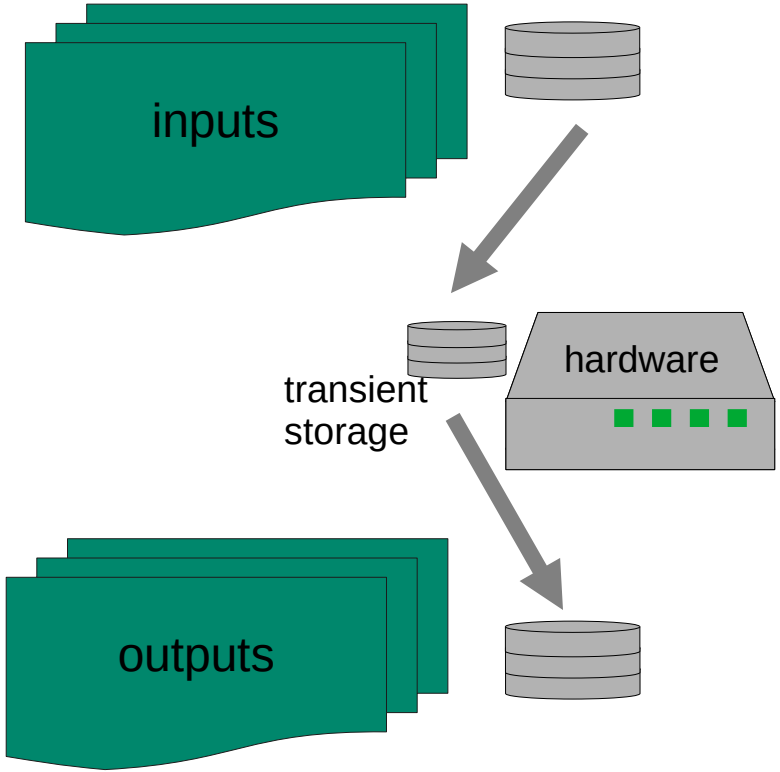
Karlsruhe Institute of Technology

Institut für Experimentelle Teilchenphysik

Checkpointing for long-running Machine Learning Tasks

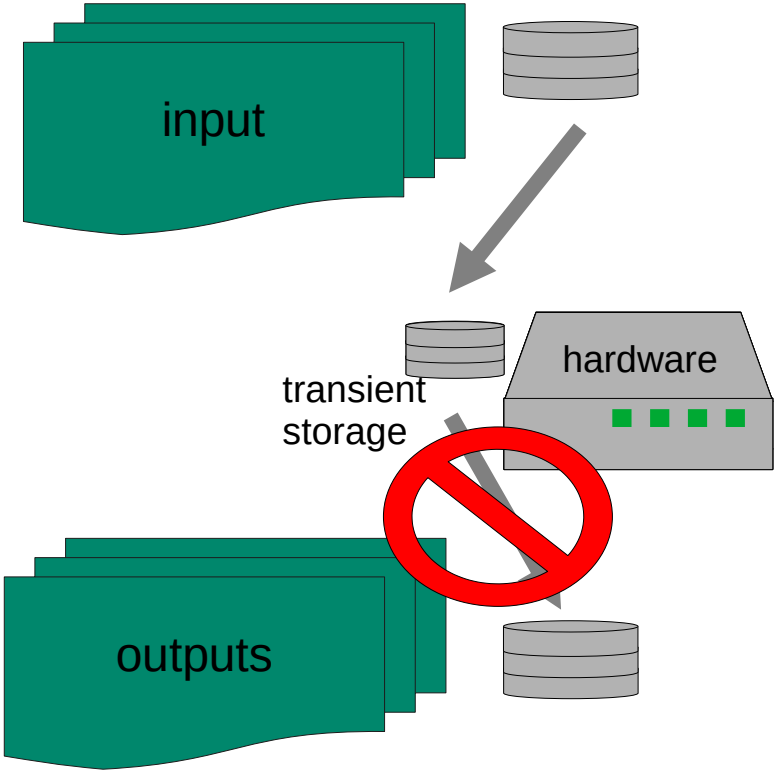
Jonas Eppelt, Matthias Schnepf, Giacomo De Pietro, Günter Quast

