Average Resolution Time per Problem Root Cause Category



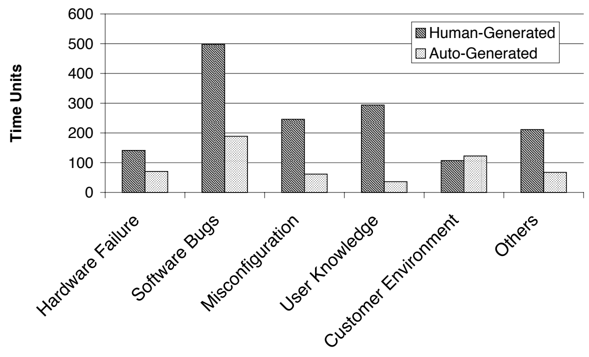


FIGURE 2: PROBLEM ROOT CAUSE CATEGORIES AND TIME TO RESOLUTION.

In Figure 2, *Hardware Failure* is related to problems with hardware components, such as disk drives or cables. *Software Bug* is related to storage system software, and *Misconfiguration* to system problems caused by errors in configuration. *User Knowledge* concerns technical questions, e.g., explaining why customers were seeing certain system behaviors. *Customer Environment-* involves problems not caused by the storage system itself. Figure 2(a) shows hardware failure and misconfiguration are the two most frequent problem root cause categories, contributing respectively, 47% and 25% to all customer cases,. Software bugs account for a small fraction (3%) of cases. We speculate that software bugs are not that common since software undergoes rigorous testing before being shipped to customers. Besides tests, there are many techniques [2, 6, 7, 8] that can be applied to find bugs in software. Figure 2(b) shows that software bugs take a longer time to resolve on average, but since their number is so small, their overall impact on total time spent on all problem resolutions is not very high, as Figure 3 demonstrates.

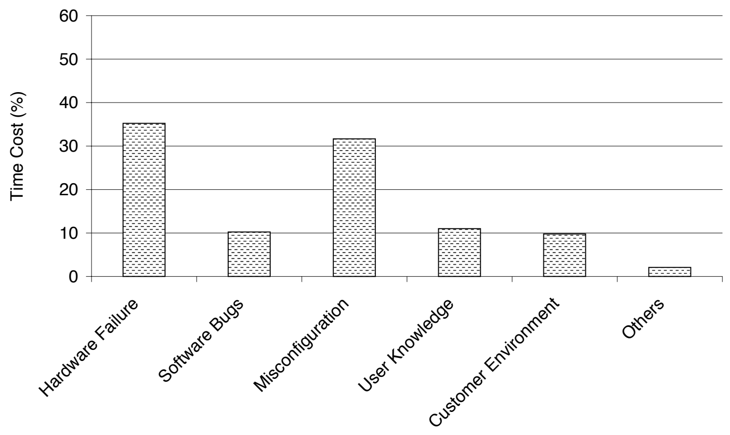


FIGURE 3: RESOLUTION TIME SPENT ON PROBLEM ROOT CAUSE BY CATEGORY. ALTHOUGH SOFTWARE PROBLEMS TAKE LONGER TO RESOLVE ON AVERAGE, HARDWARE FAILURE AND MISCONFIGURATION-RELATED PROBLEMS HAVE GREATER IMPACT ON CUSTOMER EXPERIENCE.

It is interesting to observe that a relatively significant percentage of customer problems are because customers lack sufficient knowledge about the system (11%) or customers' own execution environments are incorrect (9%) (e.g., a backup failure caused by a Domain Name System error). These problems can potentially be reduced by providing more system-training programs or better configuration checkers.

Figure 2(b) is our first indication that logs are indeed useful in reducing problem resolution time. Auto-generated customer cases i.e., those with an attached system log and problem symptom in the form of a critical event message, take less time to resolve than human-generated cases. The latter are often poorly defined over the phone or by email. The only instance where this is not true is when the problem relates to the customer's environment, which is difficult to record via an automated system.

