SCIENTIFIC AND INDUSTRIAL BENCHMARKING SYSTEM

Anđelko KATALENIĆ, Tihomir ŽILIĆ, Željko ŠITUM, Josip KASAĆ

Department of Control Engineering
Faculty of Mechanical Engineering and Naval Architecture
Ivana Lučića 1, 10000, Zagreb
Croatia
E-mail: <andelko.katalenic@fer.hr>

Abstract:
Many WEB sites offer different approaches for learning engineering and scientific topics, but is there such an online system that offers the way for collaborative solving of benchmark problems in engineering and scientific domain. Here we describe a novel online system that enables large teams of engineers to solve their problems. In order to describe and solve the problems, programming languages like Matlab, C++, Python, and PHP etc. are used. Insight into solving efficiency is achieved by comparing different programming algorithms for the same benchmark problem, so the best algorithm can be determined.

Keywords:
Scientific benchmarking, Collaboration, Online code execution.

1. INTRODUCTION
We’ve developed a new plug-in for our collaboration system called Scriptrunner [1] which enables people to easily share and present their progress on certain tasks and work together in scientific and industrial research environment.

As a part of Scriptrunner, “Benchmark plug-in” includes all of its good stuff such as: the remote file system with sharing capabilities, the user management administration, the online source compilation and the execution of the programs written in mathematical (Matlab/Octave/Scilab) and programming languages (C++, Java, Python, PHP).

The idea behind Benchmark plug-in is to create the software which enables large teams of engineers to run their problem solving source code on-line and present its outputs to others, together with complete benchmark analysis [2, 3, 4, 5] including execution time and memory usage.

2. GENERAL IDEA AND MOTIVATION BEHIND BENCHMARK PLUG-IN
Many companies these days use tools such as MATLAB, OCTAVE, Mathematica or SciLAB in their research and development. Scriptrunner’s feature “Benchmark plug-in” compiles and runs all of these programming languages together with many more including: Java, C++, Python, PHP and enables easy sharing of results of execution.

On the other hand, Scriptrunner’s sharing and messaging capabilities enable team leaders to easily manage and track people working on certain task and discuss things that should be done next or are done wrong.
Benchmark plug-in distinguishes 3 user types:

- **Team leader** creates Benchmark, assigns tasks to other people and has access to all of their present and past work including source codes, compilation history, execution results and comments.

- **Engineers** are the working force on some task. They solve certain problems and upload their step by step progress on Scriptrunner. They are given a possibility to share some or all of their work with other engineers or spectators.

- **Spectators** are users who are not directly included in problem solving and are here just to watch and comment on results shared by engineers and team leader.

Each of different user types has different user interfaces which are created using following technologies: PHP, HTML, XML, CSS, Javascript, AJAX and SQL to give users similar feel to Windows Forms applications without annoying refreshes and stiffness of plain HTML.

### 2.1. Time and memory measurement

Programming languages used by Benchmark plug-in can be divided into standard languages and scripting languages. Standard languages include C++, Java and other languages which are first translated to machine code and than are run, while scripting languages include Matlab, Octave, Python and other languages which are executed line by line by an interpreter and therefore are normally not translated into machine code.

Each group requires a different approach for elapsed time and memory usage measurement. Code translated to machine language is run and managed by operating system which is Linux in our case so programs appear as processes whose execution time and memory usage can be tracked by external application. Additionally, Scriptrunner executes all applications trough wrapper program for security reasons which create a problem while trying to measure memory usage and execution time by 3rd party programs like GNU time.

Scripting languages require a different approach because memory usage and elapsed time of executing a script are not visible outside of interpreter program, so an inner approach is required. Those languages generally have some built-in functions which can be used for that purpose but it requires altering user code to some degree.

*Figure 1 – Compiled program benchmarking*

*Figure 1* shows the use of UNIX `ps` command for fetching current status of a program and gathering complete data by cyclic execution of `ps` command during program execution.
Further, it is shown that Benchmark measures maximum allocated memory during program execution and that the smallest measurable time is 10 ms.

Drawback of this method is lengthening of original program execution time due to the CPU resources taken by repetitive `ps` command execution.

Figure 2 – Matlab code benchmarking

On the other hand figure 2 shows how to measure execution time and used memory in Matlab. Statements which do that job are bolded in figure 2 and they need to be added by Benchmark plug-in automatically before execution. Statements `tic` and `toc` are internal pre-built into Matlab and output time elapsed between calling them. Benchmark plug-in requires start-up script/function to have filename `RUNME.m` so adding `tic` and `toc` statements at right place is not so difficult and ambiguous. On the other hand, statement used for outputting all current workspace variables sizes is `whos`, but global memory usage is a bit harder to detect, especially in multi-function programs because Matlab allocates different workspaces for each function. Solution to that problem lies in adding `whos` statement to the end of every m-function inside user program and additionally marking that data by delimiters so it can be easily extracted.

Java, Octave and Python programs are benchmarked similar to presented group member.

2.2. Input cases

The very important aspect of program evaluation is the process of testing it on different sets of input data. Benchmark plug-in enables Project managers to create input data and easily test programs created by engineers to their limit.

The mechanism for injecting data into programs differs between scripting languages and languages with compilers. While programs compiled into machine code can be fed with data trough standard input, code written in pure script languages can not, because it is run by the interpreter.

