

Assignment #3: Performance Comparison

DUE: 10:30 am, Friday, May 7th.

In this assignment, you will be investigating the relative performance of direct socket communication, CORBA, and Java RMI. For each technology, you will build a simple “Counter” object to carry out a variety of timing experiments. In addition, you will build on your experience with internet connections from lab 1 to construct a stock information tool.

For both parts you will submit, using the CSE submit command, your (carefully documented) code. In addition, for the first part, you will also turn in a written lab report describing and analyzing the data you have gathered.

Both parts of this assignment are quite open-ended, so there is plenty of latitude in your design space for some very creative (and robust and thorough) solutions.

Part I: Performance Evaluation

Counter Interface

Consider the following Java RMI interface:

```
interface CountApp extends java.rmi.Remote {
    void reset() throws java.rmi.RemoteException;
    void inc() throws java.rmi.RemoteException;
    void inc_param (Vector v) throws java.rmi.RemoteException;
    long read() throws java.rmi.RemoteException;
}
```

An implementation of this interface should maintain an integer variable. The effect of `reset()` is to set this variable to 0. The effect of `inc()` (and `inc_param()`) is to increment the current value by one. The effect of `read()` is to return the current value (leaving it unchanged).

Write a server that creates a Counter and a client program that finds this object, resets its value, repeatedly increments the value, then reads the result. Implement this client/server pair using: (i) plain sockets, (ii) CORBA, and (iii) Java RMI.

Timing Experiments

Using your implementations of the Counter and corresponding client, collect timing data for how long an invocation takes in each of the three basic middleware technologies (sockets, CORBA, and RMI). To gather an accurate number, you should measure the time required for many invocations. For example:

```
startTime = System.currentTimeMillis();
for (int i=0; i<NumTrials; i++)
    counterRef.inc();
stopTime = System.currentTimeMillis();
System.out.println ("Avg ping time = " + ((stopTime-startTime)/(float)NumTrials) + "ms");
```

Gather and compare data for invocations while varying a number of factors. You should, at a minimum, consider varying:

- whether the client and server are located on the same machine vs. different (remote) machines,
- the size of parameter lists (using `inc_param()`),
- the types of objects passed as parameters (eg arrays of basic types such as ints), and
- synchronous vs asynchronous messages, as appropriate.

You may extend the original `CountApp` interface to include other methods if you feel they would help you gather more information. Remember to form a specific hypothesis and gather data to explicitly test that hypothesis.

What to turn in

Turn in a description of your timing experiments. This should include (i) experimental methods (a discussion of how you gathered these numbers and how you controlled for various uncertainties) (ii) the data itself, in a clear manner (tables, graphs) (iii) a discussion and analysis of what you observed, including a conclusion and summary.

Also submit the code you wrote for these experiments. Include a README file that clearly shows the sequence of commands needed to compile and run your implementation.

Part II: Expanding on Internet Skills

Write a simple command-line program that accepts a stock ticker symbol as its only argument and outputs the name of the corresponding company and the most recent trade price for that stock. The quote returned by your program can be a 15-minute delayed quote, as is freely available from many sources. For example:

```
nu% stockquote INTC
Intel Corp - Last Trade: 26.98
```

Upon request, your grader can provide you with hints and suggestions, with appropriate deductions from your final score.