Predictive Methods in Operating Systems

CS 239, Fall 1998
November 24, 1998

Peter Reiher

Outline

- What are predictive methods?
- Why are they important?
- Where can they be used?
- How do you use them successfully?
- Case studies in prediction
- Research areas in prediction

Operating Systems and Prediction

- Normally, operating systems work on command
- In some cases, an OS can improve situations by doing things before it's been asked
- The OS can *predict* what action to take

Benefits of Predictive Methods

- Performance, especially decreasing latency
- Preparation for changing circumstances
- Cost minimization
- Using systems resources when they're plentiful, rather than when they're in demand

The Ongoing Speed Crisis

- CPU speeds double every 18 months
- Disk speeds don't
- Neither do network speeds
- Applications of tomorrow must make effective use of the increased CPU speed
- So more and more work must be done without going to disk
  - Or waiting on the network

How To Handle Speed Mismatches

- Don't use the slow device
  - But we can't buy enough RAM to hold everything we want to store
  - And RAM isn't persistent
  - Besides, networked resources are increasingly important
- So use the slow device when it won't slow us down

Preventing Latency Disasters

- Effectively, we can't use the slow devices when the applications ask for them
  - Because the latency will kill fast applications
- So we've got to use the slow devices <u>before</u> the applications ask
- Which implies prediction

Can Prediction Save Us?

- Yes!
  - Most computers have lots of idle cycles
  - Most devices are idle most of the time
- So if we can ask for device service predictively, we can use the idle resources
  - While meeting the requests in time
- IF -
  - We can predict well enough

Other Benefits of Prediction

- Use a resource before it becomes unavailable
- Use a resource before it becomes more expensive
- Use a resource during its idle periods, rather than its peak periods

Where Can Prediction Help?

- Disk access
  - Both virtual memory and file access
- Network access
  - Both incoming and outgoing
- Scheduling of operating system resources

Prediction and Virtual Memory

- Virtual memory systems need to lower amount of paging
  - As speed differential increases, hit ratio must increase to match
- How to lower page misses?
  - Add more RAM
  - Better identify next page required

Prediction and Working Set Size

- A job's working set is the set of pages in current use
- VM system should allocate a job as many page frames as it needs for its working set
  - But no more, so other jobs can get their frames
- Working set sizes change over time
- Current VMs watch page faults and adjust working set accordingly

Can We Predict Working Set Sizes?

- Maybe
- Markatos [OS Review Oct 97] shows many processes have predictable patterns of working set size
- Predictive methods could recognize job and use these patterns to allocate page frames

File Access and Prediction

- Two important questions:
  - What files will you access?
  - What order will the blocks be read?
- For certain special cases, the first question is answerable
  - Is it for more general cases?
- For many cases, the second question is answerable

Prediction and Networks

- Web browsing is one obvious application
- Others are possible
  - File downloading
  - Predictive sending of high bandwidth data
  - Predictive use of the network before a disconnection

Predicting Use of Operating System Resources

- Predictive management of buffers
- Predictive creation and allocation of processes and threads
- Predictive handling of various cached resources
  - E.g., NFS caches

Predictive Mechanisms

- Caching
- Pre-fetching and pre-sending
- Hoarding
- Optimistic methods

Caching

- Caching is essentially predictive
- Predict that information used in the recent past will be needed in the near future
- Allows virtual memory systems to operate closer to memory speeds than disk speeds
- And allows other fast devices to mask slow devices

Cache Handling

- Store everything that's been used recently
  - On the assumption we might need it again
- But cache space is usually limited
- If you can't save it all, what should you save?
- Obviously, save what's really going to be needed soon

Least Recently Used Methods

- A primitive, but highly successful, prediction method
- If something must be thrown out from the cache, throw out the least recently used
- Predicting that more recently used stuff will be used again soon
- Often right, but . . .

When Does LRU Fail?

