

Steps of Consistency for Fault-tolerant MPI

João Gabriel Silva
University of Coimbra, Portugal

The world of parallel programming is overwhelmingly determined by performance. Provisions for other requirements like resource management, security or fault-tolerance, must be designed so that they do not compromise execution speed in any way. A proposal is made here for a very small extension of version 2 of the Message Passing Interface (MPI) standard to support fault-tolerance (FT). The added mechanisms are controllable by the user, following the philosophy of MPI of exposing all mechanisms that can impact performance. Fault-tolerant MPI programs with very low overhead due to FT can be written. These extensions to the standard are needed because MPI currently has very limited provisions to handle faults, but that must change as its usage spreads to very large parallel machines and to non-dedicated clusters used for cycle-stealing. The alternative of implementing transparent fault-tolerance imposes too many limitations on the user programs, is very difficult to implement and does not give the programmer enough control. Thus, although we agree that most of the work required by FT must be done by the MPI library, we require the programmer to identify the points in execution where the application data is consistent and, if he/she wants the library to checkpoint that data, to identify which variables should be saved. Since a parallel program becomes a sequence of steps where a consistent state of data is transformed in another consistent state, we name this model Steps of Consistency: MPI-SC. This is similar to the transaction model of database applications, where it has long been established that users must structure their applications as a sequence of transactions to enable fault-tolerance, concurrency control, persistence, and so on.

We propose three successive levels of fault-tolerance for the transition from the current non-fault-tolerant MPI to the MPI-SC model.

First is the Clean-Up level, where we must guarantee that no resource leaks exist when an MPI application aborts due to some error. Few existing implementations follow that model, surprisingly. This level is required for programs that restart the whole virtual machine on node loss.

Next comes the Partial Failure level that implements the error handling model of the MPI standard to the limit, such that when a process loss happens only the communication with that process is compromised, with all other processes being able to go on computing. Together with the dynamic process creation mechanism of MPI version 2, this enables adding new nodes to substitute for failed ones. Farm programs can be easily programmed at this level, since the master node just needs to distribute to a new node the task that the failed process was executing. This level enables continuation on process loss instead of restart, but still leaves a lot of work for the programmer, because faults happen asynchronously and it is hard to assess damage after a fault.

The last level is Steps of Consistency, where the state of the application after a fault is well defined, checkpoints can be saved by the library, and communicators reconstructed or discarded, eliminating the resource leaks of the previous model. This level is well adapted to regular decomposition codes, as are most simulation applications.