Performance on the Berkeley NOW with Virtual Networks

David Culler

Tom Anderson

Dave Patterson

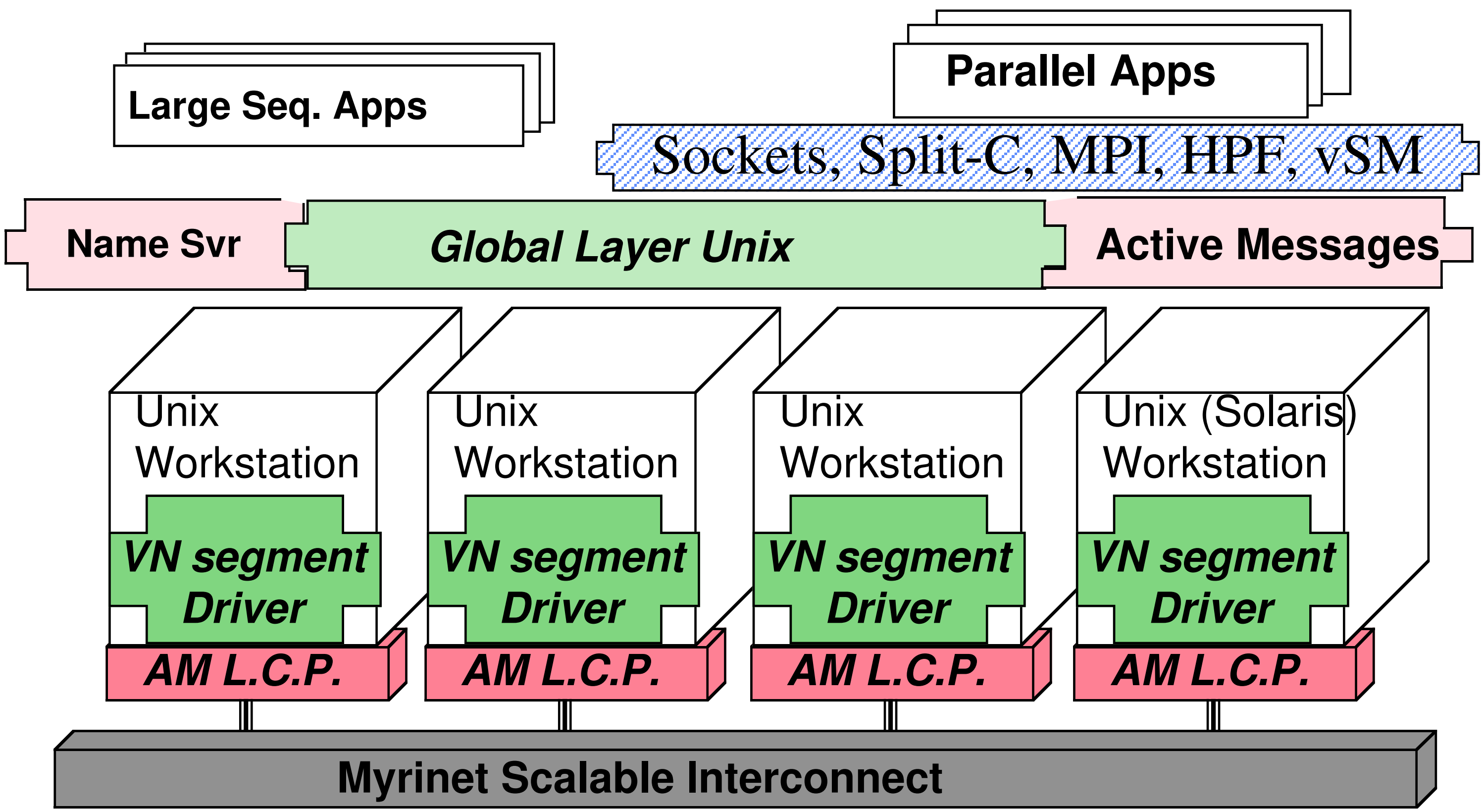
http://now.cs.berkeley.edu/

NOW Ultrasparc Cluster

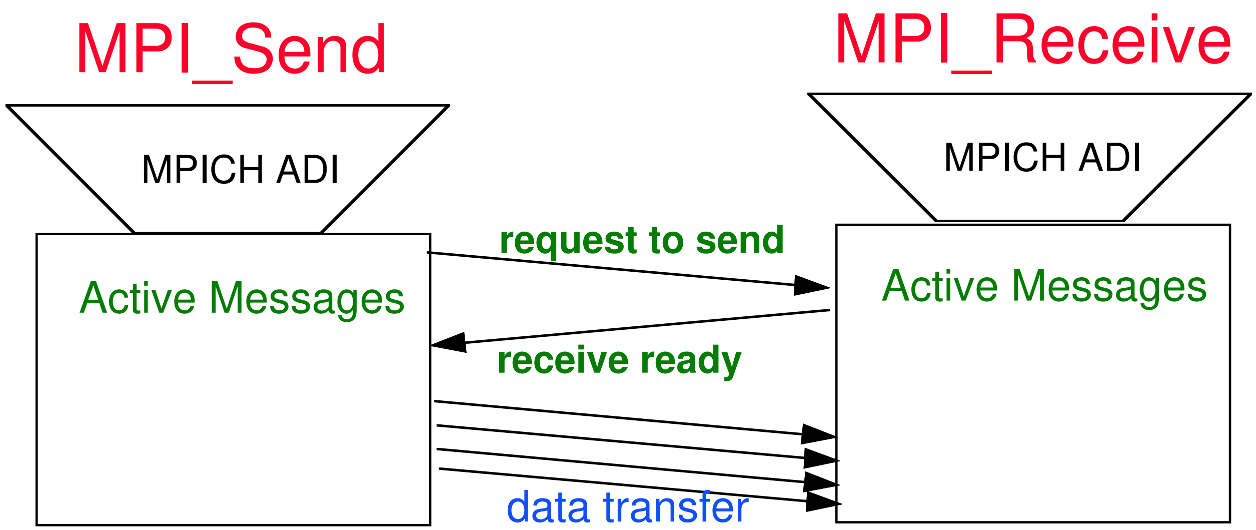
Recent Status

- 100 UltraSparc Cluster Fully Operational
  - stress tested with users across campus
- Spectrum of prog. models on Active Msgs
  - Split-C, MPI, HPF, SVM
- AM_2 with Virtual Networks in full use
- Scalability of T3D with better node perf. on NAS
- 10 Gflops LINPACK on 100 proc
  - 110 MFLOPS on one
- World Record disk-to-disk sort
  - 8.4 GBytes in a minute
- Implicit cosheduling working
- CLUMPS multiprotocol Active Messages

