SimPoint+ Artifacts Document

1. Getting Started Guide

Artifact Link： https://drive.google.com/file/d/1Ofi10CuB5h9tIbOR7H_2Nv2mEEQZ8igd/view?usp=drive_link

md5 hash: e08b3eb9a163ab11ba6064c37c4ed19a

To facilitate user environment configuration, we have packaged the necessary tools and code into a singularity image, which includes five main directories, as shown in Figure 1.

1. GEM5: high precision simulator, for collecting interval cycles.
2. NEMU: high performance simulator, for profiling basic block vectors.
3. cycle-estimate: The methodology implemented by SimPoint+.
4. riscv: build from source, include some executable utilities, such as riscv64-unknown-elf-gcc,riscv64-unknown-linux-gnu-gcc,.etc. It's subdir bin has been appended to $PATH.
5. workloads: There are some workloads build from SPEC 2006 as shown in Figure 2. For how to build workloads that NEMU can run, please checkout: https://docs.xiangshan.cc/zh-cn/latest/workloads/linux-kernel-for-xs-en/

```
→ ~/xiangshan ls
GEM5  NEMU  cycle-estimate  riscv  workloads
```

Figure 1

```
→ ~/xiangshan/workloads/bbls pwd
/root/xiangshan/workloads/bbls
→ ~/xiangshan/workloads/bbls ls
GemsFDTD-bbl-linux-spec.bin
astar_biglakes-bbl-linux-spec.bin
astar_rivers-bbl-linux-spec.bin
bwaves-bbl-linux-spec.bin
bzip2_chicken-bbl-linux-spec.bin
bzip2_combined-bbl-linux-spec.bin
bzip2_html-bbl-linux-spec.bin
bzip2_liberty-bbl-linux-spec.bin
bzip2_program-bbl-linux-spec.bin
bzip2_source-bbl-linux-spec.bin
cactusADM-bbl-linux-spec.bin
calculix-bbl-linux-spec.bin
dealII-bbl-linux-spec.bin
gamess_cytosine-bbl-linux-spec.bin
gamess_gradient-bbl-linux-spec.bin
gamess_triazolium-bbl-linux-spec.bin
gcc_166-bbl-linux-spec.bin
gcc_200-bbl-linux-spec.bin
gcc_cpdecl-bbl-linux-spec.bin
gcc_expr-bbl-linux-spec.bin
gcc_expr2-bbl-linux-spec.bin
gcc_g23-bbl-linux-spec.bin
gcc_s04-bbl-linux-spec.bin
gcc_scilab-bbl-linux-spec.bin
gcc_typeck-bbl-linux-spec.bin
gobmk_13x13-bbl-linux-spec.bin
gobmk_nngs-bbl-linux-spec.bin
gobmk_score2-bbl-linux-spec.bin
gobmk_trevorc-bbl-linux-spec.bin
gobmk_trevord-bbl-linux-spec.bin
gromacs-bbl-linux-spec.bin
h264ref_foreman.baseline-bbl-linux-spec.bin
h264ref_foreman.main-bbl-linux-spec.bin
h264ref_sss-bbl-linux-spec.bin
hmmer_nph3-bbl-linux-spec.bin
hmmer_retro-bbl-linux-spec.bin
lbm-bbl-linux-spec.bin
leslie3d-bbl-linux-spec.bin
libquantum-bbl-linux-spec.bin
mcf-bbl-linux-spec.bin
milc-bbl-linux-spec.bin
namd-bbl-linux-spec.bin
omnetpp-bbl-linux-spec.bin
perlbench_checkspam-bbl-linux-spec.bin
perlbench_diffmail-bbl-linux-spec.bin
perlbench_splitmail-bbl-linux-spec.bin
povray-bbl-linux-spec.bin
sjeng-bbl-linux-spec.bin
soplex_pds-50-bbl-linux-spec.bin
soplex_ref-bbl-linux-spec.bin
sphinx3-bbl-linux-spec.bin
tonto-bbl-linux-spec.bin
wrf-bbl-linux-spec.bin
xalancbmk-bbl-linux-spec.bin
zeusmp-bbl-linux-spec.bin
```

Figure 2

The installation steps are as follows:

代码块

```
# Unzip the sandbox
tar -xzf zebra-xiangshan.tar.gz
# Entering the container
singularity shell --writable zebra-xiangshan
```

2. How to use

We divide the entire process into two parts: data collection and using SimPoint+. The data collection part requires a significant amount of time. To facilitate validation, we provide the basic data for the `bzip2_chicken` program from SPEC 2006 in the `cycle-estimate/data` directory.

