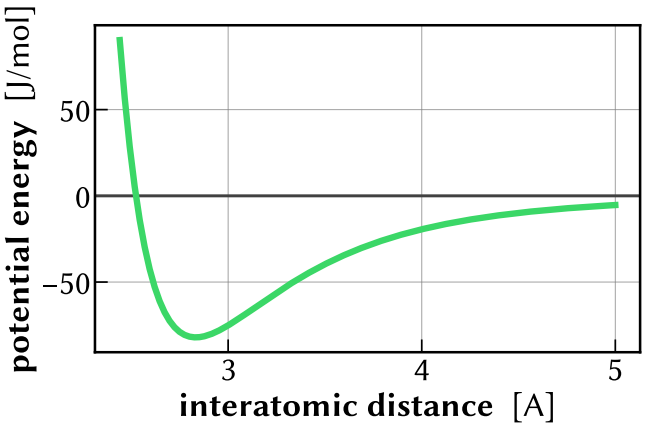
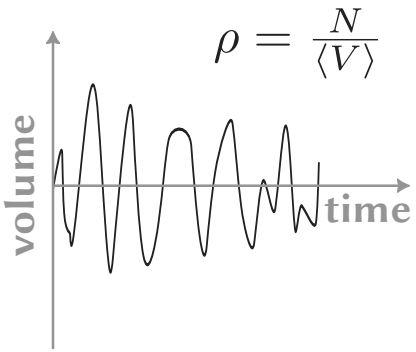
Building with Parsl: Molecular Simulation

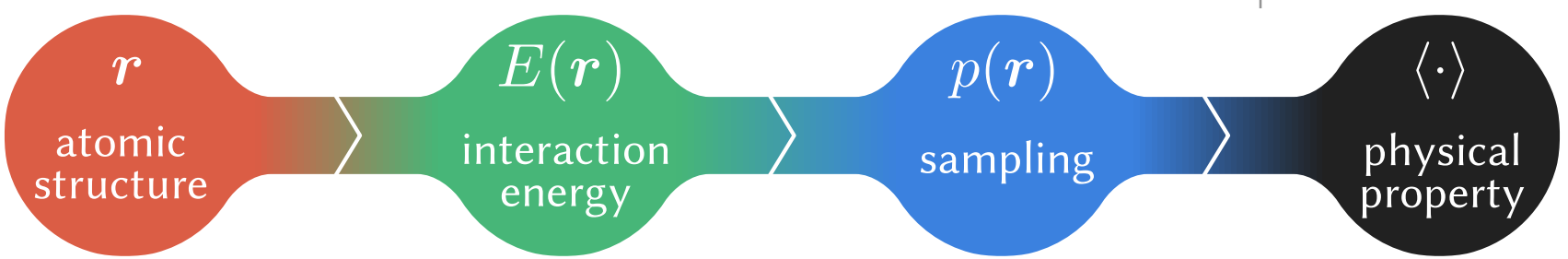
Sander Vandenhaute

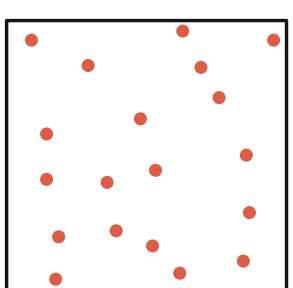
Center for Molecular Modeling, Ghent University (BE)

