Karlsruhe Institute of Technology

Belle II

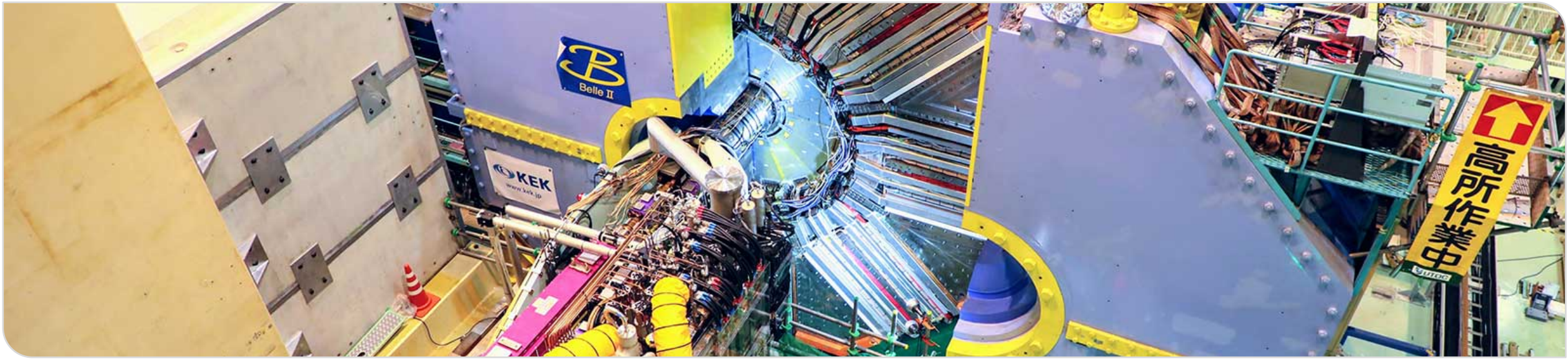
Institute of Experimental Particle Physics

