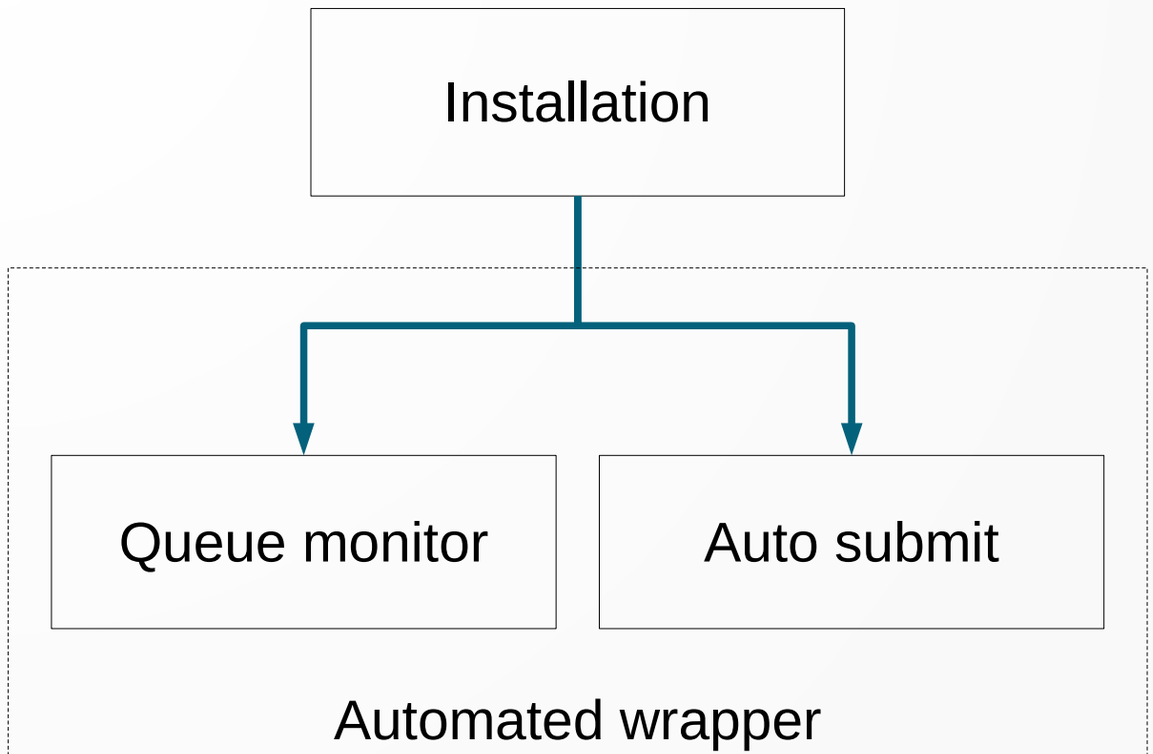
Under the hood of the automated workflow

The workflow

- Installation
- Queue monitor
- Auto submit
- Automated wrapper



Terminology

- Simulation:
  - The entire collection of realisations and the tasks to run on them
- Realisation (or rel):
  - An instance of a fault with its own SRF and stoch file
- Task
  - A block of work within the workflow pipeline of a realisation. E.G. HF, EMOD3D, BB

Installation

- Installation takes a list of faults and realisation counts, as well as a Data directory containing Sources and VMs for each fault
- Sources contains SRFs and Stoch files
  - Possibly also sim_params if perterbations have been used
- For each fault a fault directory is created, and for each realisation a directory is created in the relevant fault directory
- Parameter yaml files for the simulation, each fault and each realisation are created
- Creates a database file containing all possible realisation and task combinations

Simulation folder structure

- For installation to succed a Data directory containing the Sources and VMs directories is required
- The fault selection file may be here, but is not required
- Installation created the Runs directory, mgmt_db_queue directory and slurm_mgmt.db file

