Client-side Page Element Web-Caching

May 7, 2009

Ramya Karri

Committee Members

Chris Pollett

Soon Tee Teoh

Tsau Young Lin

Agenda

- Goal and Motivation
- Design
- Tools Used
- Implementation
- Results
- Observations
- Conclusion

Final Goal of the Project

Create a new form of web page caching at the client-side that reduces the response time of the web pages

Motivation: Why Web Caching?

- We need to be familiar with two main parameters in order to understand our project
  1. Response time of a web page – the time taken by a web page to load.
  2. Web Caching – making a copy of a web documents so as to avoid recalculation of same page on every request.
     - Other Main Purposes are :
       - To reduce bandwidth usage - amount of data that will be transmitted to display the web page
       - To reduce the load of the web server by reducing the number of web page requests
       - To enhance the web browsing by reducing the waiting time of the user

Current Statistics

80% of end-user response time is spent on the front-end while downloading all the components in the page such as images, style sheets, scripts, Flash, etc ("Best Practices for Speeding Up Your Web Site," 2008)

Caching Techniques

- Currently caching of web pages and parts of web pages are done on the **server-side** in order to reduce the computation of the same page on every request and to serve the web pages faster to the client
- Caching is also done at **Proxy server**
  - When a request is made to the proxy server for the first time it redirects the request to the web server
  - On receiving the response, the proxy server stores a copy in its own cache, and reuses it in further requests

Caching Techniques (Contd..)

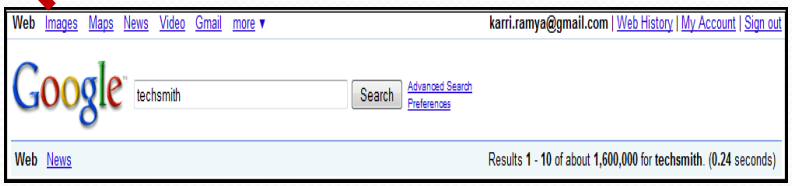
- Caching is also done at the **client-side** by the web browser
- It stores the entire web page, Images, external Javascripts, CSS in web browsers cache
- Reuses them in further web requests
- This is used in order pages faster to the user

Sharing common portions

- Most of the websites available today share many common portions across their web pages
- Common Portions include
  - Navigation bars
  - Advertisements
  - Search bars
  - Top bar
  - Bottom bar
  - etc

