



# Making the Case for PHP at Yahoo!

Michael J. Radwin

[mradwin@yahoo.com](mailto:mradwin@yahoo.com)

<http://public.yahoo.com/~radwin/talks/>

# Speaker Info

- Michael J. Radwin
  - engineer for Yahoo! since 1998
  - technical lead for the Apache web server
  - co-leading the PHP crusade at Y!
- Contact Info:
  - [mrادwin@yahoo.com](mailto:mrادwin@yahoo.com)
  - <http://www.radwin.org/michael/>

# Outline

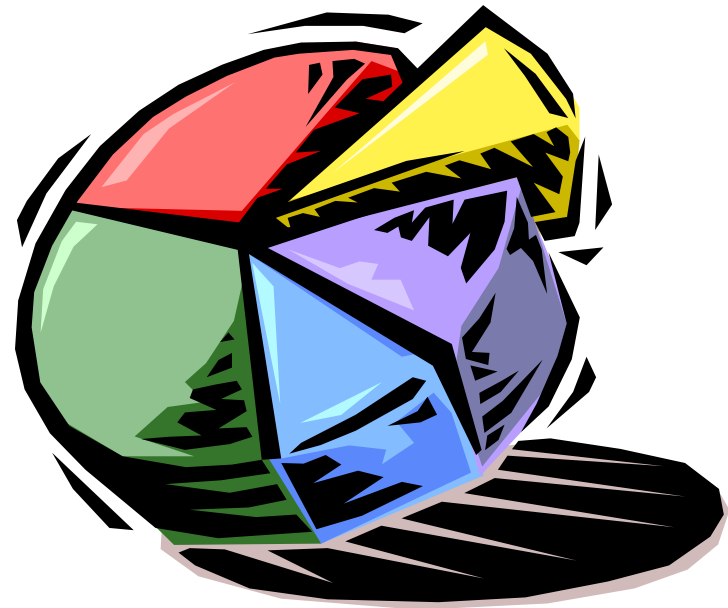
- Motivation
- History: from proprietary to Open Source
- Choosing a new server-side scripting language
  - what the ideal system would look like
  - languages we didn't choose
  - why we picked PHP
- Scaling PHP
- Lessons learned

# Motivation

What's so special about Yahoo!?

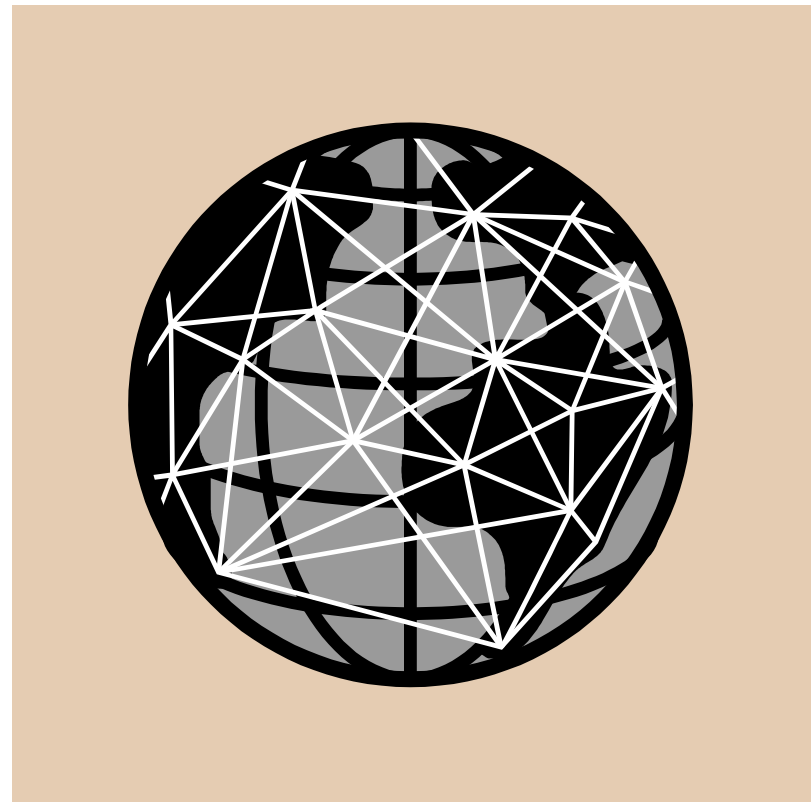
# World's Biggest Site

- World's most trafficked Internet destination
  - Nielsen//NetRatings 8/2002
- Users
  - 201M unique users
  - 93M active registered users
- Pageviews
  - more than 1.5 billion a day



