CCU-LANL Batsim User Doc Walkthrough

Craig Walker

November 4, 2024

Contents

i. Preface

This walkthrough serves to guide the user through a few scenarios of using our CCU-LANL additions to the Batsim/Batsched simulator. If you need a manual instead, make sure to read ***User_Doc_Manual.pdf***.

As the walkthrough, this document will help the user orient themselves to using our tool. This guide will *not* cover how to code for additions to the simulator, however. If one desires this (such as adding another scheduling algorithm, adding another option to Batsim, or adding sweeps) please look to our ***Developer_Doc_Manual.pdf***. It may still be advisable to read this document and/or ***User_Doc_Manual.pdf*** to get a sense of Batsim before you start changing the code.

It is our hope that our additions to Batsim can serve the community well, and that this guide will explain things as clearly as we hope. Good luck with all things Batsim, and all things HPC.

ii. Style Of Document

There is a certain style to this guide that should be made apparent.

- **Inline style:**

  1. `Commands you would run ./from --the --terminal --look --like --this.`
  2. `Just an --argument to a command will look like this.`
  3. `A config 'property': will look like this`
  4. `A/folder/or/file/path/would/look/like/this.`
  5. `Code::would #look like() this.`

- **Block style:**

  1. **Terminal**

```
user > #this is a terminal block, and this is a comment in it.
user > ./and_this_would_be_a_command & | if [[ ]] ; for ;do echo
user > cd ~/our/path # and this is a known command
user > su -
Password:
root > ./this_would_run_as_root
```

  2. **Code**

```
//A c++ code block looks like this, and this is a c/c++ comment in it
and this::is::a::function()
{
  with an int definition;
  int a=10;
  string name="CCU-LANL";
```

```
    return 10;
}
```

```
# and this is python code
import pandas as pd
with open("file.csv","r") as InFile:
  df = pd.read_csv(InFile,sep=",")
def hello:
  print("world")
  q = [ 5,10 ]
```

3. **Explanations**

(a) **Additional Info**

Explains Some Additional Info

Additional info here

(b) **Important Info**

Explains Important Info

This is very important

(c) **Warning Info**

Info That Warns You

This will certainly break the internet

1 Intro

This document will go through 3 walkthroughs of using our tool. But first we walk you through setting up our tool with its deployment.

Next we dive into one of three scenarios. Each scenario will show:

1. how to make the config file
2. how to run the config file
3. what to do while the simulation is running
4. how to aggregate the results
5. and its analysis

While you should be able to copy and paste from this document any of our config and analysis scripts, you can access them from the `.../simulator/basefiles/tests/Walkthrough_Scenarios` folder. They are appropriately named.

2 Deployment

The instructions for all different deployments are detailed in ***User_Doc_Manual.pdf***. We are going to do a bare-metal deployment. The computer is running OpenSuse Leap 15.5 . We are going to start from the beginning:

```
user >   cd /home/craig
user >   git clone https://github.com/hpc/simulator.git
user >   cd simulator/basefiles
user >   ./deploy.sh -f bare-metal -x "/home/craig/simulator"
```

So, the deploy script is now going to run a bunch of commands, download a bunch of tools, and compile a lot of tools to complete the deployment.

If there is an error in compiling one of the tools, the deploy.sh will tell you which line was the last to complete. Try to fix the error that is happening, and you can restart from where you left off with the following:

```
user >   ./deploy.sh -f bare-metal -x "/home/craig/simulator" -l <line number>
```

When it finishes the following will be displayed:

```
  ***********************************************

   Successfully Installed Batsim and Batsched!!!

  ***********************************************

  You will want to make sure .../basefiles/batsim_environment.sh
  is edited with options you need. At a bare minimum set prefix=? to
  the correct prefix. This is probably going to be the full path to your
  simulator folder.

```

So it is telling you to update the `batsim_environment.sh` file with at least the correct prefix. The prefix is the path to your 'simulator' folder. Every time you want to use batsim you should source this `batsim_environment.sh` so that the correct python environment is loaded and all your paths are set up. Plus you get a lot of handy tools, as well.

So let's set the prefix. You can set the prefix manually or automatically. I'll take you through both methods.

```
  user >   cat ~/simulator/basefiles/batsim_environment.sh
################################################################################
# Edit prefix before doing anything, this is mandatory
#
# Here you will find some SBATCH variables you can set
# All SBATCH variables can be set here, not just the ones
```

```
# included.  And of course uncomment the line to take effect.
# ALL SBATCH variables can be found here:
# https://slurm.schedmd.com/sbatch.html#SECTION_INPUT-ENVIRONMENT-VARIABLES
#
# This file can be edited after ./myBatchTasks.sh
# finishes for another batch of simulations with different
# parameters.  Just make sure you keep up with socket-start
# so no sims are overlapping with socket numbers
################################################################################




#export prefix=?

#export SBATCH_PARTITION=standard
#export SBATCH_QOS=standard
#export SBATCH_NO_REQUEUE="yes"

export basefiles_prefix=$prefix/basefiles
export install_prefix=$prefix/Install
export downloads_prefix=$prefix/Downloads
export python_prefix=$prefix/python_env

export PATH=$PATH:$prefix/charliecloud/charliecloud/bin:$basefiles_prefix:\
    $basefiles_prefix/tests:$basefiles_prefix/debug_files:$prefix:$install_prefix/bin:\
       /usr/bin:/usr/sbin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$install_prefix/lib:$install_prefix/lib64
export LMOD_SH_DBG_ON=1
source $python_prefix/bin/activate
source $basefiles_prefix/helpful_functions.sh
```

So you can edit line 20 manually in your favorite text editor. In nano it would be `nano +20 ~/simulator/basefiles/batsim_environment.sh`. What you want to replace it with in our case would be: `export prefix=/home/craig/simulator`. Notice the # is removed and the path to `simulator` is set.

You can also use a tool to do this automatically if you don't feel like doing it manually, or if you have a long path that you don't want to type or cut and paste. So to use do:

```
user >   batPrefix -s
#or
user >   cd /home/craig/simulator
user >   ./basefiles/batPrefix -p $(pwd)
```

The first way to use batPrefix is to use the `-s` option. This simply puts the path of the script, then up one level, into the prefix

assignment. The second way is to specify the path with the `-p` option. In this example we go to the simulator folder and pass the command `pwd` to it.

Now we can source the script:

```
user >   source /home/craig/simulator/basefiles/batsim_environment.sh
(batsim_env) user >   batVersion
1.1.0
```

We sourced the script first. Notice it changed our prompt to include (batsim_env). Now we can use the batsim tools like `batVersion`.

We are all set up now. We will move on to the 3 Scenarios.

3 Scenario 1 - Basic FCFS

3.1 Scenario Description

In this scenario we are going to use the fcfs_fast2 algorithm. It simply uses a first come first serve scheduling of jobs. We will not be using the grizzly workload, as we would like to show using different amount of nodes. Using the grizzly workload we would be forced to use at least 1490 nodes for jobs to be able to complete.

What we want to do is show the make-span decreasing as the amount of nodes is increasing. This is a very simple scenario, but we will be showing it used in multiple ways, and we are using it to kick-start our walkthrough.

1. Scenario 1a
   - with logging on
   - resources: 10-100 nodes
   - duration: mean time of 12 hours
2. Scenario 1b
   - with logging off
   - resources: 10-100 nodes
   - duration: mean time of 24 hours
3. Scenario 1c
   - with logging off
   - resources: 10-100 nodes
   - duration: mean time of 36 hours

3.2 Make The Config File

So here we make the config file. It is probably the most important part of running the sims other than using the correct options when spawning the sims.

You should be familiar with the format of the config file but we'll reorient you.

```
1     The general format of a config file:
2
3     {       <-----------------------------------   Opening curly brace to be proper json
4
5         "Name1":{       <------------------------   The name of an experiment comes first.
6                                                    You can have multiple experiments
7                                                    in one config file and each will end up
8                                                    in it's own folder under the --output
9                                                    folder.Notice the opening and closing
10                                                   curly brace.  Make sure you put a comma
11                                                   after the closing curly brace if you
12                                                   plan on having another
13                                                   experiment in the same config file
14
15
16                                                   Json does not allow for comments
17                                                   (unfortunately).  You may still want
18                #                          \       comments in your config,
```

```
            # python/shell comment      \
                                         \
            // c/c++ style comment        \
            /* c/c++ block style comment  /
                Comments are fun.        /
                This comment is too.    /
                                       /
*/

"input":{    <------------------
                                                    however.  You can use all of these
                                                    types of comments and it will get
                                                    removed before parsing.Be aware that it
                                                    can get difficult to trace down a
                                                    simple mistake in your config when
                                                    many comments are used due
                                                    to the line numbers being off and
                                                    generally more clutter in your config.
                                                    But comments can make things a lot
                                                    clearer, too.The original and a
                                                    stripped version will be in your
                                                    --output folder.
                                                    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
```

```
83          }  <-----------------------------------    Close "Name2"
84      }  <-------------------------------------    Close json
```

Basic Set Up

So we are going to name our experiments: 1a,1b,1c. We will use a node-sweep of 100-150 with a step of 10 for all experiments. The amount of jobs will be 50,000. All jobs will be submitted at time 0.

Logging

We will turn batsim logging on to ”info” and batsched logging on to ”CCU_DEBUG” for experiment 1a. Logging will not be turned on for experiments 1b and 1c.

Resources

The synthetic workload will use from 10-100 nodes with a uniform distribution. This will ensure all jobs will be able to run since the amount of nodes are 100-150.

Duration

The synthetic workload will have jobs with an exponential distribution and a mean time of 12 hours (43200 seconds) for 1a. The synthetic workload will have jobs with an exponential distribution and a mean time of 24 hours (86400 seconds) for 1b. The synthetic workload will have jobs with an exponential distribution and a mean time of 36 hours (129600 seconds) for 1c.

The Config File

So first I setup the experiments we are going to include in this config file with their skeletons:

```
1       {
2         "1a":{
3           "input":{
4
5           },
6           "output":{
7
8           }
9         },
10        "1b":{
11          "input":{
12
13          },
14          "output":{
15
16          }
17        },
18        "1c":{
19          "input":{
20
21          },
22          "output":{
23
24          }
25        }
26      }
```

So now each input needs its sweeps and workload and other options.

```
{
  "1a":{
      "input":{
          "node-sweep":{
              "min":100,
              "max":150,
              "step":10
          },
          "batsim-log":"info",
          "batsched-log":"CCU_DEBUG",
          "batsched-policy":"fcfs_fast2",
          "synthetic-workload":{
              "type":"parallel_homogeneous",
              "machine-speed":1,
              "number-of-jobs":50000,
              "submission-time":"0:fixed",
              "number-of-resources":"10:100:unif",
              "duration-time":"43200:exp", //12 hours
              "seed":10
          }

      },
      "output":{
          "avg-makespan":1

      }
  },
  "1b":{
      "input":{
          "node-sweep":{
              "min":100,
              "max":150,
              "step":10
          },
          //"batsim-log":"info",
          //"batsched-log":"CCU_DEBUG",
          "batsched-policy":"fcfs_fast2",
          "synthetic-workload":{
              "type":"parallel_homogeneous",
              "machine-speed":1,
              "number-of-jobs":50000,
              "submission-time":"0:fixed",
              "number-of-resources":"10:100:unif",
              "duration-time":"86400:exp", //24 hours
              "seed":10
          }
      },
      "output":{
          "avg-makespan":1
      }
  },
  "1c":{
      "input":{
          "node-sweep":{
              "min":100,
              "max":150,
              "step":10
          },
          //"batsim-log":"info",
          //"batsched-log":"CCU_DEBUG",
          "batsched-policy":"fcfs_fast2",
          "synthetic-workload":{
              "type":"parallel_homogeneous",
              "machine-speed":1,
```

```
65                  "number-of-jobs":50000,
66                  "submission-time":"0:fixed",
67                  "number-of-resources":"10:100:unif",
68                  "duration-time":"129600:exp", //36 hours
69                  "seed":10
70              }
71          },
72          "output":{
73              "avg-makespan":1
74          }
75      }
76  }
```

1a

So on line 4 we include the node sweep. On line 9 and 10 we set the logging. On line 11 we set the 'policy' which is the algorithm we are using. On line 12 we define the synthetic workload. On line 18 we set the duration time to 12 hours as a mean time with an exponential distribution. On line 19 we set the seed to 10. This is an overall seed for generating the workload. This was added so you would get the same results that we get.

On line 24 we say we just want one sim per job. This is normal because we aren't introducing any randomness in the simulation. The only randomness is in the workload itself. So we don't need to average anything.

1b

On line 30 we include the node sweep. On line 35 and 36 we have commented out the logging. On line 37 we set the algorithm to "fcfs_fast2". On line 38 we define the synthetic workload. On line 44 we set the duration time to 24 hours as a mean time with an exponential distribution. The seed is, again, set to 10.

On line 49, we again say we just want one sim per job.

1c

On line 54 we include the node sweep. On line 59 and 60 we have commented out the logging. On line 61 we set the algorithm to "fcfs_fast2". On line 62 we define the synthetic workload. On line 68 we set the duration time to 36 hours as a mean time with an exponential distribution.

On line 72, we again say we just want one sim per job.

3.3 Run The Config File

The tool we use to run the config file is called `myBatchTasks.sh`. The first thing to determine when we run the config file is where we want to set the project folder to. We are going to use the default `.../simulator/experiments/` folder. This will be used if we don't pass an absolute path. We will call our project folder `Scenario_1`.

The way we run `myBatchTasks.sh` is to set `--file ${file1}` and `--output ${experiments_folder}/<project folder>` The easiest thing to do to use `myBatchTasks.sh` is to first use `batEdit` to set your `file1` variable. It is easiest to edit a config file that is already there, which batEdit will allow you to choose a config file to edit. If you don't yet have any configs you can still create one with batEdit, simply pass batEdit the `-f <filename>` option.

So let's assume this is your first config, we run `batEdit -e -f walkthrough_1.config`. This will make a new file `walkthrough_1.config` and allow us to edit it in `nano`. I would then maybe cut and paste the config file in this pdf document. Then hit Ctrl+o to save, then Ctrl+x to exit. Now if you `echo $file1`, you will see that $file1 now holds the path to your config file.