b2luigi - Bringing Batch 2 luigi

Belle II Germany Meeting, October 1st, 2024

Alexander Heidelbach, Jonas Eppelt, Giacomo De Pietro

alexander.heidelbach@kit.edu



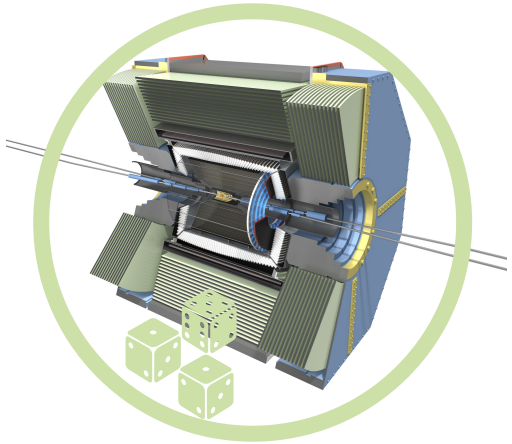
Analysis in a Nutshell

Belle II Group,
ETP, Karlsruhe Institute of Technology

Analysis in a Nutshell

Institute of Experimental Particle Physics

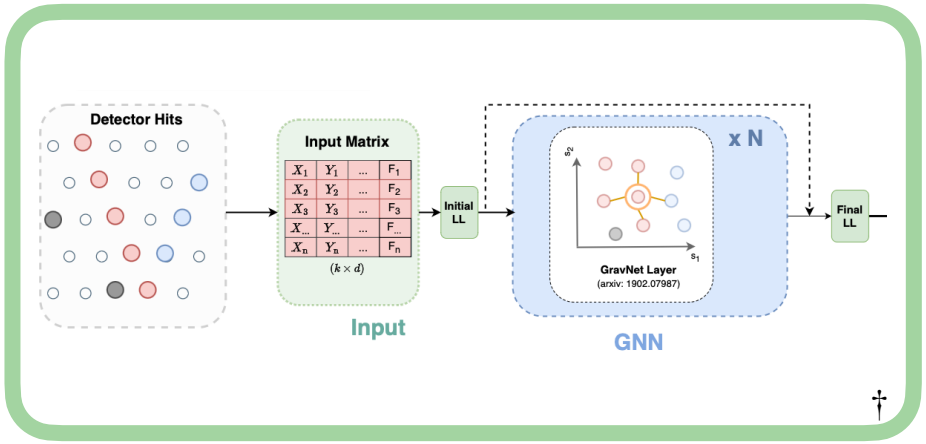
Data



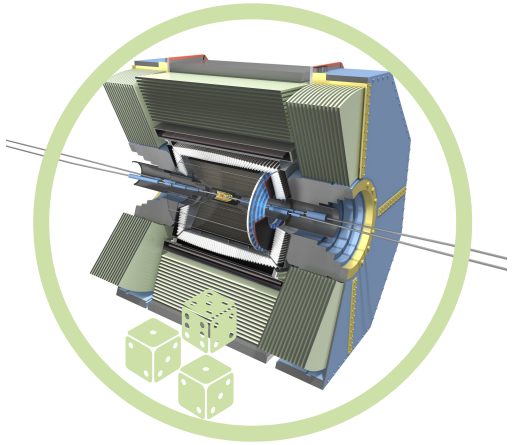
Analysis in a Nutshell

Institute of Experimental Particle Physics

† from Lea Reuter



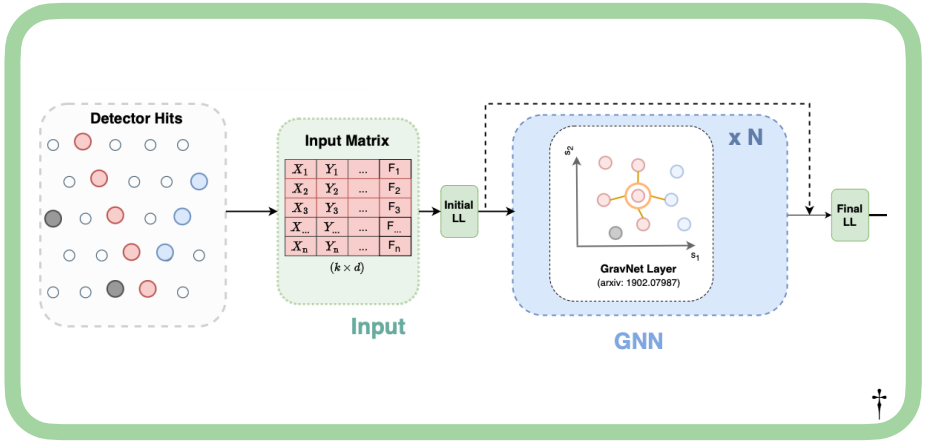
Data → Preprocessing



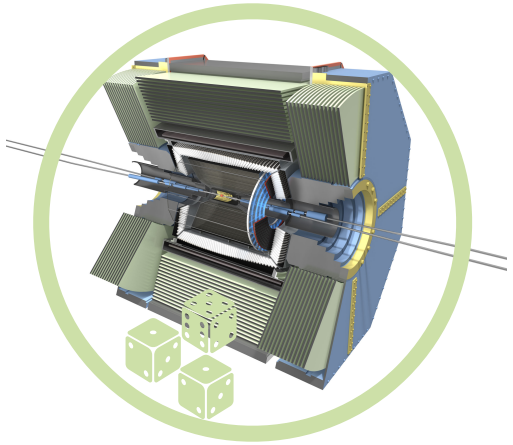
Analysis in a Nutshell

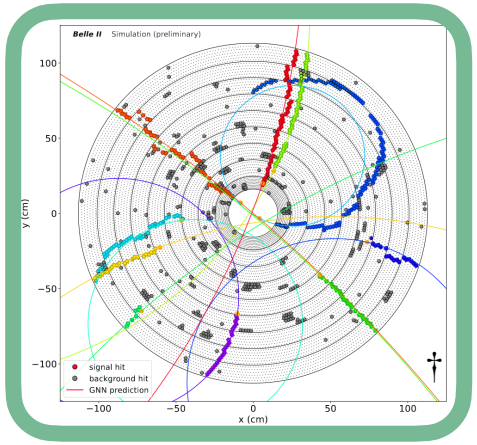
Institute of Experimental Particle Physics

† from Lea Reuter

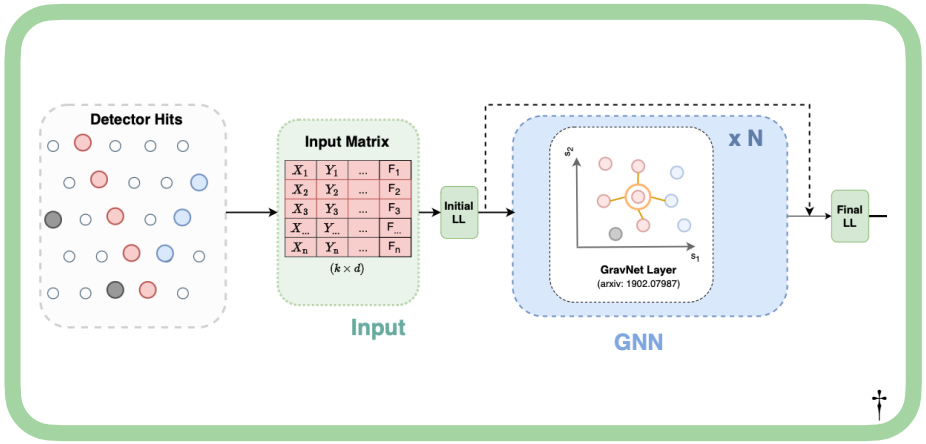


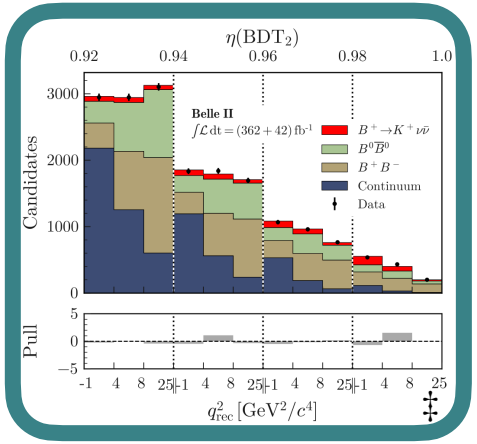
Data → Preprocessing → Reconstruction





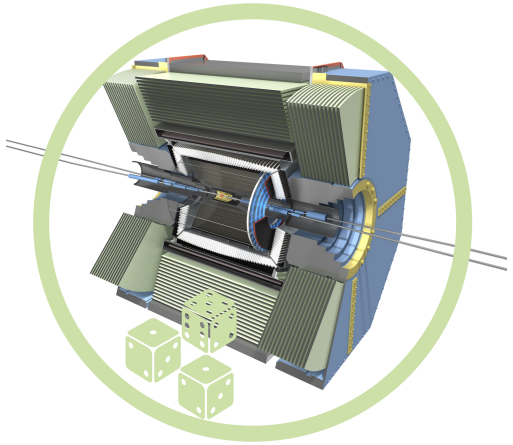
Analysis in a Nutshell

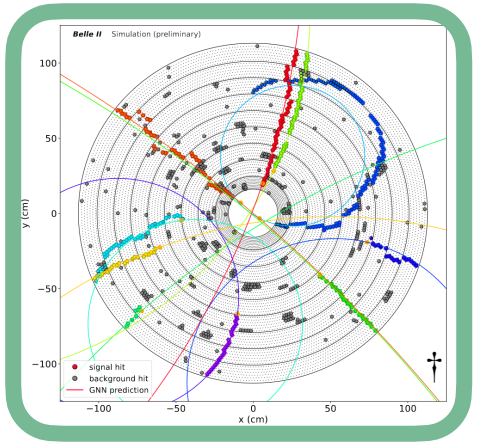




† from Lea Reuter
‡ PhysRevD.109.112006

Data → Preprocessing → Reconstruction → Selection



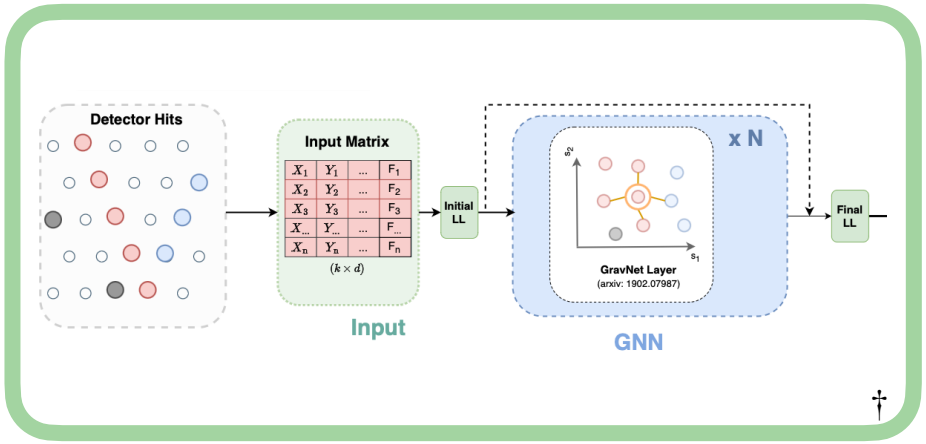


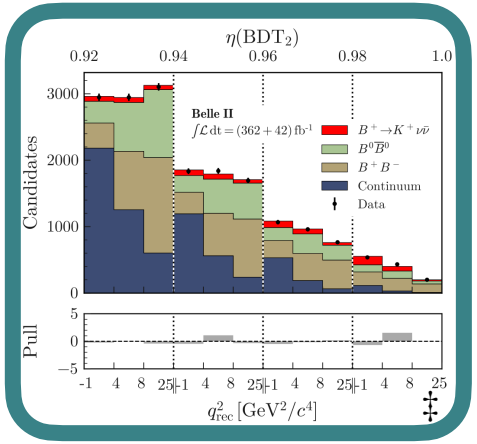
Belle II Group,
ETP, Karlsruhe Institute of Technology

