KTH Computer Science
and Communication

Code Evaluation System in Microsoft .NET Framework

Evaluating the performance of different languages on the Microsoft .NET platform

WILLIAM LUNDIN FORSSÉN

Master's Thesis at NADA
Supervisor: Alexander Baltatzis
Examiner: Olle Bälter

TRITA xxx yyyy-nn

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus in arcu dui, ac pretium ipsum. Pellentesque iaculis tincidunt sapien in iaculis. Etiam eget justo eget risus luctus rutrum in ut neque. Nam id dolor vel nisl luctus vestibulum at nec nulla. Pellentesque sagittis, erat vitae imperdiet fringilla, nulla nunc sodales libero, id viverra justo odio id metus. Nam tortor libero, consequat ac imperdiet eu, aliquet in nisl. Aliquam erat volutpat. Vestibulum enim metus, dictum a fringilla in, commodo vitae sem. Vestibulum porttitor nisl vitae libero semper sollicitudin. Etiam rutrum laoreet quam vel tristique. Pellentesque turpis lacus, vestibulum in tempor eu, ullamcorper id nisi. Vivamus a molestie sem. Sed commodo mattis neque, ac convallis mauris tristique non. Donec dui purus, aliquet id pellentesque vel, porttitor at nunc. Praesent elit justo, porta tincidunt venenatis sit amet, sollicitudin quis sem.

Donec eu nunc orci, vel pretium lectus. Sed augue mauris, porta id convallis quis, consequat sit amet lacus. Nulla venenatis tortor at risus semper dictum. Nulla ut mi eu orci dapibus vulputate. Morbi tincidunt pharetra ultrices. In fringilla velit ut enim auctor luctus. Etiam venenatis blandit condimentum. Nam fringilla aliquam leo, a aliquam arcu malesuada vitae. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque tempor nulla non dolor sagittis venenatis. Integer felis nibh, porttitor in condimentum ac, auctor quis dui.

Fusce eget tincidunt nibh. Quisque sed velit felis. Suspendisse in sagittis lorem. Fusce posuere ipsum id leo facilisis sit amet sollicitudin enim egestas. In augue massa, tincidunt ac blandit faucibus, ultricies nec tortor. Fusce et dui diam. Cras nec eros id dui laoreet lacinia. Proin malesuada est vel erat pharetra rutrum faucibus nulla porta.

Referat

System för kodevaluering i Microsoft .NET Framework

Donec eu nunc orci, vel pretium lectus. Sed augue mauris, porta id convallis quis, consequat sit amet lacus. Nulla venenatis tortor at risus semper dictum. Nulla ut mi eu orci dapibus vulputate. Morbi tincidunt pharetra ultrices. In fringilla velit ut enim auctor luctus. Etiam venenatis blandit condimentum. Nam fringilla aliquam leo, a aliquam arcu malesuada vitae. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque tempor nulla non dolor sagittis venenatis. Integer felis nibh, porttitor in condimentum ac, auctor quis dui.

Foreword

This report is my Master Thesis project which was carried out at Valtech in Stockholm. This thesis is part of my last course in the Masters Programme in Computer Science at The Royal Institute of Technology (KTH). The most difficult part of this thesis was finding out what subject to research. I wanted to make something in Microsoft .NET Framework, but I didn't know what. Then one day some words from my supervisor Alexander Baltatzis echoed in my head from one of his lectures "Write code in any language and run it on .NET". That sentence sparked my interest in finding out whether it was actually possible or not, since most people who write programs on the .NET platform do so using C# or Visual Basic, you barely ever hear about someone using any other language (even though many others are supported by the .NET platform).

Glossary

Term	**Description**
CIL	Common Intermediate Language, the lowest level human-readable programming language in .NET Framework.
CLI	Common Language Infrastructure, a specification that describes the runtime environment of Microsoft .NET Framework.
CLR	Common Language Runtime, is the virtual machine of Microsoft .NET Framework.
GUI	Graphical user interface, an interface that is image oriented rather than text oriented.
IKVM	An implementation of the Java for Mono and Microsoft .NET Framework.
IL	Intermediate Language, a language of an abstract machine.
IronPython	An implementation of the Python programming language targeting the Microsoft .NET Framework and Mono.
Mono	An open source project created to enable .NET Framework cross-platform compatability.
MSIL	Microsoft Intermediate Language, a synonym for CIL.

Contents

Chapter 1

Introduction

1.1 Problem Statement

A common challenge for Valtech and other consulting companies is to recruit the right people. The recruitment process in the IT business often involves several steps and can be very time consuming. This is due to the fact that knowledge and skill differ greatly from programmer to programmer and therefore a need to interview many people arises (just to find one that is deemed suitable for the job).

The recruitment process usually involves some kind of test to assert the skill of the programmer. The test is sometimes performed on paper (questionnaire) or on a white board. Both of these environments arenât really suitable to program in, hence most programmers tend to do their work on computers with the assistance of Google and a suitable IDE.

These two issues present a unique challenge, a challenge in which the optimal solution would result in less time spent recruiting and at the same time asserting the skills of the programmers before an interview is considered.