This is the command we used to run the config file:

```
(batsim_env) user >  myBatchTasks.sh -f ${file1} -o scenario_1_fin_3 -m bare-metal -p
↪  tasks -t 10 -s 40001
```

So let's go over it. We have set $file1 as the config file path. batEdit uses an absolute path to set $file1, but if we just used a file name and not an absolute path then it would assume the file was in the `$prefix/configs` folder. Next, the project folder is set to `scenario_1_fin_3`. Since we didn't give it an absolute path it assumes the `$prefix/experiments` folder. Next is the method we deployed batsim. We used a bare-metal deployment so we tell it that, although it is the default. We are going to run the sims in parallel and the method is using tasks. On your own machine without SLURM set up you would either use `-p "none"` or `-p "background"`. Since we ARE using `-p tasks` we set how many tasks to run on one node. In this case there is only one node, so it is basically saying how many sims to run at once. We set it to 10 given the constraints of our system. Finally we use `-s 40001` to set a socket to start experiments at. This is useful if you are running multiple configs at a time.

If you would like to know what the command looks like if you wanted to use 4 cores on your computer, to run 4 sims at a time, but you don't have SLURM setup, then the following command might suit you:

```
(batsim_env) user >  myBatchTasks.sh -f ${file1} -o scenario_1_fin_3 -m bare-metal -p
↪  background -t 4 -s 40001
```

There you have it. The following is the output of running the SLURM version:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments> myBatchTasks.sh -f ${file1} -o scenario_1_fin_3 -m
001f /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/configs/walkthrough_1.config
o scenario_1_fin_3
m bare-metal
p tasks
t 10
s 40001
Mon Oct 28 03:59:04 AM EDT 2024


 ********************  Checking JSON config file  ********************

********************  JSON Check Appears To Be SUCCESSFUL ********************
creating workloads for experiment: 1a ...
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
creating folder structure for experiment: 1a ...
         creating folder structure for: experiment_1/experiment_6 ...
         creating folder structure for: experiment_2/experiment_6 ...
         creating folder structure for: experiment_3/experiment_6 ...
         creating folder structure for: experiment_4/experiment_6 ...
         creating folder structure for: experiment_5/experiment_6 ...
         creating folder structure for: experiment_6/experiment_6 ...
creating workloads for experiment: 1b ...
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
creating folder structure for experiment: 1b ...
         creating folder structure for: experiment_1/experiment_6 ...
         creating folder structure for: experiment_2/experiment_6 ...
         creating folder structure for: experiment_3/experiment_6 ...
         creating folder structure for: experiment_4/experiment_6 ...
         creating folder structure for: experiment_5/experiment_6 ...
         creating folder structure for: experiment_6/experiment_6 ...
creating workloads for experiment: 1c ...
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
         index: 1
creating folder structure for experiment: 1c ...
         creating folder structure for: experiment_1/experiment_6 ...
         creating folder structure for: experiment_2/experiment_6 ...
         creating folder structure for: experiment_3/experiment_6 ...
         creating folder structure for: experiment_4/experiment_6 ...
         creating folder structure for: experiment_5/experiment_6 ...
         creating folder structure for: experiment_6/experiment_6 ...
. /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/batsim_environment.sh;sbatch  -N1 --exclusive --ntasks=11 --e
/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3',folder='scenario_1_fin_3',mySimTime='315360
LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1a/experiment_1/id_1/Run_1 /home/craig/LANL/Post_J
/simulator/experiments/scenario_1_fin_3:PATH:/1a/experiment_2/id_1/Run_1 /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/
H:/1a/experiment_3/id_1/Run_1 /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1a/exper
ANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1a/experiment_5/id_1/Run_1 /home/craig/LANL/Post_Ju
simulator/experiments/scenario_1_fin_3:PATH:/1a/experiment_6/id_1/Run_1 /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/e
:/1b/experiment_1/id_1/Run_1 /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1b/experi
NL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1b/experiment_3/id_1/Run_1 /home/craig/LANL/Post_Jul
imulator/experiments/scenario_1_fin_3:PATH:/1b/experiment_4/id_1/Run_1',experimentString=' 1a 1a 1a 1a 1a 1a 1b 1b 1b 1b',jobString=' 1 2 3
id_1 id_1 id_1 id_1 id_1 id_1 id_1 id_1 id_1 id_1',runString=' Run_1 Run_1 Run_1 Run_1 Run_1 Run_1 Run_1 Run_1 Run_1 Run_1',basefiles='/home/cra
y_bare_metal_2/simulator/basefiles',priority='0',socketCountString=' 40001 40002 40003 40004 40005 40006 40007 40008 40009 40010',signal_nu
ToSbatch=' ',comment='scenario_1_fin_3_1',output='/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_
                      --output=/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3/sbatch-1.ou
                      /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/experiment.sh tasks bare-metal
Submitted batch job 1637
. /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/batsim_environment.sh;sbatch  -N1 --exclusive --ntasks=11 --e
/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3',folder='scenario_1_fin_3',mySimTime='315360
LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3:PATH:/1b/experiment_5/id_1/Run_1 /home/craig/LANL/Post_J
/simulator/experiments/scenario_1_fin_3:PATH:/1b/experiment_6/id_1/Run_1 /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/
```

This shows first some debug info on what options were sent to the script. Next comes a check on the json included in the config file. Next would normally come a workload filename that was generated, but since we had already run this experiment the workload was already made with those exact options so, for us, nothing needed to be generated. You can see that for experiment 1a it made sure that it had the workload it needed for each id, or just id_1 in this case.

After that, it sets up the folder structure for the experiment and shows which job it is currently working on. The time in between job folder set up can take a few minutes if you have a lot of runs, so don't get too worried if you don't see fast progress.

Next you will see a whole bunch of stuff dumped onto the screen. This is the sbatch command that was used. When you are using `-p "none"` or `-p "background"` you will see something different. But in this case, you see that 2 batch jobs were submitted, one was numbered 1637, the other is cut-off but it is safe to assume it was 1638.

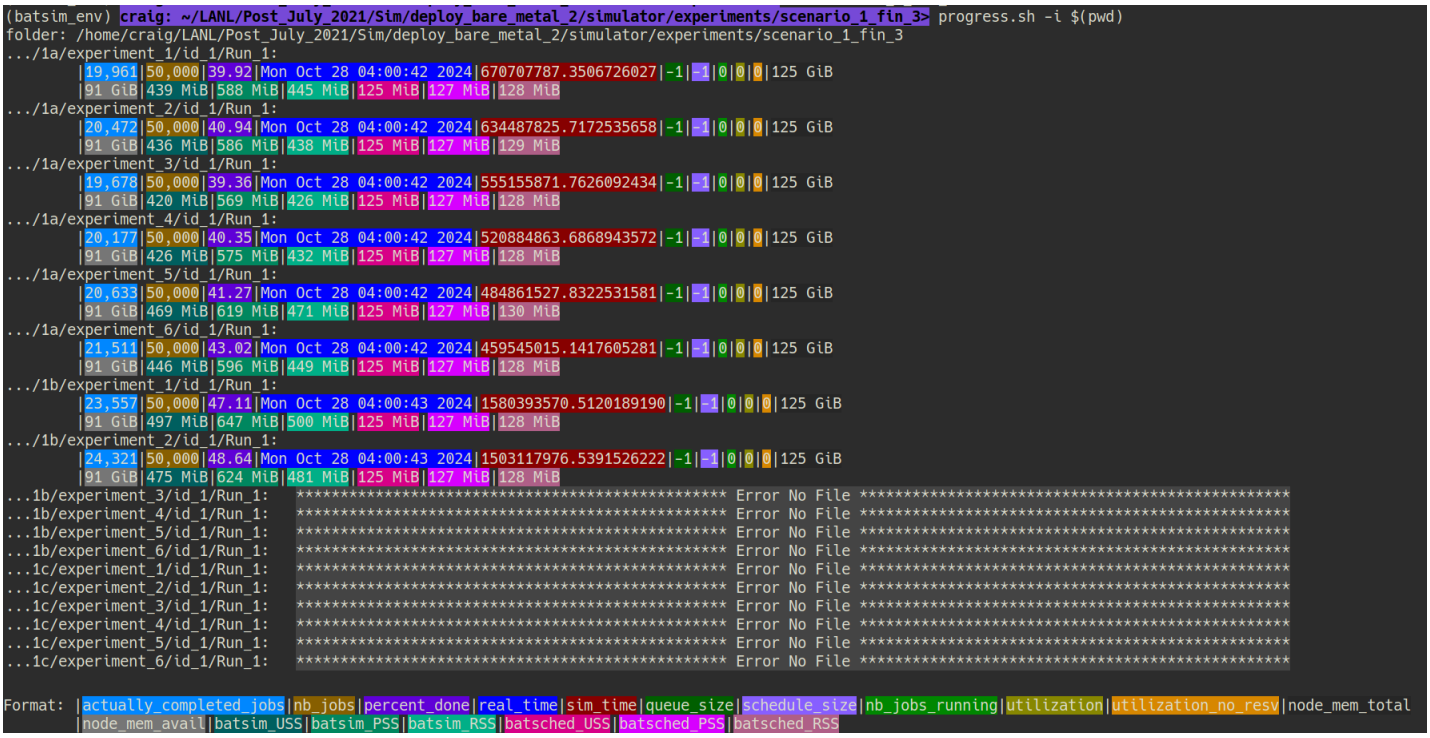
The next section details what to expect while running the sims.

3.4 While Running The Sims

3.4.1 progress.sh

While running the sims you can use our `progress.sh` script. Of course you should definitely run `progress.sh --help`, but we will cover a couple options here and show you what to expect.

First let's show you a basic invocation of it. While in the project folder, run `progress.sh -i $(pwd)`. You should get something similar to the following:



Notice with this view you get a lot of wasted space. But it is easy for a beginner to see the correlation of the colors with what the colors and info represent at the bottom.

You can hone in on the purple box as it is probably the most important as it tells what percent has completed so far.

Usually I recommend using the `-Z` option, which will condense the space:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3> progress.sh -i $(pwd) -Z
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3
1a/e_1/i_1/r_1:
|25,645|50,000|51.29|Mon Oct 28 04:01:21 2024|861715748.4123009443|-1|-1|0|0|0|125 GiB|91 GiB|509 MiB|658 MiB|515 MiB|125 MiB|127 MiB|128 MiB
1a/e_2/i_1/r_1:
|26,549|50,000|53.10|Mon Oct 28 04:01:21 2024|821576064.4563560486|-1|-1|0|0|0|125 GiB|91 GiB|450 MiB|658 MiB|452 MiB|125 MiB|127 MiB|129 MiB
1a/e_3/i_1/r_1:
|25,672|50,000|51.34|Mon Oct 28 04:01:21 2024|725163216.9276983738|-1|-1|0|0|0|125 GiB|91 GiB|493 MiB|642 MiB|499 MiB|125 MiB|127 MiB|128 MiB
1a/e_4/i_1/r_1:
|26,167|50,000|52.33|Mon Oct 28 04:01:21 2024|674760363.6631424427|-1|-1|0|0|0|125 GiB|91 GiB|432 MiB|639 MiB|439 MiB|125 MiB|127 MiB|128 MiB
1a/e_5/i_1/r_1:
|26,444|50,000|52.89|Mon Oct 28 04:01:21 2024|620876390.8819391727|-1|-1|0|0|0|125 GiB|91 GiB|482 MiB|689 MiB|484 MiB|125 MiB|127 MiB|130 MiB
1a/e_6/i_1/r_1:
|27,568|50,000|55.14|Mon Oct 28 04:01:21 2024|589106198.8419297934|-1|-1|0|0|0|125 GiB|91 GiB|461 MiB|669 MiB|465 MiB|125 MiB|127 MiB|128 MiB
1b/e_1/i_1/r_1:
|30,050|50,000|60.10|Mon Oct 28 04:01:21 2024|2027721134.0071098804|-1|-1|0|0|0|125 GiB|91 GiB|518 MiB|726 MiB|520 MiB|125 MiB|127 MiB|128 MiB
1b/e_2/i_1/r_1:
|30,860|50,000|61.72|Mon Oct 28 04:01:21 2024|1916867222.5779340267|-1|-1|0|0|0|125 GiB|91 GiB|494 MiB|701 MiB|500 MiB|125 MiB|127 MiB|128 MiB
...1b/e_3/i_1/r_1:   ********************************************** Error No File **********************************************
...1b/e_4/i_1/r_1:   ********************************************** Error No File **********************************************
...1b/e_5/i_1/r_1:   ********************************************** Error No File **********************************************
...1b/e_6/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_1/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_2/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_3/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_4/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_5/i_1/r_1:   ********************************************** Error No File **********************************************
...1c/e_6/i_1/r_1:   ********************************************** Error No File **********************************************

Format: |comp_jobs|jobs|%_done|rl_time|sm_time|q_sz|sched_sz|nb_run|util|util_no_r|tot_mem|avail_mem|bat_USS|bat_PSS|bat_RSS|sched_USS|sched_PSS|sched_RSS
```

Notice the ”Error No File” entries. There could be many reasons for this, but in this instance it is simply that those sims haven’t started yet.

You can keep running the same command, using the up arrow, to see the progress. `progress.sh` allows for a lot of customization. Suppose you don’t want the memory being used shown. You could tell it what you DO want shown and omit the memory, or use the `-z "m"` option. `-z` tells it what you DON’T want shown.

