The NOW Project

Remzi H. Arpaci-Dusseau
and the NOW Team

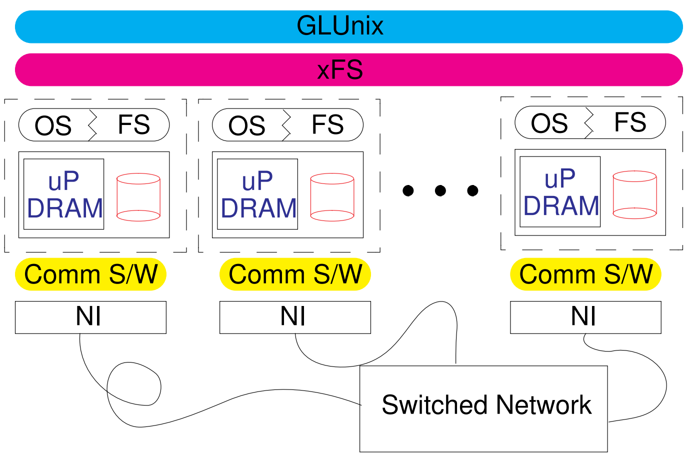
Computer Science Division
University of California, Berkeley

Outline

- NOW Project: Brief Overview
  - Project Vision
  - Research Topics
- NOW Case Study: NOW-Sort
  - Sorting: What and Why?
  - NOW-Sort: Theme and Variations
  - Criticisms and Conclusions

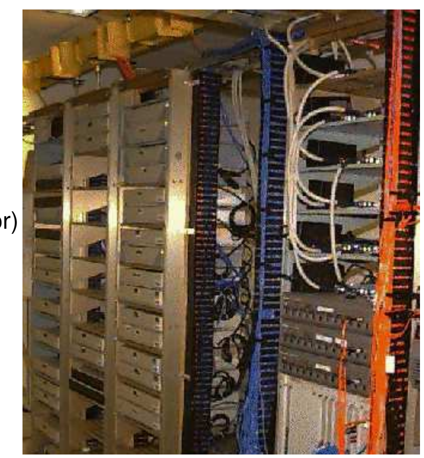
The NOW Vision

- NOW Goal: Unifying Resources of a Workstation Cluster



NOW Hardware

- The Workstations
  - 40 SPARC 10s/20s
  - 100+ UltraSPARCs
- The Network
  - Myrinet Switch (Crossbar, 180 MB/s)
  - Myrinet Card (Embedded Processor)

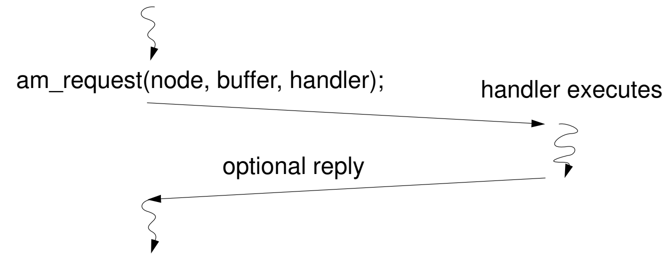


The NOW Umbrella

- High-Speed Communication
  - Active Messages 1.0 and 2.0
- Operating Systems
  - Glunix and Implicit Scheduling
- File Systems
  - xFS and Tertiary Disk
- The World Wide Web
  - WebOS/WebFS and Smart Clients
- Applications
  - Examples: Inktomi and NOW-Sort

High Speed Communication

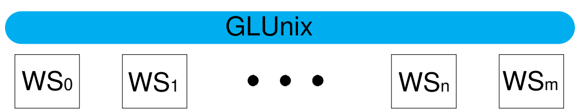
- Active Messages: Lightweight RPC = Low Overhead, High BW



- Active Messages 1.0
  - Straight-Forward Derivative Of MPP Versions (CM-5, etc.)
- Active Messages 2.0
  - Designed For Multi-programmed, Multi-user Environment
  - Virtual Endpoints Allow Arbitrary Numbers Of Concurrent Users

GLUnix: An OS For NOW

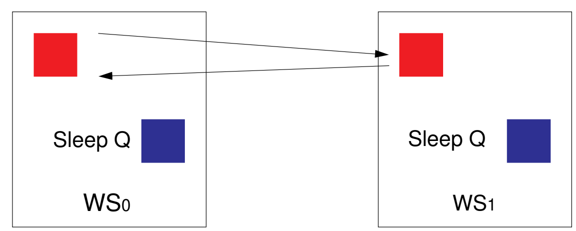
- Objective: Extend Unix Paradigm To Distributed and Parallel Environment



