PIKE Improving COTSon Interface for Easier Design Space Exploration

Andrea Mondelli, Kang Cai,
Roberto Giorgi[1]

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, University of Siena, Via Roma 56, 53100 Siena, Italy

ABSTRACT

Many steps are necessary to setup a COTSon simulation, as it requires the knowledge of many details that slow down the learning curve of this powerful simulator. Therefore, we decided to improve the knowledge transfer by developing the PIKE tool. PIKE is a wrapper for the COTSon simulator. It can automate the simulation configuration and execution generating all configuration files and scripts suitable for benchmark execution, automate the installation and simulations of a multicore-multinode CPU model. PIKE not only saves the time but also improves the efficiency of work, and especially for the first time users to make the design easier.

KEYWORDS: PIKE; COTSon; simulator;

1 Introduction

COTSon is a full system simulation infrastructure developed by HP Labs to model complete computing systems ranging from multicore nodes up to clusters with complete network simulation [EA09]. A single simulation requires the configuration of various parameters by editing a configuration file (written in the Lua language); further configuration of some scripts is recommended to allow more control of simulated events, for example to set any specific option (e.g. MPI) or specify features such as the definition of a region of interest, or even output of the simulation in a file stored in the host machine. In addition, this work should be done for each parameter of the benchmark used. Therefore, using COTSon needs a lot of manual work to modify the configuration file for the whole simulation procedures, not only consume a lot of time but also cause errors and repeat working. So, to improve COTSon interface for easier design space exploration and make the simulation higher efficiency, we developed PIKE [pik12] to automatically configure COTSon for different application.

The design goal of PIKE is to automate the simulation configuration and execution generating all configuration files and scripts suitable for benchmark execution. In addition, it allows the user to use all available host cores, and enables simulation in batch mode by means of a thread pool mechanism created according to the characteristics of the host machine.