Here it is in action:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3> progress.sh -i $(pwd) -Z -z "m"
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3
1a/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:56 2024|1667578747.3795866966|-1|-1|0|0|0
1a/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:36 2024|1538219951.7615644932|-1|-1|0|0|0
1a/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:51 2024|1404434573.7095315456|-1|-1|0|0|0
1a/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:41 2024|1285201743.6878411770|-1|-1|0|0|0
1a/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:44 2024|1171105019.7667162418|-1|-1|0|0|0
1a/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:28 2024|1065337016.9839813709|-1|-1|0|0|0
1b/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:00 2024|3335157494.7548394203|-1|-1|0|0|0
1b/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:03:50 2024|3076439903.5191154480|-1|-1|0|0|0
1b/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:21 2024|2808869147.4153695107|-1|-1|0|0|0
1b/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:51 2024|2570403487.3722653389|-1|-1|0|0|0
1b/e_5/i_1/r_1:    |44,716|50,000|89.43|Mon Oct 28 04:13:39 2024|2100631449.5304884911|-1|-1|0|0|0
1b/e_6/i_1/r_1:    |44,297|50,000|88.59|Mon Oct 28 04:13:39 2024|1893155968.0093700886|-1|-1|0|0|0
1c/e_1/i_1/r_1:    |46,260|50,000|92.52|Mon Oct 28 04:13:39 2024|4633607542.0061941147|-1|-1|0|0|0
1c/e_2/i_1/r_1:    |45,990|50,000|91.98|Mon Oct 28 04:13:39 2024|4248218570.6070952415|-1|-1|0|0|0
1c/e_3/i_1/r_1:    |46,067|50,000|92.13|Mon Oct 28 04:13:39 2024|3886822532.8487529755|-1|-1|0|0|0
1c/e_4/i_1/r_1:    |45,874|50,000|91.75|Mon Oct 28 04:13:39 2024|3542742320.6205291748|-1|-1|0|0|0
1c/e_5/i_1/r_1:    |46,700|50,000|93.40|Mon Oct 28 04:13:39 2024|3283573746.0533781052|-1|-1|0|0|0
1c/e_6/i_1/r_1:    |48,189|50,000|96.38|Mon Oct 28 04:13:39 2024|3078698054.6798233986|-1|-1|0|0|0

Format: |comp_jobs|jobs|%_done|rl_time|sm_time|q_sz|sched_sz|nb_run|util|util_no_r
```

You could also tell it you don’t want the ’schedule info’ at the same time, since this algorithm doesn’t even use the schedule, by using `-z "m S"`:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3> progress.sh -i $(pwd) -Z -z "m S"
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3
1a/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:56 2024|1667578747.3795866966|0|0
1a/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:36 2024|1538219951.7615644932|0|0
1a/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:51 2024|1404434573.7095315456|0|0
1a/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:41 2024|1285201743.6878411770|0|0
1a/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:44 2024|1171105019.7667162418|0|0
1a/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:28 2024|1065337016.9839813709|0|0
1b/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:00 2024|3335157494.7548394203|0|0
1b/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:03:50 2024|3076439903.5191154480|0|0
1b/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:21 2024|2808869147.4153695107|0|0
1b/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:51 2024|2570403487.3722653389|0|0
1b/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:26 2024|2342210039.5304083824|0|0
1b/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:29 2024|2130674033.9650659561|0|0
1c/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:14 2024|5002736242.1295967102|0|0
1c/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:16 2024|4614659855.2763834000|0|0
1c/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:15 2024|4213303721.1211547852|0|0
1c/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:18 2024|3855605231.0567278862|0|0
1c/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:09 2024|3513315059.2940406799|0|0
1c/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:13:56 2024|3196011050.9462537766|0|0

Format: |comp_jobs|jobs|%_done|rl_time|sm_time|util|util_no_r
```

There is also an elapsed time you can show along with what not to show using the `-E` option.

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3> progress.sh -i $(pwd) -Z -z "m S" -E
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3
1a/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:56 2024|1667578747.3795866966|0d-00:05:50|0|0
1a/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:36 2024|1538219951.7615644932|0d-00:05:29|0|0
1a/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:51 2024|1404434573.7095315456|0d-00:05:45|0|0
1a/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:41 2024|1285201743.6878411770|0d-00:05:34|0|0
1a/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:44 2024|1171105019.7667162418|0d-00:05:37|0|0
1a/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:28 2024|1065337016.9839813709|0d-00:05:21|0|0
1b/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:04:00 2024|3335157494.7548394203|0d-00:04:53|0|0
1b/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:03:50 2024|3076439903.5191154480|0d-00:04:43|0|0
1b/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:21 2024|2808869147.4153695107|0d-00:03:08|0|0
1b/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:08:51 2024|2570403487.3722653389|0d-00:03:41|0|0
1b/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:26 2024|2342210039.5304083824|0d-00:05:33|0|0
1b/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:29 2024|2130674033.9650659561|0d-00:05:35|0|0
1c/e_1/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:14 2024|5002736242.1295967102|0d-00:05:20|0|0
1c/e_2/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:16 2024|4614659855.2763834000|0d-00:05:22|0|0
1c/e_3/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:15 2024|4213303721.1211547852|0d-00:05:21|0|0
1c/e_4/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:18 2024|3855605231.0567278862|0d-00:05:24|0|0
1c/e_5/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:14:09 2024|3513315059.2940406799|0d-00:05:15|0|0
1c/e_6/i_1/r_1:    |50,000|50,000|100.00|Mon Oct 28 04:13:56 2024|3196011050.9462537766|0d-00:05:02|0|0

Format: |comp_jobs|jobs|%_done|rl_time|sm_time|el_time|util|util_no_r
```

And finally, there is the `-N` option which is more important when using real checkpointing which will be covered in Scenario 2:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3> progress.sh -i $(pwd) -N
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_1_fin_3
1a/e_1/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:50
1a/e_2/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:29
1a/e_3/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:45
1a/e_4/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:34
1a/e_5/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:37
1a/e_6/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:21
1b/e_1/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:04:53
1b/e_2/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:04:43
1b/e_3/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:03:08
1b/e_4/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:03:41
1b/e_5/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:33
1b/e_6/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:35
1c/e_1/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:20
1c/e_2/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:22
1c/e_3/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:21
1c/e_4/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:24
1c/e_5/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:15
1c/e_6/i_1/r_1:    |50,000|50,000|100.00|100.00|50,000|50,000|0|0d-00:05:02

Format: |comp_jobs|jobs|%_done|all_%|tot_comp|o_jobs|nb_re|el_time
```

There are many more options than that, but hopefully that gives you an idea of what to expect using `progress.sh`.

There are three more things to cover:

1. logs
2. srun output
3. sbatch output

3.4.2 logs

So while you are running the sims we set the logging to `"batsim-log":"info"` and `"batsched-log":"CCU_DEBUG"` for experiment "1a", so you can open these logs either while you are running the sims, or after.

Logs Can Get HUGE!!!

Keep in mind that these logs can get very large depending on which verbosity you set. This can cause the following problems:

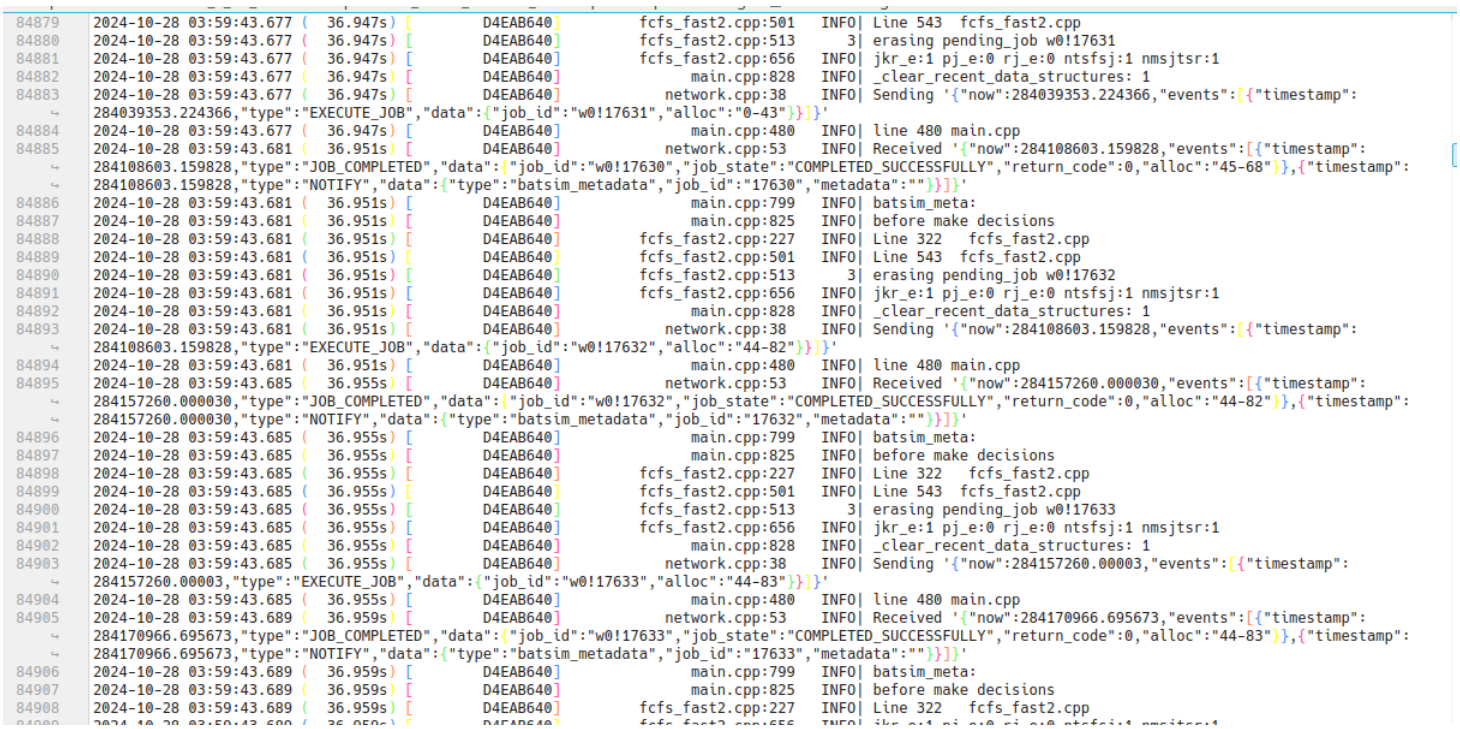
- sims become slow
- no more storage to write out the logs, or anything for that matter, which will crash the sims
- system crashes
- when opening the logs it will take a long time to open, and may use all your memory depending on the text editor you

choose.

The logs can be found in the `.../<Run Folder>/output/expe-out/log/` folder and there you will find:

- `sched.err.log`
  - where the batsched log goes. This is usually what you want to look at.
- `sched.out.log`
  - this is not used
- `batsim.log`
  - where the batsim log goes.
- `SoftErrors.log`
  - where the 'soft errors' go. These are errors that might be useful, but don't impede the running of the sim.

Here is a look at the batsched log file:



Notice on line 84883 you see the 'Sending' message. This is where the scheduler is done for this pass and is sending info over to Batsim. Line 84885 has the scheduler coming alive again with a 'Received' message. This back and forth messaging will continue for the duration of the sim.

3.4.3 srun output

When using SLURM for parallelization, each srun execution will generate a `slurm-<jobid>.out` file in the `.../<Run folder>/output/` folder. When using the real checkpointing this folder may have multiple `slurm-<jobid>.out` files, each with their respective job id.

You can look at this file, or the *latest* file in the case of real checkpointing, to see if any errors have happened.

3.4.4 sbatch output

When batches of files are being sent to SLURM, each sbatch gets its own output file. This can tell us if something went wrong at the sbatch level. The files will be named `.../<project folder>/sbatch-<consecutive sbatch number>.out`. When real checkpointing is turned on and you have requeuing on then you will have `.../<project folder>-<consecutive sbatch number>_<resubmit number>.out` as a file. This easily pollutes the project folder. In the future, hopefully we can put these files in a folder of their own.

3.5 Aggregating Results

In order to aggregate results, we have a tool called `aggregate_makespan.py`. To use, it is as simple as passing it the project folder:

```
1  user >  cd /home/craig/simulator/experiments/Scenario_1
2  user >  aggregate_makespan.py -i $(pwd)     #  $(pwd) will give us the current directory
```

If you are running lots of runs, most of the time there will be a few simulations that don't complete for one reason or another. You will see which ones didn't complete during aggregation. It will go job by job and tell you which ones did not complete and then will total it up for you before going on to the next job. A few aren't so bad, but a lot merits more investigation.

In order to save the results of the aggregation there is a file called `errors_total_makespan.txt`. It has a summary at the bottom. If everything went well with few, if any, sims not completing, then your aggregated results will be in `total_makespan.csv`

This brings us to the next step, of analyzing the `total_makespan.csv` file.

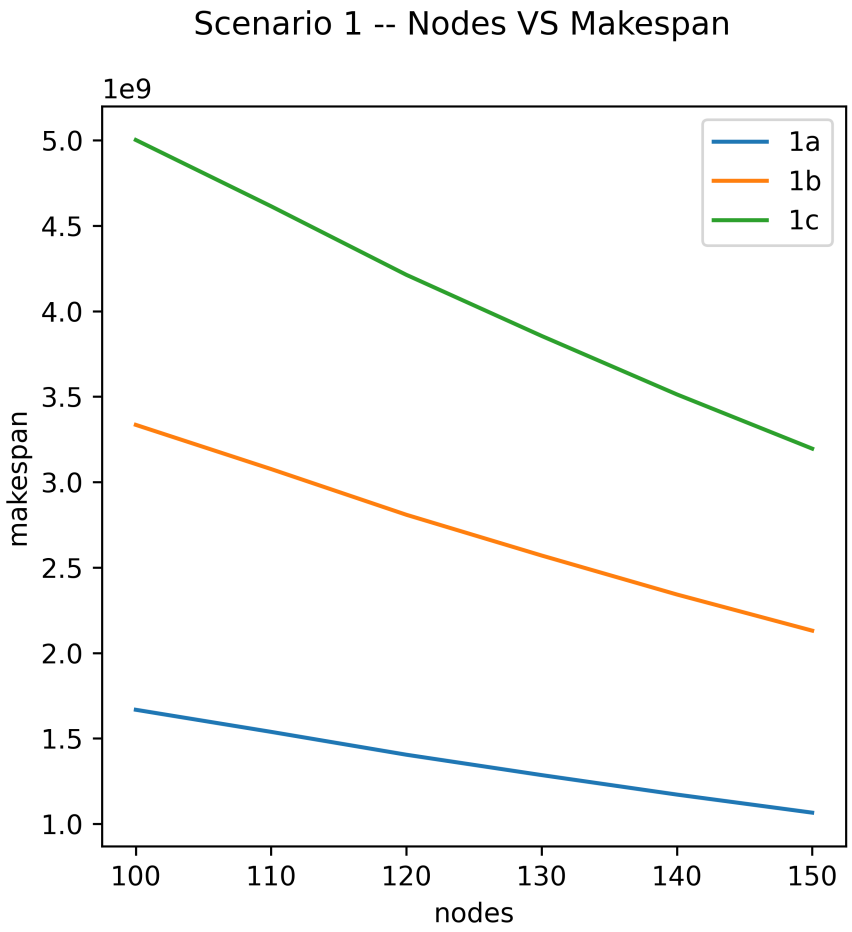
3.6 Analysis

Currently `total_makespan.csv` has the following header:

nodes	SMTBF	NMTBF	fixed-failures	repair-time	MTTR
makespan_sec	makespan_dhms	AAE	avg_tat	avg_tat_dhms	avg_waiting
avg_waiting_dhms	avg_pp_slowdown	avg-pp-slowdown-tau	number_of_jobs	submission_time	submission_compression
avg_utilization	SMTBF_failures	MTBF_failures	Fixed_failures	rejected_not_enough_available_resources	jitter
avg-pp-slowdown	avg-pp-slowdown_dhms	avg-pp-slowdown-Tau	id	job	exp