# Huge Production Network

- 4500+ servers
- 16 co-locations
  - USA: Sunnyvale, Santa Clara, San Diego, Washington DC, Dallas
  - Intl: England, Central America, South America, Taiwan, Hong Kong, Singapore, China, Australia, India, Japan, Korea



# Complicated Software

- Site
  - 74 properties
    - mail, shopping, sports, news, games, pets, etc.
  - 25 int'l sites
  - 13 languages
- Code
  - 8.1M lines of C/C++
  - 3.0M lines of Perl
  - 612 developers



# More about Y! Server Software

It didn't start out so complex...



# Y! Server Software: 1994-1995



Early Years  
Static Content

- FreeBSD 2.1 (on Intel x86)
- *Filo server* and *Filo pages*
  - 676 lines of C
  - optimized for speed
  - HTML + ads
- CGIs for “dynamic” content
  - Search & Suggest A Site
- advertisements client/server
  - *yRPC* homegrown RPC

# Y! Server Software: 1996-1998



Dynamic Content  
Personalization

- FreeBSD 2.1 and 2.2
- Apache 1.1
- Lots of home-grown software
  - free stuff wouldn't scale, immature
- *yScript1* page Dynamic content
  - similar to Apache SSI
  - HTML + ads + personalization
  - content via include & DBM files
- advertisements client/server
- UDB (user data base)
  - NFS-mounted flat files

