

Warnings and Errors: A Measurement Study of a UNIX Server

Eric Daniel, Ronjeet Lal and Gwan Choi

Department of Electrical Engineering, Texas A&M University
{edaniel, ronjeet, gchoi}@ee.tamu.edu

1. Introduction

Measurement-based system dependability analysis can provide important insight into the failure mechanisms exhibited by operational systems. Error and failure logs can be studied to determine events that trigger system malfunctions and/or generate general statistics that relate workloads, errors and failures. It is also possible to identify reliability bottlenecks in both hardware and software through measurement.

This study presents the dependability evaluation of a UNIX server system using message logs with an emphasis on warning messages. Warnings may be precursors to errors and, ultimately, failures. However, warnings are too often disregarded as non-reliability-critical information. The goal of this study is to investigate warning and error events that precede fatal errors and failures. The measurement logs are used in the following analysis: 1) categorize and classify the error and failure data gathered from the system; 2) cluster the categorized and classified data points for analysis; and 3) generate warning-error graphs for analysis.

2. Background

Previous measurement studies have addressed issues such as failure and workload dependency [1], error dependency among different system components [2], failure analysis [3], [4] and failure prediction [5]. Of these studies, [4] and [5] are conducted in UNIX environment. The measurement reported in [4] is from a Tandem system that contains a suite of hardware fault-tolerant features and the study reported in [5] modifies the device driver and error handling software in the UNIX kernel. These studies required several modifications to the kernel which may impede further software upgrades or make the operating system less robust. This paper presents a measurement-based approach which uses facilities commonly found in UNIX allowing dependability evaluations of UNIX machines where the kernel of the operating system is not significantly altered. No measurement-based studies have been conducted in the past that targets off-the-shelf UNIX operating systems.

Table 1

Classification and categorization of eesun2 event logs

Classification	Category	Number	Percent
Warnings		3068235	100.00
Errors	Hardware	34	0.09
	Disk	6	0.02
	Tape	2680	7.92
	I/O Network	7733	22.86
	Unknown	23401	69.19
	Total I/O	33820	93.10
	Software	29	0.08
	Maintenance	107	0.29
	Unknown	2338	6.43
	Total	36328	100.00
Failures	Halt/Reboot	22	91.7
	Unknown	2	8.33
	Total	24	100.00

3. Collection and Categorization of Log Data

The measurement data are collected from a UNIX server system at Texas A&M University: the general-access server *eesun2*. The server hosts over fifty NFS mounted systems, and provides on-line services to approximately 800 faculty, staff and students. The machine is operational 24 hours a day and 365 days a year. The workload of the system is diverse and ranges from general-purpose data handling to scientific computing.

The data used in our analysis consists of *syslog* messages collected over several months. The *syslog* messages may not contain all the system activities attributed to all active programs, but these do contain messages from the kernel which is the most important information on system-wide errors and failures. During the course of this study, about 3 million messages were collected, representing 400 megabytes of data. They covered the periods from June to October 1996 and April to July 1997.

Before any classification takes place, a filter that discriminates repeating messages and/or messages that stem from identical events extracts all of the unique messages from the

log files. These messages are then classified as warnings, errors, or failures. The classification process is a tedious task that involves extensive understanding of system details, system's operating states, system administrators' manual logs, and intuition.

Warnings are messages intended to gain the system administrator's attention but are not considered critical to the operation of the system, e.g., printer jams, repeated disk read/write attempts, etc. *Errors* are messages that indicate an incorrect change in the system state which may lead to a failure, for example the occurrence of a memory error. The observed errors are further categorized by their identifiable sources and the type of service affected. *Failures* in the system are catastrophic events in which the system stops operation. Failures may occur by one of two events: administrator reboots the system or an error put the system in a failure state. In either case, failures are followed by a reboot. Table 1 summarizes all of the data gathered according to the categories and classifications.

