b2luigi introduction

(For those already familiar with luigi)

Joint meeting: Luigi-based Workflow Management with b2luigi/law

Michael Eliachevitch
(meliache@uni-bonn.de, meliache, elimik31, meliache)

Thursday 18 November

Physikalisches Institut — Rheinische Friedrich-Wilhelms-Universität Bonn

INTRODUCTION — **B2LUIGI**: BRINGING BATCH 2 LUIGI!

- full documentation at b2luigi.readthedocs.io
- lightweight helper package for `luigi`, that...

INTRODUCTION — B2LUIGI: BRINGING BATCH 2 LUIGI!

- full documentation at b2luigi.readthedocs.io
- lightweight helper package for `luigi`, that...
- brings batch to (2) luigi
    - use existing tasks, be able to switch between batch systems (LSF, HTCondor, local, ...)
    - schedule and monitor thousands of parallel jobs

INTRODUCTION — `B2LUIGI`: BRINGING BATCH 2 LUIGI!

- full documentation at b2luigi.readthedocs.io
- lightweight helper package for `luigi`, that...
- brings batch to (2) luigi
  - use existing tasks, be able to switch between batch systems (LSF, HTCondor, local, ...)
  - schedule and monitor thousands of parallel jobs
- helps with the bread 🍞 and butter 🧈 in luigi
  → e.g. data management / output handling

INTRODUCTION — `B2LUIGI`: BRINGING BATCH 2 LUIGI!

- full documentation at b2luigi.readthedocs.io
- lightweight helper package for `luigi`, that...
- brings batch to (2) luigi
  - use existing tasks, be able to switch between batch systems (LSF, HTCondor, local, ...)
  - schedule and monitor thousands of parallel jobs
- helps with the bread 🍞 and butter 🧈 in luigi
  $\rightarrow$ e.g. data management / output handling
- was originally developed for Belle II
  - collection of `basf2` helper tasks
  - grid-submission via wrapper for Belle II-specific tool `gbasf2`

HISTORY AND CONTRIBUTORS

- created in 2018 by Nils Braun at KIT for his Belle II tracking studies at the KEKCC LSF
- users at KIT and Belle II started contributing, e.g.
  - 2019: Max Welsch adds HTCondor batch (inspired by `law`)
  - 2020: I add LCG support for Belle II via soft `gbasf2` wrapper
  - ...
- 2021: Nils lefts for industry and I take over as main developer
- currently still in beta (`v0.7.4`), but has several active users within Belle II

The team 🧑‍🤝‍🧑

Main developer

Michael Eliachevitch (meliache)

Original author

Nils Braun (nils-braun)

Features, fixing, help and testing

- Felix Metzner (FelixMetzner)
- Patrick Ecker (eckerpatrick)
- Jochen Gemmler
- Maximilian Welsch (welschma)
- Kilian Lieret (klieret)
- Sviatoslav Bilokin (bilokin)
- Phil Grace (philiptgrace)
- Anselm Baur (anselmbaur)
- Moritz Bauer (sognetic)
- Artur Gottmann (ArturAkh)

TARGET USERS

I imagine our users

- work with python anyway

TARGET USERS

I imagine our users

- work with python anyway

  "I write my steering file in python, I want to submit my jobs from python without having to learn some submit file syntax."

TARGET USERS

I imagine our users

- work with python anyway

  "I write my steering file in python, I want to submit my jobs from python without having to learn some submit file syntax."

