References

[1] Bergmark D. et al, On the Performance of the FPS T-Series Hypercube, Second Conference on Hypercube Multiprocessors, ISBN 0-89871-209-2, SIAM (1986)

[2] C. Witby-Strevens, *The Transputer*, Proc. 12th International Symposium on Computer Architecture, ACM (1985), pp. 292-300

[3] Ritchie, D.M. and Thompson, K., The UNIX time sharing system, Communications of the ACM, July, 1974, pp 365-375

[4] D. Cheriton et al, *Thoth, a Portable Real-Time Operating System*, Communications of the ACM, Feb. 1979, pp. 105-115

er specifies a communication link instead of a destination node and does not incur the overhead of routing.

Network Layer

Most Trollius services are built on top of the network layer, because it is this layer that unites the various nodes into one uniform distributed system. Using one set of network calls, any process can send a message to any other process on any node. Forwarding through intervening nodes is transparent to the functionality (but not the performance) of application programs.

Transport Layer

The transport layer provides reliable end to end transmission of network messages. Messages are guaranteed not to congest, block or corrupt the communication subsystem. Essentially this layer prevents the sending party from accidently introducing a ridiculous number of messages to the operating system. A message will not leave the source node until the destination process is ready to receive it.

Buffering/Non-blocking

A variable amount of message buffering is available at the datalink, network and transport layers. Also available are non-blocking (polling) calls which return an error if the operation would have blocked.

Broadcasting

A simple machine dependent broadcast process is used to efficiently send a message to all nodes.

Virtual Circuits

A virtual circuit is an end to end connection between two processes. After a message has passed through the general purpose routing of the communication network, the programmer may optionally specify that the path to the destination be ‘‘left open’’, such that all intervening system software will expect another message and forward it to the same destination. This can continue indefinitely until the programmer specifies that the path should ‘‘close’’. Trollius’s version of virtual circuits yields excellent communication speedup, but at the same time can be very dangerous if not used correctly.

Remote File Access

Processes are provided with a UNIX compatible remote file system whose requests are ultimately sent as messages to nodes that actually have secondary storage. Many of the file system calls available under UNIX are available remotely, including open , close , read , write and lseek . Standard output is directed to the user’s terminal.

Memory Management

For nodes running Trollius natively (inside the parallel machine), a simple memory management scheme is employed. Programmers can allocate blocks of memory through the familiar malloc interface, but should be warned that the underlying implementation is not the same as the UNIX implementation. Variations of malloc are available to suit the requirements of specific machines (eg: the transputer and VPU of the T-Series).

C Library

A reasonably complete C library is available which includes the following major packages as well as many other useful routines.

- standard I/O, an augmentation of the remote file system
- transputer specific functions not available from C or Fortran
- string manipulation package

A math library, separate from the C library, provides many routines vital to scientific processing.

Command Interface

Many of the Trollius services available to programmers through libraries are packaged into command programs for use in operating the machine. Trollius commands are no different than any UNIX utility and are executed from the shell, though note that the Trollius system must already be executing on the UNIX node. The major commands are described below.

Loadgo. Load an executable file into any node from the file system of any node (or all of a given class of nodes) and create it as a process. Options may be passed on the command line to the new process.

- *Doom*. Essentially kill an existing process on any node.
- *State*. Find the state of one or more user processes on any node.
- *Buffer*. Set the number of network message buffers in a given node. Also inquire on the state of buffers in a given node (shows which buffers have messages and where they are going).
- *Sweep*. Sweep any node clean of full buffers. Sweep is typically used immediately after doom to remove stray messages from the system.
- *Tping*. Ping a node to see if it is alive.

Trollius Operating System

Gregory D. Burns
Andrew K. Pfiffer
David L. Fielding
Alison A. Brown

Cornell Theory Center
Ithaca, NY

Trollius Operating System

Trollius is an operating system designed for a loosely coupled, message passing ensemble architecture, such as the hypercube class of machines. It was originally implemented for the FPS T-Series[1], a transputer [2] based machine and has been ported to other transputer based machines. Trollius presents a uniform programming environment that extends out of the parallel processors to encompass the front end computer and and personal workstations accessible over a network. C and Fortran compilers are available along with a suite of backend development tools including a transputer board simulator.

Trollius does not include any implicit support for parallel processing, but instead provides full explicit control of all facets of machine operation including reconfiguration of the operating system itself.

Key Features

- multitasking with message passing IPC on every node
- complete network subsystem to pass messages between arbitrary nodes
- remote file system
- programmable in C or Fortran
- familiar UNIX [3] command interface

System Architecture

Trollius is an operating system for a parallel machine, one or more directly attached UNIX† (4.3BSD) hosts and any other UNIX based machine capable of reaching one of the hosts over a network. Together, these machines form a complete supercomputing environment with the program development power of the UNIX operating system on the front end and the program execution power of the parallel machine on the back end.

Trollius's approach is to combine these varied machines into one system by building a uniform environment across the parallel machine and the participating UNIX based machines. Programs are source code portable throughout the system, eliminating the need to write host control programs with a substantially different set of tools and libraries.

Future references to a node , will mean not only the components of the parallel machine but also the attached front end computers and user workstations.

Multitasking

Each node is capable of creating and concurrently executing multiple processes (sometimes called tasks). Library routines are available to create, kill and inquire state on processes both locally and on any other node. Existing processes may be spawned to any node; the program is copied (if the node is remote) and created as a new process. This eliminates the overhead of loading the program from a file and has important connotations for load balancing.

Intraprocessor Communication

The kernel provides a blocking message passing service to its constituent processes [4]. Synchronization is via arbitrary numbers called events and not via process identifiers. The blocking send and receive requests are accompanied by non-blocking counterparts that essentially poll the status of an event and return an error if the requesting process would block. In addition, a type field is provided with each message which when used as a filter, allows processes to be more selective in their synchronization.

Interprocessor Communication

Using the same mode of synchronization (events and types/filters), processes may communicate with other processes running on any node. The functionality of ''network'' message passing, loosely resembles the OSI network model. Trollius provides datalink, network and transport services.

Datalink Layer

A datalink message travels only between nearest neighbours, but there are no restrictions on the number of sending or receiving processes. The us-

† UNIX is a trademark of AT&T

© ACM 1988 Proceedings of the Third Conference on Hypercube Concurrent Computers, and Applications (Mar. 88)