A typical HEP job



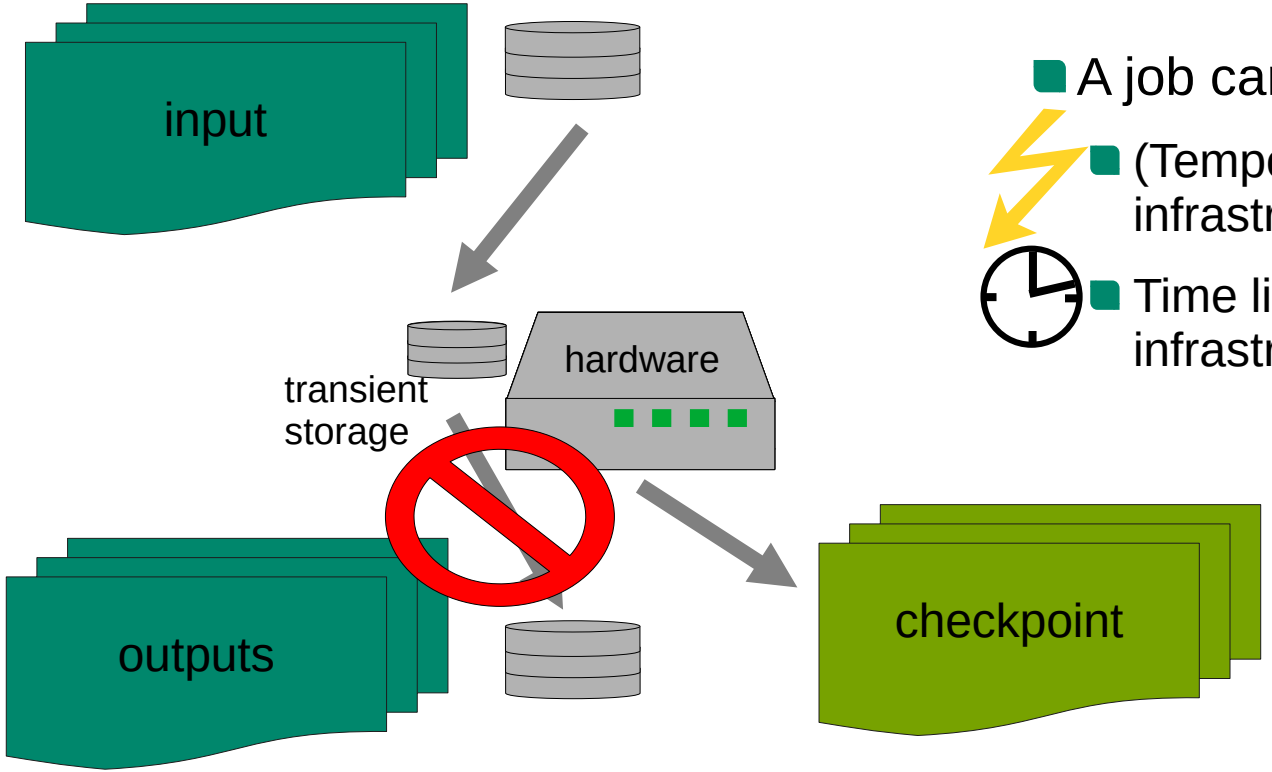
- A job needs:
  - Inputs
  - Hardware to run on
  - Outputs
- In general, transient outputs only exist for the duration of the job.

Jobs can be interrupted



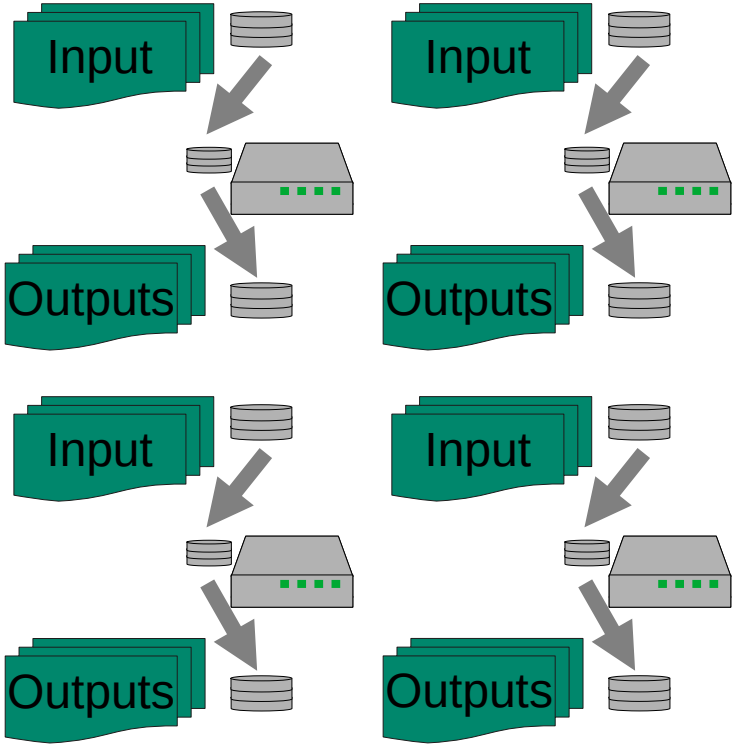
- A job can fail because of:
  - (Temporary) failure of infrastructure
  - Time limits on its infrastructure

