nnsched: A neural network based process scheduler

Radboud University Nijmegen, 12 June 2008

Peter Bex (peter.bex@xs4all.nl)

Problem statement

Computers are faster than a few years ago, but software is just as slow (sometimes slower)!

What can we do about that?

- Make faster programs (duh)
  - People want more fancy functionality, which cancels this
- See if we can prioritize programs better
  - Important programs should have priority

Priority

How do we know what programs are important?

That depends on the user:

- Multimedia is important to a home user
  - Networking performance can be sacrificed to make multimedia apps faster.
    - Example: Who cares if the next episode of "The Simpsons" finishes downloading 5 minutes later? You're watching the current episode right now and its playback must be smooth!
- On a web server, networking is more important than anything else!

Priority

How do we know what programs are important?

If we know what the user wants, we can look at a program's behaviour:

- Does it use networking?
- Does it output sound?
- Does it process keystrokes?
- Does it access the harddisk?

Use this information as features for a neural network which can instruct the OS scheduler

Scheduling 101

A CPU's task:

- Fetch instruction
- Execute instruction
- Update instruction pointer
- Repeat

We have only one CPU, so how come we can run more than one program at the same time?

Scheduling 101

Multitasking works like this:

- Run a tiny bit of program A
- Run a tiny bit of program B
- Run a tiny bit of program C
- Repeat

Switch very quickly, and it looks like they're running concurrently.

This is a lot like how cartoons work: quickly alternate pictures to make them move.

Scheduling 101

How do we prioritize?

When choosing the next program to run on the CPU, choose the program that's important more often.

- This is tricky to get right!

Existing OSes already allow you to define priorities.

- It's a lot of work for a user to define priority this for every program he uses
- Let a neural network perform this task

Implementation overview

Let’s solve this the easy way:

- Take an existing OS
  - Why NetBSD?
    - Free/libre software -> source is available
    - Clean design -> easy to learn and change
- Let a neural net determine priority
- Re-use existing priority controls
  - UNIX "nice" values

All we need to do is gather features and run the network!

- Sounds easier than it is

Implementation - kernel

Kernel extensions:

- Feature registration
  - Defines features
  - Makes it very easy to experiment with features
- Scheduler advisor
  - Every second, recalculate priorities by running registered feature values through net
- Feature monitor "device": /dev/nnfmon
  - For obtaining training data
- Network upload: /dev/nnconf
  - User can upload new networks into a running kernel

Implementation - userland

Userland (normal programs):

- Training program
  - Produces networks from feature data
- Testing program
  - To check network performance
- Configuration programs
  - To upload networks, fetch features from the kernel

And various other utilities

Features

Initial testing was done with the following features:

- Terminal reads, writes and read/write ratio
- Audio reads, writes and read/write ratio
- Network reads, writes and read/write ratio
- Disk reads, writes and read/write ratio

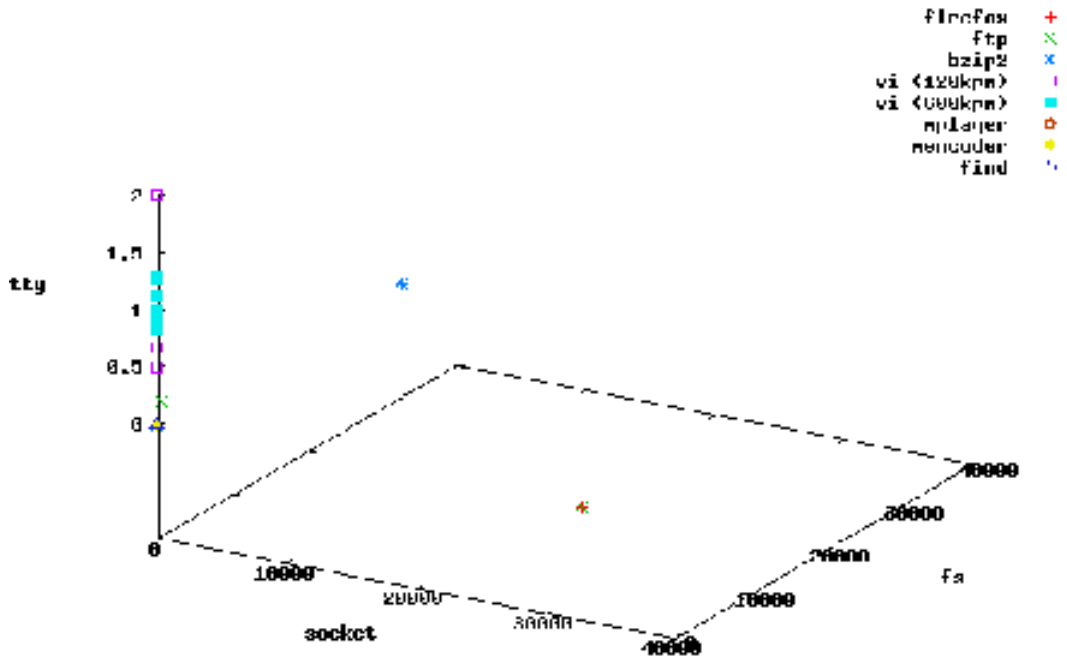
read/write ratio: hard to calculate with network

