10/19

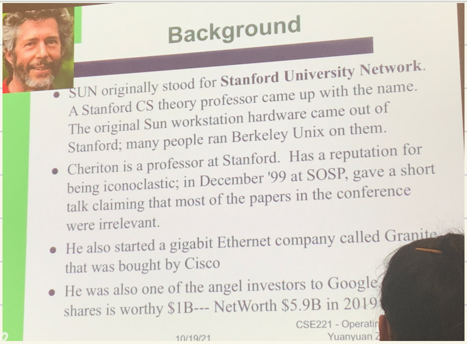
- D. R. Cheriton and W. Zwaenepoel, The Distributed V Kernel and its Performance for Diskless Workstations, Proceedings of the 9th Symposium on Operating Systems Principles, pp. 129-140, November 1983.

Q: What is the argument for diskless workstations, and do you agree/disagree with the argument?

- J. K. Ousterhout, A. R. Cerenson, F. Douglis, M. N. Nelson, and B. B. Welch, The Sprite Network Operating System, IEEE Computer, Vol. 21, No. 2, February 1988, pp. 23-36.

Q: How do the caching policies in Sprite differ from those in the V Kernel?

Sun

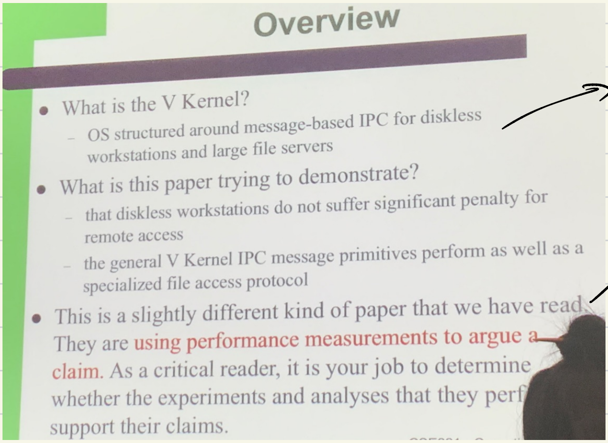


Background

- SUN originally stood for **Stanford University Network**. A Stanford CS theory professor came up with the name. The original Sun workstation hardware came out of Stanford; many people ran Berkeley Unix on them.
- Cheriton is a professor at Stanford. Has a reputation for being iconoclastic; in December '99 at SOSP, gave a short talk claiming that most of the papers in the conference were irrelevant.
- He also started a gigabit Ethernet company called Granite that was bought by Cisco
- He was also one of the angel investors to Google shares is worthy $1B--- NetWorth $5.9B in 2019

CSE221 - Operatin
Yuanyuan Z
10/19/21

Overview



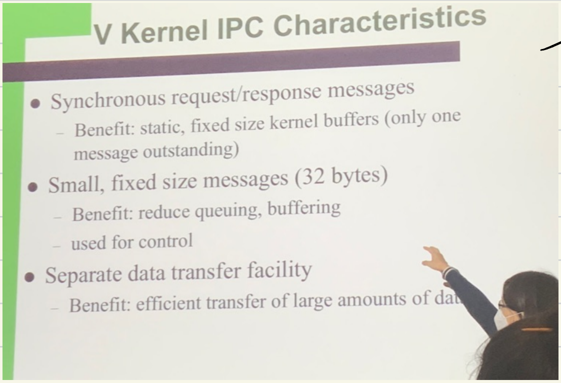
Overview

- What is the V Kernel?
  - OS structured around message-based IPC for diskless workstations and large file servers
- What is this paper trying to demonstrate?
  - that diskless workstations do not suffer significant penalty for remote access
  - the general V Kernel IPC message primitives perform as well as a specialized file access protocol
- This is a slightly different kind of paper that we have read. They are using performance measurements to argue a claim. As a critical reader, it is your job to determine whether the experiments and analyses that they perf support their claims.

promote PHD thesis.