- Aspects Of GLUnix:
  - Remote Execution
    - Users Can Utilize All Processors In System
  - Job Control
    - User Can Suspend, Kill Jobs
  - Scheduling
    - GLUnix Coordinates Parallel/Sequential Jobs Across Machines
  - Resource Monitoring
    - GLUnix Tracks Load On All Machines For Load Balancing

Implicit Scheduling

- Problem: Centralized Scheduling Not Fault Tolerant or Scalable
- Solution: Let Each Workstation Decide Independently
  - Won't That Lead To Complete Chaos?
  - NO! Use Hints From Applications To Make Smart Decisions!



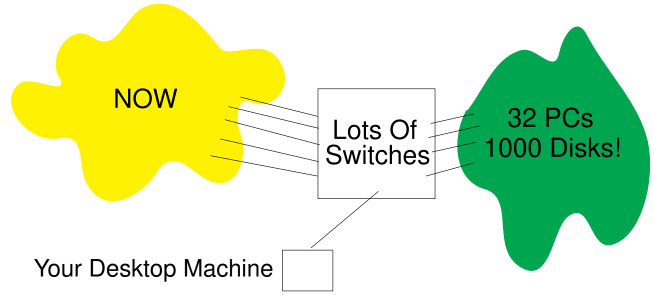
  - Thus, If Applications Are Communicating, Keep Them Co-Scheduled

xFS: A File System For NOW

- Key: Serverless File System
  - Any Workstation Might Have Your Data
- Synergy Of Previous Good Ideas Taken To Extreme
  - Log-Structured File System
  - RAID (Striping + Redundancy)
- "Cooperative Caching"
  - Allows Workstation To Get Cached Block From Other Workstations
  - Memory Of Cluster Used As Large Global File Cache

Tertiary Disk

- Alternate Disk Service For NOWs



- Use PCs As Smart Disk Controllers
  - Cost Effective Mass Storage

NOW And The Web

- WebOS
  - Distributed Computational Environment
  - World-Wide Interface to NOWs
- WebFS
  - Based On URL Name Space
  - Global File System For The World
- Smart Clients
  - Better World-Wide Access To Scalable Services
  - Intelligence In The Client, Not The Server

NOW Applications

- Scalable Web Services
  - Example: Inktomi
    - Fast, Scalable Index For The Web
    - Originally Built On Active Messages 1.0, Cluster Of 4 Workstations
- Scientific Parallel Programs
  - MPP-ish Nature Of NOW
- Batch Workloads
  - CS 252 Simulations
- Interactive Programs
  - Editors, Browsers, Etc.

NOW II: The Future

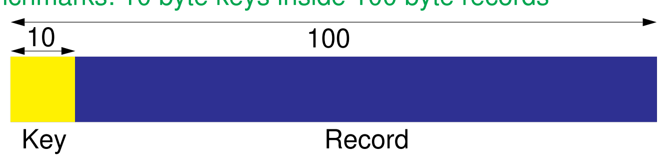
- Clusters Of SMPs
  - 4 8-Way SMPs Today
- PCs Instead Of Workstations
- Motivated Graduate Students

Outline

- *NOW Project: Brief Overview*
  - *Project Vision*
  - *Research Topics*
- NOW Case Study: NOW-Sort
  - Sorting: What and Why?
  - NOW-Sort: Theme and Variations
  - Criticisms and Conclusions

Sorting Background

- Why Sorting?
  - Important to Database Community
  - Memory and I/O Intensive
  - Tough Competition
- Benchmarks: 10 byte keys inside 100 byte records



  - Datamation (1985): How long to sort 1 million keys?
    Problem: Not I/O Benchmark Anymore
  - MinuteSort (1994): How much can you sort in a minute?

