Benchmarking Web Applications and Frameworks

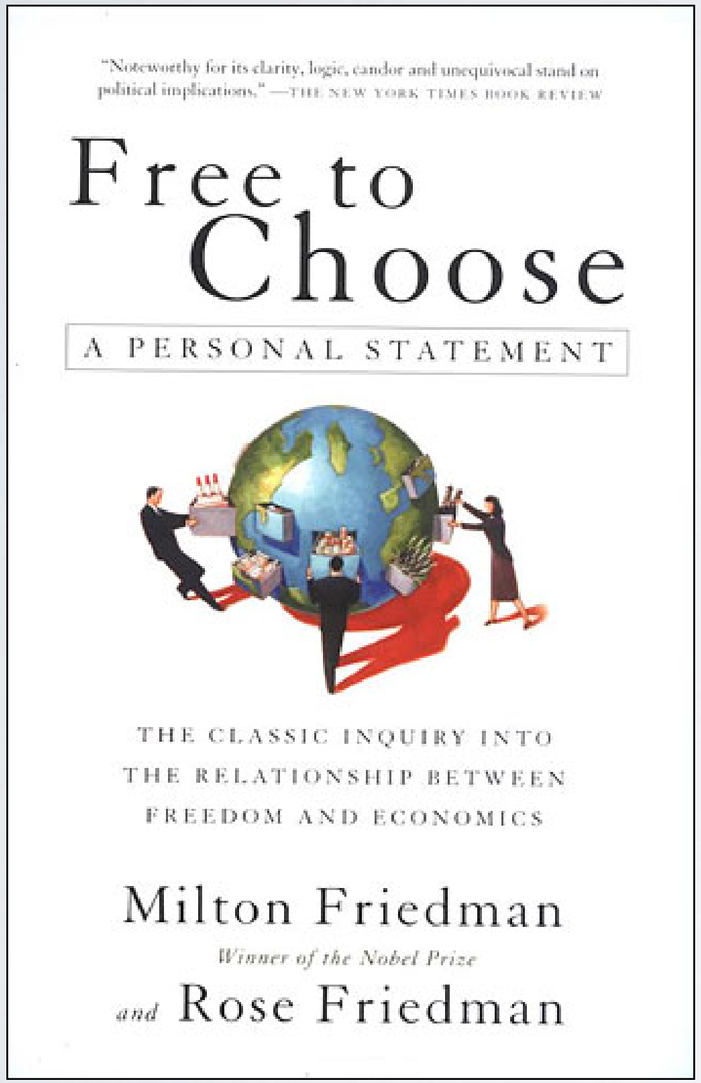
Paul M. Jones
paul-m-jones.com

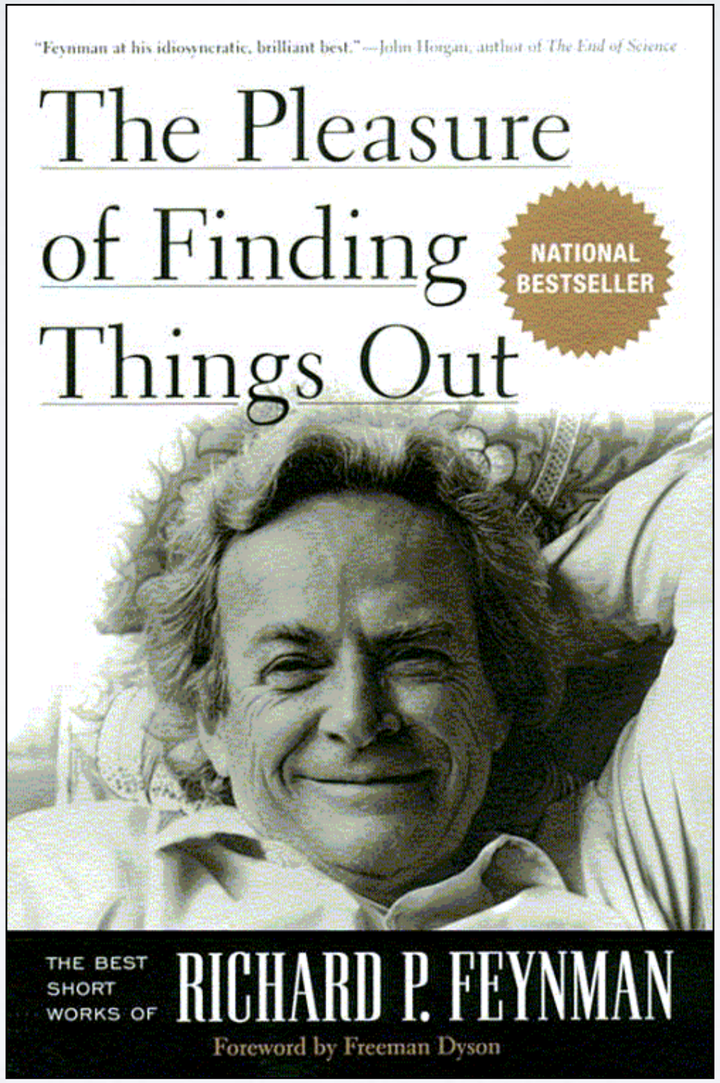
18 Oct 2011
http://joind.in/3741

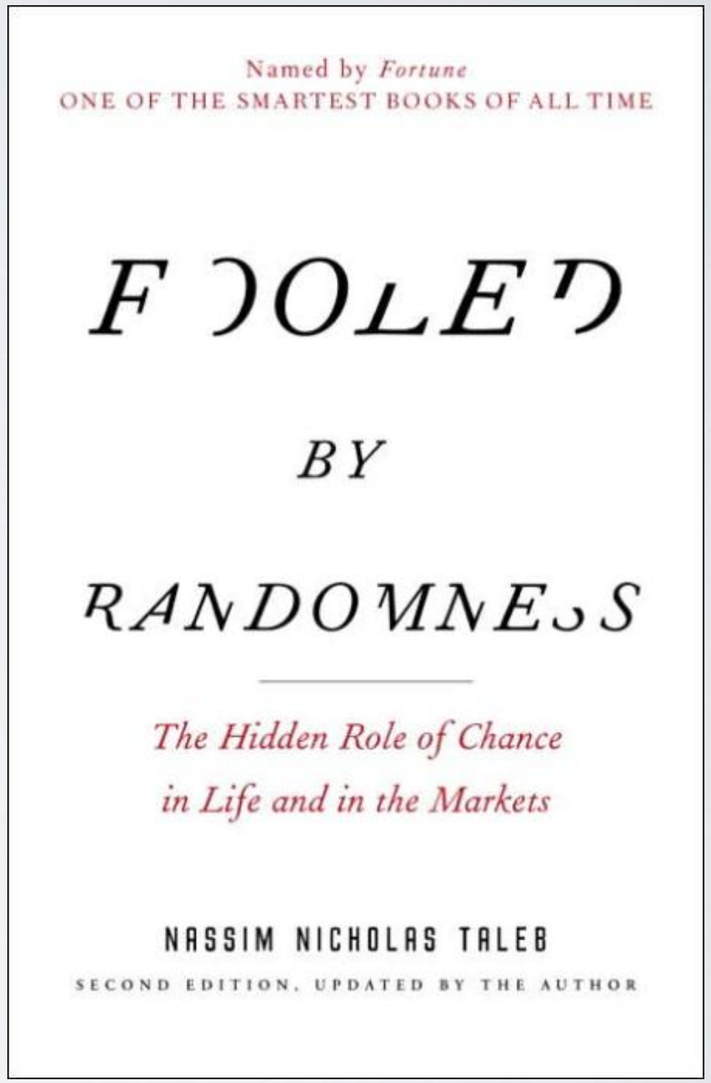
Overview

- Benchmarking methodology, results, and interpretation
- Resource usage prediction
- Engineering analysis
- Framework comparisons

Read These


"Noteworthy for its clarity, logic, candor and unequivocal stand on political implications." —THE NEW YORK TIMES BOOK REVIEW
Free to Choose
A PERSONAL STATEMENT
THE CLASSIC INQUIRY INTO THE RELATIONSHIP BETWEEN FREEDOM AND ECONOMICS
Milton Friedman
Winner of the Nobel Prize
and Rose Friedman


"Feynman at his idiosyncratic, brilliant best." —John Horgan, author of The End of Science
The Pleasure of Finding Things Out
NATIONAL BESTSELLER
THE BEST SHORT WORKS OF
RICHARD P. FEYNMAN
Foreword by Freeman Dyson


Named by Fortune
ONE OF THE SMARTEST BOOKS OF ALL TIME
FOOLED
BY
RANDOMNESS