NOW Software Components

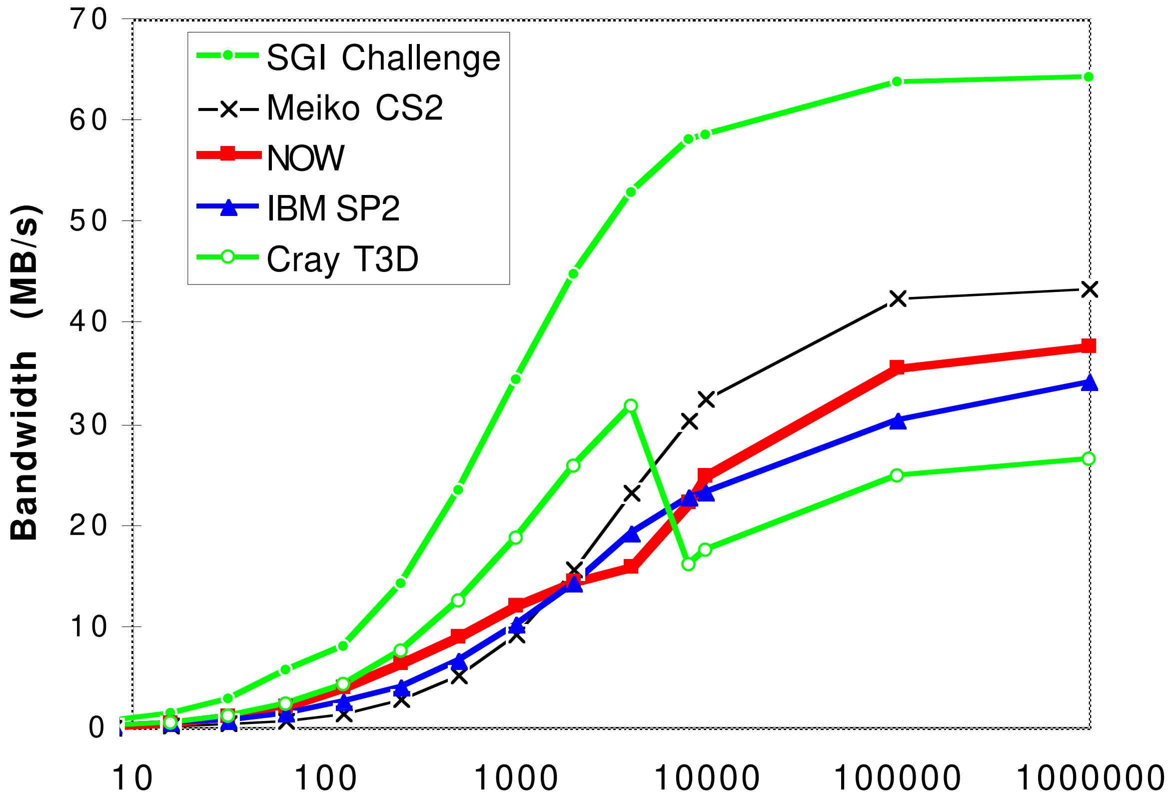


MPI over Active Messages

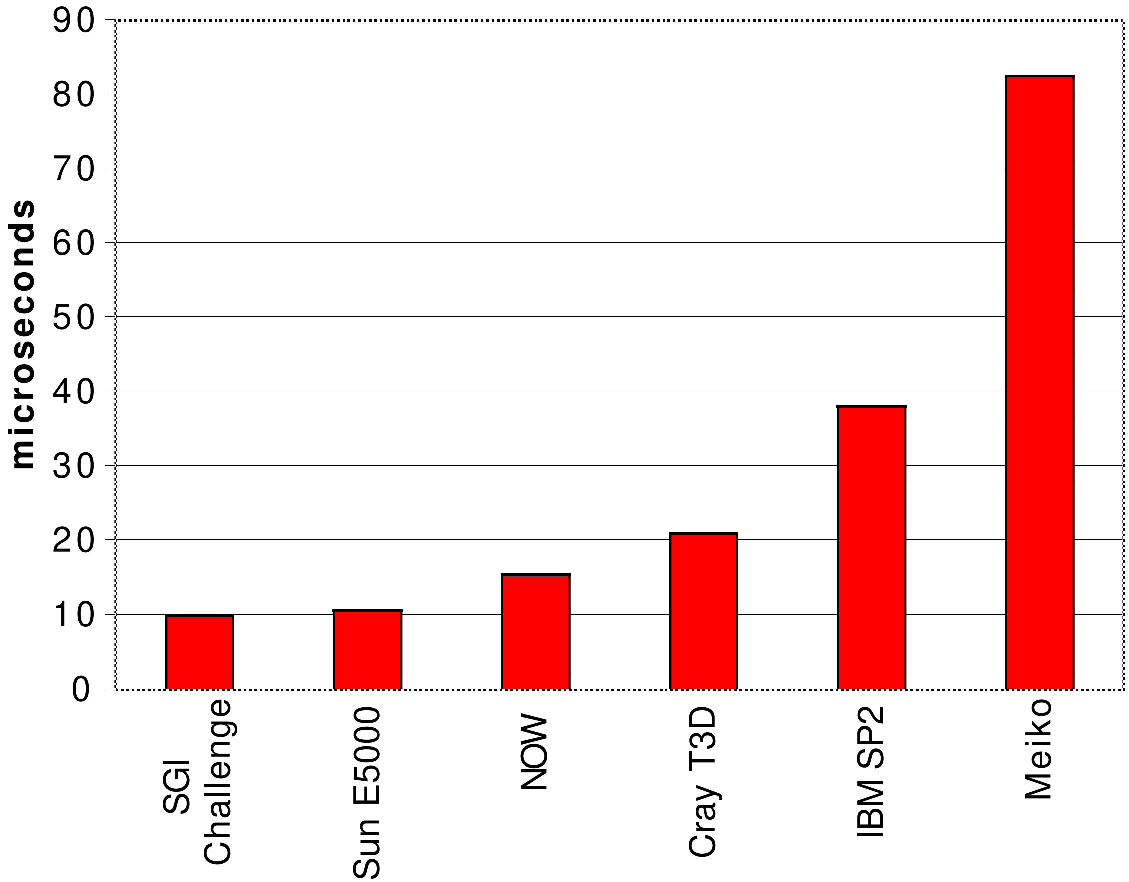


- **MPI sychronization built from Active Message Protocol**

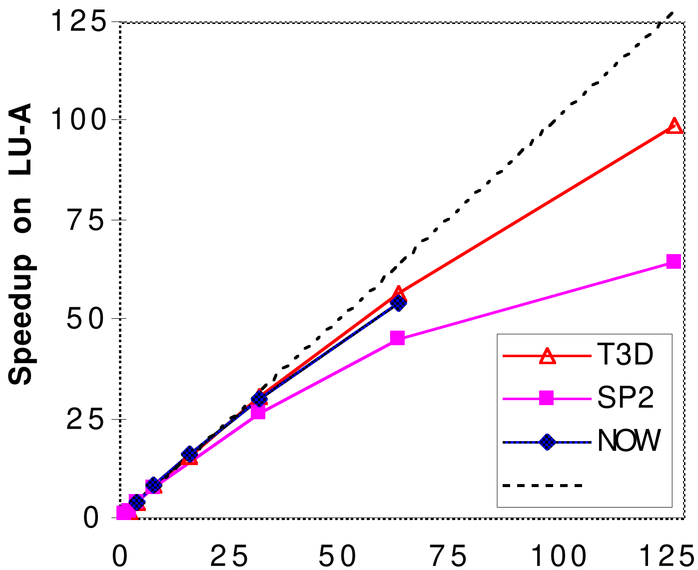
MPI over AM: ping-pong bandwidth

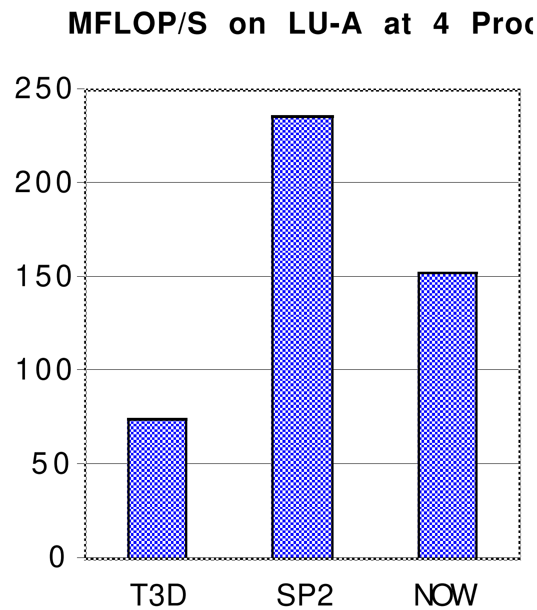


MPI over AM: start-up



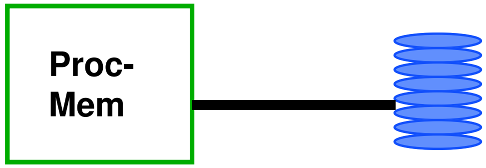
NAS Parallel Benchmarks - LU



