HOW TO USE VIMARSHA

Performance Analysis Data Collection and Visualization Tool

2013

www.vimarsha.org
Department of Computer Science & Engineering, University of Moratuwa

1 DATA COLLECTION TOOL

Performance event count collection is an essential step in our proposed approach to performance analysis. First we should collect data for mini programs in the target architecture. Then we should collect performance event counts for testing program or programs. There are three types of data collection methods for test programs. First one is whole program data collection in which the performance counts correspond to accumulated performance event counts. Second one is function level performance count collection. Final one is time sliced data collection of long running programs. To facilitate all these type of data collection we have packaged three CLI tools.

Our main target in developing these tools was to make the minimally depend on the external tools. Therefore we created them as CLI tools to avoid all dependencies we get if develop a graphical tool. CLI is also make it easy to work with remote machines connected using SSH. Main dependency of the data collection tool package is Linux Perf.

Following sections describe the data collection tools in detail.

1.1 Installation

The installation process of the data collection is designed to be simple so that even a user without root permission to the machine can install and use the tools. Users of the tool required to have Linux Perf installed in the system. To install the tool users have to download a copy of tool package to testing system and add path to bin directory in their path environment. To automate this task we have included a file called “env.sh” which can be sourced to setup the tool.

```
# Go to the root directory of the package and issue this command
source env.sh
```

Figure 1 - Installing data collection tools

1.2 Test package data collection tool (testpkgctrl)

This tool provide an interface to control the data collection process of mini programs. Both the parallel and sequential mini programs are accessible through the tool.

1.2.1 Usage

Figure 2 shows the basic usage of the tool. Here the essential attribute is the package name. Users can provide options to list the packages available in the tool.

```
testpkgctrl -p PACKAGE [OPTIONS]...
```

Figure 2 - testpkgctrl tool usage

1.2.2 Options

- **-p PACKAGE** - This option can be used to provide a list of packages or aliases on which the action is to be performed. Example package names are “all”, “parallel”, and “swap”.
- **-l** - This option is used to list all programs that are accessible through the tool.
- **-d** - This shows information for a given program or alias through the option “-p”. Users can get details about supported input types and program variations from this options.
- **-n THREADS** – This is used to instruct the tool about the list of different thread numbers used to collect data.
- **-e EVENTS** – When we omit “-c” (CPU architecture) option, this is used to provide a custom list of events.
- **-a ACTION** – This specify the data collection type. Available types are “default” and “functional”.
- **-i INPUTS** – This provide the list of inputs to be used with mini programs in data collection.
- **-o OUTPUT** – This option is used to give the output file a different name. The default output file name is “perf.out”.
- **-c CPU_ARCHITECTURE** – Instead of providing a custom event list, this option can be selects a pre-defined set of events for a given CPU architecture. Available architectures are listed in the help page.
- **--perf-binary PERF_LOCATION** – If Linux Perf is installed in a difference location or not include in “PATH”, this option can be used to instruct Perf about the correct location.
- **-h** - Displays help message for the testpkgctrl tool.

1.3 Normal program data collection tools

Following two tools can be used to collect performance counts for external programs or command. First tool is used to collect data for a single run of a program. Second tool is used to collects samples of performance event counts from different time intervals.

1.3.1 vcollect

vcollect CLI tool collects performance counter values for a given command using Linux Perf.

1.3.1.1 Usage

Figure 3 show the basic usage of the tool. Here the essential attribute is the command. Other behavior can be controlled through the available options.

```
vcollect [OPTIONS]… command
```

Figure 3 - vcollect tool usage

1.3.1.2 Options

- -c CPU_ARCHITECTURE – Instead of providing a custom event list, this option can be selects a pre-defined set of events for a given CPU architecture. Available architectures are listed in the help page.
- -e EVENTS – When we omit "-c" (CPU architecture) option, this is used to provide a custom list of events.
- -r REPEAT – Number of repetitions for the given program in collecting data. The final counts returned are the average values of all runs.
- -a ACTION – This specify the data collection type. Available types are "default" and "functional".
- -o OUTPUT – This option is used to give the output file a different name. The default output file name is "perf.out".
- --perf-binary PERF_LOCATION – If Linux Perf is installed in a difference location or not include in "PATH", this option can be used to instruct Perf about the correct location.
- -h - Displays help message for the testpkgctrl tool.