- Simulation root
  - slurm_mgmt.db
  - fault_selection.txt (optional)
  - mgmt_db_queue
  - Data
    - Sources
      - <Faults>
        - SRF
          - SRF and srfinfo file per realisation
        - Stoch
          - Stoch file per realisation
    - Vms
      - <Faults>
        - VM data
  - Runs
    - <Faults>
      - <Fault>_REL<realisation number>
        - HF
        - LF
        - BB

Queue monitor

- Checks squeue for all nesi00213 users for the current state of submitted tasks
- Checks the mgmt folder for updates on running and completed tasks
- Updates the database file with the most recent data on these tasks
- Sleeps for 5 seconds before starting again

- If tasks have failed or are not found on squeue when they were last known to be running they are reset to be submitted again, unless they have failed twice already
- The database can only be editted by one program at a time, attempting to run run two instances of queue monitor at the same time may cause failures
- Database is sqlite as it allows for lightweight creation and usage for each simulation

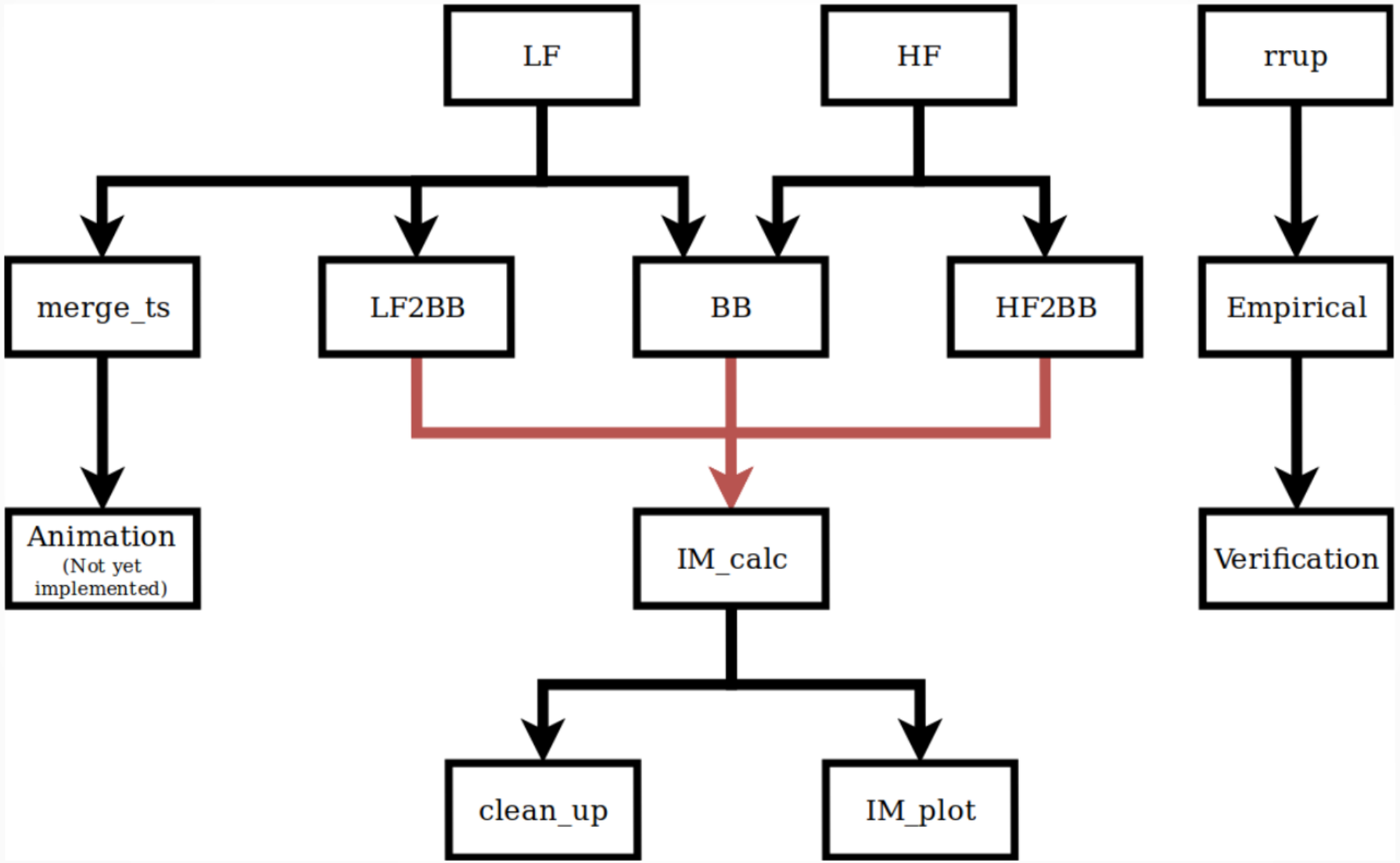
Auto submit

- Auto submit needs the name of the current user, and the root of the simulation directory
- The auto submit script checks the database for tasks that have fulfilled dependencies
- Auto submit checks squeue and determines the maximum number of tasks that can be submitted
- These tasks are submitted to slurm
- Sleeps for 5 seconds before starting again
- Other arguments allow specifying maximum number of tasks to run per machine and how long to sleep for

Tasks to run

- The tasks to be run default to running EMOD3D, HF, BB, IM_calc and clean_up for all realisations
- The tasks to be run and realisations can be customised
- Tasks can be specified using the flag --task_types_to_run
- Multiple tasks can be given with this flag
- Task dependencies are automatically added

Task dependencies



Realisations to run

- The realisations that the tasks are to be run for can be specified using the flag --rels_to_run
- Needs to be an sqlite formatted string
  - Wildcards can be used with the % character
- Only one realisation type can be given

- As auto submit does not modify the database it can be used by multiple people on the same simulation folder
- Each user should provide a --rels_to_run that does not overlap with any other user to prevent race conditions
- This allows a single simulation to have more tasks running on Maui than a single user could have
- By default 12 tasks will be run on Maui and Mahuika per user
- Queue monitor should only be run once and by only the user who installed the simulation

Automated wrapper

- The automated wrapper creates multiple threads for the other scripts to run in, allowing you to run a cybershake with one command
- The automated wrapper needs the name of the user, and a configuration file detailing which tasks to run for which realisations