The Hidden Role of Chance in Life and in the Markets
NASSIM NICHOLAS TALEB
SECOND EDITION, UPDATED BY THE AUTHOR

About Me

- Developer, Senior Developer, Team Lead, Architect, VP Engineering
- Military, education, non-profit, startups, products, consulting
- SpikeTV, MTV, Facebook, Microsoft, Apple, etc.
- ZCE Education Advisory Board
- PHP since 1999

Full Disclosure

- solarphp.com
- framework.zend.com
- auraphp.github.com
- “Rigor and integrity”
- github.com/pmjones

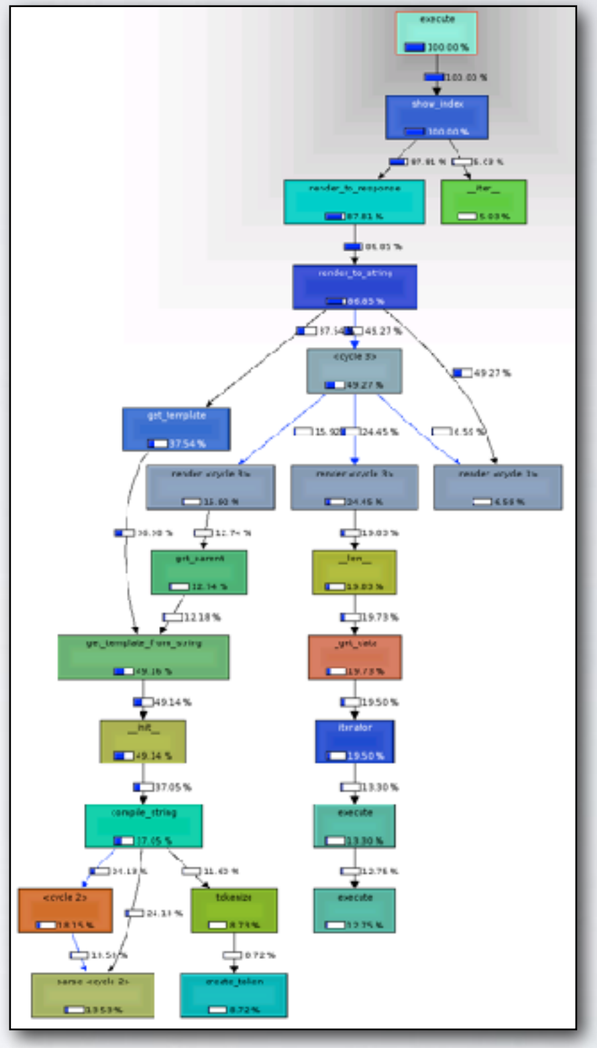
Part 1:
In General

Profiling vs Benchmarking

- Profiling applies to sub-systems (component design)

- Benchmarking applies to entire systems (architecture and interaction)

- Profiling is to unit testing, as benchmarking is to system testing



Benchmarking Subjects

- CPU
- RAM
- Disk access
- Database access
- Network access
- **Requests/second**

- Programmer productivity
- Time to initial implementation
- Time to add new major feature
- Time to fix bugs

