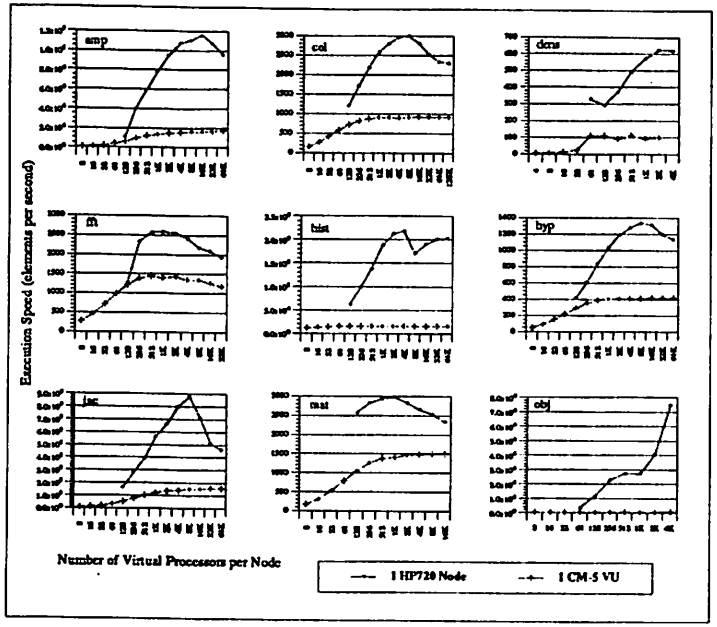
OS Support for NOWs

Larry Peterson
(llp@cs.arizona.edu)
(http://www.cs.arizona.edu/xkernel/www)

Department of Computer Science
University of Arizona
Tucson, AZ 85721

Experiences with Cluster-C*

- **What We've Built**
  - Cluster-based environment for running C* programs
  - 8 HP720 (50MHz) workstations connected by FDDI
  - Test suite of 9 image understanding programs
  - Coarse-grained, trace-based simulator
- **Experimental Results**
  - Measured: 8-node cluster outperforms 32-node (128 Vector Unit) CM-5 for some problems; always within an order-of-magnitude of CM-5
  - Simulated: 16-node Alpha-based cluster connected by GIGAswitch is 0.5 to 5 times faster than 128-VU CM-5 across test suite
- **Observations**
  - Network Latency: we had more latency-hiding machinery than we had latency to hide
  - Network Bandwidth: 3MBytes/sec bandwidth into each node sufficient for HP720s
  - Cache Size: Defined knee of execution-speed curves
  - Available Memory: With 32MBytes-per-node, cluster scales near-linearly to 16 nodes; with 1GByte-per-node, same cluster scales to 256 nodes



Networking: What We Know How to Do

- **Latency**

  Application Device Channels (ADCs) give application programs direct, but restricted access to the network interface; no kernel involvement on the send/receive path.

- **Throughput**

  Delivered 516Mbps memory-to-memory and 440Mbps cache-to-cache between a pair of Alpha workstations; fast buffers (fbufs) can be used to deliver same rate to application programs.

Memory: The Next Hard Problem

- **Bloated Operating Systems**
  - IRIX requires 32MB
  - OSF/1 requires 32MB
  - Solaris requires 32MB (SunOS version n implies 2^n MB)
- **Memory Not That Cheap**
  - A MB here, a MB there, pretty soon it adds up to real $
  - Can add only so much memory (4-6 slots)
- **Ineffective Caches**
  - High rate of I-Cache stalls in OS code
  - Poor D-Cache efficiency for network data
  - Large applications make NO effective use of the cache
- **Inefficent Memory Use by Application**
  - Trade memory for simplicity

The Scout Operating System

- **Software-Specialization**

  Networked systems will need to be specialized in software, and hence, Scout is designed to be configurable: a given instance contains exactly the functionality required by the system for which it is built.

- **Communication-Centric Design**

  The concept of a path is fundamental to a communication-oriented system, and as a consequence, it is made an explicit abstraction in Scout. The path abstraction provides a focal point for addressing resource allocation and enabling effective optimizations.

- **Managing System Entropy**

  Scout will provide performance that scales with processor performance and is predictable: improvements to components of the system will lead to the expected improvement to the overall system.

- **Compiler Support**

  Scout will leverage compiler technology for two purposes: to help put the OS on the processor performance curve, and to simplify the process of constructing the OS.