Cadillac or Nascar?
A Non-Religious Investigation of Modern Web Technologies

Akara Sucharitakul & Shanti Subramanyam
Sun Microsystems, Inc.
Agenda

• Introducing the web20kit
• Performance Results
• Tuning Suggestions
• Conclusions
Scalable Web Architecture

Clients — Internet — External Resources

Switches — Web Servers — Cache Nodes

Database Nodes — Storage Nodes
The Web20kit Overview

• Reference Architecture to evaluate common, modern web technologies

• Sample social networking application
  > 3 implementations - PHP, JavaEE and Ruby on Rails (RoR)

• Can be used to:
  > evaluate the differences in the various languages/frameworks for RoR, JEE and PHP
  > evaluate the infrastructure technologies for each implementation
  > compare the performance of the various technologies

• Plan to open source in the fall
Web20kit Architecture

- Driver
- Web/App Server (Apache/GlassFish/Mongrel)
- Database (MySQL)
- Object Store (NFS / MogileFS)
- Geocoder
- Cache (Memcached)

SUT
The Application

- A Social Event Calendar
  - Allows posting, sharing, tagging/searching, and commenting on social events
  - Events and persons have images, thumbnails
  - Events have event literature (pdf file)
  - Events can be browsed by date (Ajax)
  - One can sign up to attend events (Ajax)
  - Provides details about the individual events
  - Provides event feeds
Application Environments

- Client-side
  > AJAX/JSON

- Three implementations
  > PHP
    > UnixODBC / PDO
    > PECL Memcache Client
    > LocalFS / NFS / Distributed FS
  > Ruby on Rails
    > RoR Framework
    > Memcache (future)
    > LocalFS / NFS
    > Distributed FS (future)

> Java EE
  > Servlets, JSPs
  > JPA
  > Whalin MemCache client
  > LocalFS / NFS / Distributed FS
Implementations

- Application Deployments
  - PHP: apache/lighttpd, php, memcached, MySQL (AMMP)
  - Java EE: GlassFish, memcached, MySQL (JAMM)
  - RoR: Mongrel, memcached, MySQL

- Workload
  - Implemented using Faban
    - Open source benchmark development toolkit: http://faban.sunsource.net
  - Includes Load Generator, DB loader (in Java)
The Framework Spectrum

Language

PHP

JavaEE

Framework

RoR

O-R Mapping

MVC

O-R Mapping
Cost/Benefits of O-R Mapping

• Cost
  > No direct control over DB access
  > Not optimized for specific databases / SQL dialects
  > Inadequate tuning when needed
  > Still layer/overhead on top of SQL

• Benefits
  > OO Programming Model
  > Database agnostic
  > Easy to maintain
  > Enables automated data caching
  > Policy-based data access
  > Enables optimizations
Agenda

- Introducing the web20kit
- **Performance Results**
- Tuning Suggestions
- Conclusions
web20kit/PHP Scaling

Throughput (Ops/sec) vs. Users

Utilization vs. Users

Network and CPU Utilization
Web20kit/Java EE Scaling

- Can scale with single process
- Low memory footprint
- Java Persistence API (JPA)
  - O/R Mapping – eases development
  - Cache can be effective in reducing DB load
Web20kit/RoR

• Preliminary Performance Testing underway
  > Full functionality, no caching
  > 100 user load on 16 thins – db bottleneck

• Some findings:
  > Thin is more efficient than mongrel
  > JRuby is 3-4x better than Ruby 1.8.6
  > On Solaris, ruby in Cool Stack 1.3 gives 40% improvement
Memcached: Thread Scaling
Memcached: Scaling with Memory Size
Memcached Client

• No standard client libraries
• Performance issues are more on the client side
• Java Clients
  > Whalin issues: huge cpu (single-byte read), huge syscalls, overhead of socket-pooling
  > Spy issues: single-threaded
• PHP Client
  > PECL client seems to be the most standard
  > Many folks roll their own
MySQL Performance

![CPU Utilization Graph]

- **MySQL 5.0**
- **MySQL 5.1**

**Axes:**
- **Users** (X-axis, values: 1400, 1600, 1800)
- **CPU Utilization** (Y-axis, values: 0, 2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20, 22.5, 25)
Agenda

- Introducing the web20kit
- Performance Results
- Tuning Suggestions
- Conclusions
Apache/PHP Tuning

- **Network Stack**
  - Tune TCP time-wait if handling lots of connections

- **Apache**
  - Do not load modules that you do not need (in httpd.conf)
  - Tune ListenBacklog (8192), ServerLimit (2048), MaxClients (2048)

- **PHP**
  - Turn off `safe_mode` if you don't need it
    - `safe_mode = off`
  - Increase `realpath_cache_size` if you have lots of files
    - `realpath_cache_size = 128K`
GlassFish Tuning

- Heap Tuning: -Xms, -Xmx (upto 3GB for 32bit JVM)
- GC Tuning: -XX:+UseParallelGC
- HTTP Thread Tuning: Increase thread-count to 128
- JPA Provider: Use EclipseLink
- Run web container in production mode
  > App has to be re-deployed if changed
Memcached Tuning

• Network
  > Ensure network processing is distributed across CPUs
  > Bind memcached to CPUS not processing interrupts

• Memcached
  > Run memcached 1.2.5 with 4 threads (default)
  > Use in 64-bit mode for a large cache size
MySQL Tuning

• Tune your queries
  > Use joins over sub-queries
  > Use limits

• Innodb
  > Avoid too frequent FS flushes
    > innodb_flush_log_at_trx_commit = 2

• Separate read/write databases
  > Avoids trashing query cache

• http://blogs.sun.com/allanp/entry/tuning_mysql_on_linux
• http://blogs.sun.com/realneel/entry/tuning_mysql_innodb_for_sysbench
Conclusions

• Software performance/scalability
  > Good web server scalability for both multi-process and multi-thread models
  > Memcached – clear scalability limit on single system, needs horizontal scaling
  > MySQL – huge performance improvement in 5.1

• Software design and maintenance
  > Frameworks allow rapid development and ease of maintenance, but lack in tunability
  > Good planning and coding conventions can bring maintainability close to frameworks
Conclusions

- Networking
  - 1Gbe bottleneck for web apps on modern systems
  - 10Gbe immature
  - Link aggregation solves some short term problems
  - Large interrupt load, needs spreading across CPUs

- Web20Kit aims...
  - Proof point for scalable web software design
  - Reference architecture
  - Performance and scalability test
Resources

- Faban – open source benchmark development kit
  > http://faban.sunsorce.net
- Cool Stack – open source apps optimized for Solaris
  > http://cooltools.sunsorce.net/coolstack
- JRuby – faster, more scalable Ruby VM
  > http://jruby.codehaus.org
- GlassFish – open source Java EE Application Server
  > https://glassfish.dev.java.net