We can then make the following code:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
fig, axs = plt.subplots(1, 1, figsize=(5, 5))

df = pd.read_csv("./total_makespan.csv",header=0,sep=",")

Adf=df.loc[df["exp"]=="1a"]
Bdf=df.loc[df["exp"]=="1b"]
Cdf=df.loc[df["exp"]=="1c"]

plt.plot(Adf["nodes"],Adf["makespan_sec"],label="1a")
plt.plot(Bdf["nodes"],Bdf["makespan_sec"],label="1b")
plt.plot(Cdf["nodes"],Cdf["makespan_sec"],label="1c")
plt.legend()
axs.set_xlabel("nodes")
axs.set_ylabel("makespan")
fig.suptitle("Scenario 1 -- Nodes VS Makespan")
fig.savefig("myplot.png",dpi=600)
```



4 Scenario 2 - Easy Backfilling with Real Checkpointing

4.1 Scenario Description

In this scenario we are going to use the easy_bf3 algorithm. It does easy backfilling without using the schedule. We will use this algorithm to show a miriad of things including:

- failures
- repair times
- real checkpointing
  - checkpoint signal
  - automatic requeuing
  - wallclock option
- grizzly workload

We want to show that as repair time increases the avg-waiting time increases as well. This would entice someone to repair nodes sooner so that the quality of service is kept to a certain level. The difference in the scenarios is that Scenario 2a uses a fixed repair time sweep, whereas Scenario 2b uses a Mean Time To Repair sweep.

1. Scenario 2a
   - SMTBF: 16-64 by 16
   - repairTime: 2-10 days by 2
   - checkpoint interval: optimal
   - queue-policy: ORIGINAL-FCFS
   - real-checkpointing on: signal 37, automatic requeing, wallclock 6 minutes
   - grizzly workload: 50,000 jobs , dump-time:3%,read-time: 2%
2. Scenario 2b
   - SMTBF: 16-64 by 16
   - MTTR: 2-10 days by 2
   - checkpoint interval: optimal
   - queue-policy: ORIGINAL-FCFS
   - real-checkpointing on: signal 37, automatic requeuing, wallclock 6 minutes
   - grizzly workload: 50,000 jobs , dump-time:3%,read-time: 2%
3. Scenario 2c: Scenario 2a and 2b without SLURM, manual restart
   - SMTBF: 16-64 by 16
   - MTTR: 2-10 days by 2
   - checkpoint interval: optimal
   - queue-policy: ORIGINAL-FCFS
   - real-checkpointing on: interval every 4 minutes
   - grizzly workload: 50,000 jobs , dump-time:3%,read-time: 2%

4.2 Make The Config File

If you need a refresher on the format of a config file Scenario 1 covered that.

Here we show what the config file we used looks like:

```
    {
      "2a":{
          "input":{
              "node-sweep":{
                 "range":[1490]
              },
              "SMTBF-sweep":{
               "compute-SMTBF-from-NMTBF":true,
                  "min":16,
                  "max":64,
                  "step":16,
                  "formula":"1728000000*(1/i)"
              },
              "repairTime-sweep":{
                  "min":2,
                  "max":10,
                  "step":2,
                  "formula":"3600*24*i"
              },
              "checkpoint-sweep":{
                  "range":["optimal"]
              },
              "batsched-policy":"easy_bf3",
              "checkpointing-on":true,
              "queue-policy":"ORIGINAL-FCFS",
              "seed-failures":10,
              "seed-failure-machine":10,
              "checkpoint-batsim-signal":37,
              "checkpoint-batsim-requeue":true,
              "start-from-checkpoint-keep":2,
              "grizzly-workload":{
                  "type":"parallel_homogeneous",
                  "machine-speed":1,
                  "input":"sanitized_jobs.csv",
                  "number-of-jobs":50000,
                  "dump-time":"3%",
                  "read-time":"2%",
                  "time":"01-01-2018:"
              }

          },
          "output":{
              "avg-makespan":1

          }
      },
      "2b":{
          "input":{
              "node-sweep":{
                 "range":[1490]
              },
              "SMTBF-sweep":{
               "compute-SMTBF-from-NMTBF":true,
                  "min":16,
                  "max":64,
                  "step":16,
                  "formula":"1728000000*(1/i)"
```

```
58                },
59                "MTTR-sweep":{
60                    "min":2,
61                    "max":10,
62                    "step":2,
63                    "formula":"3600*24*i"
64                },
65                "checkpoint-sweep":{
66                    "range":["optimal"]
67                },
68                "batsched-policy":"easy_bf3",
69                "checkpointing-on":true,
70                "queue-policy":"ORIGINAL-FCFS",
71                "seed-failures":10,
72                "seed-failure-machine":10,
73                "seed-repair-times":10,
74                "checkpoint-batsim-signal":37,
75                "checkpoint-batsim-requeue":true,
76                "start-from-checkpoint-keep":2,
77                "grizzly-workload":{
78                    "type":"parallel_homogeneous",
79                    "machine-speed":1,
80                    "input":"sanitized_jobs.csv",
81                    "number-of-jobs":50000,
82                    "dump-time":"3%",
83                    "read-time":"2%",
84                    "time":"01-01-2018:"
85                }
86
87            },
88            "output":{
89                "avg-makespan":1
90
91            }
92        }
93    }
```

2a

So on line 4 we do the node-sweep but we just use a `"range"` with one value in it. This is equivalent to just setting the nodes to that number. We use the sweep sometimes instead of setting the value outright just to keep the syntax around for when we change the file.

Now on line 7 we do the SMTBF sweep. We set `"compute-SMTBF-from-NMTBF":true` so that it will interpret the values generated as NMTBF, but then convert them to the SMTBF based on each experiment's number of nodes. We set the min, max, step starting on line 9. We set a formula to operate on those generated numbers. At Los Alamos National Lab, it was given to us that their biggest cluster "Trinity" had a NMTBF of 20k days. We use this as a failure frequency but then increase it as we see fit. In this case we are using 16x to 64x what Trinity supposedly has for a failure rate.

Line 14 starts our repair time sweep. Repair time uses seconds so we do the math to convert to days using a formula. So, we are doing 2-10 days with a step of 2 days. On line 20 we set the simulated checkpoint interval to optimal (More to say about this in the 'Running the Config' section).

We then set the algorithm on line 23, and since we are using simulated checkpointing, we must turn it 'on' on line 24. We set the queue-policy (for when a job is killed, where it goes in the queue) to "ORIGINAL-FCFS". We seed the failures, and the machine that failures choose to occur on.

We then do our real checkpointing options. On line 28 we set `"checkpoint-batsim-signal":37` so that SLURM will send a signal 37 to the sims to tell it to do a real checkpoint when a certain amount of time is left before the wallclock time is done. This setting is set in the `batsim_environment.sh` file. If we do the following:

```
(batsim_env) user >   cd $basefiles_prefix
(batsim_env) user >   cat batsim_environment.sh

################################################################################
# Edit prefix before doing anything, this is mandatory
#
# Here you will find some SBATCH variables you can set
# All SBATCH variables can be set here, not just the ones
# included.  And of course uncomment the line to take effect.
# ALL SBATCH variables can be found here:
#
#
# This file can be edited after ./myBatchTasks.sh
# finishes for another batch of simulations with different
# parameters.  Just make sure you keep up with socket-start
# so no sims are overlapping with socket numbers
################################################################################




export prefix="/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator"

export SBATCH_PARTITION=main
#export SBATCH_QOS=standard
#export SBATCH_NO_REQUEUE="yes"
export SBATCH_SIGNAL="B:37@120"   <----------------------  here


export basefiles_prefix=$prefix/basefiles
export install_prefix=$prefix/Install
export downloads_prefix=$prefix/Downloads
export python_prefix=$prefix/python_env

export PATH=$PATH:$prefix/charliecloud/charliecloud/bin:$basefiles_prefix:\
       $basefiles_prefix/tests:$basefiles_prefix/debug_files:$prefix:\
       $install_prefix/bin:/usr/bin:/usr/sbin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$install_prefix/lib:$install_prefix/lib64
export LMOD_SH_DBG_ON=1
source $python_prefix/bin/activate
source $basefiles_prefix/helpful_functions.sh
```

you can see on line 27 that we set signal 37 to 120 seconds before the wallclock time ends. The 120 seconds will need adjusting based on how long it takes to write out your checkpoint. But that's what `37@120` means. As far as the `B:`, that tells it to send the signal to the sbatch script and not the individual steps. That is needed in our setup and was changed to require this when we added the automatic requeuing.

Ok, back to the config file: On line 29 we tell it we want to requeue automatically if the walltime is insufficient to complete the sim. And on line 30 we get the `"start-from-checkpoint-keep":2`. This says that since we are requeuing automatically we

will be starting from a checkpoint, possibly multiple times, and we want to keep the 'Frames' before the current one. Look at the User_Doc_Manual.pdf **Section 1.3.7:** Real Checkpointing: checkpoint_#/ and expe-out_#/

On line 31 we make a grizzly workload and tell it what file to use for the workload. The 2018 real grizzly data is the "sanitized_jobs.csv" file. We are going to start at the beginning of the year of 2018, but we only want 50k jobs so we set that on line 35.

On line 36 and 37 we set the `"dump-time":"3%"` and `"read-time":"2%"`. This is required if you are going to use simulated checkpointing. And on line 38 we say we want to start the jobs at the beginning of 2018 and go to the latest date of the file. We are talking about the times the jobs were submitted when we are talking about this `"time":` property.

2b

The only difference here is that we use a MTTR (Mean Time To Repair) sweep on line 59 instead of the repairTime sweep we used on line 14.

4.3 Run The Config File

First off, we all make mistakes, so when I went to run this config file I got an error with line 24. I will show you this error so you don't panic when you see it. It is very common to get your config in improper json.

So let's see the output:

```
(batsim_env)  (submission_compression)craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator> myBatchTasks.sh -f ${file1} -o scenar
 10 -s 40001 -w 00:06:00
f /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/configs/walkthrough_2.config
o scenario_2_1
m bare-metal
p tasks
t 10
s 40001
w 00:06:00
Mon Oct 28 07:17:58 AM EDT 2024


 ********************  Checking JSON config file  ********************

Expecting ',' delimiter: line 24 column 36 (char 640)

!!!!!!!!!!!!!!!!!!!!  ERROR with Json File.  See message above  !!!!!!!!!!!!!!!!!!!!
```

So I forgot to use a ',' is what it told me. What the case actually was - was that I put double quotes on the right side of `"checkpointing-on":true",`. See the mistake? 'true' shouldn't be quoted as it is recognized as a bool, but not only that, I only had the quote on the right side.

Now, in running this config, I had made another mistake and didn't add in the `"dump-time":` or `"read-time":` properties for the grizzly workload the first time around. Since I was using simulated checkpointing, this was required.

So having this error started a bunch of simulations, but when I looked at their progress using `progress.sh`, all of the sims were showing "Error No File". This can happen early in the sim as it may take time for a job to finish, which is what populates the progress info. But alas, a couple minutes went by and I could tell something was up. So I looked at the first experiment: "2a" and the first job: "experiment_1" and went to: `id_1/Run_1/output/slurm-<jobid>.out` and looked at that file. It showed me the following:

```
Content of Batsim's stderr log:
+ batsim -s tcp://localhost:40001 -p /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/platforms/platform_1490_-1_-1.xml -w /home/craig/LANL/Post
_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/workloads/1490_nodes_01-01-2018___time_1d0fea2c-8675-411e-9416-5ec079663d8d.json -e /home/craig/LANL/Post_July_2021/
Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1/2a/experiment_1/id_1/Run_1/output/expe-out/out --disable-schedule-tracing --disable-machine-state-tracing --forw
ard-profiles-on-submission --acknowledge-dynamic-jobs --enable-dynamic-jobs -q --checkpoint-batsim-signal 37 --seed-failure-machine 10 --seed-failures 10 --queue-policy OR
IGINAL-FCFS --checkpointing-on --compute_checkpointing --repair-time 172800 --SMTBF 72483.22147651007
[0.000000] [batsim/INFO] Workload 'w0' corresponds to workload file '/home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/basefiles/workloads/1490_nodes_01-01
-2018___time_1d0fea2c-8675-411e-9416-5ec079663d8d.json'.
[0.000000] [batsim/INFO] Batsim version: v4.0.3-18-g98c8fcb
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'host/model' to 'ptask_L07'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'maxmin/precision' to '0.0000000000001'
[0.000000] [xbt_cfg/INFO] Configuration change: Set 'surf/precision' to '0.0000000000001'
[0.000000] [xbt_cfg/INFO] Switching to the L07 model to handle parallel tasks.
[0.000000] ../src/jobs.cpp:931: [root/CRITICAL] Error with 15032646 checkpoint_interval is computed as negative.  This indicates a problem with the dump_time vs the (S)MTB
F
Backtrace (displayed in actor maestro):
```

The error at first sounds terrible. But I am familiar with this. When calculating the checkpoint interval, and especially an optimal checkpoint, the SMTBF needs to be known as well as the dump and read times. I forgot to add them to the workload description in the config file. Once I did that everything worked as planned.

Before I ran the command, however, I used my trusty `batEdit` command to edit the config file:

```
1   (batsim_env) user >   batEdit -e -o "-ltr"
```

This showed the following:

```
   44  -rw-r--r-- 1 craig users  1506 Jun 24 06:45 test_all_algorithms_easy_bf_fast2.config
   45  -rw-r--r-- 1 craig users  9066 Jun 30 01:32 test_all_algorithms.config
   46  -rw-r--r-- 1 craig users  1432 Jul 15 01:35 test_simulator_failure.config
   47  -rw-r--r-- 1 craig users   871 Jul 18 01:07 test_simulator_success.config
   48  -rw-r--r-- 1 craig users  1589 Jul 21 05:55 test_all_algorithms_easy_bf3.config
   49  -rw-r--r-- 1 craig users  3012 Aug  7 12:20 test_doc_resv_sweep.config
   50  -rw-r--r-- 1 craig users  1450 Aug 21 00:03 newProgress.config
   51  -rw-r--r-- 1 craig users  1488 Sep 16 17:03 test_requeue_ebf3.config
   52  -rw-r--r-- 1 craig users  1487 Sep 16 17:03 test_requeue_ebf3_orig.config
   53  -rw-r--r-- 1 craig users  1488 Sep 16 17:03 test_requeue_ebf2.config
   54  -rw-r--r-- 1 craig users  1487 Sep 16 17:04 test_requeue_ebf2_orig.config
   55  -rw-r--r-- 1 craig users  1525 Sep 21 03:06 test_requeue.config
   56  -rw-r--r-- 1 craig users  1333 Sep 23 12:34 AAE_test.config
   57  -rw-r--r-- 1 craig users  1738 Sep 26 07:41 aae_test.config
   58  -rw-r--r-- 1 craig users  1625 Oct  7 07:32 wacFile.config
   59  -rw-r--r-- 1 craig users   662 Oct 17 15:51 walkthrough_test.config
   60  -rw-r--r-- 1 craig users    19 Oct 21 14:14 new.config
   61  -rw-r--r-- 1 craig users  2066 Oct 28 03:06 walkthrough_1.config
Enter a choice (0 to exit):
```

So I wanted to edit walkthrough_1.config so I typed: ’61’ as a choice (no quotes). Scenario 2 was much different than Scenario 1 so I simply did a Ctrl+6 to mark the text, used page-down to go to the bottom of the file and hit Ctrl+k to cut the text. Then I copied and pasted what you see in this pdf file for Scenario 2’s config. Ctrl+o to save, but I made sure to save it to ’walkthrough_2.config’, Ctrl+x to exit.

One last step. We saved to a different file than we chose using batEdit. Run it again and choose our new file:

```
1   (batsim_env) user >   batEdit -o "-ltr"
```

This time we are not editing the file, we just want to choose it.

