Using Adaptive Fault Tolerance to Improve Application Robustness on the TeraGrid

Yawei Li and Zhiling Lan

Abstract— Application robustness becomes a major concern with the continued scaling of high performance computing (HPC). In a recent study [8], we have developed an adaptive fault management scheme called FT-Pro for improving application robustness by combining the merits of proactive process migration and reactive checkpointing. In this paper, we push forward this study by integrating FT-Pro with a production-level MPI package and investigating its effectiveness across a number of real-world parallel applications. Extensive experiments are conducted on an IA32 cluster at TeraGrid/ANL by comparing FT-Pro as against periodic checkpointing under a wide range of system parameters and failure behaviors. These preliminary experiments show the potential of using adaptive fault tolerance to improve application performance in the presence of failures.

Index Terms— Adaptive fault tolerance, Parallel applications, Process migration, Checkpointing

1 INTRODUCTION

Over the past decades, the insatiable demand for more computational power in science and engineering has driven the development of ever-growing supercomputers. Parallel computers with hundreds to thousands of processors, ranging from tightly coupled proprietary clusters to loosely coupled commodity-based clusters, are being designed and deployed. For systems of such scales, the susceptibility to failures becomes a major concern as the system-wide MTBF (mean-time-between-failure) decreases dramatically with the increasing count of components [10]. At the mean time, to accurately model realistic problems, large applications are designed to run for days, weeks, or longer until completion. A common complain from HPC users is that large jobs find it very difficult to make any forward progress because of failures [12]. This situation will be exacerbated as systems get bigger and applications become larger. Hence, improving application robustness is of vital importance to the success of HPC.

The conventional fault tolerance approach is checkpointing. It periodically stores a snapshot of the current application state, and then uses it for restarting the execution in case of failures. While checkpointing has served us well in the past, it can cause serious performance overhead when applied to large-scale applications [6]. Moreover, with the growing gap between processor speed and data access speed, frequent checkpointing can further increase the disparity between sustained performance and peak performance of HPC. Thus, checkpointing alone is unlikely sufficient to provide an efficient and robust solution for future HPC. Unlike reactive checkpointing, a proactive technique (e.g. timely process migration) allows an application to avoid failures by taking preventive actions before their occurrences based on failure predictions. Recent research results [7,11] reveal the feasibility of predicting incoming failures in large systems and thus hold the promise of using proactive approaches to improve application robustness with relatively low cost [5]; however, the fault prediction is subjected to some degree of uncertainty in practice. More specifically, failure predictions have false negative error where unexpected failures occurs and false positive error where false alarms are produced. As a result, entirely relying on failure prediction can cause significant performance loss.

In a recent study [8], we have proposed an adaptive fault management scheme called FT-Pro that combines the merits of proactive process migration and reactive checkpointing to improve application performance in the presence of failures. In this paper, we push forward this study by integrating FT-Pro with the open source package MPICH-V [2] and investigating its effectiveness across a number of real-world parallel applications on an IA32 system at TeraGrid/ANL. FT-Pro is evaluated against periodic checkpointing under a wide range of system parameters and failure prediction accuracies. Moreover, we use a real failure trace in the experiments to reflect the failure situations in realistic HPC production environments. Our preliminary results show that the proposed FT-Pro outperforms the traditional checkpointing by up to 43% in terms of reducing application execution time, even under a modest predictive accuracy. To the best of our knowledge, this study is among the first to exploit adaptive fault management with various real-world applications by using real failure logs.

The rest of the paper is organized as below. Section 2 gives an overview of FT-Pro. In Section 3, we present the design and implementation of FT-Pro with the production-level MPI package MPICH-V. Section 4 describes our evaluation methodologies, followed by experimental results. Finally, we conclude the paper in Section 5.