PROBLEM IMPACT

In the previous subsections, we have treated all problems as equal in their impact on customers. We now consider customer impact for each problem category. To do this, we divide customer cases into six categories based on impacts ranging from *system crash,* which is the most serious, to the low-impact *unhealthy* status. "System crash" here means crash of a single system, which might not lead to service downtime with a cluster configuration. The other categories, from higher to lower impact, are *usability* (e.g., inability to access a volume), *performance*, *hardware component failure*, and *unhealthy status* (e.g., instability of the interconnects, low spare disk count). Hardware failures typically have low impact since the storage systems are designed to tolerate multiple disk failures [3], power-supply failures, filer-head failures, etc. However, until the failed component is replaced, the system operates in degraded mode where the potential for complete system failure exists, should its redundant component fail.

Since human-generated customer cases do not have all impact information in structured format, we randomly sampled 200 human-generated cases and manually analyzed them. For auto-generated problems, we include all the cases, and leverage the information in Customer Support Database.

For both human-generated and auto-generated cases, the classification is exclusive: each problem case is classified to one and only one category. The classification is based on how a problem impacts customers' experience. For example, a disk failure that led to a system panic will be classified as an instance of *system crash*. If it did not lead to system crash (i.e., RAID handled it), it is classified as an instance of *hardware component failure*. It is important to notice that, in our study, the *performance* problems are problem cases that lead to unexpected performance slow down. Therefore disk failures leading to expected slow down with RAID reconstruction processes are classified as*hardware component failures* instead of *performance* problems.

(a) Distribution of Problems with Different Impact

(b) Average Resolution Time of Problem with Different Impact

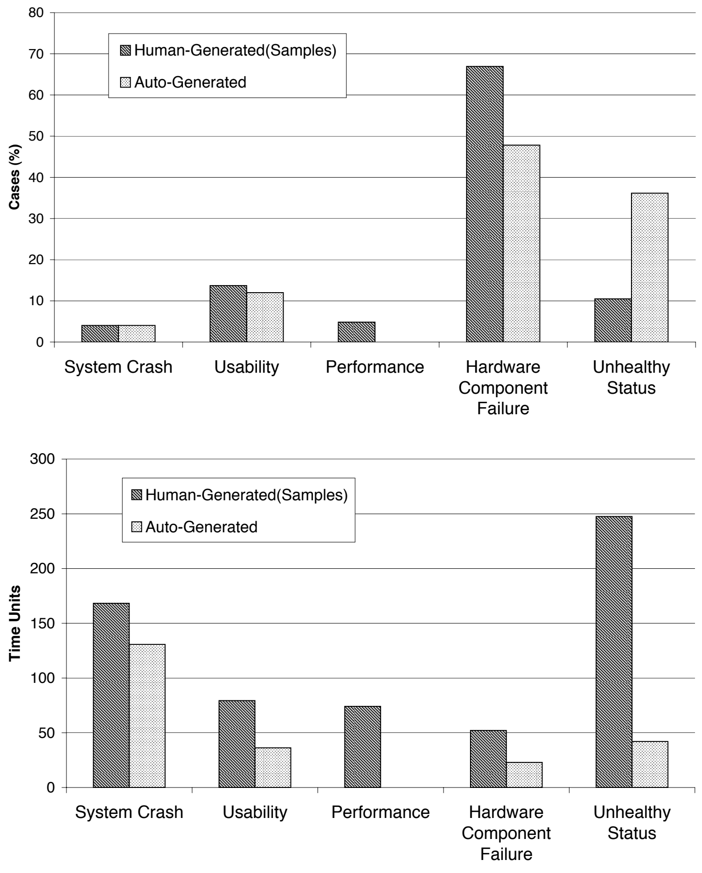


FIGURE 4: PROBLEM IMPACT.

Figure 4(a) shows the distribution of problems by impact. One obvious observation is that there are far fewer high-impact problems than low-impact ones. More specifically, *system crash* only contributes about 3%, and *usability* problems contribute about 10%. Low-impact problems such as *hardware component failure* and *unhealthy status* contribute about 44% and 20%, respectively.

While high-impact problems are much fewer, as Figure 4(b) shows, they are more time-consuming to troubleshoot. This is due to the complex interaction between system modules.

