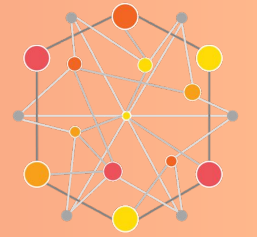
ExaWorks
https://exaworks.org

Tutorial: Developing Robust and Scalable Next Generation Workflows Applications and Systems

PEARC 2022



Lawrence Livermore National Laboratory
Argonne NATIONAL LABORATORY
Brookhaven National Laboratory
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY | Office of Science

RADICAL Cybertools

https://radical-cybertools.github.io/

Introduction

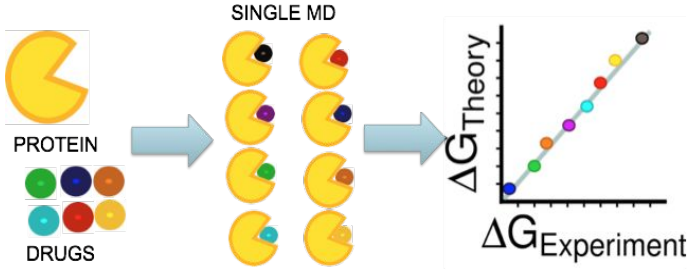
Ensemble-based applications: Applications where the collective outcome of one or more ensembles of tasks is of importance.

Importance of ensemble-based applications:

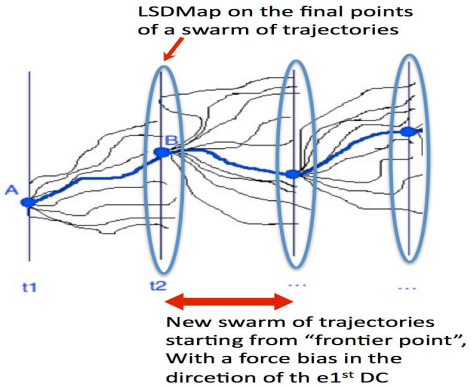
- better algorithms: take advantage of growing **number of processors** on computing infrastructures.

Common in biophysical systems, climate science, seismology, and polar science domains.

RADICAL-Ensemble Toolkit (EnTK): address challenges of **scale, diversity and reliability**



O(1000) simulations to study the binding affinity of multiple drugs on different proteins



O(1000) simulations to understand physical processes of large systems

Advanced Sampling using Adaptive Ensembles

Ensemble-based Adaptive Algorithms:

- runtime data used to determine next stages.

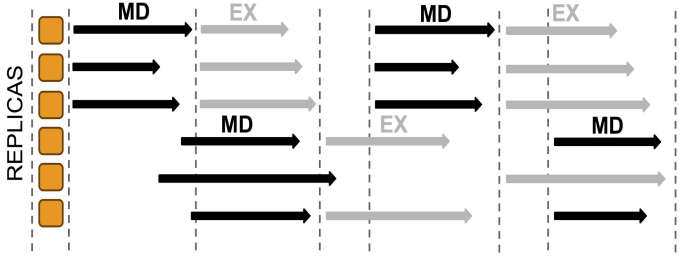
Many biomolecular sampling algorithms formulated as adaptive:

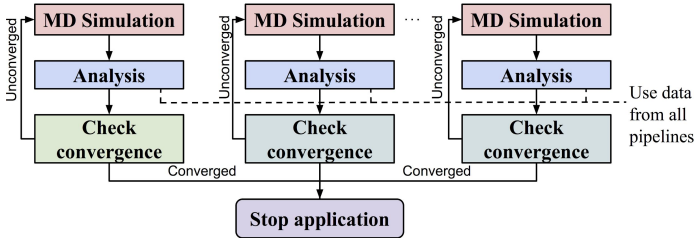
- replica-exchange, Expanded Ensemble, etc.
- Improved simulation efficiency (MSM: $\mathbf{10^3}$).

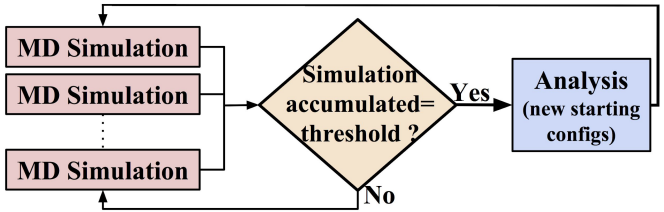
Different types of Adaptivity and coupling:

- task parameter(s), order, count, …

Adaptive logic/MD code should be separated.



