Distributed Calculator

By

Jon Vegard Jansen and Robin Tollisen

Project report for compulsory assignment in IKT414 in Spring 2013

Faculty of Engineering and Science
University of Agder
Grimstad, 23th of January 2013

Status: Final

Keywords: Distributed system, Calculator, C#, XML-RPC

Abstract:

This report reviews a mandatory task given at the University of Agder, campus Grimstad, in the course IKT414 Distributed and Agent-Based Systems. The task is to build a calculator, which divides its workload on four different machines.

Contents

1 Introduction

In this chapter we highlight the background for this task, as well as our problem statement.

1.1 Background

The task is part of the mandatory assignments in the course IKT414 Distributed and Agent-based Systems, at the University of Agder, campus Grimstad. During lectures we have learned about what a distributed system is, and can thus try to build our own.

1.2 Problem Statement

The goal of this task is to program a calculator, that runs different different kind of operations on four different machines. There should be one master, which takes care of input, user interface and simple calculator operations; Addition, subtraction, multiplication and division. The second machine takes care of modulus and factorial, while the third and fourth respectively perform sinus/cosine and power. It is also required that the machines use XML RPC to communicate with each other. [1]

2 Implementation

We implemented our calculator using C#. For communication between the machines we used the XML-RPC.net library, which handles communication between machines using the XML RPC protocol. [1]

The implementation uses one master machine which delegates tasks to three others when it receives an expression to calculate.

2.1 Evaluation of expressions

Following is the basics of how our calculator functions when it receives an expression:
1. Does some simple formatting required for correct reading, such as lower casing all letters, replacing . with , for decimals etc.
2. Splits out all paranthesis by starting at the first left paranthesis it can find, and replaces it with a placeholder so that it knows which paranthesis belonged there.
3. Then for each paranthesis it does the following:
3.1 Puts back in values for any placeholders in the current parantheses, which starting at the leftmost left-parantheses ensures that expressions such as cos(cos(sin(5mod4))) would work correctly.
3.2 Splits the current parantheses by basic operators and puts placeholders.
3.3 For each of the expressions between basic operators it checks whether they contain cos, sin, mod, ^ or ! and sends them to another machine for calculation if they do.
3.4 Replaces calculated values back into the original parantheses expression.
3.5 Calculates * / + and - in the expression in said order.
3.6 Returns the paranthesis value.
4. Returns the final value.

2.2 User interface

We chose to use a command line for the user to give input, instead of a graphical user interface. Since the task was not graphically or user-friendly related, we did not find it as useful.

3 Discussion

Our implementation does not use any sort of threading, since we currently wait for each machine to calculate their part when distributing work. Although this is one of the strengths with a distributed system, we did not see it important to speed up the calculator, as it is performing very well already.

We looked at using binary trees to evaluate expressions, but we found it more convenient to operate on strings instead of first implementing a binary tree, based on the expression. While iterating through a binary tree, for instance using in-order traversal, the difficulty connected to build the tree in the first place, seemed more difficult than our solution.

The calculator takes input from a command line, and not a graphical user interface (GUI). Using a GUI takes a little more time to create, with respect to graphics and design, but it would have given the user more input restrictions. Since we chose to not use a GUI, the program has to do more work on the input string, before it can calculate anything. Also, the user has to be careful to use parentheses, where required.

4 Conclusion

While we would like to add threading of some sort, and myabe a GUI, our distributed calculator functions well, and computes correctly for all expressions we have tried so far. Input can be a bit picky on requirements for parantheses, but it works and performs well.

5 References

[1] http://xml-rpc.net/ - XML-RPC.Net