BenchLab
An Open Testbed for Realistic Benchmarking of Web Applications

http://lass.cs.umass.edu/projects/benchlab/

Emmanuel Cecchet, Veena Udayabhanu, Timothy Wood, Prashant Shenoy

University of Massachusetts Amherst
WEBAPPS YESTERDAY AND TODAY
TPC-W BENCHMARK

- Online bookstore emulating... amazon.com!
- Reference benchmark still in use today in the Systems community
TYPICAL E-COMMERCE BENCHMARK AVAILABLE TO THE RESEARCH COMMUNITY

- Setup for performance benchmarking
  - Browser emulator
  - Static load distribution

- Missing features
  - Content: Javascript, AJAX, cookies, ...
  - Network: HTTP 1.1 request pipelining, caching, SSL
  - Latencies: WAN, rendering time...
BENCHMARK DESIGN

Traditional approach (TPC-W, RUBiS...)

Workload definition + Web Emulator → Application under Test

BenchLab approach

HTTP trace + Real Web Browsers → Application under Test
OUTLINE

- What has changed in WebApps
- Benchmarking real applications with BenchLab
- Experimental results
- Demo
Web applications have changed

- Web 2.0 applications
  - Rich client interactions (AJAX, JS…)
  - Multimedia content
  - Replication, caching...
  - Large databases (few GB to multiple TB)
- Complex Web interactions
  - HTTP 1.1, CSS, images, flash, HTML 5...
  - WAN latencies, caching, Content Delivery Networks…
EVOLUTION OF WEB APPLICATIONS

<table>
<thead>
<tr>
<th>Applications</th>
<th>HTML</th>
<th>CSS</th>
<th>JS</th>
<th>Multimedia</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUBiS</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>eBay.com</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>31</td>
<td>38</td>
</tr>
<tr>
<td>TPC-W</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>amazon.com</td>
<td>6</td>
<td>13</td>
<td>33</td>
<td>91</td>
<td>141</td>
</tr>
<tr>
<td>CloudStone</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>21</td>
<td>28</td>
</tr>
<tr>
<td>facebook.com</td>
<td>6</td>
<td>13</td>
<td>22</td>
<td>135</td>
<td>176</td>
</tr>
<tr>
<td>wikibooks.org</td>
<td>1</td>
<td>19</td>
<td>23</td>
<td>35</td>
<td>78</td>
</tr>
<tr>
<td>wikipedia.org</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>20</td>
<td>36</td>
</tr>
</tbody>
</table>

Number of interactions to fetch the home page of various web sites and benchmarks
HTTP vs Browser Replay

- GET home page and receive HTML (0.25s)
- Parse HTML (0.06s)
- GET CSS and Javascript on 6 connections in parallel (Firefox)
- Starts rendering and execute Javascript
- GET additional Javascript imports
- Complete Javascript and continue rendering
- Download images
- More rendering
- Download final images

- Browsers are smart
  - Caching, prefetching, parallelism...
  - Javascript can trigger additional requests

- HTTP replay cannot approximate real Web browser access to resources
Typing Speed Matters

- Auto-completion in search fields is common
- Each keystroke can generate a query
- Text searches use a lot of resources

```
GET /api.php?action=opensearch&search=W
GET /api.php?action=opensearch&search=Web
GET /api.php?action=opensearch&search=Web+
GET /api.php?action=opensearch&search=Web+2
GET /api.php?action=opensearch&search=Web+2.0
```
STATE SIZE MATTERS

- Does the entire DB of Amazon or eBay fit in the memory of a cell phone?
  - TPC-W DB size: 684MB
  - RUBiS DB size: 1022MB

- Impact of CloudStone database size on performance

<table>
<thead>
<tr>
<th>Dataset size</th>
<th>State size (in GB)</th>
<th>Database rows</th>
<th>Avg cpu load with 25 users</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 users</td>
<td>3.2</td>
<td>173745</td>
<td>8%</td>
</tr>
<tr>
<td>100 users</td>
<td>12</td>
<td>655344</td>
<td>10%</td>
</tr>
<tr>
<td>200 users</td>
<td>22</td>
<td>1151590</td>
<td>16%</td>
</tr>
<tr>
<td>400 users</td>
<td>38</td>
<td>1703262</td>
<td>41%</td>
</tr>
<tr>
<td>500 users</td>
<td>44</td>
<td>1891242</td>
<td>45%</td>
</tr>
</tbody>
</table>