2 OVERVIEW OF FT-PRO

FT-Pro adopts a cooperative approach where the application programmer or user can insert fault management re-

- *Y. Li is with the Scalable Software Laboratory, Illinois Institute of Technology, 10 West 31st Street, Stuart Building 006,Chicago,IL 60616. Email:liyawei@iit.edu*
- *Z. Lan with the Scalable Software Laboratory, Illinois Institute of Technology, 10 West 31st Street, Stuart Building 226D, Chicago, IL 60616. Email:lan@iit.edu*

quests denoted as decision points in the application and FT-Pro makes run-time adaptation decision on what type of preventive action should be taken on the application at each decision point. The goal of adaptation is to optimize the application execution time by considering both the failure impact and different prevention costs. Three prevention actions are considered in FT-Pro: (1) SKIP, where the application proceeds with its computation; (2) CHECKPOINT, where the application stops to conduct a coordinated checkpointing; and (3) MIGRATION, where the processes residing on failure-prone computing nodes are transferred to healthy spare ones.

The primary adaptation question with regard to FT-Pro is, "How does FT-Pro decide which fault prevention action should be taken based on fault prediction?" Towards this end, we have developed probabilistic models to quantitatively evaluate the application performance by considering a range of factors, including the prediction accuracy, the cost and benefit associated with each prevention action and the characteristics of the application. More specifically, the models are developed to calculate the expected application execution time E_{next} during the next adaptation interval. It involves in three steps:

1. Identifies the number of suspicious nodes W_f and the number of healthy spare nodes S_h according to the failure prediction.
2. Calculates the application failure probability P_f and the application downtime C_d during the next interval.
3. For each action, estimates E_{next} based on the total probability law conditioning on the occurrence of failures. In this step FT-Pro considers the impacts of potential failures and the use of different preventive actions on the application running time.

The detailed calculation of E_{next} is described in Equation (1) - (3). Here, I is the adaptation interval, L is the index of the current interval, f_p is the false positive error rate of prediction, C_{cp} is the checkpointing cost, C_{pm} is the cost of migration and $C_{f(i)}$ is the predicted failure downtime on node i. FT-Pro selects the prevention action with the minimum E_{next}. For more detail of the FT-Pro decision algorithm, please refer to [8].

- if taking SKIP action:

$$E_{next} = \left[(L - L_{ckp} + 2) * I + C_d\right] * P_f + I * (1 - P_f) \tag{1}$$

$$P_f = \begin{cases} 1\text{-}\prod_{i=1}^{W_f} f_p & if \quad W_f > 0 \\ 0, & if \quad W_f \le 0 \end{cases} \quad \& \quad C_d = \frac{1}{W_f} * \sum_{i=1}^{W_f} C_{f(i)}$$

- if taking CHECKPOINT action:

$$E_{next} = (2I + C_d + C_{cp}) * P_f + (I + C_{cp}) * (1 - P_f) \tag{2}$$

$$P_f = \begin{cases} 1\text{-}\prod_{i=1}^{W_f} f_p & if \quad W_f > 0 \\ 0, & if \quad W_f \le 0 \end{cases} \quad \& \quad C_d = \frac{1}{W_f} * \sum_{i=1}^{W_f} C_{f(i)}$$

- if taking MIGRATION action:

$$E_{next} = (2I + C_d + C_{cp} + C_{pm}) * P_f + (I + C_{cp} + C_{pm}) * (1 - P_f) \tag{3}$$

$$P_f = \begin{cases} 1 - \prod_{i=1}^{W_f - S_h} f_p & if \ W_f > S_h \\ 0 & if \ W_f \le S_h \end{cases} \quad \&$$

$$C_d = \begin{cases} \frac{1}{W_f - S_h} * \sum_{i=1}^{W_f} C_{f(i)} & if \ W_f > S_h \\ 0 & if \ W_f \le S_h \end{cases}$$

3 Design and Implementation

We have implemented FT-Pro with MPICH-V, an open-source checkpointing library for MPI applications [2]. The package consists of three major components: MPICH-V daemons that are collocated with application processes on computing nodes, a dispatcher that is responsible for managing computing resources, and a CKP server that implements the Chandy-Lamport algorithm [4] based coordinated checkpointing protocol and provides storages for the checkpointing images. Figure 1 presents the integration of FT-Pro with MPICH-V. To allow preemptive process migration, when submitting a job, the user needs to request spare nodes to help the application avoid foreseeable failures during execution. Our study based on real error logs indicates that in practice the probability of multiple simultaneous failures is low, hence the allocation of only one or a couple of spare nodes is sufficient.