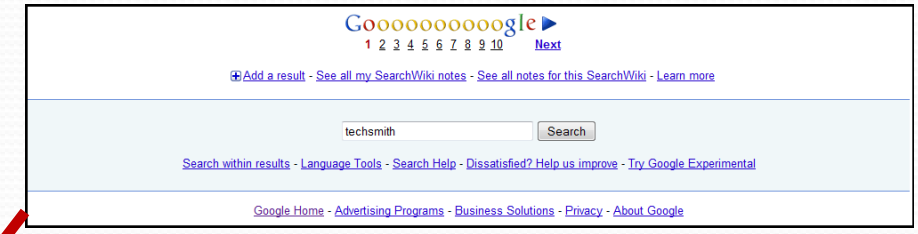
Google Search Engine

- Top Bar



Google Search Engine

- Bottom Bar



Google Search Engine

- Advertisements – Right Bar

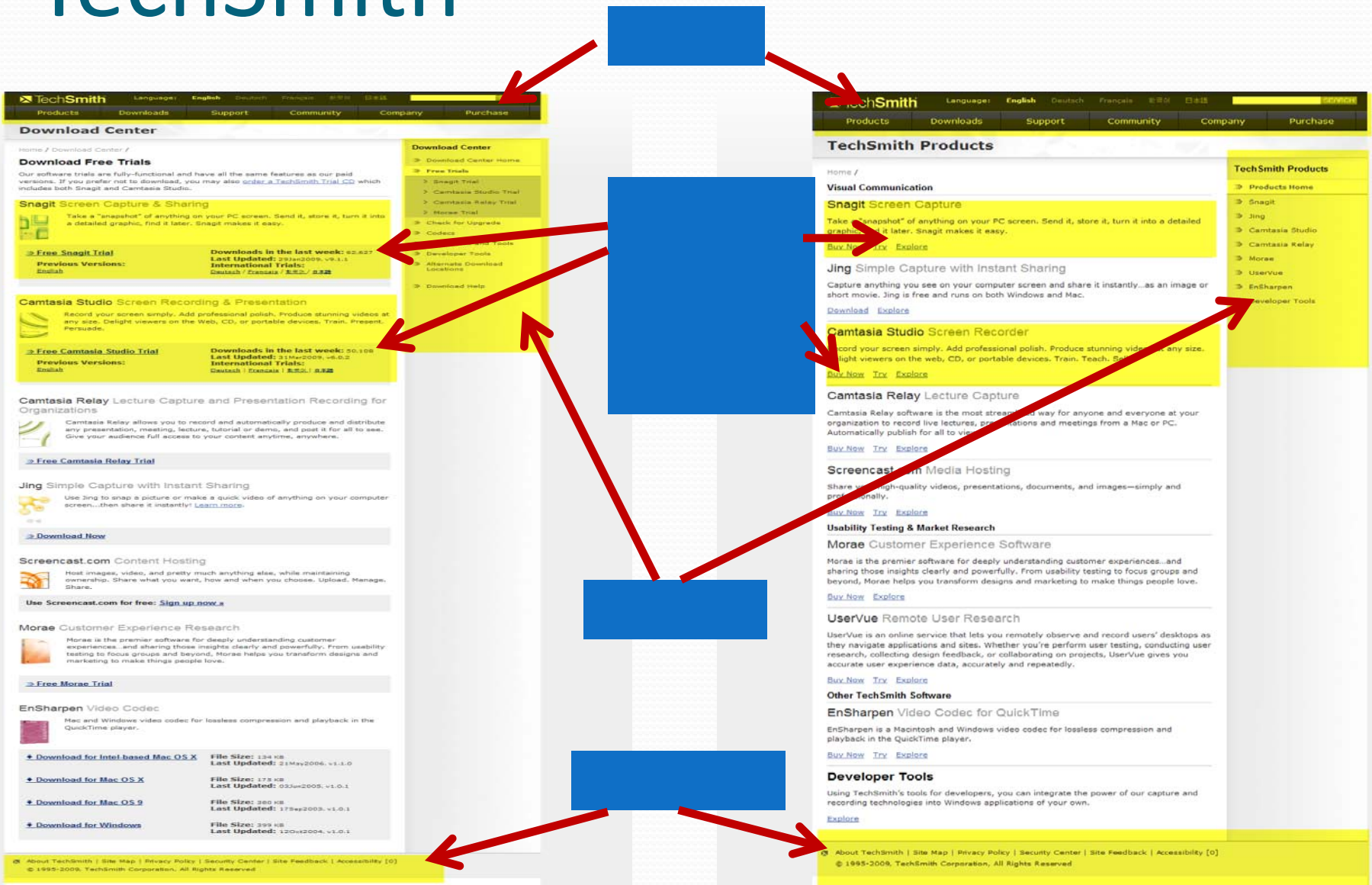
Sponsored Links

TechSmith Snagit 9.0
Powerful Screen Capture Software
Download Securely & Safely Today!
www.SoftwareCasa.com/Snagit

Google Search Engine

- Advertisements – Below the Top Bar

TechSmith


TechSmith
Products
Downloads
Support
Community
Company
Purchase
Download Center
Download Free Trials
Snagit Screen Capture & Sharing
Camtasia Studio Screen Recording & Presentation
Camtasia Relay Lecture Capture and Presentation Recording for Organizations
Jing Simple Capture with Instant Sharing
Screencast.com Content Hosting
Morae Customer Experience Research
EnSharpen Video Codec
TechSmith Products
Visual Communication
Snagit Screen Capture
Jing Simple Capture with Instant Sharing
Camtasia Studio Screen Recorder
Camtasia Relay Lecture Capture
Usability Testing & Market Research
Morae Customer Experience Software
UserVue Remote User Research
Other TechSmith Software
EnSharpen Video Codec for QuickTime
Developer Tools

