Hot Areas and Topics in Operating Systems Research

CS 239, Spring 1995

June 7, 1995

Some Hot Areas

- Mobility
- Multimedia support
- Extensibility
- Highly optimized message handling
- Load balancing and process migration
- System support for electronic commerce
- Cryptography for the masses

Mobility

Portable computers are becoming more powerful

Wireless communications are becoming available

So users will increasingly use computers that move from place to place

Systems Software Problems for Mobile Computers

- Power management
- Handling limited local resources
- Varying communications capabilities
- Security

Power Management for Portable Computers

Typical portable computers have battery lives of a couple of hours to six hours

If you run the portable the way you run an equivalent plugged in machine, battery life is short

So what do you do to make batteries last longer?

Power Management Issues

What uses power?

- Display
- Disk
- CPU
- Memory
- Wireless communications, if you have them

Different approaches required for each

Basic Approach for Power Management

Turn it off when you don't need it

One simple case - if the user has shut the machine's lid, power nearly all down

One step further - if the screen hasn't been updated recently, turn it off

And another step - if the disk is not in use, spin it down

And another step - shut off the CPU

Where's the Research in Power Management?

- When do you turn components off?
- How do you assure state is preserved when they come back on?
- How do you make necessary performance guarantees?

One Example - Disk Power Management

Disks burn a lot of power keeping the disk spinning

If no data is being fetched, that's wasted power

So you can stop spinning the disk

But it takes significant time to start it spinning again

Spinning Disks Up and Down

Need to predict when a disk can safely be spun down

Could set a fixed threshold

Perfect knowledge algorithm can save 35-50% over fixed threshold

Short fixed thresholds (small number of seconds) seem best

Helping Out the Disk Power Management

Cache aggressively

- Cache directory attributes that might be needed
- Use large write-back cache

Live with occasional spin-up delays

- And perhaps try to conceal them

Handling Other Limited Resources in Mobile Machines

Disk space is limited on mobile machines

If the machine is totally disconnected, it's totally dependent on its disk

So it's important to fill the disk wisely

If a wireless connection is available, it's got low bandwidth

- So prioritizing communications vital

Dealing With Varying Communications Capabilities

With mobile machines, sometimes you have megabytes per second

- Sometimes kilobytes per second
- Sometimes bytes per second
- Sometimes nothing

And it changes as you move

Other communications characteristics also change

Problems With Varying Communications

- Applications need to “hand off” their communications from one device to another
- Applications need to adapt to lower/higher communications rates
- Applications may want to split their communications between multiple channels

Security for Mobile Computers

Wireless communications is easier to tap

Mobile computers can move anywhere

- How do they deal with trust in an unfamiliar environment?

Mobile computers may need to request lots of data/programs from semi-trusted sources

Multimedia Support

Users increasingly want audio and video delivered to their machines

Some from saved files

Some from devices attached to the machine

Some over the network

What must a system do to support these needs?

Multimedia and Real Time

Audio and video delivery requires real-time-like guarantees

Delays in delivery of this kind of data lead to highly undesirable effects

But most machines are not used exclusively for these tasks

What Problems Does the System Face?

- Scheduling
- Buffer management
- Graceful degradation
- Smoothing jitter in the delivery

Scheduling for Multimedia Support

Audio and video require periodic service
- If you don't schedule them, the output suffers

But you don't want to spend all resources on supporting a video stream

And what about handling multiple simultaneous video streams?

Overloading Multimedia Systems

When a system supporting multimedia gets overloaded, what should you do?

- Degrade video before audio
- Degrade incoming streams before outgoing
- Degrade stream opened the longest

But answers vary by system and desired result

Buffer Management for Multimedia

Multimedia data may arrive while the display process is swapped out

Or it may be coming in too fast to deal with right now

So you must buffer it
- Need buffers fast, and possibly a lot of them

Need intelligence about handling full buffers

Graceful Degradation in Multimedia Systems

When system can't keep up with all multimedia demands, it shouldn't freeze up

Especially in video, relatively easy to offer slightly degraded services
- Color goes to B/W
- Frame rate goes down
- Resolution goes down

Partially the job of the application

Smoothing Jitter for Multimedia

Data may be delivered at different intervals than it needs to be displayed

In which case special buffering requirements to ensure that display doesn't have jitter

Should buffer close to the actual displaying device

Multimedia For Wireless Networks and Mobile Computers

Must have good way to hand off data transmission from wireless cell to cell
- Can't afford to drop frames because the receiver is moving

Mobile computer doesn't have power to customize its format
- So server should store data in multiple formats

Extensibility in Operating Systems

Widely regarded as key issue in designing a modern operating system

Microkernels provide it by allowing out-of-kernel servers

Stackable services provide it by allowing addition of stacked layers/protocols

Some approaches allow user code to execute in a privileged mode

Why is Extensibility So Hot?

Experience shows system designers never get it completely right

And individual users/applications have special needs

Example - multimedia output can be very high-bandwidth

- So it's bad to pass the bits through the kernel

Why Isn't a Simple Microkernel Design the Whole Answer?

