ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

PyMOR Usage

Usage Examples: Modifying your Compute Resources

Paul Gierz

HELMHOLTZ

Motivation

- Depending on the amount of data you have to process, it might be desirable to scale your computing resources
- From the user config, you can very easily:
  - Switch between processing backends (Dask vs. Dask-Jobqueue)
  - Scale how many SLURM worker jobs start
    - Other Jobqueue systems (e.g. PBS) are not yet supported!
  - Disable Dask entirely (good for debugging)

User Config Settings

- `pymor.dask_cluster` can be `slurm` or `local`
- `pymor.dask_cluster_scaling_mode` can be `fixed` or `adapt`
- In adaptive mode, Dask (and SLURM) will automatically add or remove workers for you based upon how much number-crunching you need to do

```yaml
general:
  cmor_version: CMIP6
  CMIP_Tables_Dir: /work/ab0995/a270243/pymor_workshop/cmip6-cmor-tables/Tables/
  CV_Dir: /work/ab0995/a270243/pymor_workshop/cmip6-cmor-tables/CMIP6_CVs/
pymor:
  warn_on_no_rule: False
  dask_cluster: slurm
  dask_cluster_scaling_mode: fixed # TODO: Change this during the exercise
  dask_cluster_scaling_fixed_jobs: 1 #TODO: Change this during the exercise

rules:
  - name: "linear trend example"
    cmor_variable: tas
    experiment_id: "piControl"
```

User Config Settings

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

You can further specify settings for Dask in your configuration yaml!

See Dask Documentation for more:

- https://docs.dask.org/en/stable/configuration.html
- https://docs.dask.org/en/stable/configuration.html#distributed-worker
- https://jobqueue.dask.org/en/latest/clusters-configuration.html

```yaml
# Settings for using dask-distributed
distributed:
  worker:
    memory:
      target: 0.6 # Target 60% of worker memory usage
      spill: 0.7 # Spill to disk when 70% of memory is used
      pause: 0.8 # Pause workers if memory usage exceeds 80%
      terminate: 0.95 # Terminate workers at 95% memory usage
    resources:
      CPU: 4 # Assign 4 CPUs per worker
    death-timeout: 600 # Worker timeout if no heartbeat (seconds): Keep workers alive for 5
# SLURM-specific settings for launching workers
jobqueue:
  slurm:
    name: pymorize-worker
    queue: compute # SLURM queue/partition to submit jobs
    account: ab0995 # SLURM project/account name
    cores: 4 # Number of cores per worker
    memory: 128GB # Memory per worker
    walltime: '00:30:00' # Maximum walltime per job
    interface: ib0 # Network interface for communication
    job-extra-directives: # Additional SLURM job options
      - '--exclusive' # Run on exclusive nodes
      - '--nodes=1'
    # Worker template
    worker-extra:
      - "--nthreads"
      - 4
      - "--memory-limit"
      - "128GB"
      - "--lifetime"
      - "25m"
      - "--lifetime-stagger"
      - "4m"
    # How to launch workers and scheduler
    job-cpu: 128
    job-mem: 256GB
    # worker-command: dask-worker
    processes: 32 # Limited by memory per worker!
    # scheduler-command: dask-scheduler
```

HELMHOLTZ

Exercises

1. Run one of your examples locally if you have been using SLURM. What changes?

2. Try adding more worker nodes by increasing the `dask_cluster_scaling_fixed_jobs` field