# Y! Server Software: 1999-2000

Mail 

GetLocal   
Maps

GeoCities 

SHOPPING 

Boom Years

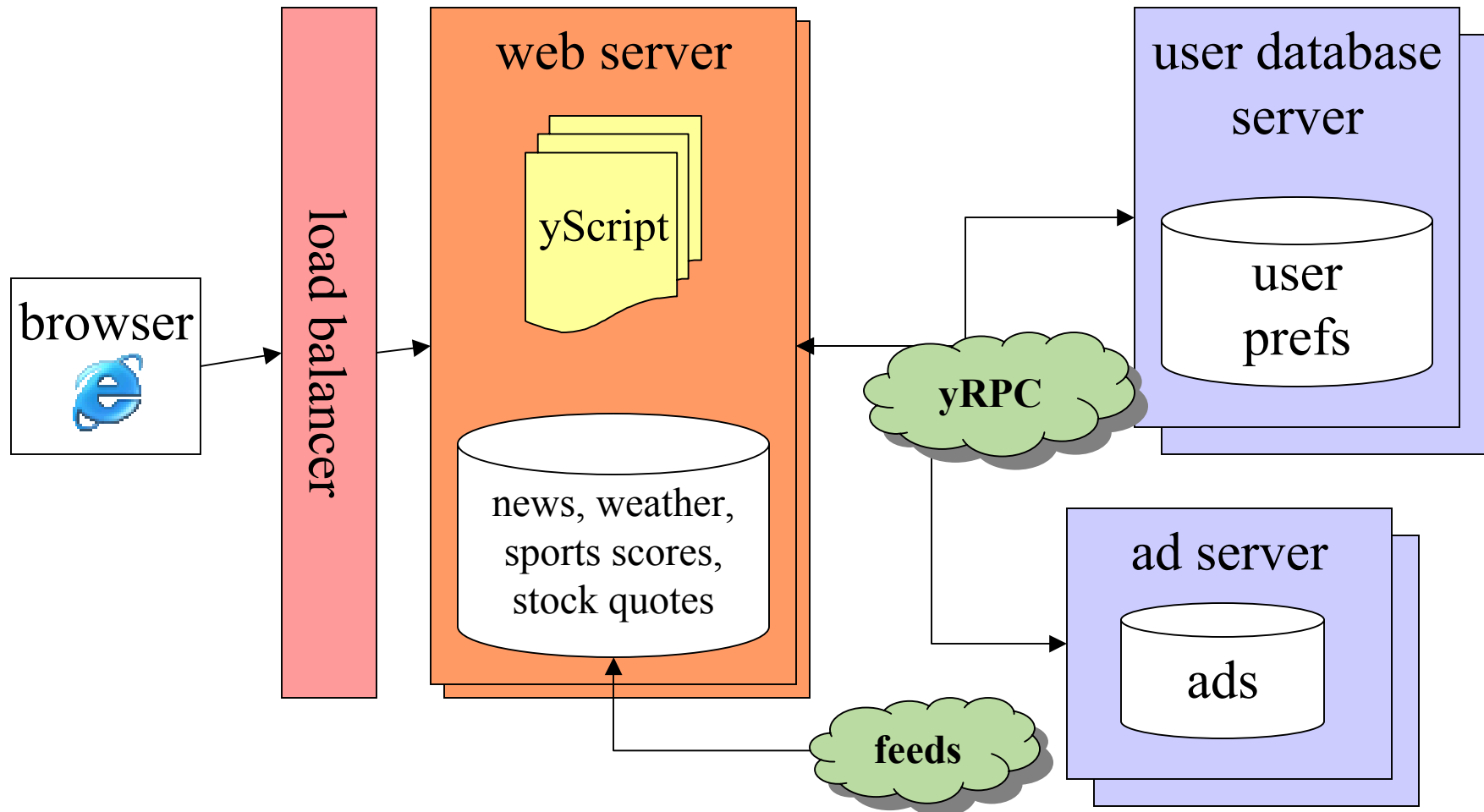
Communications, Commerce,  
Communities

- FreeBSD 4.1
  - a few Solaris boxes (Mail, Geo)
- Apache 1.3.x
- *yScript2* pages
  - like *yScript1*, but more powerful
  - interactive forms
  - business logic in C++
- mod\_python (Maps, YP)
- UDB goes client/server
  - *yRPC* homegrown RPC

# Tradeoffs: App Logic in C++

- Advantages
  - fast execution speed
  - strongly typed, mature language
- Disadvantages
  - edit, compile, link, debug cycle
  - not conducive to rapid prototyping
  - too easy to make mistakes with memory

# Example: my.yahoo.com



# Yahoo! in 2002

## Moving towards Open Source

# Yahoo!'s Open Source Paradox

- Open Source software runs our business
  - Perl
  - Apache
  - FreeBSD
  - GCC (+ GNU toolset)
- Yet we seem to build a lot of our own stuff, too
  - RPC
  - server-side page languages
  - databases



# Are We Re-inventing the Wheel?

- When Y! started in '94
  - free stuff did not scale
  - too immature
  - small community
- How about today?
  - performance
  - integration
  - legacy & inertia
  - “Not Invented Here” syndrome





# Costs of Proprietary Languages



- Maintenance
  - 3 different variants
  - C++ bugs
- Training overhead
  - engineers
  - design folks
- No integration
  - authoring tools, DBs
- Limited functionality
  - *yScript2* lacks subroutines!

# Moving to Open Source

- Open Source tech eventually matures
  - Y! replaced *Filo* server with Apache in 1996
  - replacing some DBM and Oracle with MySQL
- Server-side languages natural next step
  - features, performance, integration, community
- Y! is a cheap company
  - economic recession 2001-2002
  - can't afford to waste engineering resources

# Choosing a Language

How we ended up picking PHP

# Language Criteria

1. C/C++ extensions
2. loops, conditionals
3. complex data-types
4. pleasant syntax
5. runs on FreeBSD
6. high performance
7. robust, sand-boxed
8. interpreted (or dynamically compiled)
9. low training costs
10. i18n support
11. clean separation of presentation/content/app semantics
12. doesn't require CS degree to use

# Why not Apache mod\_include?



- Pros
  - built into Apache, easy to learn/use
- Limited language (no loops, subroutines)
- Doesn't interface with Y! code
  - Ads, User Database, etc.
- Poor performance
  - parses file every time you hit page