Analysis in a Nutshell

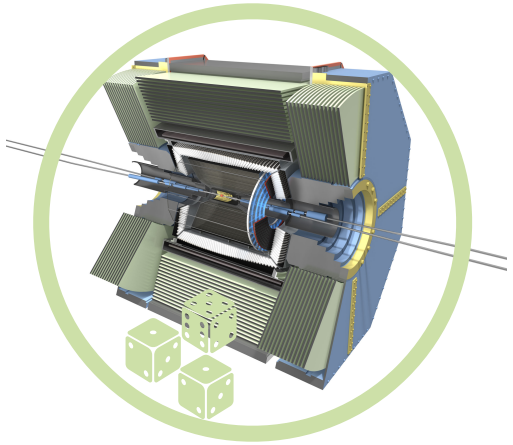
Institute of Experimental Particle Physics

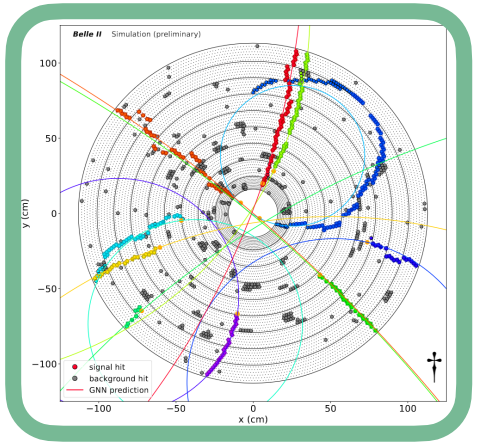
† from Lea Reuter
‡ PhysRevD.109.112006

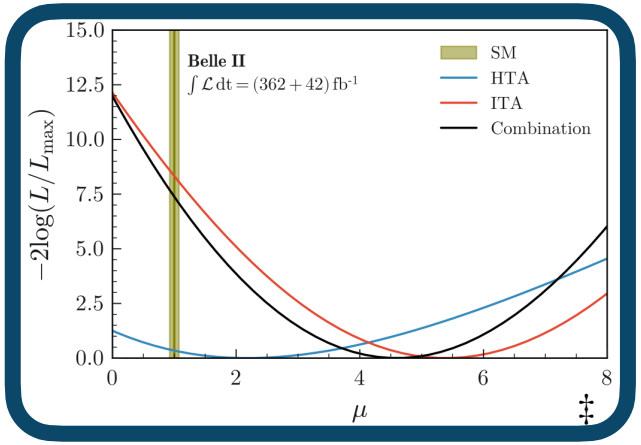




Data → Preprocessing → Reconstruction → Selection → Analysis







The Analysis Code

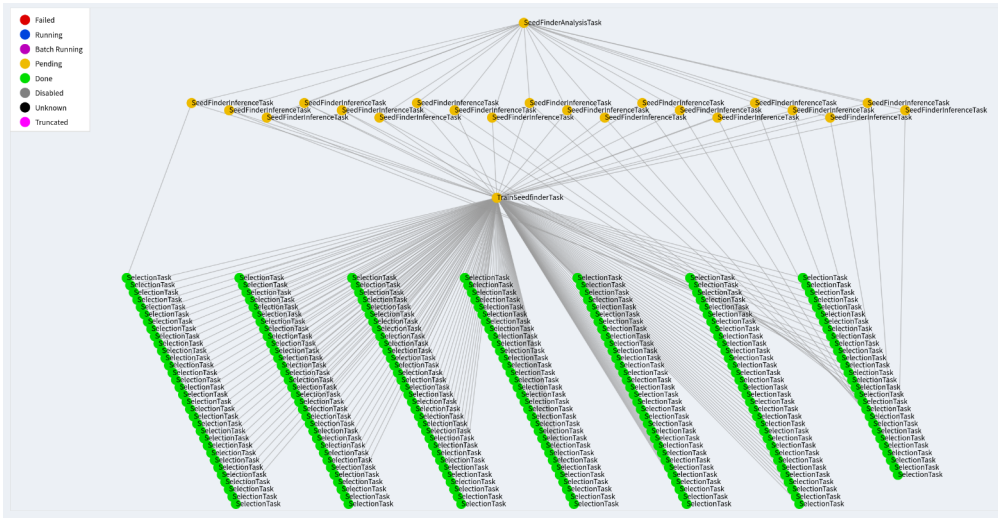
Institute of Experimental Particle Physics

Without b2luigi

```
1_gamma                  feature_ana_paper
2_gamma                  losses.py
9_1_22_3.csv             models.py
9_1_22_4.csv             print_crystal_data.py
9_2_1_56.csv             __pycache__
9_2_1_57.csv             rebatch.py
dataset.py               run_1_barrel.py
feature_ana              run_2_barrel.py
feature_analysis.ipynb   run_2_full.py
```

```
run_2_nominal.py         utils_gnn.py
run_34_early.py          utils_train_inf.py
run_34_nominal.py        zaper_run_1_early.py
run_feature_ana_0b.py    zaper_run_1_nominal.py
run_feature_ana_0.py     zaper_run_feature_ana_early.py
run_feature_ana_1b.py    zaper_run_feature_ana_nominal.py
run_feature_ana_1.py
run_feature_ana_2.py
test.py
```

With b2luigi



- Workflow Management Systems…
  - … enable the user to structure the workflow by defining dependencies
  - … build the dependency graph and schedule the execution

Belle II Group,
ETP, Karlsruhe Institute of Technology

Luigi in a Nutshell

- **Building a pipeline**
  - Dependency Resolution
  - Workflow Management
  - Visualisation
  - Handling Failures
  - Command Line Integration
  - …
- https://github.com/spotify/luigi

