**SMAUG Framework setup and simulation**

Github link to smaug: <https://github.com/harvard-acc/smaug>

Github link to gem5-aladdin: <https://github.com/harvard-acc/gem5-aladdin>

Use docker image provided by the smaug developers to ease out setting up the environment to run smaug. Olympus servers have singularity instead of docker. The following link provides the commands to use dicker image on singularity

<https://tamuengr.atlassian.net/wiki/spaces/helpdesk/pages/1983512577/Using+Singularity+for+Container+Support>

Follow the instruction on the github repo to build the simulation. The smaug docker image creates a folder for LLVM and gem5-aladdin under /workspace for which we wont have root permission even while using singularity. To mitigate this, setup gem5-aladdin on a different folder. Start an instance of the smaug docker image and change the env variable ALADDIN\_HOME to point to gem5-aladdin\_installation\_folder/src/Aladdin and SMAUG\_HOME to point to the location where smaug is installed. Since this step must be done every time the docker image is entered, it is advised to create a bash file that sets up these environment variables and it could be sourced every time the image is entered. Refer to setup.txt file inside the repo. Once the environment variables are set, build the simulator following the build instructions in the smaug repo.

I had trouble with the “which” command, so I used

python2 /usr/bin/scons build/X86/gem5.opt PROTOCOL=MESI\_Two\_Level\_aladdin -j49

to build gem5-aladdin.

Follow the getting started page to build and run your first experiment

<https://harvard-acc.github.io/smaug_docs/build_python_model.html>

The first step is to create ML model files using smaug python api. This page gives basic explanation for creating such a file. The experiments/models folder has model files of many ML models. Of these, MobileDet, MobileNetV2, EDSR and MOSAIC were added additionally. Since we were concerned about ML inference, where batch normalization is usually converged into previous convolution layer and bias, those layers were replaced with matrix additions respectively. The proto files were generated by running the command

“python $SMAUG\_HOME/experiments/models/MobileNetV2/mobile\_net\_v2.py” inside smaug/build/bin folder. The pbtxt file contains a description of each layers in the model and the .pb file contains the weights of each layer.

Follow this link to build tracer binary called “smaug-instrumented” and the smaug binary:

<https://harvard-acc.github.io/smaug_docs/run_model_in_simulation.html>

To run an ML model using SMAUG framework,

1. Model the ML model layer by layer using smaug python APIs.
2. Generate the .pbtxt and .pb files for the model.
   1. Move to smaug/build folder.
   2. Run “python $SMAUG\_HOME/experiments/models/MobileNetV2/mobile\_net\_v2.py” command.
   3. For other models, provide the location of the respective python model file.
   4. The command will generate .pbtxt and .pb files. Copy this to the respective model folder.
3. Build the tracer “smaug-instrumented”.
   1. Move to smaug/
   2. Run “ make clean-trace && make tracer -j49”
   3. It generates the tracer file inside smaug/build/bin
4. Using the tracer, generate the trace of running the ML model, typically named dynamic\_trace\_acc0.gz
   1. Move to the respective model folder
   2. Smaug/build/bin/smaug-instrumented mobilenetV2\_sm\_topo.pbtxt mobilenetV2\_smv\_params.pb –gem5=false –sample-level=very\_high --sample-num=1
5. Build the smaug binary.
   1. Move to smaug/
   2. Make clean-trace && make all -j49
   3. Generates smaug binary inside smaug/build/bin folder
6. Modify the run.sh script to include the respective model files (.pbtxt and .pb files )
7. Modify gem5.cfg file to include the respective trace file and configure the accelerator parameters as required.
8. Modify the Aladdin\_se.py config file in gem5-aladdin/configs/Aladdin/ folder to configure the gem5-aladdin setup.
9. Once the simulation is complete, the stats.txt file inside the outputs folder will have the results.

To run simulation using sbatch, the sbatch\_script.sh script schedules the job using sbatch. It invokes the singularity image and calls trace\_script.sh. Trace\_script can be used to run the trace or call the run\_script or run\_script\_multi bash files which were used for single and multi-core experiments respectively. The multicore experiment simulates a two core system and runs spec-2006 benchmarks on core-1 while running the desired ML model on core-0.

To debug gem5 using gdb, compiled gdb binary can be downloaded from this github repo

<https://github.com/leommxj/prebuilt-multiarch-bin/tree/bin/x86_64_tools/gdb-8.3.1>

execute this command after calling gdb

set libthread-db-search-path 0

The binary application running on one of the gem5 cores can be debugged using gdbserver in a separate terminal.

Folder structure and contents inside smaug folder:

* Build/bin: smaug binary used to run on gem5 and smaug-instrumented tracer binary.
* Experiments/sims/smv/tests/Minerva: example folder containing all contents needed to run Minerva ML model
* Experiments/models/: Contains folders pertaining to different ML models. The scripts and config files to run a simulation and their output files are all stored inside these folders.
* Experiments/models/MobileDet/smv-accel.cfg: configures the accelerator to be 256GOPS
* Experiments/models/MobileDet/smv-accel\_1TOPS.cfg: configures the accelerator to be 1TOPS
* Gem5-aladdin/config/Aladdin: contains the python file to configure the gem5-aladdin system.
* Gem5-aladdin/src/Aladdin/gem5: contains the gem5 source files that links the gem5 datapath with Aladdin. Additionally, gem5 source files inside src/sim folders were also modified to link the accelerator with the processors.
* Gem5-aladdin/src/mem/cache/replacement\_policies: contains the src files that model the replacement policies. Files modelling SPRP policy can be found here.
* Smaug/core: Common files required to run an ML model are found here
* Smaug/core/backend: This file describes the class for accelerator backend. The spad memory size used by the accelerator can be modified here.
* Smaug/operator/smv/kernels: contains the files that model the operation of different ML operators on the accelerator
* Smauf/operators/smv/: files that manage the operation of each ML operator. This includes tiling and iterating through the tiles and gathering the final outputs.
* Smaug/operators/ref : reference implementation of each ML operator. These files model the operators as if they were run on cpu. It doesn’t include tiling.
* Smaug/operators: common class for all the ML operators currently implemented in SMAUG