4. Clustering

To analyze the warning-error-failure trends, all events are grouped into clusters for each categories. Grouping is done in a similar manner to tuples found in [6]. The clusters graphically show a relationship between error categories in time. Burst of errors can also be seen when the clusters are graphed. A cluster is generated by grouping errors together within a fixed amount of time (ϵ) of each other. The cluster may grow longer than ϵ if the inter-arrival time of the errors is within ϵ of each other.

5. Warning-Error-Failure model

The collected data are then used to build a model that relate warnings and errors for a time period prior to a failure. The assumption is that a warning-and-error state eventually progresses to a system-wide failure. We tried to observe this relationship by clustering and analyzing the warning and error events, beginning at the time-point of fatal system crash and, recursively identifying the proceeding warning/error states.

Since we do not have access to the computer's internal state, we use number of warnings or errors per minute as a measure of a state. Figure 1 shows an example of the approach: each data point in the graph represents an ($\epsilon = 5min.$)-cluster, and the variables considered are the number of errors the cluster contains and the number of warnings. Successive data points are joined by lines, so one can have an idea of the evolution over time of the machine's state.

The linear correlation between the number of warnings and the number of errors can be explained by the large

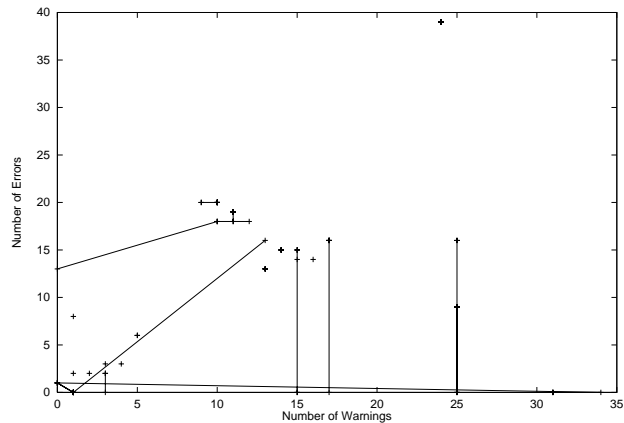


Figure 1. A plot of the number of warnings per cluster vs. the number of errors per cluster

number of messages that are related to the quota system. Each time users exceed their disc quota, two messages are logged: one that we consider a warning, and one that we classify as an error. We did not exclude *a priori* these types of messages from our analysis, but it appears that they predominate, including under failure-free conditions. Therefore, in a future analysis, ignoring these messages may yield more definitive conclusions.

References

- [1] X. Castillo and D. P. Siewiorek, "A workload dependent software reliability prediction model," in *Proc. 12th Int. Symp. Fault-Tolerant Computing*, 1982, pp. 279–286.
- [2] Inhwan Lee, Ravishankar K. Iyer, and Dong Tang, "Error/failure analysis using event logs from fault tolerant systems," in *Proceedings of the 21nd Annual International Symposium on Fault-Tolerant Computing*, Los Alamitos, CA, June 1991, pp. 10–17, IEEE Computer Society Press.
- [3] Michael F. Buckley and Daniel P. Siewiorek, "VAX/VMS event monitoring and analysis," in *FTCS-25: 25th International Symposium on Fault Tolerant Computing Digest of Papers*, Pasadena, CA, Jun 1995, pp. 414–423.
- [4] Anshuman Thakur, Ravishankar K. Iyer, Luke Young, and Inhwan Lee, "Analysis of failures in the Tandem NonStop-UX operating system," in *Proceedings of the Sixth International Symposium on Software Reliability Engineering*, 24-27 Oct 1995, pp. 40–50.
- [5] Ting-Ting Y. Lin and Daniel P. Siewiorek, "Error log analysis: Statistical modeling and heuristic trend analysis," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 419–432, Oct. 1990.
- [6] Michael F. Buckley and Daniel P. Siewiorek, "A comparative analysis of event tupling schemes," in *Proceedings of the Twenty-Sixth International Symposium on Fault-Tolerant Computing*, Washington, DC, June 1996, pp. 294–303, IEEE.