September 26, 2024







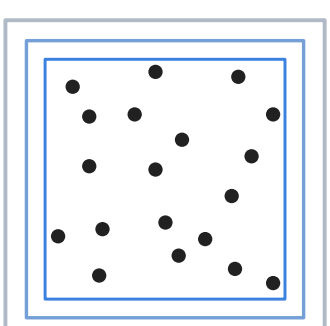


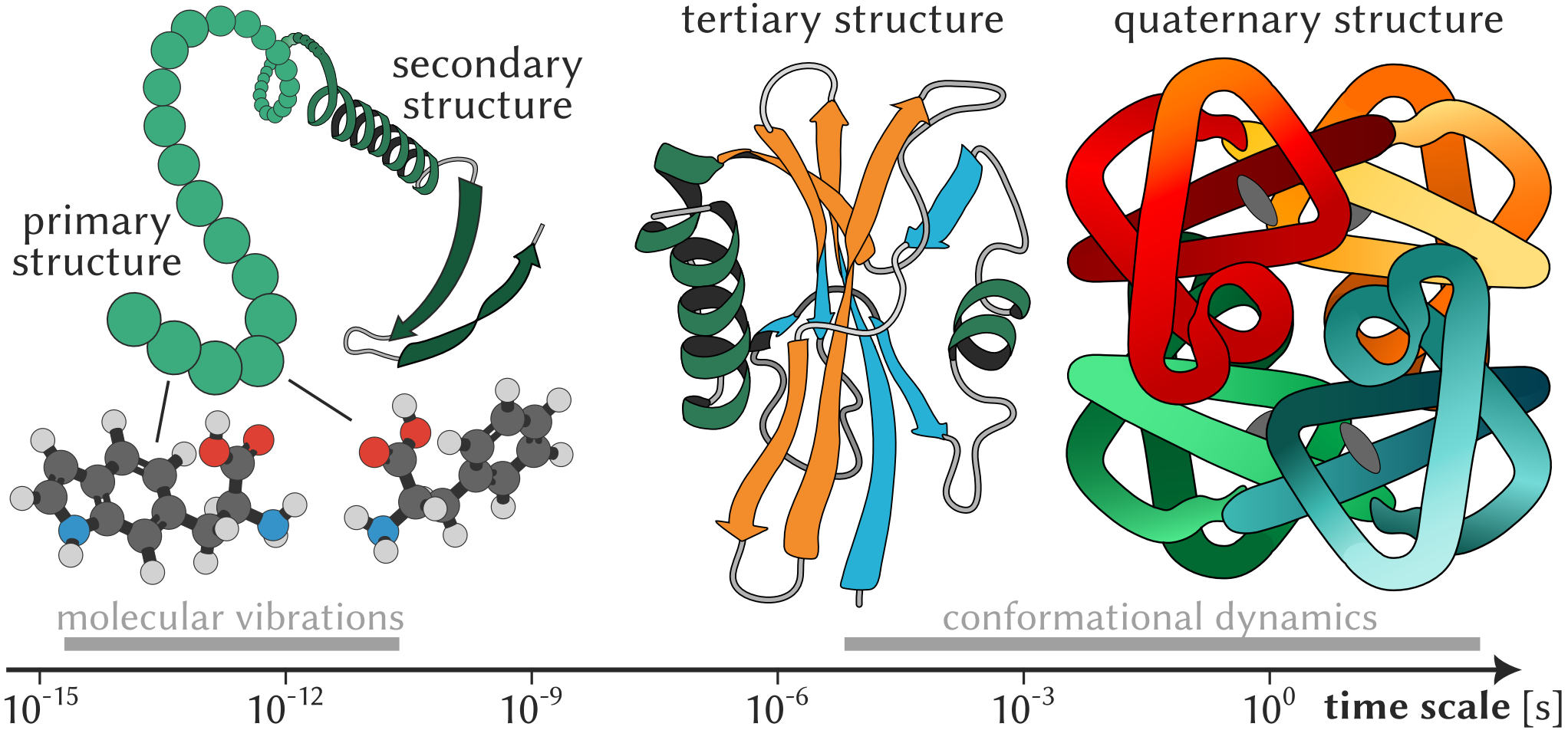
unit cell with N atoms

molecular dynamics

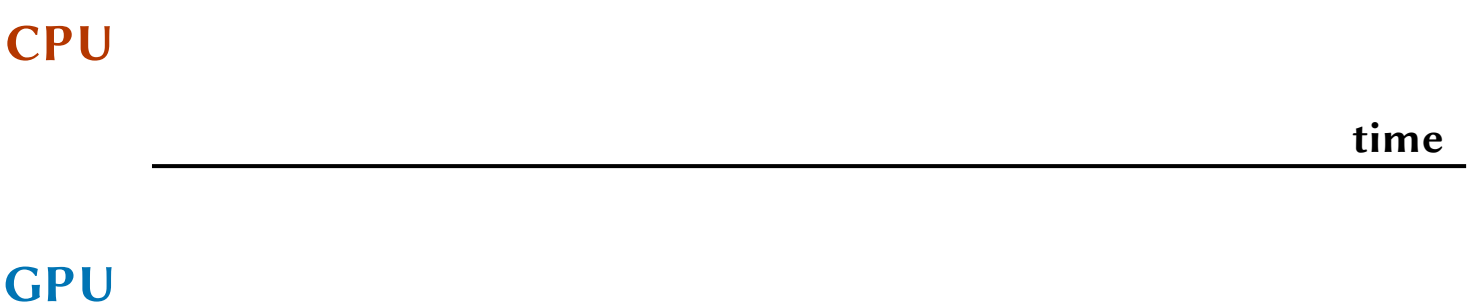
constant N, P, T

variable volume
variable energy




secondary structure
primary structure
tertiary structure
quaternary structure
molecular vibrations
conformational dynamics
10^{-15}
10^{-12}
10^{-9}
10^{-6}
10^{-3}
10^{0}
time scale [s]

