CS 570 Programming Assignment

A Weather and Traffic Analytics System

Due Time: Feb. 21, 2016 @ 23:59:59

1. Objective

- Understand the components, the core technologies, the architecture and the protocols enabling a web service-based distributed system.
- Design and implement a web service-based distributed system for weather and traffic analytics.
- Design and implement big data analytics and prediction tools for planning vehicle routes.

2. Preparation

Web service technology provides a standard means of building a distributed system over the Internet. In simple terms, it provides methods for a sophisticated remote procedure call. The sophistication arises out of the elegant mechanisms it supports for enabling (1) various transparencies (platform, language, and hardware), (2) application to application data exchange and interoperability, and (3) composability of complex web services from a set of simple web services. The significant difference between the regular HTTP-based technologies and web service is the standardization realized through XML and SOAP (SOAP, XML over HTTP). Recently, Representational State Transfer (REST) is another protocol gaining popularity. All these techniques make web services an ideal fit for large-scale enterprise level application integration.

3. Implementation

3.1 Programming environment

You will write **Java** code that compiles in Linux (e.g., Ubuntu) or Windows system.

3.2 Overview

In this project, you are asked to build a multi-tier distributed system that consists of two major sub-systems, which are (1) an RMI-based simple data acquisition system and (2) a web service-based web application that processes and analyzes the collected data. The two sub-systems are loosely coupled through a database. The architecture of the system is shown in Fig. 1.

The RMI part: There are many websites that updates the national-wide weather information every day, e.g., Yahoo weather (https://weather.yahoo.com/). The RMI Server (box 2) needs to stream in the page and parses it for the relevant data and stores the weather information in a persistent storage. You need to accumulate the weather information for a period of at least 1 week (or 7 days). The data collected will be stored in a database (box 4). The daily data can be visualized using graphics on an RMI client (box 3).

The web service part: The data collected in the database will be processed by the server (box 5) for analytics information, such as the average temperature for a given city, the highest/lowest temperature for a particular date for a city and the rain/wind condition for a given zipcode. The web service client (box 6) will be able to query the server for various information related to the data collected. Your task is to design and implement the complete web services-based system indicated by Fig 1 and implement creative big data analytics tools.

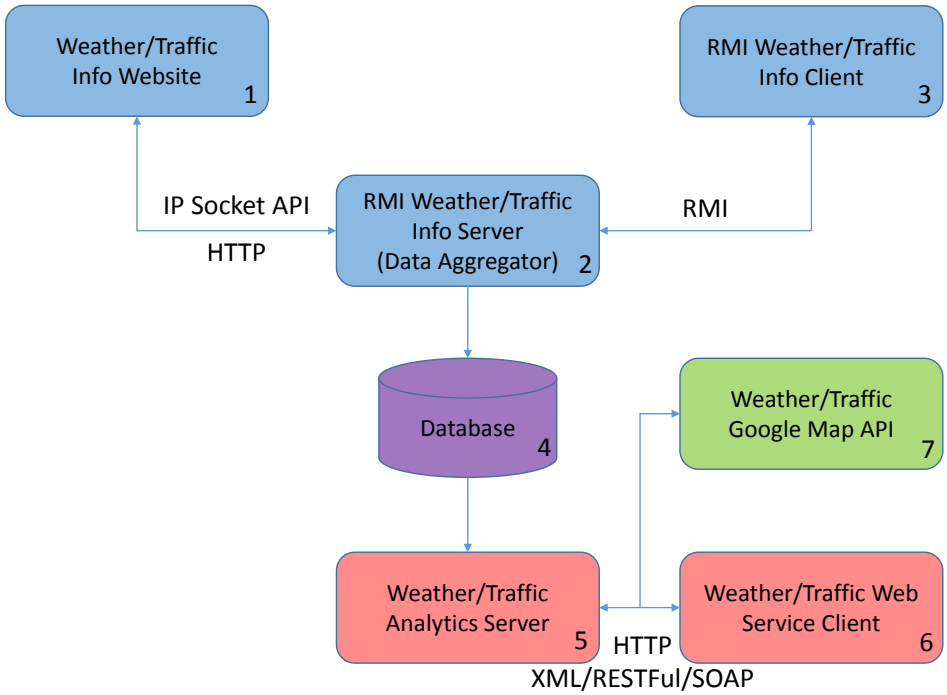

Fig. 1. The architecture of the weather and traffic analytics system

3.3 Detailed Implementation Steps

1. Getting used to building client-server systems: When you implement a simple client side application program there are just two steps involved: compile and execute the code. In a client-server system, you will have to take care the server side as well as the client side. On the server side, you will compile the code, generate stubs or proxies using special compilers, deploy the service, register and publicize the service for the clients to use. On the client side you will prepare the client code with appropriate stubs, and during execution lookup the service needed and use it. To understand the process, you can study the RMI-based system code and implementation from http://www.datadisk.co.uk/html_docs/java/rmi.htm. Deploy the data aggregation part and make sure you can collect the data and visualize it.

2. Working with the database: In this project you will store the data in a relational table and access it using SQL statements embedded in Java language. Or, you can use any database you like, e.g., MongoDB.

3. Building systems using build tools such as Ant: In order to tackle complexities in configuration and deploying server-side applications, you will need to use special build tools. Apache Ant is a XML-based build tool which similar to “make” utility that most of you are familiar with.

4. Study and understand the web services building and deployment details: For the web service part, you can refer to any "Hello World" web service example online, e.g., http://www.mkyong.com/webservices/jax-ws/jax-ws-hello-world-example/. This can provide framework and the client, a build file and a configuration file in XML. You need to build the server, deploy the server, build the client which is a web application and access the service provided by the server.

5. Design, implement and test your weather web service: Using the frame work given in the Step 4 above design the Web Service for dispensing and answering user queries about the weather information of various cities. This is expected to be the most time consuming part of the project due to the novelty of the topic.

6. Deploy the integrated system: The various components listed above were deployed and tested. In this step you will run the entire integrated system. The RMI part can be scheduled to acquire data once a day to update the database that will be used by the web service part.

3.4 Extension with Bonus Points

- In box 7, a user inputs "From" and "To" on. We figure out the weather predicted for a given route and display it in a user-friendly form. Here, we need to use a Google Map API web service. **[10 bonus points]**
- Instead of collecting the weather information, you can collect the vehicle traffic information in LA area and design creative big data analytics tools. For example, in box 7, given a "From" and "To" points on the Google Map API, the web service can provide 3 alternative routes with information such as time or distance for driving (similar to your Google Map App). Another example could be traffic pattern learning and prediction in LA area. **[20 bonus points]**

4. Submission and Grading

4.1 What to Submit

Your submission should contain a tar file – Name it as < your cin_name>.tar, which includes:

- All source files (.java files);
- A report that explains the detailed implementations and findings, which should be around **5 pages**.

4.2 How to submit

Use CS Network Service (CSNS) to submit the tar file.

4.3 Grading Criteria

- *Each group needs to show a demo during the office hour. All the members should attend the demo, and describe their coding contribution in this project.*
- Correctness of output. Please refer to the grading guidelines for more details.
- Organization and documentation of your code.