Spotify®

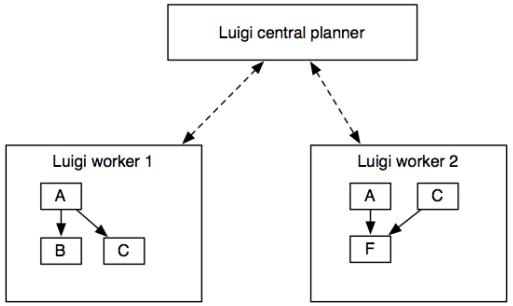
"Hello World" in luigi:

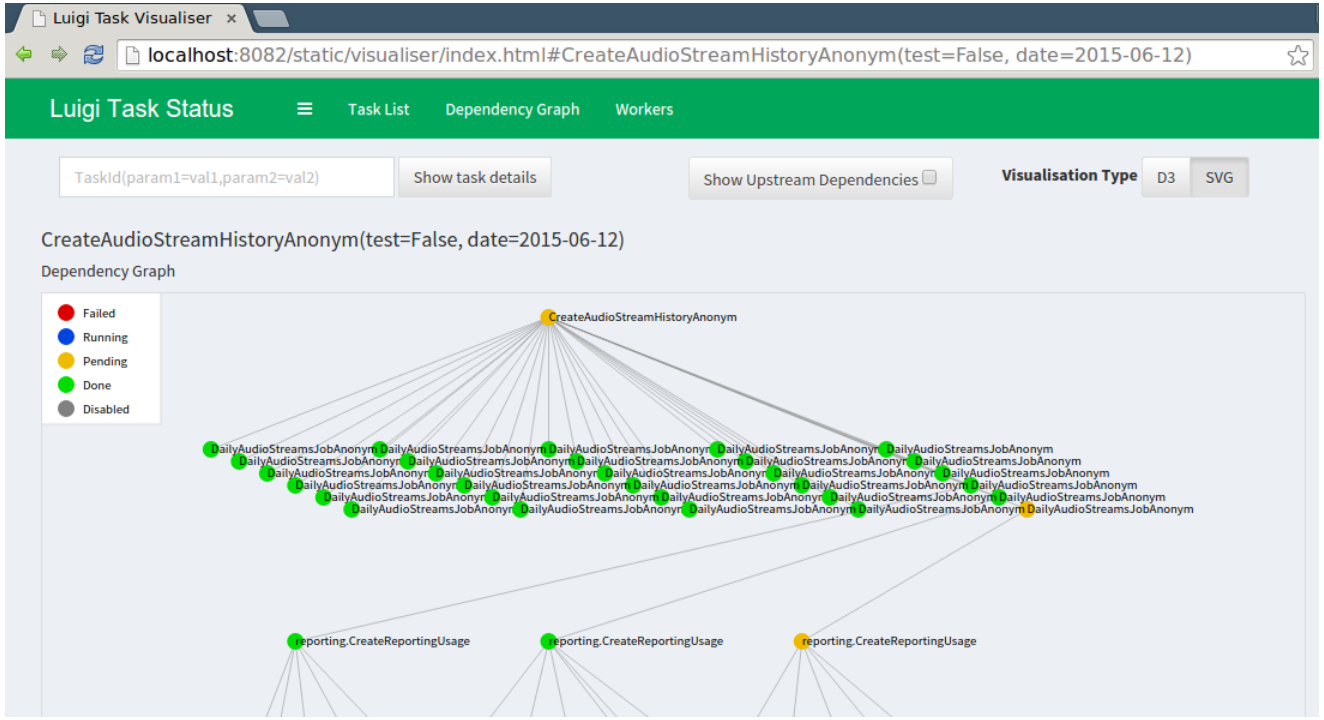
```
class MyTask(luigi.Task):
        parameter = luigi.Parameter()

        def run(self):
                do_smth(self.parameter)

        def output(self):
                return Target("some/file")

        def requires(self):
                yield OtherTask()
```


Belle II Group,
ETP, Karlsruhe Institute of Technology

The luigi Scheduler

- User encodes dependencies of tasks
- The scheduler builds dependency graph and makes sure that multiple workers don't execute the same job
- Graphical visualisation provided





Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task

Institute of Experimental Particle Physics

```
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

A Simple Task

```python
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

- Parameter object for task steering
- Define dependency
  - Dependency is fulfilled when output exists
- Target object
  - Simple case: LocalTarget
- Execution logic

A Simple Task

```python
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

- Parameter object for task steering
- Define dependency
  - Dependency is fulfilled when output exists
- Target object
  - Simple case: LocalTarget
- Execution logic

A Simple Task

```
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

- Parameter object for task steering
- Define dependency
  - Dependency is fulfilled when output exists
- Target object
  - Simple case: LocalTarget
- Execution logic

Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task

```
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

- Parameter object for task steering
- Define dependency
  - Dependency is fulfilled when output exists
- Target object
  - Simple case: LocalTarget
- Execution logic

Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task

```python
import b2luigi
import random
from analysis import Other Task

class MyNumberTask(b2luigi.Task):
    some_parameter = b2luigi.IntParameter()
    batch_system = “local”

    def requires(self):
        yield OtherTask(other_parameter=self.some_parameter/2)

    def output(self):
        yield self.add_to_output("output_file.txt")

    def run(self):
        random_number = random.random()

        with open(self.get_output_file_name("output_file.txt"), "w") as f:
            f.write(f"{random_number}\n")
```

- Parameter object for task steering
- Define dependency
  - Dependency is fulfilled when output exists
- Target object
  - Simple case: LocalTarget
- Execution logic

Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task Output

```
===== Luigi Execution Summary =====

Scheduled 100 tasks of which:
* 100 ran successfully:
    - 100 MyTask(some_parameter=0,1,10,11,12,13,14,15,16,17,18,...)

This progress looks :) because there were no failed tasks or missing dependencies