Finally, as we observed in the previous section, auto-generated cases take less time to resolve than human-generated ones.

Feasibility of Using Logs for Automating Troubleshooting

We investigated the feasibility of using additional information from system logs and answered the following two questions: does problem root cause determination improve by considering log events beyond critical events? what kind of log events are key to identifying the problem root cause?

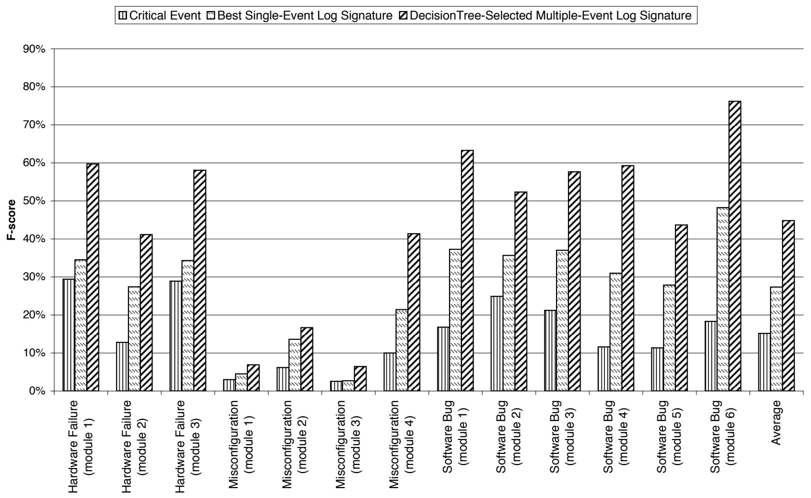


FIGURE 5: COMPARISON BETWEEN THREE METHODS OF USING LOG EVENTS. F-SCORE INDICATES HOW ACCURATE A PREDICTION CAN BE MADE ON A MODULE-LEVEL PROBLEM ROOT CAUSE USING LOG INFORMATION.

ARE ADDITIONAL LOG EVENTS USEFUL?

To study whether additional log events are useful, we considered three methods of using log event information, and compared how well they can be used as a module-level problem root cause signature. We defined a signature as a set of relevant log events that uniquely identify a problem root cause. Such a signature can be used to identify recurring problems and to distinguish one problem from another unrelated one, thereby helping with customer troubleshooting. It is important to note that we are not designing algorithms to find log signatures; instead, we are manually computing log signatures to study how they improve problem root cause determination.

As a baseline, our first method is to only use the problem's critical event as its signature. The second method is similar to method one, but instead of just looking at critical events to deduce a root cause signature, we search all log events looking for the one log message that best indicated the module-level root cause. The third method is to use a decision tree [1] to find the best mapping between multiple log events and the problem root cause. The resulting multiple log events can be used as the root cause signature. For each module-level problem root cause, *F-score* is used to measure how well the signature can predict the problem root cause [9]. For more details about the methodology, refer to our conference paper [5].

As Figure 5 shows, for all customer cases, using only critical events as the problem signature is a very poor predictor of root cause. On average, it only achieves an *F-score* of about 15%. Using the best-matched log event, instead of just critical events, can achieve an *F-score* of 27%. By comparison, the average *F-score* achieved by the decision tree method for computing problem signatures is 45%, 3x better than using critical events. Based on these results, we conclude that accurate problem root cause determination requires combining multiple log events rather than a single log event or critical event.

This observation matters, since customer support personnel usually focus on the critical event, which can be misleading. Furthermore, as we show in the next section, there is often a lot of noise between key log events, making it hard to manually detect problem signatures.

Although we use the decision tree to construct log signatures that are composed of multiple log events, we do not advocate this technique as the solution for utilizing log information. First of all, the accuracy (F-score) is still not satisfactory due to log noise, which we discuss later. Moreover, the effectiveness of the decision tree relies on training data. For problem root causes that do not have a large number of diagnosed instances, a decision tree will not provide much help.

CHALLENGES OF USING LOG INFORMATION

To understand the challenges of using log information and identifying key log events to compute a problem signature, we manually analyzed 35 customer cases sampled from the Engineering Case Database. These customer cases were categorized into 10 groups, such that cases in each group had the same problem root cause.

