Chime: A Windows NT based parallel processing system

Shantanu Sardesai
Tandem Computers Incorporated
Partha Dasgupta
Arizona State University

Chime: A Windows NT based parallel processing system[1]

Shantanu Sardesai
Tandem Computers Incorporated
shantanu.sardesai@tandem.com

Partha Dasgupta
Arizona State University
partha@asu.edu

1. Introduction

Shared memory multiprocessors are the best platform for writing parallel programs. These platforms support a variety of parallel processing languages (such as CC++ which provide programmer-friendly constructs for expressing shared data, parallelism, synchronization and so on. However the cost and lack of scalability and upgradability of shared memory multiprocessor machines, make them a less than perfect platform.

Distributed Shared Memory (DSM) has been promoted as the solution that makes a network of computers look like a shared memory machine. This approach is supposedly more natural than the message passing method used in PVM and MPI.

However, most programmers find this is not the case. The shared memory in DSM systems do not have the same access and sharing semantics as shared memory in shared memory multi-processors. For example, only a designated part of the process address space is shared, linguistic notions of global and local variables do not work intuitively, parallel functions cannot be nested and so on.

Chime is the *first* system that provides a true shared memory multiprocessor environment on a network of machines. It achieves this by implementing the CC++ language (shared memory) on a distributed system. In addition to shared memory, parallelism and synchronization features of CC++, Chime also provides fault-tolerance and load balancing.

2. Chime Features

Chime addresses most of the problems in a simple, clean and efficient manner by providing a multiprocessor-like shared memory programming model on network of workstations, along with automatic fault-tolerance and load balancing. Some of the salient features of the Chime system are:

1. Complete implementation of the shared memory part of the CC++ language.
2. Support for nested parallelism; i.e. a parallel task can spawn more parallel tasks.
3. Consistent memory model, i.e. the global memory is shared and all descendants (which execute in parallel) share the local memory of a parent task.
4. Machines may join the computation at any point in time (speeding up the computation) or leave or crash at any point without affecting the progress (slowdowns will occur).
5. Faster machines do more work than slower machines, and the load of the machines can be varied dynamically (load balancing).

In fact, there is very little overhead associated with these features, over the cost of providing DSM. This is a documented feature that Chime shares with its predecessor Calypso.

3. Chime Architecture

A program written in CC+ is preprocessed to convert it to C++. Then it is linked with the Chime runtime library and a single executable file is generated. This executable is executed on a network of machines (or workstations). One of the workstations is designated as the *manager* and the rest as *workers*. All run the same executable.

A program starts on the manager. When the program reaches a parallel construct, parallel tasks are generated and are allocated by the manager to the waiting workers. During the execution of the parallel step, the manager does scheduling and allocation of parallel tasks, as well as memory management.

The manager and worker are multithreaded, the programmer written code is executed by one thread and the other thread handles all the runtime functions. The runtime functions include servicing DSM requests, inter-task synchronization, cactus stacks for proper stack sharing and scheduling that handles fault tolerance and load balancing. Chime is implemented on Windows NT.

The performance tests of Chine show that it it performs well for a variety of parallel programs. The nested parallelism and synchronization support however adds considerable overhead in a distributed system. The Chime system can be downloaded from http://milan.eas.asu.edu.

[1] This research is partially supported by grants from DARPA/Rome Labs, Intel Corporation and NSF.