Jobs can be interrupted



- A job can fail because of:
  - (Temporary) failure of infrastructure
  - Time limits on its infrastructure
- A checkpoint stores the job's current state to allow its continuation.

Typical solution in HEP: parallelize by data

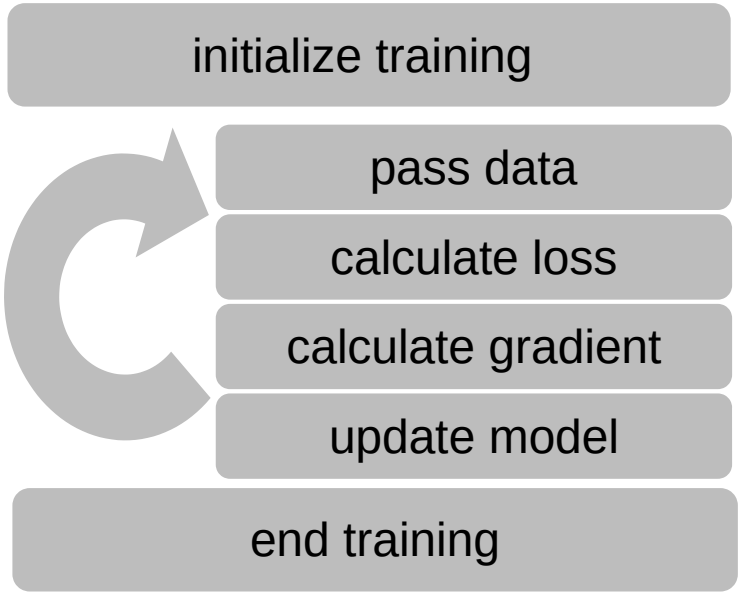


- Jobs are trivial parallelizable:

- Reduces runtime per Job to abide by site restrictions.

- Only the failed batches have to be reprocessed.

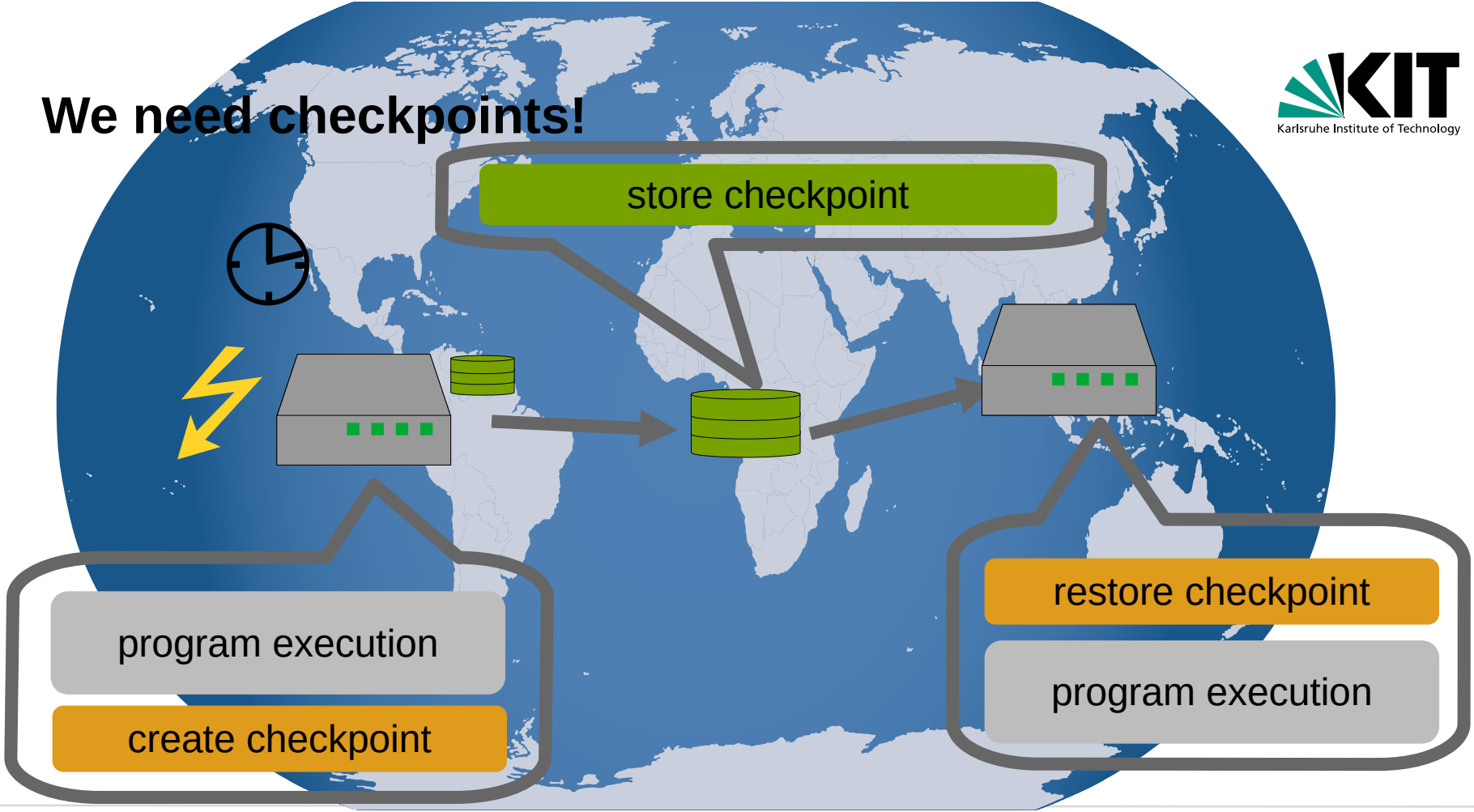
The special case of Machine Learning (ML) trainings