1.3.2 vtimecollect

This tool collects performance counter values periodically for a given command using Linux Perf. Data collection stops when the test program is killed or the specified number of samples collected.

Figure 4 show the basic usage of the tool. Here the essential attribute is the PID. Other behavior can be controlled through the available options.

```
vtimecollect [OPTIONS]… PID
```

Figure 4 - vtimecollect tool usage

1.3.2.1 Options

- **-c CPU_ARCHITECTURE** – Instead of providing a custom event list, this option can be selects a pre-defined set of events for a given CPU architecture. Available architectures are listed in the help page.
- **-e EVENTS** – When we omit "-c" (CPU architecture) option, this is used to provide a custom list of events.
- **-o OUTPUT** – This option is used to give the output file a different name. The default output file name is "perf.out".
- **-t TIME** – This specify the length of time interval for one sample.
- **-n INSTANCES** – Using this option we can specify the required number of samples. If target program is killed before collecting this number of samples, the output contains only the collected samples.
- **--perf-binary PERF_LOCATION** – If Linux Perf is installed in a difference location or not include in "PATH", this option can be used to instruct Perf about the correct location.
- **-h** - Displays help message for the testpkgctrl tool

2 DATA VISUALIZATION TOOL

The data visualization tool provides a graphical user interface to comprehensively analyze the performance event data which is collected by the data collection tool. The tool is packaged with all its external dependencies such that it can be run out of the box.

Using the graphical interface, users can perform various tasks such as analysis of the distribution of data collected, obtain classification results for a whole program, analysis of a program by function wise classification and analysis of a program by time sliced classification.

This tool also provides other functionality such as letting the user input an already formatted arff file or if not inputting a raw file containing performance event data and then converting into the ARFF format (which is recognized by Weka and other machine learning tools) and also allowing the user to save output file at a preferred location.

The said functionality with each interface is described in detail below.

2.1 Installation

The tool can be run as with any Java jar file with the following command from the tool's root directory.

```
java –jar vimarsha1.0.jar
```

2.2 Data tab

The data tab allows users to input a performance event data file in the ARFF format or input a raw file and then convert it into ARFF format. Once that is done the user can then proceed to select the platform for which the data was collected from the drop down menu. At present both the Intel Nehalem platform as well as the IMB Power 7 platform events are offered in for classifications. Special reduced event sets are offered for time sliced classification of programs.

The attributes in the respective event list is displayed in the attribute pane. Selecting one of the attributes will give statistical information such as the minimum and maximum values, mean and variance of the data for that attribute. Also a chart with the attribute data binned is displayed in this tab.

Users can also save the data file which was converted to the ARFF format at a selected location using this interface by clicking on the “Save to ARFF file” button.