identity

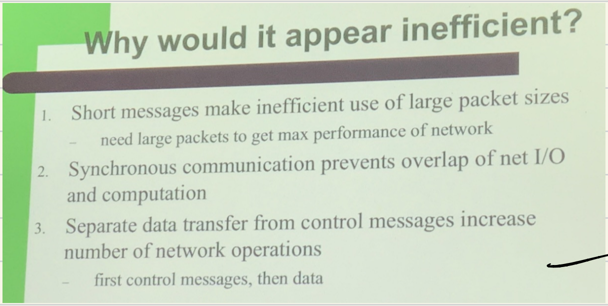
if it is realible



V Kernel IPC Characteristics

- Synchronous request/response messages
  - Benefit: static, fixed size kernel buffers (only one message outstanding)
- Small, fixed size messages (32 bytes)
  - Benefit: reduce queuing, buffering
  - used for control
- Separate data transfer facility
  - Benefit: efficient transfer of large amounts of dat

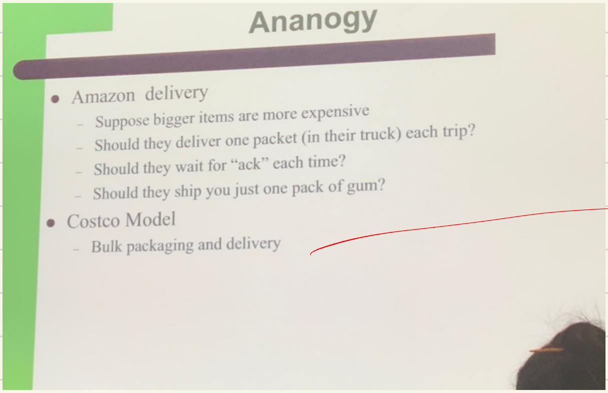
→ Not efficient



Why would it appear inefficient?

1. Short messages make inefficient use of large packet sizes
   - need large packets to get max performance of network
2. Synchronous communication prevents overlap of net I/O and computation
3. Separate data transfer from control messages increase number of network operations
   - first control messages, then data

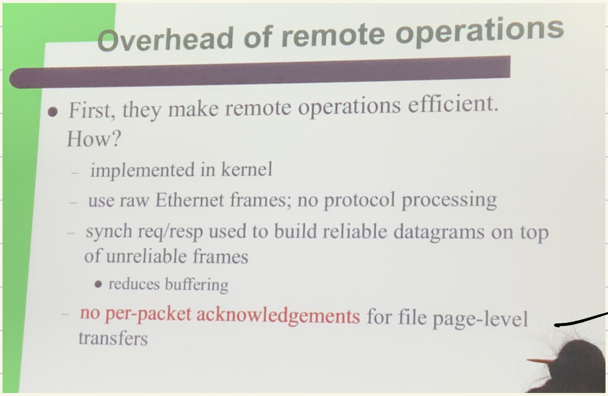
→ combine message together



Ananogy

- Amazon delivery
  - Suppose bigger items are more expensive
  - Should they deliver one packet (in their truck) each trip?
  - Should they wait for "ack" each time?
  - Should they ship you just one pack of gum?
- Costco Model
  - Bulk packaging and delivery

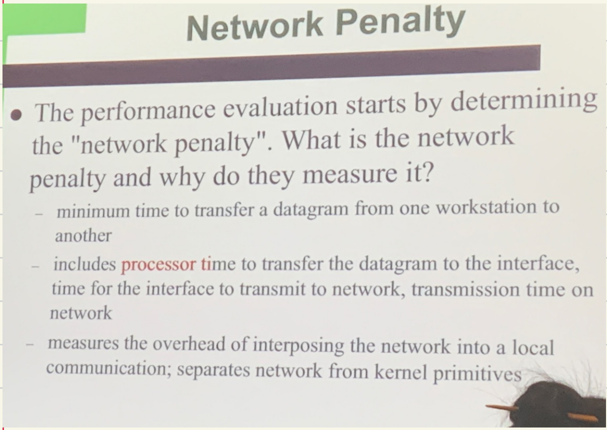
→ buffer.



Overhead of remote operations

- First, they make remote operations efficient. How?
  - implemented in kernel
  - use raw Ethernet frames; no protocol processing
  - synch req/resp used to build reliable datagrams on top of unreliable frames
    - reduces buffering
  - no per-packet acknowledgements for file page-level transfers

→ implicit ack.