- A step always relies on the previous step:
  - We can not use the typical HEP strategy of running on batches.
- They can have a long runtime (days/weeks).

- Failure results in a complete retraining.

We need checkpoints!



The term “checkpoint” in Computing and ML

Computing checkpoints:

- Goal: Continue work on different place and time
  - Includes all necessary information on the jobs state.
  - Only the latest state is needed.
  - Stored persistently

ML checkpoints:

- Goal: Find the best performing model
  - Do not always include the full state of the training
    - State of Callbacks, Logs, ...
  - Multiple checkpoints are kept to analyze after the training
    - Find best model by higher level metric, create ensembles, debug training, ….
  - (Usually) stored on transient storage

What do we need to checkpoint?

Create/Restore ML checkpoints:

- Already included in major ML libraries

When to checkpoint:

- Induced by the side
- Regularly

Storing/transferring Checkpoints

- Shared file system
- Grid storage
- Batch system

Rescheduling

- Batch systems
- Workflow management systems

We need a place to bring them together.

A Python class to bring everything together

- One tool to configure everything needed.
- Set custom checkpoint and restore function.
- Not depending on a specific ML framework.

```python
from checkpointer.checkpointer import Checkpointer

checkpointer = Checkpointer(
    local_checkpoint_file=Path("checkpoint.pt"), # define checkpoint file
    # define a function, that saves the checkpoint
    checkpoint_function=lambda path, model: torch.save(model.state_dict(),
        path),
    # define a function, that restores the checkpoint
    restore_function=lambda path: model.load_state_dict(torch.load(path)["
        model_state_dict"]),

# Reload checkpoint and give default value if there is none
model = checkpointer.restore(model)
# Trigger the checkpoint creation and transfer
checkpointer.checkpoint(i)
```

Get started here:
https://github.com/JonasEppelt/Checkpointer

When to checkpoint

Planned end of job:

- Site has time to give a signal
- This signal must be relayed to the python process.
- Internally, the Checkpointer is already setup to catch the signals 10 and 15.
- Upon receiving, it will:
  - Ensure the current checkpoint exist.
  - Transfer it as configured.
  - Exit with 85

```
# push program to the background
#!/bin/bash
python3 train.py &
pid=$! # get pid
trap "kill -15 $pid" 15 #SIGINT
trap "kill -10 $pid" 10 #SIGTERM
wait $pid
```

When to checkpoint

Unplanned end of job:

- No time to send a signal
- Proactively do regular checkpoints
- Frequency can be configured with ”checkpoint_every” to only create checkpoints every i-th call of the step function.

```
checkpointer = Checkpointer(
    ...
    checkpoint_every = 100)
    ...
for i in range(epochs):
    ...
    checkpointer.step(i, model)
```

