Alexandria University
Faculty of Engineering Computing
Computer and Systems Engineering
Fourth Year
Spring 2024

CS432: Distributed Systems and Net-Centric

Assigned: Monday 29th April 2024
Due: Saturday 18th May, 2024

Incremental calculation of shortest path in dynamic graphs.

Don't forget to read the notes and the grading policy.

Objective

This project aims at understanding the basics of RMI/RPC implementation and applying it.

Problem Description

In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized. This is a fundamental and well-studied combinatorial optimization problem with many practical uses: from GPS navigation to routing schemes in computer networks; search engines apply solutions to this problem on website interconnectivity graphs and social networks apply them to graphs of peoples' relationships.

In this project, the task is to answer shortest path queries on a changing graph, as quickly as possible. We will provide an initial graph that you may process and index in any way you find necessary. Once this is done, we will begin issuing a workload consisting of a series of sequential operation batches. Each operation is either a graph modification (insertion or removal) or a query about the shortest path between two nodes in the graph. Your program is expected to correctly answer all queries as if all operations had been executed in the order they were given.

The graphs are directed and unweighted. Input to your program will be provided via standard input and files, and the output must appear on the standard output and files.

Specifications

There are three operations that are applied to the original graph. The three operation types are as follows:

- **'Q'/query:** this operation needs to be answered with the distance of the shortest (directed) path from the first node to the second node in the current graph. The answer should appear as output in the form of a single line containing the decimal ASCII

representation of the integer distance between the two nodes, i.e., the number of edges on a shortest directed path between them. If there is no path between the nodes or if either of the nodes does not exist in the graph, the answer should be -1. The distance between any node and itself is always 0.

- **'A'/add:** This operation requires you to modify your current graph by adding another edge from the first node in the operation to the second. As was the case during the input of the original graph input, if the edge already exists, the graph remains unchanged. If one (or both) of the specified endpoints of the new edge does not exist in the graph, it should be added. This operation should not produce any output.
- **'D'/delete:** This operation requires you to modify your current graph by removing the edge from the first node in the operation to the second. If the specified edge does not exist in the graph, the graph should remain unchanged. This operation should not produce any output.

Each operation is represented by one character ('Q', 'A' or 'D') that defines the operation type, followed by a space and two positive integer numbers in decimal ASCII representation, also separated by a space. The two integer numbers represent node IDs.

Any sequence of operations can be produced by any node arranging operations will be based on your design choice.

Sequence

- First the initial graph is entered into your program's standard input. The graph is represented as a list of edges, each of which consists of a pair of node ids (the edge starting node, followed by the edge destination node) represented as non-negative integer numbers. Your program will receive multiple lines (each representing one edge) containing exactly 2 integer numbers in decimal ASCII representation separated by a single space. The initial graph ends with a line containing the character 'S'.

1. Your program's standard output should type a line containing the character 'R' (case insensitive, followed by the new line character '\n'). Your program uses this line to signal that it is done ingesting the original graph, has performed any processing, and/or indexing on it and is now ready to receive the workload.

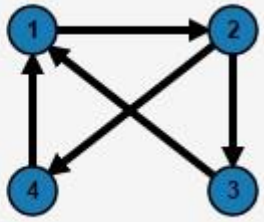
2. The workload is delivered in batch requests. Each request consists of a sequence of operations provided one per line followed by a line containing the single character 'F' that signals the end of the batch. Randomly generate the batches based on a random variable representing the percentage of writes.

3. After the end of every batch, a requester will wait for output from your program before providing the next batch. You need to provide as many lines of output as there are query ('Q') operations in the batch - each line containing the distance of the shortest path as described above. Your program is free to process operations in a batch concurrently or inter batch requests simultaneously. However, the query results in the output must reflect the order of the queries within the batches.

Examples

The initial graph is entered as follows

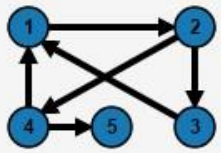
```
1 2
2 3
3 1
4 1
2 4
S
```



The following batches are then added to the graph

Batch 1:

```
Q 1 3
A 4 5
Q 1 5
Q 5 1
F
```

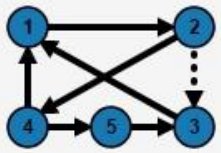


Output:

```
2
3
-1
```

Batch 2:

```
A 5 3
Q 1 3
D 2 3
Q 1 3
F
```



Output:

```
2
4
```

<u>You can batch the different operations preserving the order, as long as no query answer is stale (changed in the time of response).</u>

Alexandria University
Faculty of Engineering Computing
Computer and Systems Engineering
Fourth Year
Spring 2024

CS432: Distributed Systems and Net-Centric

Assigned: Monday 29th April 2024
Due: Saturday 18th May, 2024

Network Description

Server Specification

Our server is non-blocking that is serving several requests simultaneously while handling synchronization. The server thread needs to maintain the number of nodes and requests and processing times for performance reporting.

Reader and Writer Client Specification

Clients are started as processes. Readers send their requests to the server as read and receive a packet that contains the answer.

To simulate a real situation, each reading and writing operation takes a random amount of time (0 to 10000ms). A node must sleep for a random time between any two requests (recall that no request is sent before the previous is answered).

How to Start Processes on Remote Machines To start a process on a remote machine, you should use the remote login facility ssh in Unix systems. It accepts the host (computer) name as a parameter and commands to execute separated by semicolons.

RMI Implementation

RMI Registry

RMI registry is a Naming Service that acts as a broker. RMI servers register their objects with the RMI registry to make them publicly available. RMI clients look up the registry to locate an object they are interested in and then obtain a reference to that object in order to use its remote methods. Of course, servers register only their remote objects, which have remote methods.

