Performance Implications of Failures in Large-Scale Cluster Scheduling

Yanyong Zhang[1], Mark S. Squillante[2], Anand Sivasubramaniam[3], and Ramendra K. Sahoo[4]

[1] Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854, USA
yyzhang@ece.rutgers.edu
[2] Mathematical Sciences Department, IBM T.J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598-0218 USA
mss@watson.ibm.com
[3] Department of Computer Science and Engineering, Pennsylvania State University, 316 Pond Laboratory, University Park, PA 16802-6106 USA
anand@cse.psu.edu
[4] Exploratory Server Systems Department, IBM T.J. Watson Research Center, 1101 Kitchawan Road, Yorktown Heights, NY 10598-0218 USA
rsahoo@us.ibm.com

Abstract. As we continue to evolve into large-scale parallel systems, many of them employing hundreds of computing engines to take on mission-critical roles, it is crucial to design those systems anticipating and accommodating the occurrence of failures. Failures become a commonplace feature of such large-scale systems, and one cannot continue to treat them as an exception. Despite the current and increasing importance of failures in these systems, our understanding of the performance impact of these critical issues on parallel computing environments is extremely limited. In this paper we develop a general failure modeling framework based on recent results from large-scale clusters and then we exploit this framework to conduct a detailed performance analysis of the impact of failures on system performance for a wide range of scheduling policies. Our results demonstrate that such failures can have a significant impact on the mean job response time and mean job slowdown under existing scheduling policies that ignore failures. We therefore investigate different scheduling mechanisms and policies to address these performance issues. Our results show that periodic checkpointing of jobs seems to do little to ease this problem. On the other hand, we demonstrate that information about the spatial and temporal correlation of failure occurrences can be very useful in designing a scheduling (job allocation) strategy to enhance system performance, with the former providing the greatest benefits.

1 Introduction

Our growing reliance on computing and information processing services mandates not only deploying systems that can meet the performance demands imposed on such systems, but also those that are available when needed. Several technological factors are accentuating the problem of system failures, which are highly undesirable since these

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 233–252, 2005.

systems could be servicing the needs of hundreds of users. At the same time, solutions for this problem need to keep the high costs of system maintenance personnel in mind, which is growing to be a much more important factor in Total Cost of Ownership (TCO). A deep understanding of the occurrence of failures in real environments can be useful in several ways towards enhancing overall system availability. It can provide realistic data when evaluating proposed solutions, together with developing strategies for proactive prediction and remedies of faults ahead of their occurrence. Application demand for high performance is continuing to fuel research and development of large scale parallel systems. The need for processing larger datasets in existing applications, and the stringent demands of emerging applications necessitate parallelism in computational and storage devices for their deployment. The cost-effectiveness in using off-the-shelf hardware to put together clusters has contributed to a large extent in the widespread availability of parallelism, and its successful usage. At the same time, several important and challenging applications are driving the development of large scale parallel machines, such as IBM's BlueGene/L which is anticipated to have 65536 nodes.

As we continue to develop such large scale parallel systems, there are several important technological factors to keep in mind:

- Denser integration of semiconductor circuits, though preferable for performance, makes them more susceptible to strikes by alpha particles and cosmic rays [41]. At the same time, there is an increasing tendency to lower operating voltages in order to reduce power consumption. Such reduction in voltage levels can increase the likelihood of bit-flips when circuits are bombarded by cosmic rays and other particles, leading to transient errors. While memory structures are typically the target for protection against errors using informational redundancy, more recent studies [31] have pointed out that the error rates in combinational circuits are likely to surpass those of memory cells in the next decade.
- At the macro granularity, we have dense blade-systems being packed in a rack as a cluster. With a high load imposed on these dense systems – both on the CPUs and on the disks – heat dissipation becomes a very important concern, potentially leading to thermal instability that can cause system/node breakdowns [25,9].
- We find system software and applications becoming more complex. Such complexity makes them more prone to bugs and other software failures [35,23,38] (e.g. memory leaks, state corruption, etc.). These bugs/failures can cause system crashes, and it has even been suggested that one should perform pro-active shutdown/rejuvenation [39,38] to avoid catastrophic consequences.

All these factors point to the increasing occurrence of system failures in the future. Failures become more commonplace when we consider parallel systems with thousands of nodes. Rather than treat them as an exception, system design needs to recognize fault occurrence, and manage the resources across the parallel system effectively so as to hide their impact from the end users. One would ideally like to achieve the performance of a system without any failures. Even if this is difficult to attain, there should be at most a "graceful degradation" in performance under the presence of failures. Towards this goal, the present paper specifically targets the *management of CPU resources on a large scale parallel system using a general failure modeling framework that accurately*