Sample EE699 Project Presentation

Hank Dietz
Professor and James F. Hardymon Chair in Networking
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506-0046
http://aggregate.org/hankd/

My Project: `chess`

- Bell Labs *Belle* was best chess program...
  - Good because it was very fast, not very smart
  - Distributed with PDP11 unix as optimized machine code
- John Buck "reverse assembled" it to optimized C
  (to port to unix on Gould/SEL computers)
- I optimized the code and ported it to CP/M, XENIX
- That was twenty years ago! What can it do now?

New Life For Old Code?

- Found a copy of the CP/M version on a floppy....
- K&R C code with very loose typing
- Added option to play itself (for benchmarking):
  - Tests run using GCC and 1GHz Athlon 4 under Linux
  - Self play ends in forced mate after 57 moves
  - Default 2-ply search takes 7.67s for the full game
  - Potential for very deep search is quite good...

What's Slow?

- Changed `Makefile` to use `-pg`
- Gprof results inconsistent; too many too fast routines
- Typically, move generation is at the top:

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 24.65      1.23     1.23   596648     2.06     2.06  wgen
 20.84      2.27     1.04   701510     1.48     2.48  bgen
  8.02      2.67     0.40  5765268     0.07     0.07  bslide
  6.01      2.97     0.30  9100589     0.03     0.03  b_try
  5.01      3.22     0.25   939226     0.27     0.47  battack
  4.81      3.46     0.24  4258414     0.06     0.06  wadiag
  4.41      3.68     0.22   635062     0.35     3.41  bquies
```

Optimizing The Move Generator

- Not much code in move generator to optimize...
- `slide()` and `try()` routines inlined
- Global pointers to board positions and move list; replaced references/updates by local registers

What To Optimize Next?

- Converted everything to correctly-typed ANSI C; can compile with aggressive GCC optimizations
- More `register` and `inline` stuff....
- Move sort routine is a major delay source; called `qsort()`, but bubble, not quicksort; converted to optimized insertion sort

Did it all work?

% time	cumulative seconds	self seconds	calls	self us/call	total us/call	name
22.27	0.47	0.47	347720	1.35	1.35	bgen
22.27	0.94	0.47	338600	1.39	1.39	wgen
9.48	1.14	0.20	294564	0.68	2.22	bquies
9.48	1.34	0.20	290793	0.69	2.22	wquies
7.58	1.50	0.16	2307030	0.07	0.07	warank
5.69	1.62	0.12	145023	0.83	5.92	bplay1
4.27	1.71	0.09	2099063	0.04	0.04	badiag
4.27	1.80	0.09	194240	0.46	0.46	attack
3.79	1.88	0.08	144368	0.55	5.65	wplay1
2.84	1.94	0.06	1948866	0.03	0.03	barank
2.84	2.00	0.06	13119	4.57	10.29	mater
1.90	2.04	0.04	2569568	0.02	0.02	wadiag
1.42	2.07	0.03	9792	3.06	9.82	xmater

The Final Results

- Removed `-pg` from CFLAGS
- CPU time goes from 7.67s to 1.80s...
  4.26x speedup!
- Surprises:
  - No data structure changes
  - Only algorithm change was sort
  - No machine-specific changes