533 E1

Adding fault-tolerance and load-balancing to JavaSpaces

Tuplespaces or object based distributed shared memory provides a simple flexible paradigm for building distributed systems. JavaSpaces is a Java based object-oriented tuplespace with support for storing Java objects including code and polymorphic tuple type matching.

Early tuplespace implementations suffered from limitations inherent in their centralised design; namely a single point of failure and a scalability bottleneck. These problems can be overcome by tuplespace distribution using replication and partitioning techniques. Although the JavaSpaces specification leaves open the possibility of a distributed implementation, all available implementations are centralised in design.

We have developed JavaSpaces-WRAP (JavaSpaces With Replication And Partitioning); a client-side class library that emulates a distributed JavaSpace by federating several JavaSpace services. The library consists of two primary classes; one provides fault-tolerance via operation

replication, while the other provides scalability and load balancing via deterministic partitioning of tuples. JavaSpaces-WRAP follows the decorator design pattern to layer functionality atop a basic centralised space, maintaining most of the syntax and semantics of the centralised JavaSpace. Clients access the distributed JavaSpace transparently by instantiating the appropriate wrapper and invoking standard JavaSpace operations.

Tests to determine fault-tolerance provided by the replication wrapper were carried out by modifying an online auction application to use the replication wrapper. Replica failure was simulated by killing one of the replicas while the application is being used. The fault-tolerance provided by the replication wrapper enables the application to continue functioning normally until all replicas fail. Scalability and load balancing behaviour was tested by using a fractal generator application modified to use the partitioning wrapper. The master-worker pattern of the fractal generator uses objects of various types for communication. These objects are distributed among partitions according to their hash value to achieve load-balancing. We are in the process of obtaining data to calculate the speedup provided by partitioning under varying loads.