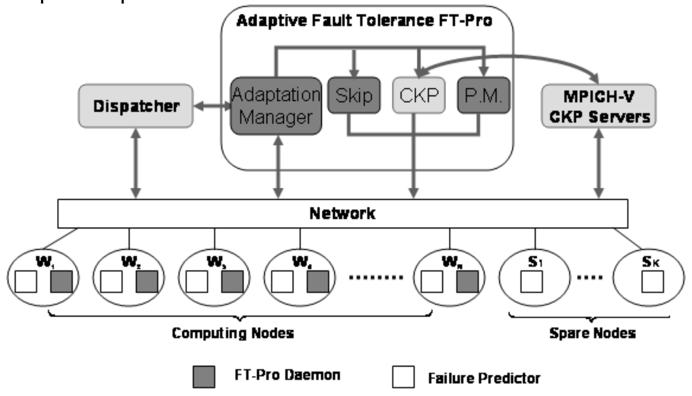


Fig. 1. Design of FT-Pro with MPICH-V

As shown in the figure 1, we have modified and developed several FT-Pro components within MPICH-V to enable adaptive fault tolerance. A new component called *adaptation manager* is developed, which is responsible for runtime decision-making. The *failure predictor* runs on each application node to monitor the RAS state of the host node and report failure prediction results. The *FT-Pro daemon* running on each computing node cooperates with the adaptation manager to fulfill the FT-Pro adaptation actions.

At each decision point, the adaptation manager polls the failure predictor on each node, makes a runtime prevention decision, and then communicates with FT-Pro daemons to invoke the decided action:

- Upon a SKIP action, the adaptation manager broadcasts a SKIP message to daemons and no action will be taken to avoid unnecessary prevention cost.
- Upon a CHECKPOINT action, the adaptation manager broadcasts a CKP message to daemons and invokes the CKP server to conduct a checkpointing operation, and thus

the checkpoint image files are generated and stored at the remote server.

- Upon a MIGRATION action, the adaptation manager broadcasts a PM message along with the addresses of the suspicious and destination nodes to daemons and invokes the CKP server to conduct a checkpoint followed by a proactive restart on the new nodes. We develop process migration support on top of MPICH-V packages. To be less intrusive, currently process migration is implemented as a coordinated checkpointing immediately followed by a proactive restart. That is, all the application processes are terminated and then restarted on the new set of computing nodes which consist of the original healthy nodes belonging to the application and the healthy spare nodes which have been swapped in by the process migrations.

4 Performance Evaluation

4.1 Evaluation Methodology

4.1.1 Environment

To gain the practical insight on the effectiveness of FT-Pro, we conduct trace-based simulations to compare FT-Pro with periodic checkpointing (CKP) . The testbed is the IA32 Linux cluster at Argonne National Laboratory, which is part of the TeraGrid infrastructure. The system consists of 96 computing nodes, each equipped with two 2.4GHz Intel Xeon processors and 4G MB memories. All the nodes are connected via Gigabyte Ethernet and sharing 4TB disk via NFS, which also provide stable remote storage for checkpointing image files in FT-Pro. The operation system is SuSE Linux v8.1, and the job scheduler is Torque Resource Manager with Moab Scheduler. The MPICH-V is of version 0.76 and the compiler is GCC 2.95.3.

4.1.2 Failure Trace

To reflect the realistic failure impact on parallel applications, in the experiments we simulate the failure events based on a real failure trace collected from a 480-node HPC production system [9]. According to the failure trace, the mean-time-before-failure (MTBF) of the entire system is 0.79 hour and it is 14.2 days per node. We use the failure trace from 96 randomly selected nodes to match the system size of the testbed. For every failure entry in this trace, there are four associated properties: node location, failure time, failure type and downtime. Based on these entries we generate corresponding failure events to interrupt the parallel applications. And the failure prediction is simulated in such a way that its accuracy is controlled with two parameters:

- To simulate false-negative value of f_n: If there exists a failure on a node between the current and the next decision points, the event-based predictor on the node reports a failure of its type with the probability of 1- f_n;
- To simulate false-positive value of f_p: Suppose the predictor has totally reported x failures for those intervals with actual failure events. According to the definition of *fp*, the ratio of number of false positives to the number of true positives is $f_p/(1-f_p)$. Hence, out of all the intervals without actual failure occurrences, the predictor randomly selects $(x \cdot f_p)/(1-f_p)$ intervals and raises a false failure alarm for each of them.

