Fault Tolerant Computing on the Grid: What are My Options?

Jon B. Weissman
Division of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
(jon@cs.utsa.edu)

Extended Abstract

High-performance distributed computing across wide-area networks has become an active topic of research. Achieving large-scale distributed computing in a seamless manner introduces a number of difficult problems. This paper examines one of the most critical problems, fault tolerance.

We have examined fault tolerance options for a common class of high-performance parallel applications, single-program-multiple-data (SPMD). Performance models for two fault tolerance methods, *checkpoint-recovery* (CR) and *wide-area replication* (WR), have been developed. These models enable quantitative comparisons of the two methods as applied to SPMD applications.

Executing SPMD applications in a fault tolerant manner can be achieved by checkpointing or replication. For the purposes of a direct quantitative comparison, a simple checkpoint model is assumed in which each SPMD task saves its portion of the data domain on disk at a set of pre-determined iterations. We studied two configurations for checkpointing: a single NFS-mounted disk from a file server and a parallel array of locally attached disks. The dominant overhead for checkpoint-recovery is the cost of stopping the application and writing the checkpoints to disk. For wide-area replication, each application replica runs in a different site to avoid overloading the available computation and communication resources of a particular site.

The cost model is based on the following parameters: N (problem size), P (number of processors), I (number of iterations), k (checkpoint frequency), m (number of wide-area replicas or sites), λ (average rate of site failure in failures/minute), and β (desired reliability or probability of application success). A set of cost equations are expressed in terms of these parameters and several system-dependent constants (see below). We make the assumption that the failure rate is exponentially distributed as is commonly done and that it is identical for all sites. We have experimentally derived the constants for these cost functions in a local- and wide-area network testbed environment. We model two modes each for CR and WR. For CR, we experimented with local and NFS-mounted file systems. For WR, we experimented with remote sites in which application files (binaries and input files) were either pre-staged or required uploading at runtime.

We compare the methods as follows. Given P_f (the probability that the application does not fail), we first determine the optimal number of checkpoints to perform. The optimal number of checkpoints k_{opt} balances the cost of checkpointing with the cost of re-executing old iterations in the event that the application fails and needs to be restarted from the last stored checkpoint. This derivation is omitted. For WR, P_f plays a different role. Given P_f, we can determine the number of replicas *m* that achieve a desired level of reliability β, where β is the probability that at least one replica finishes. The probability that at least one replica finishes is $1-P_f^m$ assuming independent site failures each with probability P_f. Given a desired value of β, we can easily compute *m* and T_{WR}, and compare it with T_{CR} for a given problem instance.

We performed an evaluation of CR and WR to answer the following questions. Given a problem instance (N, P, I), network characteristics (local/NFS, m, P_f), and a desired degree of reliability (β) will CR or WR perform best? What is the impact of uploading on the suitability of WR? We answer these questions for two specific experimental environments using a canonical SPMD application (STEN) that solves Poisson's equation on a NxN grid using an iterative method. We have two versions of STEN: (1) a CR version with user-level checkpoint code inserted and (2) a WR version that uses an experimental testbed containing Internet sites. We compare the methods for different failure and reliability parameters. For P_f, we vary λ to be 1 site failure per the following time intervals: 6 months, 1 month, 1 week, 1 day, half day, and 1 hour (corresponding to failure rates = l1, l2, ..., l6 respectively on the graphs). We vary β to be .999, .999999, .99999999 (corresponding to B = 1, 2, 3, respectively on the graphs). We obtained the T_{CT} values off-line, and we show results

T_{CT} (N, P) = I * T_c(N, P)

- T_c is the average time per iteration and includes computation and communication

T_{CP} (N, P, NFS) = $T_{NFS\_latency}$ + (N^2/P)T_{NFS_bw}

- The latency and bandwidth terms include disk and network overhead; (N^2/P) is the per processor data size

T_{CP} (N, P, local) = $T_{disk\_latency}$ + (N^2/P)T_{disk_bw}

- Local checkpointing only involves disk overhead; (N^2/P) is the per processor data size

T_{CR} (N, P, disk-config) = k_{opt} * T_{CP} (N, P, disk-config)

- k_{opt} is the optimal number of checkpoints

T_{WR}(N, m) = $T_{WR\_scheduling}$ (N, m) + T_{WR_upload} (m, bsize)

- WR consists of scheduling costs and file upload costs (which depends on binary and input file sizes)

$T_{WR\_scheduling}$ (N, m) = $T_{WR\_sched\_latency}$ + m ($T_{WR\_sched\_overhead}$)

- Scheduling overhead consists of a base cost and a per site cost

T_{WR_upload} (m, bsize) = $T_{WR\_upload\_latency}$ + m ($T_{WR\_upload\_overhead}$)

- Upload consists of a base cost and a per site file transfer cost

P_f (N, P, λ) = $1 - e^{-\lambda T_{CT}}$

- Exponential failure distribution - this gives the probability the application will NOT fail (in a site)

for P=4 and I=1000. We show the total overhead under either method (T_{WR} vs. T_{CR}) on the y-axis as a function of varying failure rate for all graphs. The first row compares WR without uploading to CR for local and nfs respectively. CR with a local disk configuration exhibits slightly better performance for small problems (N=128, 256, and 512; 128 is not shown), while WR is a clear winner for large problems (N=1024). WR performance does not vary much as β increases which suggests that a very high rate of reliability appears to be affordable. The results also indicate that for small problems, WR and CR offer similar performance. When a NFS-mounted disk configuration is used, WR is clearly superior for virtually all problem sizes (N=256, 512, and 1024; 1024 is off the graph). The flat-line indicates that the checkpoints are so expensive, that only a single checkpoint is affordable.

The second row compares WR with uploading to CR for local and nfs respectively. For both configurations, CR is clearly superior to WR due to high cost of uploading the STEN binaries. However as compared to the NFS configuration, WR is competitive for the largest problem (N=1024) for β = .999.

The results obtained for the stencil application indicate that both methods appear to be appropriate under different conditions: CR is generally cheaper for small problems providing fast local disks are available, while WR is generally cheaper for bigger problems provided binaries are pre-staged.