Measuring Performance of Distributed Systems

Objectives:
1. Learn how to measure floating point arithmetic performance in **M**illions of **FL**oating-point **OP**erations a **S**econd (MFLOPS).

2. Learn how to measure network latency of a computing environment.

Laboratory Assignment:
Note: In order to achieve useful measurements, this lab must be done in 213 Dana. Do not do the lab remotely.

1. Measuring MFLOPS of a Computing Environment

Different computer environments perform differently. Computational performance depends on many variables including the hardware platform, the operating system, the programming language, the compiler and the compiler options used.

In this exercise, we want to measure the performance of floating point arithmetic in a programming language that is considered "fast" (Fortran 77) and compare it with Java. Since Java is considered slow, we want to measure how slow.

Copy the Fortran 77 file `mflops.f` from `~cs355/Lab10` and study it. This is a modified version of a standard benchmark to compute the MFLOPS of a computer platform.

Compile the Fortran 77 program by typing the following at the Unix prompt:

```
g77 mflops.f -o mflops
```

Note: The Suns use `f77` while Linux uses `g77` for the name of the GNU Fortran compiler.

Run the run unit: `mflops`

Sometimes when you are measuring computer performance, you will get "funny" results due to a fluke in the system. Always run a test several times to gain confidence in the results. If the results are varying wildly, you need to investigate why. In this case, you may need to adjust the value of `Numint` in the program.

What is Fortran MFLOPS? _____ What is host name used? _____

Today's compiler technology is very sophisticated. Though the g77 compiler does some optimization of the generated machine code by default, you can request the compiler to spend extra time to optimize the generated code even further. Type the following and rerun the run unit on the same host. The new option is a capital Oh.

```
g77 -O mflops.f -o mflops
```

What is Fortran MFLOPS using O optimization? _____

What do you conclude about compiler optimization? _____

Using the Fortran code as an example, write a corresponding Java program. In place of `ETIME()`, use Java's `System.currentTimeMillis()` for timing. Be warned that the resolution of Java's `System.currentTimeMillis()` is only one millisecond and it measures real time and *not* CPU time. Therefore, when you run these tests make sure your system is lightly loaded, i.e., nothing else running. Use the Unix `top` command to check that no one else is on your machine.

A small Java program that uses `System.currentTimeMillis()` can be found at

`~cs355/Lab10/TimeTest.java`.

Make sure you are using the Java 1.5 compiler. Check this by using: `java -version`

What is performance of Java version of program in MFLOPS? _____

What is ratio of Fortran MFLOPS without O optimization to Java MFLOPS? _____

What is ratio of Fortran MFLOPS with O optimization to Java MFLOPS? _____

2. Turn off JIT

The `java` command uses SUN's *Just-In-Time* (JIT) compiling feature by default. That is, SUN's JIT feature converts compute-intensive sections of Java byte code to C which is compiled. To turn off the JIT feature, use the -D option as follows:

```
java -Djava.compiler=none classname
```

What is non-JIT Java performance in MFLOPS? _____

How many times faster is JIT version over non-JIT version? _____

How well does the JIT feature work? _____

3. Network Latency - `traceroute`

The purpose of this and the next two exercises is to measure network latency between two workstations in room 213 Dana. User end-to-end network latency depends on the hardware platform, the operating system, the network protocol and the application. We want to measure the latency three ways - 1) traceroute, 2) across a Java stream socket and 3) a Java RMI request to a remote object. If you can't remember how to use `traceroute`, see Lab 3.

What is the time in milliseconds for `traceroute` between two Linux workstations in 213 Dana? _____

Names of the two workstations used: _____ and _____

Since `traceroute` measures the round trip time, the network latency is half the time. Network latency in milliseconds : _____

4. Network Latency - Java Stream Socket

Write a Java application that sends one character as string to a second Java application across a Java Stream Socket which sends a different character back. Do this 100 times and time it. Make sure no other input or output statements are in the timed section. For comparision purposes, use same workstations as in exercise 3. Use the `ObjectInputStream` and `ObjectOutputStream` classes.

What is measured network latency in milliseconds to send one character: _____

5. Network Latency - Java RMI

Write a Java application to call a remote object where you pass one character as parameter and return one character. Do this 100 times and time it. Again use same workstations as in exercise 3.

What is measured network latency in milliseconds to send one character: _____

Hand In:
Hand in a copy of the listing of the Java code and sample output for each of the exercises 1, 4 and 5. Also, hand in the answers to the questions.