Proactive Fault Manager for High Performance Computing

Yawei Li and Zhiling Lan
Department of Computer Science
Illinois Institute of Technology, Chicago, IL 60616
{liyawei, lan}@iit.edu

1. Introduction

As the scale of high performance computing (HPC) grows, the failure rate of supercomputers also increases dramatically. For example, the mean-time-to-failure (MTTF) of the ASCI White system, which is composed of 8,192 CPUs, is only about 40 hours. Checkpoint/restart is the most pervasive fault-tolerance (FT) technique used in HPC, yet it suffers from severe performance degradation, enormous requirement of storage, and significant overhead of writing to disks. Furthermore, this reactive fault tolerance approach can not avoid long failure downtime, which is intolerable for performance-demanding applications. To address the performance issue incurred by failures and avoid the disadvantages associated with reactive solutions, we adopt the proactive fault management concept and propose a novel adaptive prevention manager (APM) for high performance computing, which is part of our proposed FT-Pro framework as shown in Figure 1. The APM aims at minimizing the total execution time of parallel applications, e.g. MPI applications, by proactively migrating or checkpointing suspicious process(es) based on the failure prediction.

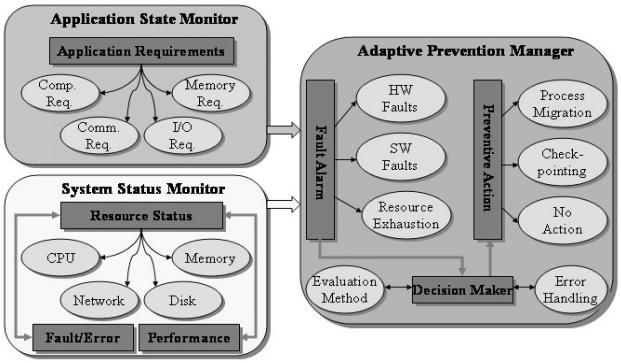

Figure 1. APM in the FT-Pro Framework

This work makes two major contributions. First of all, a proactive fault management framework is proposed for high performance computing. It targets general HPC applications on massively parallel systems with the objective to mitigate the application-level computation loss caused by both hardware and software failures. Instead of maximizing dependability of critical systems or improving availability of business services in most related research works, this work emphasizes on optimizing the application's performance that is the principle concern of HPC users. Secondly, a fault prevention manager is designed that adaptively takes suitable preventive actions based on the observation of the status of applications and resources. A novel evaluation scheme is proposed to either dynamically invoke process migration to avoid imminent failures or perform checkpointing to prevent unpredictable failures.

2. Adaptive Prevention Manager

The APM is a crucial part in the FT-Pro framework. It consists of three major components: fault alarm, decision maker, and preventive actions.

The fault alarm component collects the status information from the system and application monitors so as to forecast faults, which might be hardware-related, software-related or resource exhaustion errors. The predicted faults will then be signaled to the decision maker component.

The decision maker chooses a preventive action based on its evaluation of the performance gain and cost of each possible preventive action. Currently, three possible actions are included in the APM:

- No action. When the system is in a fault-free state, there is no need to take any action and it can avoid unnecessary overhead. This option optimistically eliminates any interruptions from fault handling.
- Process migration: When a node is in a fault-prone state, the process is migrated to another healthy node so as to avoid the potential severe failure.
- Checkpointing: Some failures may not be predicted at all. For these unpredictable failures, checkpointing is performed to save intermediate results so as to reduce the recovery cost associated with unpredictable failures.

The design of gain/cost model is a challenging problem. A good tradeoff should be made between an aggressive solution that interrupts the running application as less as possible and a conservative solution that conducts preventive actions frequently to guarantee dependability. We have designed a heuristic scheme (as shown in Figure 2) that always takes the best immediate solution so as to find the overall optimal solution for the problem, which is to minimize the execution time of the target application. In this

scheme, the aggregated execution time of all the processes is used as the evaluation criterion. This is due to two major reasons: 1) Failure recovery and checkpointing involve all the processes while process migration only affects the suspicious process. 2) HPC users are charged by amount of service units (SU) used which are calculated based on total CPU time on most supercomputers or clusters.