```
    49  -rw-r--r-- 1 craig users  3012 Aug  7 12:20 test_doc_resv_sweep.config
    50  -rw-r--r-- 1 craig users  1450 Aug 21 00:03 newProgress.config
    51  -rw-r--r-- 1 craig users  1488 Sep 16 17:03 test_requeue_ebf3.config
    52  -rw-r--r-- 1 craig users  1487 Sep 16 17:03 test_requeue_ebf3_orig.config
    53  -rw-r--r-- 1 craig users  1488 Sep 16 17:03 test_requeue_ebf2.config
    54  -rw-r--r-- 1 craig users  1487 Sep 16 17:04 test_requeue_ebf2_orig.config
    55  -rw-r--r-- 1 craig users  1525 Sep 21 03:06 test_requeue.config
    56  -rw-r--r-- 1 craig users  1333 Sep 23 12:34 AAE_test.config
    57  -rw-r--r-- 1 craig users  1738 Sep 26 07:41 aae_test.config
    58  -rw-r--r-- 1 craig users  1625 Oct  7 07:32 wacFile.config
    59  -rw-r--r-- 1 craig users   662 Oct 17 15:51 walkthrough_test.config
    60  -rw-r--r-- 1 craig users    19 Oct 21 14:14 new.config
    61  -rw-r--r-- 1 craig users  2066 Oct 28 03:06 walkthrough_1.config
    62  -rw-r--r-- 1 craig users  2605 Oct 28 07:23 walkthrough_2.config
Enter a choice (0 to exit): 62
```

And here is the command I used to run it.

```
(batsim_env) user >   myBatchTasks.sh -f ${file1} -o scenario_2_1 -m bare-metal -p tasks
↪  -t 10 -s 40001 -w 00:06:00
```

It is a lot like before except we added on `-w 00:06:00`. This is the wallclock time. It tells SLURM that our sbatch script should only take 6 minutes. And as you know from our `batsim_environment.sh` file, we send signal 37 at 120 seconds before the wallclock is going to be reached. So in this case, after running 4 minutes, we are going to do a real checkpoint and respawn the jobs.

4.4 While Running The Sims

4.4.1 progress.sh

So we covered a lot of basic options for `progress.sh` in Scenario 1. What is important in this Scenario is to show the resubmission elements.

So here are the sims just starting. Notice I use `progress.sh -i $(pwd) -N -e "2a"`. The `-N` option gets me most of what I want for looking at resubmitted jobs and the `-e "2a"` shows just experiment "2a" which cuts down on how much we see.

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1> progress.sh -i $(pwd) -N -e "2a"
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1
2a/e_1/i_1/r_1:    |31,932|50,000|63.86|63.86|31,932|50,000|0|0d-00:06:14
...2a/e_10/i_1/r_1:   ********************************************** Error No File **********************************************
2a/e_11/i_1/r_1:   |34,237|50,000|68.47|68.47|34,237|50,000|0|0d-00:06:25
2a/e_12/i_1/r_1:   |33,852|50,000|67.70|67.70|33,852|50,000|0|0d-00:06:25
2a/e_13/i_1/r_1:   |34,598|50,000|69.20|69.20|34,598|50,000|0|0d-00:06:25
2a/e_14/i_1/r_1:   |34,210|50,000|68.42|68.42|34,210|50,000|0|0d-00:06:25
2a/e_15/i_1/r_1:   |33,784|50,000|67.57|67.57|33,784|50,000|0|0d-00:06:25
2a/e_16/i_1/r_1:   |33,555|50,000|67.11|67.11|33,555|50,000|0|0d-00:06:25
2a/e_17/i_1/r_1:   |34,784|50,000|69.57|69.57|34,784|50,000|0|0d-00:06:25
2a/e_18/i_1/r_1:   |34,895|50,000|69.79|69.79|34,895|50,000|0|0d-00:06:25
...2a/e_19/i_1/r_1:   ********************************************** Error No File **********************************************
2a/e_2/i_1/r_1:    |31,685|50,000|63.37|63.37|31,685|50,000|0|0d-00:06:14
...2a/e_20/i_1/r_1:   ********************************************** Error No File **********************************************
2a/e_3/i_1/r_1:    |31,151|50,000|62.30|62.30|31,151|50,000|0|0d-00:06:14
2a/e_4/i_1/r_1:    |31,172|50,000|62.34|62.34|31,172|50,000|0|0d-00:06:14
2a/e_5/i_1/r_1:    |31,768|50,000|63.54|63.54|31,768|50,000|0|0d-00:06:14
2a/e_6/i_1/r_1:    |31,540|50,000|63.08|63.08|31,540|50,000|0|0d-00:06:14
2a/e_7/i_1/r_1:    |30,973|50,000|61.95|61.95|30,973|50,000|0|0d-00:06:14
2a/e_8/i_1/r_1:    |30,889|50,000|61.78|61.78|30,889|50,000|0|0d-00:06:14
...2a/e_9/i_1/r_1:   ********************************************** Error No File **********************************************

Format: |comp_jobs|jobs|%_done|all_%|tot_comp|o_jobs|nb_re|el_time
```

Some things to mention: The completed jobs and the total completed are the same number in resubmit 0. That red column is the resubmit number. The jobs and original jobs are the same number as well, and the percent done and overall percent done, they are all the same numbers.

So lets look a little later after the jobs have respawned and have been queued in SLURM and have now moved into the RUNNING state.

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1> progress.sh -i $(pwd) -N -e "2a"
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1
2a/e_1/i_1/r_1:    |22,619|24,081|93.93|97.08|48,538|50,000|1|0d-00:03:19
2a/e_10/i_1/r_1:    |29,128|50,000|58.26|58.26|29,128|50,000|0|0d-00:04:49
2a/e_11/i_1/r_1:    |29,419|50,000|58.84|58.84|29,419|50,000|0|0d-00:04:49
2a/e_12/i_1/r_1:    |29,253|50,000|58.51|58.51|29,253|50,000|0|0d-00:04:49
2a/e_13/i_1/r_1:    |30,302|50,000|60.60|60.60|30,302|50,000|0|0d-00:04:48
2a/e_14/i_1/r_1:    |29,701|50,000|59.40|59.40|29,701|50,000|0|0d-00:04:48
2a/e_15/i_1/r_1:    |29,513|50,000|59.03|59.03|29,513|50,000|0|0d-00:04:48
2a/e_16/i_1/r_1:    |28,801|50,000|57.60|57.60|28,801|50,000|0|0d-00:04:48
2a/e_17/i_1/r_1:    |27,796|50,000|55.59|55.59|27,796|50,000|0|0d-00:04:48
2a/e_18/i_1/r_1:    |27,241|50,000|54.48|54.48|27,241|50,000|0|0d-00:04:48
2a/e_19/i_1/r_1:    |26,601|50,000|53.20|53.20|26,601|50,000|0|0d-00:04:48
2a/e_2/i_1/r_1:    |22,381|24,506|91.33|95.75|47,875|50,000|1|0d-00:03:19
2a/e_20/i_1/r_1:    |26,665|50,000|53.33|53.33|26,665|50,000|0|0d-00:04:48
2a/e_3/i_1/r_1:    |21,571|24,703|87.32|93.74|46,868|50,000|1|0d-00:03:19
2a/e_4/i_1/r_1:    |22,210|25,109|88.45|94.20|47,101|50,000|1|0d-00:03:19
2a/e_5/i_1/r_1:    |22,752|23,966|94.93|97.57|48,786|50,000|1|0d-00:03:19
2a/e_6/i_1/r_1:    |22,325|24,751|90.20|95.15|47,574|50,000|1|0d-00:03:19
2a/e_7/i_1/r_1:    |22,116|24,881|88.89|94.47|47,235|50,000|1|0d-00:03:19
2a/e_8/i_1/r_1:    |21,959|25,066|87.60|93.79|46,893|50,000|1|0d-00:03:19
2a/e_9/i_1/r_1:    |30,105|50,000|60.21|60.21|30,105|50,000|0|0d-00:04:49

Format: |comp_jobs|jobs|%_done|all_%|tot_comp|o_jobs|nb_re|el_time
```

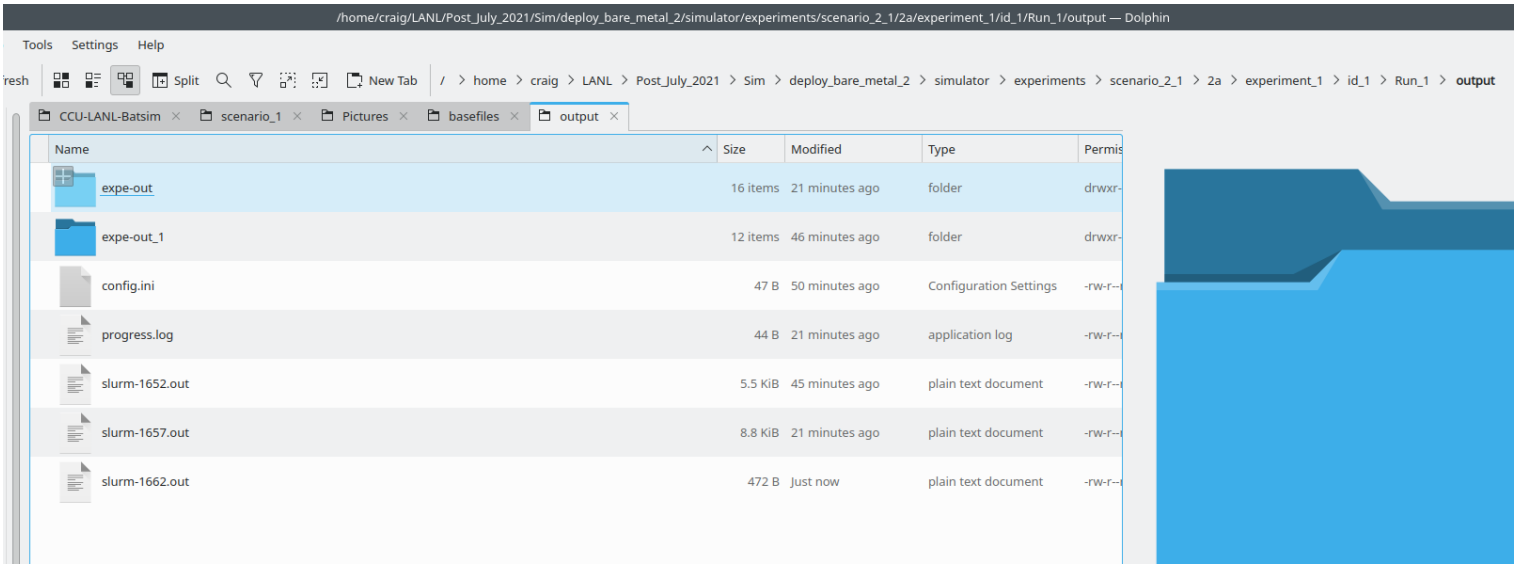
Now you see a difference for the jobs that have a nb_resubmit of 1. The overall percent is different than the %_done. The jobs is different than the original_jobs and the completed_jobs is different than total_completed.

And finally you can see that all of them are completed:

```
(batsim_env) craig: ~/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1> progress.sh -i $(pwd) -N -e "2a"
folder: /home/craig/LANL/Post_July_2021/Sim/deploy_bare_metal_2/simulator/experiments/scenario_2_1
2a/e_1/i_1/r_1:    |24,081|24,081|100.00|100.00|50,000|50,000|1|0d-00:03:38
2a/e_10/i_1/r_1:    |25,019|25,019|100.00|100.00|50,000|50,000|1|0d-00:03:28
2a/e_11/i_1/r_1:    |24,698|24,698|100.00|100.00|50,000|50,000|1|0d-00:03:33
2a/e_12/i_1/r_1:    |24,922|24,922|100.00|100.00|50,000|50,000|1|0d-00:03:38
2a/e_13/i_1/r_1:    |23,937|23,937|100.00|100.00|50,000|50,000|1|0d-00:03:08
2a/e_14/i_1/r_1:    |24,498|24,498|100.00|100.00|50,000|50,000|1|0d-00:03:28
2a/e_15/i_1/r_1:    |24,493|24,493|100.00|100.00|50,000|50,000|1|0d-00:03:21
2a/e_16/i_1/r_1:    |25,267|25,267|100.00|100.00|50,000|50,000|1|0d-00:03:46
2a/e_17/i_1/r_1:    |25,999|25,999|100.00|100.00|50,000|50,000|1|0d-00:03:53
2a/e_18/i_1/r_1:    |26,488|26,488|100.00|100.00|50,000|50,000|1|0d-00:04:19
2a/e_19/i_1/r_1:    |27,073|27,073|100.00|100.00|50,000|50,000|1|0d-00:04:29
2a/e_2/i_1/r_1:    |24,506|24,506|100.00|100.00|50,000|50,000|1|0d-00:03:47
2a/e_20/i_1/r_1:    |27,001|27,001|100.00|100.00|50,000|50,000|1|0d-00:04:32
2a/e_3/i_1/r_1:    |24,703|24,703|100.00|100.00|50,000|50,000|1|0d-00:03:59
2a/e_4/i_1/r_1:    |25,109|25,109|100.00|100.00|50,000|50,000|1|0d-00:03:56
2a/e_5/i_1/r_1:    |23,966|23,966|100.00|100.00|50,000|50,000|1|0d-00:03:34
2a/e_6/i_1/r_1:    |24,751|24,751|100.00|100.00|50,000|50,000|1|0d-00:03:50
2a/e_7/i_1/r_1:    |24,881|24,881|100.00|100.00|50,000|50,000|1|0d-00:03:54
2a/e_8/i_1/r_1:    |25,066|25,066|100.00|100.00|50,000|50,000|1|0d-00:03:58
2a/e_9/i_1/r_1:    |24,159|24,159|100.00|100.00|50,000|50,000|1|0d-00:03:12