Resource requirements: heterogeneous and unbalanced

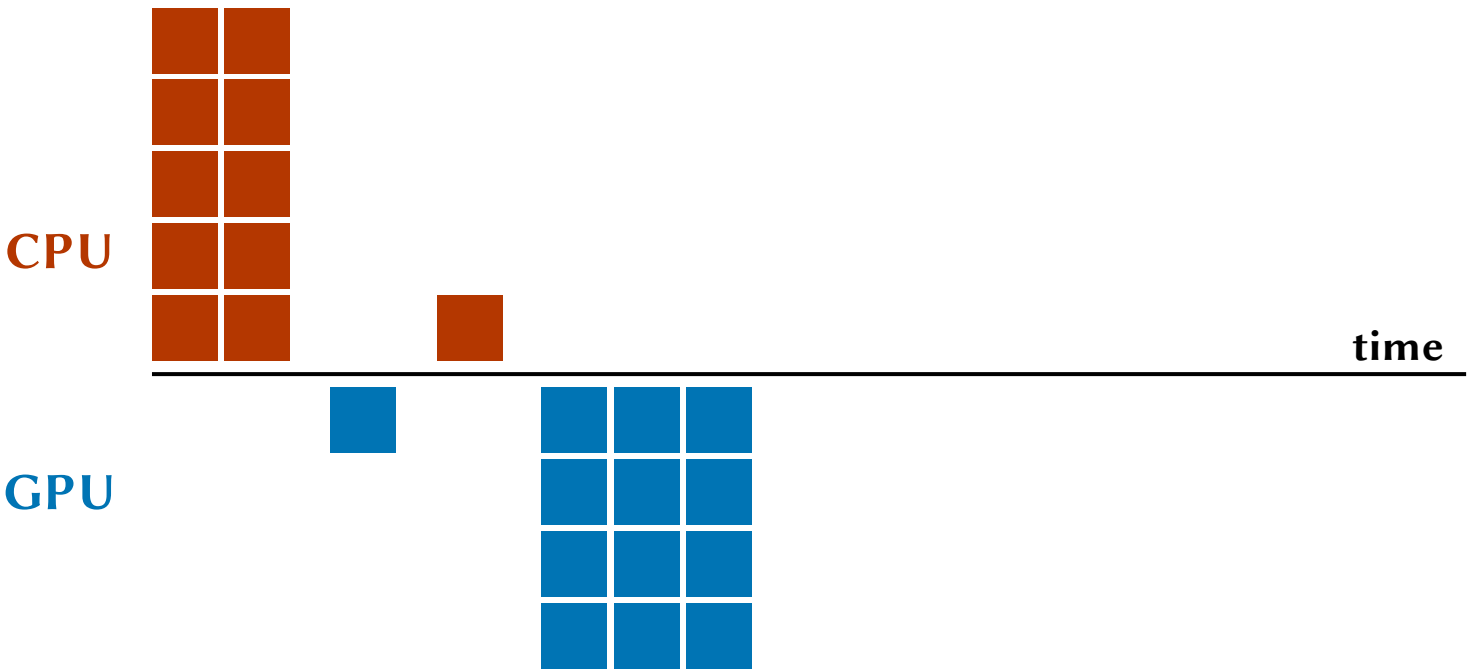


Resource requirements: heterogeneous and unbalanced

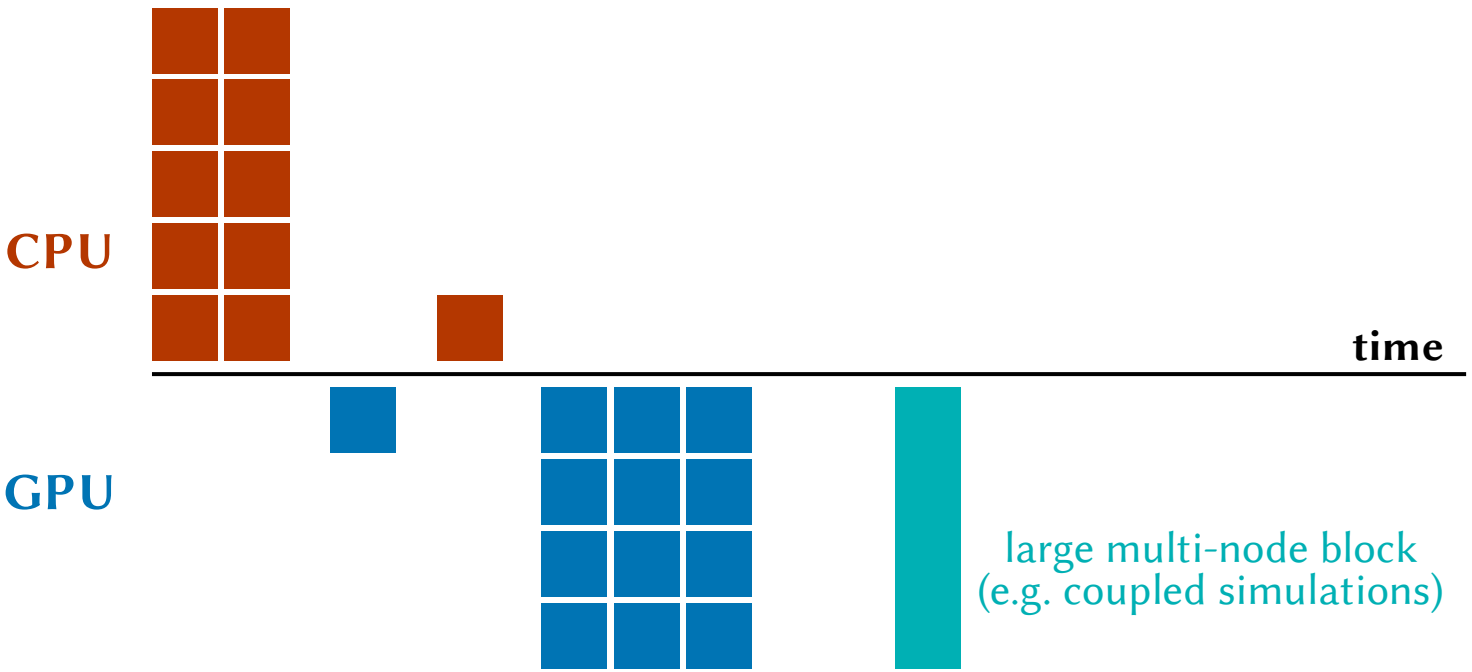
Resource requirements: heterogeneous and unbalanced