Storing Checkpoints

- If no mode is set, the local checkpoint is assumed to be persistent.
- Custom behavior can be set using the manual mode.
- For shared file systems:

```
checkpointer = Checkpointer(
    ...
    checkpoint_transfer_mode = "shared",
    checkpoint_transfer_target = Path("...")
)
```

- For grid storage:

```
checkpointer = Checkpointer(
    ...
    checkpoint_transfer_mode = "xrootd",
    checkpoint_transfer_target = "/pnfs/...",
    xrootd_server_name = "..." )
```

Interplay with HTCondor Software Suite

- HTCondor has its own mechanism to store checkpoints and reschedule.
- Settings needed in the JDL file:
  - *checkpoint_exit_code* (the code your program will exit with, to signal a checkpoint exists and it wants to be rescheduled)
  - *transfer_checkpoint_files* defines the files to checkpoint
  - *when_to_transfer_output = ON_EXIT_OR_EVICT*
- Python class can infer these settings.
  - The provided settings will be overwritten with this!

```
checkpointer = Checkpointer(
    ...
    checkpoint_transfer_mode="htcondor"
)
```

Interplay with Keras and Lightning

- In high-level libraries the training loop is not directly accessible.
  - Instead Callbacks are used.
- The Checkpointer comes with callbacks for Keras and Lightning.

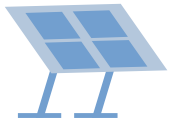
```python
from checkpointer.lightning_callback import LightningCheckpointerCallback

checkpointer = LightningCheckpointerCallback(
    local_checkpoint_file=Path(checkpoint_path),
    checkpoint_every=1
)
trainer = Trainer(
    max_epochs=epochs,
    accelerator='gpu',
    callbacks = [checkpointer]
)

trainer.fit(
    model,
    train_dataloader,
    test_dataloader,
    ckpt_path= checkpointer.restore()
)
```

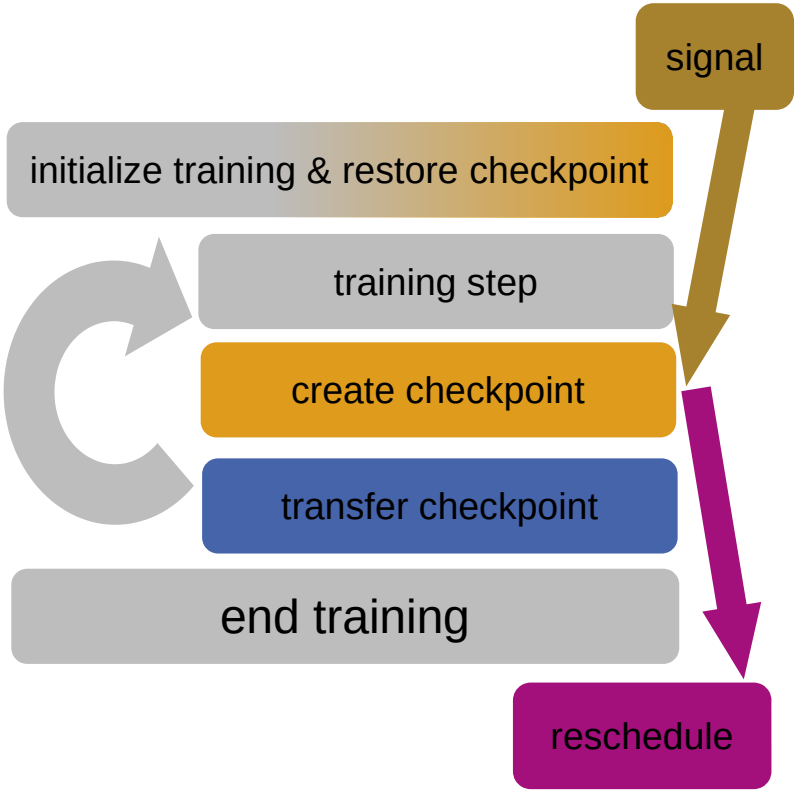
A few notes:

- Checkpointing also helps with any other kind of crashes:
  - Crashing trainings due to numerical instabilities.
  - Crashes due to configuration errors in a later training phase.
  - OOM Errors when working with sparse inputs (e.g. GNNs)
- Currently, there is no support to run in Jupyter.
  - Though, you might succeed with some versions of Jupyter.
- Checkpointing can also be used to do “greener” computing:
  - Jobs are run if renewable energy is plentiful.
  - Jobs are stopped if renewable energy is scarce.