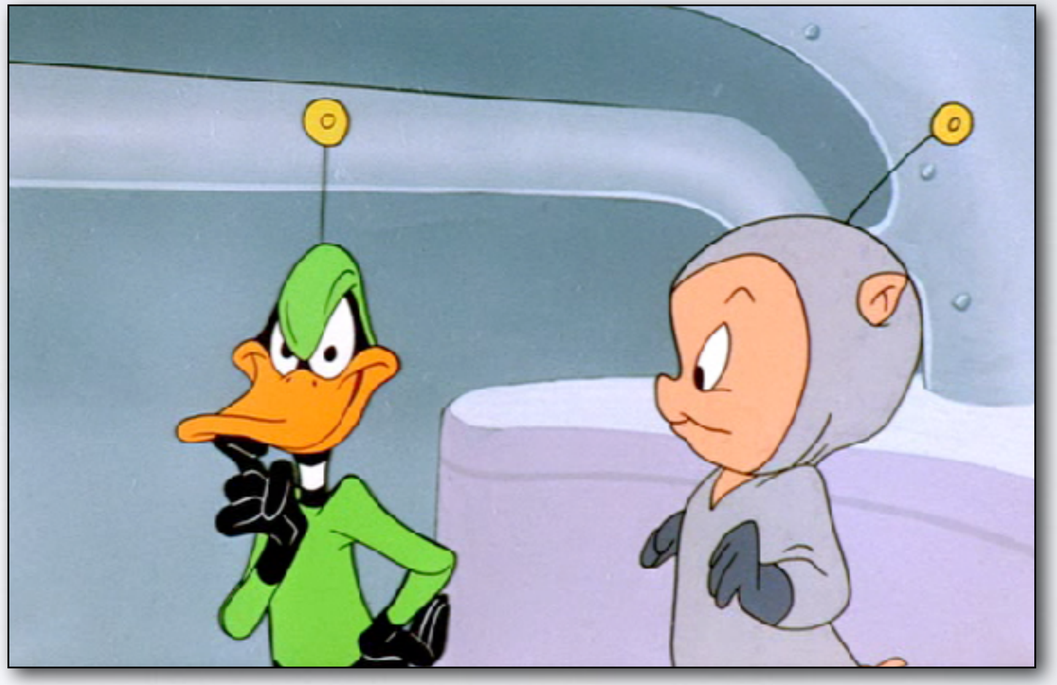
-- numeric measurement --
-- control for variables --

Why Bother With Benchmarking?

- New site will be submitted to Slashdot, Digg, etc.
- "Can we handle the traffic, Dodgers?"
- "Sure thing, boss."

Why Bother With Benchmarking?

- "...(time to load-test.)"
- How many req/sec to expect?
- Load-test, optimize, balance ...
- ... in a hurry, in the blind.



Luck and Limitations

- Do you feel lucky?
- What limits are imposed, and where?
- Where in the stack to expend effort?

Part 2:
Getting Started

Benchmarking Methodology

- Refining and reporting since 2007
- Benchmark the basic system
- Add a component, benchmark again
- Track responsiveness at each point

Benchmarking Tools

- ab, http_load, siege, ...
- Emulate concurrent users hitting a URL
- Reports req/sec, latency, min/max times, etc

```
ab -c 10 -t 60 http://example.com/foo/bar
```

Combined Tooling

- Each tool (ab, http_load, siege) has different options, params, and target specifications
  - github.com/pmjones/php-framework-benchmarks

```
./bench/(ab|http_load|siege) target/all.ini
```

Target Server Setup

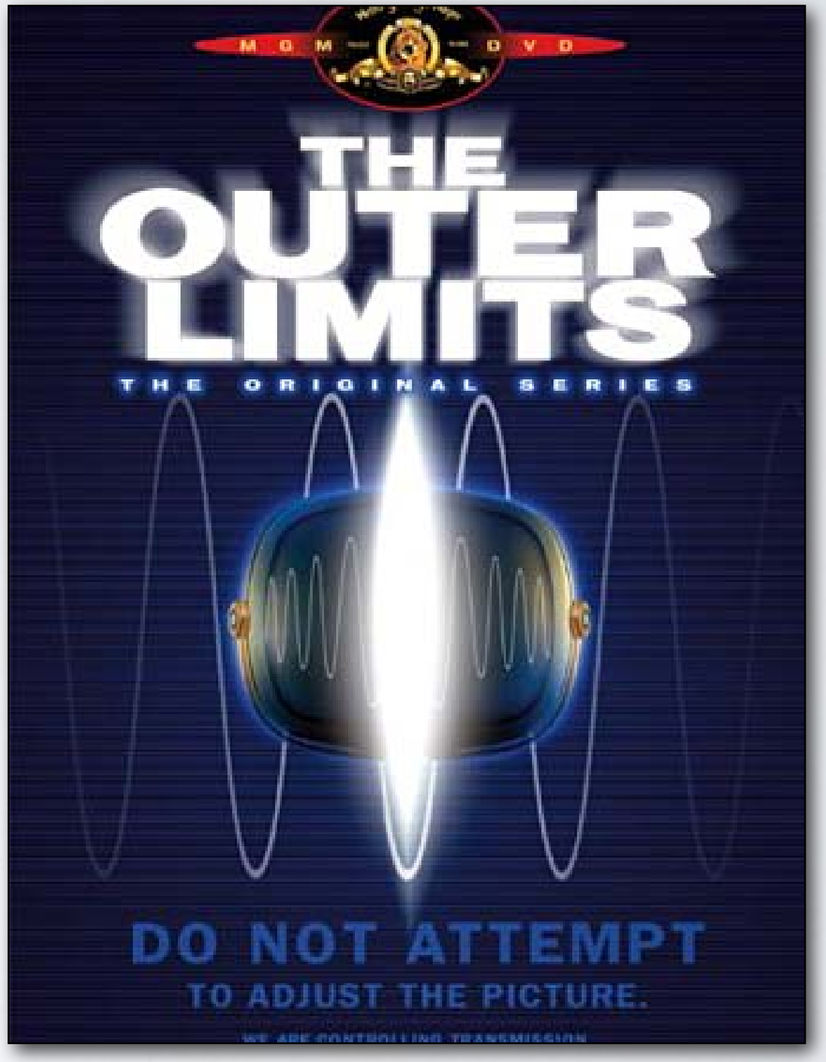
- Run against a test installation
- As similar to production as possible
- Hardware, OS, server, language version
- No framework or application code

