

Faster Programs = Happier Users

David McKinnis

SureTech.com

davidmck@suretech.com

www.suretech.com/Performance_Talk_Slides

Who Am I?

- * David McKinnis
- * Co-Founder, CTO of SureTech.com

BTI 5000

Apple II

Macintosh

Mac Word 4.0

Windows 1.x

Word for Windows 1.0

Windows 3.x

Mac Word 5.0

Mac Word 6.0

Office 95

Windows Installer

Office 2000

PHP

MySQL

iPhone

If I am successful

- * You will have some new tools in your toolbox:
 - * Tools to measure performance
 - * Strategies to improve performance
- * You will teach me something



Fast Applications Take Effort

- * Blaise Pascal

I would have written a shorter letter, but I did not have the time.

- * Everything is working to slow things down—

- * New Features

- * Every new line of code

Example #1

SureTech.com

800.882.8701 HOME | NEWSLETTER | CUSTOMER SUPPORT | CONTACT US | HELP/ME | register | log in

SureTech.com SOLUTIONS ABOUT US NEWS

Google™ Custom Search

SUREOFFICE

IT SOLUTIONS FOR YOUR WORKFLOW

get help TODAY
Lost data? Broken Server?
Too much spam
Network or security trouble?

Go

solutions for TOMORROW
Optimize powerful solutions,
flexibility, security and cost

Go

GET OUR FREE
INSIGHTS newsletter

SIGN UP NOW

contact us

News (more...)

The Future Is Here
Virtual Reality through a computer is nothing compared to (skip the first 50 seconds)...

Software Release (and a little)

Internet | Protected Mode: On

Why Performance is Important

- * It affects how people view your application
- * It affects how many people run your application
- * It affects how people view your company
- * It affects your success

The Process for Making Performance Improvements

1. Decide what scenarios to look at
2. Measure a scenario
3. Decide which scenarios are too slow
4. Figure out what's slowing you down
5. Make some changes that should improve it
6. Measure again
7. If there's no change, remove your fixes – they are most likely buggier than what was there before.

How do you decide what to work on?

- * What scenarios are important to your users (or you)?
- * Identify a few key ones (to start with)
- * Listen to complaints from your users
- * Find areas that aren't being used as you would expect
- * Time those and see which ones are too slow
- * Periodically go back and notice if there are significant increases

Scenarios Looked At

- * Loading home page with empty cache
- * Loading home page with full cache

How do you measure performance

- * Think about the whole scenario – beginning to end. What does the user see or perceive?
- * Understand who your users are and what they are trying to do
- * Understand the conditions under which your users use your product
- * Clearly define hardware, setup and initial conditions (make sure you can recreate those later)

Tools I Used

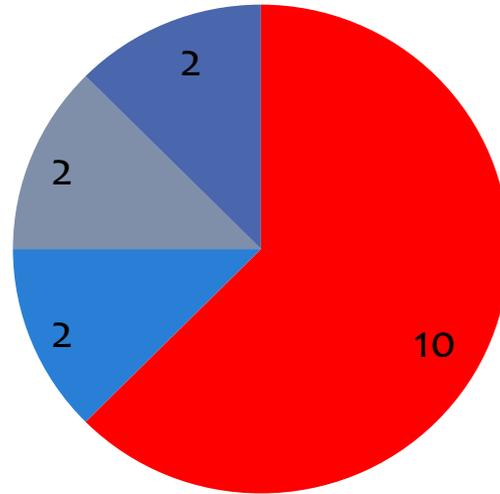
- * FireBug
- * YSlow
- * Fiddler

What I Discovered

- * Multiple JS and CSS files
- * Nothing cached on the client side
- * Lots of graphics and a video being loaded

Which Part Do You Work On?

