IMPLEMENTATION OF CASHMERE

Michael L. Scott (speaker)
Wei Li
Sandhya Dwarkadas
Leonidas Kontothanassis*
Galen Hunt
Maged Michael
Robert Stets
Nikolaos Hardavellas
Wagner Meira
Alexandros Poulos
Michal Cierniak
Srinivasan Parthasarathy
Mohammed Zaki

University of Rochester
Computer Science Department
Rochester, NY 14627-0226
scott@cs.rochester.edu

* Digital Equipment Corporation
Cambridge Research Laboratory

The Cashmere project attempts to capture the "knee of the curve" in price-performance for shared-memory parallel computing: it exploits recent advances in local-area networks that provide low-latency, user-level access to remote locations in hardware, but implements coherence in software. The project has recently moved from simulation to implementation, using a 32-processor Alpha-server cluster (eight 4-processor nodes) on DEC's Memory Channel network. This talk will focus on the Cashmere implementation and on early experience as a Memory Channel field test site.

The Cashmere coherence protocol is characterized by

(1) multi-writer lazy release consistency,
(2) directories to keep track of who is sharing pages, and
(3) update merging via write-through to a unique main-memory copy of each page.

Like many software coherent systems, Cashmere uses virtual memory protection at the granularity of pages to catch changes to the set of sharing processors, and to perform invalidations at synchronization release points. We have adapted the protocol to several different hardware variants. For the Memory Channel, which does not support loads from remote locations, we replicate pages to all nodes in which a processor needs access.

Simulation studies, reported at HPCA-1, Supercomputing '95, and HPCA-2, indicate that on an idealized remote-access network, the Cashmere protocol can achieve dramatic performance improvements over twin-and-diff-based distributed shared memory, and can in fact approach the performance of full hardware coherence. Two practical issues limit the extent to which we can duplicate these results in our prototype implementation. First, cross-sectional bandwidth in the first-generation Memory Channel, while impressive at approx. 100MB/sec, is far short of what can be achieved in tightly-coupled multiprocessors such as the T3E or Paragon. Second, OS overheads, which could in theory be relatively modest, are substantial in OSF-1. In particular, the need to create a separate "memory object" for every remote-mapped page places stresses on the VM system unanticipated by its designers.

Work on Cashmere is proceeding on several fronts:

(1) We are extending the coherence protocol to encompass a three-level system consisting of hardware coherence within SMP nodes, hybrid hardware/software coherence within LANs with remote memory access, and software-only coherence on message-passing networks. Longer-term, we expect to encompass heterogeneous processor architectures as well.
(2) We have developed a low-overhead fault-tolerance mechanism for twin-and-diff-based coherence, and are attempting to adapt it to Cashmere.
(3) We are integrating Cashmere with Rochester's Carnival system for performance prediction and analysis. We plan to use feedback from performance measurements to tune the coherence protocol, automatically, to the needs of individual applications.
(4) We are integrating run-time, "behavior-driven" coherence with compile-time static analysis. We expect this integration to be increasingly important in our work.

This research is supported by NSF grants numbers CCR--93--19445, CCR--94--09120, and CDA--94--01142; by ARPA contract number F19628--94--C--0057, subcontract 3531427; and by an external research grant from Digital Equipment Corporation.