

Web 2.0 Scalability

Issues with & tactics for allowing sites and services to survive on the Social Web

Web 2.0 Redux

- ▶ The most misunderstood buzzword...ever
 - Entrepreneurial
 - ▶ A roadmap for establishing a web-based company
 - ▶ A means of soliciting funding
 - ▶ Anything mashed-up
 - Marketing
 - ▶ An excuse to retrofit any commodity as a Long Tail candidate
 - ▶ The latest attempt to monetize web-syndicated data
 - ▶ A content delivery philosophy
 - ▶ A corny company name
 - Technical
 - ▶ Any web app using AJAX, JSON, RSS, screen scraping or web services
 - ▶ Absolutely contingent upon REST, SOA and/or SaaS
 - ▶ Anything built with Python and/or Ruby
 - ▶ A web experience with a social component
 - ▶ An extension of the LAMP stack
 - Design
 - ▶ New font set, fancy colors, non-traditional UIs
 - ▶ Thin clients with rich desktop-like functionality

On Scaling

- ▶ **“Scaling is the act of removing bottlenecks”**
 - *David Heinemeier Hanson, 37 Signals*
- ▶ **“Until you reach the point when you really need a four-person tent, stay in a two-person tent”**
 - *Matthew Ogle, Last.FM*
- ▶ **“PHP drives Web 2.0”**
 - *Andi Gutschman, creator of Zend*
- ▶ **“Go big or go home”**
 - *J. Allard, Microsoft*

The Problems

- ▶ Social features produce greater traffic
 - Peer communities foster very sticky services
- ▶ Network effects consume massive resources
 - Constant communication
 - Personalized content
 - Dynamic UIs
 - Rampant multimedia
 - Semantic data
 - Folksonomies

Rich Functionality via the WWW

► Desktop capabilities

- E-mail
- Calendaring
- Word processing
- Image manipulation
- Video editing
- Media encoding

The Solutions

- ▶ Throw more resources at the site
 - CPUs
 - RAM
 - Databases
 - Bandwidth
 - Web servers
 - Storage
- ▶ Rewrite the system
 - Design new backend architecture
 - Develop higher-end codebase
- ▶ Change partners
 - More robust service providers

Scaling Strategies

▶ Scale Up

- Add new, more powerful hardware

▶ Scale Out

- Add modular processing via CPU clustering

▶ Scale Cost

- Ensure revenue scales proportionately with growth

Web 2.0 Traffic Drivers

- ▶ Network effects
 - AJAX-driven functionality
 - A litany of clients across numerous platforms on a multitude of devices tapping your data sources
 - ▶ RSS/Atom feeds
 - ▶ Web services
 - ▶ Mashups
 - Microchunking
 - ▶ Widgets, gadgets, badges
 - RIAs providing offline access
 - ▶ Embedded databases, internal storage, internal web server, event-driven connectivity detection & synching

The Digg Effect

- ▶ The classic Web 2.0 traffic threat
 - Essentially a friendly DDoS attack
 - digg, Slashdot, Reddit, Fark
- ▶ Media embedding a la MySpace
- ▶ Gracefully handling downtime/outages due to drowning in requests

Scalability by Framework

▶ .NET & J2EE

- Inherent scalability
 - ▶ Enterprise platforms that do smaller jobs, too

▶ LAMP

- A world of opportunity (and headache) in PHP
 - ▶ More hands-on tuning than closed platforms
- Python viewed as the programming language of scale

▶ Ruby on Rails

- Does Rails really scale?
- Speed concerns with Ruby

▶ Others

- ColdFusion, Django, TurboGears, Zope, Pylons, Seaside

Hosting Issues

- ▶ Shared vs. dedicated
 - At what point do you need your own box?
- ▶ Establishing upgrade thresholds
 - Stress testing
 - Performance warning notifications
- ▶ Optimizing resources
 - Memory, distributed processing
 - Dedicated DB servers
- ▶ Can you afford to microchunk your data?
 - YouTube, FaceBook, Slide.com

Development Concerns

- ▶ Caching
 - Caching proper
 - Caching multimedia
 - Platform synchronization
 - ▶ Web, mobile, desktop, RSS, remote embedded media
 - Caching AJAX & web services
 - ▶ Use small payloads
 - ▶ Output as JSON
- ▶ Data
 - Managing CRUD
 - Master/slave DBs
 - Read-only tables
 - Write highly-performant database queries/SPROCs
- ▶ Coding
 - Employ best practices from the get-go
 - ▶ Write componentized code
 - ▶ Design tiered applications
 - MVC, 3-tier/n-tier
 - Optimize HTTP headers
 - Wisely use page encoding
 - Support Unicode for internationalization

Performance Tuning

- ▶ Config file optimization
 - web.config, PHP.ini, database.yml
- ▶ Web servers
 - Clustering, web gardens, web farms
 - Gzip, IIS 5.0 compression
 - Utilities
- ▶ Accelerators, mods, filters, load balancers
 - Memcached
 - ISAPI utils
- ▶ DB servers
 - Indexing strategies
 - MySQL storage engines
 - ▶ MyISAM vs. InnoDB
 - Separate servers
 - ▶ Content, multimedia, search

Database Optimization

- ▶ ORM vs. custom SQL
- ▶ Normalization
- ▶ 90% of performance problems rooted in indexing
- ▶ Difference between Access and SQL Server/Oracle
- ▶ Custom developed code vs. code generation
- ▶ Web services not the right solution in all situations

Case Study: Google

- ▶ Cheap hardware
- ▶ Custom Linux distro
- ▶ Custom software
- ▶ Custom database
- ▶ Communal computing power
- ▶ Distributed processing

Case Study: digg

- ▶ LAMP + Memcached
 - 100+ Linux boxes
 - PHP 4
 - MySQL
- ▶ 1,000,000 registered users
- ▶ 10,000,000 served pages/day
- ▶ 1,200,000 unique visitors/day

Case Study: Yahoo!'s 14 Rules

1. Make fewer HTTP requests
2. Use a CDN
3. Add an 'Expires' header
4. Gzip components
5. Put CSS at the top
6. Move JavaScript to the bottom
7. Avoid CSS expressions
8. Make JS and CSS external
9. Reduce DNS lookups
10. Minify JavaScript
11. Avoid redirects
12. Remove duplicate scripts
13. Turn off ETags
14. Make AJAX cacheable and small

Helpful Resources

▶ Whitepapers

- [LiveJournal's Backend – A history of scaling](#)
- [The Google Legacy – Inside the Googleplex](#)

▶ Books

- [Performance Tuning and Optimizing ASP.NET Applications](#)
- [Real World ASP.NET Best Practices](#)

▶ Blogosphere

- ["5 Question Interview with Twitter's Alex Payne"](#)
- ["Scalability, Sweet Scalability"](#)

▶ [Presentations](#)

- [The Technology Behind the Development of Digg](#)
- [Yahoo! - High Performance Webpages](#)

▶ Podcasts

- ["Scale & Scalability"](#)
 - ▶ Om Malik, Niall Kennedy & David Heinemeier Hanson
- [2006 Future of Web Apps Conference](#)
 - ▶ Kevin Rose, Josh Schachter, Cal Henderson