===== Luigi Execution Summary =====
```

```
❯ ls
results  simple-example.py
❯ ls results
'some_parameter=0'   'some_parameter=27'  'some_parameter=45'  'some_parameter=63'  'some_parameter=81'
'some_parameter=1'   'some_parameter=28'  'some_parameter=46'  'some_parameter=64'  'some_parameter=82'
'some_parameter=10'  'some_parameter=29'  'some_parameter=47'  'some_parameter=65'  'some_parameter=83'
'some_parameter=11'  'some_parameter=3'   'some_parameter=48'  'some_parameter=66'  'some_parameter=84'
'some_parameter=12'  'some_parameter=30'  'some_parameter=49'  'some_parameter=67'  'some_parameter=85'
'some_parameter=13'  'some_parameter=31'  'some_parameter=5'   'some_parameter=68'  'some_parameter=86'
'some_parameter=14'  'some_parameter=32'  'some_parameter=50'  'some_parameter=69'  'some_parameter=87'
'some_parameter=15'  'some_parameter=33'  'some_parameter=51'  'some_parameter=7'   'some_parameter=88'
'some_parameter=16'  'some_parameter=34'  'some_parameter=52'  'some_parameter=70'  'some_parameter=89'
'some_parameter=17'  'some_parameter=35'  'some_parameter=53'  'some_parameter=71'  'some_parameter=9'
'some_parameter=18'  'some_parameter=36'  'some_parameter=54'  'some_parameter=72'  'some_parameter=90'
'some_parameter=19'  'some_parameter=37'  'some_parameter=55'  'some_parameter=73'  'some_parameter=91'
'some_parameter=2'   'some_parameter=38'  'some_parameter=56'  'some_parameter=74'  'some_parameter=92'
'some_parameter=20'  'some_parameter=39'  'some_parameter=57'  'some_parameter=75'  'some_parameter=93'
'some_parameter=21'  'some_parameter=4'   'some_parameter=58'  'some_parameter=76'  'some_parameter=94'
'some_parameter=22'  'some_parameter=40'  'some_parameter=59'  'some_parameter=77'  'some_parameter=95'
'some_parameter=23'  'some_parameter=41'  'some_parameter=6'   'some_parameter=78'  'some_parameter=96'
'some_parameter=24'  'some_parameter=42'  'some_parameter=60'  'some_parameter=79'  'some_parameter=97'
'some_parameter=25'  'some_parameter=43'  'some_parameter=61'  'some_parameter=8'   'some_parameter=98'
'some_parameter=26'  'some_parameter=44'  'some_parameter=62'  'some_parameter=80'  'some_parameter=99'
❯ ls results/some_parameter=0
output_file.txt
```

- The scheduler **tracks** the **state** of each job and **takes action**
- **Job is done = Output file exists**
  - Unique output names per job
  - Add parameters to job output name
  - Create a folder structure for each parameter
  - b2luigi does this automatically
  - **No check of content!**

Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task Settings

```
class MyTask(b2luigi.Task):
    some_setting = “some value”
```

```
class MyTask(b2luigi.Task):
    @property
    def some_setting(self):
        return “some value”
```

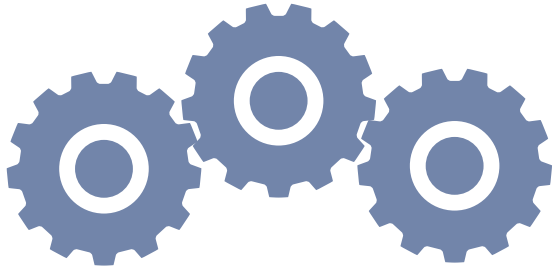
```
b2luigi.set_setting(“some_setting”, "some value")
```

settings.json

```
{
    “some_setting”: “some value”
}
```

Priority

- **Settings** are handled by b2luigi!
- **Control**:
  - batch system choice
  - output and log path
  - environment
  - workflow specific settings



A Simple Task Continued

```python
class MyAverageTask(b2luigi.Task):
    def requires(self):
        for i in range(100):
            yield self.clone(MyNumberTask, some_parameter=i)

    def output(self):
        yield self.add_to_output("average.txt")

    def run(self):
        summed_numbers = 0
        counter = 0
        for input_file in self.get_input_file_names("output_file.txt"):
            with open(input_file, "r") as f:
                summed_numbers += float(f.read())
                counter += 1

        average = summed_numbers / counter

        with open(self.get_output_file_name("average.txt"), "w") as f:
            f.write(f"{average}\n")
```

- More **helper functions**:
  - b2luigi.Task.get_input_file_names()
  - b2luigi.Task.get_output_file_name()

Belle II Group,
ETP, Karlsruhe Institute of Technology

A Simple Task Continued

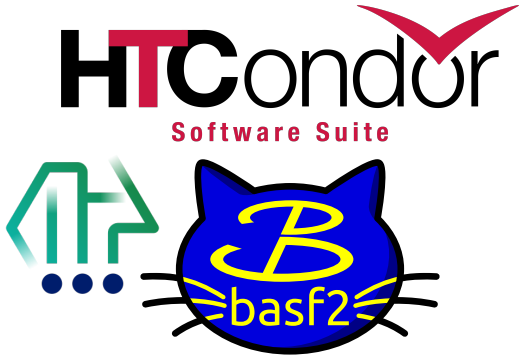
```python
class MyAverageTask(b2luigi.Task):
    def requires(self):
        for i in range(100):
            yield self.clone(MyNumberTask, some_parameter=i)

    def output(self):
        yield self.add_to_output("average.txt")

    def run(self):
        summed_numbers = 0
        counter = 0
        for input_file in self.get_input_file_names("output_file.txt"):
            with open(input_file, "r") as f:
                summed_numbers += float(f.read())
                counter += 1

        average = summed_numbers / counter

        with open(self.get_output_file_name("average.txt"), "w") as f:
            f.write(f"{average}\n")
```

- More **helper functions**:
  - b2luigi.Task.get_input_file_names()
  - b2luigi.Task.get_output_file_name()
- Jobs that are done will not be run again!

```
❯ python3 simple-example.py --show-output
average.txt
        /home/aheidelbach/b2luigi/results/average.txt

output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=0/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=1/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=2/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=3/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=4/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=5/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=6/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=7/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=8/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=9/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=10/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=11/output_file.txt
        /home/aheidelbach/b2luigi/results/some_parameter=12/output_file.txt
```

Batch Processing

- Initial motivation: **Make batch submission easy!**
- Currently **fully supported**:
  - HTCondor
  - LSF
  - Gbasf2
- **Challenges**:
  - Using your environment
    - **Settings**: env_script, env,..
  - Ensuring consistent locations
    - **Settings**: working_dir, result_dir,…




```
class MyTask(b2luigi.Task):
    batch_system = "htcondor"
```

```
class MyTask(b2luigi.Task):
    @property
    def htccondor_settings(self):
        return {"request_memory": 4096}
```

```
b2luigi.set_setting("result_dir", "path/to/result")
```

settings.json

```
{
    "env_script": "setup.sh"
}
```


Belle II Group,
ETP, Karlsruhe Institute of Technology

Example: HTCondor

Institute of Experimental Particle Physics

```
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

Example: HTCondor

```python
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side

Belle II Group,
ETP, Karlsruhe Institute of Technology

Example: HTCondor

```
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side
- Location needs to be accessible on the batch job side

Belle II Group,
ETP, Karlsruhe Institute of Technology

Example: HTCondor

```
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side
- Location needs to be accessible on the batch job side
- Wrapper tasks need no output

Belle II Group,
ETP, Karlsruhe Institute of Technology

Example: HTCondor

```python
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side
- Location needs to be accessible on the batch job side
- Wrapper tasks need no output
- Definition of the batch system