4.1.3 Parallel Applications

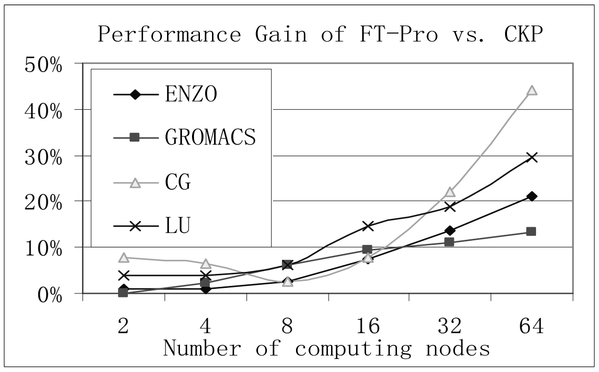
A number of parallel applications, including both parallel benchmarks and two real world parallel simulation codes, are used in the experiments. This test suite is from a mixture of scientific domains, which can provide us a fair evaluation of FT-Pro across a variety of representative parallel applications:

- NAS Parallel Benchmarks: We use the benchmark CG and three pseudo applications (BT, LU, SP), with dataset size of class C representing typical large dataset for modern HPC systems. BT represents applications that are dominated with point-to-point communications; CG represents applications that are dominated by unstructured long-distance communications; LU represents the computation of implicit CFD algorithms with a large amount of messages; and in SP, multiple non-diagonally dominant and scalar pentadiagonal equations are solved.
- Cosmology Application ENZO: It is a parallel cosmology simulation code designed for high-resolution, multi-physics, cosmological structure formation simulations by using the SAMR (Structured Adaptive Mesh Refinement) algorithm [3]. ENZO entails the detailed computations that simulate the formation and evolution of cosmic structures such as galaxies and clusters of galaxies from shortly after the big bang to the present day.
- Molecular Dynamics Application GROMACS: GROMACS is a molecular dynamics (MD) simulation code [1]. MD is a numerical simulation technique where the time evolution of a set of interacting atoms is followed by integrating their equations of motion. MD is widely used in the area of computational chemistry for discovering complex chemical systems in terms of realistic atomic models.

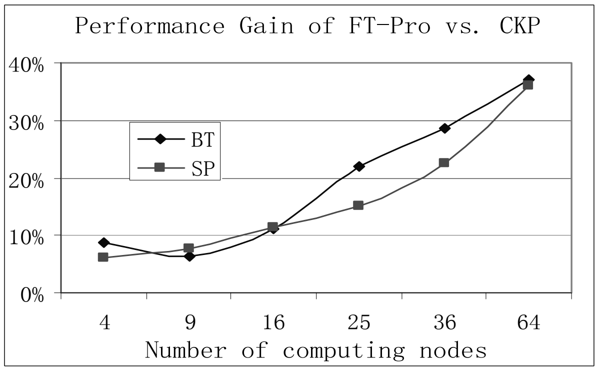
4.2 Performance Results

For the convenience of comparison, a relative metric namely *performance gain* is used, which is defined as the relative reduction of execution time achieved by FT-Pro over periodic checkpointing.

Figure 2(a) and 2(b) shows the performance gain achieved by FT-Pro with these applications under different computing scales, where the default values of f_p and f_n are set to 0.3 to reflect the quality of the prevailing predictors. We vary the number of computing nodes from 2 to 64. The results show that for all the six applications, FT-Pro demonstrates good scalability. That is, the performance gains introduced by FT-Pro increase with the application scales. For instance, FT-Pro achieves significant performance gain ranging from 10% to 42% as the MPI applications run on the 64-node scale. This indicates that FT-Pro is more superior to checkpointing when the computing scale is larger because checkpointing imposes more serious overhead on applications due to the increasing synchronization cost in the checkpointing protocol.