More Sorting Background

- Terminology:
  - One-Pass Sort: Data Can Fit in Main Memory
  - Two-Pass Sort: Data Can't Fit in Main Memory
- The Previous Record Holders:

System	CPUs	Disks	Memory	Datamation	Minute Sort	Cost
AlphaSort (1994)	3	36	1.2 GB	7.0 s	1.1 GB	$1 Million
SGI XFS (1995)	12	96	2.2 GB	3.52 s	1.6 GB	~$2 Million
NOW-8	8	32	1 GB	??	??	$200 K
NOW-32	32	64	2 GB	??	??	$600 K

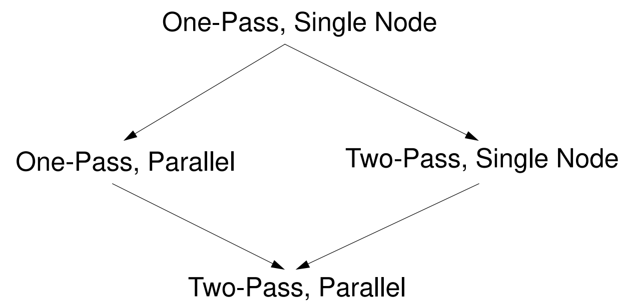
- Can NOW-Sort Challenge The Records?
  - Previous Record Holders were SMPs: Can NOWs Compete?

NOW-Sort Leverages NOW Technology

- Active Messages 1.0
  - Key To Moving Data Between Nodes
  - For This Application, Bandwidth is Important, Not Latency
- GLUnix
  - Need Real Parallel Environment (Job Control, Process Start-up)
  - Scheduling Not An Issue (Dedicated Environment)
- xFS
  - NOW-Sort Does NOT Use xFS (yet...)

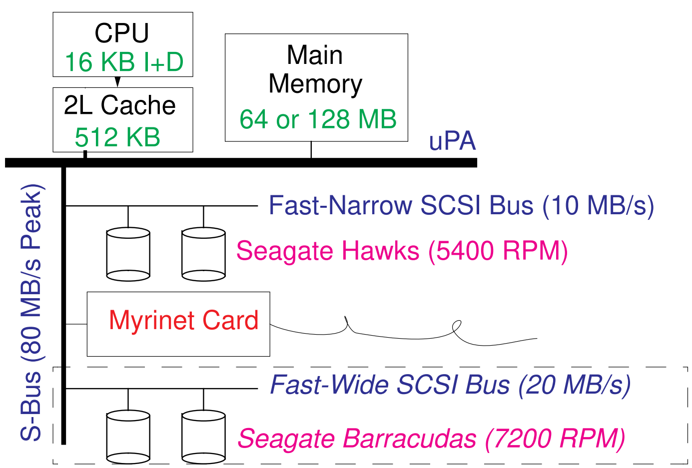
NOW-Sort Progression

- Start With Single-Node Algorithm
  - Tune for Performance
- Scale To Multiple Node Version



UltraSPARC I Architecture

- UltraSPARC I Block Diagram



One-Pass, Single Node Sorting

- Three Obvious Steps
  - Read Records from Disk
  - Sort Records
  - Write Sorted Records toDisk
- Questions To Answer
  - How To Optimize Each Phase?
  - How Much Overlap Can Occur Between Phases?

Step One: Reading From Disk

- How to Best Utilize Multiple Disks?
  - Striping!

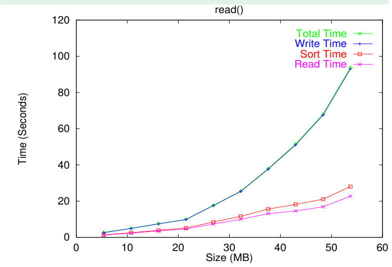
Read Bandwidth From One Seagate Hawk: 5.5 MB/s
Read Bandwidth from One Seagate Barracuda: 6.5 MB/s

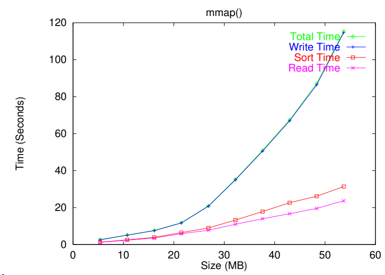
Questions:

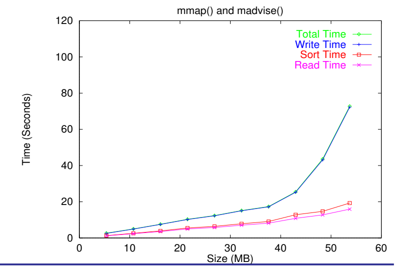
Stripe File Across Two Hawks; Read Bandwidth?
Stripe File Across Two Barracudas; Read Bandwidth?
Stripe File Across All Four Disks; Read Bandwidth?