Resource requirements: heterogeneous and unbalanced

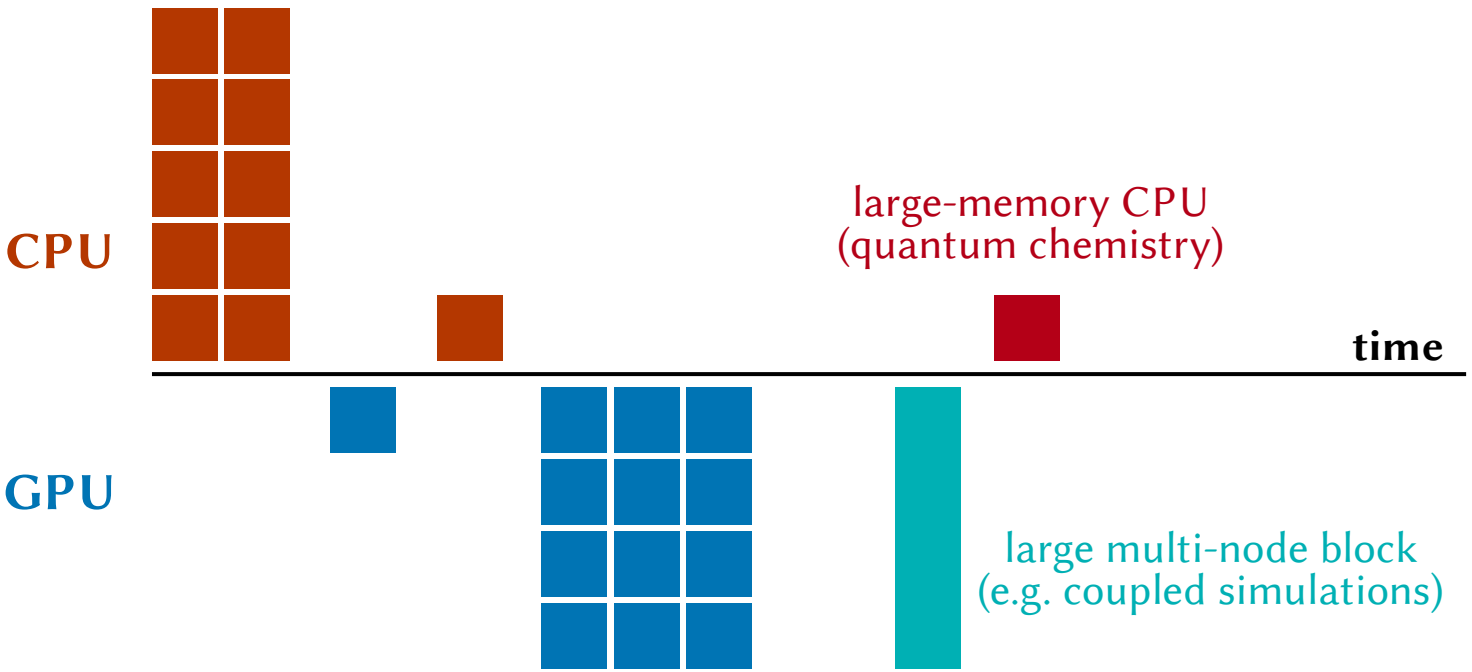
Resource requirements: heterogeneous and unbalanced