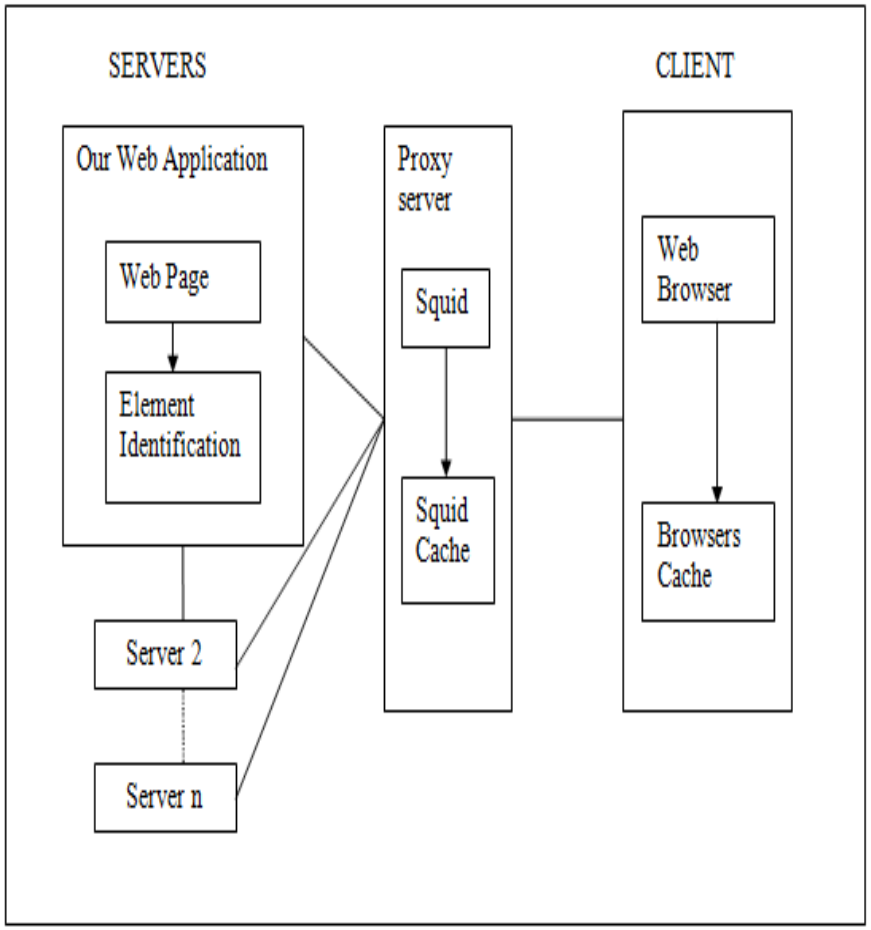
Aim

- Reduce the response time of the web pages, by caching the most common portions of the websites at the client-side, at-least by 20%
- What do we need?
  - New caching mechanism at the client-side that does the following
    - Identify the most common portions of the website
    - Caches these common portions at the client
    - Reuses them in further page requests

Design

- We have used CakePHP to develop our application, the design process followed is the Model View Controller (MVC)
- The Model acts as an access point to the database. It represents the tables in the database needed to develop the website
- The Controller is used to manage the logic of the application
- The view is usually the html portion of the website needed to display the web pages

Design (Contd..)



- We have three main components in our project
  - Web Server – used to serve our web pages
  - Web Client – Requests web pages using a web browser
  - Caching Proxy Server – Serves the web page requests on behalf of the web server

Tools Used

- Apache HTTP Server, a freely available web server is useful for producing static and dynamic web pages
- We have tested our caching mechanism on four different web browsers – Firefox, Internet Explorer, Opera, Safari
- Squid Proxy Server is used the caching proxy server
- In order to measure the response time of the web page
  - For Firefox browser, we used a Firefox add-on called **YSlow**
  - For Internet Explorer, we used **iMacros** – a browser based macro reader
  - For Opera and Safari, we have used **Curl** – Command Line tool for transferring files
- PHP, a scripting language used to develop the web pages
  - We have used **CakePHP**, a platform used to develop PHP applications
    - Consists of HTML, Javascript, and database helpers used to develop our web applications easily