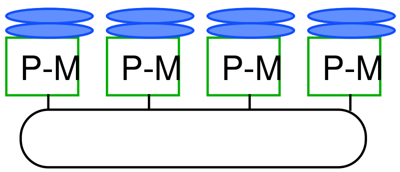


Exercising Cluster File I/O

- Traditional I/O system

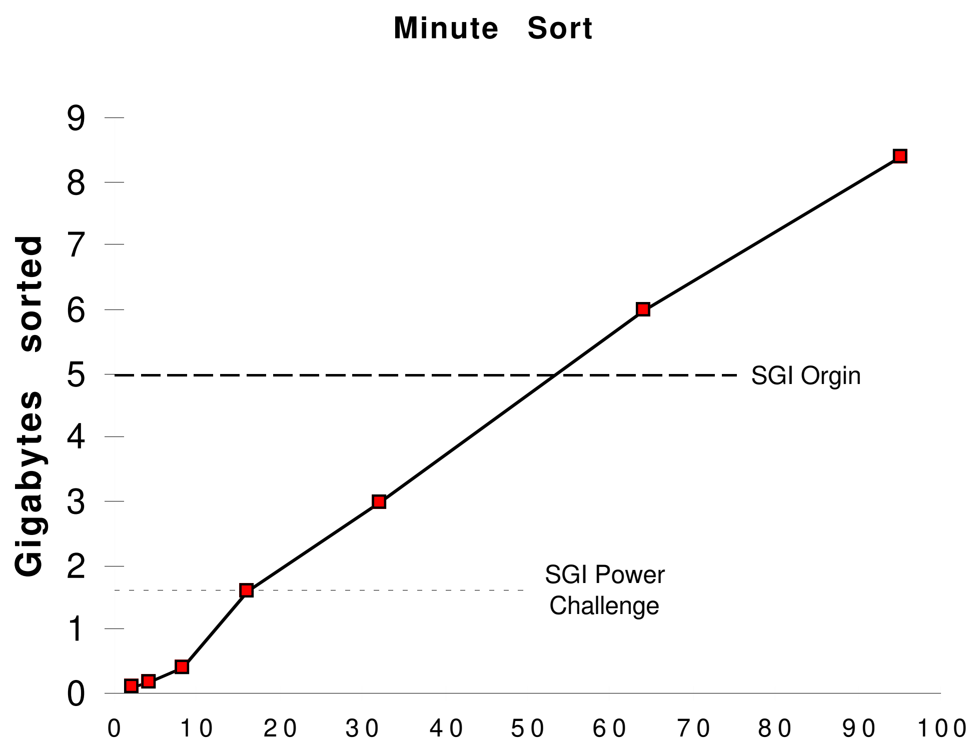


- NOW I/O system



- Benchmark Problem: sort large number of 100 byte records with 10 byte keys
  - start on disk, end on disk
  - accessible as files (use the file system)
  - Datamation sort: 1 million records
  - Minute sort: quantity in a minute

New World Record Disk-to-Disk Sort



- **on-going architectural study, including CLUMPS**

Per-Node Performance Isolation

- Stripe data across multiple disks so transfer time is balanced
- Test memory performance to determine available buffer space
  - use 2-pass sort of data will not fit
- Use controlled paging instead of file read/write
  - MMAP and MADVISE to get sequential read-ahead
- Overlap disk read and data distribution
- Use threads to get sequential write-behind