Reading From Disk: OS Alternatives

- Option 1: read() system call
- Option 2: mmap()
- Option 3: mmap() + madvise()

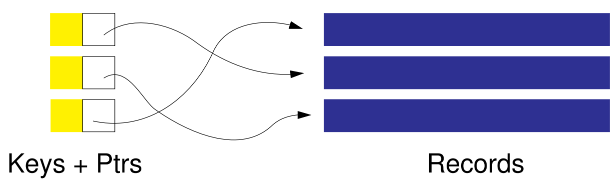






Step Two: Sorting

- Key Question: How to Utilize UltraSPARC Memory Hierarchy?
  - Separate Keys from Records, Sort Keys, Gather and Write



- Use Sorting Algorithm With Good Locality Properties
  - Quicksort?
  - Partial Bucket Sort + Quicksort?
  - Partial Bucket Sort + Radix Sort?

Sorting: Comparing Alternatives

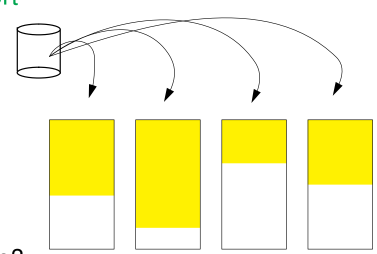
- Quicksort
  - Sorts All Key+Ptr Pairs
- Partial Bucket Sort + Quicksort

Examine Key, Distribute
To Bucket Based On Value
Of Top *B* Bits

(Records Still In Separate Array)

Sort Each Bucket Individually

Question: How Big Should *B* Be?

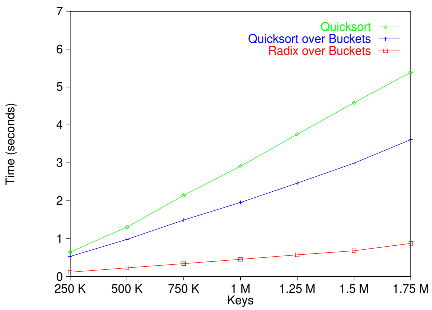


B=4 Buckets

- Partial Bucket Sort + Radix Sort
  - Buckets Only Get Half As Many Keys

Sorting: Results

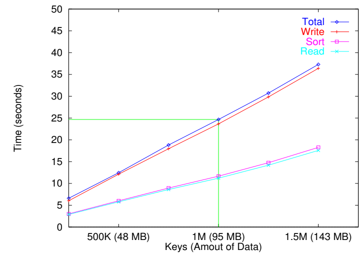
- Quantitative Comparison:

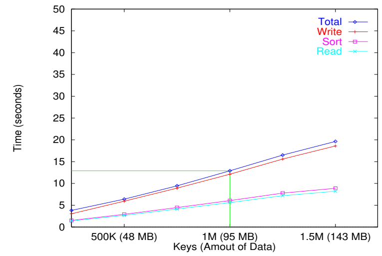


  - Simple Quicksort: Not Cache Sensitive
  - Bucket + Quicksort: Better, But Does Too Much Work
  - Bucket + Radix Sort: The Winner!

Single-Node Summary

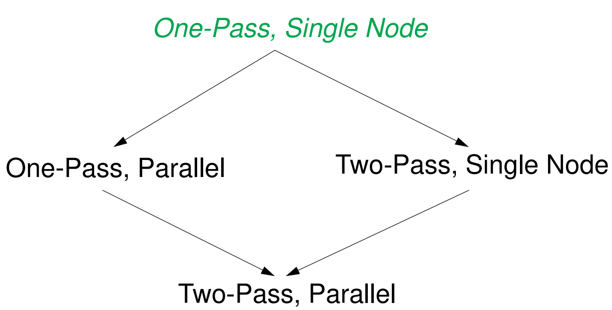
- Performance on 2-disk and 4-disk Systems





- Single Node Lessons
  - Architecture: Internal SCSI should be Fast-Wide, not Narrow
  - File Access: Careful Striping Yields Good Performance
  - File Access: mmap() + madvise() Sufficient
  - Sorting: Sensitive To Memory Hierarchy Makes Sort Time Tiny

NOW-Sort: Where Are We?



