

Empirical testing of Open Source Operating System Reliability: Failure analysis by co-relating event logs and crashes

Ananddeep Pannu and John Kew
Open Source Software Lab, Microsoft
{*apannu, a-johkew*}@*microsoft.com*

Abstract

Open Source Operating Systems offer many advantages over proprietary systems in studying reliability. They are now in widespread use in a variety of specialized versions but are all accessible at the source code level for analysis and modification. In this paper we look at failure analysis within distributions of the Linux operating system. Event logs are vital for failure analysis – but our experiments so far have shown that the event logs of Linux leave a lot to be desired. In this paper we describe our exploration of the event logs and the experiments we performed to look at failure analysis through event logs. We then introduce enhancements to the Linux kernel and describe the advancements that are possible in failure analysis based on these enhancements. These enhancements are again tested through a set of experiments. We analyze the results of these experiments and suggest a few paths to enhance Linux event logs for better failure analysis..

1. Introduction

It is generally accepted that the analysis and assessment of computer systems based on failure and error data collected during operation provides the basis of reliability and dependability enhancements. In most systems, error and failure data is obtained from event logging systems. Event logs include a large amount of information about the occurrence of various types of events; some of these events occur during normal activity and others indicate errors or failures in the system. A number of studies have been performed on dependability analysis using event logs from Windows operating systems [1]. However there has not been a similar emphasis on assessing reliability using event logs from the Linux operating system. This is surprising for two reasons – (1) the Linux event log system is based on the UNIX event log system and there have been reliability studies performed based on event logs for UNIX systems [2] and (2) Linux is an open source operating system which allows for unfettered exploration of the event log system by researchers. In this paper we describe our explorations

of the event log system within several distributions of Linux. We started with the assumption that a modern operating system would capture events with enough detail and fidelity to allow easy diagnosis of certain failures and errors. We then tested this assumption by injecting common failures into the OS and analyzing the event logs after every such failure for several hardware configurations. This method is different from common crash data collection methods where events are collected over a period of time in production systems. There is a flavor of fault injection techniques such as that in [3] our method, but our method introduces only a few faults under very controlled conditions rather than a large number of statistically significant faults introduced in the papers referenced. We contend that based on the findings in the literature, we can safely proceed by testing a few failure modes and draw significant conclusions from them. One of the conclusions we were able to draw was that almost all Linux distributions lag behind in event log capability as compared to Windows just from a study of [1] and our experiments. This then encouraged us to postulate that certain enhancements to Linux event log capability would reduce this gap. We then performed these experiments and were able to use the same failure modes we had chosen earlier to see that the enhancements did indeed provide better diagnosis of failures. We also contend that using such an evolutionary approach we can progress to experiments wherein we can test the failure diagnosis capability of Linux in large scale deployments or under large scale fault injection experiments.

In Section 2 we explain our experimental setup. In Section 3 the experiments we ran are explained and Section 4 details our results and conclusions. Section 5 concludes by talking about planned future work.

2. Experimental setup

The experiments were performed using various distributions and hardware configurations. The distributions chosen were *Red Hat Enterprise Linux*

version 4 (RHEL 4), SUSE Linux Enterprise Server (SLES version 10) and Ubuntu 6.06 (Dapper) Workstation with current patches. These distributions differ widely in modules included but they all use the same Linux kernel 2.6.xx (though with different patch levels). These distributions were used to confirm that there were no significant distribution based differences in the event logs. Since events generated and logged can also depend on the hardware on which commodity operating systems such as Linux run, three different hardware configurations conforming to low, mid and high range hardware were used in the experiment.

The *syslogd* daemon is the standard event logging tool for Linux and UNIX, although some distributions are replacing this with its successor, *syslog-ng*. This background process records events generated by different local sources: kernel, system components (disk, network interfaces, memory), daemons and application processes that are configured to communicate with *syslogd*. The *syslogd* generated event logs for each machine were recorded locally and collected over a network link to a central repository.

3. Running the experiments

The faults and failures shown in

Table 1 were injected into the system. These injected failures capture a wide range of failure modes, since a kernel panic may be induced by a driver and a memory fault may be induced by a application accessing illegal memory locations.

Table 1

Scenario	Type
Crashme – user space testing program	Forced Crash
Unplug Non Boot Hard Drive	Abnormal Operation
Unplug Boot Hard Drive	Abnormal Operation
Unplug Power on machine	Abnormal Operation
Errant kernel module – Panic	Forced Crash
Errant kernel module – Memory Overflow	Forced Crash

The experiments were first performed with the base syslog capabilities and then repeated with the inclusion of a heartbeat mechanism and the use of the *kdump* module in the kernel with a stack trace logging. In each case after a fault was injected, the event logs were looked at to see if that fault could be detected from the event log.

4. Experimental Results & Conclusions

In the case of the default syslog capabilities the following events could *not* be detected

- Kernel panic
- Memory overwrite
- Boot disk failures
- User space program induced freezes
- Most power failures

This is indeed surprising for a modern operating system since in [1] the authors clearly indicate that Windows NT and 2K systems were able to identify abnormal shutdowns of this kind.

Using the heartbeat and *kdump* mechanisms, which were relatively simple enhancements, the following were able to be diagnosed

- Kernel panic
- Memory overwrite
- User space program induced freezes

We can safely conclude that standard Linux distributions need enhancements in their event logs and the enhancements needed are minor and can be incorporated by re-configuration of already existing modules.

5. Future work

We intend to test our conclusions from these simple experiments in production environments. We have identified sites in which we can put a mix of enhanced and baseline Linux systems. This will allow us to draw conclusions about the impact of event log based failure analysis on reliability.

10. References

- [1] Crista Simache, Mohamed Kaaniche and Ayda Saidane. “Event Log based dependability Analysis of Windows NT and 2K systems”, *Proceedings of the 2002 Pacific Rim International Symposium on Dependable Computing*, IEEE, 16-18 Dec. 2002 ,pp311 - 315.
- [2] Crista Simache and Mohamed Kaaniche. “Measurement-based availability analysis of Unix systems in a distributed environment”, *Proceedings of the 12th International Symposium on Software Reliability and Engineering*, 2001. *ISSRE 2001.*, IEEE, 27-30 November 2001.
- [3] Weining Gu, Zbigniew Kalbarczyk, Ravishankar K. Iyer, Zhenyu Yang. “Characterization of Linux Kernel Behavior under Errors”, *Proceedings of the 2003 International Conference on Dependable Systems and Networks*, *DSN 2003.*, IEEE, 22 -25 June 2003.