- When cached data is mixed
  - Some is reused, some isn't
- When more complex patterns of usage are occurring
- When none of the data is reused
  - But we can predict what will be needed next

Pre-Fetching

- Figure out what the program will need
  - Fetch or send it before the request
- File prefetching
  - OS predicts a particular file will be needed soon
  - So open and read the file before requested
- Web pre-fetching an obvious winner

Pre-Sending

- Push, rather than pull
- Someone else figures out I'll ask for something
- And they send it to me before I ask
- Possible technique for network file servers
- Or Web servers

Hoarding

- Get everything you need before it's too late
- Typically important for mobile computers
  - They are frequently disconnected
- Also important for economic reasons
  - If the network is cheap now, expensive later, get what you need now
- Related to pre-fetching
  - But some different characteristics

Optimistic Methods

- Actually perform calculations before the system is sure of them
- Assume or guess what's going to happen
- Proceed accordingly
- Check if you were right
- Fix up problems if you were wrong
- Already successful in hardware and specialized systems

How To Predict?

- All these methods can only work if we predict successfully
- Begging the question, how do we predict what will happen next?
- A very fruitful area for research
- But there's one time-honored answer

Prediction Based on Past Behavior

- The core of almost all successful prediction techniques in any field
- Based on careful observation of the past, obtain insight into likely future behavior
- Observe ongoing activities and match them against the insight of the past
- Take action accordingly
- And watch what happens, to learn more

Mechanics of Prediction

- Observation of activities
- Data analysis
- Clustering and other similarity methods
- Making predictions
- Applying prediction and handling errors

Observation and Prediction

- If you're going to predict based on the past, you need to know what happened
- Thus, you must observe system activity
- Non-trivial, since lots going on
- And the observation must not get in the way of the real work

Data Analysis

- Typically, you observe more than you can permanently store
- And the observations are full of meaningless noise
  - Which is anything that won't help prediction
- And you need to make predictive decisions quickly

Data Analysis Techniques

- Noise filtering
  - Possibly requiring some semantic knowledge
- Data reduction
  - Summarizing lots of data while keeping key predictive content
- Merging data
  - Since the process is ongoing

Clustering and Similarity Recognition

- Prediction is often aided by uncovering relationships
  - Between files, pages, disk blocks, processes, Web pages, etc.
- How to reduce raw data into format that describes relationships?
- Clustering techniques offer one method
  - Gather all observations into a few clusters

Clustering Algorithms

- Typically, clustering algorithms use some distance metric between data points
- Algorithms then either build new clusters from data or split off new clusters from old clusters
- Challenging to get good results with large numbers of observations

Other Similarity Recognition Techniques

- Markov chain techniques
  - Giving probability that some event will happen given others
- Pattern matching techniques
  - Are two events similar or not?

Making Predictions

- Sometimes trivial, once you have data
- Sometimes using specialized algorithm
- Analytic models sometimes helpful
- Real-time simulation useful in some cases
- Expert systems can be useful

Applying Prediction and Handling Errors

- Once a prediction is made, you must do something with it
- Typically issuing commands through normal OS interfaces
- But what if the prediction is wrong?

  1). Live with it

  2). Don't do anything till you're sure

  3). Be prepared to undo errors

Living With Predictive Errors

- Acceptable for caching situations
- Or situations in which pre-computation was harmlessly discarded
- Less acceptable if the results of prediction must be used
- Or if incorrect prediction means the application cannot proceed
- Or if you wasted too many resources

Waiting Until You're Sure

- If you pre-compute based on an assumption, wait for confirmation before applying results
- You've still reduced latency by the computation time
- Not always fast enough
  - Or even always possible
- May be wasteful of resources

Undoing Errors

- Save enough state to undo incorrect predictions
- Not always possible
- Usually not cheap
- Generally requires garbage collection
- But feasible for some cases

Case studies in prediction

- Seer
- Smiley
- Time Warp
- File pre-fetching

Seer - A File Hoarder for Mobile Computers

- Mobile computers are often disconnected
- When disconnected, all file requirements must be met from their local disks
- A portable's disk can hold only a fraction of the files available on a LAN
- So, which files should be stored on the mobile computer?
- Ideally, exactly those required

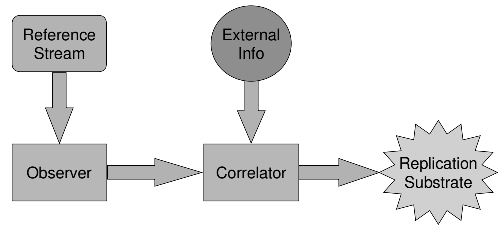
Hoarding Files for Mobility

- Why not have users do it?
  - File interrelationships are complex
- Why not use LRU?
  - Takes up too much space
  - Easy to construct cases where it fails
- Need some more automated prediction facility with greater "intelligence"