For these customer cases, we noticed that engineers used several key log events to diagnose the root cause. Table 1 summarizes these cases and characteristics of their key log events.

Symptom	Cause	# of Key Log Events	Distance (secs)	Distance (# events)	Fuzziness?
Battery Low	Software Bug	2	5.8	1.6	no
Shelf Fault	Shelf Intraconnect Defect	3	49.4	3.8	yes
System Panic	Broken SCSI Bus Bridge	4	509.2	34.4	no
Performance Degradation	FC Loop Defect	2	3652	69.4	no
Power Warning	Incorrect Threshold in Code	2	5	2.4	yes
RAID Volume Failure	Software Bug	3	196	66.5	no
RAID Volume Failure	Non-zeroed Disk Insertion	3	80	35	yes
RAID Volume Failure	Interconnect Failure	3	290.5	126	yes
Shelf Fault	Shelf Module Firmware Bug	4	18285.5	21.5	no
Shelf Fault	Power Supply Failure	3	31.5	3.5	no

TABLE 1: CHARACTERISTICS OF LOG SIGNATURES. WE MANUALLY STUDIED 35 CUSTOMER CASES AND PLACED THEM INTO 10 GROUPS, WHERE THE CASES IN EACH GROUP HAD THE SAME PROBLEM ROOT CAUSE. BASED ON DIAGNOSIS NOTES FROM ENGINEERS, WE WERE ABLE TO IDENTIFY THE KEY LOG EVENTS, WHICH CAN DIFFERENTIATE CASES IN ONE GROUP FROM CASES IN ANOTHER. "# OF KEY LOG EVENTS" IS THE TOTAL NUMBER OF IMPORTANT LOG EVENTS (INCLUDING CRITICAL EVENTS) NEEDED TO IDENTIFY THE PROBLEM. "DISTANCE" IS CALCULATED AS THE LONGEST DISTANCE FROM A KEY LOG EVENT TO A CRITICAL EVENT FOR EACH CUSTOMER CASE, AVERAGED ACROSS ALL CASES.

Based on these 10 groups, we made the following major observations:

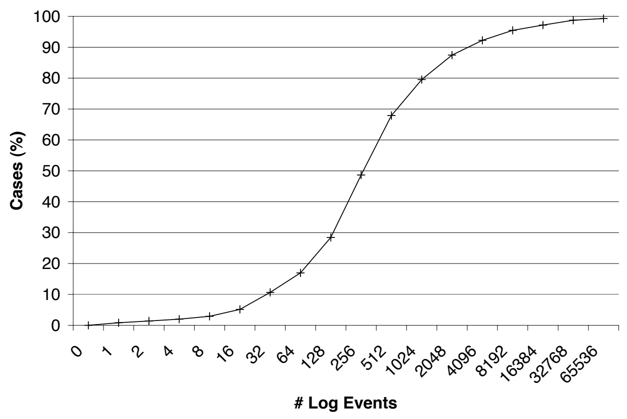


FIGURE 6: CUMULATIVE DISTRIBUTION FUNCTION (CDF) OF NUMBER OF LOG EVENTS WITHIN ONE HOUR OF CRITICAL EVENT. FOR THIS FIGURE, WE USE THE SAME DATA SET AS FIGURE 5. WE ONLY COUNT THE LOG EVENTS GENERATED AND RECORDED BY THE AUTOSUPPORT SYSTEM WITHIN ONE HOUR BEFORE THE CRITICAL EVENT, SINCE ENGINEERS OFTEN ONLY EXAMINE RECENT LOG EVENTS FOR PROBLEM DIAGNOSIS.

(1) Logs are noisy.

Figure 6 shows the Cumulative Distribution Function (CDF) of the number of log events in AutoSupport logs corresponding to customer cases. As can be seen in the figure, for a majority of the customer cases (75%), there are more than 100 log events recorded within an hour before the critical event occurred, and for the top 20% customer cases, more than 1000 log events were recorded.