- comfortable with python classes
  (understands inheritance, properties, methods)

TARGET USERS

I imagine our users

- work with python anyway

  "I write my steering file in python, I want to submit my jobs from python without having to learn some submit file syntax."
- comfortable with python classes
  (understands inheritance, properties, methods)
- want to use bare `luigi` but wish for more convenient batch submission
  $\longrightarrow$ drop-in replacement: `import b2luigi as luigi`

TARGET USERS

I imagine our users

- work with python anyway
  "I write my steering file in python, I want to submit my jobs from python without having to learn some submit file syntax."
- comfortable with python classes
  (understands inheritance, properties, methods)
- want to use bare `luigi` but wish for more convenient batch submission
  $\rightarrow$ drop-in replacement: `import b2luigi as luigi`
- use `b2luigi` as a tool to build their own workflow
  $\rightarrow$ many users use it very differently

TARGET USERS

I imagine our users

- work with python anyway
  "I write my steering file in python, I want to submit my jobs from python without having to learn some submit file syntax."
- comfortable with python classes
  (understands inheritance, properties, methods)
- want to use bare `luigi` but wish for more convenient batch submission
  $\longrightarrow$ drop-in replacement: `import b2luigi as luigi`
- use `b2luigi` as a tool to build their own workflow
  $\longrightarrow$ many users use it very differently
- are potential contributors

WHY NOT USE THE ALREADY CREATED BATCH TASKS?

- luigi/contrib already contains many batch scheduling and monitoring tasks (b2luigi was inspired by sge and lsf implementations)
- b2luigi solves some limitations of those systems, the improvements are:
  - submit *many* parallel jobs
    in other batch implementations each job needs a monitoring task, but number of processes is limited on many systems
    $\rightarrow$ b2luigi requires only single process on submitting machine

WHY NOT USE THE ALREADY CREATED BATCH TASKS?

- luigi/contrib already contains many batch scheduling and monitoring tasks (b2luigi was inspired by sge and lsf implementations)
- b2luigi solves some limitations of those systems, the improvements are:
    - submit *many* parallel jobs
      in other batch implementations each job needs a monitoring task, but number of processes is limited on many systems
      $\rightarrow$ b2luigi requires only single process on submitting machine
    - batch-submit a large set of luigi tasks
      in other implementation you often need to implement a `work()` function instead of `run()` or define an external command to run

WHY NOT USE THE ALREADY CREATED BATCH TASKS?

- luigi/contrib already contains many batch scheduling and monitoring tasks (b2luigi was inspired by sge and lsf implementations)
- b2luigi solves some limitations of those systems, the improvements are:
    - submit *many* parallel jobs
      in other batch implementations each job needs a monitoring task, but number of processes is limited on many systems
      $\rightarrow$ b2luigi requires only single process on submitting machine
    - batch-submit a large set of luigi tasks
      in other implementation you often need to implement a `work()` function instead of `run()` or define an external command to run
    - flexibility in choosing a batch systems
      $\rightarrow$ you can write your task first, test locally and then choose or change the batch you process them on

BATCH SYSTEM WORKFLOW

- write normal (b2)luigi task

```python
import b2luigi

@b2luigi.requires(SomeOtherTask, foo_parameter="bar")
class MyTask(b2luigi.Task):
    cut_value = b2luigi.IntParameter():
    input_file = b2luigi.Parameter(hashed=True):

    def run(self):  # …
    def output(self):  # …
```

BATCH SYSTEM WORKFLOW

- write normal (b2)luigi task

```python
import b2luigi

@b2luigi.requires(SomeOtherTask, foo_parameter="bar")
class MyTask(b2luigi.Task):
    cut_value = b2luigi.IntParameter():
    input_file = b2luigi.Parameter(hashed=True):

    def run(self):  # …
    def output(self):  # …
```

- use `b2luigi.process(tasks)` to run tasks locally or on a batch system

```python
if __name__ == '__main__':
    b2luigi.set_setting("batch_system", "htcondor")  # "local" for local submission
    tasks = [MyTask(cut_value=0.5, input_file=input_file) for input_file in input_files]
    b2luigi.process(tasks, batch=True)
```

