Fault Tolerance for PCJ

Michał Szynkiewicz

Faculty of Mathematics and Computer Science Nicolaus Copernicus University, Toruń, Poland

Abstract. Nowadays, distributed computations are often run on large amounts of nodes for a significant amount of time. Having a 24 hour long computation on 1000 nodes means almost 3 years of computations. As calculated in [1], with 6 month mean time between failures, it gives less than 1% chance of success.
PCJ library is a high-performance parallel computing library for Java implementing PGAS model.
This paper describes changes introduced in PCJ to provide basic fault tolerance and fault tolerance strategies that are planned to be implemented in future.

1 Introduction

Before the changes described in this paper, any network or hardware failure resulted in hanging of a PCJ-based program.

The changes introduce *Ignore failure* fault tolerance policy. This is a minimum-overhead, no-checkpoint strategy.

For programs that do not require results from all nodes to finish, like Monte Carlo randomized algorithms, the strategy is enough to finish calculations properly. For others, the strategy provides a useful base on which the programmer may build a dedicated solution (e.g. replaying failed nodes work on other node).

2 PCJ library overview

PCJ library is an implementation of PGAS model. The smallest unit of computation is called a *thread*. Each *thread* has its own local address space and shared address space. The owner thread accesses local and global address spaces variables as usual Java fields. Following is a description of PCJ features. For more details please see [3].

2.1 Synchronization

Threads synchronization in PCJ is realized by `barrier`. Barrier can be made either across all threads or two threads. In first case each task has to call `barrier` method.

2.2 Shared memory

Fields that should be put into shared address space have to be annotated in `@Shared` annotation. Aformentioned fields are accessible from other threads via following methods:

- `get(threadId, variableName)`, `getFutureObject(threadId, variableName)` - read the value of given variable in a synchronous and asynchronous fashion
- `put(threadId, variableName, value)` - asynchronously update variable value on given node.
- `broadcast(variableName, value)` - asynchronously update variable value on all nodes.

`waitFor(variableName)` and `monitor(variableName)` can be used to lock the current thread until other thread updates given variable on this node.

3 Literature overview

- czy to dobry tytuł? TODO!!

4 API

The main concern for the API changes provided by the implementation is not to break

4.1 Impact on API

5 Node failure handling implementation

The main elements of implementation are: monitoring nodes for failure and adjusting configuration after node failure.

5.1 Node monitoring

Monitoring works in two ways::

- The immortal *node 0* is monitoring other nodes by sending PING message and waiting for PONG. If, for some node, sending PING fails, or the node does not answer with PONG for a long (configurable) time, it is assumed failed. Please see Fig. 1 Node monitoring.
- If *node A* encounters a communication error when trying to communicate with *node B* (e.g. if updating a variable value on *node B* fails), *node A* informs *node 0* that *node B* has failed.

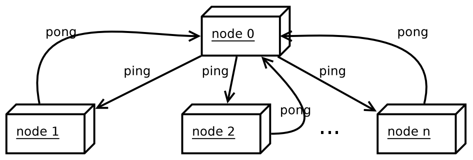

Fig. 1. Node monitoring

5.2 Reconfiguration

On node failure *node 0* sends a proper reconfiguration message to all nodes. Depending on the type of selected strategy there are two...

6 Performance overhead

7 Resilient storage implementation

8 Future plans

9 Conclusion

References

1. Leslie Lamport, Dave Cunningham, Dave Grove, Ben Herta, Arun Iyengar, Kiyokuni Kawachiya, Hiroki Murata, Vijay Saraswat, Mikio Takeuchi, Olivier Tardieu *Resilient X10. Efficient failure-aware programming*, 19th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Orlando, Florida, 2014, Lecture
2. Marek Nowicki, Piotr Bała *PCJ - New Approach for Parallel Computations in Java* Applied Parallel and Scientific Computing, 11th International Conference, PARA 2012, Helsinki, Finland, June 10-13, 2012, Revised Selected Papers
3. Marek Nowicki, Piotr Bała *PCJ. Parallel Computing in Java*, 2013, [Online]. Available: `http://pcj.icm.edu.pl/c/document_library/get_file?uuid=e8b5a416-644d-43d0-a19c-74e14555ef96&groupId=43801` [2015, April 14]