Control for Known Variables

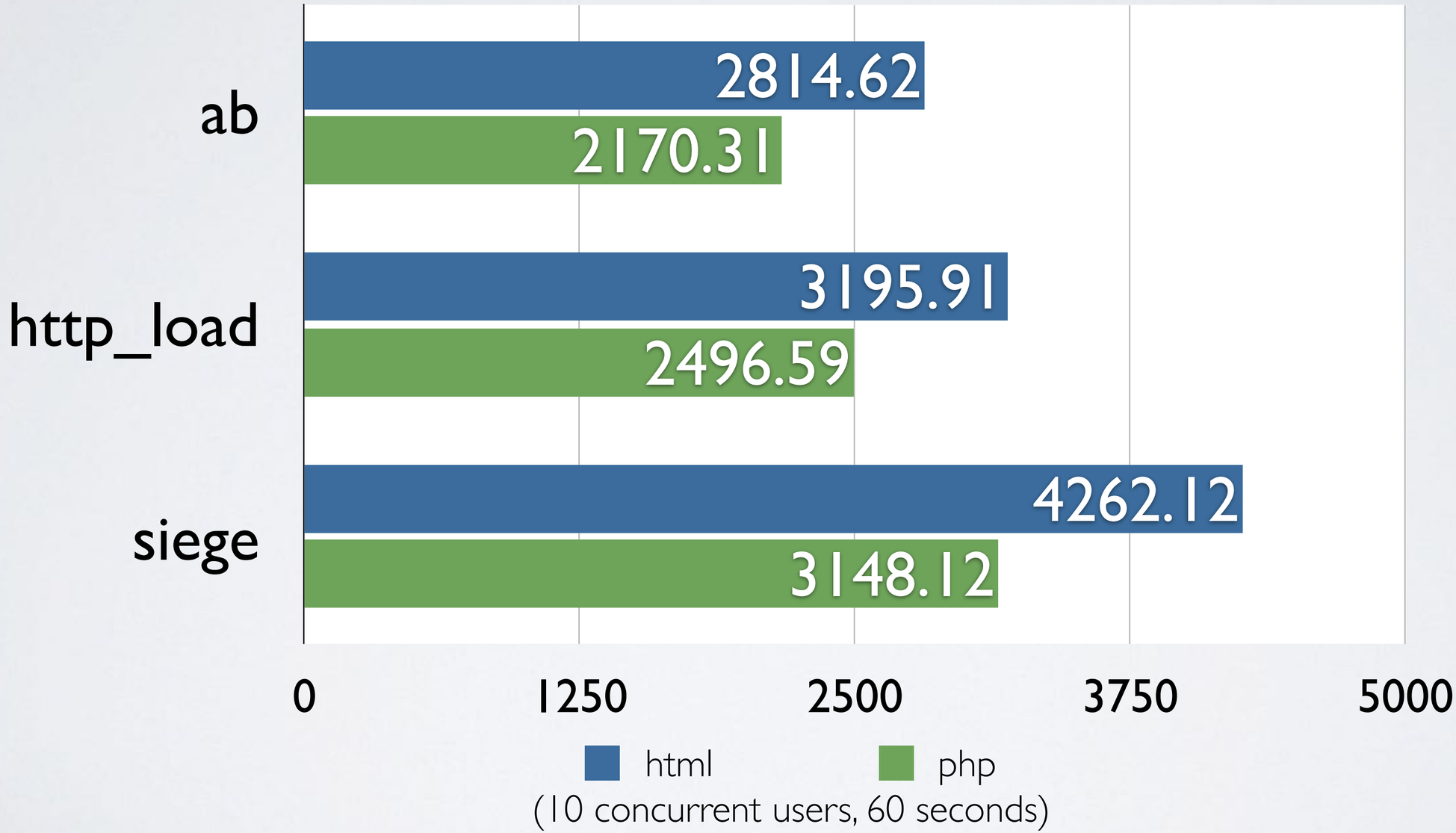
- Network latency
- Server processes

Outer Limits Of Responsiveness

- Amazon EC2 “Large” instance, 64-Bit Ubuntu 10.10 (Alestic)
- Apache 2.2 (stock install)
- PHP 5.3.3, APC, Suhosin (stock install)
- static index.html:
  `Hello World!`
- dynamic index.php:
  `<?php echo ‘Hello World!’; ?>`



Static and Dynamic Graph

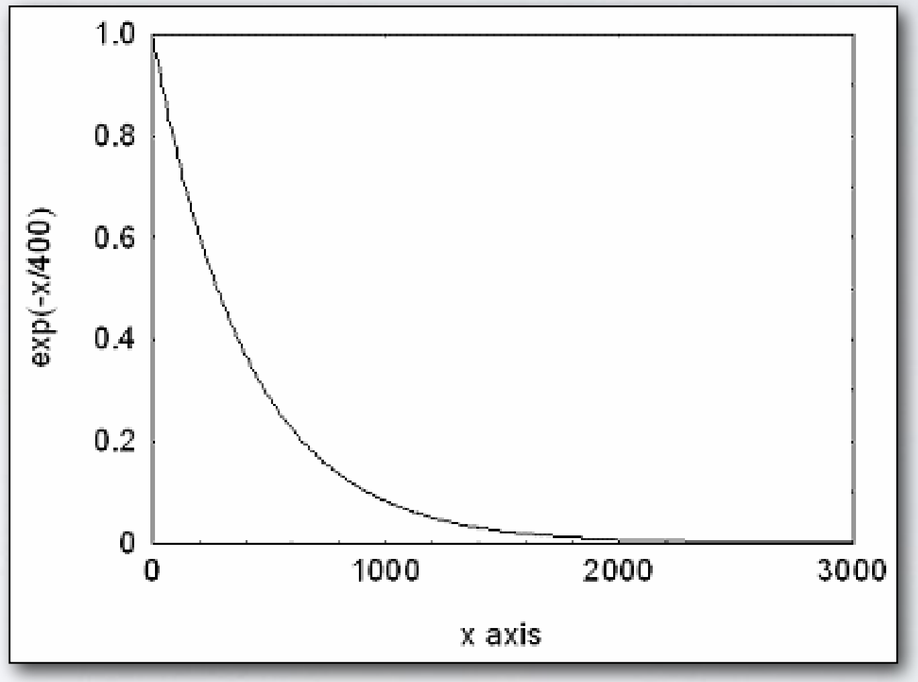


HTML and PHP Percentage Change

	html	php	pct
ab	2814.62	2170.31	77.11%
http_load	3195.91	2496.59	78.12%
siege	4262.12	3148.12	73.86%

