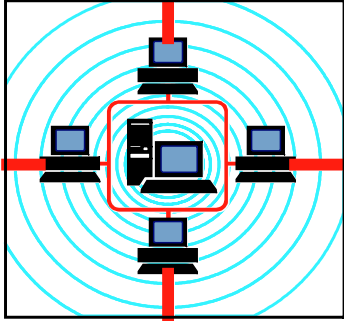


# CHIME

Reliable Shared Memory Computing  
on Networks



## Shared Memory Parallel Processing

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

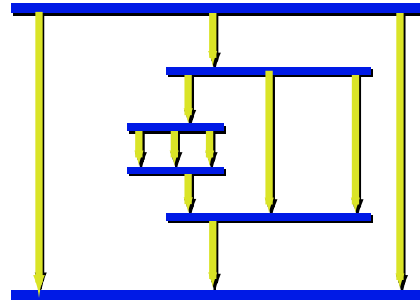
Shared memory multiprocessors are the best platform for writing parallel programs. These platforms support a variety of parallel processing languages (such as C++) which provide programmer-friendly constructs for expressing shared data, parallelism, and synchronization. However the cost and lack of scalability 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. 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 multiprocessors. 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 provides true shared memory semantics in a distributed system

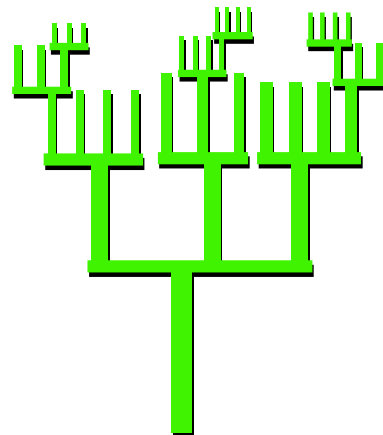
## Chime Features

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 C++ 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. This is achieved via a *distributed cactus stack*.



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 balancing).

In fact, there is very little overhead associated with these features, over the cost of providing DSM.

### 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, intertask synchronization, cactus stacks for proper stack sharing and scheduling that handles fault tolerance

and load balancing. Chime is implemented on Windows NT.

### Using Chime

Chime is now available for download. The programs written for Chime uses CC++ for at least the parallel part of the program. CC++ is quite easy to learn as it adds two keywords to C++ for expressing parallelism.

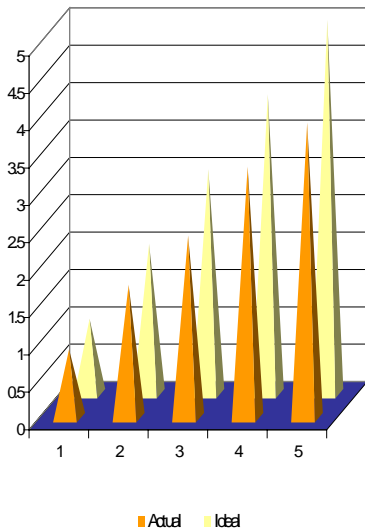
The Chime runtime system runs the program on a set of networked computers and a visual front-end shows the progress of the program as it runs. The performance of parallel program under Chime is quite competitive to other systems, including PVM (which is the best performer). The nested parallelism and synchronization support however adds considerable overhead in a distributed system.

The Chime system can be downloaded from the Milan Project pages at <http://milan.eas.asu.edu>.

**MILAN is a joint project of  
New York University  
Arizona State University**

**Zvi M. Kedem**  
New York University  
+1 212 998 3101 (phone)  
+1 212 477 3265 (fax)  
[kedem@cs.nyu.edu](mailto:kedem@cs.nyu.edu)  
<http://www.cs.nyu.edu>

**Partha Dasgupta**  
Arizona State University  
+1 602 965 5583 (phone)  
+1 602 965 2751 (fax)  
[partha@asu.edu](mailto:partha@asu.edu)  
<http://milan.eas.asu.edu>



Speedup of Chime