[1]E-mail: {mondelli,kang,giorgi}@dii.unisi.it

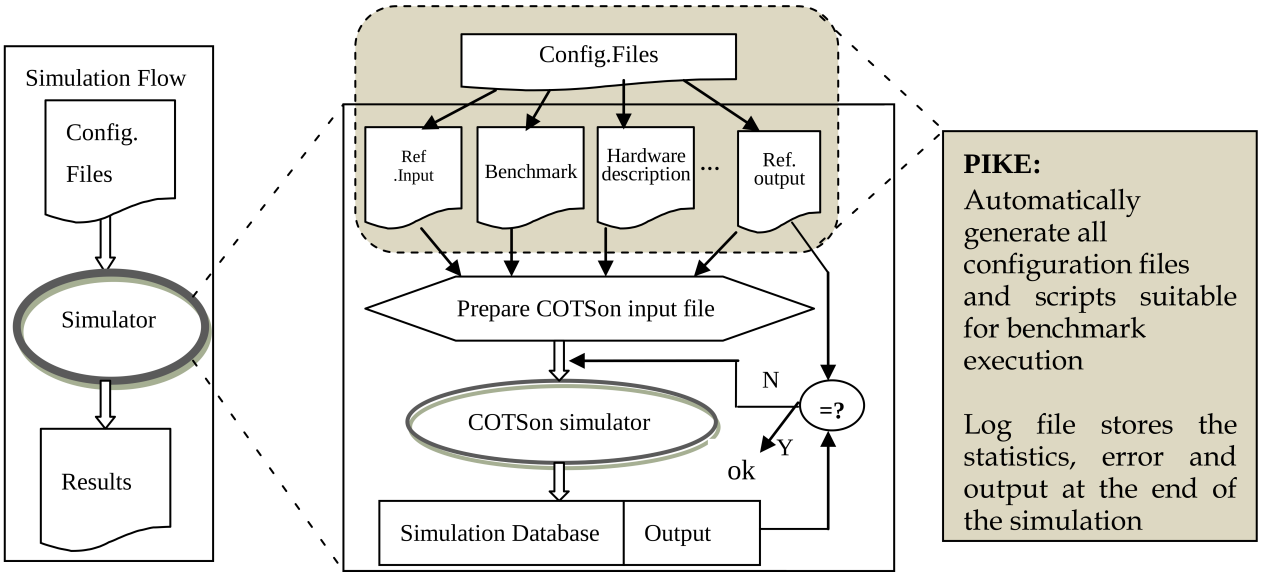

Figure 1: PIKE work flow

2 Work Flow of PIKE

PIKE requires ruby (version >= 1.9). It uses a single configuration file to set the parameters of the simulation. Figure 1 shows the work flow of PIKE. The configuration file is used to set:

- The list of simulations to run
- Software configuration like communication type, input file name and region of interest
- Hardware properties like cache configuration, timing model, node number and core number

Through the single configuration file, PIKE produces simulation output and statistics inside a specified folder named WorkingDirectory, which is the system path for benchmarks binaries, , log file, COTSon and SimNow [SN09] path. With different configuration parameters in one configuration file, PIKE can create the simulation threads in parallel. The directory has the structure shown in Table 1.

PIKE can be configured to run in two different modes: *silent* and *verbose*. Silent mode shows only the result without trace, verbose mode is a debug mode in which every single operation performed by PIKE is traced.

3 Functions Exposed to the User

PIKE currently allows the user to automate the execution of batch simulations for design space exploration. It allows specifying custom parameters in order to explore different hardware configurations for the target system, together with control parameters eventuallyneeded by the benchmarks. Such parameters can be specified in the PIKE configuration file. The list of main sections of the configuration file is reported below.

system Allows to specify a custom path for PIKE. Appropriate links to any SimNow ISO images(the virtual hard-disk of SimNow Node) will be automatically created in the COTSon data directory

Table 1: PIKE structure

`bin`	pike binary , like env_ and bsd_creator
`lib/pike`	pike library, used by binary
`tools`	supports like conf example, skeleton creator script, bash script binary
`pike.conf`	pike configuration base file

log Allows specifying the output directory of the log produced by the simulation, together with the names for the output files if needed. If such names are not customized, PIKE creates log files using an alphanumeric code as simulation's identifier

file Characteristics of the simulation as the BIOS-ROM file (if present) and custom Hard-Disk image file (if any)

hardware Guest hardware configuration to be used in the benchmark, i.e. the number of nodes, number of cores, and the size of the ram

software Software packages to be installed on the guest before running the simulation. Supports both deb and rpm based packages

support Simulation support files

simulator Binaries to run and parameters. For each entry a different simulation will be launched. Each run will be identified by a different alphanumeric code

options To enable or disable mpi support

cache Cache parameters and configuration

mediator Mediator configuration inside the simulation

4 Example and Conclusions

In this section we show an example of PIKE configuration file to test a single node of multip-benchmarks with different cache size. The configuration file has the following parameters:

- Config.image_name = debian6.img
- Number of cpu = 8
- Benchmark = [CJPEG, simulator_binary1]
- Cache = [1M,256K]

Then PIKE will generate the configuration files for COTSon simulation files, then according the different parameters like different cache size, it will create simulation files to run the simulations of different benchmark in parallel. Figure 2 shows the SimNow console for the simulations.

In this work, we have presented a brief description of PIKE, a wrapper for COTSon simulator. We use PIKE in TERAFLUX project [ea13] [Gio12] [Ter14] for both knowledge transfer purpose and design space exploration [AP12], it improved work efficiency. What is more, it has a significant meaning as an education tool for people to use COTSon quickly. Future work will focus on how to make PIKE more versatile.

