Towards Application Level Node Failure Recovery

Using Fault-Tolerant Open MPI in a PDE Solver

Mohsin Ali, ANU CECS Computer Systems Group, md.ali@anu.edu.au*. Supervisor: Peter Strazdins, Peter.Strazdins@cs.anu.edu.au*.

Overview

Imagine that there is a big application, like weather forecasting, is executing on a supercomputer, and after finishing 99.99% of its execution one of the components, such as node, fails. Then what will happen?

- The application could not complete its execution.
- The 99.99% of execution all are meaningless.
- There are wastages of lots of resources and time.

The probability of node failures of a supercomputer is roughly proportional to its size. The next generation supercomputers will have 100 times more nodes than today. So they will fail 100 times more quickly [1] (Fig. 1).

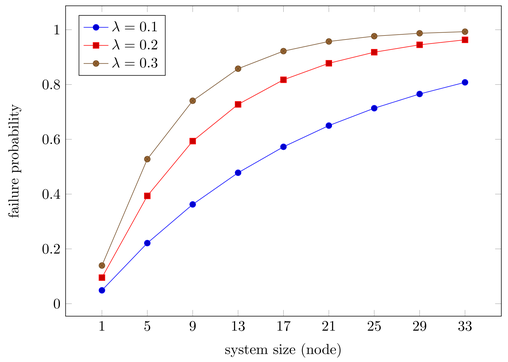

Figure 1: Failure probability increases with the increase of system size. They also varies with the change of failure rate, λ.

Failure Recovery Techniques

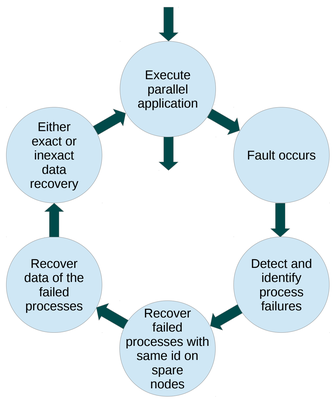

Figure 2: Node failure recovery cycle.

*Research School of Computer Science, College of Engineering & Computer Science, Australian National University, Canberra, ACT 0200, Australia

Detection and Identification of Process Failure

- When node failure occurs, not only the processes but also their data lost.
- How to detect and identify the process failures is a critical issue.
- Recently a new version of Open *Message Passing Interface* (Open MPI) library [2] do this.

Process Recovery

- Recovering a failed node involves recovering processes of that node.
- Recovery is done by creating new processes on a reserved node with **same process id**.
- Open MPI do this.

Data Recovery

There are two kinds of data recovery: exact or inexact.

- One way of recovery is to re-run the processes, but in supercomputing context it takes too long time (Fig. 3).
- One example is Google's famous MapReduce [3] implementation, Hadoop [4].
- Other than re-running the processes, we could use existing data to calculate an inexact recovered data (Fig. 4).
- Should keep the approximation error minimum.
- *Robust combination* [5] technique for the solution of 2D *Partial Differential Equations* (PDEs) is such an example.

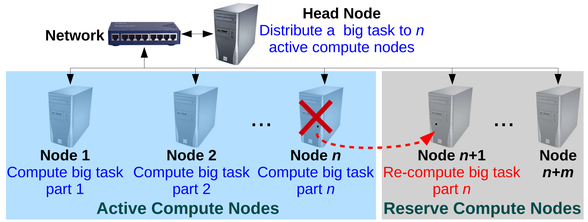

Figure 3: Exact recovery of a node failure.

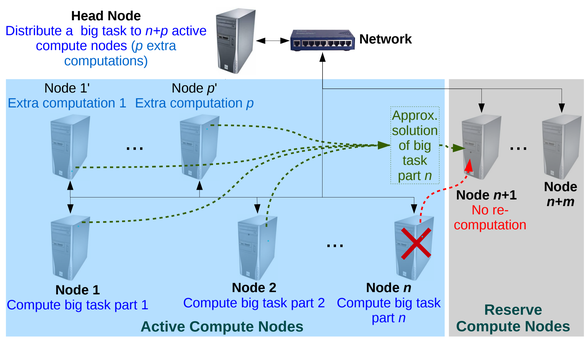

Figure 4: Inexact recovery of a node failure.