Wrapper configuration file

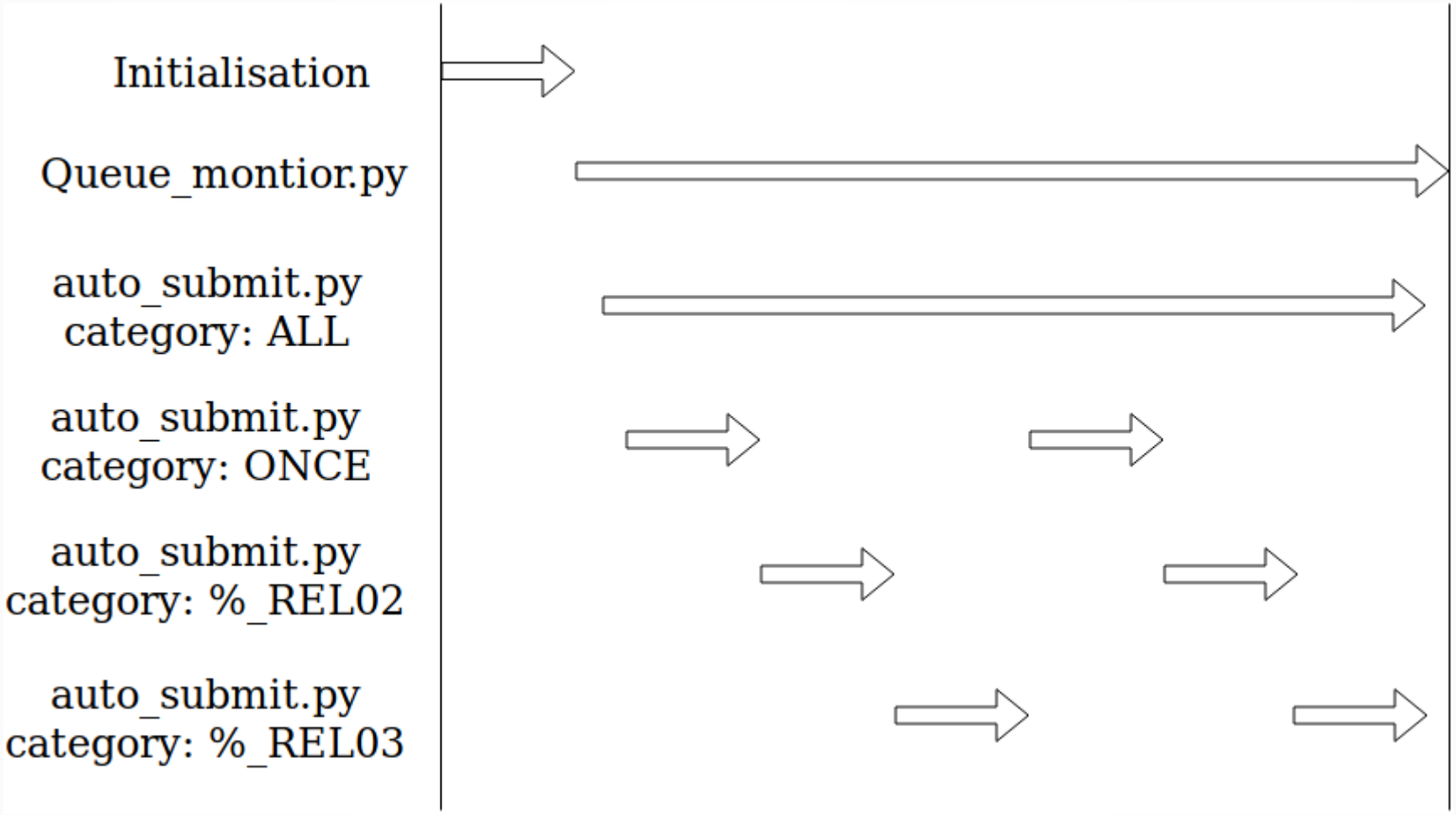
- The task type followed by which realisations to run on
- Task names need to match up with the string from qcore.constants
- Realisation keywords:
  - Once: Runs for the first realisation in each fault (Using _RELXX naming)
  - ALL: Runs for all realisations
  - “” strings: Allow for custom specification, the same as auto submit
- Task dependencies are not automatically managed, you need to explicity state all tasks to be run

```
EMOD3D: ALL
HF: ALL
BB: ONCE
IM_calc: ALL
clean_up: ALL
LF2BB: "%_REL03"
HF2BB: "%_REL02"
```

Threading

- A thread is created for queue monitor to run in.
- Queue monitor will run until all other threads complete
- A thread is also created for auto submit tasks that are to be run by all realisations
- The main thread runs any tasks that are to be run by a limited number of realisations
- Python can only run one thread at a time
- Because each script sleeps for about 5 seconds between each loop, allowing each thread to run while the others are sleeping