Total Time - 16ms

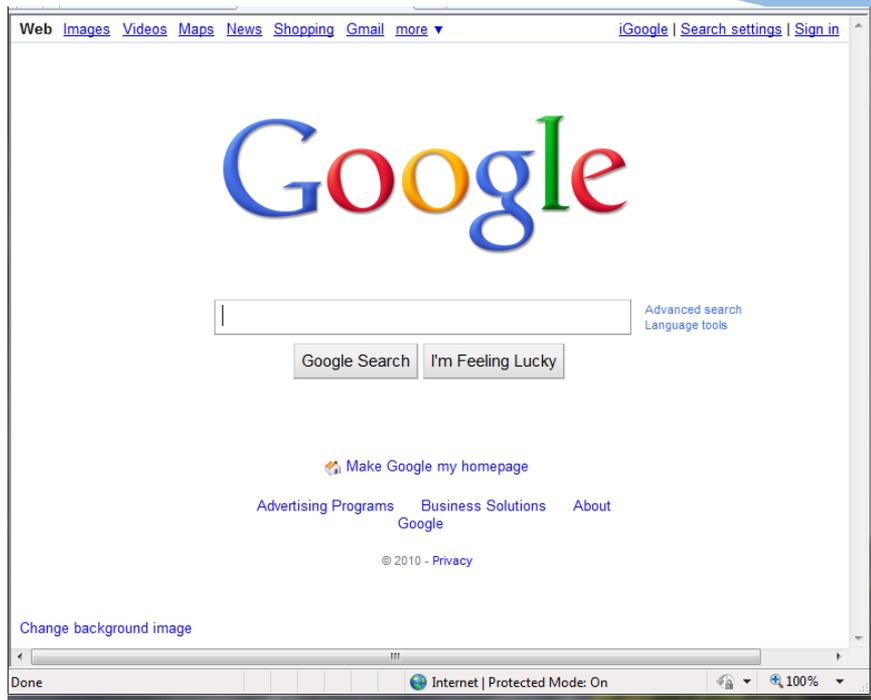


- Function 1
- Function 2
- Function 3
- Function 4

Five Ways to Improve Speed

- * Do Less work
- * Do things at a different time or place
- * Don't repeat work
- * Make your code more efficient
- * Increase the capacity of your hardware

Doing Less Google.com vs SureTech.com

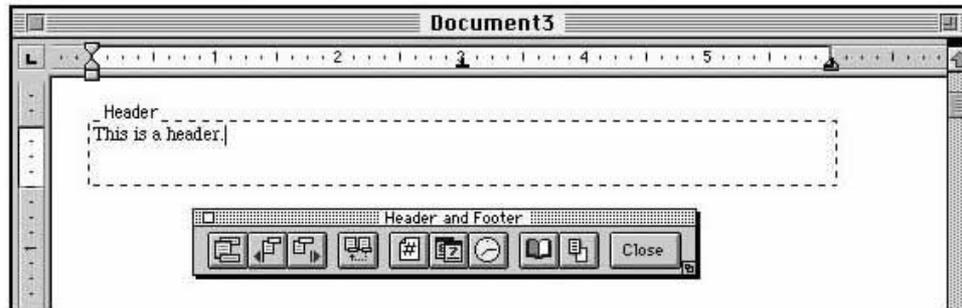


Improvements

- * Use Apache settings to cache on the client side
 - * Don't repeat work
- * Combine and compress CSS and JS files
 - * Increase the capacity of your hardware
 - * Don't repeat work

Example #2

Mac Word 6 & Word 95



Scenarios Looked At

- * Mac Word 6.0
 - * Layout
 - * Boot
 - * Find
- * Word 95
 - * Boot

Tools to Measure Performance

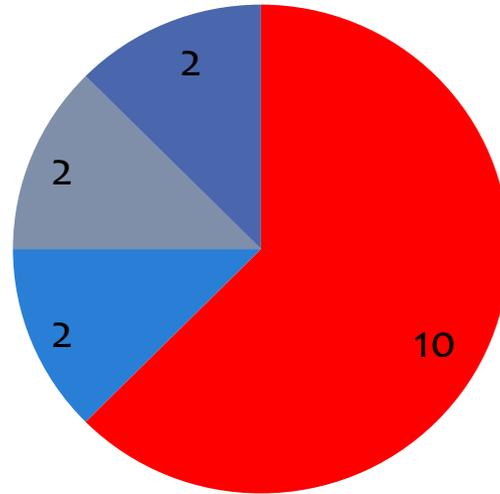
- * Stop Watch
- * C Language Profiler

Profilers

- * Should show you
 - * How many times a function is called
 - * How much time that function takes
 - * How much time that function and it's children take
- * Helps to find bugs
- * Various types of profilers
 - * Code profilers in your development environment
 - * Client-side profilers like YSlow or Firebug

Which Part Do You Work On?