CloudStone Web application server load observed for various dataset sizes using a workload trace of 25 users replayed with Apache HttpClient 3.
OUTLINE

- What has changed in WebApps
- Benchmarking real applications with BenchLab
- Experimental results
- Demo
BenchLab

- Capture application workloads
- Replay captured traces in real Web browsers
- Store detailed performance results

Benchmark repository
- Store virtual machines of applications under test
- Store test traces, configurations and results
- Repeat experiments
- Compare results
RECORDING HTTP TRACES

- 3 options to record traces in HTTP Archive (HAR) format
  - directly in Web browser
  - at HA proxy load balancer level
  - using Apache httpd logs


**BenchLab WebApp**

- JEE WebApp with embedded database
- Repository of benchmarks and traces
- Schedule and control experiment execution
- Results repository
- *Can be used to distribute / reproduce experiments and compare results*

![Diagram of BenchLab WebApp](image)

- Upload traces / VMs
- Define and run experiments
- Compare results
- Distribute benchmarks, traces, configs and results

**Web Frontend**

- Results upload
- Trace download
- Experiment start/stop
- Browser registration

**Experiment scheduler**

- Traces (HAR or access_log)
- Results (HAR or latency)
- Experiment Config
- Benchmark VMs
**BenchLab Client Runtime (BCR)**

- Replay traces in real Web browsers
- Small Java runtime based on Selenium/WebDriver
- Collect detailed response times in HAR format
- Can record HTML and page snapshots
- Upload results to BenchLab WebApp when done
Wikimedia Foundation Wikis

- Wikimedia Wiki open source software stack
  - Lots of extensions
  - Very complex to setup/install
- Real database dumps (up to 6TB)
  - 3 months to create a dump
  - 3 years to restore with default tools
- Multimedia content
  - Images, audio, video
  - Generators (dynamic or static) to avoid copyright issues
- Real Web traces from Wikimedia
- Packaged as Virtual Appliances
RUNNING AN EXPERIMENT WITH BENCHLAB

- Use your own deployment framework and monitoring
- Real browser replay
  - Full rendering with AJAX, JS, multimedia...
  - Emulate human typing speed while filling form, click buttons...
- Analyze, compare & share results

1. Upload traces
2. Define experiment
3. Start experiment

• View results
• Repeat experiment
• Export setup/traces/VMs/results
OUTLINE

- What has changed in WebApps
- Benchmarking real applications with BenchLab
- Experimental results
- Demo
EMULATED VS REAL BROWSER

- CPU and IO usage varies greatly on server for same workload
- CloudStone with 25 users workload from Amazon EC2 East Coast to Umass Amherst
  - Avg CPU emulated: 63.2%
  - Avg CPU real browser: 77.7%
- Resource access pattern affects file caching on server
**JavaScript Effects on Workload**

- Server side resource usage varies greatly
- Additional queries during form processing

---

**Good Input**

- Emulated Browser
- Real Browser

**Bad Input**

- Emulated Browser
- Real Browser
LAN VS WAN LOAD INJECTION

- Deployed BCR instances in Amazon EC2 data centers
  - As little as $0.59/hour for 25 instances for Linux
  - Windows from $0.84 to $3/hour
- CPU usage varies greatly on server for same workload
  (LAN 38.3% vs WAN 54.4%)
CONCLUSION

- Web Applications have changed
- Real browsers needed for modern WebApp benchmarking
- BenchLab provides
  - Infrastructure for Internet scale Benchmarking of real applications
  - Virtual Appliances of real applications
  - Repository of traces, benchmarks and results

- A lot to explore...
Q&A

COME AND SEE OUR POSTER AND DEMO

http://lass.cs.umass.edu/projects/benchlab/

DILBERT™ by Scott Adams

I HAVE A THERE ARE NO STUPID QUESTION... STUPID QUESTIONS.

THAT'S RIDICULOUS... IF THERE ARE NO STUPID QUESTIONS THEN WHAT KIND OF QUESTIONS DO STUPID PEOPLE ASK? DO THEY GET SMART JUST IN TIME TO ASK QUESTIONS?

WERE YOU GOING TO ASK ME SOMETHING? SEE... NOW THERE'S A STUPID QUESTION.