Format: |comp_jobs|jobs|%_done|all_%|tot_comp|o_jobs|nb_re|el_time
```

4.4.2 Checkpointing Folders

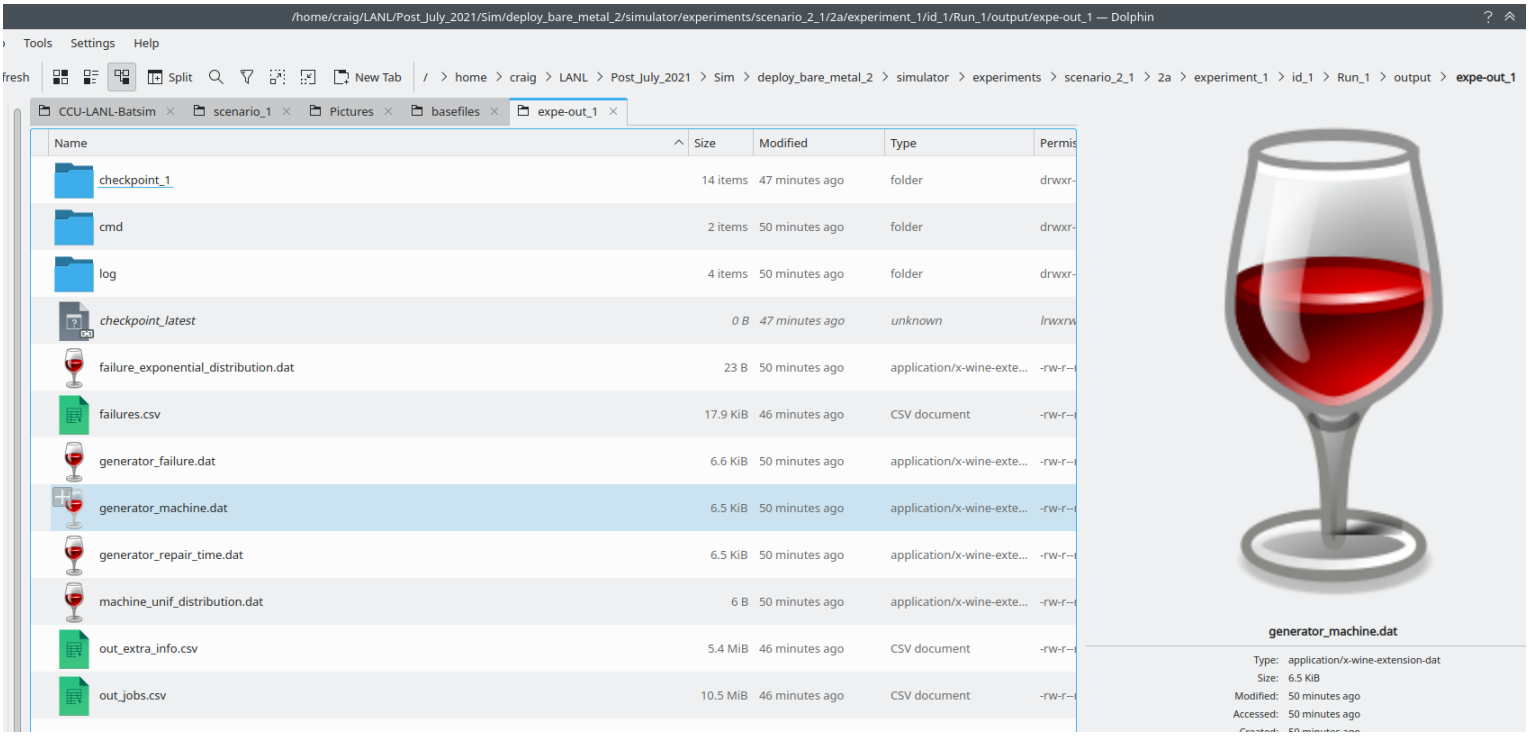
Now that everything is finished, you can see the folder structure that has been left:



Here you can see that since we kept a max of 2 'Frames' and there was only 1 resubmit, that there is only one `.../output/expe-out_#` and that folder is `.../output/expe-out_1`.

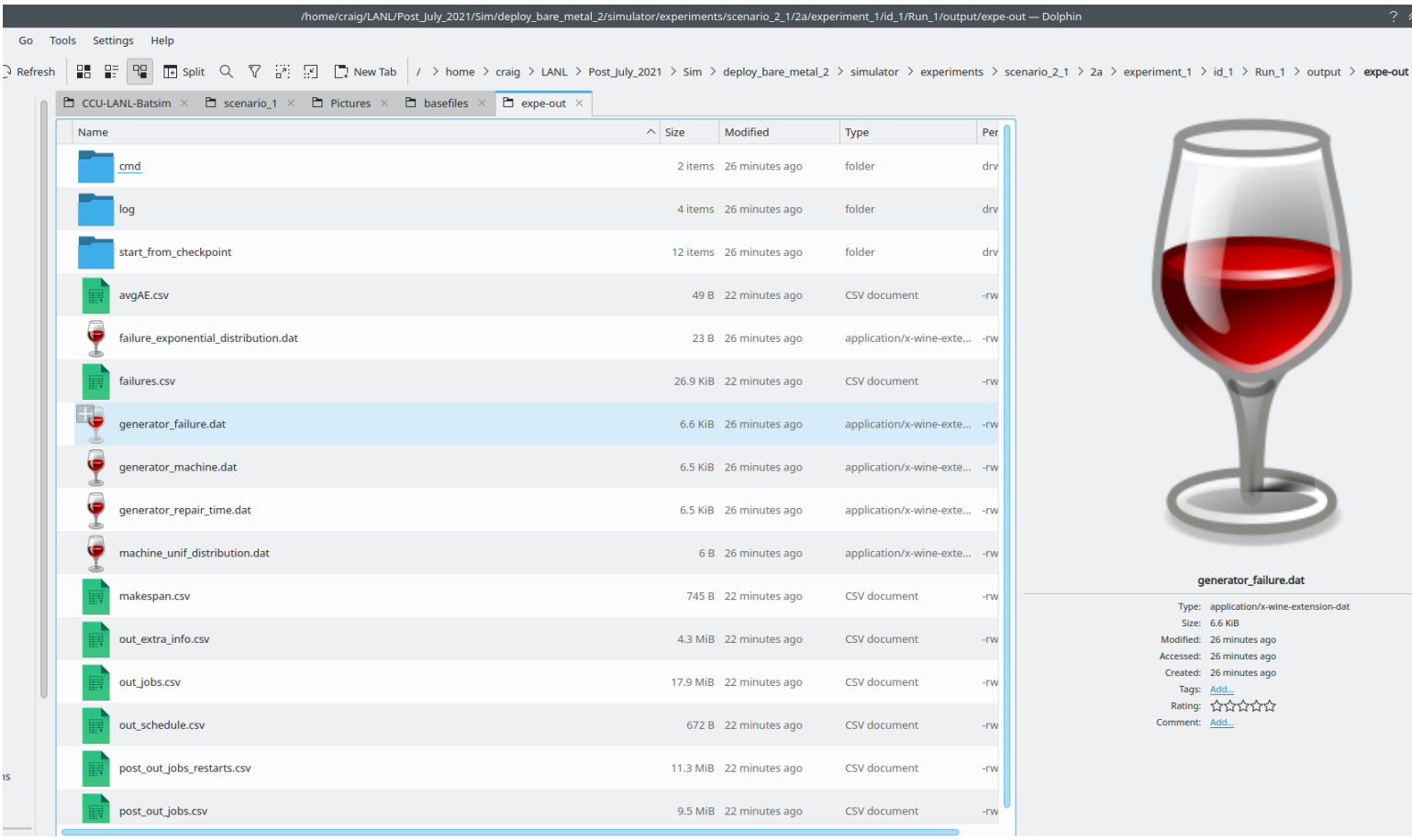
If there had been more resubmits, there would be an `.../output/expe-out_2`, but since we only said to keep 2, that would be the highest the 'Frames' would go to.

Let's look into `expe-out_1`



We can see that a single checkpoint did occur in this older 'Frame'.

Now lets go back up a folder and peek into `.../output/expe-out/`



You can see here that this 'Frame' started from a checkpoint given the folder: `.../output/expe-out/start_from_checkpoint`.

You also DON'T see a `checkpoint_1` folder since it didn't need to checkpoint. You do see a `makespan.csv` file, which tells us everything finished on this sim.

4.5 Aggregating Results

So now we aggregate the results like normal:

```
(batsim_env) user >  cd ${prefix}/experiments/scenario_2_1
(batsim_env) user >  aggregate_makespan.py -i $(pwd)
```

4.6 Analysis

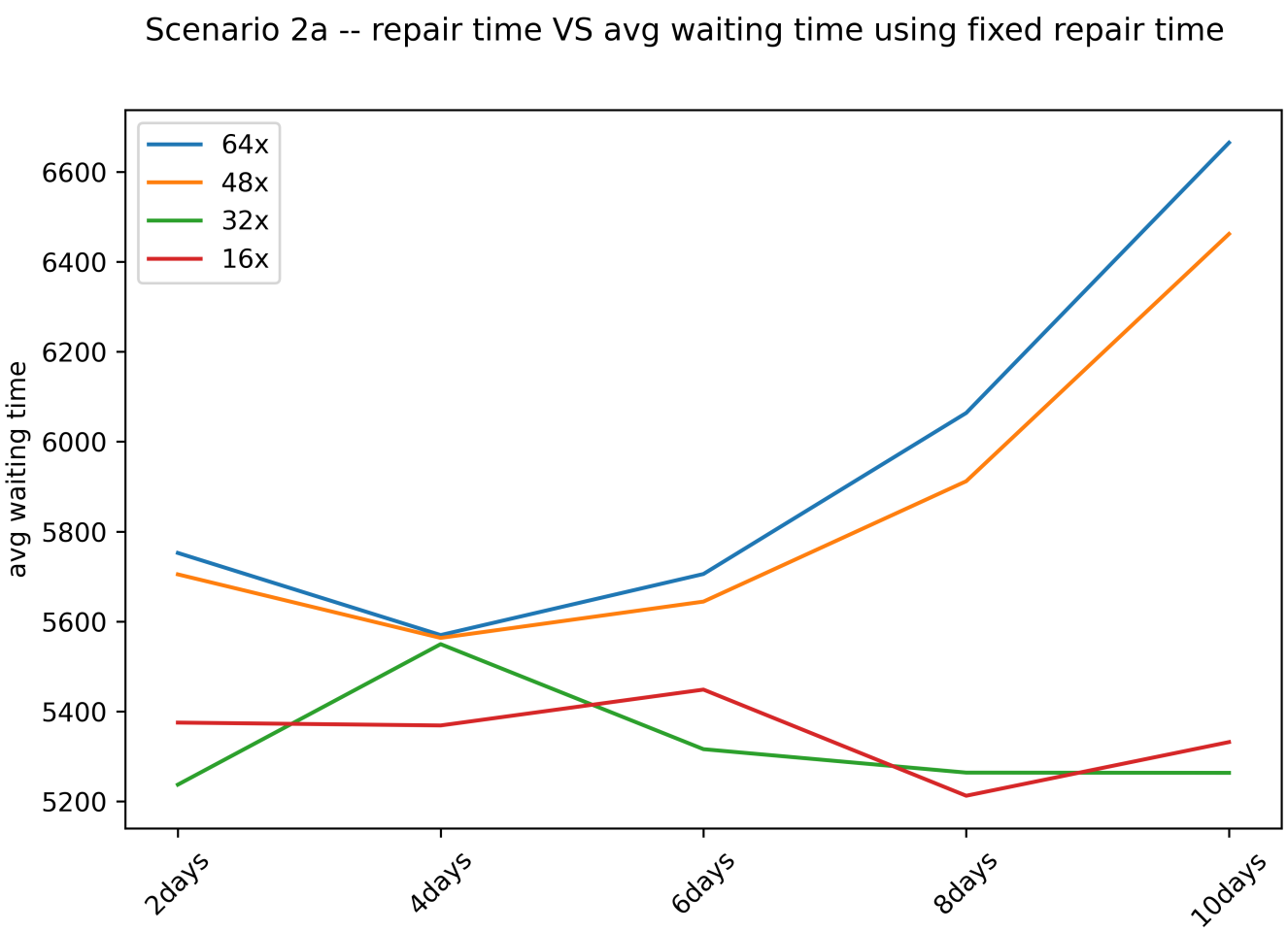
Currently `total_makespan.csv` has the following header:

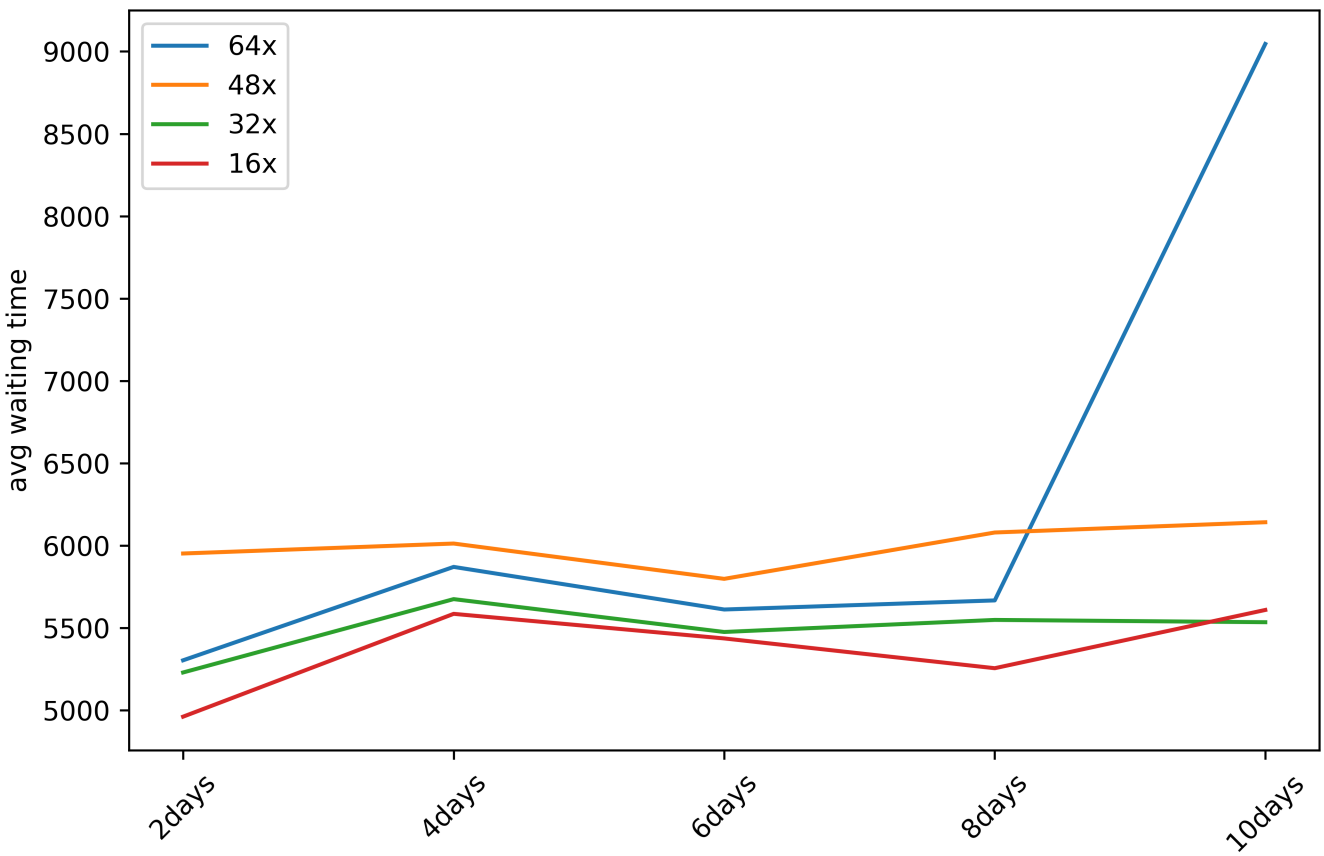
nodes	SMTBF	NMTBF	fixed-failures	repair-time	MTTR
makespan_sec	makespan_dhms	AAE	avg_tat	avg_tat_dhms	avg_waiting
avg_wait-ing_dhms	avg_pp_slow-down	avg-pp-slowdown-tau	number_of_jobs	submission_time	submission_com-pression
avg_utilization	SMTBF_failures	MTBF_failures	Fixed_failures	re-jected_not_enough_avail-able_resources	jitter
avg-pp-slowdown	avg-pp-slowdown_dhms	avg-pp-slowdown-Tau	id	job	exp

We can then make the following code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

width=5
height=5
BASE_SECONDS = 1728000000

fig, axs = plt.subplots(1, 1, figsize=(width, height))

df = pd.read_csv("./total_makespan.csv",header=0,sep=",")
#change NMTBF to be normalized to BASE_SECONDS.  Example 1x,2x,...,32x
df.loc[:,"NMTBF"]=BASE_SECONDS / df["NMTBF"]

#get a list of failure rates and sort in reverse
NMTBF_times = list(df["NMTBF"].unique())
NMTBF_times.sort(reverse=True)

Adf=df.loc[df["exp"]=="2a"]
Bdf=df.loc[df["exp"]=="2b"]


current_df=Adf
for nmtbf in NMTBF_times:
    nmtbf_df=current_df.loc[current_df["NMTBF"]==nmtbf]
    plt.plot(nmtbf_df["repair-time"],nmtbf_df["avg_waiting"],label=f"{int(nmtbf)}x")
plt.legend()
axs.set_xlabel("repair time(days)")
axs.set_ylabel("avg waiting time")
ticks=list(range(3600*24*2,3600*24*12,3600*24*2))
labels=[f"{i}days" for i in range(2,12,2)]
axs.set_xticks(ticks)
axs.set_xticklabels(labels,rotation=45)
fig.suptitle("Scenario 2a -- repair time VS avg waiting time using fixed repair time")
fig.savefig("myplot.png",dpi=600)

fig, axs = plt.subplots(1, 1, figsize=(width, height))
current_df=Bdf
for nmtbf in NMTBF_times:
    nmtbf_df=current_df.loc[current_df["NMTBF"]==nmtbf]
    plt.plot(nmtbf_df["MTTR"],nmtbf_df["avg_waiting"],label=f"{int(nmtbf)}x")
plt.legend()
axs.set_xlabel("repair time(days)")
axs.set_ylabel("avg waiting time")
ticks=list(range(3600*24*2,3600*24*12,3600*24*2))
labels=[f"{i}days" for i in range(2,12,2)]
axs.set_xticks(ticks)
axs.set_xticklabels(labels,rotation=45)
fig.suptitle("Scenario 2b -- repair time VS avg waiting time using MTTR")
fig.savefig("myplot2.png",dpi=600)
```

And here are the results:





Now these graphs may not seem like much at this point. With the variability in when failures happen and what machines they occur on, we really need statistical information here in the form of a 'Monte Carlo' set of Runs. Here are the same graphs with 400 runs each.

5 Scenario 3 - Conservative Backfilling and Reservations

5.1 Scenario Description

In this scenario we are going to use the conservative_bf algorithm. It schedules everything in the queue. This makes it easier to add placeholders into the schedule for ”reservations” or DST’s. Unfortunately the algorithm can be quite slow.

What we want to show is using reservations and some of the analysis you can do on it.

We will use:

- reservations
- grizzly workload
- output bins

The idea in this scenario is that we have a cluster the size of grizzly at 1490 nodes. If we do weekly maintenance on the cluster, is it better to bring the whole cluster down for some time or break parts of the cluster up through the week to do maintenance? If it is better to break the cluster up, how many partitions should we do? Lastly, we can evaluate how this impacts big jobs vs small jobs.

Scenario 3

- Workload:
  - grizzly 2018
  - wallclock limit: 101% of runtime
  - from Feb 1, 2018 to April 1, 2018
- Reservations:
  - 8 hours long reservations
  - repeated every 7 days
  - submitted at the start
  - Subdivisions:1,2,4,8
  - Subdivisions unit: 1 week

5.2 Make The Config File

If you need a refresher on the format of a config file Scenario 1 covered that.

Here we show what the config file we used looks like:

```
{
  "3a":{
          "input": {

                  "node-sweep":{
                      "range":[1490]

                  },


```

```
11                      "reservation-sweep":{
12                              "name":"resv1",
13
14                              "reservations-array":[
15                                      {
16
17                                              "time":"[00]:[480]:[00]",
18                                              "subdivisions":"[1]",
19                                              "subdivisions-unit":"[7]days [00]:[00]:[00]",
20
21                                              "{1} subdivisions":"[2,4,8]"
22                                      }
23
24                              ]
25                      },
26                      "reservations-resv1":{
27
28                            "reservations-array":[
29                                {
30                                  "type":"parallel_homogeneous",
31                                  "machines":{
32                                      "prefix":"a",
33                                      "machine-speed":1,
34                                      "total-resources":"0-1489",
35                                      "interval":"0-1489"
36                                  },
37                                  "repeat-every":"7days 00:00:00",
38                                  "time":"09:00:00",
39                                  "start":"7days 12:00:00",
40                                  "submit":-1,
41                                  "count":10
42                                }
43                             ]
44                      },
45                          "batsched-policy":"conservative_bf",
46                          "queue-depth":1000,
47                          "grizzly-workload":{
48                                              "type":"parallel_homogeneous",
49                                              "machine-speed":1,
50                                              "reservations":"resv1",
51
52
53                                              "input":"sanitized_jobs.csv",
54                                              "wallclock-limit":"101%",
55                                              "time":"02-01-2018:04-01-2018"
56
57                                          }
58                      },
59                      "output": {
60                          "avg-makespan":1,
61                          "bins":"[0,2,4,8,16,32,64,128,256,512,1024,+]"
62
63                      }
64
65              }
66      }
```

3a

So on line 5 we have our node sweep, but we just use the 1490 nodes that grizzly has since we are using the grizzly workload. Next we have the reservation sweep, but before we get into that, let's look at the reservations that the sweep acts on. Line 26 starts the reservations, and we have called it 'resv1'. Every `"reservations-<name>"` section has an array of reservation types. In this

example we just use one type of reservation, so we only define one type within the array.

We start on line 30 with the profile type `"type":"parallel_homogeneous"`. The thing about using `"type":"parallel_homogeneous"` is that the reservation could speed up or slow down depending on if the flops/sec sped up on the machines or slowed down. If we used `"type":"delay"` then it would be a strictly time sensitive reservation.

Next, on line 31 we define the machines this reservation type affects. On line 34 we say that the total resources available for this reservation type is the whole cluster. Defining this can be useful if we are doing some random machine intervals. But in this example we are just doing a simple interval of the whole cluster.

On line 39 we set when we want this reservation type to start. So at 7 days and 12 hours after the first job, we have a reservation. The duration of that reservation is set for 9 hours on line 38. On line 37 we tell it to repeat the reservation every 7 days. On line 40 we say we want all of the reservations to submit at the start. This means the reservation is not going to come in and kill jobs because the reservation is already there and nothing should have been scheduled on it to begin with. There may be cases when you want to say how far in advance a reservation is scheduled before its start time.

On line 41 we have a clunky property named `"count"`. This requires you to keep up with the amount of simulated time your simulation takes, as well as how many reservations you need to have to last that simulated time. I am doing 2 months of jobs and the reservations happen every week. There are about 4 weeks in a month so that would mean I need at least 8 reservations. I give it 2 extra just to make sure. There really isn't too much problem with going over. It can inflate your makespan, however, if you were using that as a metric. The reason we need this property is in how reservations are made. They are part of the workload and operate much like a job, so they are made in advance, and if you are using things like failures, there isn't much way of knowing how long a simulation is going to run. So you must use your best judgement.

So now that we described the reservation type, let's go over the sweep of it. On line 12 we set the name of the type that this sweep applies to. We then make an array of sweeps. Each sweep applies to the corresponding type in the resv1 reservations-array definition.

Ok, so our base modifications are done on lines 17-19. On line 17 we set just one duration of 8 hours (480 minutes). This will overwrite the 9 hours we had set. So setting it in our type definition was not needed. On line 18 we start with 1 subdivision. That subdivision is going to be spread over 7 days. Since we start with 1 subdivision the unit of time is moot in that case.

Then on line 21 we add in a multiplier. It is going to use the 480 minute duration and the 7 day unit, but it is going to change the subdivisions to 2,4,and 8. This is going to result in 3 more jobs and therefore 3 more simulations for a total of 4.

On line 45 we tell it we want the `"conservative_bf"` algorithm. On line 46 we use a queue-depth of 1,000. This will prevent the algorithm from taking so long, however, it can cut back on accuracy if what you wanted to simulate was a scheduler that scheduled everything in the queue in advance. In our scenario we tell it to quit after taking the first 1,000 jobs in the queue.

You've seen the grizzly workload before, but on line 50 we set which reservations we are using for this workload. Also, on line 55 we set the time from February 1 until April 1, so two months.

Everything else is the same except line 61. Here we tell our post-processing that we not only want a makespan.csv for the total summary of our simulation, but we also want a summary for the specified bins. The bins are operating on the amount of nodes the jobs requested. So we will have a summary for jobs that requested (0-2] nodes, a summary for (2-4] nodes etc... until we get to the '+'. This says 'anything above' the previous number: 1024. You can use a '-' at the beginning to specify 'below the next value', as well.

The binned summaries will be in your `expe-out/bins` folder.

5.3 Run The Config File

After setting $file1 there isn't anything different in this command except the output folder:

```
1  (batsim_env) user >  myBatchTasks.sh -f ${file1} -o scenario_3_3_bins -m bare-metal -p
   ↪  tasks -t 8 -s 40001
```

5.4 While Running The Sims

You can go ahead and use the `progress.sh` tool as you see fit. You will notice that you can see the 'schedule' components in this scenario. This is because conservative_bf uses the schedule class. You may also notice the speed of jobs completing can go up and down based on the queue size.