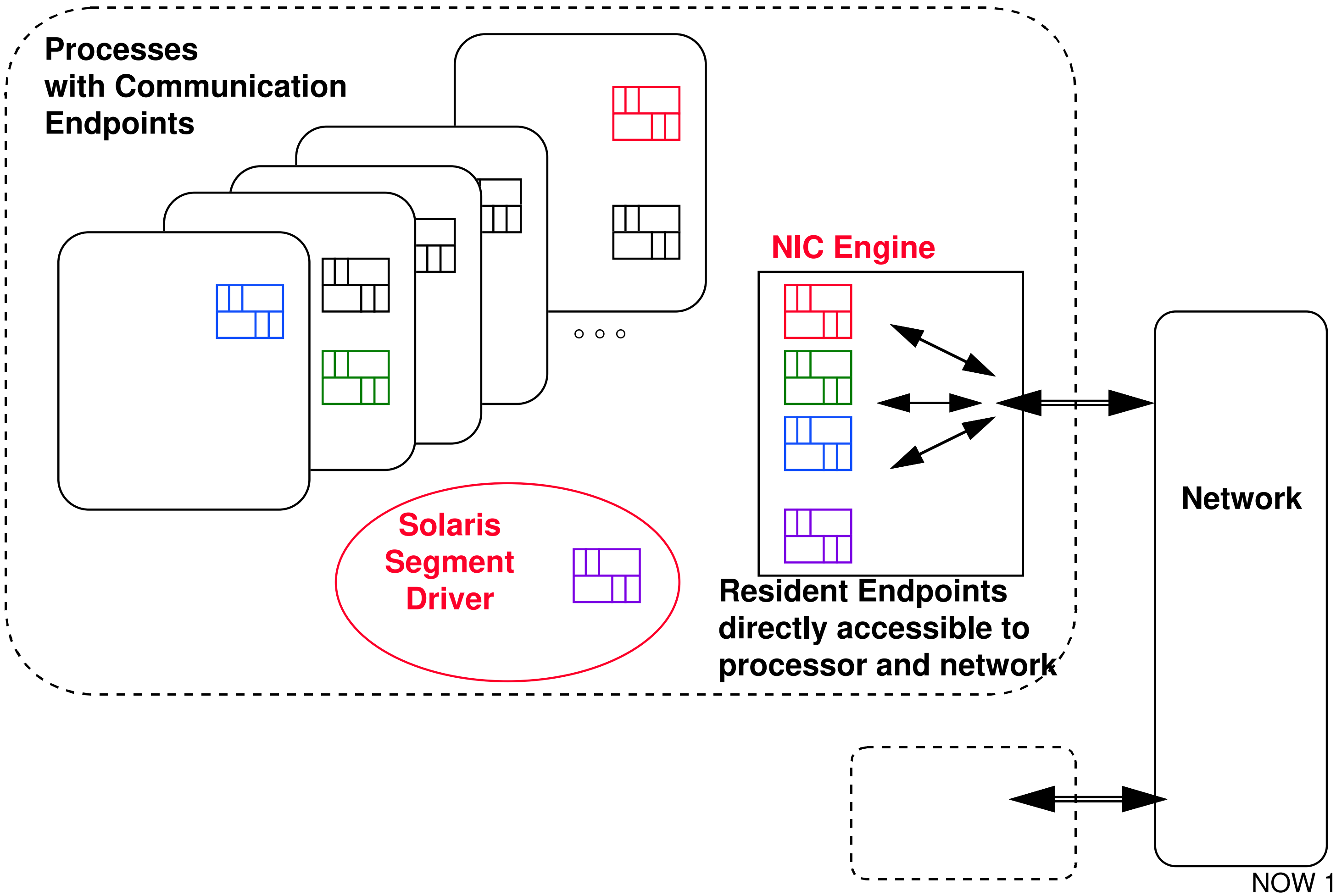
\+ fast communication

=> full O.S. capability is essential

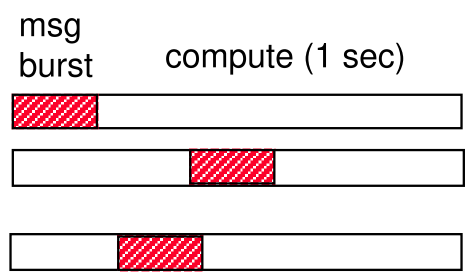
Virtual Networks

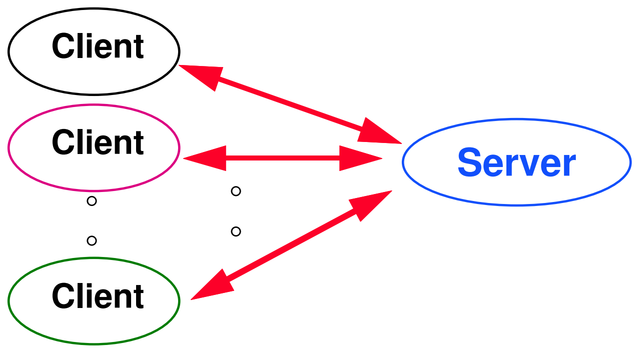
- **Low latency, high bandwidth => direct hardware access to the network**
- **At any time, only a small fraction of the potential applications are really communicating.**
- **Communication within an application is very bursty**
- **=> Provide each application with its own virtual network (s)**
- **Bind virtual networks on demand to physical communication resources**