# Why not ASP or Cold Fusion?



- Pros
  - lots of 3<sup>rd</sup>-party integration
  - professional support
- Cons
  - CF has ugly syntax
  - \$\$ for languages
  - \$\$ for Microsoft Windows



# Why not Perl?



- Pros
  - FreeBSD support and performance is great
  - huge CPAN library
  - we already use it for offline processing
- Cons
  - There's More Than One Way To Do It
  - poor sandboxing, easy to screw up server
  - wasn't designed as web scripting language

# Why not JSP, Servlets, or J2EE?



- Pros
  - strongly typed
  - good performance (JIT), sandboxing
  - works w/lots of off-the-shelf software
- But... you can't really use Java w/o threads
- Threads support on FreeBSD is not great



# Why not XSLT or ClearSilver?



- Pro: separates HTML presentation from app logic
- XSLT
  - complicated to set up and understand
- ClearSilver
  - small developer community
- Neither is “procedural” language
  - totally different models from PHP/ASP/JSP/yScript2
  - difficult transition for Y! engineering

# So Why Did We Pick PHP?

1. Designed for server side web scripting
2. Large, Open Source developer community
  - integration, libraries
  - documentation & training
3. Debugging & profiling tools
4. Simple and clear syntax (fits Y! paradigm)
5. Performs well in our tests
  - efficient (with acceleration)
  - small enough memory footprint

# Benchmarking PHP

“But is it as fast as yScript2?”

# Performance Tests



A close-up, blue-tinted image of a table with numerical values. The table has a grid pattern with horizontal and vertical lines. The values are positive numbers with two decimal places, ranging from +1.062 to +5.000. The image is slightly blurred, giving it a grainy appearance.

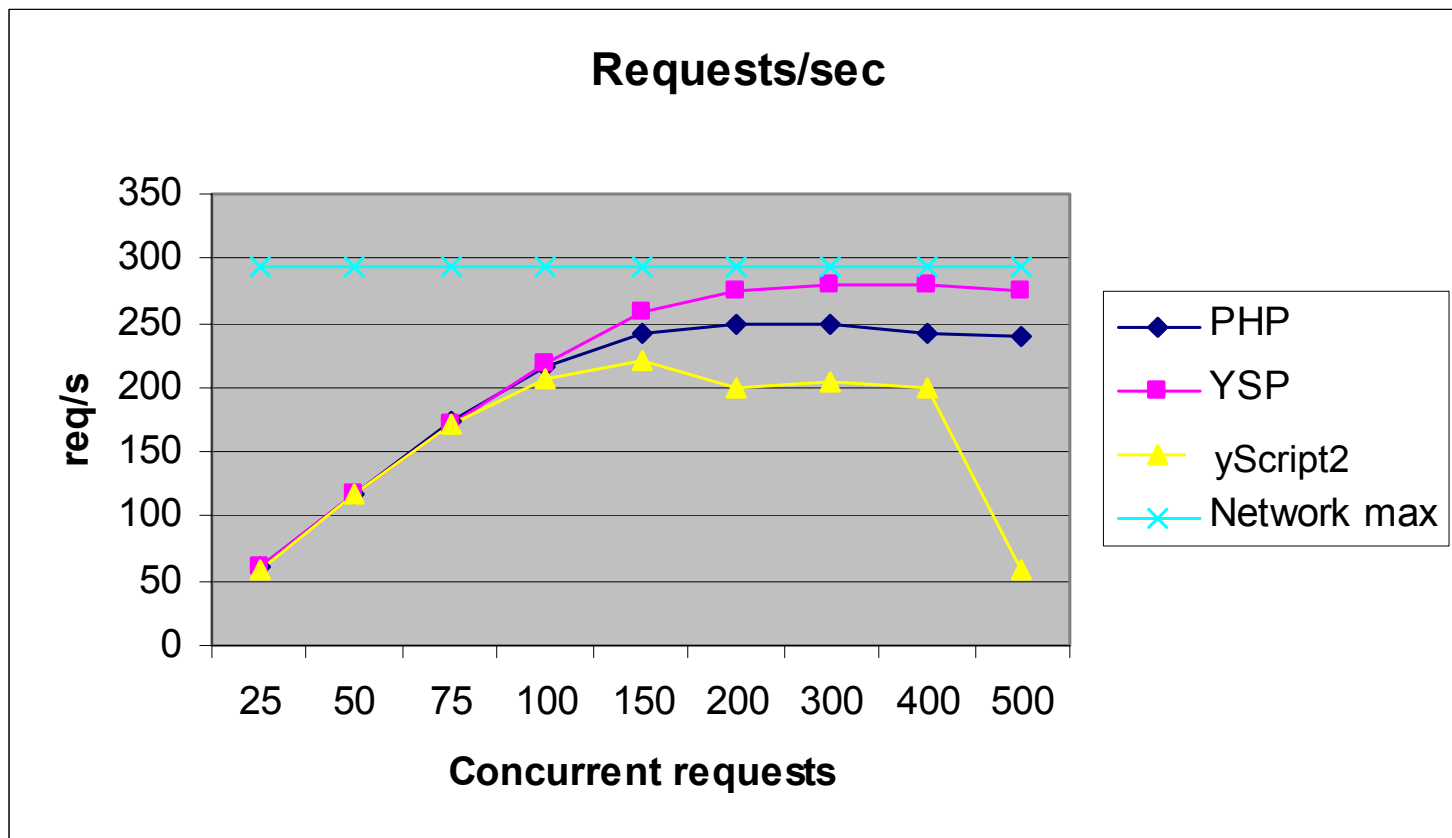
5	+2.688
0	+5.000
1	+1.500
0	+1.125
0	+1.062