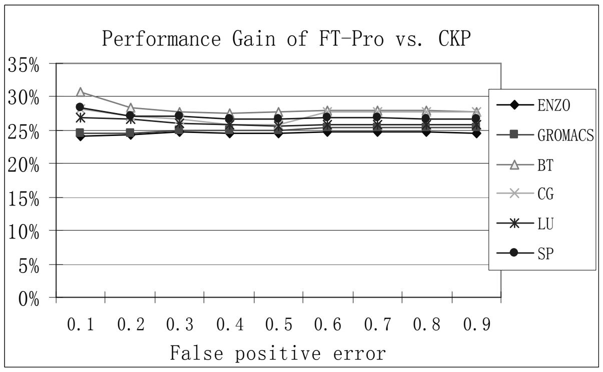


(a)

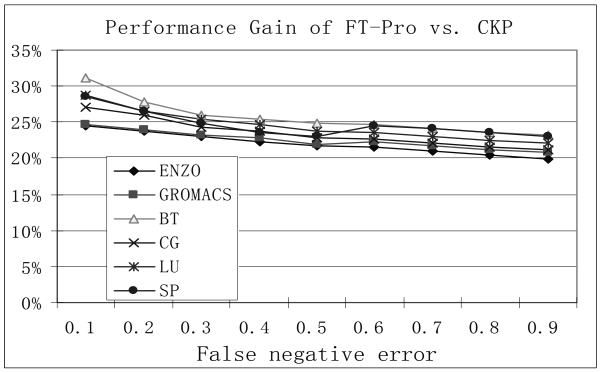


(b)

Fig. 2. Relative Improvement achieved by FT-Pro with Varying Computing Scales (f_p = f_n=0.3, under the condition that only one spare node is allocated.)



(a)



(b)

Fig. 3. Relative Improvement achieved by FT-Pro with Varying Prediction Accuracy Values (No. of computing nodes is 64, and the no. of spare nodes is 1)

Figure 3 shows the performance gain achieved by FT-Pro with varying prediction accuracies: figure 3(a) presents the result where the false-positive value f_p is changed from 0.1 to 0.9 and the false-negative value f_n is fixed at 0.3, and the figure 3(b) gives the result where the false-negative value f_n is changed from 0.1 to 0.9 and the false-positive value f_p is fixed at 0.3. For all of these applications, the performance gain is always between 24% and 30%. The results also show that performance variance incurred by the changes of false positive f_p is trivial, whereas performance degradation caused by false negative fn is more pronounced because the prevention effect of FT-Pro is essentially based on the capability of predicting failures.

Figure 4 presents the performance gain achieved by FT-Pro with different number of spare nodes S varying from 1 to 4. The values of f_p and f_n are set to 0.3 and the number of computing nodes is set to 64. As figure 4 shows, with the increasing number of spare nodes, the performance gain brought by FT-Pro grows steadily. We also observe that even with only one spare node, the performance gain is still significant (e.g. between 13% - 43%) for all these applications. This is due to the fact that the probability of multiple simultaneous failures is rare. In other words, a modest allocation of spare nodes (less than 5%) can result in a substantial performance gain.

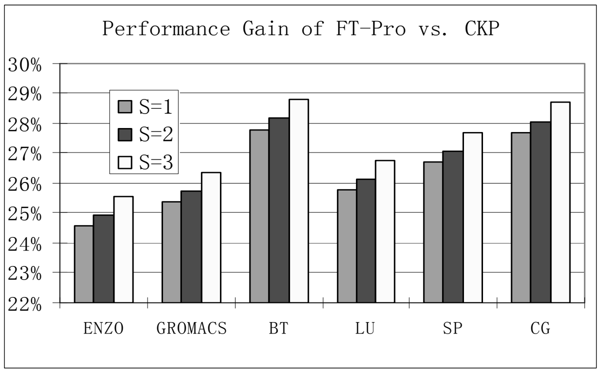


Fig. 4. Relative Improvement Achieved by FT-Pro with Varying Numbers of Spare Nodes (No. of computing nodes is set to 64)