1.2 Goal

The main goal was to evaluate a system which could satisfy the demands stated above. Thus, there was a need to first build the system and then evaluate it through test cases with performance and memory consumption in mind.

1.3 About Valtech

Valtech is an independent, global IT-consulting company with offices in Europe, United States and Asia. The company was founded 1993 in France and has over 1600 employees worldwide. The Swedish branch is primarily oriented towards web solutions. Valtech's most well-known award winning projects include the websites 1177.se, Antagning.se and Riksbank.se.

All work concerning this thesis was carried out at Valtech in Stockholm, Sweden.

Chapter 2

Background

In this chapter the history of automatic code evaluation is explored.

2.1 What is automatic code evaluation?

Automatic code evaluation, or automatic grading, is a computer system that has the ability to judge code. This is usually done in the following steps:

1. A user is given a programming task or problem.
2. The user attempts to solve this problem by writing code.
3. The user sends this code to the automatic code evaluation system.
4. The system compiles the code (if needed).
5. The resulting program is then run by the system with some test data as input.
6. The system verifies that the program output is correct.
7. An answer is then returned to the user indicating the status of the code which he or she submitted (i.e. “Accepted” or “Wrong answer”).

2.2 History

Evaluating code automatically is nothing new. The earliest known system was built in 1960 by Hollingsworth (referens Hollingsworth). This system was used in an introductory course in Algol programming. The results from using this student-system approach rather than the traditional student-teacher was that it cut costs considerably for the staff since the time they needed to grade the students work was reduced by as much as one third. The students themselves also spent less time, since they were able to have their work graded immidiately instead of waiting for a teacher to do it. This system also made it possible to considerably increase the number of students taking the course. It did, however, have some shortcomings. For instance, a student's program could modify the grader program, making cheating possible.

An article from 2005 describes three generations of automatic graders (referens p1-douce). The first generation systems were those regarded as being built and/or used in the 1960's and 1970's. Unsurprisingly, they were using code that were close to pure machine code, some even used punched cards. In order to make them work, it was necessary to modify both compiler and operating system.

The second generation introduced script-based tools which not only involved various verification schemes but also asserted that the code was written in a certain manner (decided by the teacher). Typically these graders involved some sort of command-line GUI and languages like C and Java were used extensively.

The third generation differ from the second primarily in two ways. One is that they mostly use web based GUIs. The other is that they make extensive use of plagiarism detection since it had become more common for students to share code amongst each other. Some minor problems arose among these detection systems (referens (ja det finns en sÃ¥dan)???). If the programming task was too simple or if a lecturer had been excessively thorough when describing the homework, the submissions would tend to be very much alike and thus picked up by the plagiarism detection system. This made it somewhat difficult to distinguish between real plagiarism and the false positives.

2.3 Todays systems

Chapter 3

Method

This chapter discusses the implementation details of the system.

3.1 IKVM

IKVM.NET is an implementation of Java for Mono and the Microsoft .NET Framework. It includes the following components:

- A Java Virtual Machine implemented in .NET
- A .NET implementation of the Java class libraries
- Tools that enable Java and .NET interoperability

It also includes (among other things) a Java bytecode to .NET IL translator, which is what this project used to achieve its Java compatability.

3.2 IronPython

The website for IronPython describes it in the following way: "IronPython is an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework. IronPython can use the .NET Framework and Python libraries, and other .NET languages can use Python code just as easily." (Source: http://ironpython.net/)

3.2.1 IronPython vs Python

The HelloWorld test

This test simply prints "Hello World". In terms of performance, Python has a strong lead on IronPython. This would suggest that IronPython has quite the overhead (refer to later section about overhead testing).

Results Python 2.7: 0.0677239 seconds IronPython 2.7: 1.0527518 seconds

Source: http://ironpython.codeplex.com/wikipage?title=IP27A1VsCPy27PerfreferringTitle=IronPyt

PyStone 1.1

This test is just a rewrite of the Dhrystone test for the C programming language. In short the test intends to represent general processor performance by using integer only operations. Due to the fact that IronPython uses .NET Dynamic Language Runtime it can also use site caching. In short site caching basically remembers previous performed operations (i.e. a+b) which is why IronPython outperforms Python (might need reference to site caching here).

Results Python 2.7: 41187 PyStones/second IronPython 2.7: 52101.1 PyStones/second

Source: http://en.wikipedia.org/wiki/Dhrystone Source2: Source: http://ironpython.codeplex.com/w

PyBench 2.0

This is a collection of low level benchmarks. For example it tests string concatination and comparisons between different data types. More can be read at about the specific tests at [ref same as above]

Overall results: Python is 61.6% faster than IronPython.

Source: Source: http://ironpython.codeplex.com/wikipage?title=IP27A1VsCPy27PerfreferringTitle=

3.3 About CELINE

3.4 Implementation

3.4.1 IKVM

3.4.2 IronPython

Appendix A

RDF

And here is a figure

Figure A.1. Several statements describing the same resource.

that we refer to here: A.1