- Languages
  - PHP 4.1.2 (w/Accel)
  - yScript2 (proprietary)
  - YSP (mod\_perl)
- Hardware
  - Pentium III 800Mhz
  - 512 Mb RAM
  - FreeBSD 4.3

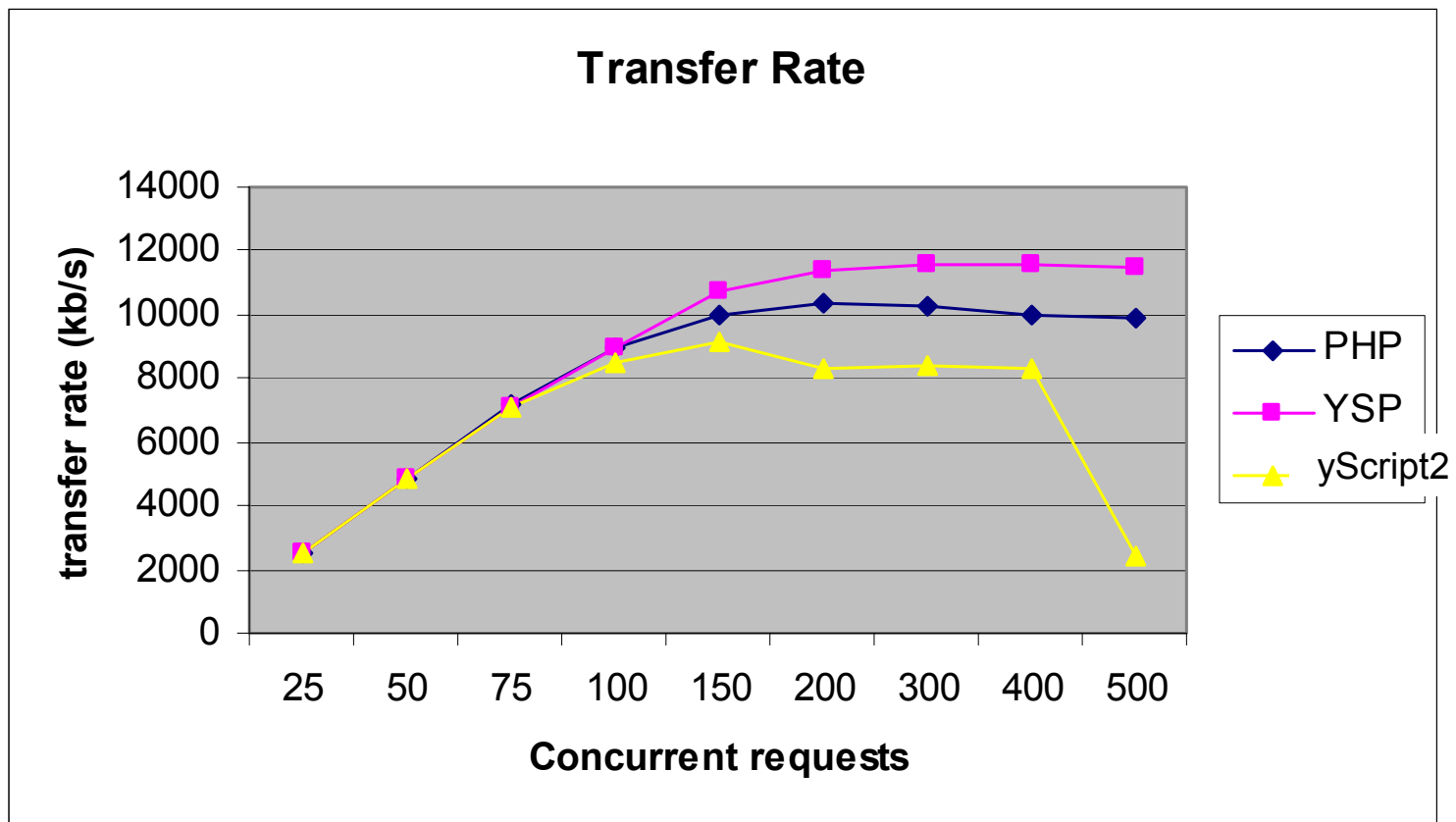
# Performance Tests

- 33K input script, 41K output
- Included and evaluated 3 other files
  - header, navbar, footer
- Echoed environment variables
- Pseudo-personalization
  - “Hello, mradwin”
- Called external C++ library for Ads/UDB
  - network delay to fetch data