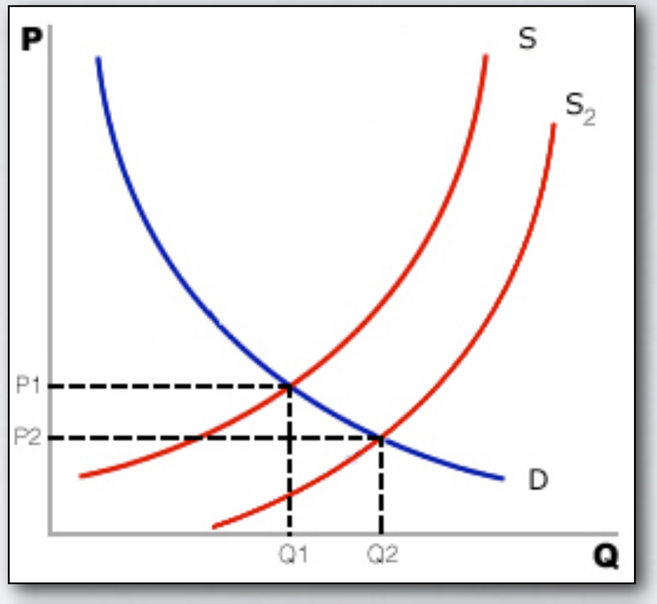
Control for Unknown Variables

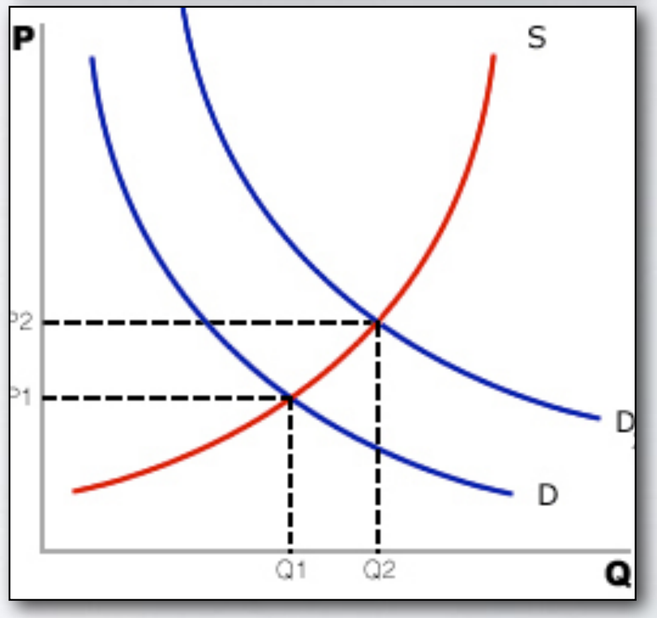
- Run multiple times
- No two runs will be identical; look for trends



Benchmarking as Resource Prediction

- About resource planning, not speed
- With a supply of "R" requests/second, you need "S" servers for demand of "T" traffic
- What stack components can you modify to increase "R" so you can handle more "T" with same or fewer "S"?



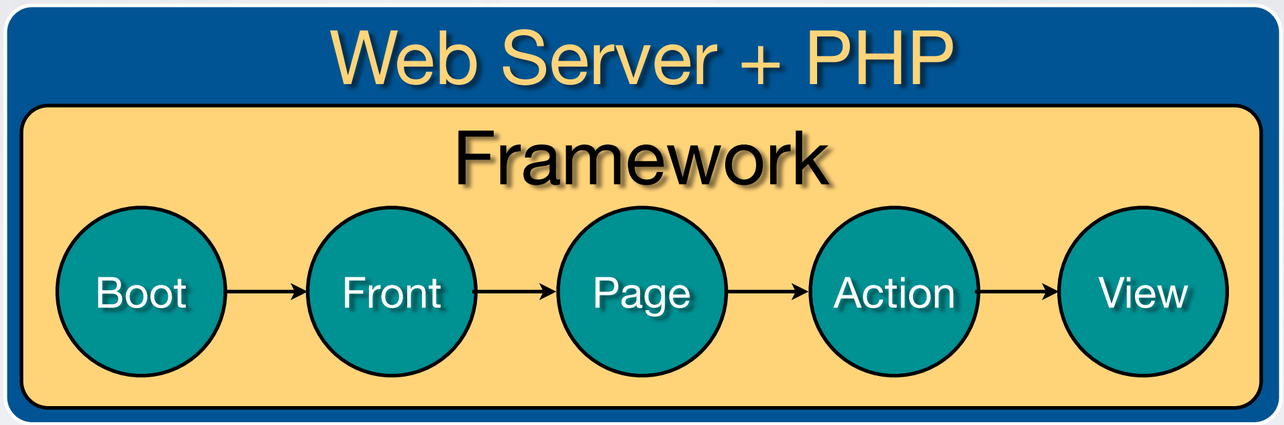


Part 3:
Single Framework

Single-Framework Methodology

- Minimal application
- Sane config
- No action code
- No view layout or helpers
- Static view text ("Hello World!")
- No page caching

