Hot Topics in Operating Systems
CS239
Advanced Operating Systems
June 3, 1997

Outline

- Mobile computing, proxies, and agents
- Prediction, introspection, and self-organization

Mobile Computing, Proxies, and Agents

- What are the special challenges of mobile computing?
- What are proxies and agents?
- How can they help mobile computers?
- Research issues

Mobile Computing

- Portable computers are becoming increasingly popular
- Portables computers are inherently mobile
  - They compute while moving
  - They communicate from various places

Systems Issues for Mobile Computers

- Power management
- Handling disconnection
- Handling limited connectivity
- Handling communications while mobile

Power Management Issues

- Batteries have limited lives
- The lifetime of batteries isn't increasing much
- The amount of power you're willing to carry isn't enough
  - For many purposes

How To Save Power

Well, use less

- Shut down things you don't need
- Burn the power elsewhere

Shutting Things Down

What can be profitably shut down?

- Screen
- Disk
- Other I/O devices
- Memory
- Processor

Disk Shutdown Strategies

Disks burn lots of power while spinning

So spin disk down when you don't need it

The problem - spinning a disk down and spinning it back up takes more power

In practice, aggressive choices work best

E.g., spin down with 1 second idle time

Burning the Power Elsewhere

Since portables don't have much power, make someone who does do your work

- Migrate big user tasks
- Run system-related tasks elsewhere

Handling Disconnections

Most portable computers will eventually have to run with zero connectivity

Best if they run well when disconnected

How do we make that happen?

Make it look to the user and applications like it is connected

Making Disconnected Computers Appear Connected

Make sure everything the user needs is local

If it normally was remote to the portable, pull it onto the machine

Caching, replication, hoarding

We'll have more to say about that later

Handling Limited Connectivity

E.g., modem or wireless links

You have some bandwidth, but a lot less than usual

What should the system do?

- Limitation of transmissions
- Spooling of transmission
- Prioritization

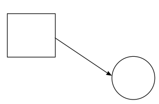
Handling Communications While Mobile

What if someone's sending you a lot of data?

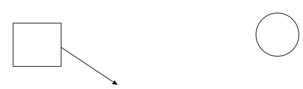
But you're moving fast

How can you make sure all of your data gets delivered?

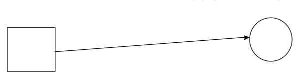
Communicating With a Mobile Machine



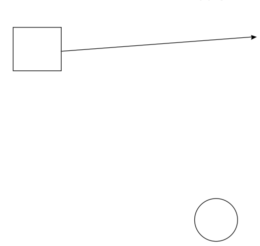
Communicating With a Mobile Machine



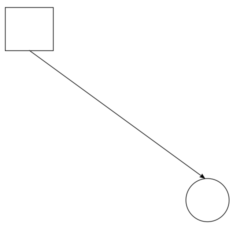
Communicating With a Mobile Machine



Communicating With a Mobile Machine



Communicating With a Mobile Machine



CS 239, Spring 1997 — Lecture 18, Page 19

Methods of Handling Rapid Movement

- Send data before it's needed (buffer)
- Track movement of machine and predict where to send future data
- Send data to more than one place

CS 239, Spring 1997 — Lecture 18, Page 20

What Are Proxies and Agents?

Programs working on your behalf

Why do I need those?

More precisely, what's different about them from any other program?

They operate semi-autonomously at remote sites

CS 239, Spring 1997 — Lecture 18, Page 21

The Proxy Model of Distributed Computing

The distributed system is a big, bad world

Rather than have the user (or his OS) worry about all the problems,

Assign handling of certain problems to proxies working on his behalf

CS 239, Spring 1997 — Lecture 18, Page 22

Examples of Proxies

- Handling limited bandwidth over a network
- Handling changes in network characteristics
- Buffering data for a disconnected computer
- Taking care of security needs

CS 239, Spring 1997 — Lecture 18, Page 23

An Example of A Proxy

A proxy that knows how to adapt HTTP transmissions for low bandwidth

Developed at Berkeley

A general-purpose proxy that calls *distillers* to adapt particular types of data

CS 239, Spring 1997 — Lecture 18, Page 24

Details of the Berkeley Proxy Solution

All WWW traffic goes through proxy

Proxy is knowledgeable about its client machine and the network

Proxy sits at boundary between good and bad connectivity

Proxy calls proper distiller for each piece of data delivered to client

Berkeley Distillers

Programs that can adapt a piece of data to match the capabilities of the machine and network

E.g., if you have a B/W display and little bandwidth, change color to grayscale

Must calculate what things will cost and what is possible

Difficult Issues in Proxy Research

- Application specific or not?
- Must the application be aware of the proxy?
- May proxies interact with each other?
- Where can proxies live?
- How do proxies get there?