COMMAND LINE INTERFACE

- `b2luigi.process()` adds a CLI interface to your script:

```
$ python3 steering_file.py --help
usage: htcondor_example.py [-h] [--show-output] [--test] [--batch] [--batch-runner] [--dry-run]
↪  [--scheduler-host SCHEDULER_HOST] [--scheduler-port SCHEDULER_PORT] [--task-id TASK_ID]

optional arguments:
-h, --help            show this help message and exit
--show-output         Instead of running the tasks, show which output files will/are created.
--test                Run the task list in test mode by printing the log directly to the screen
                        instead of storing it in a file.
--batch               Instead of running locally, try to submit the tasks to the batch system.
--batch-runner        Expert option to mark this worker as a batch runner.
--dry-run             Do not run any task but set the return value to 0, if the tasks are complete.
--scheduler-host SCHEDULER_HOST
                        If given, use this host as a central scheduler instead of a local one.
--scheduler-port SCHEDULER_PORT
                        If given, use the port on this host as a central scheduler instead of a local
                        ↪  one.
--task-id TASK_ID     EXPERT.
```

  - select run-mode: `batch`, `test`, `dry-run` or `show-output`
  - connect to central scheduler

B2LUIGI SETTINGS

- provide additional options to b2luigi
- Can be set via
  1. via class attributes (e.g. static properties, luigi parameters or property functions)
  2. via `b2luigi.set_setting(key, value)`
  3. `settings.json` configuration file
- issue #31: *Support luigi's own config*

B2LUIGI SETTINGS

- provide additional options to b2luigi
- Can be set via
  1. via class attributes (e.g. static properties, luigi parameters or property functions)
  2. via `b2luigi.set_setting(key, value)`
  3. `settings.json` configuration file
- issue #31: *Support luigi's own config*

Example

```
class RecoTask(b2luigi.DispatchableTask):
    result_dir = "/path/to/results"
    batch_system = b2luigi.Parameter(significant=False)

    @propery
    def htcondor_settings(self):
        return {"+requestRuntime": int(self.get_nevents() *
        ↪   0.2)}

if __name__ == '__main__':
    b2luigi.set_setting("env_script", "./setup.sh")
    b2luigi.process(RecoTask(batch_system="htcondor"))
```

`settings.json`:

```
{"log_dir" : "/path/to/logs"}
```

ADD A NEW BATCH SYSTEM

- define interface `BatchProcess` (*not* as `luigi.Task`) with
  - `start_job()`
  - `kill_job()`
  - `get_job_status() -> JobStatus`

  ```
  class JobStatus(enum.Enum):
      running = "running"
      successful = "successful"
      aborted = "aborted"
      idle = "idle"
  ```

- issue #2: *Include new batch systems.*
  Contributions welcome!

SUPER-HERO TASK CLASSES

`b2luigi.Task` is a supercharged version of `luigi.Task`

- methods to help with data management

```python
import b2luigi
import random

class MyNumberTask(b2luigi.Task):
    random_seed = b2luigi.IntParameter()

    def output(self):
        # ./random_seed=<seed>/output_file.txt
        yield self.add_to_output("output_file.txt")

    def run(self):
        random.seed(self.random_seed)
        random_number = random.random()
        # ./random_seed=<seed>/output_file.txt
        out_path = self.get_output_file_name(
            "output_file.txt"
        )
        with open(out_path, "w") as f:
            f.write(f"{random_number}\n")
```

SUPER-HERO TASK CLASSES

`b2luigi.Task` is a supercharged version of `luigi.Task`

- methods to help with data management
    - helps you organize your outputs with the structure

      `<result_dir>/param1=foo/param2=bar/…/filename`

```python
import b2luigi
import random

class MyNumberTask(b2luigi.Task):
    random_seed = b2luigi.IntParameter()

    def output(self):
        # ./random_seed=<seed>/output_file.txt
        yield self.add_to_output("output_file.txt")

    def run(self):
        random.seed(self.random_seed)
        random_number = random.random()
        # ./random_seed=<seed>/output_file.txt
        out_path = self.get_output_file_name(
            "output_file.txt"
        )
        with open(out_path, "w") as f:
            f.write(f"{random_number}\n")
```