The Remote Interface

Remote methods that will be available to clients over the network should be declared. This is accomplished by defining a remote interface. This interface must extend java.rmi.Remote class and must be defined as public to make it accessible to clients. The body of this interface definition consists of only the declaration of remote methods. The declaration is nothing more than the signatures of methods namely method name, parameters, and return type. Each method declaration must also declare java.rmi.RemoteException as the throws part.

System Configuration and Utility

Your main class is Start.java/c/cpp; it is responsible for starting the server and the clients. First, it should create a thread, which will run the server code. Then, it should start with clients. This program will read a configuration file and start up the system accordingly.

The system configuration is in the file ***system.properties*** as below:

```
GSP.server=192.168.1.4
GSP.server.port=49053
GSP.numberOfnodes= 4
GSP.node0=lab204.1.edu
GSP.node1=lab204.2.edu
GSP.node2=machine3
GSP.node3=machine4
GSP.rmiregistry.port=1099
```

GSP.server is the address of the host on which the server runs. GSP.server.port is the well-known port number on which the server socket will be listening. GSP.node0 is the address of the host on which the node with ID 0 runs, and so on. GSP.numberOfnodes is the number of nodes. GSP.rmiregistry.port specifies the port of rmiregistry (we assume that the rmiregistry runs on the same host as the GSP sever).

Note that the above is only a sample configuration file, you need to update it according to your machine's names/IPs.

Implementation of Remote Interface

After defining the remote interface, you need to implement it. Implementing the interface means writing the actual code that will form the bodies of your remote methods. For this purpose, you define a class from which you create your remote objects that will serve the clients. This class must extend RemoteObject provided by the java.rmi.server package.

There is also a subclass UnicastRemoteObject which is also provided by the same package that provides sufficient basic functionality for our purpose. When you call its constructor, necessary steps are taken for you so that you will not need to deal with them. An RMI server registers its remote objects by using bind() or rebind() method of Naming class.

RMI Clients

RMI clients are mostly ordinary Java programs. The only difference is that they use the remote interface. As you now know remote interface declares the remote methods to be used. In addition, the clients need to obtain a reference to the remote object, which includes the methods declared in the remote interface. The clients obtain this reference by using lookup() method of the Naming class.

Readings

An Overview of RMI Applications: http://docs.oracle.com/javase/tutorial/rmi/overview.html

Server-Client Specifications Reminder

- Output files must be named like log0, log1, etc. These are to be plain text files.
- All the file names must be exactly as specified in this document.
- nodes should be given names 1,2, 3,...,n (where n is the number of nodes).
- The server log format should be detailed in the report.
- The client logs should be at the clients' machines and their format should be detailed in the report.

Alexandria University
Faculty of Engineering Computing
Computer and Systems Engineering
Fourth Year
Spring 2024

CS432: Distributed Systems and Net-Centric

Assigned: Monday 29th April 2024
Due: Saturday 18th May, 2024

Performance Analysis

Functional Requirements (50%)

This includes the functionality of the program:

1- RMI implementation, a request is sent and received successfully.
2- A minimum of two client nodes should be able to contact the server node (three nodes correctly operational).
3- The initial graph processing and queries are correct.

Basic Operations (50%)

This includes some basic operations

1- Implementing another variant or method to enhance the response time.
2- Performance analysis that includes the following
    a. Recording response time Vs frequency of requests for the two variants (every point is a summary (choose your measure) of several readings). Include your table in the report.
    b. Recording response time Vs percentage of add/delete operations of requests for the two variants (every point is a summary (choose your measure) of several readings). Include your table in the report.
    c. Recording response time Vs number of nodes [1,5] for the two variants (every point is a summary (choose your measure) of several readings). Include your table in the report.

Stress Testing (extra 10%)

We will need to test performance of the server under stress for a wider range of nodes, i.e. Recording response time Vs number of nodes [5,15] for the two variants (every point is a summary (choose your measure) of several readings). Include your table in the report.

Alexandria University
Faculty of Engineering Computing
Computer and Systems Engineering
Fourth Year
Spring 2024

CS432: Distributed Systems and Net-Centric

Assigned: Monday 29th April 2024
Due: Saturday 18th May, 2024

Notes

- This assignment is adapted from the ACM Sigmod Programming contest 2016.
- Develop this assignment in Java or C/C++ use the alternative RMI.
- However, for this project the required is that your program correctly follows the steps above.
- Your solution will be evaluated for correctness and execution time. Execution time measurement **does not start until your program signals (with 'R') that it is finished ingesting the initial graph.**
- Concurrent request execution within each batch is allowed and encouraged, as long as the results mimic a sequential execution of the operations within the batch. In particular, the result for each query must reflect all additions and deletions that precede it in the workload sequence, and must not reflect any additions and deletions that follow it.
- You should deliver a report, please follow the template.
- You can reuse shortest path Algorithms (***ONLY***) or code fragments (***AFTER CITING THEM AND STATING IN THE REPORT WHY YOU CHOSE THEM***… And no, "it was the first one in my search" is not a reason).

Grading Policies

- No Late submission is allowed.
- You should work in groups of 3 or 4 students.
- Submit your code by zipping up your files and naming the zip file. Project_id1_id2_id3_lang.zip where lang is the language you used (java/c/cpp).
- Plagiarizing is not acceptable. Sharing code fragments between groups is prohibited and all the groups that are engaged in this action will be severely penalized. Not delivering the project will be much better than committing this offense.