Belle II Group,
ETP, Karlsruhe Institute of Technology

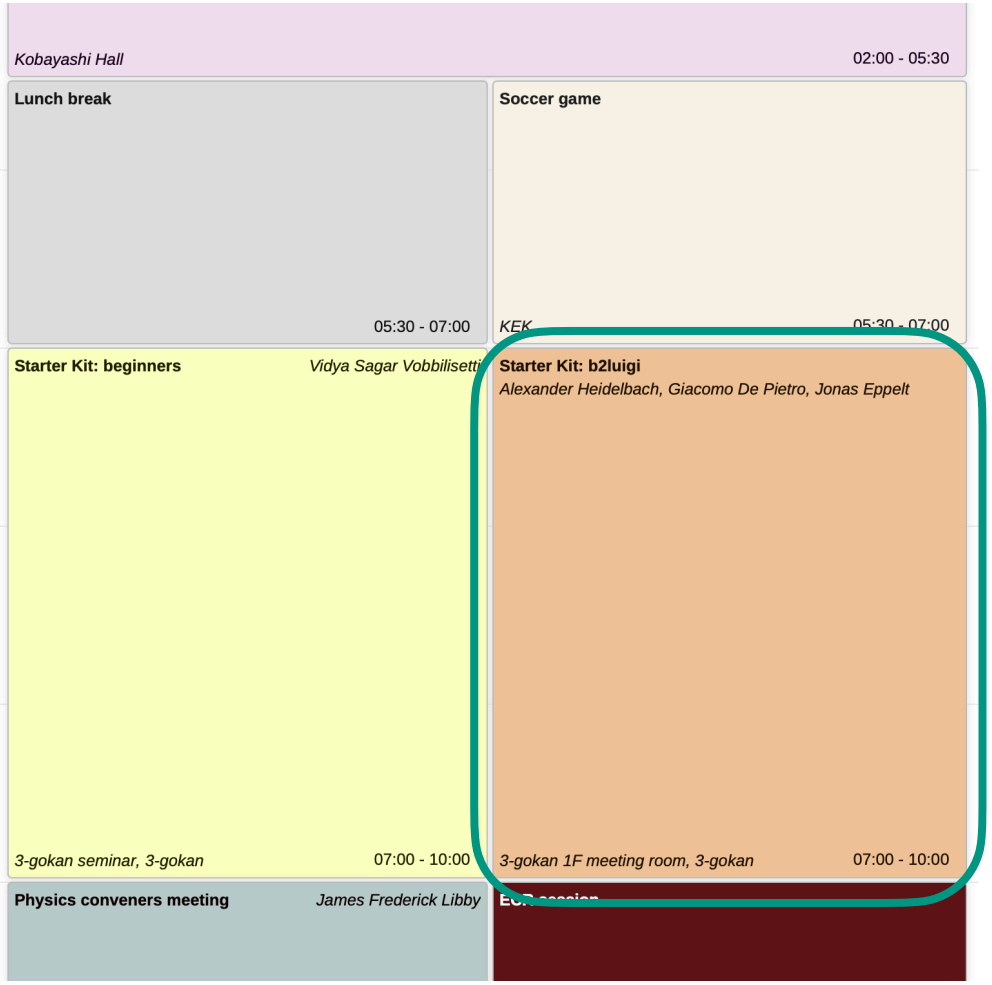
Example: HTCondor

```python
class MyTask(b2luigi.Task):
    parameter = b2luigi.IntParameter()
    batch_system = "htcondor"

    @property
    def executable(self):
        return ["$MY_PYTHON"]

    def output(self):
        yield self.add_to_output("test.txt")

    def run(self):
        with open(self.get_output_file_name("test.txt"), "w") as f:
            f.write(f"Test {self.parameter}")

class Wrapper(b2luigi.WrapperTask):
    def requires(self):
        for i in range(10):
            yield MyTask(parameter=i)

if __name__ == "__main__":
    b2luigi.set_setting("env_script", "setup.sh")

    b2luigi.set_setting("result_dir", "results")

    b2luigi.process(Wrapper(), batch=True, workers=100)
```

- Script executed on the batch job side
- Location needs to be accessible on the batch job side
- Wrapper tasks need no output
- Definition of the batch system
- Executable for this specific task
  - E.g. environment variable set in setup.sh

Advertisement

- **b2luigi workshop at the next B2GM**
  - Friday 11th October at 3pm
  - 3-gokan 1F meeting room
  - **In person only!**
- **Topics**:
  - (b2)luigi basics
  - basf2 analysis from simulation to ntuple
  - Sending b2luigi tasks to batch systems
  - Offline analysis hints

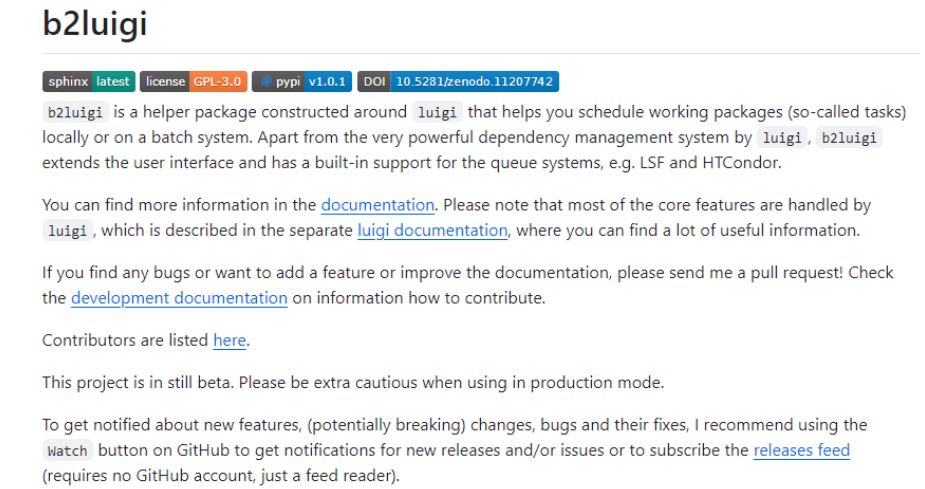


Belle II Group,
ETP, Karlsruhe Institute of Technology

Summary & Discussion

Institute of Experimental Particle Physics

- b2luigi provides a simple and flexible implementation to run your workflow on batch systems!
- In case of questions, issues, wishes…
  ⇒ Reach out to us via GitLab issues, b2questions, email, …
- **Implementations in Belle II**
  - VIBE - Validation Interface for the Belle II Experiment
  - Systematic Corrections Framework
  - Your Analysis/Tool?



