GLUnix: A New Approach to Operating Systems for Networks of Workstations

Tom Anderson, Mike Dahlin, Doug Ghormley, Steve Rodrigues, Drew Roselli, Amin Vahdat, Keith Vetter, Randy Wang, Kristin Wright, and Dave Patterson

Computer Science Division
University of California, Berkeley

Outline

Structure:

Build global layer on top, not from scratch

Sociology:

Benefit both sequential and parallel users

I/O:

Serverless network file service

Why a NOW OS now? Enabling technology: high bandwidth, low latency switched networks.

Hardware Argument for NOWs vs. MPPs

Build large systems out of off-the-shelf components:

- Volume manufacturing

  => low cost

- No time lag integrating system

  => better CPU performance (4%/month)

- LAN converging to MPP interconnect

  => No fundamental difference in communication performance

Why doesn't the same argument apply to software?

Getting Out of the Way of the OS

Reality: >>$100M/year commercial OS investment

Research situation same as with processors, compilers:

Build from scratch?

Never catch up to commercial systems

Modify existing system?

Can't track changes in commercial systems

Time lag => lost performance, reliability

Why not layer on top of existing off-the-shelf OS's?

Enabling Technology: Software Fault Isolation

Traditional approach: use hardware to enforce protection between different address spaces

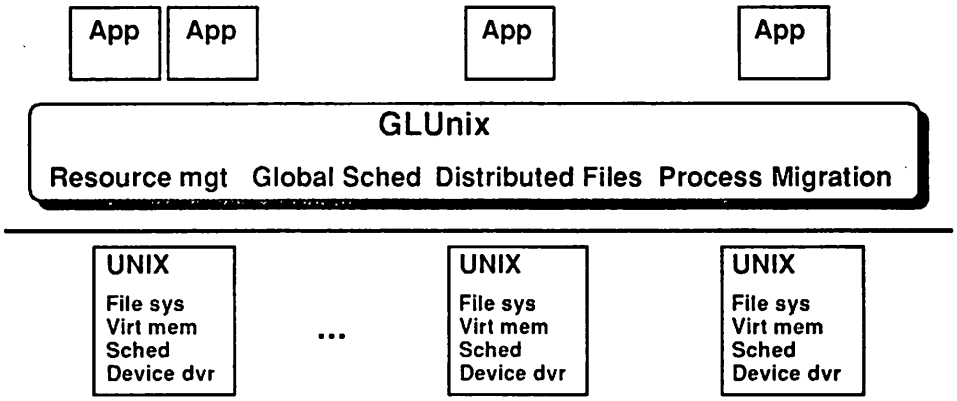
New approach: enforce fault isolation within single address space (Wahbe, Symp. O.S. Princ. '93)

- Insert language-independent software checks into object code (stores and indirect branches), optimize
- Overhead for SPEC is 3-7% on MIPS, Sparc, Alpha

Enables fuzzing of kernel-user boundary

- Run arbitrary user code safely in the kernel
- Run protected kernel code in application libraries

GLUnix: A Virtual Operating System Layer



GLUnix glues workstation UNIXs together to provide global services.

Outline

Structure:

Build global layer on top, not from scratch

Sociology:

Benefit both sequential and parallel users

I/O:

Serverless network file service

Historical Perspective

Personal computers originally assumed no longer needed features of time-shared systems:

1 process

no communication

no file sharing

no security

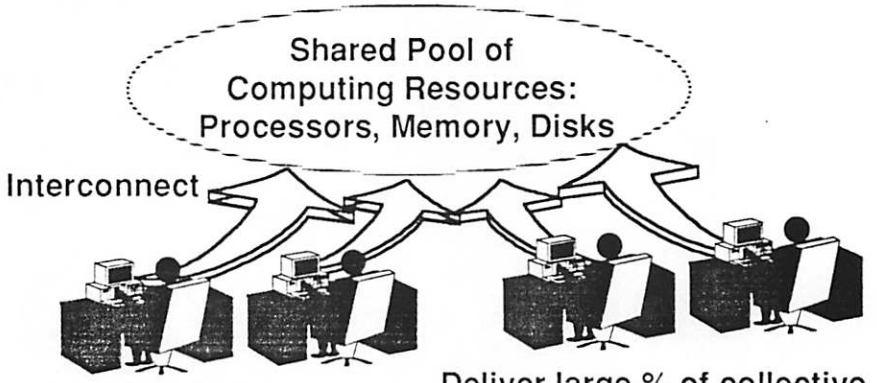
no shared resources

One by one, PC OS's have added these back in.

GLUnix goal is to eliminate the last restriction!

GLUnix Goal

Use same infrastructure for sequential and parallel users; benefit both sets of users!



Guarantee at least one workstation to many individuals (when active)

Deliver large % of collective resources to few individuals at any one time

GLUnix Technical Challenges

Preserving interactive performance

- Must quickly restore state to be as good as dedicated workstation for uniprocessor jobs
- Focus on memory state as well as CPU cycles

  Roadblock is delay in restoring entire workstation context
- Time to save or restore:

64MB over Ethernet, single disk	60 seconds
64MB over ATM, parallel file sys	4 seconds

Improving sequential performance

- network RAM
- file system cooperative caching, disk striping

GLUnix Technical Challenges (cont'd)

Availability as good as standalone workstation

- hot swap of hardware, software components

Parallel performance

- must gang schedule parallel jobs to be as good as dedicated MPP for parallel jobs

Outline

Structure:

Build global layer on top, not from scratch

Sociology:

Benefit both sequential and parallel users

I/O:

Serverless network file service

The Dark Side of Client-Server Computing

Traditional approach:

- Central server provides global file system to clients
- Caching at client, server for performance

Centralized server is a bottleneck:

- Performance -- all file bytes go through server
- Availability -- single point of failure, unless replicated
- Price -- servers as overpriced as MPP's!

xFS: Serverless Network File Service

Client workstations cooperate to provide file service:

Dynamic location of data, meta-data, and control

- Easier migration, easier failure recovery

Multiprocessor-style cache coherence

- Maximize locality via write ownership, direct client-to-client transfers

Software RAID and LFS

- Parallel I/O, highly available despite client crashes

Cooperative caching

- Aggregate client memory as giant cache for disk; client disks as giant cache for tape

Cooperative Caching

Time to transfer 8KB file block:

	Remote Mem	Remote Disk
Ethernet	6.9 ms	21.9 ms
ATM (155Mb/s)	1.1 ms	16.1 ms

Results for cooperative caching: (Sprite file trace)

	Miss Rate	Response Time
Client/Server	16%	2.8 ms
xFS	8%	1.6 ms

(42 WS, 16 MB/WS, 128 MB/server)

Conclusions

Structure:

Build global layer on top, not from scratch

Sociology:

Benefit both sequential and parallel users

I/O:

Serverless network file service

Initial working implementation. Papers available via Mosaic: now.cs.berkeley.edu.