On the other hand, very important part of the case of testing is synchronization between project manager and engineers. For that purpose it is necessary to exactly define type and order of input data so engineers can write their programs accordingly.

<table>
<thead>
<tr>
<th>input.txt</th>
<th>RUNME.c</th>
<th>Execution syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5</td>
<td>.... scanf(&quot;%d %d&quot;, &amp;n, &amp;m); for (int i=0;i&lt;n;i++) for (int j=0;j&lt;m;j++) scanf(&quot;%lf&quot;, &amp;A[i][j]); ....</td>
<td>RUNME.c &lt; input.txt</td>
</tr>
</tbody>
</table>

Table 1 – example how to pass data to C++ program
Table 2 – example how to pass data to Matlab script

Table 1 and table 2 show simple examples of input data feeding systems. Input.txt or input.m is created by Benchmark plug-in prior to any program execution. As you can see, data is passed to compiled program (c++ in our case) trough standard input and to Matlab program trough RUNME function arguments. First column of both tables shows input data format which is important to be known by engineers so they can include code from second column in their programs. Explanation of input data is written in richtext format with pictures by project manager during Benchmark creation process.

2.3. Program execution

Benchmark plug-in needs a good system for executing program code and retrieving output data, and there is no better way for executing server-side programs in background than AJAX. Each time a program is supposed to be executed, Benchmark plug-in instances new xmlHttpRequest object and call program execution/data gathering PHP script in the background. This way, program can be executed multiple times for all defined input cases and outputted data can be shown and stored into database interactively without user having to wait or be bugged by window refreshes.

Security issues like accessing system files, using too much memory or too long execution time (infinite loops) are solved through sr_wrap wrapper program and constant memory and time measurements. Sr_wrap is a wrapper program which runs programs with limited user privileges instead of www-data privileges. On the other hand, because the executed program is constantly watched by Benchmark plug-in (figure 1), it can be terminated when it exceeds memory or time limits. Script languages like Matlab, Octave and Python are all previously run by Linux user with limited access so scripts written in them can do little harm to the system.

3. CREATING A NEW BENCHMARK

In the heart of Benchmark plug-in lies a problem which needs to be solved. The key element of any benchmark is therefore a good explanation of the problem so engineers have everything in place before engaging in problem solving.

Benchmark plug-in offers nice and intuitive interface for engineers and for team leaders.

Figure 3, for example, shows a part of team leaders user interface which features tabs located at the top for easy switching between screens designed for certain tasks. Currently opened tab “General” includes options regarding problem description, supported programming languages and preloaded files upload.
Problem description is written in rich-text format inside build-in FCKeditor that is very powerful open-source HTML/XML editor available on-line that enables team leaders to write rich explanation including pictures and/or animation. Additionally, this user interface enables easy upload of files that can serve as part of problem description as well.

Second part of benchmark creation process is People manager. It enables team leaders to easily invite, ban, accept or decline request of future engineers that work on their problem.

3.1. Statistics tab

Statistics tab is the heart of Benchmark plug-in. It contains interface for examining, testing and comparing present work of all engineers engaged in problem solving together with interface for comments and marking.

During engineers work Benchmark stores every bit of data into database and all of that data needs to be visible and accessible trough statistics tab. Stored data includes: time and date when user was on-line, all outputted data by any program ran during development together with compile time, execution time and memory usage. Also, program automatically stores old versions of source code into database so it can be executed any time latter for comparing present and past versions.

There are few options available to project manager designed for overviewing and comparing engineers work. In the middle of everything lies a table that is highly adjustable by adding or removing rows or columns. Rows actually present people and columns present different data you are interested in. Available columns include:

- **Version date** - date and time of last updated version
- **Outputted data** – text and pictures outputted by program
- **Files** - list of all files contained inside the project
- **Execution time** – execution time of last source build
- **Max. memory** – maximal memory usage of the program
- **Sharing** – share data with other users
- **Comments and grading** – interface for commenting and grading
- **Test cases** – interface for testing program to a set of input cases.
- And few more!
3.2. Problem solving and uploading

Scriptrunner and Benchmark plug-in give many possibilities to do your work on-line including source code editors, various compilers, graphical and numerical data presentation but no on-line javascript + AJAX interface can provide the same feel and options available in Windows or Unix graphical interface development studios like MSDEV, Eclipse or Matlab editor. Having that in mind, Benchmark was created upload oriented which means that engineers are supposed to do their work off-line and then just upload source code and compile it on-line. Uploading compiled executables is not allowed for security reasons and because it wouldn’t fit into the idea of Benchmark that was build on.

Ideally, engineers should upload their solutions in standard form so the source code can be compiled and executed automatically. Standard form assumes:

- One zip, tar or similar archive.
- All source code written in the same programming language.
- Script/program source file named RUNME.extension for automatic execution.
- Optional authors comment and manuals written in Tex or PDF.

Besides upload module, Engineers can access certain statistics of their and shared programs trough interface similar to one shown in picture 3 but with less features.

4. CONCLUSIONS

Our task was to create a system where people can share their knowledge and work together on the same problem sharing their creativity to produce desired end result – a solved problem. What we’ve created saves a lot of time; keeps people in touch and most importantly keeps all the data in one place. Scriptrunner is a powerful system and by creating this plug-in we’ve given it a new purpose and proved it’s worth taking a look ☺.

5. REFERENCES