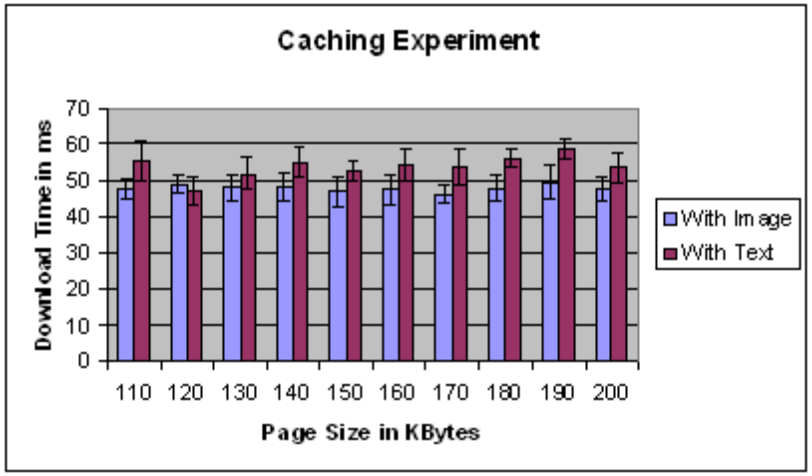
Initial Research

- Developed a Course Listing Website
  - Motivation: Get Real Hands-on Experience using CakePHP

- Balance Load Between Servers – Using Squids
  - Motivation: Setup a proxy server configuration

Initial Research (Contd..)

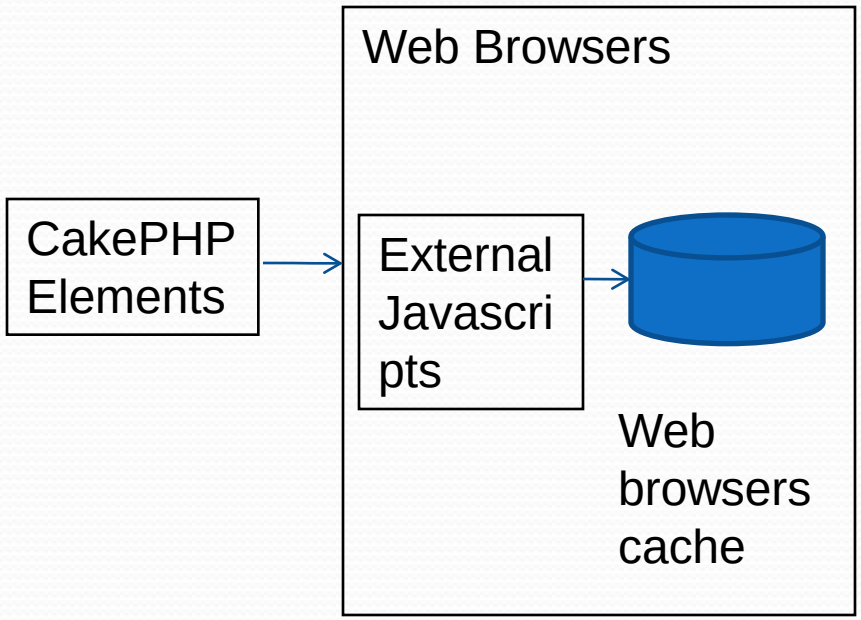
- Advantage of Caching Text
  - Motivation: Observe the behavior of various web browsers
  - Created several web pages with images and only text; compared the total download times of pages having images with pages having only text.



Implementation

- In CakePHP, common portions that needed to repeated across several other pages or at different portion of the same page are stored as elements
- These are added to the view using **Element** function
- Most of the latest web browsers **cache external Javascripts**
- To develop our caching mechanism, we have used CakePHP Element function + Web Browser Caching mechanism

Implementation(Contd..)



- Identification of the most common portions of the website is done by the web developer
- Store these common portions separately as CakePHP elements
- Procedure that converts these elements and serves them to the browser as external Javascripts
  - The elements are replaced by empty div tags
  - External Javascripts are created for the elements which replace the empty div tags with the actual contents when the page is displayed to the user

Implementation (Contd..)