Framework Benchmarking



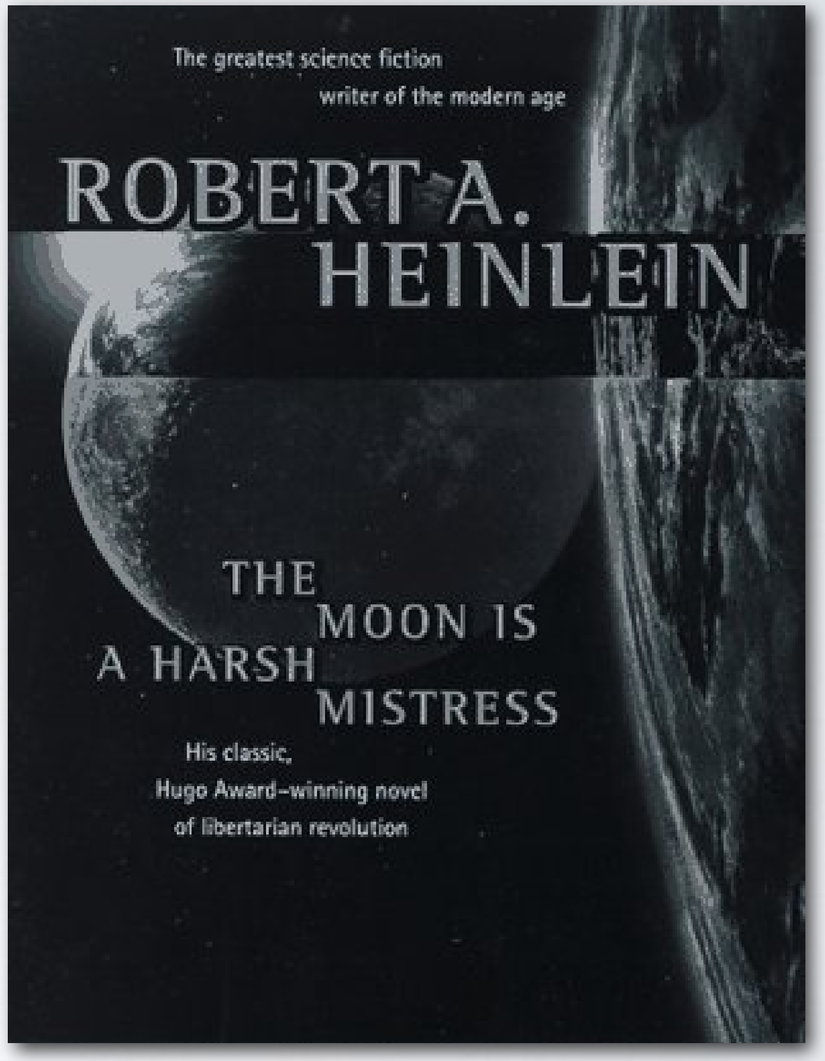
- Dynamic dispatch cycle as the limiting factor
- Improvements off critical path not likely to increase responsiveness
- “What about page caching?”

Single-Framework Analysis

- Benchmark static, dynamic, and framework (minimal application)
- Now you know limits on each
- Iterative benchmarking

Benchmarking as Engineering Aid

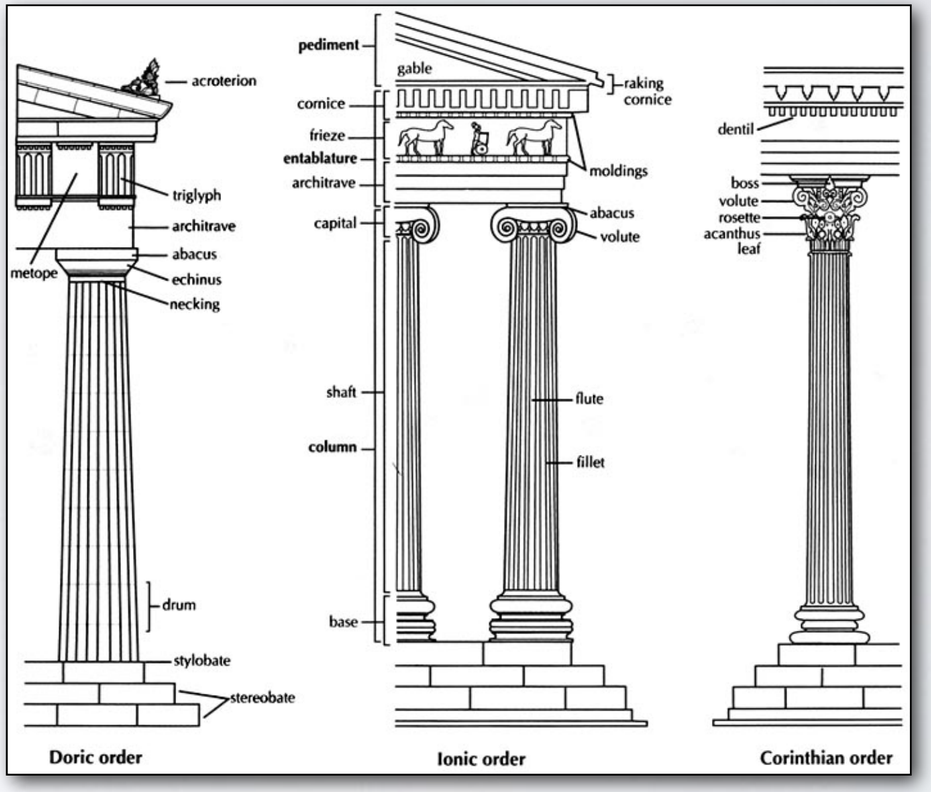
- Every architecture decision has a cost
- Compare stats between versions
- Are added features worth req/sec cost?
- (Thanks, Laura Thomson.)



Part 3:
Multiple Frameworks

Multiple-Framework Comparison

- Examine responsiveness in context of other systems
- Compare architecture costs between systems
- Use the same minimal app to remove variability in app code



Compare Like With Like

- Frameworks to be compared should be reasonably similar in approach

- PHP5, design patterns, front & page controllers, controller & view separation, plugins or extensions, no globals

- Aura, Cake, Lithium, Solar, Symfony, Zend

- CI, Kohana

Multiple-Framework Methodology

- Minimal application
- Reduced or optimized config
- Production mode
- No debugging or logging
- No database connection
- No layouts or view helpers

Multiple-Framework Methodology

- APC to reduce filesystem access
- Restart web server between runs (APC)
- 5 runs of 10 concurrent users for 60 seconds
- Average, then calculate req/sec as percent of PHP (thanks, Richard "Cyberlot" Thomas)

