Exploring Techniques for Scalable Fault-Tolerant Software on Large Scale Parallel Cluster Supercomputers

Robert Pamely, supervised by Mr. Simon McIntosh-Smith

University of Bristol, Department of Computer Science

BACKGROUND

Cluster supercomputers are becoming larger. Software must deal with more frequent faults.

- A cluster supercomputer is a large collection of individual computers (nodes) that communicate via a fast local interconnect to work together as a single, unified computing resource. Nodes can run many processes. The fastest supercomputer today has ~18k nodes [1].
- There are predictions that by 2018 clusters will be 1 billion-way parallel, comprised of between O(100k) and O(1M) nodes [2].
- Predictions for mean time to interrupt (MTTI) in these systems vary, with a conservative estimate being O(1 day), but as systems increase in size the MTTI will decrease [2]. Therefore the software that runs on these systems must be fault tolerant.

1. CURRENT TECHNIQUES

There are many fault tolerance techniques. This project focuses on the following two:

Checkpoint-Restart

Copies application state to reliable storage at intervals. On failure, any work completed since the last backup is discarded. Research shows this approach is inviable on very large systems [3].

Virtual Processors

Back up both messages and data for each process to a “buddy” node. On failure, restart process backup on a spare node. This approach is faster and decentralised because only part of the program is restarted. [4]

2. RESEARCH OBJECTIVES

- Implement a scalable and decentralised fault tolerant message passing system that allows for fast recovery from faults, and does not rely on a pool of spare hardware.
- Outperform the efficiency of checkpoint-restart fault recovery.
- Investigate techniques for fault tolerance on cluster supercomputers and their advantages.

3. PROJECT TECHNICAL SOLUTION

Extends the “buddy” virtual processor protocol and removes the requirement of a pool of spare nodes. The new solution requires at most one spare node and is designed to affect as few nodes as possible when a failure occurs. By keeping the number of active nodes even, nodes will always have a “buddy” to backup to. Inactive nodes stay dormant but online in case of failure. See diagram below.

4. BUSINESS STRATEGY

- Develop and license scalable fault tolerance software utilities to end users. Will also advise clients on how to incorporate fault tolerance techniques into their software.
- Initially will run pilot study of software with a small number of companies to receive early feedback.

5. PROJECT STATUS & OPEN QUESTIONS

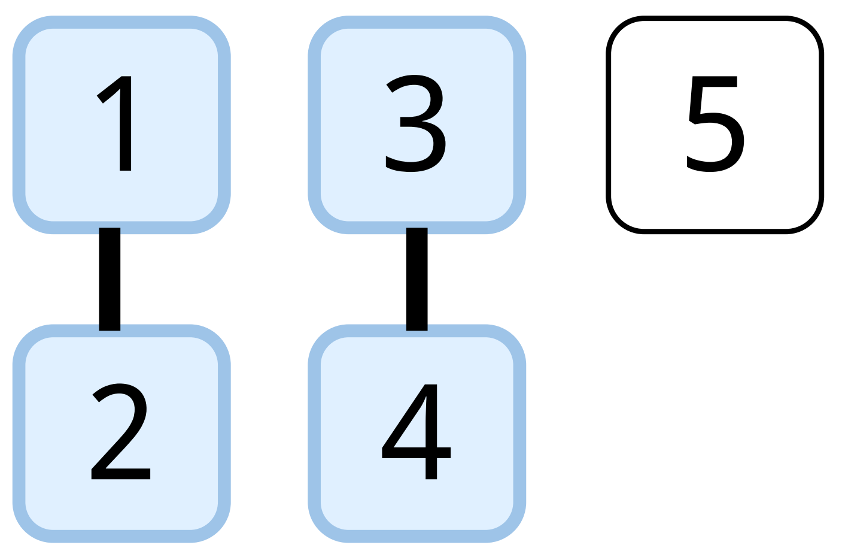
Currently implementing prototype solution to prove concept and evaluate approach. There are several open questions:

- How do processes know whether an unresponsive recipient of a message has failed or has completed its execution?
- How do the inactive nodes know when execution has finished so they can terminate?
- How should nodes keep track of which nodes are inactive?
- How do you detect unrecoverable failures? If you need to fall back to checkpoint-restart, how is this coordinated?