- Turned out to be a useless feature

Features - ratios

read/write ratio: hard to calculate with network

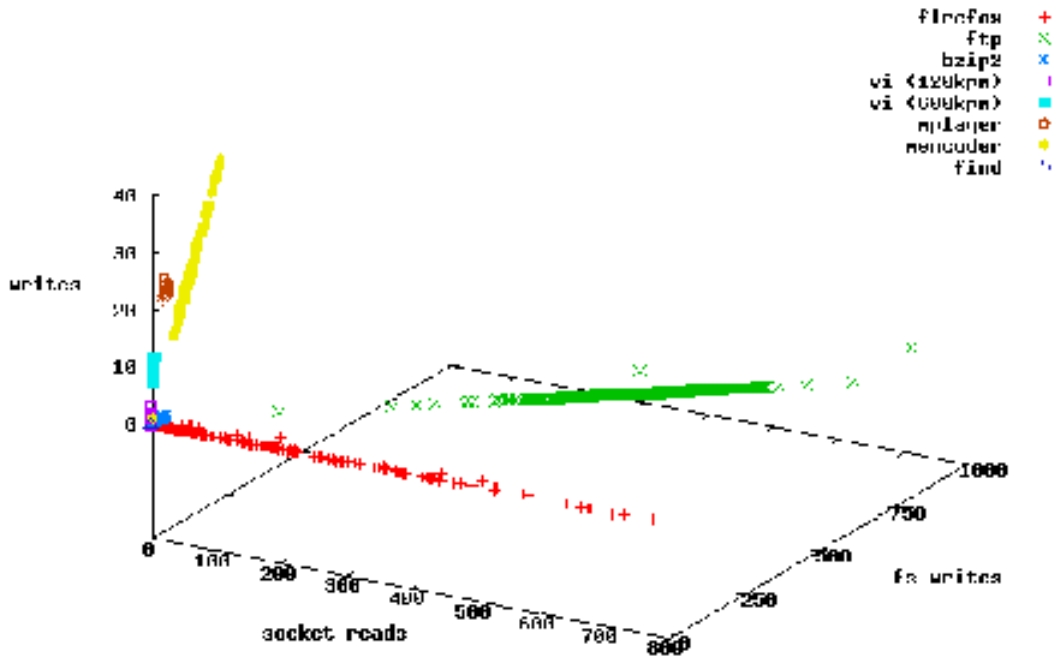
- Turned out to be a useless feature



Features - others

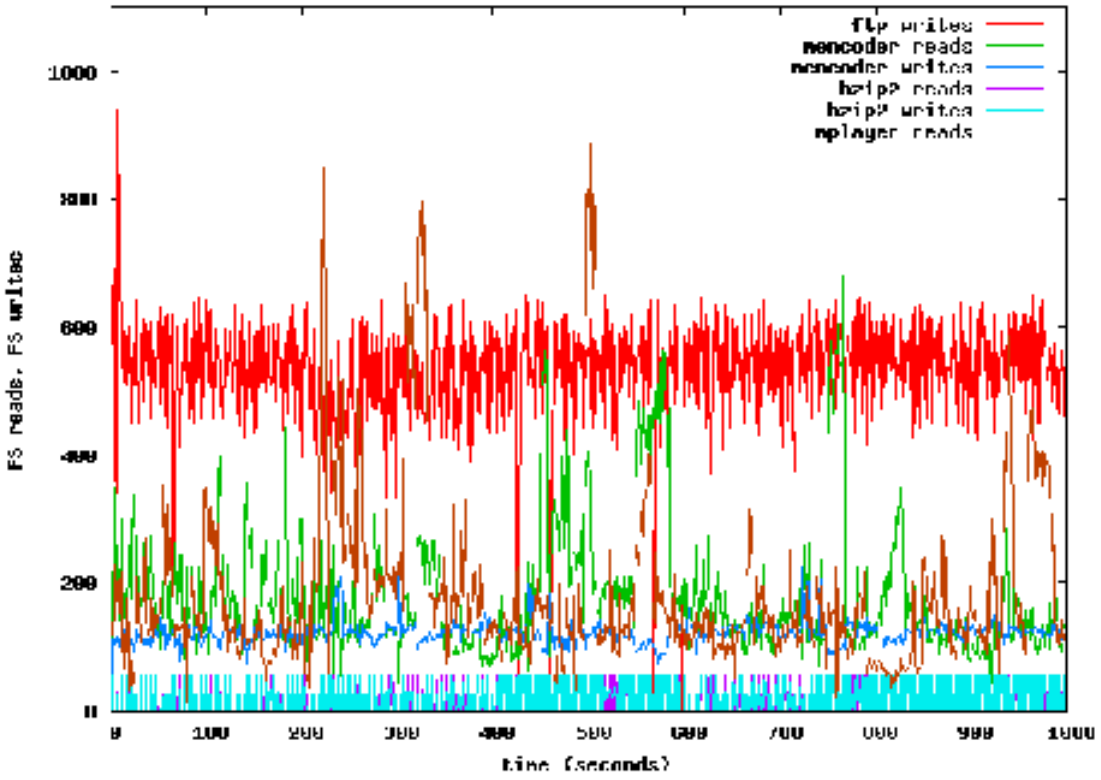
Other features: very useful

- Reads or writes, doesn't matter which



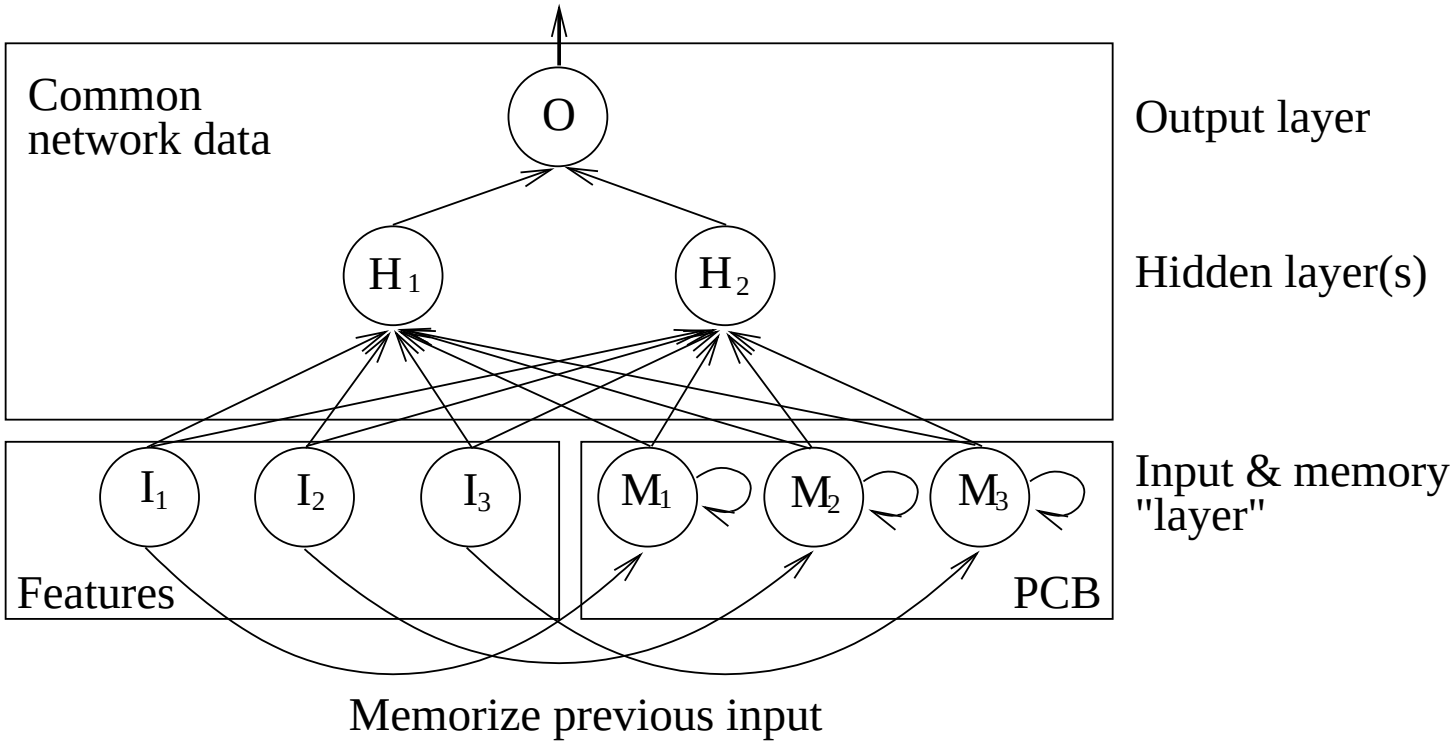
Features - in detail

Clusters from the previous slide make sense, but the real-time data is very chaotic:



Network topology

To smooth out the features a bit, a "memory" layer was added to the network:



Results

Project was a reasonable success:

- Network takes work out of user's hands
- User-friendly: just load a pre-trained network
- Very easy to experiment with features
- OK performance overhead

Room for improvement/future reasearch:

- Better features/more feature research
  - Integration with the X windowing system (HARD!)
  - At least adding features is very easy!
- Faster training algorithms (quickprop, rprop, ...)
- Different networks (SOM, ...)

Demo

Thank you

Code and master thesis available from
http://nnsched.sourceforge.net

Questions?