- As these external Javascripts get cached in the web browsers cache
- In further requests, these elements will be used from the browser cache
- Retransmission of these common portions does not take place

Challenges

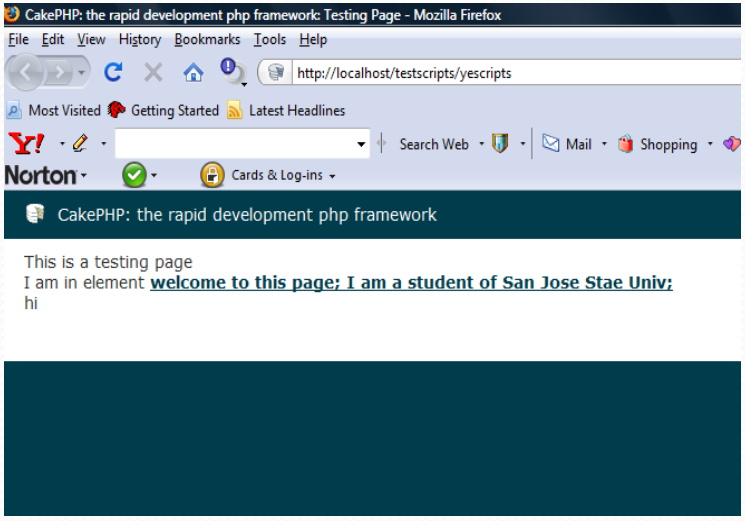
- How to work when the web browsers Javascript is turned off
  - We created an extension to CakePHP Element function
    - When Javascript is off, the actual html content is served
    - When Javascript is on, the element is rendered using our caching mechanism i.e., an empty div tag is created and an external Javascript is used to replace the content of the div tags with the actual elements at runtime
- To make sure the html portion of the elements is properly rendered as external Javascripts

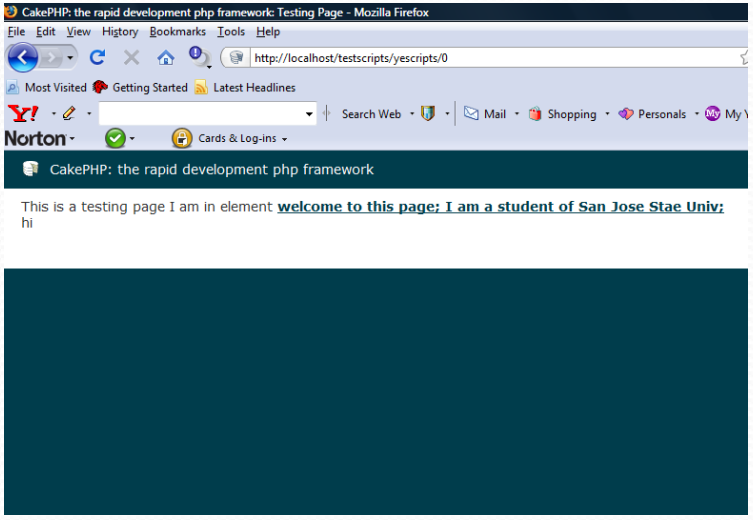
Challenges (Contd..)

- We made sure that Javascript does not break on multiple lines and with special characters
- Updating the elements by giving **frequency** to the elements i.e., when to refresh the Javascripts and where to find them in CakePHP
- Apache redirection to make CakePHP search our Javascripts in its cache instead of regular Javascript folder

Testing

- Phase 1: Make sure our caching system works
- Created a web page with single element
- Using both our new caching mechanism and the conventional way





