A Framework for Assessing the Dependability of Supercomputers via Automatic Log Analysis

Catello Di Martino[1], **Domenico Cotroneo**[1],**Zbigniew Kalbarczyk**[2], **Ravishankar K. Iyer**[2]
(1) *Dipartimento di Informatica e Sistemistica, Universita' di Napoli Federico II,*
Via Claudio 21, 80125 Napoli, Italy - ph: +39 081 7683812, fax: +39 0817683816
(2) Center for Reliable and High Performance Computing, University of Illinois at Urbana - Champaign
1307 W. Main St., 61801 Urbana, IL, USA - ph:
{*catello.dimartino, cotroneo*}*@unina.it*,{*kalbar, rkiyer*}*@uiuc.edu*

1 Introduction

We are witnessing an era where large scale machines are a reality. Today's trend in building such machines is to use hundreds of thousands of off-the-shelf CPUs such as the PowerPC, Itanium, or x86-64, combined with custom [2] and/or off-the-shelf interconnections. This trend is further assisted by recent advantages in system integration scaling, making plausible the use of multi core architectures to further improve the computational capabilities of these systems. However, these technological trends, together with the inclination in decreasing die's voltage, make these systems more susceptible to transient hardware failures and memory bit flips [6]. Moreover, the high power consumption for these supercomputers leads to immense heat dissipation, which consequently can accelerate the failure rates of different devices like the memory, the I/O subsystems and the CPUs [4]. As the hardware scales, the software also becomes more complex and more vulnerable to failures (higher failure rate, and larger number of computing nodes). Hence, system reliability is exacerbated with respect to former computing systems [5]. To address this trend a plethora of fault tolerant techniques have been proposed in literature including a variety of checkpointing methods[1]. However, recent studies [4, 5] indicate that conventional techniques in the field, such as periodic checkpointing, can significantly impact the potential performance gains, especially under failure conditions. Checkpointing and recovering a job that involves thousands tasks may take up to tens of minutes. This overhead, and the accentuated failure behavior tend to invalidate common assumptions [3] (e.g. Mean Time To Failure $>>$ checkpointing time). Making quantitative assessments of dependability and characteristics of today's supercomputers still an open issue. Field Failure Data Analysis (FFDA) represent in this scenario an effective viable and viable strategy, heading toward a better understanding of such characteristics.

This fast abstract presents a framework for dependability assessment of supercomputers via failure data analysis. The proposed framework is tailored i) to provide an automatic classification of the information contained in the failure data, ii) to provide detailed statistics on failure/error distributions, and iii) to assess the effectiveness of fault tolerance techniques (e.g. checkpointing) through the evaluation of a set of metrics such as: Availability, number of jobs to roll back due to a failure, average number of nodes available for computation, mean time to completion of a job. A preliminary application of the framework is proposed in the context of a FFDA campaign conducted on two different supercomputers at National Center for Supercomputing Applications NCSA[1] at the University of Illinois Urbana-Champaign. Failure data has been collected through three different types of log files, i.e. the Linux syslog , the Moab platform log [2], and the IBM General Parallel File System (GPFS) log [3]. Currently different types of failures have been encountered, and they have been used to identify a model of the logs, here referred as *dictionary of failures*.

2 Approach Description

In this study, logs are obtained from cluster logging daemons. Periodically the logs are moved to a server in charge of executing the framework. The three types of logs provide information on i) the date and time of the event, ii) the name of the node involved with the event, iii) the nature and the gravity of the event, and iv) a message with the description of the event. In a preliminary deployment of the framework, we collected about a 100 million entries for MoabLogs, 52 millions for the syslog and about 8 million for the GPFS log, totalizing of about 30 Gigabyte of data.

Figure 1 depicts the approach accomplished by the framework. We can broadly describe two macro-phases: the setup phase (first phase) and the operational phase (second phase). The setup phase is in charge of building a model of log files, throughout a classification of all the log messages. The raw logs contain a tremendous amount of data, much of which is repeated or redundant. Before we can use these logs to study machines' failure behavior, we must first filter out the useless data and isolate unique failure messages. Hence, in this phase, we first substitute all addresses (e.g. IP, MAC), machine IDs, dates, user names etc. with a set of wildcards. For instance the username Jane_Doo is replaced with the wildcard $USER. This way, log entries that differ only by these parameters are described by the same string. We are thus able to collapse several million of entries in only 823 distinct log entries for syslog (2314 and 57 respectively for moablog and GPFS), by simply querying this set for distinct event messages [4]. We now have a set of distinct entries, that are representative of all the possible event logged in the analyzed data. However, they still contain a lot of noise. Hence,we build a set of regular expression that matches all

[1] www.ncsa.uiuc.edu

[2] Moab Workload Manager is a scheduling, resource and checkpointing/recovery coordinator. - www.clusterresources.com

[3] www-03.ibm.com/systems/clusters/software/gpfs/

[4] The framework uses a database and a web-based interface as support for data mining