3. Collect baseline data (in Singularity)

3.1 Use NEMU to collect bbv

The time required for data collection varies based on the benchmark size, with BBV collection taking from a few minutes to several hours. To simplify the use of the main SimPoint+ code, we provide the BBV data collected from `bzip2_chicken` using NEMU and the cycle data collected using gem5.

The specific steps are as follows:

代码块

```
# 1. Enter NEMU_HOME
export NEMU_HOME='/opt/xiangshan/NEMU'
export PATH=/opt/xiangshan/riscv/bin:$PATH
cd $NEMU_HOME

# 2. This will generate the executable file at: $NEMU/resource/simpoint/simpoint_repo/bin/simpoint
cd resource/simpoint/simpoint_repo
make

# 3. generate gcpt.bin
cd /opt/xiangshan/NEMU/resource/gcpt_restore
make

# 4. Configure NEMU
cd /opt/xiangshan/NEMU
```

```
# Generate executable files for NEMU
make -j

# 5. please carefully read $NEMU_HOME/scripts/zebra/utils.sh.
# Generate BBV using gcc_166 as an example, as shown in Figure 3.
cd $NEMU_HOME/scripts/zebra
# For gcc_166, it may be a few minutes, but for some other, it may cost a few hours.
# The results will be saved in $NEMU_HOME/parallel_result/profiling/gcc_166, as shown in Figure 4.
./utils.sh

```

```
# Usually, I use parallel_<operation> to process a few benchmarks in parallel.

# If you want to process a single workload, you can use the following commands:
# profiling <benchmark>
# cluster <benchmark>
# checkpoint <benchmark>


# parallel_<operation> is used for parallel process
# parallel_profiling
# parallel_cluster
# parallel_checkpoint

# workload_list is used for parallel process
# workload_list is a file that contains the list of workloads to be processed
:<<workload_list_example
perlbench_checkspam
bzip2_chicken
gcc_166
bwaves
gamess_cytosine
milc
gobmk_13x13
hmmer_nph3
workload_list_example

profiling gcc_166
```

Figure 3

```
➜ ~/xiangshan/NEMU/parallel_result/profiling/gcc_166 git:(zebra) ✗ ls
simpoint_bbv.gz
➜ ~/xiangshan/NEMU/parallel_result/profiling/gcc_166 git:(zebra) ✗ gzip -dc simpoint_bbv.gz > bbv.txt
➜ ~/xiangshan/NEMU/parallel_result/profiling/gcc_166 git:(zebra) ✗ ls
bbv.txt  simpoint_bbv.gz
```

Figure 4

3.2 Use SimPoint to get raw cluster result

For every benchmark, this step only takes a few minutes or even less.

SimPoint is a submodule of NEMU used for clustering, as provided by the default SimPoint. To modify the clustering settings, please edit the `$NEMU_HOME/scripts/zebra/utils.sh` file (as shown in Figure 5). This change simply switches the operation from profiling to clustering. For the `gcc_166` example, please follow these steps:

```
# Usually, I use parallel_<operation> to process a few benchmarks in parallel.

# If you want to process a single workload, you can use the following commands:
# profiling <benchmark>
# cluster <benchmark>
# checkpoint <benchmark>


# parallel_<operation> is used for parallel process
# parallel_profiling
# parallel_cluster
# parallel_checkpoint

# workload_list is used for parallel process
# workload_list is a file that contains the list of workloads to be processed
:<<workload_list_example
perlbench_checkspam
bzip2_chicken
gcc_166
bwaves
gamess_cytosine
milc
gobmk_13x13
hmmer_nph3
workload_list_example

#profiling gcc_166
cluster gcc_166
"utils.sh" 121L, 3179B written
```