Total Time - 16ms



- Function 1
- Function 2
- Function 3
- Function 4

Problems Found and Changes Made

- * Find
 - * Too many calls to check for user cancelling
 - * Solution - Stop polling the user so frequently
- * Layout
 - * Time critical portion of program
 - * Solution – Hand written assembly language code
- * Boot speed Mac Word
 - * Font enumeration taking too long
 - * Solution – Cache font information and reuse if possible

Problems Found and Changes Made (part 2)

- * Boot speed in Word 95
 - * Splash screen slowing down boot
 - * Solution – Cache bitmap of splash screen so calculations aren't repeated

Example #3

GoCrossCampus.com

The screenshot shows the GoCrossCampus.com interface. At the top, the logo reads "go cross campus win your world." with a castle icon. To the right, a navigation menu includes "rules", "faq's", "browse", "invite", "my games v", "blog", "contact", and a "BETA" badge. Below the logo, the game title "Drexel Tournament (Resimuturs)" is displayed, along with a timer "Turn 1 ends in: 0 min 47 sec". A "Recruitment Mode" notification states: "For the first 3 turns, you cannot attack opposing teams. Use this time to build your armies, place them strategically, and recruit teammates!". A navigation bar contains "game", "players", "superlatives", "history", and a "recruit teammates!" button. The main map area shows a city grid with colored territories (yellow, green, purple, red, orange, blue) and numbers. A "Refresh the map!" button is visible. On the right, a "Battle Plan!" section for Turn 1 indicates that no battle plan has been proposed. Below it, the "Place Armies!" section shows the user has received 1 army and is currently placing 1 army in the "University Crossings" territory. At the bottom, there are "Team Chat!" and "Public Chat!" buttons, and a status bar with "autorefresh is off" and a "refresh" button.

go cross campus win your world.

rules
faq's
browse
invite

my games v
blog
contact

BETA

→ Drexel Tournament (Resimuturs) Turn 1 ends in: 0 min 47 sec

Recruitment Mode: For the first 3 turns, you cannot attack opposing teams.
Use this time to build your armies, place them strategically, and recruit teammates!

game players superlatives history recruit teammates!

Refresh the map!

Battle Plan! for Turn 1

A battle plan has not been proposed for this turn.

Update!

Place Armies!

You have received:
 $(1 - 0) \times 1 - 0 = 1$ armies. **Hub?**

You are placing 1 of 1 armies.

Place 1 armies in:
University Crossings

→ click placement w/ another territory

Place armies

Team Chat! Public Chat! autorefresh is off refresh

Problems Found

- * Usage was not evenly distributed
- * Creation of map data was exceptionally slow
- * Queries to give users feedback on battle outcomes were extremely complex and slow
- * Slow response caused users to repeat their requests (which just slowed things down more)

Solutions

- * Created separate MySQL server
- * Moved emailing function to separate server
- * Moved chat function to separate server
- * Moved static items to separate server
- * Cached results of complex queries
 - * Stored in the database
- * Cached map data on a regular basis
 - * Using memcached

Any Additional Questions?

Useful Resources

- * Webpage Tools (in no particular order)
 - * FireBug <http://getfirebug.com/> (for FireFox)
 - * YSlow <http://developer.yahoo.com/yslow/> (for FireFox)
 - * Fiddler <http://www.fiddler2.com/fiddler2/>
 - * Developer Tools in IE8 (F12)
- * PHP Tools
 - * Xdebug - <http://xdebug.org/> - for profiling
 - * APC - Alternative PHP Cache – PHP Accelerator
 - * XCache - PHP Accelerator

Interesting Reading

- * Building Scalable Websites by Cal Henderson (a bit dated, but the basics are still solid)
- * Raymond Chen's blog I believe interesting even for non-Windows programmers - <http://blogs.msdn.com/b/oldnewthing>. This article in particular - <http://blogs.msdn.com/b/oldnewthing/archive/2010/11/04/10085797.aspx> talks about taking into account performance when designing a feature.