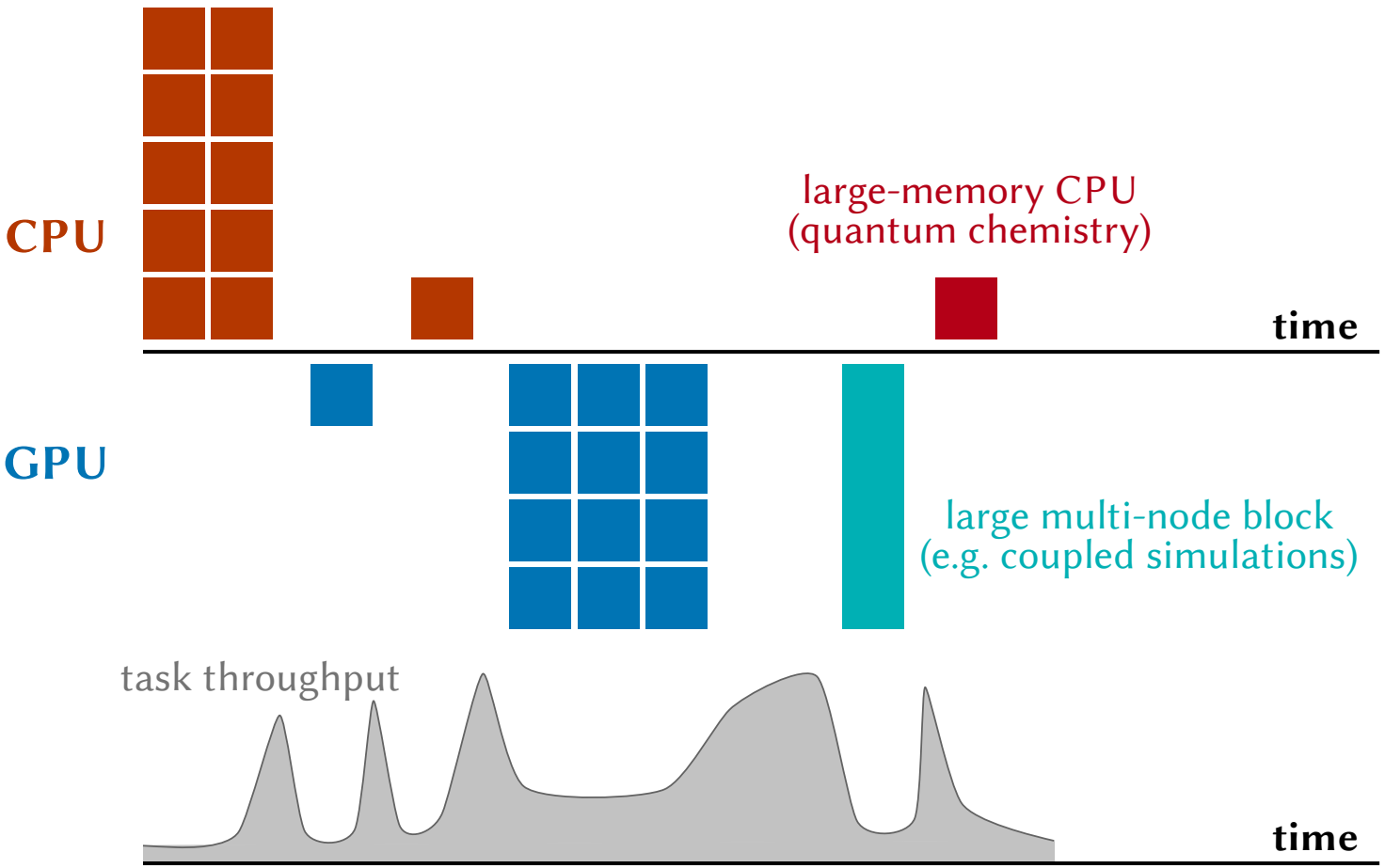
Resource requirements: heterogeneous and unbalanced



Resource requirements: heterogeneous and unbalanced



Resource requirements: heterogeneous and unbalanced



a Parsl-based library for molecular simulation

- **tasks**
  -
  -
- **resources**
  -
  -
- **user**
  -
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  -
- **resources**
  -
  -
- **user**
  -
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  - pre-/post-processing tasks ($10^4 - 10^6$ tasks, data volume > 0)
- **resources**
  -
  -
- **user**
  -
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  - pre-/post-processing tasks ($10^4 - 10^6$ tasks, data volume > 0)
- **resources**
  - heterogeneous: CPU, GPU, large-memory CPU
  -
- **user**
  -
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  - pre-/post-processing tasks ($10^4 - 10^6$ tasks, data volume > 0)
- **resources**
  - heterogeneous: CPU, GPU, large-memory CPU
  - some tasks require multi-node resources
- **user**
  -
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  - pre-/post-processing tasks ($10^4 - 10^6$ tasks, data volume > 0)
- **resources**
  - heterogeneous: CPU, GPU, large-memory CPU
  - some tasks require multi-node resources
- **user**
  - scientific researcher – barely knows Python
  -