Experimental Results

Experimental Setup

Experiments were conducted on a 432-core OPL cluster with 36 dual-socket nodes each consisting of two 6-core Intel(R) Xeon(R) CPU (X5670 @ 2.93GHz), 24.0 GB of RAM, and with InfiniBand QDR connection blades in each chassis.

Total Failures

- Probability of node failure is exponential.
- Number of node failure increases with the increase of system size and failure rate, λ (Fig. 5).

Overall Execution Time

- Inexact recovery takes less time than exact recovery (Fig. 6).
- Recovery is more expensive on large system with high failure rate, λ.

Approximation Error

- Error is calculated from the analytical solution of 2D PDE solver (Fig. 7).
- 3 nodes failure contributes 5 times more approximation error for inexact recovery.
- This is fairly acceptable.

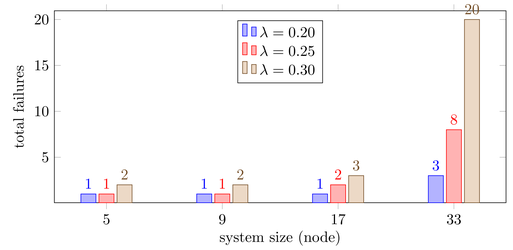

Figure 5: Effect of system size and failure rate on the total number of failures.

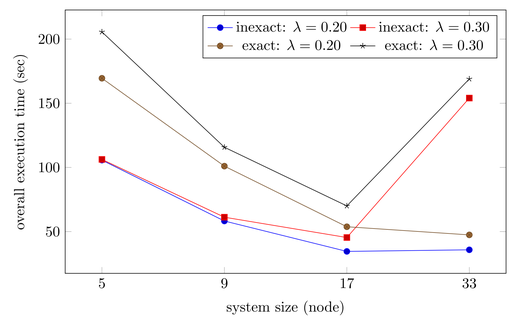

Figure 6: Overall performance of a 2D PDE solver (solver's execution time + process recovery time + data recovery time) which experiences a node failure.

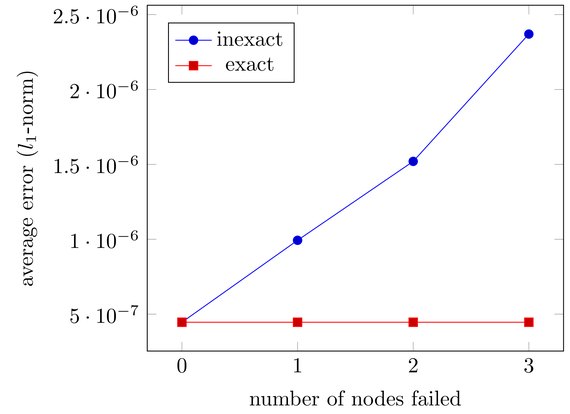

Figure 7: Average approximation errors of a 2D PDE solver.

References

[1] G. Gibson, B. Schroeder, and J. Digney, "Failure tolerance in petascale computers," *Software Enabling Technologies for Petascale Science*, vol. 3, no. 4, November 2007, pp. 4–10.

[2] Fault Tolerance Working Group, "Run-through stabilization interfaces and semantics." [Online]. Available: svn.mpi-forum.org/trac/mpi-forum-web/wiki/ft/run_through_stabilization

[3] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *Proceedings of the 6th Symposium on Opearting Systems Design & Implementation (OSDI 2004)*, vol. 6. Berkeley, CA, USA: USENIX Association, 2004, pp. 1–13. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251254.1251264

[4] Hadoop, http://hadoop.apache.org/.

[5] B. Harding and M. Hegland, "A robust combination technique," *ANZIAM Journal*, vol. 54, no. 0, 2013. [Online]. Available: http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/6321

Australian Government
Australian Research Council

FUJITSU

Australian National University

ANU College of Engineering & Computer Science