Conclusions

- Checkpointing ML training enables
  - resistance to failures.
  - abiding by time constraints.
- A Python class unifies the necessary configurations.
  - https://gitlab.desy.de/jonas.eppelt/checkpointer



Backup

Rescheduling with luigi

- Only the final model must be in the Tasks outputs
- Use a central scheduler:

```
[scheduler]
retry_count=100
retry_delay=5 #seconds
[worker]
keep_alive=true
max_reschedules=100
```

Technical Details – Core Functions

checkpoint(self, value)

If no value is given:

Load value from internal value

Call the configured checkpoint function

Set internal checkpoint to value

restore(self, default)

Copy checkpoint file
from configured storage

If a local checkpoint exists:

Read it with configured
restore function

Else:

Return default

Technical Details – Helper Functions

transfer_checkpoint(self)

Call configured transfer method

step(self, value)

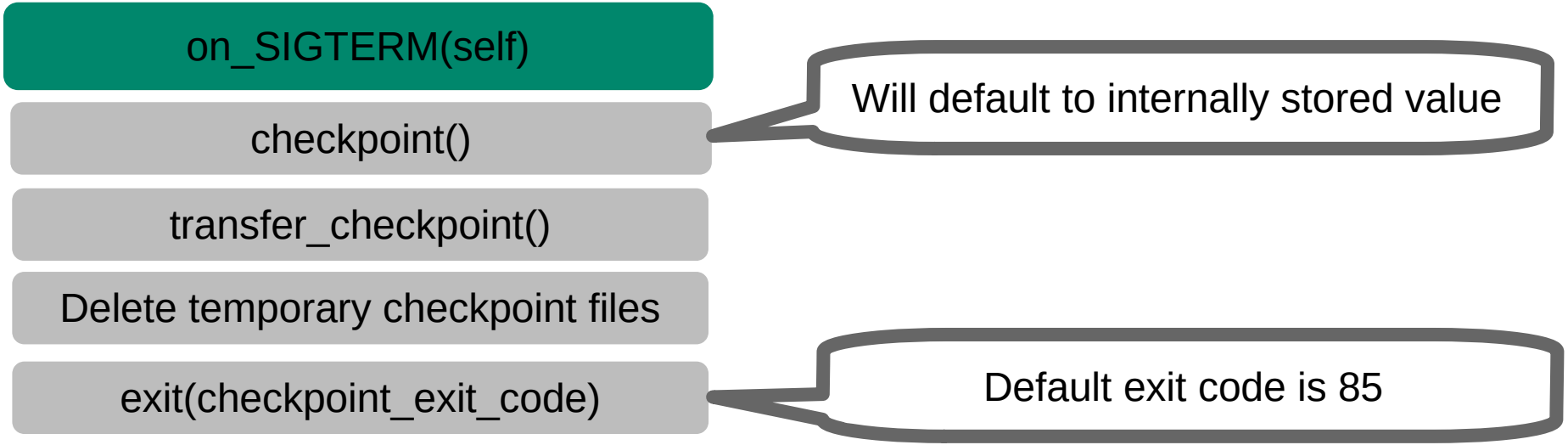
Set internal checkpoint to value

If checkpoint frequency reached:

checkpoint(self, value)

transfer_checkpoint(self, value)

Technical Details – Signal Traps



Technical Details – key configuration

- local_checkpoint_file: A local path where the current checkpoint will be stored
- checkpoint_function: A function, that takes a path and an arbitrary object and writes the checkpoint file
- restore_function: A function, that takes a path and returns the check-pointed objects.
- checkpoint_transfer_mode: How to transfer the checkpoints to persistent storage. Supported are “None”, “shared”, “htcondor”, “xrootd” and manual.
- checkpoint_every: Checkpoint frequency used in the “step” function.

Demonstrator Project: “Green” Tier3 node

- Checkpointing can also be used to do “greener” computing:
  - Jobs are run, if renewable energy is plentiful.
  - Jobs are stopped, if renewable energy is scarce.
- One node with GPUs on the Tier 3 center TOpAS at GridKa is configured.
  - Goals:
    - How can we get user acceptance?
    - What challenges arise, when putting such a system in practice?
    - How large is a potential CO2 saving?
  - German electrical energy mix is monitored using API provided by Fraunhofer Institute for Solar Energy Systems.