5 Conclusions and Future Work

In this paper, we present the design and implementation of adaptive fault tolerance by integrating our recently developed adaptive scheme FT-Pro with the open-source checkpointing library MPICH-V. The resulting runtime library intelligently chooses prevention actions among proactive process migration, reactive checkpointing, and no-action on-the-fly based on failure prediction. The primary goal is to optimize application execution time in the presence of failures. We have evaluated the developed adaptive fault tolerance mechanism as against periodic checkpointing across a number of real-world parallel applications. Our experimental studies show that FT-Pro can significantly reduce the application completion times by up to 43% as compared to checkpointing. The performance benefit brought by FT-Pro is caused by its intelligence to avoid imminent failures and to remove unnecessary checkpointing operations.

Our future work includes integrating with online failure prediction [7] to provide an end-to-end fault tolerance support for parallel applications and developing lightweight migration protocols to support live migration. Our ultimate goal is to develop an adaptive fault tolerance support for the production usage on TeraGrid.

ACKNOWLEDGEMENT

The authors would like to thank TeraGrid and Argonne National Laboratory for the use of computing resources.

REFERENCES

[1] Berendsen, H.J.C., van der Spoel, D. and van Drunen, R, "GROMACS: A Message-Passing Parallel Molecular Dynamics Implementation", Comp. Phys. Comm. 91 (1995), 43-56.
[2] Aurelien Bouteiller, Thomas Herault, Geraud Krawezik, Pierre Lemarinier, Franck Cappello, "MPICH-V Project: A Multiprotocol Automatic Fault Tolerant MPI", International Journal of High Performance Computing and Applications, 2005.
[3] G.Bryan, T.Abel, and M.Norman, "Achieving Extreme Resolution in Numerical Cosmology Using Adaptive Mesh Refinement: Resolving Primordial Star Formation", Proc. of SC2001, Denver, CO, 2001.
[4] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. ACM Transactions on Computer Systems, 3(1):63--75, 1985.
[5] Sayantan Chakravorty, Celso L. Mendes and Laxmikant V. Kale, "Proactive Fault Tolerance in Large Systems", Proc. of HPCRI Workshop in conjunction with HPCA 2005, 2005.
[6] E. Elnozahy and James S. Plank, "Checkpointing for Peta-Scale Systems: A Look into the Future of Practical Rollback-Recovery", IEEE Transactions on Dependable and Secure Computing, Volume 1, Number 2, April-June, 2004, pp. 97-108.
[7] Z. Lan, P.Gujrati, Y.Li, Z.Zheng, R.Thakur and J.White, "A Fault Diagnosis and Prognosis Service for TeraGrid Clusters", The 2nd TeraGrid Conference, IN, 2006 (remove this one and list AUTODP TG07 paper)
[8] Yawei Li, Zhiling Lan, "Exploit Failure Prediction for Adaptive Fault-Tolerance in Cluster Computing", CCGrid 2006
[9] Charng-Da Lu, "Scalable Diskless Checkpointing for Large Parallel Systems" Ph.D. thesis, University of Illinois at Urbana-Champaign, 2005
[10] Daniel A. Reed, Charng-da Lu, Celso L. Mendes, "Big Systems and Big Reliability Challenges", Proc. of Parallel Computing 2003, Dresden, Germany, September 2003.
[11] Ramendra K. Sahoo, A. Oliner, et al., "Critical Event Prediction for Proactive Management in Large-scale Computer Clusters", Proc. of KDD 2003: 426-435
[12] Y. Tanaka, H. Takemiya, S. Sekiguchi, S. Ogata, A. Nakano, R. Kalia, P. Vashishta, "Adaptive Grid-enabled SIMOX Simulation on Japan-US Grid Testbed", The 1st Annual TeraGrid Conference, Indianapolis, IN, 2006.