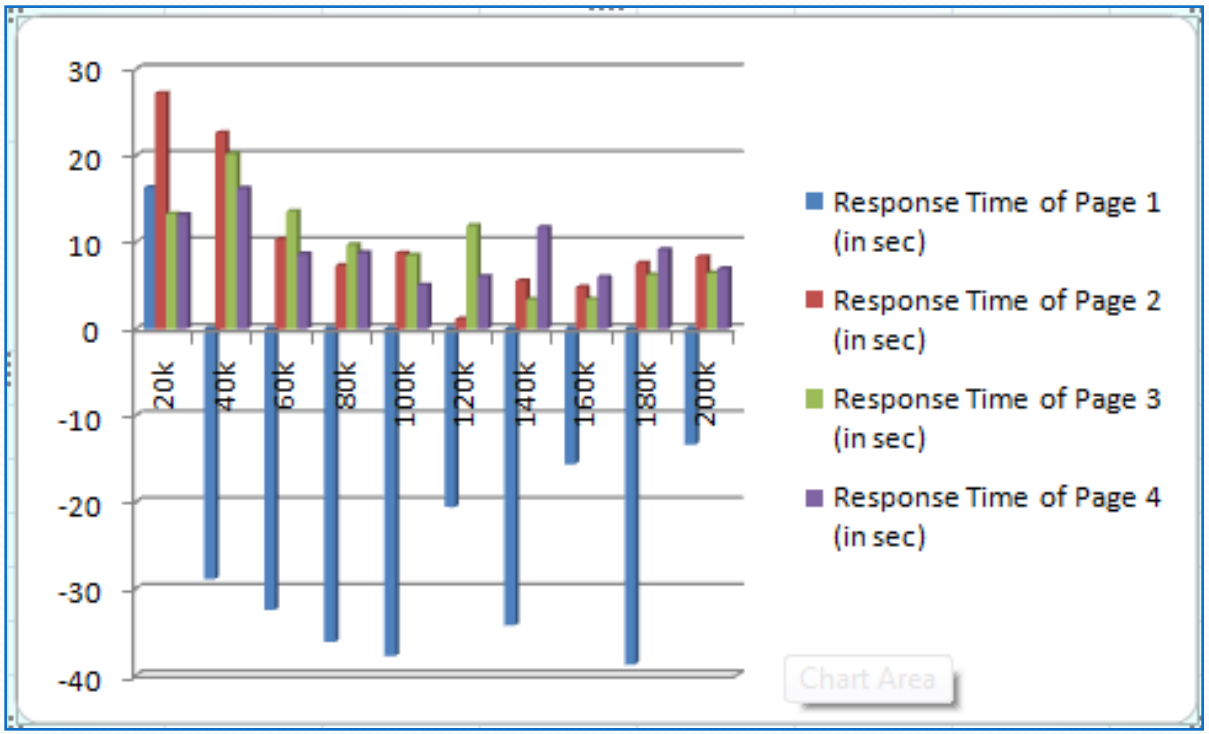
Testing (Contd..)

- Phase 2: Analyze the performance of our caching system
- Four web pages
- five components – Content and four elements
- In a particular phase, all the pages have different content but contain common elements
- Increased the size of the elements by 20k in each phase
- Recorded the web page response times both using our new caching mechanism and conventional method