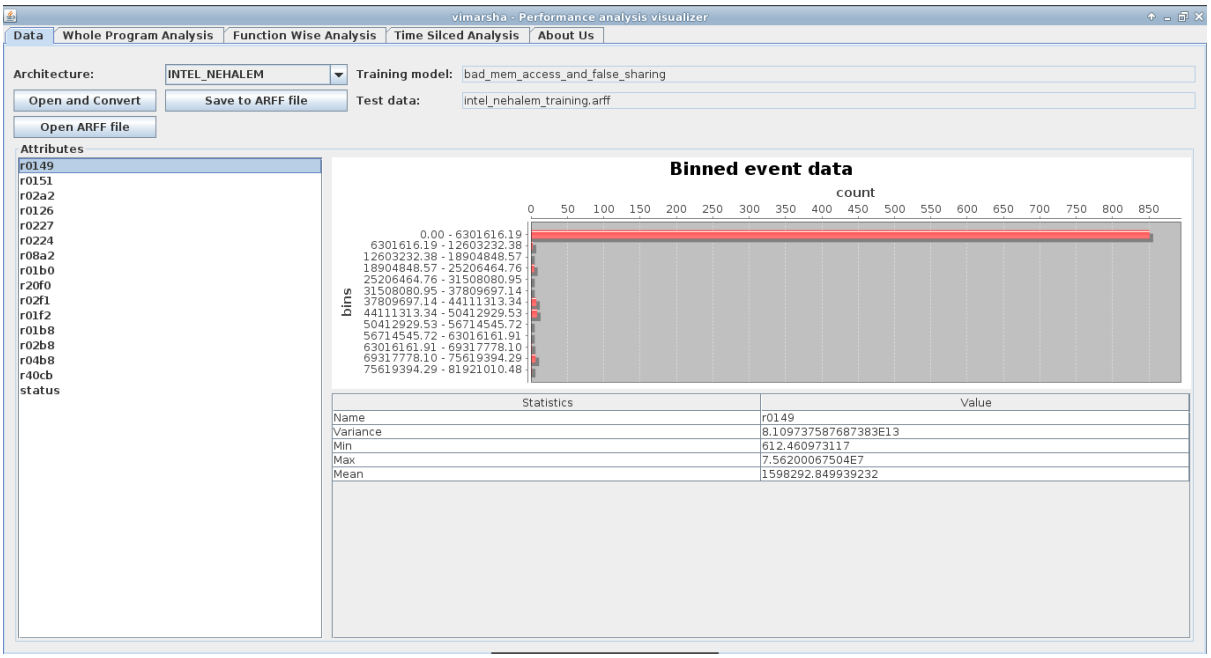

Figure 21: Data tab – Data visualization

2.3 Whole program Analysis tab

After inputting the performance event data and selecting the platform for which the data was collected through the data tab the whole program analysis tab can be used to then classify the data using the trained model contained in the tool for that particular platform. The classification result pane will display the tree model for the classification.

All classification made within the session are saved by time stamps and can be accessed from the Test Data Set pane.

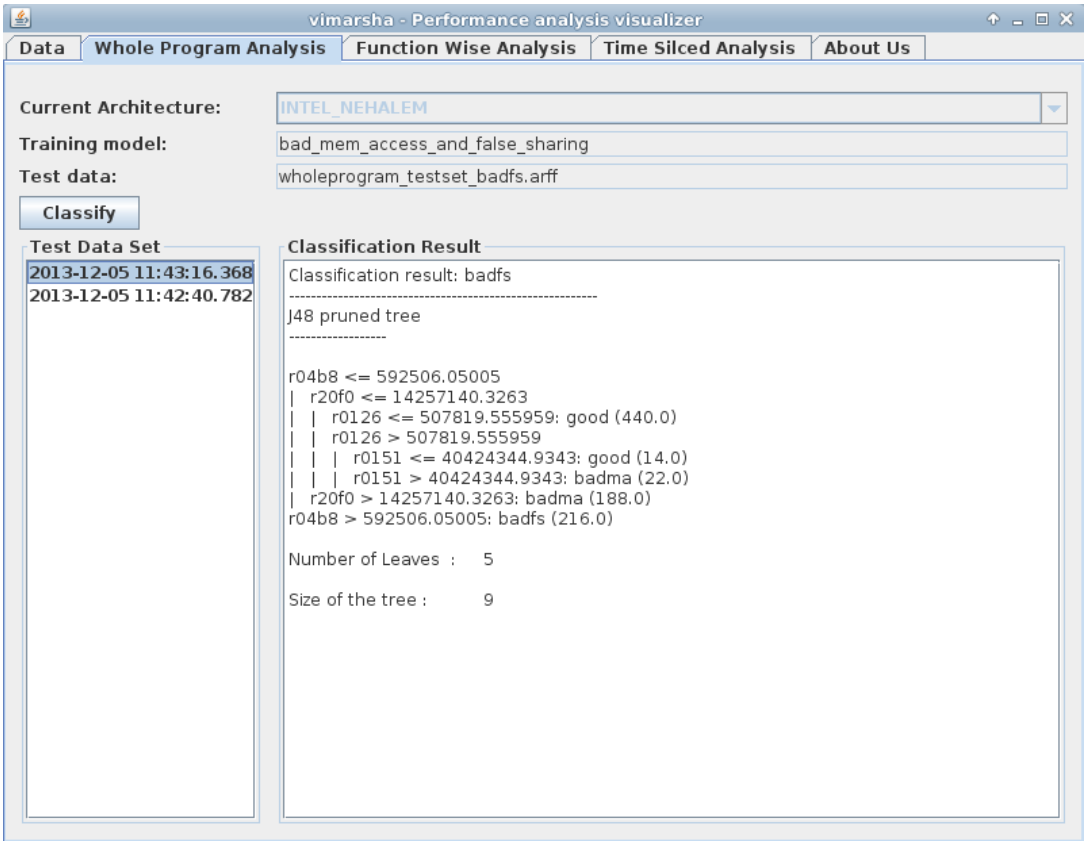

Figure 22: Whole program analysis tab – Data visualization

2.4 Function-wise analysis tab

After inputting the performance event data collected for functions of a program by the data collection tool and selecting the platform for which the data was collected through the data tab the function-wise program analysis tab can be used to then classify the data using the trained model contained in the tool for that particular platform. The function-wise result pane will display a table with the name of each function and the corresponding classification of that function as ‘badma’, ‘badfs’, ‘good’ which corresponds to having inefficient memory access, having false sharing and not having the two memory issues. The export results button will convert the file into a CSV file format and save it at a user’s preferred location.