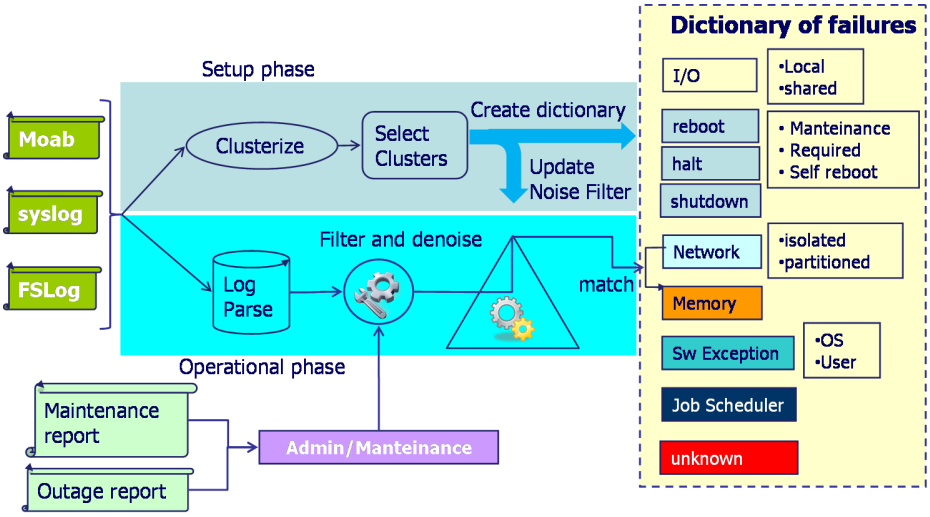

Figure 1: The approach pursued in the framework

the noisy messages thus creating a black list. We then further process the data in order to gather a clean set of entries that we believe belonging to the events of interest[5]. With this filtering, we achieved a set of 122 distinct failure event messages for syslog, and 170 and 30 respectively for the Moab and the GPFS.

These further shrunk sets allowed us to manually tag each message format, specifying the subsystems they refer to. This allows us to implement a simple, but effective, automatic classification of the messages based on the observed message format present in the analyzed data. These classified messages, constitute our *dictionary of failures*, as outlined in Figure 1. It is worth noting that we extend the *dictionary* of the observed failures by adding further data from maintenance/outage reports, so that, in our analysis, we can easily take into account system downtime due to maintenance/general outage (e.g. power outage).

The second phase is the true operational phase. Logs are analyzed by the framework, filtering out all the messages tagged as noise in phase 1. Filtered logs are then coalesced using a hybrid coalescence technique, i.e. content-based and temporal (spatial) coalescence. With this technique, a failure event is first coalesced on a specific class of failures, in accord to the log format present in the *dictionary*. After that, failure events within the same class are further coalesced with respect to time, within a given windows size. A suitable width of the window size is chosen after performing a sensitivity analysis on the number of collapsed entries in a tuple, as a function of the windows size itself.

In order to exploit possible correlations between entries of different classes of failure and/or different nodes, we consider that i) failures with timestamp in the same window are likely representative of the same event, and ii) failures on different nodes within the same window are likely to be correlated.

At the end of the described workflow, all the logs filtered so far are finally used for evaluating i) a set of metrics (e.g. dependability attributes and metrics like the average number of rolledback jobs due to a failure, and average time needed for checkpointing.), and ii) a set of statistics and distributions of the observed failure behaviors.

[5]We used very extensively both the GPFS and the Moab manuals to find out the actual failure/error event messages, while for the syslog we rely on the experience from our past work.

3 Conclusions and Future Work

This paper briefly presents a framework conceived for supporting supercomputer Failure Data analysis. The objective is i) to automate the process of building a field data model by means of a *dictionary of failures*, and ii) to simplify the off line assessment of supercomputers by means of Field Failure Data Analysis, making possible to evaluate a set of dependability metrics based on real data. Future work will deal with a FFDA campaign on the considered machines in order to extensively assess their dependability, and the effectiveness of the adopted fault tolerance mechanisms.

References

[1] Checkpointing for peta-scale systems: A look into the future of practical rollback-recovery. *IEEE Trans. Dependable Secur. Comput.*, 1(2):97–108, 2004. Member-Elmootazbellah N. Elnozahy and Member-James S. Plank.

[2] José E. Moreira et Al. The blue gene/l supercomputer: a hardware and software story. *Int. J. Parallel Program.*, 35(3):181–206, 2007.

[3] G. Kavanaugh and W. Sanders. Performance analysis of two time-based coordinated checkpointing protocols. In *PRFTS '97: Proceedings of the 1997 Pacific Rim International Symposium on Fault-Tolerant Systems*, page 194, Washington, DC, USA, 1997. IEEE Computer Society.

[4] Yinglung Liang, Yanyong Zhang, Anand Sivasubramaniam, Morris Jette, and Ramendra Sahoo. Bluegene/l failure analysis and prediction models. In *DSN '06: Proceedings of the International Conference on Dependable Systems and Networks*, pages 425–434, Washington, DC, USA, 2006. IEEE Computer Society.

[5] Zbigniew Kalbarczyk Ravishankar K. Iyer Lawrence Votta Alan Wood Long Wang, Karthik Pattabiraman. Modeling coordinated checkpointing for large-scale supercomputers. In *DSN '05: Proceedings of the 2005 International Conference on Dependable Systems and Networks*, pages 812–821, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Jayanth Srinivasan, Sarita V. Adve, Pradip Bose, and Jude A. Rivers. The impact of technology scaling on lifetime reliability. In *DSN '04: Proceedings of the 2004 International Conference on Dependable Systems and Networks*, page 177, Washington, DC, USA, 2004. IEEE Computer Society.