CCU-LANL Batsim Developer Doc Manual

Craig Walker

November 11, 2024

Contents

i. Preface

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

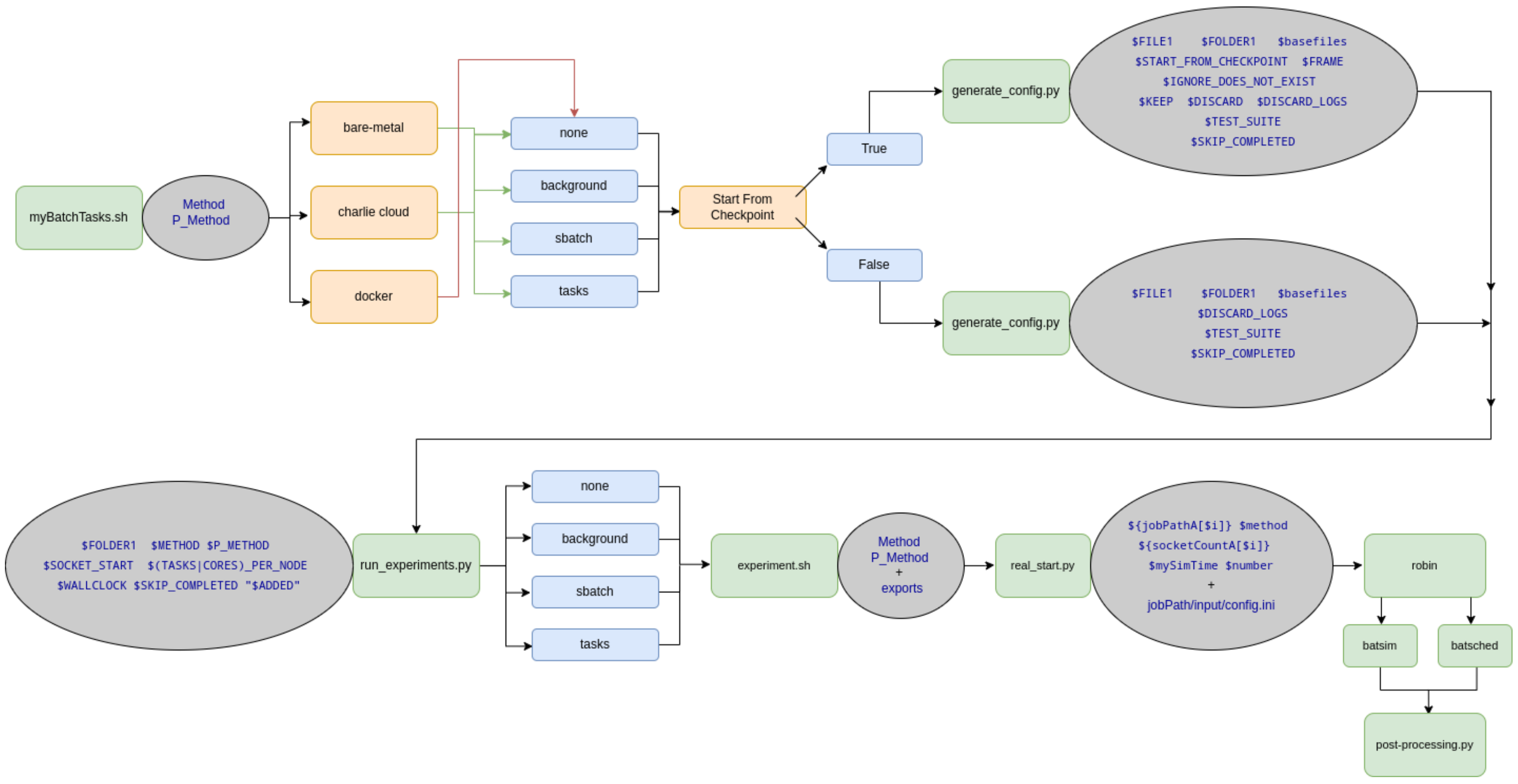
Batsim uses a library called Simgrid under the hood. Simgrid is responsible for making simulated nodes and connections and for all the simulated calculations on them and between them. Batsim is able to sit on top of it using an api called s4u that Simgrid provides. Batsim doesn't use all of Simgrid's functionality, but it adds quite a bit of its own functionality to the cluster/scheduler paradigm.

So, Batsim is responsible for taking a workload file and following each job's progress through the simulation. It sends messages over a socket to a scheduler. Batsim doesn't care if this scheduler is written in C, python, shell, java, etc…as long as it is able to follow the protocols of sending messages over the socket. The scheduler makes scheduling decisions. So to make this simpler, here is an example:

Workload file 'w0'

2 simulator

2.1 overview



2.1.1 myBatchTasks.sh

2.1.2 generate_config.py

2.1.3 run-experiments.py

2.1.4 experiment.sh

2.1.5 real_start.py

2.1.6 post-processing.py

2.2 config.ini Schema

2.3 progress.sh

2.4 analysis

2.4.1 aggregate_makespan.py

3 batsim4

3.1 batsim options

3.2 workload

3.3 passing messages

3.3.1 protocol reader

3.3.2 protocol writer

3.4 batsim_tools

4 batsched4

4.1 batsched options

4.2 isalgorithm

4.3 batsched_tools