Tejas Instructions

December 20, 2021

1 How To Setup Tejas

1.1 Pre-requisites

1. Download Intel PIN. Tested for version 3.21 https://software.intel.com/en-us/articles/pin-a-binary-instrumentation-tool-downloads
2. Java version: openjdk8
3. make
4. ant
5. g++

1.2 Commands To Build

Make changes to (set absolute path as per your machine) :
`src/simulator/config/config.xml`
`src/emulator/pin/makefile_linux_mac`
After the changes are done, run
ant make-jar
If everything goes well, this will create a jars folder with tejas.jar in it.
To test everything works, run
cd tests
./test.sh

2 Running a Bag of Tasks

Note: This will work for file mode only.

1. Search for the *Applications* tag in the config file.
2. Add the paths of the trace files (benchmarks) and the number of threads which you want to run for each benchmark.

3 Intermediate Statistics

You can dump the intermediate statistics from a running simulation by sending a "kill -1" signal to the Java process. example: kill -1 <java-pid>

4 Cache Warmup

Set the parameter $NumInsForCacheWarmup$ in config.xml for cache warmup.

5 Generating Traces

Set the parameters in the config file to generate traces to true and give a path for the traces. Then use the command,

java -jar tejas.jar −−generate-trace <rest of the arguments>

You will know that it works if you see a message on stdout stating that no instructions will be executed.

6 Constraint Based Debugging

See example config.xml in src/simulator/config package. To enable CBD, you must enable it in the config file and also provide a time delta. Time Delta specifies how often the constraints are run. These do not correspond to cycles on a core but to the global clock in Tejas. In general, the higher the value, the less often constraints are run.

To specify which constraints to run, you must specify it in a $Constraints$ tag in the config file. $Constraints$ contains a list of $Constraint$ tags. Each $Constraint$ specifies a $Class$ in which the constraint is present, and a $Method$,

which is the actual function you define as a constraint. Currently, constraints can take four types of input, *Double*,*Long*,*Boolean*, and *String*.

6.1 Writing Constraints

Your constraint function must be a **public static** function. This is to allow Java reflection to search and cache it. Order of arguments must agree with the specified order in the config file. *Note:* The function must take object values (Use *Integer* instead of *int*).

Your function may optionally return a *Boolean* object, which should be set to true if the constraint 'Succeded' (You may define the notion of success as you please). If such a value is received by the system, it is reported on stdout.

Get creative with the system! Your function may write to a file, and you can use external tools to analyze the data!