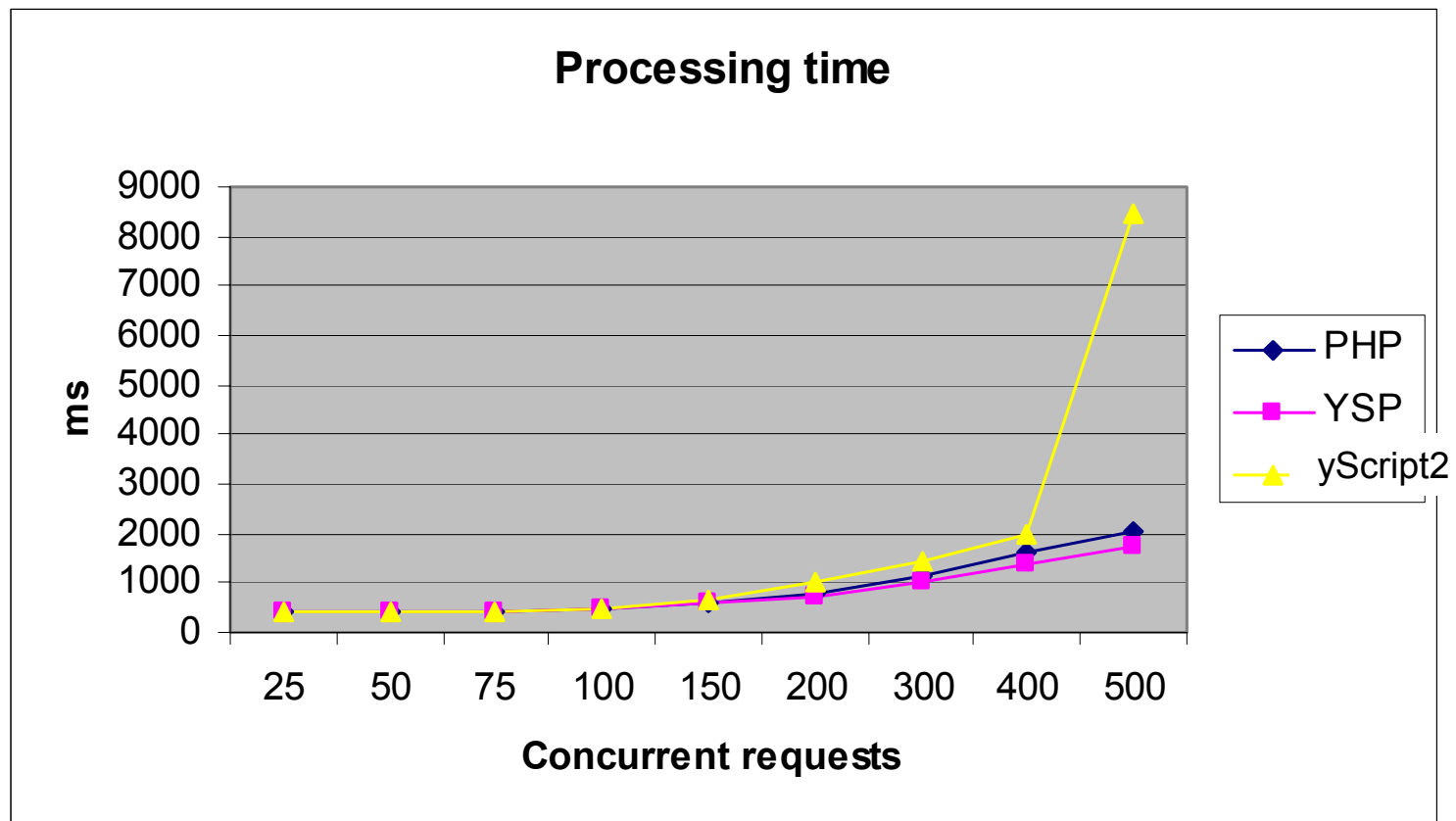
# Performance: Requests



# Performance: Transfer Rate

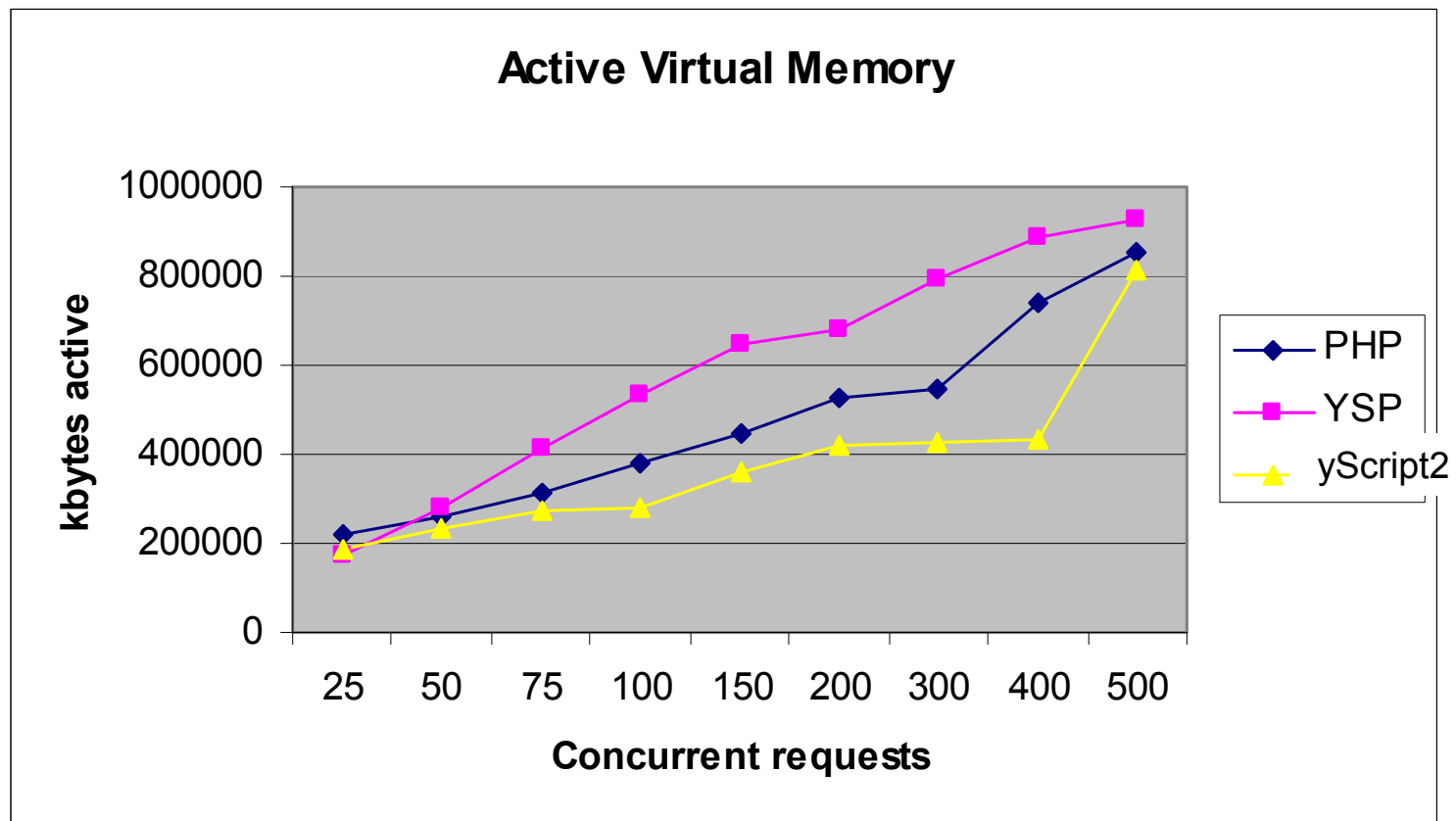


# Performance: Processing Time





# Performance: Memory



# Performance: Scaling PHP

- Profile your code

```
foreach ($_SERVER as $k => $v)
    if (substr($k, 0, 5) == "HTTP_")
        $str .= substr($k, 5) . ": $v\n";
```

versus:

```
if (strncmp($k, "HTTP_", 5) == 0)
```

- Implement C and C++ extensions
  - when you're willing to trade flexibility for speed
- Use an Accelerator



# Lessons Learned

4 months after we started using PHP

# Early Adopters

- PHP for new properties
  - remember.yahoo.com for Sep 11 2002
- Internal tools
  - content mgmt, package repository, aclviewer
- Most Y! properties integrating slowly
  - no plans to rewrite entire site
  - mix PHP, Apache DSOs, yScript1 & yScript2 pages

# Coding PHP Takes Discipline

- Shallow learning curve
  - very easy to get some pages up quickly
- But mixed app/presentation problematic
  - PHP code and HTML forever intertwined
  - coding conventions help
    - \*.inc for function and class libraries
    - \*.php for web pages (call functions, echo \$vars)
  - use Smarty to enforce separation?

# PHP != Perl

- The “implement twice” problem