GitLab

Github

Zenodo

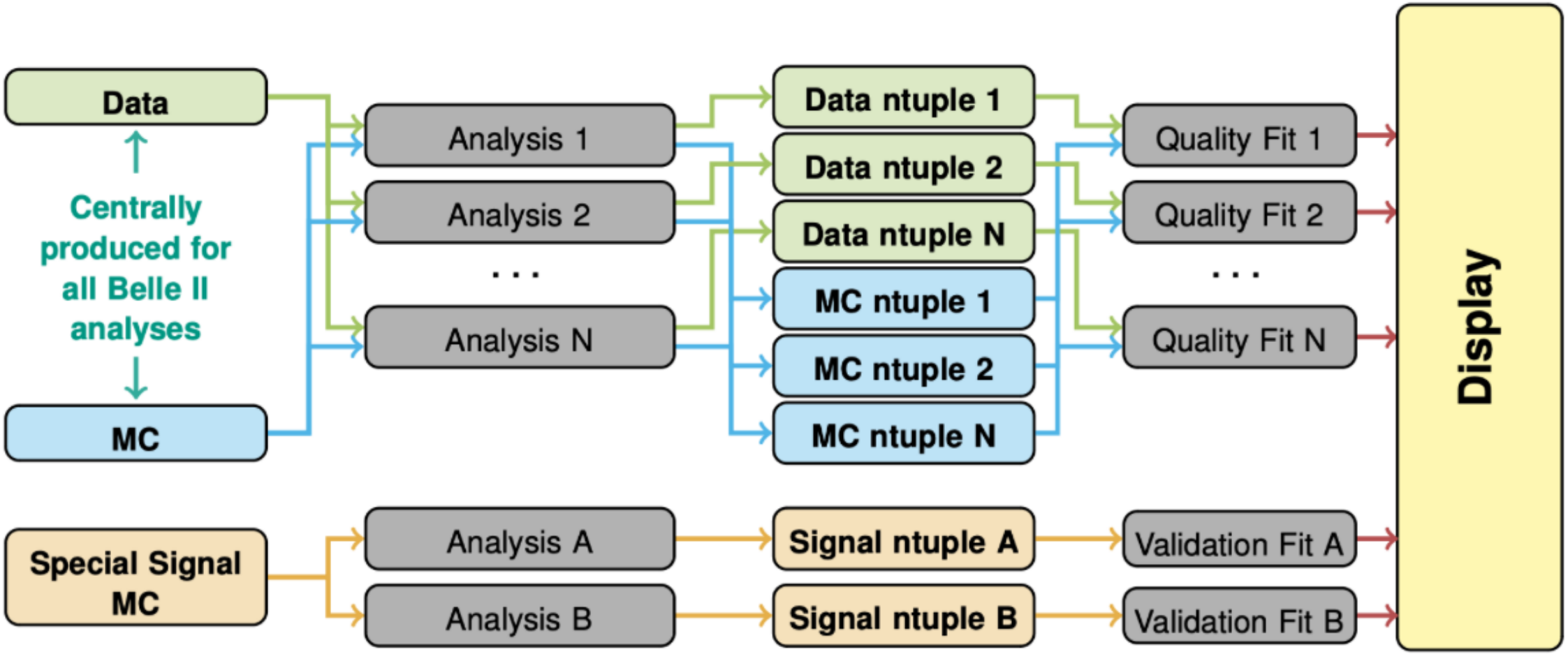
Documentation

PyPi

Belle II Group,
ETP, Karlsruhe Institute of Technology

Backup

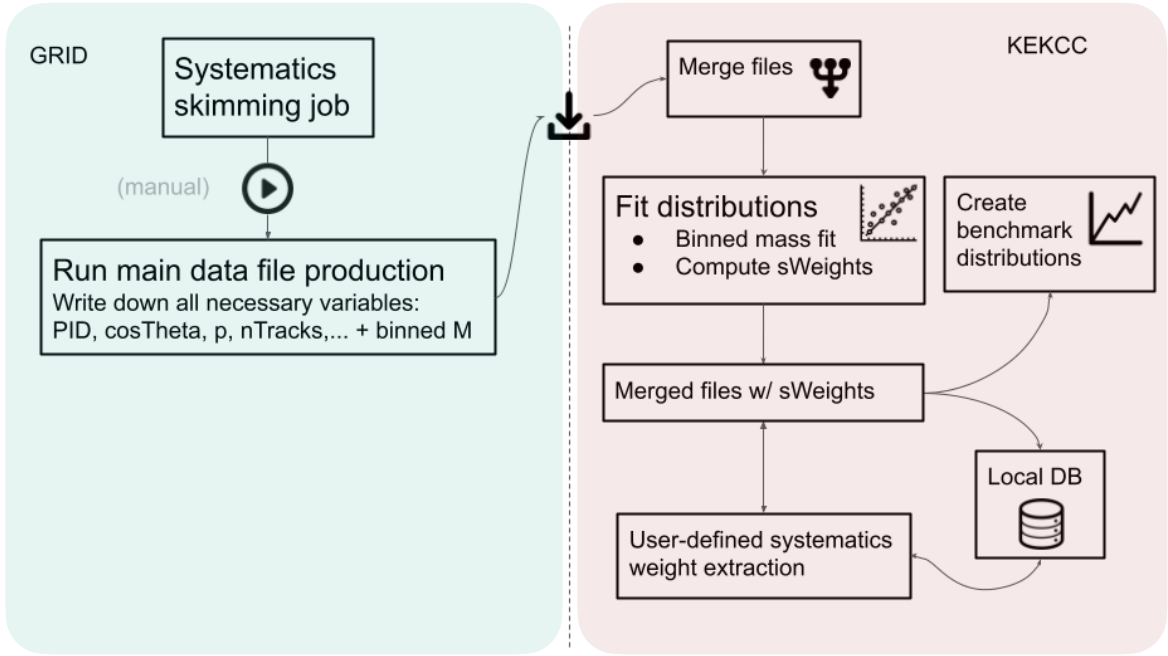
Validation Interface for the Belle II Experiment



Belle II Group,
ETP, Karlsruhe Institute of Technology

Systematic Corrections Framework

- 2 Stage Algorithm
  - Ntuple Production
    - Centrally run for every campaign
    - Running: Gbasf2
  - Data/MC Corrections
    - User runs their specific selection
    - Running: Locally, HTCondor, LSF
- Also: Validation of different datasets