SUPER-HERO TASK CLASSES

`b2luigi.Task` is a supercharged version of `luigi.Task`

- methods to help with data management
  - helps you organize your outputs with the structure

    ```
    <result_dir>/param1=foo/param2=bar/…/filename
    ```

  - `self.add_to_output("<filename>")`
    - yield this `self.output()` generate a `luigi.LocalTarget` with a path under `result_dir` that encodes the values of the luigi parameter.

```python
import b2luigi
import random

class MyNumberTask(b2luigi.Task):
    random_seed = b2luigi.IntParameter()

    def output(self):
        # ./random_seed=<seed>/output_file.txt
        yield self.add_to_output("output_file.txt")

    def run(self):
        random.seed(self.random_seed)
        random_number = random.random()
        # ./random_seed=<seed>/output_file.txt
        out_path = self.get_output_file_name(
            "output_file.txt"
        )
        with open(out_path, "w") as f:
            f.write(f"{random_number}\n")
```

SUPER-HERO TASK CLASSES

`b2luigi.Task` is a supercharged version of `luigi.Task`

- methods to help with data management
  - helps you organize your outputs with the structure

    ```
    <result_dir>/param1=foo/param2=bar/…/filename
    ```

  - `self.add_to_output("<filename>")`
    - yield this `self.output()` generate a `luigi.LocalTarget` with a path under `result_dir` that encodes the values of the luigi parameter.
  - `self.get_output_file_name("<filename>")` :
    - use this in other methods like `run()` to get the generated file path.

```python
import b2luigi
import random

class MyNumberTask(b2luigi.Task):
    random_seed = b2luigi.IntParameter()

    def output(self):
        # ./random_seed=<seed>/output_file.txt
        yield self.add_to_output("output_file.txt")

    def run(self):
        random.seed(self.random_seed)
        random_number = random.random()
        # ./random_seed=<seed>/output_file.txt
        out_path = self.get_output_file_name(
            "output_file.txt"
        )
        with open(out_path, "w") as f:
            f.write(f"{random_number}\n")
```

DISPATCHING

- Problem: Many HEP-tasks are steering-files that run unsafe C++ framework code that can crash (`segfault`) and stop all tasks from running

DISPATCHING

- Problem: Many HEP-tasks are steering-files that run unsafe C++ framework code that can crash (`segfault`) and stop all tasks from running
- Solution: *Dispatchable tasks* that emulate batch submission on local computer and runs it in separate execution path
  - decorate your `run()` method with `@b2luigi.dispatch`
  - or inherit from `b2luigi.DispatchableTask` and implement `process()` method instead of `run()` (this is what `Basf2Task` and its implementations do)

ATOMIC OUTPUT CREATION

- Problem: Long-running tasks often create output before filling it.
  $\rightarrow$ If task fails during execution, but the output is already created, it's mistakenly marked as *complete* ("thanksgiving bug")

ATOMIC OUTPUT CREATION

- Problem: Long-running tasks often create output before filling it.
  → If task fails during execution, but the output is already created, it's mistakenly marked as *complete* (”thanksgiving bug”)
- Solution: use `@b2luigi.on_temporary_files` decorator, e.g. for `run()` or `process()`
  → Modifies `get_output_file_name()` to return temporary file location, to which output is written first. After the decorated function was successful, the output is moved to the actual output location.

DEVELOPMENT PRACTICES

- development happens on github, if you want something add an Issue or fork and PR
- unit tests encouraged
    - core-functionality well-covered
    - batch-systems not so (but still WIP)
- github actions for CI
    - pre-commit for style and static syntax checking
    - run unittests and calculate coverage
- see development documentation for a guide how to contribute

Thanks for listening.

It’s time for live action!

Backup