Setting up Sigil

Contents

1. Supported Frameworks

Table 1shows supported frameworks for sigil.

Table 1 supported framework

Distribution	Machine	valgrind
CentOS 5,6	Dragon	3.7
Ubuntu	Ubuntu 14.04	vmware

2. Prerequisite packages for Sigil

Table 2. prerequisite packages for sigil

Dependencies		
Package name	Ubuntu	CentOS
Git	sudo apt-get install git	yum install git
Auto-apt	sudo apt-get install auto-apt	yum install auto-apt
Automake	apt-get install automake;	yum install automake
Autoconf	apt-get install autoconf	yum install autoconf
python-dev	apt-get install python-dev	yum install python-dev

2.1. To automatically prepare your system

sigil_pre_setup.sh is a bash file provided for Ubuntu and CentOS operating systems. It is designed to only update your system if prerequisite packages haven't been installed. The bash script needs sudo permissions, so it is advised to check the contents of the script. PLEASE USE AT YOUR OWN RISK. Table 2 shows sigil prerequisite packages. Note that sigil_pre_setup.sh is provided in sigil scripts folder.

- ./sigil_pre_setup.sh

2.2. To manually prepare your system

- Use table 2 and configure your system.

3. Setting up Sigil

3.1 Setting up Sigil toolchain

3.1.1 to automatically setup Sigil Toolchain

To automatically setup Sigil toolchain

- ./sigil_setup.sh

The script will setup the sigil toolchain and post processing process. It has been explained in Sigil script documentation can be found in tutorial folder. Sigil_setup.sh will ask the user to provide the script with a path to download sigil to. The scripts are explained in sigil_script_document.

3.1.2 To manually setup sigil toolchain

- Clone sigil from https://github.com/snilakan/Sigil
- Run "./autogen.sh" to setup the environment (you need the standard autotools))
- Run "./configure "
- Run "make"
- Running a quick test

Where is the make check? Does make regtest do the make check?

To build and run all the Valgrind regression tests (including Sigil tests), run "make [--quiet] regtest". To run a subset of the regression tests, execute:

- perl tests/vg_regtest <name>

where <name> is a directory (all tests within will be run) or a single .vgtest test file, or the name of a program which has a like-named .vgtest file. Eg:

- perl tests/vg_regtest callgrind
- perl tests/vg_regtest callgrind/tests/sigil1.test
- perl tests/vg_regtest callgrind/tests/sigil2
- Running the performance tests

You can find more detailed information about these steps in Valgrind's README_USERS file that is included in Sigil folder.

3.2 Setting up Post Processing Running Sigil for the HW/SW partitioning flow

Sigil's data can be analyzed with post-processing for the Task Graph Partitioning problem. Sigil enables the first step in selecting which accelerators to include when performing HW/SW co-design.

3.2.1 To automatically setup post processing

- ./sigil_setup will automatically set this up; No extra step needed. This step has been explained in 3.1.1.

3.2.2. To manually setup the post processing

- The post processing python script resides in <EXTRACTDIRECTORY>/Sigil/post_processing/aggregate_costs_gran.py" . To use the post processing, the aggregate_cost.py must be edited. However, the steps are explained in this section. For more information you can refer to the README file located in “<EXTRACT DIRECTORY>/Sigil/postprocessing/” folder.

Post processing can be done in 3 steps:

I. Find the two lines in the post-processing script that say: 'cg_anno = "perl /…/Profilers/valgrind-3.7_original/callgrind/callgrind_annotate --inclusive=yes --threshold=100 " + callgrind_filename'
II. Edit the lines to replace "/…/Profilers/valgrind-3.7_original/callgrind/callgrind_annotate" with the full path to the callgrind_annotate perl script in the Callgrind folder.

1. Note that “ < >” is not part of command line.
2. Threshold is a constant; it should be equal or greater than 100.