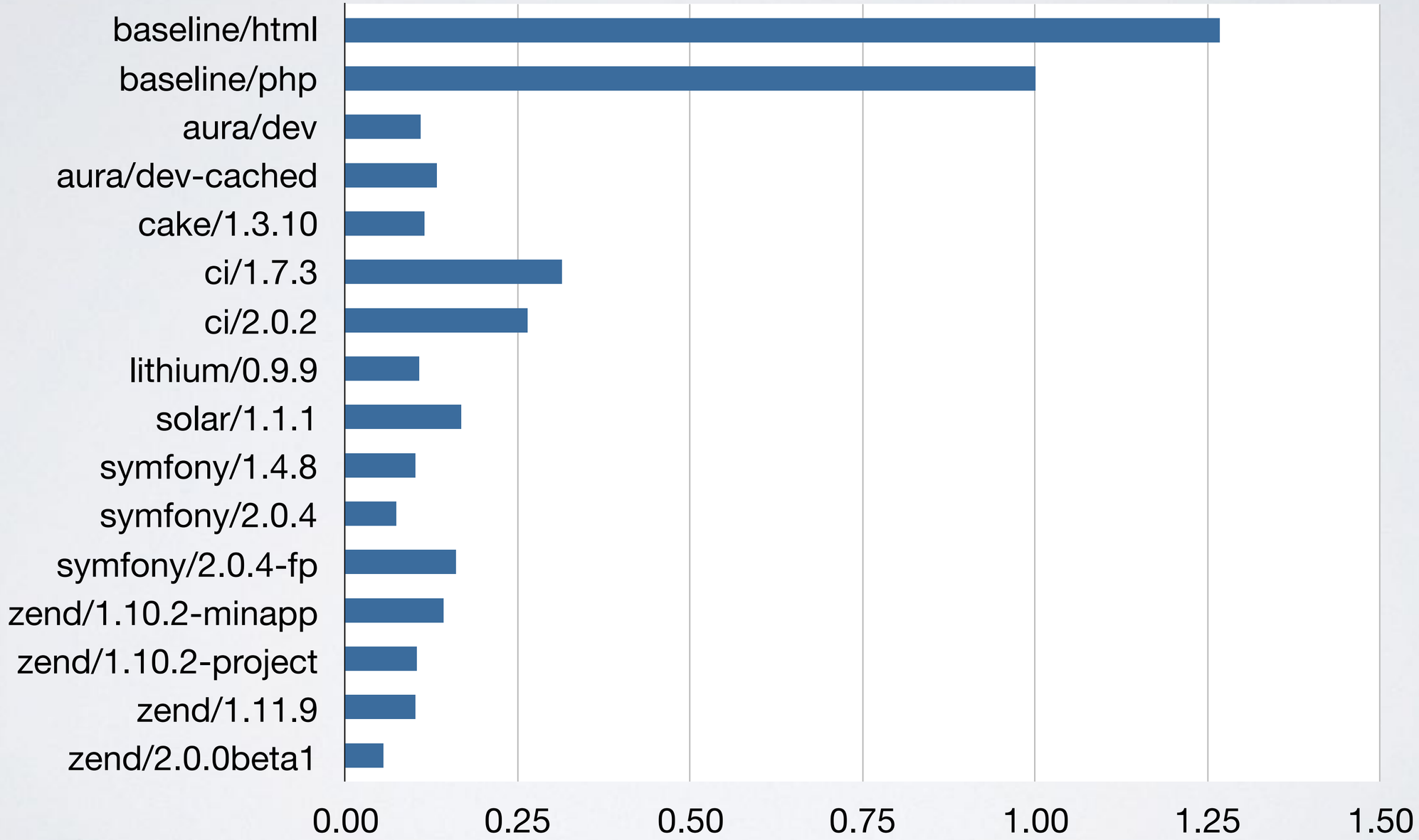
Framework Numbers

Target	*relative*	*average*	*1*	*2*	*3*	*4*	*5*
baseline/html	1.2669	2915.01	2935.71	2928.76	2864	2922.95	2923.63
baseline/php	1	2300.95	2316.01	2298.45	2306.55	2290.59	2293.13
aura/dev	0.1082	248.9	263.15	249.3	260.73	236.47	234.86
aura/dev-cached	0.1327	305.37	313.35	310.42	294.89	287.53	320.65
cake/1.3.10	0.1141	262.58	257.2	264.4	269.24	264.51	257.55
ci/1.7.3	0.313	720.16	717.63	729.75	706.64	723.17	723.61
ci/2.0.2	0.2637	606.78	629.07	589.96	636.2	575.59	603.06
kohana/3.1.3.1	0.2433	559.74	576.89	541.47	551.59	567.14	561.6
lithium/0.9.9	0.1077	247.75	264.83	211.66	272.27	201.67	288.32
solar/1.1.1	0.1682	387.13	403.42	382.57	362.32	398.34	388.98
symfony/1.4.8	0.1011	232.66	238.56	227.17	231.23	231.77	234.59
symfony/2.0.4	0.074	170.3	170.35	170.55	170.25	169.67	170.67
symfony/2.0.4-fp	0.1601	368.44	369.87	349.32	372.84	362.68	387.48
zend/1.10.2-minapp	0.1426	328.12	325.37	322.02	322.1	337.54	333.56
zend/1.10.2-project	0.1039	239.12	241.07	239.5	238.25	237.06	239.73
zend/1.11.9	0.1024	235.61	244.65	235.82	228.65	235.84	233.07
zend/2.0.0beta1	0.0554	127.55	127.28	127.75	127.23	127.62	127.88

