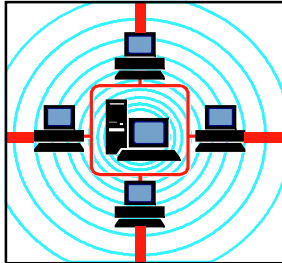


The MILAN Project

Metacomputing in Large Asynchronous Networks

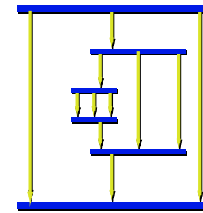
CHIME



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 part) on a distributed system.

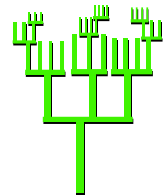
Thus Chime provides true multiprocessor semantics for computations and memory in a network of machines (as opposed to DSM systems, that only provide a shared memory segment).

The salient features of Chime include nested parallelism, inter-thread synchronization, proper scoping of shared variables (including stack variables), automatic load balancing and fault-tolerance. The system is well suited for a large variety of parallel applications, and the software is freely available for download.

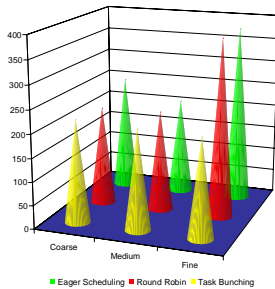


Nested parallelism allows parallel routines to call other parallel routines. Fault tolerance is preserved.

Distributed Cactus Stacks allow variables to be shared by concurrent threads in a consistent manner.



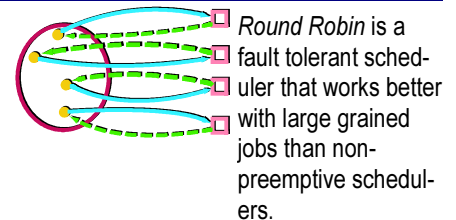
Preemptive Scheduling



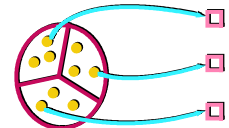
All multitasking operating systems (including multiprocessors) use preemptive scheduling. But, preemptive scheduling in distributed systems is rare, if not nonexistent.

Preemptive scheduling is very important for proper task management in distributed systems. Some long running jobs need to be migrated from one machine to another. Barrier synchronized jobs of differing lengths have to be balanced amongst machines of unequal speeds. Also, when granularities of jobs are variable, granularity management is important. *Granularity is increased by bunching and decreased via preemption.*

We have developed two effective preemptive schedulers: **Fault-Tolerant Round Robin** and **Preemptive Task Bunching**, which perform well in distributed systems.



Preemptive Task Bunching is well suited for a wide range of granularities and has better performance than most

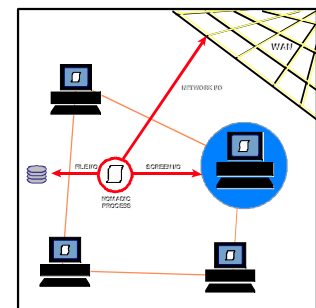


Nomadic Processes

A **Nomadic Process** is a regular process, running any general-purpose computation. However, the process runs on top of a middleware layer, which enables the process to move from machine to machine, upon external scheduling decisions.

A Nomadic Process may move, even though it has already established connections to files, I/O, networks, keyboards, screens and so on. It may also be a multi-threaded process.

We expect to be able to make any program (legacy, binary only) run as Nomadic Processes. Currently, we need source access and use a preprocessor, but we are working on more sophisticated techniques using DLL's.



Sponsors: DARPA, NSF, Intel, Microsoft

Partha Dasgupta
Arizona State University
partha@asu.edu
<http://milan.eas.asu.edu>

Zvi M. Kedem
New York University
kedem@cs.nyu.edu
<http://www.cs.nyu.edu>