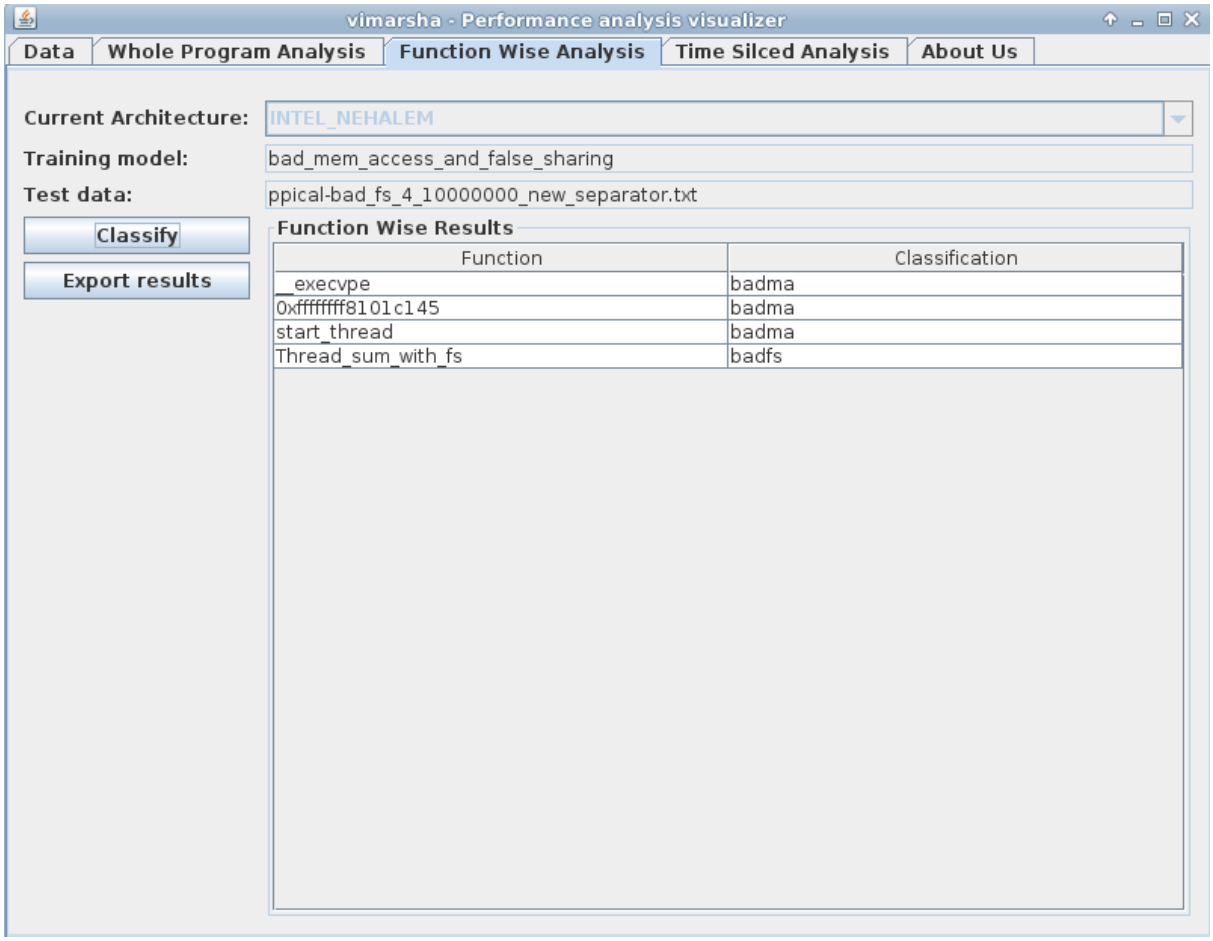

Figure 23: Function-wise analysis tab – Data visualization

2.5 Time sliced analysis tab

After inputting the performance event data collected by time slices for a program by the data collection tool and selecting the platform for which the data was collected through the data tab the time-sliced analysis tab can be used to then classify the data using the trained model contained in the tool for that particular platform. The time-sliced graph will display the classification as ‘badma’, ‘badfs’, ‘good’ which corresponds to having inefficient memory access, having false sharing and not having the two memory issues with the progress of time. The export result button will convert the file into a CSV file format and save it at a user’s preferred location.