In comparison, as Table 1 shows, there are usually only 2–4 key log events for a given problem, implying that most log events are just noise for the problem.

(2) Important log events are not easy to locate.

Table 1 shows the distance between key log events and critical events, both in terms of time and the number of log events. For 5 out of 10 problems, at least one key log event is more than 30 log events away from the critical event, which captures the failure point. For all problems, there are always some irrelevant log events in between the key log events and the critical event. In terms of time, the key log events can be minutes or even hours before the critical event.

(3) The pattern of key log events can be fuzzy.

Sometimes, it is not necessary to have an exact set of key log events to identify a particular problem. For example, in Table 1's problem 7, it is not necessary to see the "raidDiskInsert" log event, depending on how the system administrator added the disk drive. In problem 2 the same shelf intraconnect error can be detected by different modules, and different log messages can be generated for it depending on which module reports the issue.

Conclusion

In this article, we present a study of the characteristics of customer problem troubleshooting from logs, using a large set of customer support cases from NetApp. Our results show that customer problem troubleshooting is a very time-consuming and challenging task that can benefit from automation to speed up resolution time. We observed that customer problems with attached logs were invariably resolved sooner than those without logs. We show that while a single or critical log event is a poor predictor of problem root cause, combining multiple key log events leads to a threefold improvement in root-cause determination. Our results also show that logs are challenging to analyze manually because they are noisy and that key log events are often separated by hundreds of unrelated log messages. Please refer to our conference paper [5] for our ideas for an automatic log analysis tool that can speed up problem resolution time.

As with similar studies, it was impossible to study a handful of different data sets, especially for customer support problems, due to the unavailability of such data sets. Even though our data set (which is already very large with 636,108 cases from 100,000 systems) is limited to NetApp, we believe that this study is an important first-step in quantifying both the usefulness of and challenge in using logs for customer problem troubleshooting. We hope that our study can inspire and motivate characteristic studies about other kinds of systems as well, and motivate the creation of new tools for automated log analysis for customer problem troubleshooting.

REFERENCES

[1] Leo Breiman, J.H. Friedman, R.A. Olshen, and C. . Stone, *Classification and Regression Trees*, Statistics/Probability Series (Belmont, California: Wadsworth Publishing Company, 1984).

[2] Ben Chelf and Andy Chou, “The Next Generation of Static Analysis: Boolean Satisfiability and Path Simulation—A Perfect Match,” *Coverity White Paper*, 2007.

[3] Peter Corbett, Bob English, Atul Goel, Tomislav Grcanac, Steven Kleiman, James Leong, and Sunitha Sankar, “Row-Diagonal Parity for Double Disk Failure Correction,” *Proceedings of the 3rd USENIX Conference on File and Storage Technologies (FAST)*, USENIX Association, 2004, pp. 1–14.

[4] Crimson Consulting Group, “The Solaris 10 Advantage: Understanding the Real Cost of Ownership of Read Hat Enterprise Linux,” *Crimson Consulting Group Business White Paper*, 2007.

[5] Weihang Jiang, Chongfeng Hu, Shankar Pasupathy, Arkady Kanevsky, Zhenmin Li, and Yuanyuan Zhou, “Understanding Customer Problem Troubleshooting From Storage System Logs,” *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST ’09)*, USENIX Association, 2009.

[6] S. Johnson, “Lint, a C Program Checker,” Computer Science Technical Report 65, Bell Laboratories, December 1977.

[7] Zhenmin Li, Shan Lu, Suvda Myagmar, and Yuanyuan Zhou, “Cp-Miner: A Tool for Finding Copy-Paste and Related Bugs in Operating System Code,” *Proceedings of the 6th USENIX Conference on Symposium on Operating Systems Design and Implementation (OSDI)*, USENIX Association, 2004.

[8] Zhenmin Li and Yuanyuan Zhou. “Pr-Miner: Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code,” *ESEC/FSE-13: Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2005, pp. 306-315.

[9] C.J. Van Rijsbergen, *Information Retrieval* (Newton, MA: Butterworth-Heinemann), 1979.

[10] Association of Support Professionals, “Technical Support Cost Ratios,” 2000.