Schematic Outline

Local Machine

all trajectories stay on HPC

Local FS

input files
initial.pdb
integrator.xml
⋮

read/write

Application

python script or jupyter notebook

adaptive strategy
uses generators
has activation conditions

generators
create task descriptions

outgoing connection
creates tasks

DB Resource

needs to be accessible by application and nodes
can run on local machine

Shared DB

mongodb

file contents
initial.pdb
integrator.xml
⋮

MD parameters

file locations on HPC
meta information

model parameters

task descriptions
states ['*queued*', '*running*', …]
assigned workers
required/create/modified files

logs

creation and execution of tasks
is completely independent

consume tasks
outgoing connection

outgoing connection

HPC Resource

all trajectories stay on HPC

Shared FS

project files
traj000000.xtc
traj000000.stc.restart
traj000001.xtc
⋮

staged files
initial.pdb
integrator.xml
⋮

read/write

Nodes

python script

adaptivemdworker
execute tasks as bash script
pull task descriptions
stage files to/from shared FS
add files/models to DB

adaptivemdworker

⋮

Nodes

number of workers can change at will

Some Code Examples

currently python 2.7

Project Setup

set shared fs path and hpc specifics like module loading, etc

```
project = Project('alanine-example')
project = Initialize(AllegroCluster({options}))
pdb_file = File('initial.pdb')
engine = OpenMMEngine(pdb_file, …)
```

Run trajectory

pick pdb_file as initial coordinates

```
traj1 = project.new_trajectory(initial=pdb_file, length=100)
task1 = engine.run(traj1)
project.queue(task)
```

From frame

pick frame 28 from traj1 as initial coordinates

```
traj2 = project.new_trajectory(initial=traj1[28], length=50)
project.queue(traj2)
```

a shortcut that will generate the task

Submit worker

```
adaptivemdworker -d {path_to_db} alanine-example
```

Make strategy

```
@event
def strategy(length, loops):
  for _ in range(loops):
    new_frames = {your way to pick new frames}
    trajs = [project.new_trajectory(f, length) for f in new_frames]
    tasks = [engine.run(t) for t in trajs]
    project.queue(tasks)
    yield [x.is_done for x in tasks]

project.add_event(strategy(100, 10))
```

runs asynchroneously

use yield to halt execution until all given functions evaluate to true, then continue

```
if all(x.is_done() for x in tasks) -> continue
```

Where to get it ?

Status

```
pre-alpha release
```

first beta planned for end of march

GitHub

feel free to contribute

currently python 2.7

```
https://github.com/markovmodel/adaptivemd
```

Installation

to get the latest version

```
git clone https://github.com/markovmodel/adaptivemd.git
cd adaptivemd/
python setup.py install
```

if you are using conda (not the latest version)

```
conda install -c omnia adaptivemd-dev
```