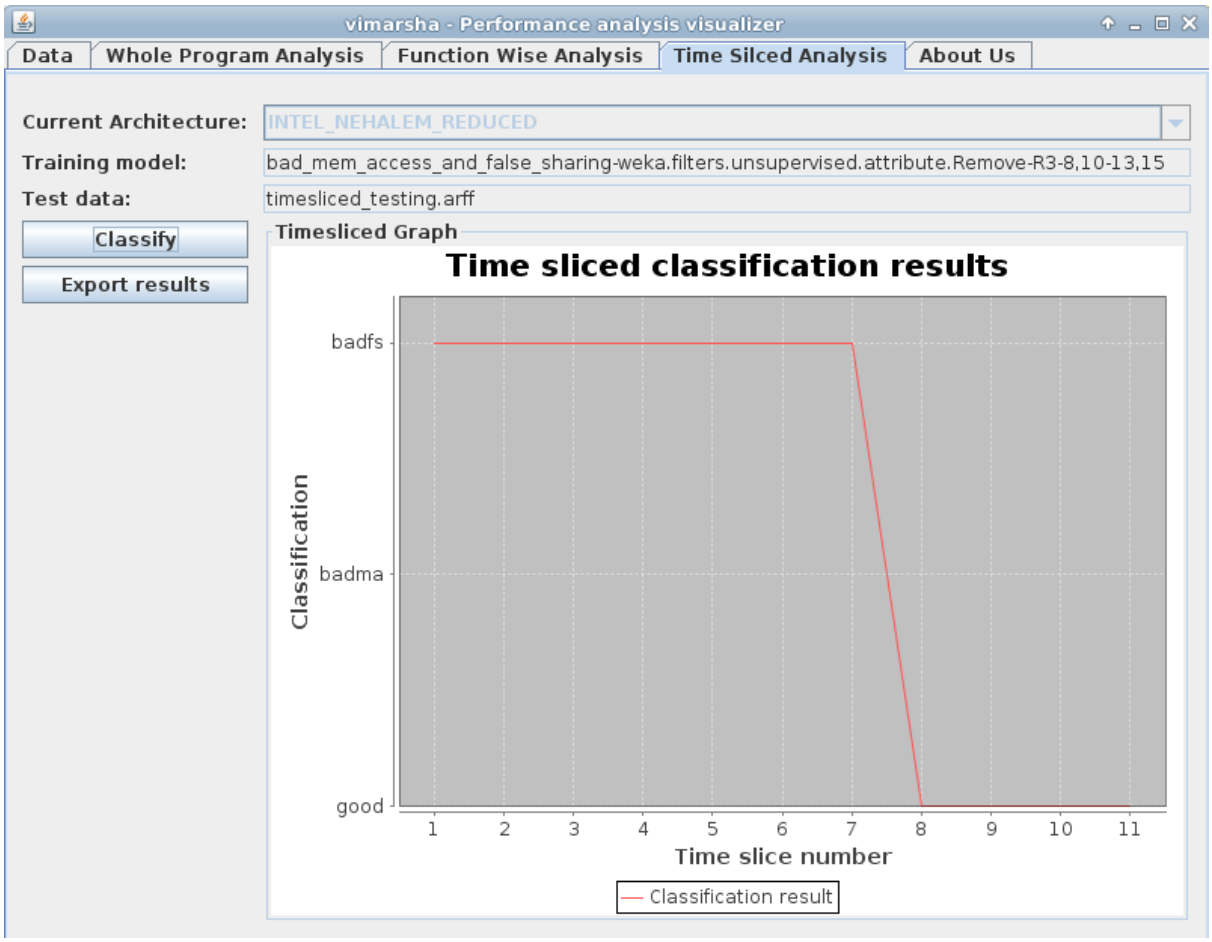

Figure 23: Time-sliced analysis tab – Data visualization