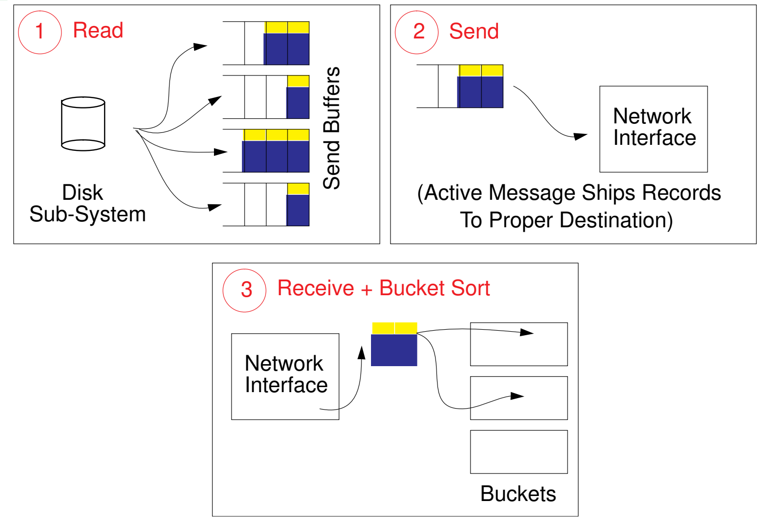
Cluster Architecture

- 8-Node Cluster
  - 128 MB Memory
  - 4 Disks (2 Hawks, 2 Barracudas)
- 32-Node Cluster
  - 64 MB Memory
  - 2 Disks (2 Hawks)

One-Pass Parallel Sorting

- Builds From One-Pass, Single-Node Sort
  - Read Records From Disk
  - New: Communicate
  - Sort Records
  - Write Records To Disk
- Questions To Answer
  - How To Optimize Each Step?
    - Any Difference From One-Node Sort?
  - How Much Overlap Can Occur Between Phases?

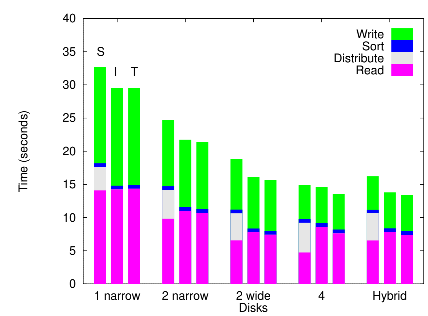
How Does Communication Work?



One-Pass Parallel Sort: Alternatives

- Synchronous
  - Read All Records, Then Communicate, Then Sort & Write
- Interleaved
  - Reads Records & Periodically Communicates, Then Sorts & Writes
- Threaded
  - Read Thread & Communicate Thread, Then Sort & Write

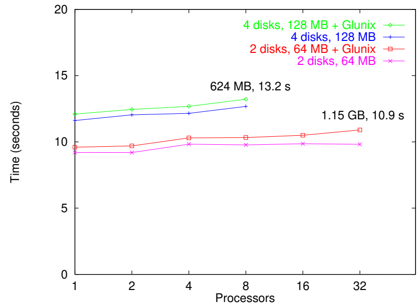
Quantitative Comparison



- Conclusions:
  - Threaded Algorithm Most Robust
  - Not Much Gain From 4-Disks... Why?

Scaling

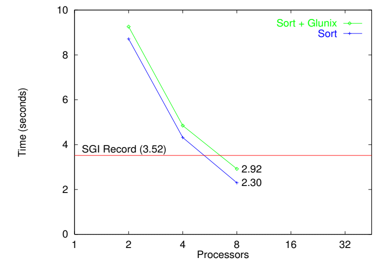
- Records Per Processor Kept Constant

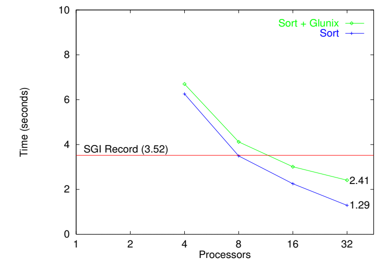


- Scaling Results:
  - Performance Stays High
  - Glunix Overhead Noticeable

The Datamation Benchmark

- Datamation Benchmark: 1 Million Records





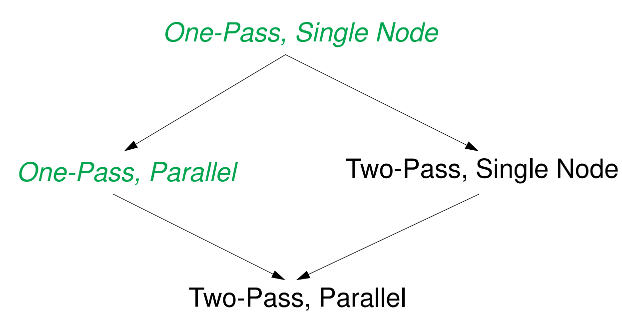
- Conclusions:
  - NOWs Can Be Reasonable Platform For Low-Latency Jobs
  - Start-up Time Needs To Improve