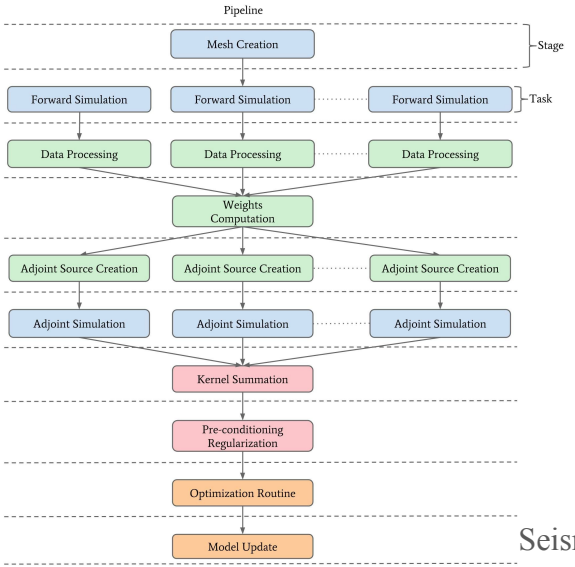




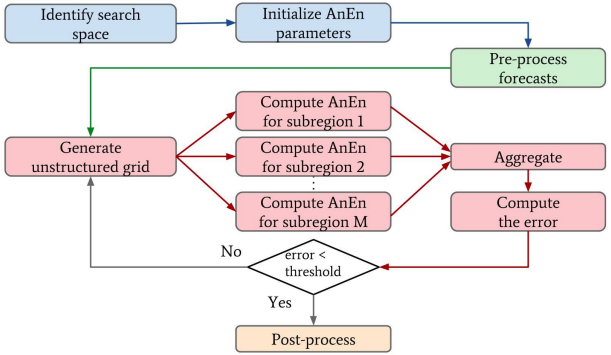
Exemplar Use Cases: Seismic Inversion & Meteorological Forecast

Tomographic technique to study the Earth's interior.

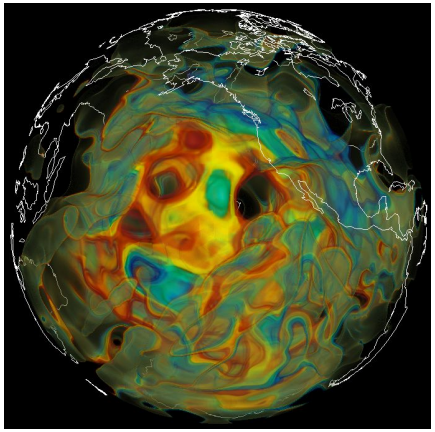
Probabilistic meteorological forecast to study optimal placement of solar power plants.

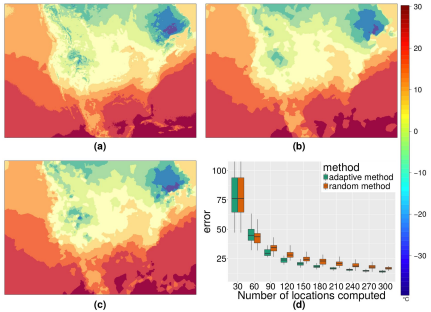


Seismic Inversion



Probabilistic Meteorological Forecasts





RADICAL-EnTK: API Abstractions

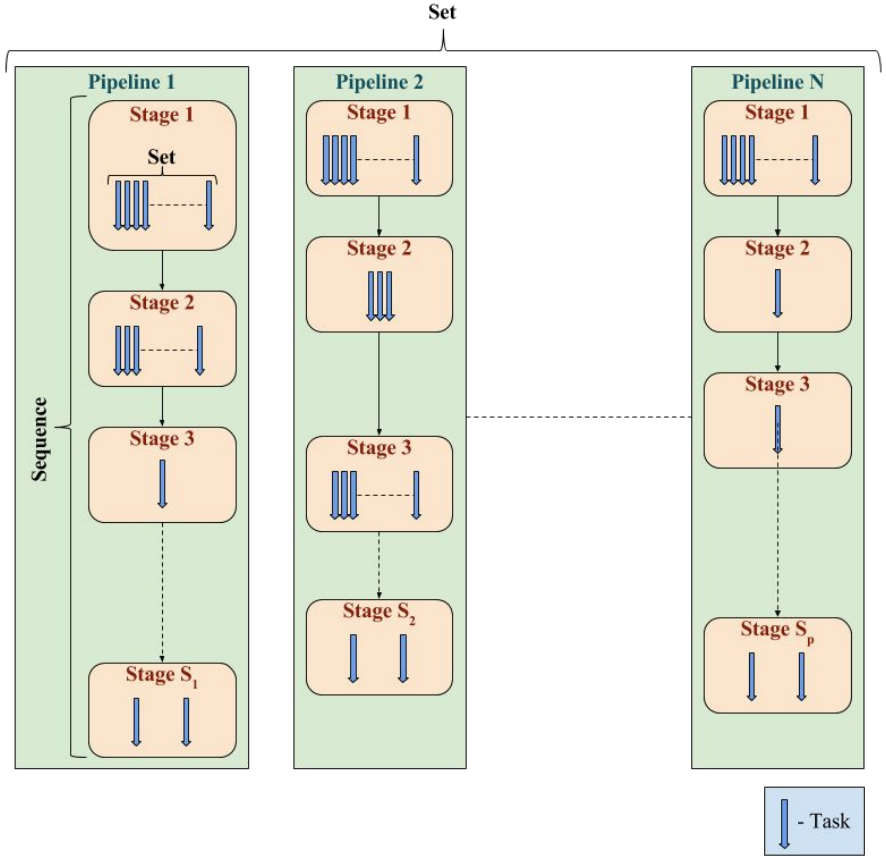
Two (pythonic) collections of objects:

- **Set**: contains objects that have no relative order with each other
- **Sequence/List**: contains objects that have a linear order, i.e. object ‘i’ depends on object ‘i-1’

Three objects:

- **Task**: description of executing kernel
- **Stage**: set of Tasks, i.e. all tasks of a stage may execute concurrently
- **Pipeline**: sequence of Stages, i.e. Stage 2 may only commence after Stage 1 completes

*An ensemble **application** is described as a set of Pipelines.*



RADICAL-EnTK Example 1: Ensemble of Simulation Pipelines

Each simulation pipeline has multiple stages
Simulation stage consists of multiple tasks
Data dependencies exist between stages

Prepare tutorial environment

```
cd $HOME
cd tutorial/
. 1-ensemble-rct.env
cd 1-ensemble-rct
```

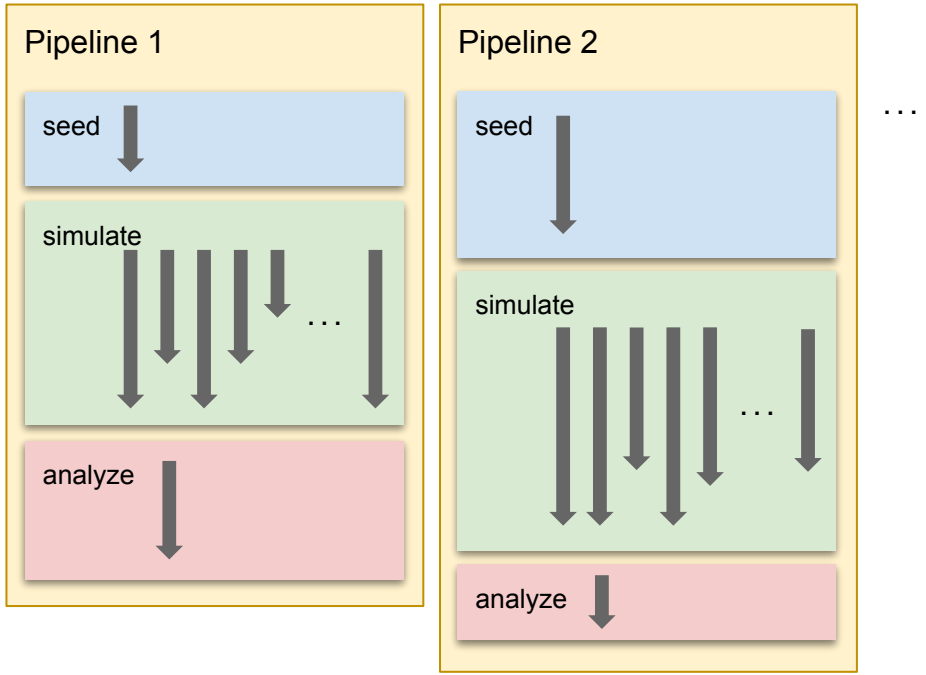
Open first example

```
vim radical_entk_1.py
```

Run example

```
./radical_entk_1.py
```

Original files are backed up in `orig/`



RADICAL-EnTK Example 1: Exercises

1. Switch execution backend to use FLUX instead of default `**fork/exec**`*:
   - Look at the `**appman.resource_desc**` in the program's `**main**` section
   - Solution is in `**solution_1.1.py**`

2. Change the number of ensemble members (number of pipelines) and number of simulations per pipeline:
   - Your are presumably running on a small resource - be gentle ;-)
   - Look at the `**for**` loop in the program's `**main**` section
   - Look at the construction of Stage 2 (`**s2**`)
   - Solution is in `**solution_1.2.py**`

3. Add a fourth stage which computes the square root of the sum:
   - The kernel could be something like: `**echo "sqrt($(cat sum.txt))" | bc**`
   - Output staging should move from previous last stage (`**s3**`) to the new stage (`**s4**`)
   - Solution is in `**solution_1.3.py**`

(*) RCT execution backends include `Slurm`, `LSF`, `APRUN`, `PRTE`, `mpirun/mpiexec`, `ibrun`, etc.