Results – Using Firefox Browser

- Maximum – 27.2%

Percentage Difference b/w Our new caching mechanism and conventional method

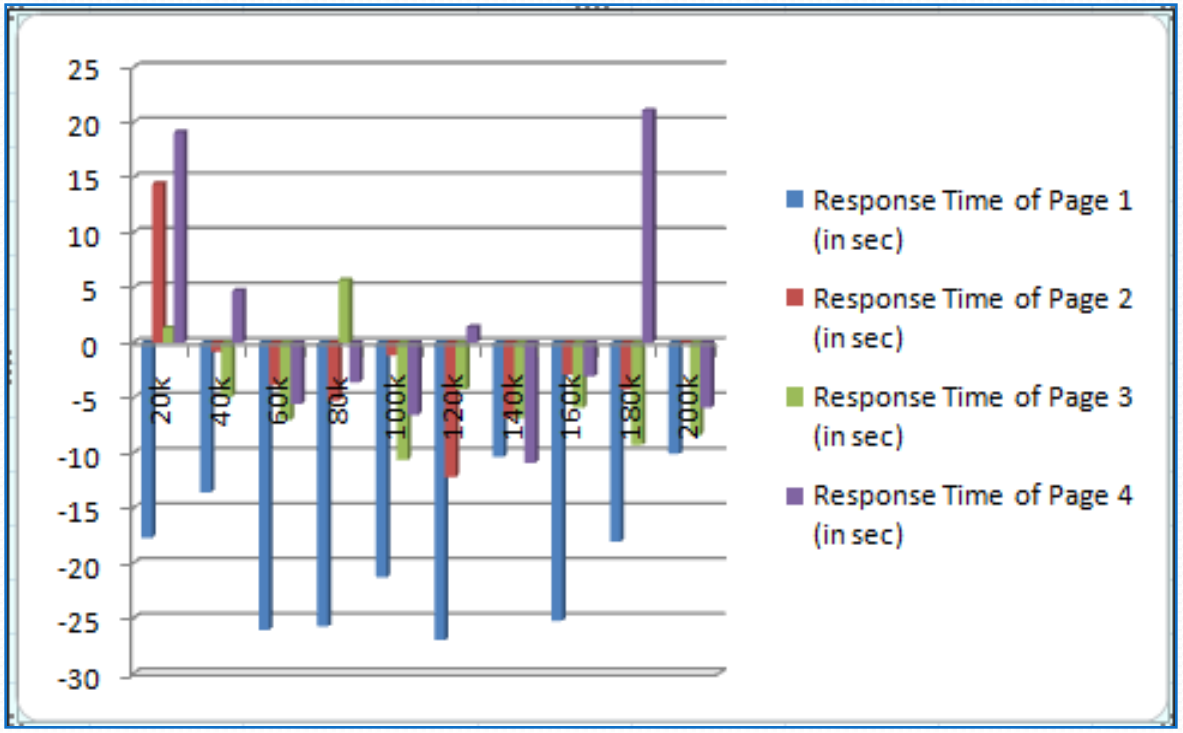


Size of the Element

Results – Using Internet Explorer

- Took a 500-600 ms to get the elements from its cache

Percentage Difference b/w Our new caching mechanism and conventional method

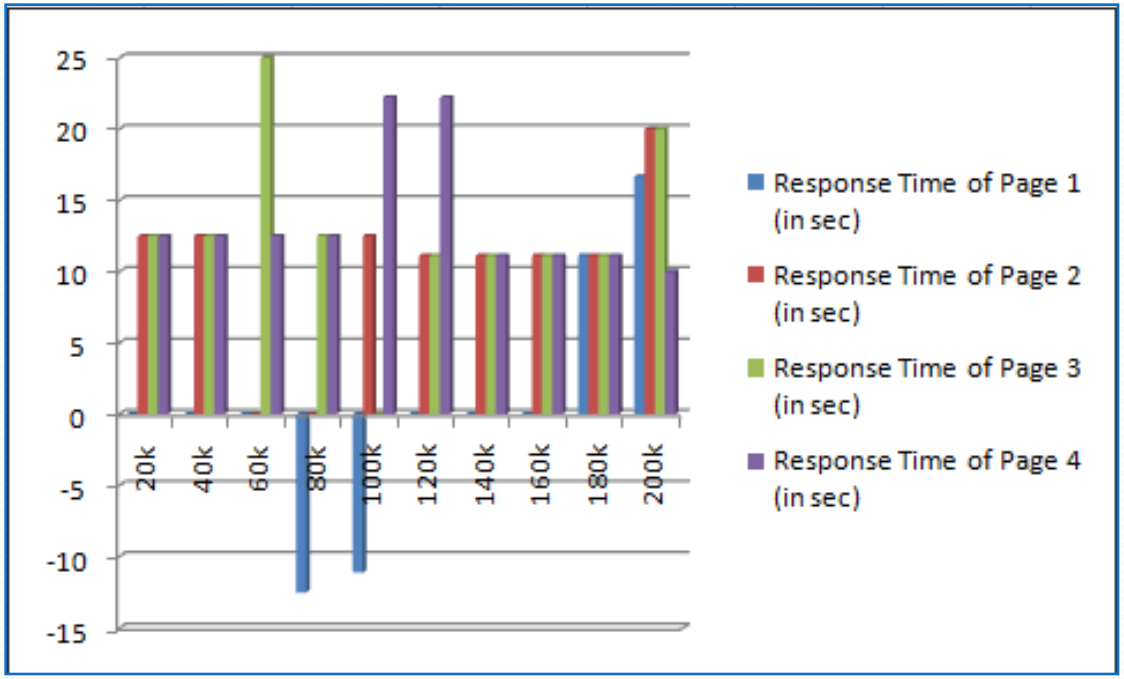


Size of the Element

Results – Using Opera

- Maximum – 25%

Percentage Difference b/w Our new caching mechanism and conventional method

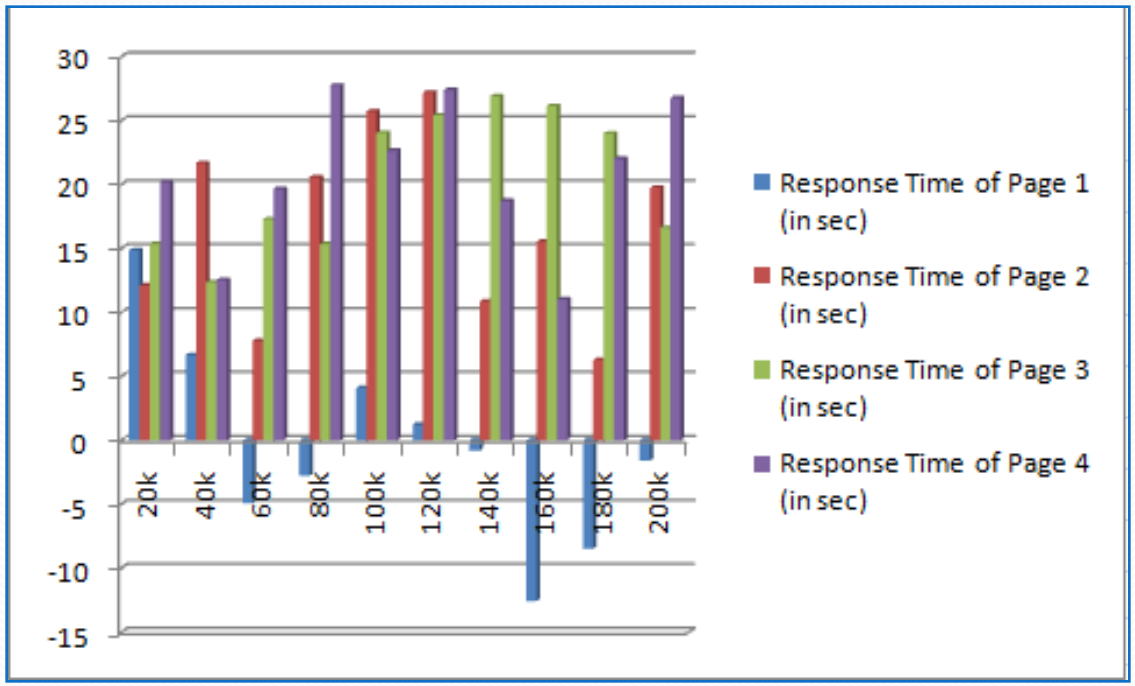


Size of the Element

Results – Using Safari

- Maximum - 27.7414 %

Percentage Difference b/w Our new caching mechanism and conventional method



Size of the Element

Observations

- Reduction in the response time of web pages can be obtained using our caching technique when the web pages are displayed in Firefox, Opera, and Safari
- Internet Explorer took more time to get the elements from its own cache
- YSlow helped us to further analyze the time taken by individual elements of the web pages in Firefox

Observations (Contd..)

- Firefox browser also took about 100 – 150 ms to get the Javascripts from its own cache
- The images are cached in the memory cache
- The external Javascripts and CSS files are cached in disk cache
- Therefore Firefox is taking few hundred milliseconds to get the elements from the disk
- If this can be reduced, using our caching technique there will be more improvement in the response times of the web pages

Conclusion

- Caching the most common elements of the web page on the client-side is definitely a new direction to the way how web caching is currently done.
- Using our new caching technique there is at least 15-25% reduction in the response time of the web pages
- However it also depends on the browsers mechanism as to how they obtain the external Javascripts from their own cache.

References

Best Practices for Speeding up Your Web Site. Retrieved September 27, 2008, from http://developer.yahoo.com/performance/rules.html

CakePHP. Retrieved November 12, 2008, from http://cakephp.org/

Cevasco, F. (2006). The CakePHP Framework: Your First Bite. Retrieved July 12, 2006, from http://www.sitepoint.com/article/application-development-cakephp

PHP. Retrieved November 12, 2008, from http://www.php.net/

Proxy Server, Retrieved December 8, 2008, from http://en.wikipedia.org/wiki/Proxy_server

Speed up your web pages with YSlow. Retrieved April 11, 2009, from http://developer.yahoo.com/yslow/

Squid: Optimizing Web Delivery. Retrieved November 13, 2008, from http://www.Squid- cache.org/

Thank you

Questions