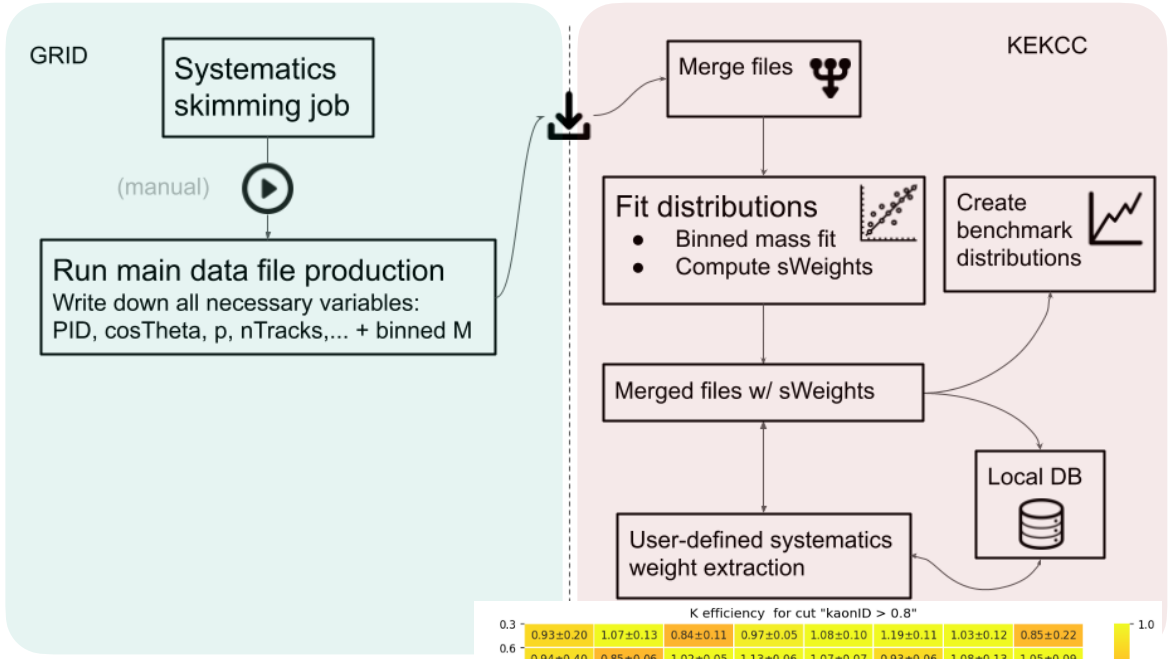


Documentation

Systematic Corrections Framework

Institute of Experimental Particle Physics

- 2 Stage Algorithm
  - Ntuple Production
    - Centrally run for every campaign
    - Running: Gbasf2
  - Data/MC Corrections
    - User runs their specific selection
    - Running: Locally, HTCondor, LSF
- Also: Validation of different datasets

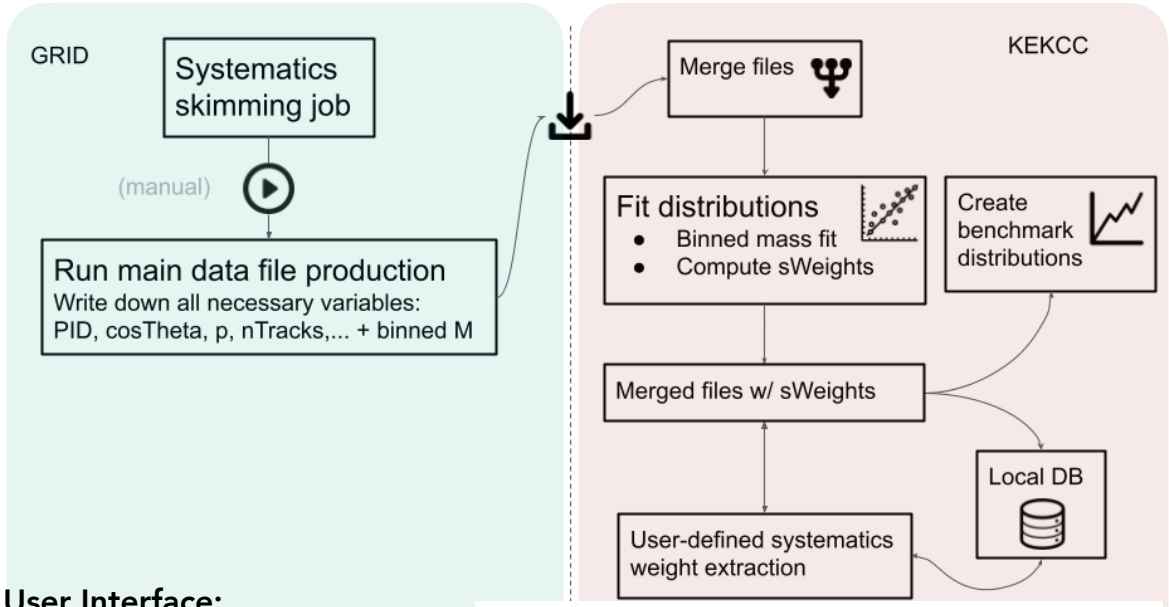


[Documentation]

Systematic Corrections Framework

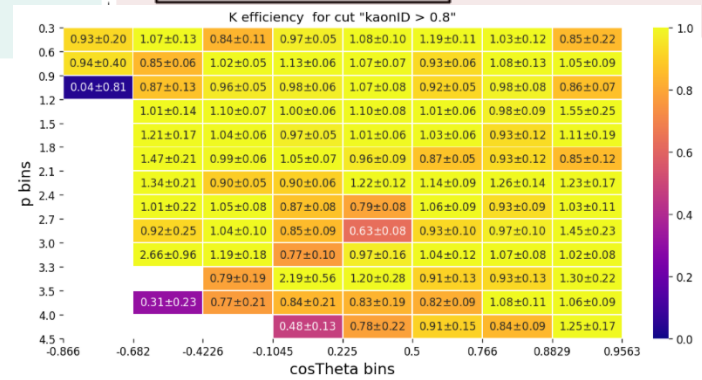
Institute of Experimental Particle Physics

- 2 Stage Algorithm
  - Ntuple Production
    - Centrally run for every campaign
    - Running: Gbasf2
  - Data/MC Corrections
    - User runs their specific selection
    - Running: Locally, HTCondor, LSF
- Also: Validation of different datasets



User Interface:

```
# ================================== #
#  Hadron ID weight configuration file  #
# ================================== #
weight_dir: 'fixed_weights/'
remove_tmp_files: True
weight_cfg_list:
-
  prefix_name: Rdtmc_v1
  efficiency particle type: 'K'
  fakerate particle type: 'pi'
  binning: [[0.5, 2.5, 4.5],
            [-0.8, 0.2, 0.9563]]
  track variables: [ "p", "cosTheta" ]
  cuts: [ "kaonID > 0.2", "kaonID > 0.8"]
  precuts: [ "", "charge > 0", "charge < 0" ]
  mc proc query: [ "MC14ri_1" ]
  data_proc_query: [ "proc12e7" ]
```



[Documentation]