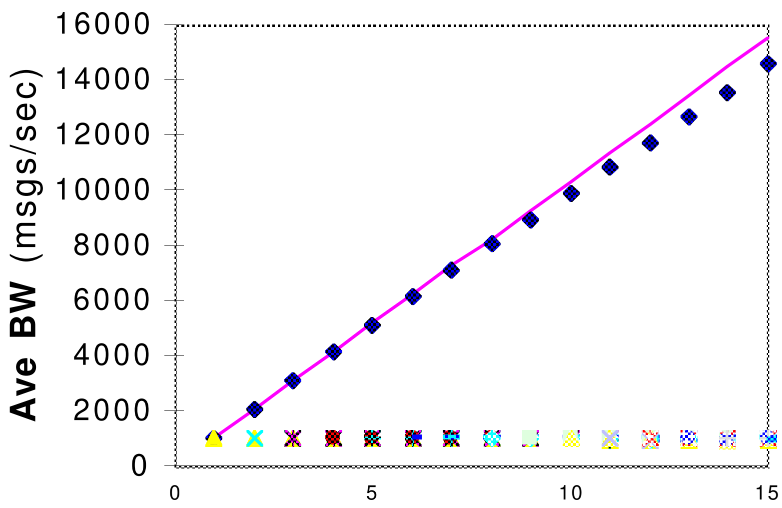
Virtual Networks meet Virtual Memory

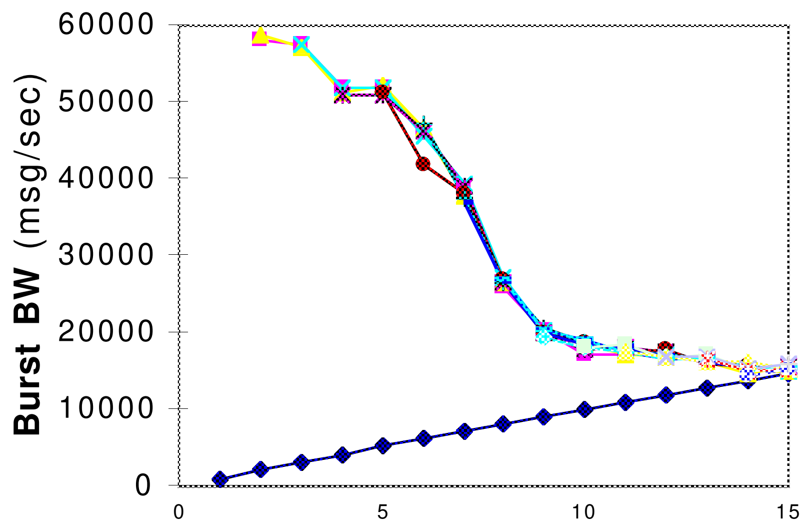


Service to Multiple Virtual Networks









What does this mean for core OS?

- What is the communication working set like?
- When should end-points be made resident?
  - msg send, msg use?
  - msg arrival?
- How does this interact with process scheduling?
  - within a node
  - across the cluster?

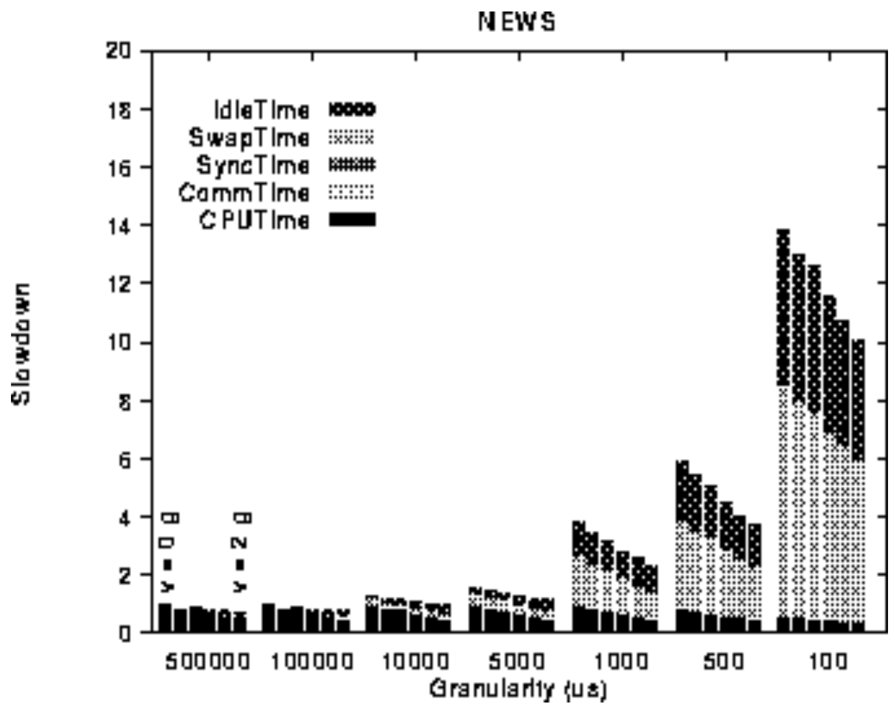
Example: implicit co-scheduling

- **Conventional Local Schedulers do not understand how to run parallel programs in parallel.**
- **Can result in 10 - 100 fold slowdown compared to (ideal) co-scheduling**
- **Co-scheduling (gang scheduling) is bad for:**
  - **mixed workloads**
  - **poorly balanced, infrequently communicating pgms**

=> use operations naturally occurring in the program to coordinate local scheduling

