CCU-LANL-Batsim-Framework-Guide

Craig Walker

June 8, 2022

Contents

1 Config

To use the batsim framework you must first make a config file. You can see how to make this file using basefiles/generate_config.py

```
python3 generate_config.py --help
```

```
Usage:
   generate_config.py --config-info <type>
   generate_config.py -i FILE -o PATH [--basefiles PATH] [--output-config] [--increase-
       heldback-nodes]

Required Options 1:
   --config-info <type> Display how the json config is supposed to look like
                                as well as how each part of it can look like.
                                <type> can be:
                                      general | sweeps |
                                      node-sweep | SMTBF-sweep | checkpoint-sweep |
                                          checkpointError-sweep | performance-sweep |
                                      grizzly-workload | synthetic-workload |
                                      input-options | output
Required Options 2:
   -i <FILE> --input <FILE> Where our config lives
   -o <PATH> --output <PATH> Where to start outputing stuff

Options:
   --basefiles <PATH> Where base files go. Make sure you have a 'workloads' and 'platforms'
       folder
                                in this path.
                                [default: ~/basefiles/]

   --output-config If this flag is included, will output the input file to --output directory
                                as --input filename

   --increase-heldback-nodes If this flag is included, will treat heldback nodes as additional
        nodes
```

As you can see you can get –config-info so type:

```
python3 generate_config.py --config-info general
```

```
The general format of a config file:

  { <------------------------------------ Opening curly brace to be proper json

     "Name1":{ <------------------------- The name of an experiment comes first. You can have
         multiple experiments
                                        in one config file and each will end up in it's own
                                            folder under the --output folder.
                                        Notice the opening and closing curly brace. Make sure
                                            you put a comma after the closing
                                        curly brace if you plan on having another experiment in
                                             the same config file

           "input":{ <-------------------- Always make sure you have an input and an output in
                your experiment

              "node-sweep":{ <------------ It is MOST advisable to always start with a node-
                  sweep. All other sweeps can come after this one

              },
              "synthetic-workload":{ <----- Always include either a synthetic-workload or a
                  grizzly-workload after your sweeps

              },
              "option":value, <----- Include any options that will affect all of the jobs on
                  the outside of any sweep or workload

           }, <-------------------------- Make sure you separate your input options with
               commas, but also remember to separate input
                                        and output with a comma
           "output":{ <------------------- Again, always make sure you have an input and
               output in your experiment

              "option":value, <---------- Output is a bit simpler than input. Just make sure
                  it is valid json
              "option":value

           }


     }, <--------------------------------- This closes the experiment and here we have a
         comma because we included another experiment "Name2"
     "Name2":{
        "input":{

           ... <------------------------- Make sure you replace this ellipsis with at least:
                                           * a node-sweep
                                           * a workload
        },
        "output":{

           ... <------------------------- You should replace ellipsis with at least:
                                           * "AAE":true | "makespan":true

        } <------------------------------- Close output
     } <------------------------------------- Close "Name2"
  } <----------------------------------------- Close json
```

1.1 Run Config

You will then want to run the config, for example:

```
python3 generate_config.py -i /ac-project/cwalker/basefiles/$file1 -o /ac-project/cwalker/
    experiments/$folder1 --basefiles ${basefiles} --output-config

# Here $file1 is the json config file we already mentioned
# $folder1 is where you want the output to go for all the simulations corresponding to this
    config file
# $basefiles is where all the scripts are located, including generate_config.py
```

2 Run Experiments

Next you will want to run the experiments

```
python3 run-experiments.py -i /ac-project/cwalker/experiments/$folder1 --time 1200 --sim-time-
    minutes 1200 --socket-start 100003

--time is in minutes as well as --sim-time-minutes
--time is for slurm wall time
--sim-time-minutes is a setting for batsim for how long the sim should go. We used to set it
    to a super high value, I cut back and set it for the walltime in case there was a problem
    with the simulation so SLURM would be happier.
```

run-experiments.py will:

- sbatch real_start.py for each simulation, which will:
  - generate a yaml for the simulation with robin
  - run the simulation with robin
  - when finished the simulation, run post-processing.py

3 Aggregate Results

You will probably want to aggregate all the results into one csv file. Use aggregate_makespan.py for this. It is our main aggregater, even though it was originally used just for makespan.
You can get its usage using this:

```
python3 aggregate_makespan.py --help
```

```
Usage:
   aggregate-makespan.py -i FOLDER [--output FOLDER] [--start-run INT] [--end-run INT]

Required Options:
   -i FOLDER --input FOLDER where the experiments are

Options:
   -o FOLDER --output FOLDER where the output should go
                         [default: input]
   --start-run INT only include runs starting at start-run

   --end-run INT only include runs ending at and including end-run
```

I usually run it like this:

```
python3 aggregate_makespan.py -i ~/ac/experiments/$folder1

#where folder1 equals the location set when you ran generate_config.py
```

This will output a file total_makespan.csv as well as raw_total_makespan.csv

The difference is that raw gathers every makespan.csv file into one file
wheras total_makespan.csv will average all runs together and output the averages for each experiment. If you want to graph variance and such you will need raw

In order to graph single job variances you will need to use the out_jobs.csv file located in each experiment:
$folder1/name_of_experiment/experiment_#/Run_#/output/expe-out/out_jobs.csv

Typically I will tar.gz $folder1 and sftp that back to my computer

4 In Real Life

I have things setup on the ac-cluster like so:

1. Edit an existing config file, and make sure to rename it before saving it
2. Set file1=that new config file
3. Set folder1=a new folder name for this config's set of simulations
4. Set base=`pwd` as long as I'm in the basefiles folder
5. sbatch -p IvyBridge -N1 -n1 -c1 –output=./myBatch.log –export=folder1=$folder1,file1=$file1,basefiles=$base ./myBatch

That's it. Here is my myBatch script

```
#!/bin/bash
source /ac-project/cwalker/start.sh
date
python3 generate_config.py -i /ac-project/cwalker/basefiles/$file1 -o /ac-project/cwalker/
    experiments/$folder1 --basefiles ${basefiles} --output-config
date
python3 run-experiments.py -i /ac-project/cwalker/experiments/$folder1 --time 1200 --sim-time-
    minutes 1200 --socket-start 100003
date
```

run-experiments will sbatch experiment.sh
Here is experiment.sh

```
#!/bin/bash
date
echo "in experiment.sh"
hostname
source /ac-project/cwalker/start2.sh
echo "after source"


echo "real_start.py" && date
python3 /ac-project/cwalker/basefiles/real_start.py --path $jobPath --socketCount $socketCount
    --sim-time $mySimTime
#rm $jobPath/output/expe-out/out* # do all these rm's to cut down on folder size
#rm $jobPath/output/expe-out/post_out_jobs.csv
#rm $jobPath/output/expe-out/raw_post_out_jobs.csv
#rm -rf $jobPath/output/expe-out/log/
date
```

real_start.py is there in basefiles to look at, but here is my start2.sh

```
export MODULEPATH=/opt/ohpc/admin/modulefiles:/opt/ohpc/pub/modulefiles:/ac-project/software/
    modules/linux-centos7-x86_64/Core/linux-centos7-ivybridge && \
export PATH=$PATH:/ac-project/cwalker/Install/bin:/ac-project/cwalker/Downloads/goroot/bin/:/ac
    -project/cwalker/go/bin/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/ac-project/cwalker/Install/lib:/ac-project/cwalker/
    Install/lib64
export LMOD_SH_DBG_ON=1
source /ac-project/cwalker/new_env/bin/activate
```