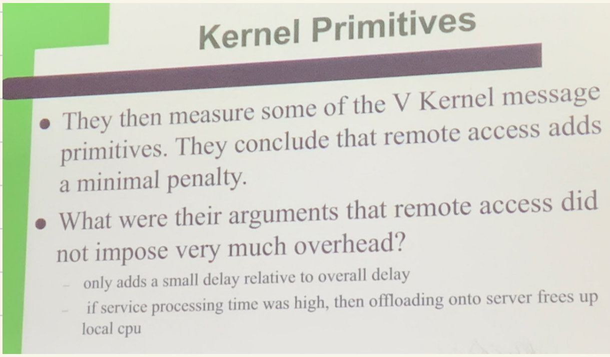


Network Penalty

- The performance evaluation starts by determining the "network penalty". What is the network penalty and why do they measure it?
  - minimum time to transfer a datagram from one workstation to another
  - includes processor time to transfer the datagram to the interface, time for the interface to transmit to network, transmission time on network
  - measures the overhead of interposing the network into a local communication; separates network from kernel primitives

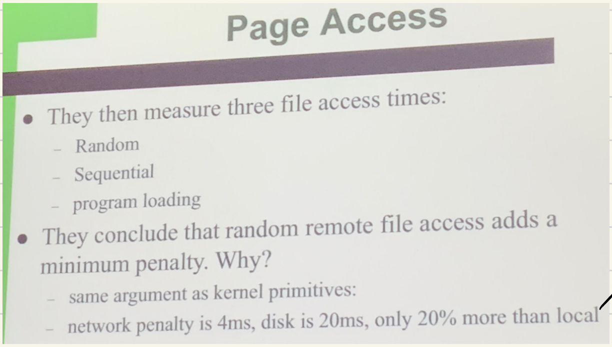
It does not include TCP layer

but UDP and IP still exist



Kernel Primitives

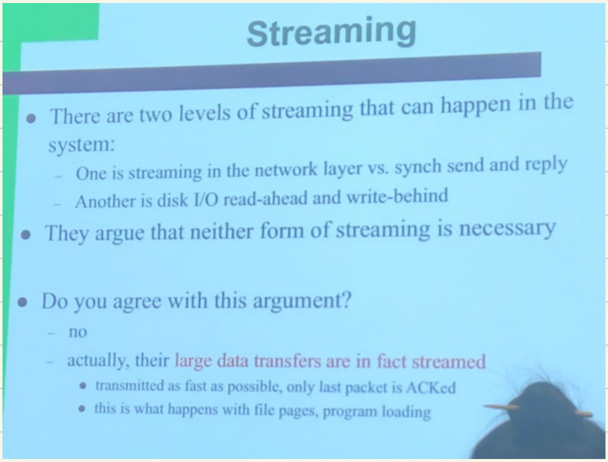
- They then measure some of the V Kernel message primitives. They conclude that remote access adds a minimal penalty.
- What were their arguments that remote access did not impose very much overhead?
  - only adds a small delay relative to overall delay
  - if service processing time was high, then offloading onto server frees up local cpu



Page Access

- They then measure three file access times:
  - Random
  - Sequential
  - program loading
- They conclude that random remote file access adds a minimum penalty. Why?
  - same argument as kernel primitives:
  - network penalty is 4ms, disk is 20ms, only 20% more than local

→ the percentage drop.

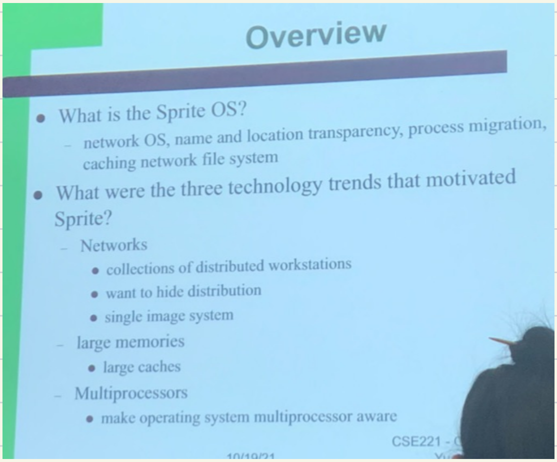


Streaming

- There are two levels of streaming that can happen in the system:
  - One is streaming in the network layer vs. synch send and reply
  - Another is disk I/O read-ahead and write-behind
- They argue that neither form of streaming is necessary
- Do you agree with this argument?
  - no
  - actually, their large data transfers are in fact streamed
    - transmitted as fast as possible, only last packet is ACKed
    - this is what happens with file pages, program loading