a Parsl-based library for molecular simulation

- **tasks**
  - heavy-lifting tasks with duration $\lesssim$ block walltime | task logs necessary
  - pre-/post-processing tasks ($10^4 - 10^6$ tasks, data volume > 0)
- **resources**
  - heterogeneous: CPU, GPU, large-memory CPU
  - some tasks require multi-node resources
- **user**
  - scientific researcher – barely knows Python
  - dependencies? Python + Parsl, QM software, PyTorch (ROCm/CUDA)

How can users create a Parsl `config` if they don't know Parsl?

Force them into a **template** with `yaml` syntax

tasks in scientific workflow can be boiled down to a few 'types'. Let users define the `Provider` for each type.

dynamics **ML training** **QM calculations**

How can users create a Parsl `config` if they don't know Parsl?

Force them into a **template** with `yaml` syntax

tasks in scientific workflow can be boiled down to a few 'types'. Let users define the `Provider` for each type.

dynamics ML training QM calculations

```yaml
ModelEvaluation:
  ...
  slurm:
    account: "..."
    partition: "gpu"
    nodes_per_block: 8
    ...
```

How can users create a Parsl `config` if they don't know Parsl?

Force them into a **template** with `yaml` syntax

tasks in scientific workflow can be boiled down to a few 'types'. Let users define the `Provider` for each type.

dynamics

```yaml
ModelEvaluation:
  ...
  slurm:
    account: "..."
    partition: "gpu"
    nodes_per_block: 8
    ...
```

ML training

```yaml
ModelTraining:
  ...
  slurm:
    account: "..."
    partition: "gpu"
    nodes_per_block: 1
    ...
```

QM calculations

How can users create a Parsl `config` if they don't know Parsl?

Force them into a **template** with `yaml` syntax

tasks in scientific workflow can be boiled down to a few 'types'. Let users define the `Provider` for each type.

dynamics

```yaml
ModelEvaluation:
  ...
  slurm:
    account: "..."
    partition: "gpu"
    nodes_per_block: 8
    ...
```

ML training

```yaml
ModelTraining:
  ...
  slurm:
    account: "..."
    partition: "gpu"
    nodes_per_block: 1
    ...
```

QM calculations

```yaml
CP2K:
  ...
  slurm:
    account: "..."
    partition: "cpu"
    nodes_per_block: 8
    ...
```

Launchers, Executors? Hardcoded!

Launchers, Executors? Hardcoded!

Dependencies? `Apptainer/Singularity` containers!

API: minimal interaction with Futures

API: minimal interaction with Futures

```
potential = MACEHamiltonian.mace_mp0()  # wraps a DataFuture (pytorch model)

walkers = Walker(start, potential, pressure=0).multiply(32)

outputs = sample(walkers, steps=10000)  # returns list of wrapped DataFutures

data = sum([o.trajectory for o in outputs])   # wraps a DataFuture
labeled = data.evaluate(CP2K(...))            # wraps a new DataFuture
```

API: minimal interaction with Futures

```
potential = MACEHamiltonian.mace_mp0()  # wraps a DataFuture (pytorch model)

walkers = Walker(start, potential, pressure=0).multiply(32)

outputs = sample(walkers, steps=10000)  # returns list of wrapped DataFutures

data = sum([o.trajectory for o in outputs])   # wraps a DataFuture
labeled = data.evaluate(CP2K(...))            # wraps a new DataFuture
```

```
error = compute_rmse(   # THIS IS A FUTURE (of a numpy array)
        labeled.get('forces'),
        potential.compute(data, 'forces'),
)
```

github.com/molmod/psiflow