Automated wrapper process flow



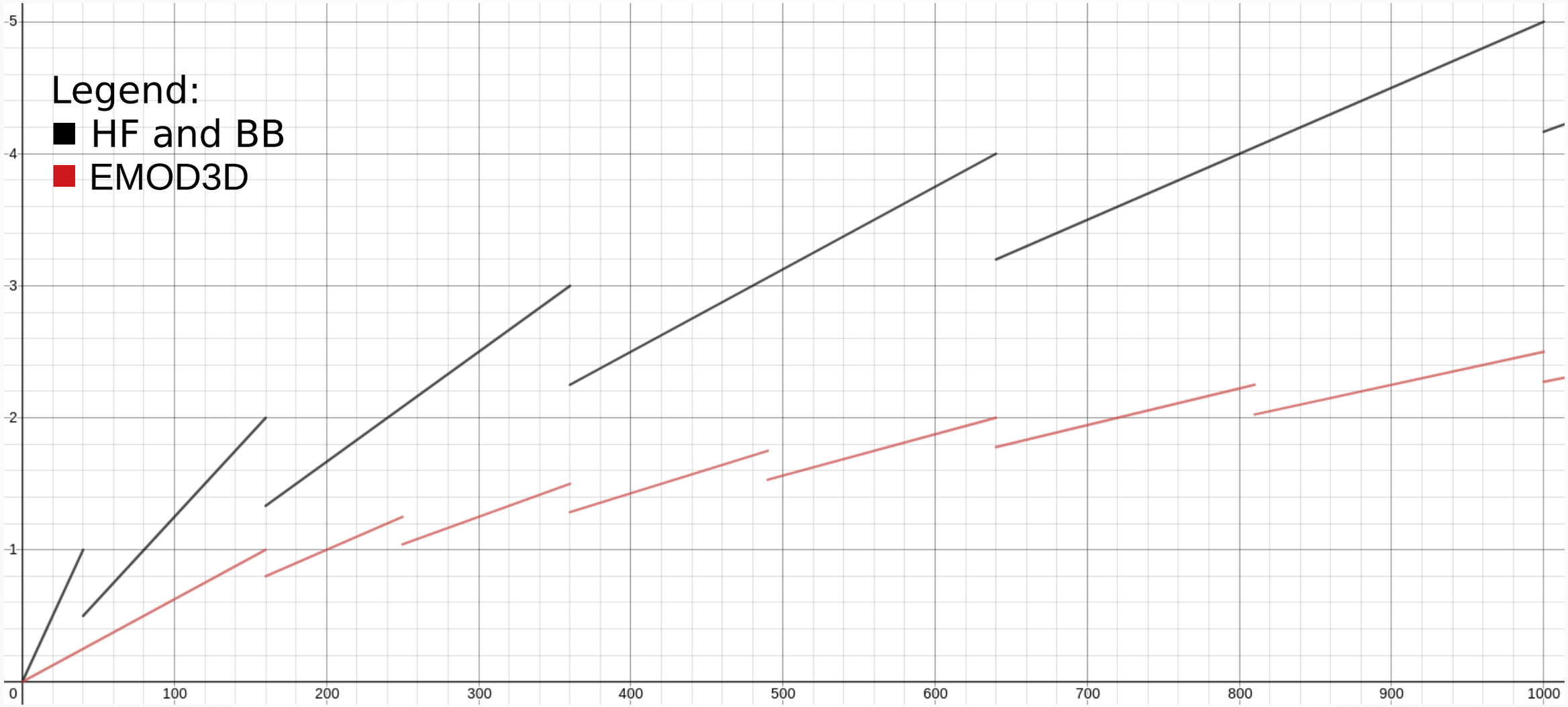
Task submission

- When a task is selected for submission a slurm script must be made, or a generic one used.
- Tasks like EMOD3D, HF, BB have a template that is filled
- Less complicated tasks like clean_up, LF2BB and HF2BB use a generic script with arguments that are passed in
- The duration the task will take is estimated using a neural network

- Task duration estimation is important as an underestimate will cause the task to terminate before completion, while over estimation may cause the task to spend longer in the queue as shorter jobs have a higher priority
- If EMOD3D or HF terminate before completion, usually due to under estimating how long the will take, they are able to resume from a check point, reducing the number of core hours wasted

- If a task has a significant run time, then the number of cores used to run the task will be increased until the run time is deemed acceptable
- The current configuration for this is that jobs estimated to take over one hour in wall clock time will have cores added, with this threshold increasing as the number of core increases

Wall clock time as a function of core hours



- If the task is to be submitted to a different machine than the host then the name of the environment is passed in, and the slurm script is responsible for loading the environment on the remote machine
- Currently this only works when submititng to Mahuika from Maui
- As a result it is currently recommended to submit jobs from Maui

Logging

- All automation scripts have logging
- This provides a stored record of what occured during a simulation run, allowing for debugging if an error occured and in some cases metadata collection
- The time of the event, recording function and message are alway recorded
- If the message is recorded in a code section relating to a specific task and realisation combination they will be recorded
- If the automated wrapper script is used then the name of the thread making the log will be recorded
- If in doubt: Check the logs!

Future plans

- Increased efficiency with current processes
- SRF/VM generation to be added to the workflow
- Installation by auto submit
- More post processing tasks to be added
- A system to determine which machine and partition a task should be sent to
- Greater cross platform submission support