Better Internet make remote cheap and fast

Sprite

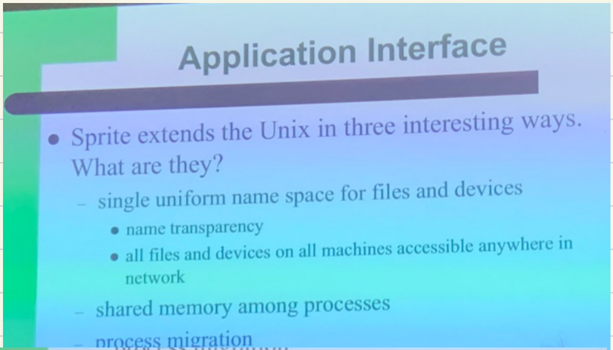


Overview

- What is the Sprite OS?
  - network OS, name and location transparency, process migration, caching network file system
- What were the three technology trends that motivated Sprite?
  - Networks
    - collections of distributed workstations
    - want to hide distribution
    - single image system
  - large memories
    - large caches
  - Multiprocessors
    - make operating system multiprocessor aware

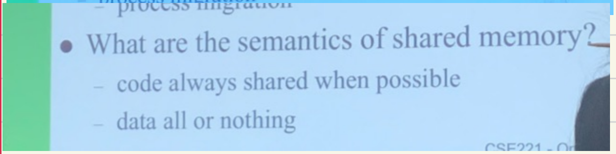
CSE221 - C

10/19/21



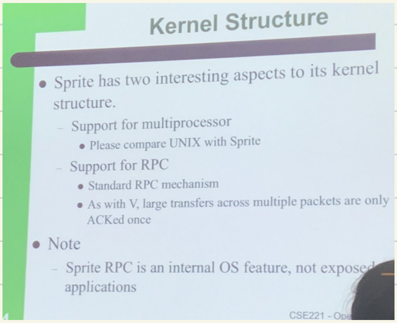
Application Interface

- Sprite extends the Unix in three interesting ways. What are they?
  - single uniform name space for files and devices
    - name transparency
    - all files and devices on all machines accessible anywhere in network
  - shared memory among processes
  - process migration



- What are the semantics of shared memory?
  - code always shared when possible
  - data all or nothing

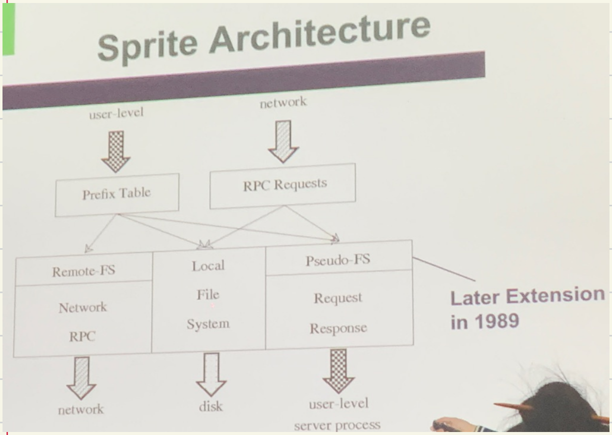
CSE221 - Or



Kernel Structure

- Sprite has two interesting aspects to its kernel structure.
  - Support for multiprocessor
    - Please compare UNIX with Sprite
  - Support for RPC
    - Standard RPC mechanism
    - As with V, large transfers across multiple packets are only ACKed once
- Note
  - Sprite RPC is an internal OS feature, not exposed applications