The Seer Approach

- Discover file relationships
  - By watching file accesses
  - Remove noise from the access data
- Form clusters of related files that represent projects
- Hoard only complete projects

Seer System Diagram



Discovering File Relationships

- Observe user's reference stream
- Calculate *semantic distance* among files
  - Represents semantic relationship among files
  - Based on positional interval between references
  - A per-file-pair metric

Calculating Semantic Distance

- Consider only position in the stream of references
  - Not time of reference
- Consider references (opens) as an ordered sequence
- Count the opens between the two files in question

Complications of Semantic Distance

- Considering multiple data points per file pair
- Interspersed activities
- Meaningless activities
- Space and time complexity of calculation

Handling the Complications

- Multiple data points per file pair
  - Average over many runs
- Interspersed activities
  - Average over many runs, use process tree
- Meaningless activities
  - Identify them and filter them out
- Space/time complexities
  - Only keep distance for nearest neighbors

Forming Clusters in Seer

- So you've got a bunch of distances between file pairs
- Which files are part of a single project?
- Try to cluster the most related files together
- Problem of overlapping clusters

Seer Clustering Algorithm

- Based on Jarvis & Patrick, 1973
- Similarity based on shared neighbors
- Pass 1 (cluster formation): combine clusters when two members have more than n_1 shared neighbors
- Pass 2 (generating overlaps): overlap clusters when two files have more than n_2 ($<n_1$) shared neighbors

Which Projects to Hoard?

- Seer currently uses Most-Recently-Used techniques to choose clusters
- But all files in a cluster hoarded regardless of recency of access to any given file
- Only complete clusters are hoarded
- More complex predictive answers to this question are possible
- Replication used to copy files

How Well Does Seer Work?

- Live use: 96% of all disconnections successful
  - Remaining 4% were mostly minor problems
- Simulation: hoard size needed is within 5-73% of optimum
- LRU 180-1087% over optimum

A Cautionary Tale

- Seer works really well
- Seer uses clustering to group related files together
- Experience during development suggested heuristics that would help Seer behave
  - E.g., always hoarding frequently used files
- How much of Seer's success was really due to clustering?

Investigating Causes of Seer's Success

- Seer's behavior can be replicated by running traces through its clustering algorithm
- We can alter the clustering algorithm to observe different effects
- We ran tests to isolate the effects of clustering and different heuristics

What Makes Seer Work?

- Not clustering
- The success of Seer's prediction is purely based on heuristics!
  - Clustering is worthless overhead
- With caveats:
  - Only certain kinds of traces tested
  - So other behaviors may benefit from clustering

What Should We Learn From This Cautionary Tale?

- Prediction is good
  - Seer works well
- The simplest methods worked much worse
- But complicated prediction is not always needed
  - Small adjustments to the simplest methods worked very well

Other Lessons From Seer

- Proper handling of large amounts of data is key, and tricky
- Real systems have characteristics that don't match theory - plan to use some heuristics

Smiley

- A Web browser pre-fetching tool
  - Combined with a feedback-augmented browser
- Based on past behavior, when a user requests a page, start fetching other pages
- Balancing probability of needing them, size of the page, and characteristics of network links

Time Warp

- A parallel discrete event simulation method
- Extremely optimistic prediction of when it is safe to run an event
- Requires complete rollback mechanism
  - Since it's so happy to guess wrong
- Successful in real use
- Important for prediction, as it's been heavily studied

File Pre-Fetching

- Many machines fetch most files from a server, on demand
- Much faster if the file is locally stored
- Various researchers have investigated how to determine which file to pre-fetch
- Usually using methods based on analyzing past file access traces
- Some systems very successful

Possible Areas for Research in Prediction

- File system layout
- Virtual memory handling
- Scheduling of reconciliation
- Control of automatic remote process execution
- Parallel processing applications
- Many others

When Is Prediction a Likely Winner?

- When low latency is vital
- When system capabilities are unbalanced
- When cycles and memory can be spared
- When tasks have a repetitive character
- When you can afford to be wrong
  - But want being right to be cheap