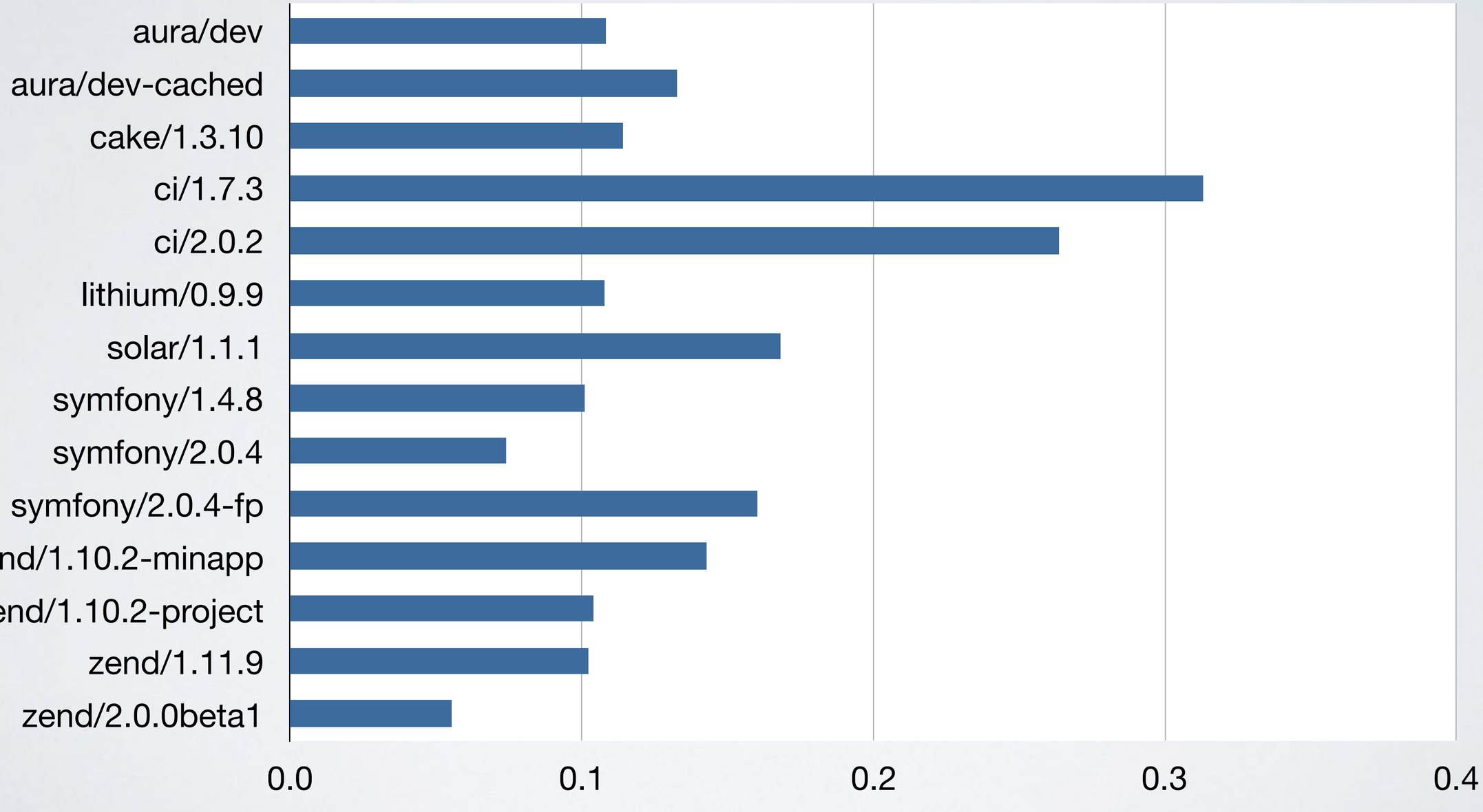
Framework Numbers

Target	*relative*	*average*	*1*	*2*	*3*	*4*	*5*
baseline/html	**1.2669**	**2915.01**	2935.71	2928.76	2864	2922.95	2923.63
baseline/php	**1**	**2300.95**	2316.01	2298.45	2306.55	2290.59	2293.13
aura/dev	0.1082	248.9	263.15	249.3	260.73	236.47	234.86
aura/dev-cached	0.1327	305.37	313.35	310.42	294.89	287.53	320.65
cake/1.3.10	0.1141	262.58	257.2	264.4	269.24	264.51	257.55
ci/1.7.3	0.313	720.16	717.63	729.75	706.64	723.17	723.61
ci/2.0.2	0.2637	606.78	629.07	589.96	636.2	575.59	603.06
kohana/3.1.3.1	0.2433	559.74	576.89	541.47	551.59	567.14	561.6
lithium/0.9.9	0.1077	247.75	264.83	211.66	272.27	201.67	288.32
solar/1.1.1	0.1682	387.13	403.42	382.57	362.32	398.34	388.98
symfony/1.4.8	0.1011	232.66	238.56	227.17	231.23	231.77	234.59
symfony/2.0.4	0.074	170.3	170.35	170.55	170.25	169.67	170.67
symfony/2.0.4-fp	0.1601	368.44	369.87	349.32	372.84	362.68	387.48
zend/1.10.2-minapp	**0.1426**	**328.12**	325.37	322.02	322.1	337.54	333.56
zend/1.10.2-project	**0.1039**	**239.12**	241.07	239.5	238.25	237.06	239.73
zend/1.11.9	**0.1024**	**235.61**	244.65	235.82	228.65	235.84	233.07
zend/2.0.0beta1	**0.0554**	**127.55**	127.28	127.75	127.23	127.62	127.88

Framework Graph (100%)



Framework Graph (40%)



What Does This Mean?

- No such thing as “fast” or “slow”
- “Proves” nothing
- Unlikely to change minds of devotees
- Resource usage, engineering aid
- Even the slowest is still pretty fast
- Each AJAX call is one request

Failure Modes

- Incorrect setup (aka “I got it wrong”)
  - Apache/Lighty, APC/XCache, mod_php/fcgid
  - Different concurrency or duration
  - Added features and app code will reduce dynamic response ... but curve is unknown.

Links

- github.com/pmjones/php-framework-benchmarks
- New Year's Benchmarks: paul-m-jones.com/archives/238
- Do All Frameworks Really Suck?: laurathomson.com/2007/07/do-all-frameworks-really-suck
- Google: "cargo cult science"

- Questions?
- Comments?
- Criticism?

Thanks!

- paul-m-jones.com
- slideshare.net/pmjones88
- twitter.com/pmjones
- joind.in/3741

Bonus Slides!

Objections

- "Highly profiled."
- "'Hello world' is unrealistic."
- "Not optimized for 'hello world.'"
- "Faster for 'real world' applications."

Avoid Frameworks?

- If framework code is only at 4-30% of max PHP, should we not use frameworks?
- "Slow" relative to what?
- Compare like with like, and proposed systems to existing systems.
- You will use a framework, even if you write it yourself.
- The good comes not from the code or the features, but process standardization.

Thanks Again!

- paul-m-jones.com
- twitter.com/pmjones
- http://joind.in/3741