CSE221 - Ope



Sprite Architecture

user-level

network

Prefix Table

RPC Requests

Remote-FS

Network RPC

Local File System

Pseudo-FS

Request Response

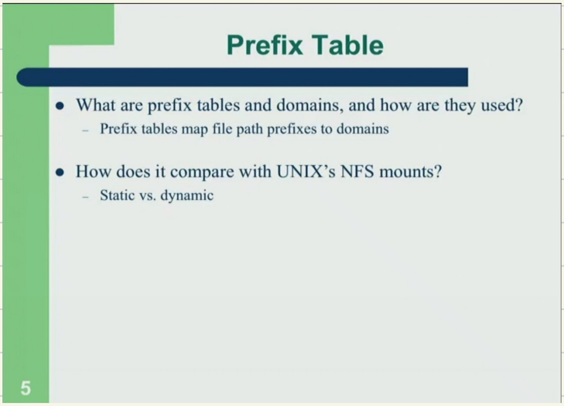
Later Extension in 1989

network

disk

user-level server process

Prefix table

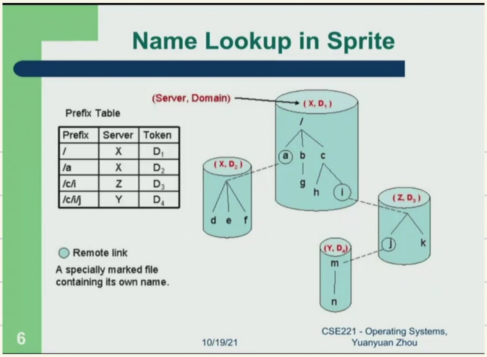


Prefix Table

- What are prefix tables and domains, and how are they used?
  - Prefix tables map file path prefixes to domains
- How does it compare with UNIX's NFS mounts?
  - Static vs. dynamic

5

It is mostly works as hint.



Name Lookup in Sprite

Prefix Table

Prefix	Server	Token
/	X	D_1
/a	X	D_2
/c/i	Z	D_3
/c/i/j	Y	D_4

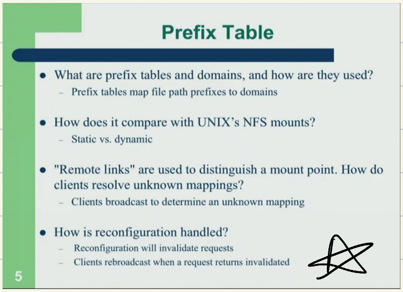
○ Remote link

A specially marked file containing its own name.

6

10/19/21

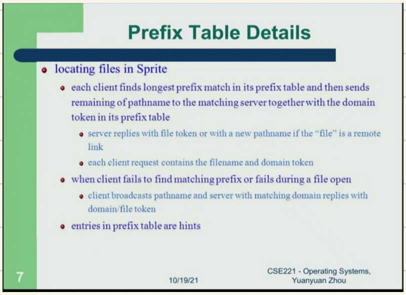
CSE221 - Operating Systems, Yuanyuan Zhou



Prefix Table

- What are prefix tables and domains, and how are they used?
  - Prefix tables map file path prefixes to domains
- How does it compare with UNIX's NFS mounts?
  - Static vs. dynamic
- "Remote links" are used to distinguish a mount point. How do clients resolve unknown mappings?
  - Clients broadcast to determine an unknown mapping
- How is reconfiguration handled?
  - Reconfiguration will invalidate requests
  - Clients rebroadcast when a request returns invalidated

5



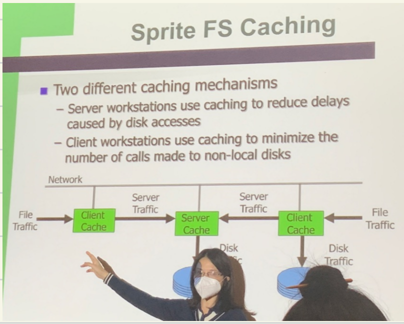
Prefix Table Details

- locating files in Sprite
  - each client finds longest prefix match in its prefix table and then sends remaining of pathname to the matching server together with the domain token in its prefix table
    - server replies with file token or with a new pathname if the "file" is a remote link
    - each client request contains the filename and domain token
  - when client fails to find matching prefix or fails during a file open
    - client broadcasts pathname and server with matching domain replies with domain/file token
  - entries in prefix table are hints

7

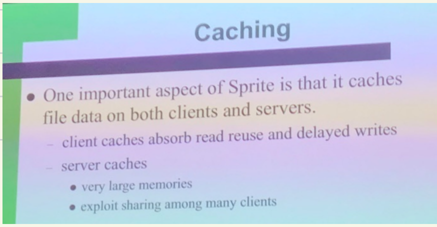
10/19/21

CSE221 - Operating Systems, Yuanyuan Zhou



Sprite FS Caching

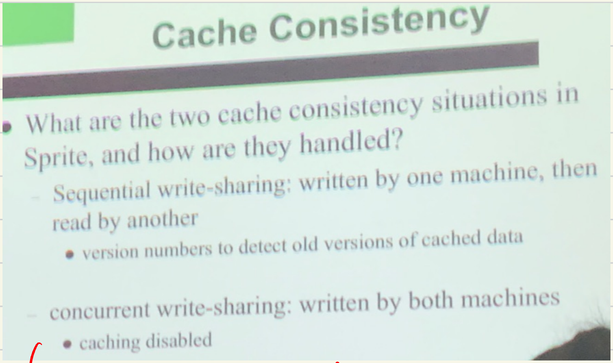
- Two different caching mechanisms
  - Server workstations use caching to reduce delays caused by disk accesses
  - Client workstations use caching to minimize the number of calls made to non-local disks