RADICAL-EnTK Example 2: Ensemble of Dynamic Pipelines

After each simulation stage, check results
`if` divergent: re-seed the pipeline
`elif` convergent: finish pipeline
`else`: continue simulation
Stage `post_exec` directives are used to decide on pipeline progression:

- Intermediate data are staged out
- New stages are added dynamically
- Pipelines are completed as needed

Reset tutorial environment

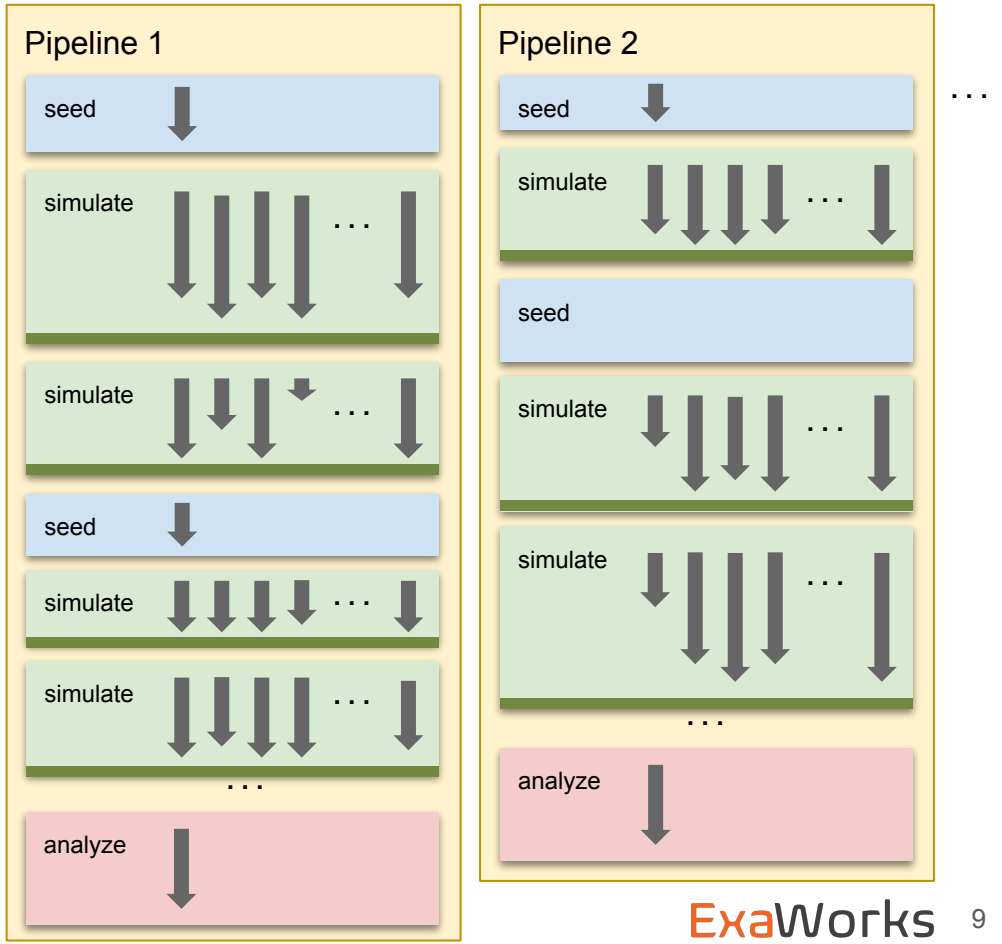
```
make clean
```

Open second example

```
vim radical_entk_2.py
```

Run second example

```
./radical_entk_2.py
```



RADICAL-EnTK Example 2: Exercises

Calculating intermediate results is costly: all data need to be staged back and analyzed. Instead, insert a `stage 2b` which analyzes data on the target resource and only then stages back the result to decide about pipeline continuation.

- This exercise is very similar to previous exercise 3
- `stage 2b` would be very similar to `stage 3`
- Consider what stage then needs to hold the `post_exec`
- Solution is in `solution_2.1.py`

- Note: to avoid very long output, you can turn off the reporter with:

```
export RADICAL_REPORT=False
```

Exercises

https://tinyurl.com/exaworks