

The MODV Manual

1 Introduction

MODV is a Memory Order Dynamic Verifier which is used to verify whether an execution complies with a certain memory model. MODV generates a pseudo-random multi-processor test program without conditional branches first. Then the user runs the test program on the system that implements a certain memory consistency model and gathers scan information about the test program. As this module is architecture dependent, MODV doesn't provide it. At last, the test program and the gathered information will be input in MODV to verify if this program complies with the model. The types of the memory models that can be verified by MODV now are Sequential Consistency (SC), Processor Consistency/x86/TSO (PC), Release Consistency (RC) and Scope Consistency (ScC). In this version, ScC refers to Godson-T which is a variation of ScC. MODV has only been tested in Linux RHEL 5.4 64bit, and this version is special for verifying the memory model of Godson-T. The next version of MODV will be developed soon.

2 Installation

MODV is an open source software. You can install MODV by the following steps and you can make any change if you need on the source code.

Step 1 Download the zipped source file or executable file of MODV. The latest versions are MODV-0.2.2-src.tgz and MODV-0.2.2-bin-x86_64.tgz respectively. If you choose the executable file, step 2 can be omitted.

Step 2 If you download the source file, save the file to one directory and decompress the archive. Assume the directory where you put the archive is /home/workspace.

```
$ cd /home/workspace
$ tar -xvzf MODV-0.2.2-src.tgz
$ cd modv/src
$ make
$ make install
```

After decompressing, there are 3 folders in MODV. The folder bin contains the executable file. The folder src contains the source files. The folder test contains some examples.

Step 3 To run MODV, the test program files and the information files for test program should be put in the same directory with the executable file modv. Then you can simply run MODV with the command shown below which is used to verify whether an execution complies with Processor Consistency.

```
$ ./modv -p 4 -n 500 -m PC
```

3 Usage

The bash shell script generator can be used to generate pseudo-random multi-processor test programs with the command ./generator. There are two options for this command: -n is used to denote the number of processors and -p is used to denote the number of operations in each processor.

The bash shell script modv can be used to verify the test programs. There are several options for this command shown below:

-h, --help	Output the help information, which is a simplified version of manual.
-p NUM, --processor NUM	Denote the number of processors.
-n NUM, --nolist NUM	Denote the number of operation in each processor.
-m NAME, --model NAME	Denote the name of the memory model to be verified. This version of MODV can verify SC, PC, RC and ScC.
-b, --baseline	If -b option is used, then MODV will use the baseline algorithm which doesn't use the time order information.
-i DIR, --input DIR	Denote the directory of the test program files and the scan information files.

4 Notes

There are several notes about MODV when using it to verify memory models which are shown below:

1) The default processor number is 2. The default operation number is 500. The default memory model to be verified is ScC, i.e. Scope Consistency (This version is Godson-T, a variant of Scope Consistency). Thus if one of

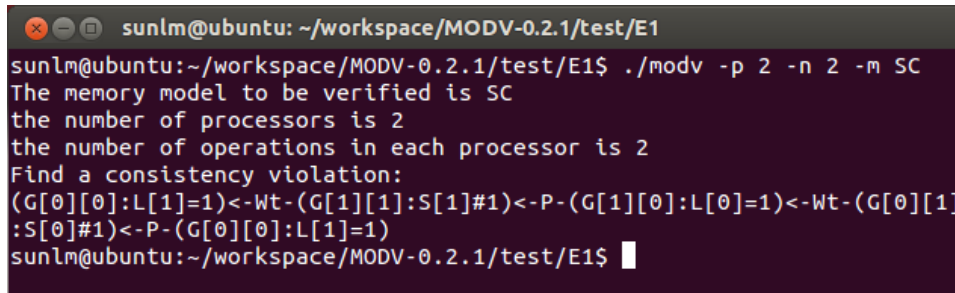
the three arguments is not the default value, the argument can't be omitted, otherwise errors will be reported.

2) The result files which record the program execution and scan information should be put in the same directory with modv, just like the examples shown in the test directory. Otherwise you should use -b option to change the directory.

3) The way to name and log the results should also be the same as the files of examples stored in the test directory.

5 Examples

We use MODV to verify whether an execution complies with memory consistency model. We use the command `./modv -p 2 -n 2 -m SC` to verify whether the execution complies with Sequential Consistency.



```
sunlm@ubuntu: ~/workspace/MODV-0.2.1/test/E1
sunlm@ubuntu:~/workspace/MODV-0.2.1/test/E1$ ./modv -p 2 -n 2 -m SC
The memory model to be verified is SC
the number of processors is 2
the number of operations in each processor is 2
Find a consistency violation:
(G[0][0]:L[1]=1)<-Wt-(G[1][1]:S[1]#1)<-P-(G[1][0]:L[0]=1)<-Wt-(G[0][1]:S[0]#1)<-P-(G[0][0]:L[1]=1)
sunlm@ubuntu:~/workspace/MODV-0.2.1/test/E1$
```

A cycle is detected. Then we can draw the conclusion that this execution doesn't comply with Sequential Consistency.