  - much offline processing done in Perl
  - example: tax/shipping calculation for Shopping
- PEAR != CPAN
  - installer doesn't work in PHP 4.2.x
  - repository smaller, less mature than CPAN
- Surprises for people used to coding Perl

# Giving Back to Open Source

- We customize Open Source software we use
  - often improvements are not sent back
  - many are gross Y!-specific hacks
- Improving our relationship with OS community
  - FreeBSD (Peter Wemm)
  - Apache (Sander van Zoest)
  - PHP (Rasmus Lerdorf)
  - MySQL (Jeremy Zawodny)

# Questions and Answers

Slides online at:

<http://public.yahoo.com/~radwin/talks/>



# Legal Mumbo-Jumbo

- Text of this presentation is Copyright © 2002 Michael J. Radwin.
- Clip art is Copyright © 2002 Microsoft Corporation.
- Yahoo!, the Yahoo! logo, the “Jumpin’ Y Guy” logo, and other Yahoo! logos, product & service names are trademarks of Yahoo! Inc.
- The Yahoo! Engineering logo is Copyright © 2000 John “JR” Conlin.
- The PHP logo is Copyright © 2001, 2002 The PHP Group.
- The Open Source, Apache Feather, Active Server Pages, Cold Fusion, “Powered By FreeBSD”, mod\_perl, Apache::ASP, Mason, Java, W3C, Neotonic, and ionCube logos are Copyright © their respective owners.