Caching

- One important aspect of Sprite is that it caches file data on both clients and servers.
  - client caches absorb read reuse and delayed writes
  - server caches
    - very large memories
    - exploit sharing among many clients

avoid caching on client to avoid consistency issue.

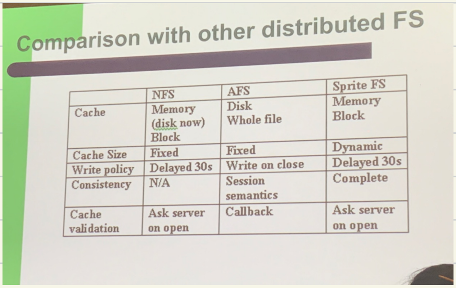


Cache Consistency

- What are the two cache consistency situations in Sprite, and how are they handled?
  - Sequential write-sharing: written by one machine, then read by another
    - version numbers to detect old versions of cached data
  - concurrent write-sharing: written by both machines
    - caching disabled

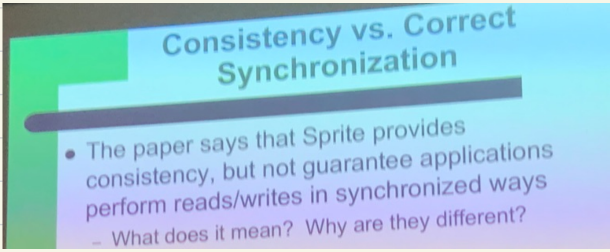
one writer mode

→ too complicated, no cache.



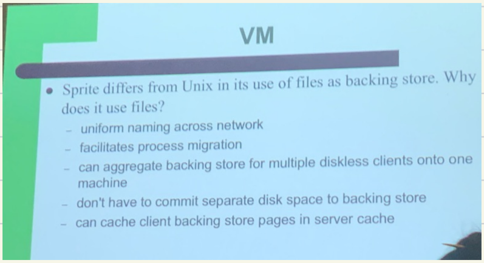
Comparison with other distributed FS

	NFS	AFS	Sprite FS
Cache	Memory (disk now) Block	Disk Whole file	Memory Block
Cache Size	Fixed	Fixed	Dynamic
Write policy	Delayed 30s	Write on close	Delayed 30s
Consistency	N/A	Session semantics	Complete
Cache validation	Ask server on open	Callback	Ask server on open



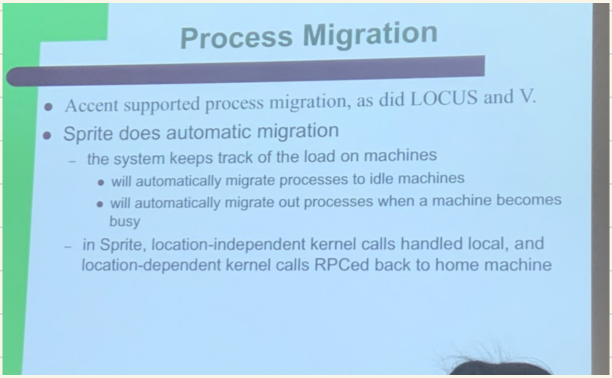
Consistency vs. Correct Synchronization

- The paper says that Sprite provides consistency, but not guarantee applications perform reads/writes in synchronized ways
  - What does it mean? Why are they different?



VM

- Sprite differs from Unix in its use of files as backing store. Why does it use files?
  - uniform naming across network
  - facilitates process migration
  - can aggregate backing store for multiple diskless clients onto one machine
  - don't have to commit separate disk space to backing store
  - can cache client backing store pages in server cache



Process Migration

- Accent supported process migration, as did LOCUS and V.
- Sprite does automatic migration
  - the system keeps track of the load on machines
    - will automatically migrate processes to idle machines
    - will automatically migrate out processes when a machine becomes busy
  - in Sprite, location-independent kernel calls handled local, and location-dependent kernel calls RPCed back to home machine