More Proxy Research Issues

- If instantiated dynamically, how do you start up a proxy?
- What do you do when conditions change?
- How does mobile routing match up with proxies?
- In what ways may the proxy alter the data?

Yet More Proxy Research Issues

- What about security?
  - How much do I trust my proxy?
  - How much do I trust someone else's proxy?
  - What if I don't even know the proxy is there?

A Generalized Proxy Approach

Under development here at UCLA

Proxies/agents can live at any node

Agents assume no application-awareness

Several agents at different sites can cooperate to support one data stream

The collection of agents must be able to adapt

Handling Agent Generality

Build an agent framework

Capable of

- Setting up agents
- Establishing strings of agents
- Monitoring and responding to changes
- Accepting new agents

Prediction, Introspection, and Self-Organization

Operating systems "out of the box" are tuned for general purpose work

Since the seller doesn't know what you'll do with it

Most users are not capable of tuning systems

But can the system tune itself?

Related Motivations

- Self-organization for system administration
- Automatic response to changing system conditions
- Handling distributed systems

Some Sample Areas for Predictive Methods

- File system optimization
- Page caching algorithms
- Predictive hoarding
- Prefetching web pages

File System Optimization

The game in file system performance is to minimize physical movement in the disk

By trying to keep related disk blocks together

Previous FS Optimization Strategies

- Spacial locality
  - Keep blocks of same file, files in same directory together on disk
  - E.g., FFS
- Temporal locality
  - Keep blocks written at the same time close together
  - E.g., LFS

A Possibly Better Optimization Method

Neither of these are exactly what you want

What you really want is *referential locality*, and these are mere approximations

Referential locality - keep blocks that are referenced together close together

How To Achieve Referential Locality

Figure out what blocks are referenced together

Re-organize the disk so that they are stored close together

Sounds simple, right?

But . . .

Potential Problems in Obtaining Referential Locality

- Gathering the data
- Properly understanding the data
- Minimizing cost of data gathering
- Trading off costs of disk re-organization vs. minimization of access time

A Cousin System - Seer

Seer is a working system using predictive methods

For a very different purpose, but with many relations

Seer hoards files for disconnected mobile computers

The Problem Seer Solves

When a portable computer is disconnected, its disk should store all the files the user will need

But the disk can't store all the files he could potentially access

So what do you store on the disk?

The *hoarding* problem

The LRU Approach to Hoarding

Why not just use LRU methods to hoard files?

"Throw out" the least recently used candidates until it all fits

More precisely, hoard the most recently used files

Problems With LRU

- Doesn't always hoard what you need
- Hoard size to get what you need can be really big
  - Far off optimal

CS 239, Spring 1997 Lecture 18 Page 43

Another Approach - User Hints

Have the user create a list of what he wants hoarded

Either manually or with some tool

Perhaps combined with LRU methods

CS 239, Spring 1997 Lecture 18 Page 44

Problems With the Hint Approach

- Modern programs are very complex, so the user cannot possibly "get it right"
- The user-provided list tends to grow indefinitely
- Doesn't always hoard what you need

CS 239, Spring 1997 Lecture 18 Page 45

The Seer Approach

Figure out which files are related to each other

Gather them in clusters

Hoard the clusters

Using something like LRU on a cluster basis

CS 239, Spring 1997 Lecture 18 Page 46

Finding File Relationships

Many methods are possible

- Using directory membership
- Using various forms of links
- Deducing relationships from patterns of file accesses

Eventually, you need to reduce the relationship between files to a number

CS 239, Spring 1997 Lecture 18 Page 47

Challenges in Finding File Relationships

- Noisy data to work with
- Size of data
- Processing time to analyze data
- Lots of special cases in the real world

CS 239, Spring 1997 Lecture 18 Page 48

Building File Clusters

Once you know the relationships between pairs of files, how do you build clusters of related files?

Basically, run an algorithm that converts the relationships into clusters

Building such an algorithm that works well and fast is challenging

The Seer Experience

- It works really well
- Few hoard misses
- Seer's hoard size is very close to the true working set
- Space/time complexities and overheads are acceptable