Speed

- Microkernel systems tend to be slow

Safety

- Hard to guarantee safe operation among several users/services sharing a service

One Solution - Run User Code in Kernel Space

SPIN (at U. of Washington) is doing this

Key issues -

- Providing safety of overall system
- High performance

Safety provided by object oriented techniques and explicit guards

Performance provided by run-time compilation of code into OS

Other Extensibility Approaches

- x-kernel - allows extensibility by defining new communications protocols
- Stackable file systems - allows extensibility to file systems
- Spring file systems - allows extensibility to file systems through object orientation
- Frigate - Uses stackable file systems to build object oriented files into the FS

Extensibility Through Code Interpolation

Typical method of providing extensibility -

- Allow user to slip code between applications and system calls

Common problem in such systems - either

- 1). Only part of the system interface is available; or
- 2). User must write code to handle all system calls

Getting Around the Costs of Interposing User Code

Abstract the system's calls into manipulation of a few object types

Construct toolkit that manipulates them properly

Allow easy way for user to replace only the parts of the toolkit he needs

One example provided by Michael Jones, for 4.3 BSD

Highly Optimized Message Handling

Many systems built on RPC, client/server, or object oriented paradigm

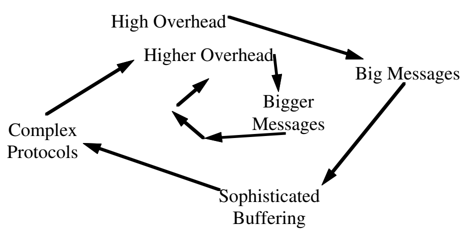
Which means lots of messages between entities in different contexts

Success of such systems relies on extremely fast message delivery

Messages also important in parallel processing

Why Not Just Send Bigger Messages?

Because of the Big Message Spiral



Basic Ideas in Speeding Up Message Passing

- Avoid buffering and copying
  - Buffering requires management
  - Copying takes time
- Avoid context switching
  - Context switching is heavy operation
  - Naive microkernel designs do it a lot
- Use any available hardware assistance

Protocol Service Decomposition

Done in the context of particular communications protocols (e.g., TCP/IP)

Separate application's interface to the network from its interface to the OS

In other words, provide direct path from application to the network for common ops

OS takes care of heavyweight network management

Architecture Components of Protocol Decomposition

- Operating system server - provides the heavyweight services (connection establishment/teardown, routing info, etc.)
- Multithreaded application library - implements basic protocol (send, receive)
- Network interface - thin layer of software controlling the network hardware

How Protocol Decomposition Fits Together

Application (containing the multithreaded library) asks operating system server to start a network session

OS provides session that allows application to send certain operations directly to network interface

So normal reads/writes don't go through the OS

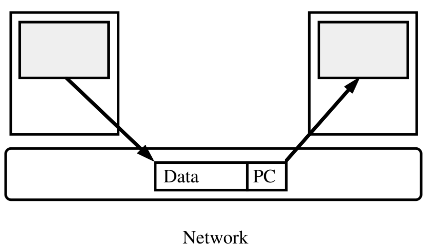
Active Messages

In context of parallel processors and cooperating threads within task

Have handlers for messages compiled into all threads

Handlers know precisely what to do with specific message types

Sending Active Messages



Load Balancing and Process Migration

Important on parallel processors

Imbalances in load impede speedup

Also important in certain network-of-workstation situations

Particularly when you're trying to make use of spare cycles

Basic Problems of Load Balancing

- Recognizing load imbalance
- Identifying what to move
- Avoiding thrashing

Recognizing Imbalance

Requires getting information from other nodes

Global methods versus clustering methods

How frequently do you gather information?

And what information do you gather?

Length of ready queue?

Percent idle cycles?

Average waiting time for ready processes?

Identifying Something To Move

Generally assume near future will be like recent past

Try to find thread/object/process to move that will lead to balance

But watch out for hidden interactions

Do you really get more parallelism by moving the entity?

Avoiding Thrashing

Need to avoid situations where you're perpetually moving things back and forth

Either dampen frequency of movement

- Or tie down nodes involved in migration
- Or otherwise inhibit normal load management

Process Migration

Relatively easy in shared memory machines

- Since process can still access its memory

But if it's a NUMA processor, must migrate some or all of the state

How much to migrate, and when?

Process Migration Choices

- Migrate all at once atomically
- Start migration while still running copy at source, transfer changed data from source to destination at the end
- Demand page the process

Electronic Commerce

How can we do business over computer networks?

A fundamental issue is how to charge for services

- Bill the user
- Credit cards transactions
- Electronic checks
- Electronic cash

Security and Electronic Commerce

Billing, credit card transactions, electronic checks require strong authentication

Servers that have never heard of users before may need to authenticate them

Notary public solutions

Networks of trusted authentication servers

Electronic Cash

- Different properties than other forms of electronic commerce
- Anonymous
- Transferrable
- Non-traceable
- But also non-forgeable
- Issued by electronic banks

Other Security Issues for Electronic Commerce

- Can we provide services free from eavesdropping?
- Can we provide one-time-per-payment services?
- Can we prevent dissemination of valuable data?

Cryptography for the Masses

- Technical problems like key distribution
- But, more importantly, social problems
  - Should terrorists to be able to conduct untappable phone conversations?
  - Should child pornographers be allowed to hide their wares behind encryption?
  - But we don't trust the government to listen to everything

Key Escrow Solutions

- Allow ordinary users to use cryptographically strong cipher
- But make it possible for the government to obtain the key
  - Without assuming users will willingly provide it
- Based on storing parts of the key with trusted agencies