The preventive action component runs in the background as a multi-threaded daemon. It is triggered by decision maker and cooperates with the target application through callback interface to carry out actual operations.

Since APM is based on fault prediction which might not be always correct, error handling is an important part of APM. In particular, false-positive and false-negative cases should be taken care of. The error-handling part is responsible for recording the actual result of preventive decisions and dynamically changing the value of *threshold* used in the decision making to adjust the frequency of checkpointing or process migration.

c: fault prediction confidence;
p: process set;
T: current time;
T_f : predicted failure occurrence time;
T_w: predicted remaining time per process;
T_{last}: last checkpoint time;
$T_{recovery}$: recovery time per process;
T_{cp}: cost of checkpointing per process;
T_{pm}: cost of process migration per process;
T_{no_total}: aggregated cost with no preventive action;
T_{cp_total}: aggregated cost if checkpointing;
T_{pm_total}: aggregated cost if process migration;

if $((T_w < T_f) \| (c<threshold))$ take no action option
else {

$$T_{no\_total} = \sum_p ((1-c) * T_w + c*(T_w+T_f-T_{last}+T_{recovery}))$$
$$T_{cp\_total} = \sum_p ((1-c) * T_w + c*(T_w+T_f-T+T_{cp}+T_{recovery}))$$
$$T_{pm\_total} = \sum_p T_w + T_{pm}$$

}
if $(T_{no\_total} < T_{cp\_total}$ and $T_{no\_total} < T_{pm\_total})$
take no action option
else if $(T_{cp\_total} < T_{pm\_total})$

Figure 2. Heuristic Decision Making Scheme

3. Related Works

Sahoo [4] analyzed the statistical properties of system errors and failures from a network of nearly 400 AIX servers at IBM. R. Vilalta[6] propose a rule-based learning system to predict failure events. These works show that both analytical and machine learning techniques can be used for failure prediction in HPC.. However, they do not provide any end-to-end fault management solution for practical applications.

Software rejuvenation [2,5] proactively stops and restarts system in a time-based or prediction-based approach to solve the soft aging problem. It cannot handle hardware failures and has a limited applicability as it requires privilege access in a shared environment. The AMPI-based approach [1] utilizes the processes virtualization to enables a process to dynamically migrate its objects to a safe location when a fault is imminent. This work depends on the specific Charm++ framework and the performance evaluate model of the migration are not given. The MEAD project [3] combines the replicate technique with the failure prediction of resource exhaustion to provide proactive for CORBA applications. Differing from these works that rely on either periodic restart or failover or object migration, our work can adaptively take a preventive action from three options.

4. Ongoing and Future Work

Currently, we are conducting a series of experiments with real-world parallel applications on a 64-node SUN cluster. The objective of these experiments is to compare the proposed proactive fault management framework with the traditional rollback-recovery schemes. We are also investigating more sophisticate preventive schemes that can take into consideration the stochastic characteristic of faulty processes and the correlation of multiple failures on massive parallel systems.

5. Reference

[1] S.Chakravorty, C.L.Mendes, L. V. Kale, "Proactive Fault Tolerance in Large Systems", HPCRI workshop 2005

[2] Y. Huang, C. Kintala, N. Kolettis and N. Fulton, "Software Rejuvenation : Analysis, Module and Applications", IEEE Intl. Symposium on Fault Tolerant Computing, FTCS 25

[3] S.Pertet, P.Narasimhan, "Proactive Recovery in Distributed CORBA Applications," DSN 2004

[4] R.K.Sahoo, A.Sivasubramaniam, M.S. Squillante, Y.Zhang, "Failure Data Analysis of a Large-Scale Heterogeneous Server Environment".DSN 2004

[5] K. Vaidyanathan, R. E. Harper, S. W. Hunter and K. S. Trivedi, "Analysis and Implementation of Software Rejuvenation in Cluster Systems",ACM SIGMETRICS 2001/Performance 2001, June 2001

[6] R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, "Predictive algorithms in the management of computer systems", IBM Systems Journal issue 41-3,2002