Lecture 11: Case Study

CS178: Programming Parallel and Distributed Systems

March 5, 2001
Steven P. Reiss

I. Overview

A. Last two lectures looked at the architecture of web-based applications

1. Client is a web browser

a) With added code for the application

b) Code options

2. Web server acts as middleware

a) Code for the application is inserted here as well

b) Code options

3. Web components provide standard functionality

a) Authentication (MS Passport)

b) Real audio, ...

4. Server is as before

a) Controls access to resources

b) Handles clients through messages (or RMI)

B. This time I want to look at a case study and see the various tradeoffs and options

II. The Problem

A. Problem definition

1. We want to define a front end to Internet search

a) Takes a search query from user

b) If its ambiguous, asks user to disambiguate

c) Expands query with additional words

d) Sends new query to multiple search engines

e) Merges the result

2. **The server controls a semantic database**
    a) Given a word, return set of meanings
    b) Given a meaning, return a set of associated words
3. **User front end should be easy to use, fast, etc.**
    a) Should work over the web on most browsers
4. **Back end should scale to thousands of users**
5. **Back end might track info about a user**
    a) Allow personal definitions of terms
    b) Allow disambiguation based on past experience

B. An extension of this provides categories

1. **Idea is to provide context for the search**
2. **User selects category, gets various options for that category**
3. **Example: automobiles ...**

C. How would you go about building this

III. Client

A. What are the requirements

1. **Let user enter what they want to search for**
2. **Let user clarify what they want to search for**

B. What are the principles of a good UI

1. **Consistency**
2. **Feedback**
3. **Minimize error possibilities**
4. **Provide error recovery**
5. **Accommodate all levels of users**
6. **Minimize memorization**
7. **Meet response time expectations**
8. **The interface should look good**
    a) Visual clarity
    b) Visual codings
    c) Attention getting
    d) Layout: balance, griding, proportion, consistency

            e) Color

- **C. What might the user interface look like**
- **D. How would you implement this**
- **E. Suppose you wanted to allow categories**

IV. What goes in the server

- **A. Server needs to control a resource**
  - **1. Semantic database**
  - **2. Maps words to meanings**
  - **3. Provides meanings with words and related words**
- **B. Server needs to handle requests**
  - **1. Requests can come from various sources**
    - a) Command line for debugging
    - b) Sockets or RMI from web server
  - **2. Define a server model that manages sockets, files, etc.**
  - **3. Define a command class and a processor**
- **C. What are the commands**
  - **1. Request for Meanings of a word or phrase**
  - **2. Request for search**
    - a) Initial request
    - b) Specialize a previous search
  - **3. Handling users**
    - a) New user, set property, get property, login
    - b) Session ID
  - **4. Session handle**
  - **5. Handling categories**
  - **6. Server management**
    - a) Reload database
    - b) Shutdown
- **D. Multithreaded to handle multiple commands**
  - **1. Arbitrary threads (one per socket connection, ...)**
  - **2. Fixed number of threads and a work queue\**

E. Handling request for meanings

1. **Should be high priority**
2. **Should have immediate response**

F. Search request processing

1. **Lower priority (can wait)**
2. **Need to go out to multiple search engines**
3. **This should be done multithreaded**
4. **Parse and merge results as they come in**
5. **Return a new web page**

G. How might this be handled as a service

1. **XML return of items**
2. **Let client format it as they want**

V. What goes in the middleware

A. Pass requests on to server

1. **Let server generate valid http responses**
2. **Let server generate valid html**
3. **Trivial cgi-bin program that**
   a) Started server if it wasn't running
   b) Created socket to pass on request
   c) Waited for reply on that socket
   d) Passed response back to browser
   e) Closed the socket

B. Handling more users

1. **Moved to a servlet model**
2. **Servlet opened k sockets to the server at once**
3. **Requests were queued for one of these sockets**
4. **Sockets were kept open to the server**

C. Scaling even further

1. **Allow multiple servers**
2. **Servlet sends request to appropriate back end server**

VI. Experiences

A. User interface

1. Simple is better
2. Categories not used (never really set up anyway)

B. Server

1. Most commands handled very fast
2. Sometimes a command would take seconds rather than milliseconds

C. Application

1. Original database was inadequate
2. Categories were difficult to set up
3. Search engines keep changing output formats

VII. Next Time

A. Internet Agents

B. What is an internet agent

C. How to organize applications around agents

D. Homework

1. Find an example of an internet agent. This can be something that already exists or something that you would like to exist.
2. Describe a scenario as to how this agent might work.
3. We'll start next class by looking at these