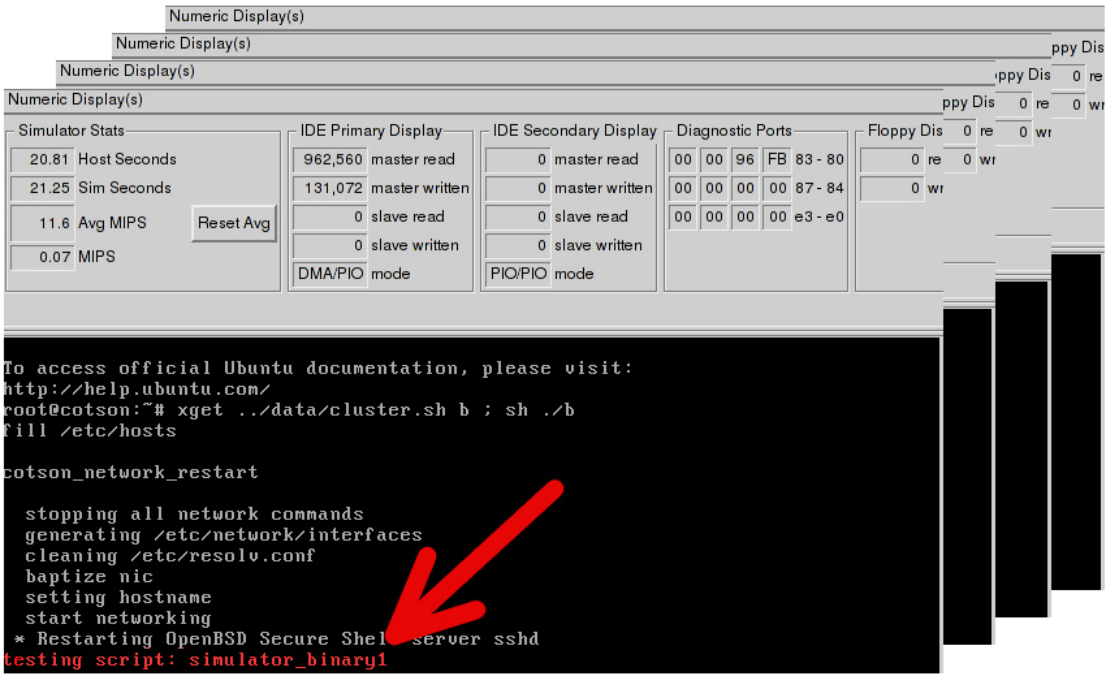

Figure 2: SimNow instance with 2 benchmarks with two different cache size(4 windows)

Acknowledgment

This work was partly funded by HiPEAC id. 287759 and the European FP7 project TERAFLUX id. 249013

References

[AP12] Z. Yu P. Faraboschi C. Concatto L. Carro A. Garbade S. Weis T. Ungerer and R. Giorgi A. Portero, A. Scionti. Simulating the future kilo-x86-64 core processors and their infrastructure. In *Proceedings of the 45th Annual Simulation Symposium*, ANSS '12, pages 9:1–9:7, San Diego, CA, USA, 2012. Society for Computer Simulation International.

[EA09] P. Faraboschi M. Monchiero D. Ortega E. Argollo, A. Falcón. Cotson: infrastructure for full system simulation. *SIGOPS Oper. Syst. Rev.*, 43(1):52–61, 2009.

[ea13] M. Solinas et al. The TERAFLUX project: Exploiting the dataflow paradigm in next generation teradevices. 2013.

[Gio12] R. Giorgi. TERAFLUX: exploiting dataflow parallelism in teradevices. In *Proceedings of the 9th conference on Computing Frontiers*, CF '12, pages 303–304, New York, NY, USA, 2012. ACM.

[pik12] Pike4cotson. `http://sourceforge.net/projects/pike4cotson/`, 2012.

[SN09] Amd simnow simulator 4.6.1 user's manual, 2009.

[Ter14] Exploiting dataflow parallelism in teradevice computing. `http://www.teraflux.eu`, 2010-2014.