- **communication**
- **synchronization**

Local scheduling vs. Ideal Coscheduling



3 parallel processes

32 processors per process

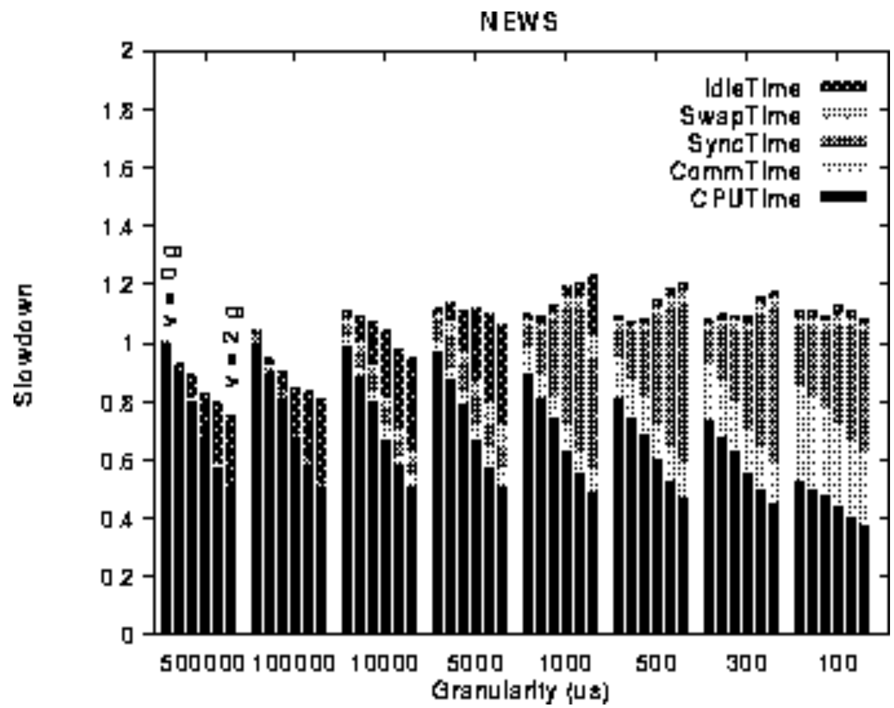
10 µs comm.

100 µs context switch

load imbalance:

0, 0.25. 0.5, 1, 2 x granularity

With Implicit Scheduling



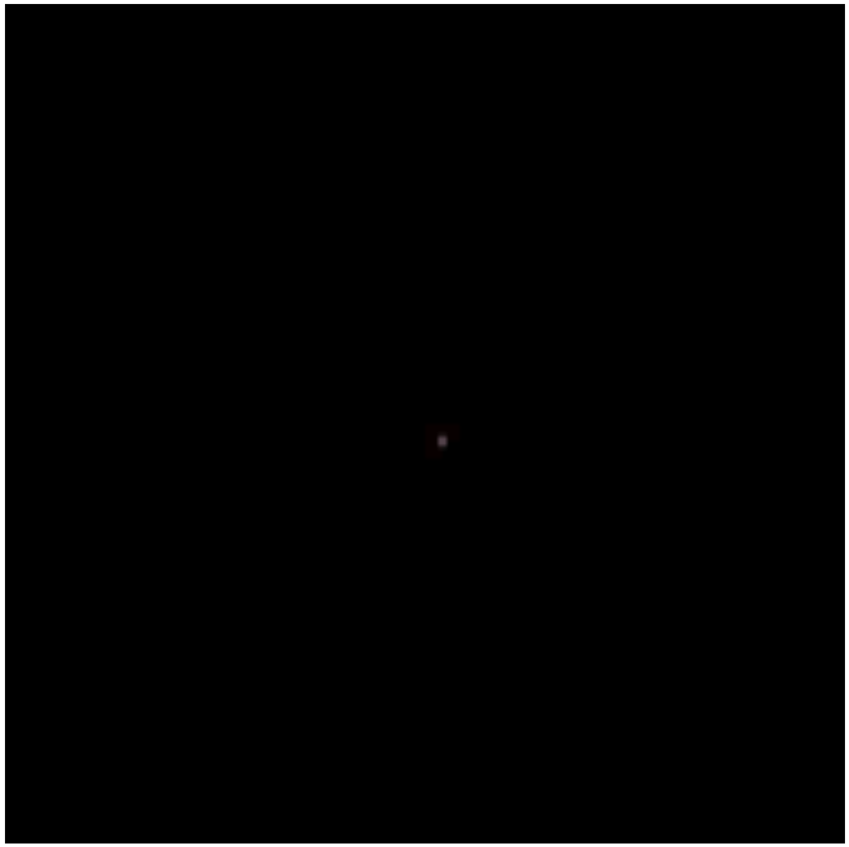
Measured Performance with Implicit Coscheduling

Privileged communication daemons

- **Virtualization provides protection boundary**
- **Privileged programs to have special access to hardware resources**

- **Example: network mapper daemon**
  - **probes the network using absolute routes to discover physical topology**
  - **derives set of deadlock-free routes and distributes them to NIC**
- **Most of its messages go nowhere**
  - **stress the Active Message error model**
  - **return-to-sender**

Automatic Mapping of NOW network



Mapping Scalability