5.5 Aggregating Results

So now we can aggregate the results as normal.

```
(batsim_env) user >   cd ${prefix}/experiments/scenario_3_3_bins
(batsim_env) user >   aggregate_makespan.py -i $(pwd)
```

5.6 Analysis

Currently `total_makespan.csv` has the following header:

nodes	SMTBF	NMTBF	fixed-failures	repair-time	MTTR
makespan_sec	makespan_dhms	AAE	avg_tat	avg_tat_dhms	avg_waiting
avg_waiting_dhms	avg_pp_slowdown	avg-pp-slowdown-tau	number_of_jobs	submission_time	submission_compression
avg_utilization	SMTBF_failures	MTBF_failures	Fixed_failures	rejected_not_enough_available_resources	jitter
avg-pp-slowdown	avg-pp-slowdown_dhms	avg-pp-slowdown-Tau	id	job	exp

and the individual binned makespan csv files in the `expe-out/bins` folder have the following header:

nodes	SMTBF	NMTBF	fixed-failures	repair-time	MTTR
makespan_sec	makespan_dhms	AAE	avg_tat	avg_tat_dhms	avg_waiting
avg_waiting_dhms	avg_pp_slowdown	avg-pp-slowdown-tau	number_of_jobs	submission_time	submission_compression
avg_utilization	SMTBF_failures	MTBF_failures	Fixed_failures	rejected_not_enough_available_resources	jitter
avg-pp-slowdown	avg-pp-slowdown_dhms	avg-pp-slowdown-Tau			

both of which allow for this code:

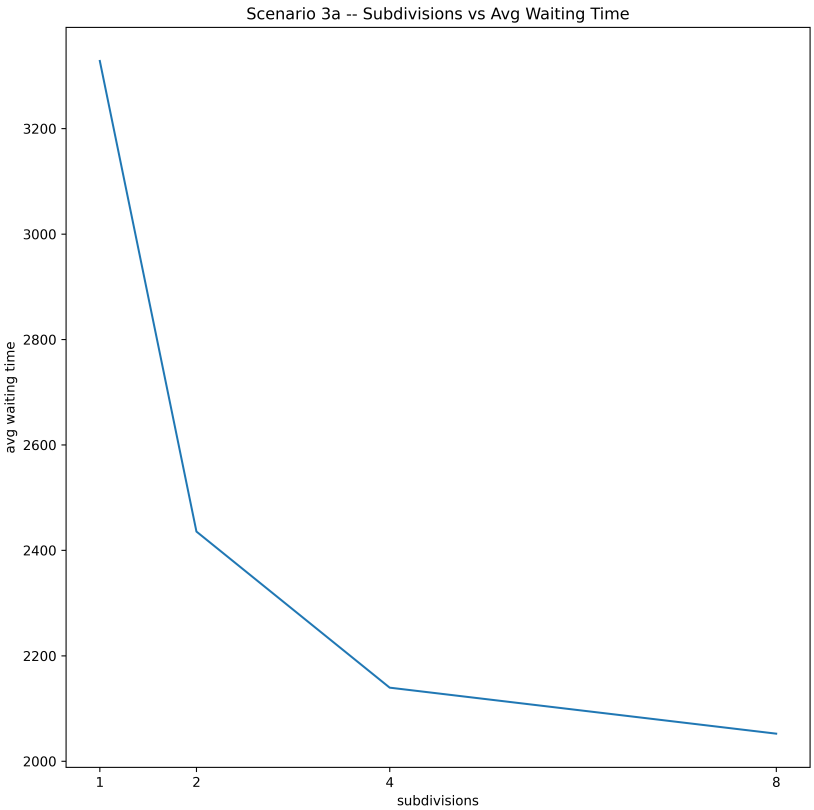
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pathlib
import os

# a function that is in the .../basefiles/functions.py module
# it takes a project path, a function that you want to apply to each simulation,
# and the starting arguments that will be passed to the passed function
```

```
    def traverse_project_folder(path,myFunc,inputArgs):
        experiments=[i for i in os.listdir(path) if os.path.isdir(path+"/"+i)]
        for exp in experiments:
            jobs=[i for i in os.listdir(f"{path}/{exp}") if \
              os.path.isdir(f"{path}/{exp}/{i}")]
            for job in jobs:
                ids=[i for i in os.listdir(f"{path}/{exp}/{job}") if \
                  os.path.isdir(f"{path}/{exp}/{job}/{i}")]
                for ourId in ids:
                    runs=[i for i in os.listdir(f"{path}/{exp}/{job}/{ourId}") if \
                      os.path.isdir(f"{path}/{exp}/{job}/{ourId}/{i}")]
                    for run in runs:
                        current_run=f"{path}/{exp}/{job}/{ourId}/{run}"
                        pathArgs={"experiments":experiments,"jobs":jobs,"ids":ids,\
                          "runs":runs,"path":path,"exp":exp,"job":job,"ourId":ourId,\
                          "run":run,"current_run":current_run}
                        inputArgs=myFunc(pathArgs,inputArgs)

        return inputArgs

    # this is the function that I want to be run on each sim folder
    # it requires that pathArgs and inputArgs be passed into it

    # this particular function will gather all the avg_waiting times from each binned
    # makespan summary.
    def myFunc(pathArgs,inputArgs):
        if inputArgs["first"]==True:
            bin_files=[i for i in os.listdir(f"{pathArgs['current_run']}/output/expe-out/bins") \
              if os.path.splitext(i)[1] == ".csv"]
            bins=[os.path.splitext(i)[0].split("_")[1:] for i in bin_files]
            starts=[]
            ends=[]
            avg_waitings={}
            avg_waiting=[]
            for ourBin in bins:
                starts.append(ourBin[0])
                ends.append(ourBin[1])
                file=f"{ourBin[0]}_{ourBin[1]}.csv"
                df = pd.read_csv(\
                  f"{pathArgs['current_run']}/output/expe-out/bins/makespan_{file}",\
                  sep=",",header=0)
                avg_waiting=df["avg_waiting"].values[0]
                avg_waitings[f"{ourBin[0]}_{ourBin[1]}"]=[avg_waiting]
        else:
            bin_files=inputArgs["bin_files"]
            bins=inputArgs["bins"]
            starts=inputArgs["starts"]
            ends=inputArgs["ends"]
            avg_waitings=inputArgs["avg_waitings"]
            for ourBin in inputArgs["bins"]:
                file=f"{ourBin[0]}_{ourBin[1]}.csv"
                df = pd.read_csv(\
                  f"{pathArgs['current_run']}/output/expe-out/bins/makespan_{file}",\
                  sep=",",header=0)
                avg_waiting=df["avg_waiting"].values[0]
                avg_waitings[f"{ourBin[0]}_{ourBin[1]}"].append(avg_waiting)
        inputArgs={"bin_files":bin_files,"bins":bins,"starts":starts,"ends":ends,\
         "avg_waitings":avg_waitings,"first":False}
        return inputArgs



    scriptPath = str(pathlib.Path(__file__).parent.absolute())
```

```
    width=10
    height=10
    BASE_SECONDS = 1728000000  #not used in this scenario

    fig, axs = plt.subplots(1, 1, figsize=(width, height))

    df = pd.read_csv("./total_makespan.csv",header=0,sep=",")

    #1 subdivision  is 'job':experiment_1
    #2 subdivisions is 'job':experiment_2
    #4 subdivisions is 'job':experiment_3
    #8 subdivisions is 'job':experiment_4

    # First we graph an overall summary of the amount of Subdivisions vs
    # the avg waiting time

    # unfortunately much of the info of the reservations used is not in any summary
    # you can find the reservation info used for this sim in the Run_#/input/config.ini file

    # we get the experiment num and use that to determine how many subdivisions this sim had
    experiments=df.job.str.extract(r'experiment_(?P<experiment>\d+)')
    experiments["experiment"]=experiments.experiment.astype(int)
    df = pd.concat([df,experiments],axis=1)
    df["subdivisions"]=2**(df["experiment"]-1)

    # we plot the subdivisions vs avg_waiting
    plt.plot(df["subdivisions"],df["avg_waiting"])
    axs.set_xlabel("subdivisions")
    axs.set_ylabel("avg waiting time")
    axs.set_xticks([1,2,4,8])

    axs.set_title("Scenario 3a -- Subdivisions vs Avg Waiting Time")
    fig.savefig("myplot.png",dpi=600)


    # we are then going to graph the binned avg waiting times

    fig, axs = plt.subplots(1, 1, figsize=(width, height))

    # we get the hsv color map
    cmap = plt.get_cmap('hsv')
    # we then evenly space the colors out.  I don't want the redish ones at the end
    # of the color map looking like the red ones at the beginning of the color map
    # so I do a linspace of 0 to only .75.  I want 11 colors in that range for the
    # 11 bins.
    colors = cmap(np.linspace(0,.75,11))

    # I traverse the project folder and apply my 'myFunc' function
    inputArgs={"first":True}
    inputArgs = traverse_project_folder(scriptPath,myFunc,inputArgs)

    # I plot the graphs each with a different color
    count=0
    for ourBin in inputArgs["bins"]:
        avg_waiting=inputArgs["avg_waitings"][f"{ourBin[0]}_{ourBin[1]}"]
        plt.plot([1,2,4,8],avg_waiting,color=colors[count],label=f"({ourBin[0]},{ourBin[1]})")
        count+=1

    plt.legend(loc="lower right",bbox_to_anchor=(1.2,0.0))
    axs.set_xlabel("subdivisions")
    axs.set_ylabel("avg waiting time")

    axs.set_title("Scenario 3a -- Subdivisions vs Avg Waiting Time By Bin")
    axs.set_xticks([1,2,4,8])
```

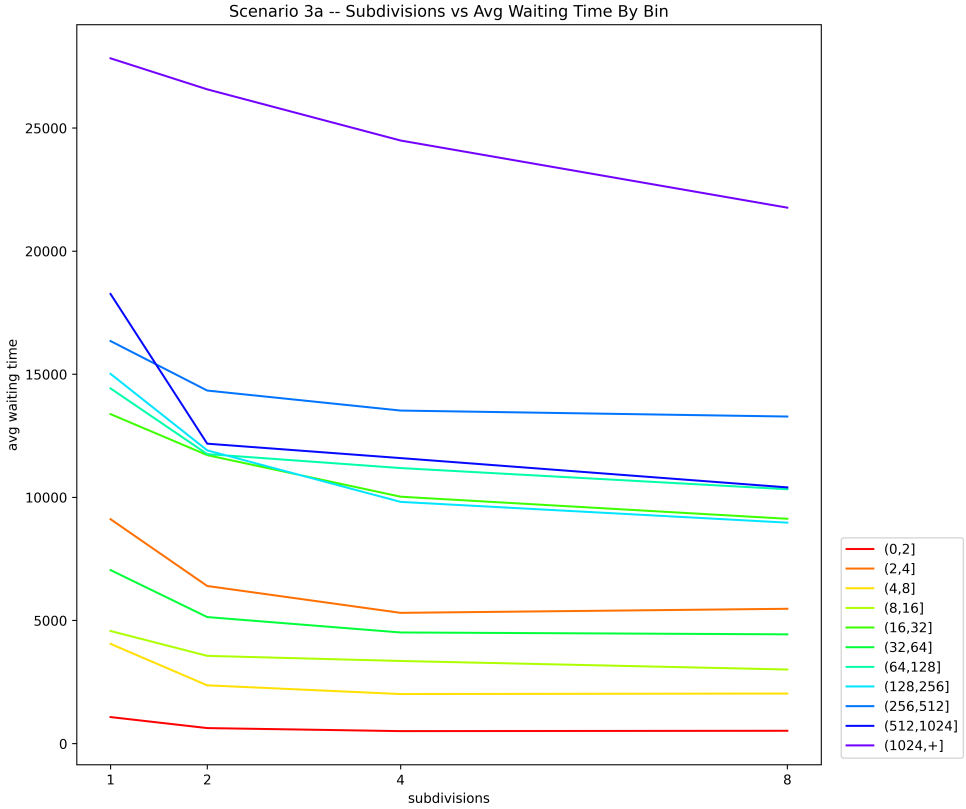
```
    fig.savefig("myplot2.png",dpi=600, bbox_inches='tight')



  # Now I want to normalize the plots.  Each line is normalized to where
  # they started at 1 subdivision.


    fig, axs = plt.subplots(1, 1, figsize=(width, height))
    count=0
    for ourBin in inputArgs["bins"]:
        avg_waiting=inputArgs["avg_waitings"][f"{ourBin[0]}_{ourBin[1]}"]

        # here is where things get normalized
        avg_waiting=avg_waiting/avg_waiting[0]

        plt.plot([1,2,4,8],avg_waiting,color=colors[count],label=f"({ourBin[0]},{ourBin[1]}]")
        count+=1

    plt.legend(loc="lower right",bbox_to_anchor=(1.2,0.0))
    axs.set_xlabel("subdivisions")
    axs.set_ylabel("avg waiting time normalized")

    axs.set_title("Scenario 3a -- Subdivisions vs Avg Waiting Time By Bin Normalized")
    axs.set_xticks([1,2,4,8])
    fig.savefig("myplot3.png",dpi=600, bbox_inches='tight')
```

And here are the results:

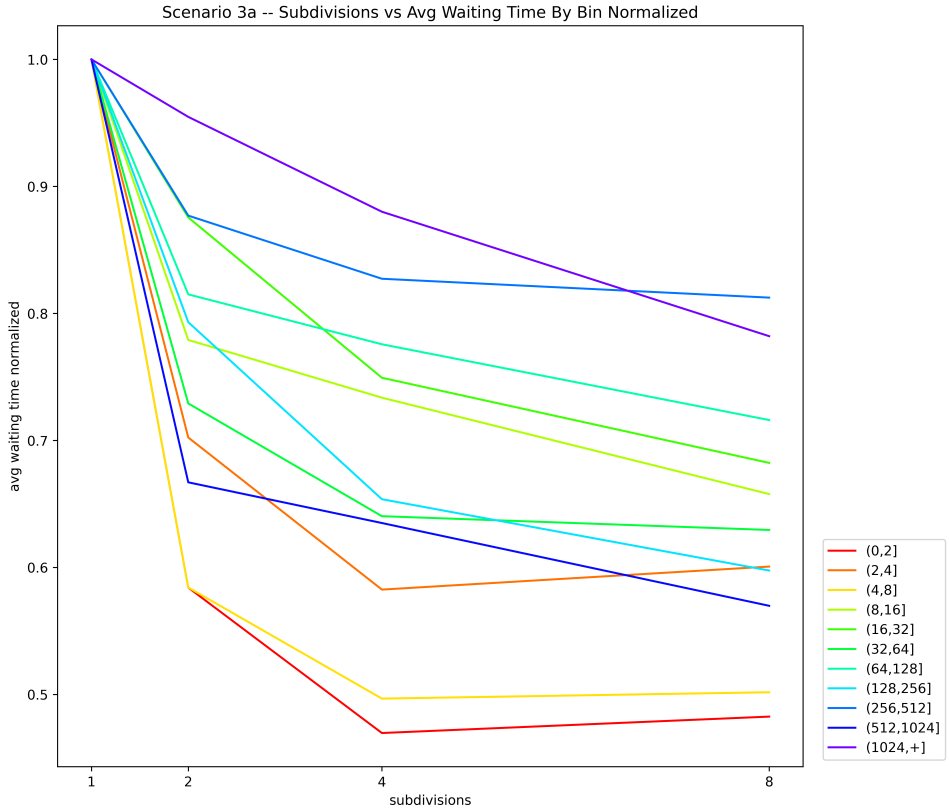


As you can see, there is a tremendous benefit to splitting the cluster up into 8 subdivisions, but the trend is that the improvement is diminishing. Breaking it up into 16 probably isn't so advantageous.

Keep in mind, this is a simulation with very specific dials set. Things would change a bit with a different reservation duration, subdivision unit time, etc...



Here it looks like the largest jobs do the best with breaking the cluster up and the smallest jobs don't see much improvement at all.



Here you can see that, actually, small jobs see a huge improvement in avg waiting time when going from 1 subdivision to 2 and to 4. You can also see that compared to not partitioning the maintenance up at all, a subdivision of 8 can have about a 20% difference in average waiting time for the largest jobs.