FM-DCS: An Implementation of Dynamic Coscheduling on a Workstation Cluster

Patrick Sobalvarro Scott Pakin Andrew Chien William Weihl

October 7, 1996

User-level messaging layers have made it possible to run parallel jobs that perform fine-grained communication on dedicated workstation clusters. If a workstation cluster is to be timeshared, however, performance suffers terribly — we have measured user-CPU-time slowdowns of as much as a factor of 50 over batch when running a barrier synchronization benchmark using spinning message receipt in a timeshared environment where 4 competing processes were present on each processing node.

The problem is uncoordinated scheduling, first documented by Ousterhout in [1]. His solution, traditional coscheduling, is difficult to implement on a workstation cluster because it requires global context switches across the cluster and placement of arriving jobs on different nodes. Sobalvarro and Weihl suggested instead the solution of *dynamic coscheduling* in [3]. In dynamic coscheduling, message arrivals are used to trigger scheduling decisions; when a message arrives at a node that is not running the process to which the message is addressed, the node preempts the running process and switches to the process to which the message is addressed if it would be fair to do so. The result is coordinated scheduling among *communicating* processes — the only processes that can actually benefit from coscheduling.

We have implemented dynamic coscheduling on a cluster of Sun workstations running Solaris 2.4 using Illinois Fast Messages [2] to communicate across a Myrinet switch. We call the resulting system FM-DCS.

Our results show that FM-DCS successfully coordinates scheduling of parallel jobs on our workstation cluster, resulting in efficient execution of parallel jobs in the presence of competing serial jobs. Our results hold for a variety of communication patterns — programs using request-reply communication, barrier synchronizations, and stencil communication all showed improved performance — and for a broad range of communication granularities.

FM-DCS is implemented without modification of the Solaris 2.4 scheduler, by instead manipulating its data structures from within a loadable device driver. A fairness mechanism is used to assure that jobs with fine-grained communication do not starve competitors of CPU cycles. Interrupts are not required on message arrival in the common case where the process to which the message is directed is curently scheduled; the network interface processor only interrupts the host in the uncommon case where a scheduling decision may be required. Overhead added by the scheduling mechanism is minimal, and the coordinated scheduling it achieves is sufficient to allow networks of workstations to run parallel jobs while timeshared with users running serial jobs.

References

[1] John K. Ousterhout. Scheduling techniques for concurrent systems. In *Proceedings of the 3rd International Conference on Distributed Computing Systems*, pages 22–30, October 1982.

[2] Scott Pakin, Mario Lauria, and Andrew Chien. High performance messaging on workstations: Illinois Fast Messages (FM) for Myrinet. In *Supercomputing*, December 1995. Available from `http://www-csag.cs.uiuc.edu/papers/myrinet-fm-sc95.ps`.

[3] Patrick G. Sobalvarro and William E. Weihl. Demand-based coscheduling of parallel jobs on multiprogrammed multiprocessors. In *Proceedings of the Parallel Job Scheduling Workshop at IPPS '95*, 1995. Available from `http://www.psg.lcs.mit.edu/~pgs/papers/jsw-for-springer.ps`. Also appears in Springer-Verlag Lecture Notes in Computer Science, Vol. 949.