Figure 5

代码块

```
cd $NEMU_HOME/sctipts/zebra
# The results will be saved in $NEMU_HOME/parallel_result/cluster/gcc_166, as shown in the Figure 6.
./utils.sh
```

```
➜ ~/xiangshan/NEMU/parallel_result/cluster/gcc_166 git:(zebra) ✗ ll
total 600K
-rw-r--r-- 1 root root 2.8K May  9 17:01 centroid.txt
-rw-r--r-- 1 root root  44K May  9 17:01 labels.txt
-rw-r--r-- 1 root root 544K May  9 17:01 reduced-dimension.txt
-rw-r--r-- 1 root root  138 May  9 17:01 simpoints0
-rw-r--r-- 1 root root  234 May  9 17:01 weights0
```

Figure 6

In the `$NEMU_HOME/parallel_result/cluster/gcc_166` directory, there will be five files:

1. `centroid.txt` : Each line contains one vector. The vectors of the cluster centers; there will be K lines for K cluster centers.
2. `labels.txt` ： 2 columns. Each sample is on a separate line. The first column is the cluster center ID, and the second column is the distance to the cluster center.
3. `reduced_dimension.txt` ： Each line contains one vector. Each sample is on a separate line. These are the reduced-dimensional data obtained from the original samples through random projection.
4. `simpoints0` ： 2 columns. The first column is the sample (interval) ID corresponding to the cluster center, and the second column is the cluster center ID.
5. `weights0` : 2 columns. The first column is the weight corresponding to the cluster of that cluster center and the second column is the cluster center ID.

3.3 Use GEM5 to collect cyle

For every benchmark, this step takes a few days and even more, so we provide you with a `cycle.txt`.

代码块

```
cd /opt/xiangshan/GEM5
scons build/RISCV/gem5.opt -j $(nproc)

# Please read~/xiangshan/GEM5/til/zebra/run.sh carefully.
# Collect cycle using gcc_166 as an example, as shown in Figure 7.
# Modify the workload path in single_run and switch to single_run, as shown in Figure 8.
cd /opt/xiangshan/GEM5/util/zebra
./run.sh
# After the simulation ends, several files can be found in ~/xiangshan/GEM5/out/single.
# The interval_cycle.txt file is simply an alias for the cycle.txt file mentioned above.
# The result is shown in Figure 9.
```

```
function single_run() {
        work_dir=${output_dir}/single
        rm -rf $work_dir
        mkdir -p $work_dir

        warmup_inst=$((20 * 10 ** 6))
        max_inst=$((40 * 10 ** 6))

        workload=/root/xiangshan/workloads/bbls/gcc_166-bbl-linux-spec.bin
        run $workload $warmup_inst $max_inst $work_dir >$work_dir/$log_file 2>&1
}
export -f single_run
```

Figure 7

```
# Usually, I use parallel_run to simulate benchmark, and use single_run to debug.

single_run
# parallel_run
```

Figure 8

```
➜  ~/xiangshan/GEM5/output/single git:(zebra) X ls
completed  dramsim3.json  dramsim3.txt  interval_cycle.txt  log.txt  m5out
```

Figure 9

4. Use SimPoint+

代码块

```
# 1. Modify environment variables:
export CYCLE_ESTIMATE_DIR='/opt/xiangshan/cycle-estimate'

cd $CYCLE_ESTIMATE_DIR
pip install -r requirements.txt

# 2. To run the program
cd /opt/xiangshan/cycle-estimate/src
# If you have other benchmark BBV and cycle data, you can replace bzip2_chicken with your program.
python main bzip2_chicken

# 3. To view the results as shown in Figure 10.
The running results will be saved in the {root_dir}/data/{benchmark}/results.xlsx file:
Phase1_Num represents the number of clusters in the first stage clustering.
```

```
Phase1_Err represents the error of the first stage clustering.
Phase2_Num represents the number of clusters in the second stage clustering.
Phase2_Err represents the error of the second stage clustering.
Speedup represents the acceleration ratio between the final result and the full simulation.
Final_err represents the final result of SimPoint+.
```

cycle-estimate > data > bzip2_chicken > results.xlsx

	A	B	C	D	E	F
1	Phase1_Num	Phase1_Err	Phase2_Nun	Phase2_Err	Speedup	Final_Error
2	12	2.25	12	6.52	762.86	0.00

Figure 10