One-Pass Parallel Summary

- Architectural Bottleneck: Sun I/O Bus
  - Workstations Can't Utilize More Than Two Disks!
  - Converse: Balanced System Has Two Disks Per Node

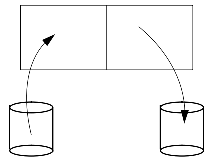
- Distributed OS Bottleneck: Start-up Time
  - Prototype System: Should Get Better
  - Similar Start-up Problem for SMPs

NOW-Sort Progression

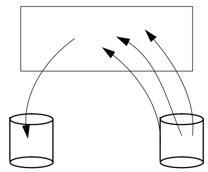


Two-Pass, Single Node Overview

- Two Passes Must Be Used When Records Don't Fit In Memory
- Phase 1: Create Sorted Runs
  - Read In Records To Fill Memory
  - Sort & Write Out Records
  - Repeat Until All Data Has Been Read

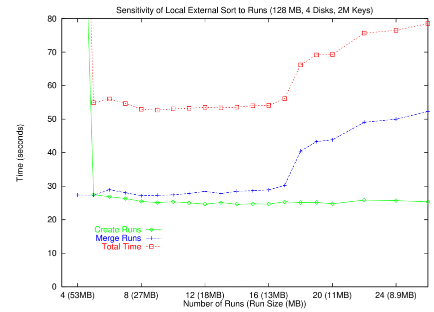


- Phase 2: Merge Sorted Runs
  - Read From All Runs
  - Take Lowest Key, Write Out To Disk
  - Repeat Until Finished



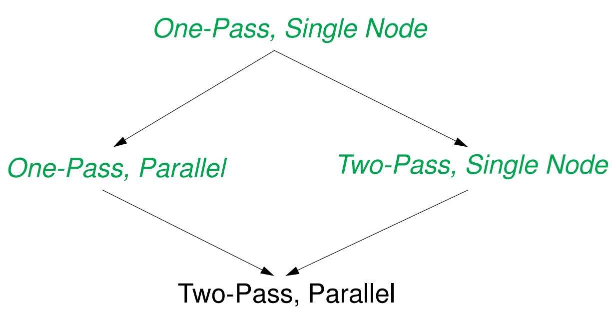
Two-Pass Performance

- Let's Vary The Run Size:



- What Are The Different Performance Regimes?

NOW-Sort: Almost Finished!

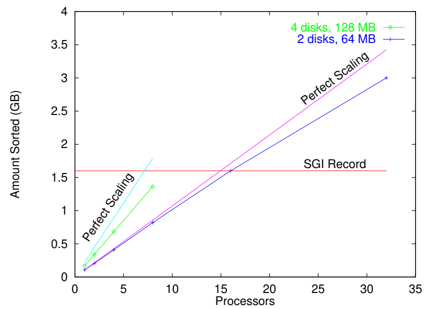


Two-Pass, Parallel Sorting

- Just Combine All The Pieces
- Phase 1: Create Runs (ala Multiple One-Pass, Parallel Sorts)
  - Read Records and Send To Other Processors
  - Sort Records and Write To Disk
  - Repeat Until All Data Has Been Read
- Phase 2: Merge Sorted Runs (ala Two-Pass, Single-Node Sort)
  - Read From All Runs
  - Take Lowest Key, Write Out To Disk
  - Repeat Until Finished

The MinuteSort Benchmark

- How Much Can NOW-Sort Sort In One Minute?



NOW-Sort: Summary

- UltraSPARC I Architecture
  - Fast-Narrow SCSI Bus: Bad Decision
  - S-Bus: Antiquated, Needs To Improve
  - Application Must Heed The Memory Hierarchy
    - Importance of Second-Level Cache
- Operating System
  - Careful Striping Can Fully Utilize Heterogeneous Disks
  - mmap() + madvise() Interface Adequate
  - Parallel Start-up Time Currently Problematic (But Getting Better)

NOW-Sort: Final Performance

System	CPUs	Disks	Memory	Datamation	Minute Sort	Cost
AlphaSort (1994)	3	36	1.2 GB	7.0 s	1.1 GB	$1 Million
SGI XFS (1995)	12	96	2.2 GB	3.52 s	1.6 GB	~$2 Million
NOW-8	8	32	1 GB	2.92 s	1.4 GB	$200 K
NOW-32	32	64	2 GB	2.41 s	3.0 GB	$600 K

Obligatory URLs:

http://now.cs.berkeley.edu

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