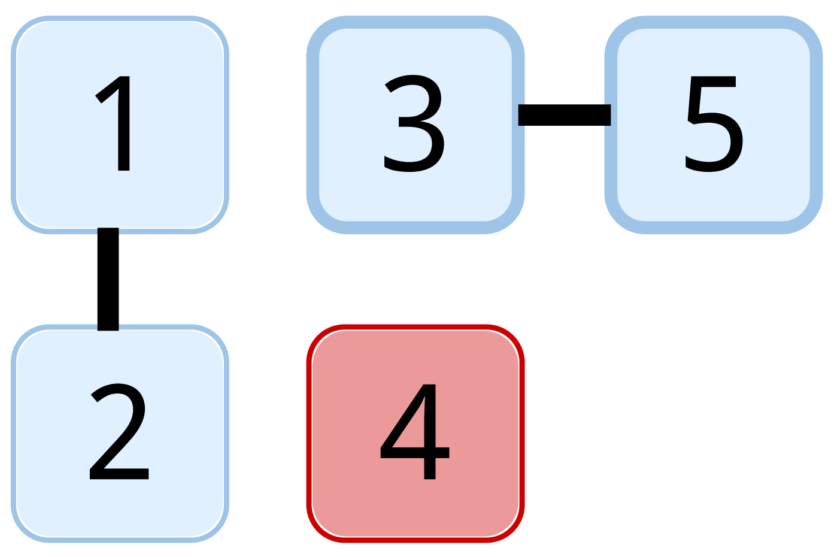
6. FUTURE RESEARCH

- Trade off between the frequency of backing up and the amount of data loss on failure. Can this trade off be analysed to maximise the efficiency of the fault tolerance algorithm?
- How can workload be redistributed between nodes both after failures occur and during normal execution in a fault tolerant system?

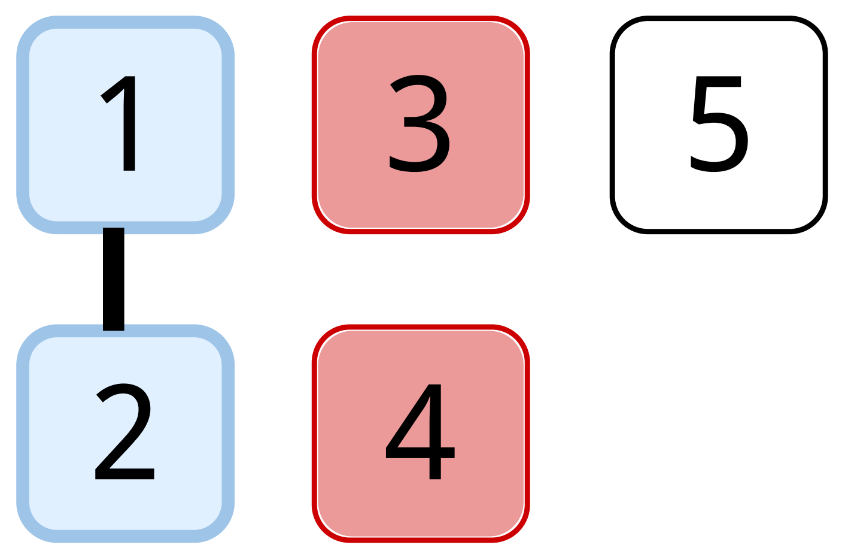
Connected nodes are buddies. 5 does not have a buddy and so is inactive.



Node 4 dies. Its buddy moves the data from 4 onto 5, which is now no longer inactive.



Node 3 dies. Node 5 redistributes all data controlled by 3 and 5 to all alive nodes. 5 becomes inactive.



Time

References:

[1] Titan supercomputer. http://www.olcf.ornl.gov/titan/ March 2012. Declared fastest by Top500.Org.
[2] European exascale Software Initiative (2011); Final report on roadmap and recommendations development.
[3] R. Oldfield, S. Arunagiri, P. Teller, S. Seelam, M. Varela, R. Riesen, and P. Roth. (2007); Modeling the Impact of Checkpoints on Next-Generation Systems
[4] S. Chakravorty, C. Mendes, and L. Kale (2007); A fault tolerance protocol with fast fault recovery.