OACIS Hands-on(session3)

Discrete-Event Simulation Research Team
RIKEN R-CCS

OACIS Hands-on
2019/6/28 @ Tokyo

Agenda of session 3

- A brief overview on OACIS APIs
  - How to write a Python script which automates the job submission.
- A brief introduction of OACIS-Jupyter integration to make your research note.

References

- API documents
  - http://crest-cassia.github.io/oacis/en/api_python.html
- An introductory Python API sample
  - https://gist.github.com/yohm/ee7e607d63660cf67da31c8bb44f3738
- A sample code used in this tutorial
  - https://gist.github.com/yohm/f01ce95973acc8a6632a56efcb87c73c

What can we do with APIs?

Basically we can conduct any operations on OACIS.
OACIS web-UI is implemented based on these methods.

- creating
- finding
- getting info
- deleting

X

- Simulator
- ParameterSet
- Run
- Host
- Analyzer
- Analysis

Basic Usage of APIs

Set "OACIS_ROOT" environment variable to the path where OACIS is installed.

```
export OACIS_ROOT="$HOME/oacis"
```

(In the docker container, OACIS_ROOT is already set.)

Using Interactive Python

```
$ python3
```

Writing a Python script

```
$ python3 my_script.py
```

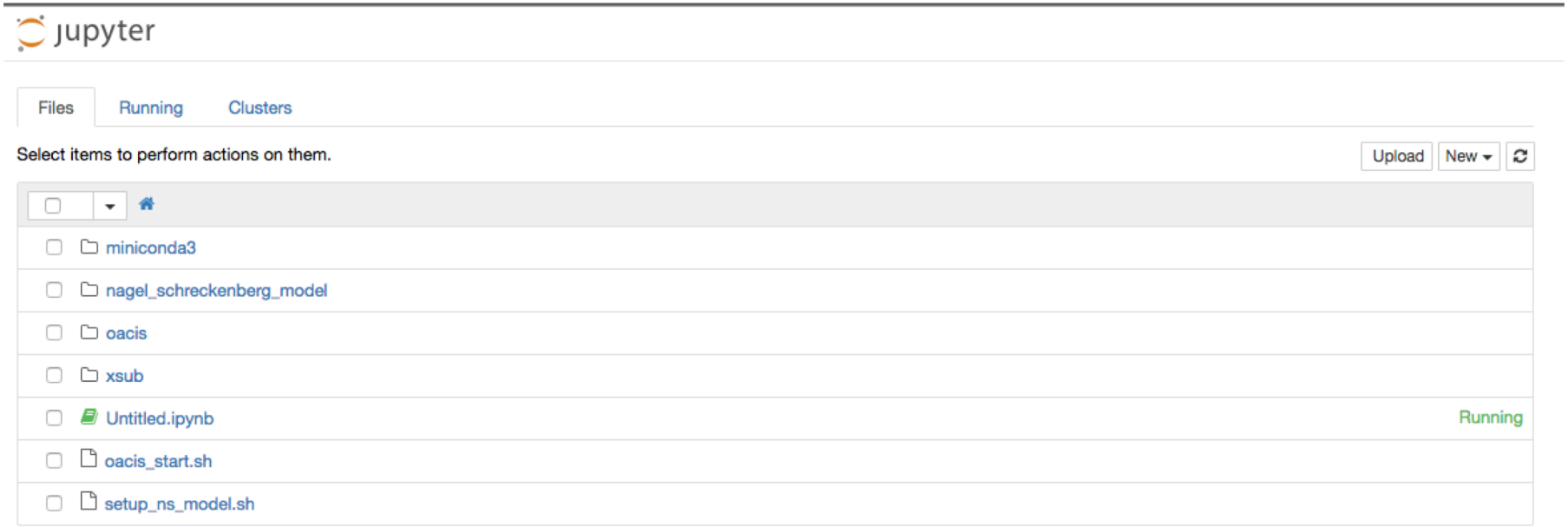
Load "oacis" module.
(In the docker container, PYTHON_PATH is already set. You can skip the first two lines.)

```
import os,sys
sys.path.append( os.environ['OACIS_ROOT'] )
import oacis
```

- A sample of APIs is at
  - http://gist.github.com/yohm/ee7e607d63660cf67da31c8bb44f3738

Using Jupyter Notebook

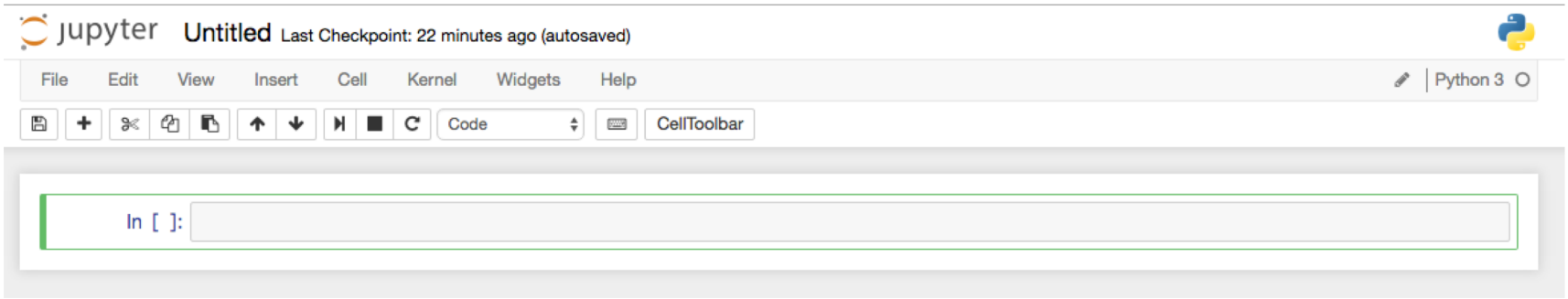
Open http://localhost:8888



Basic Usage of Jupyter

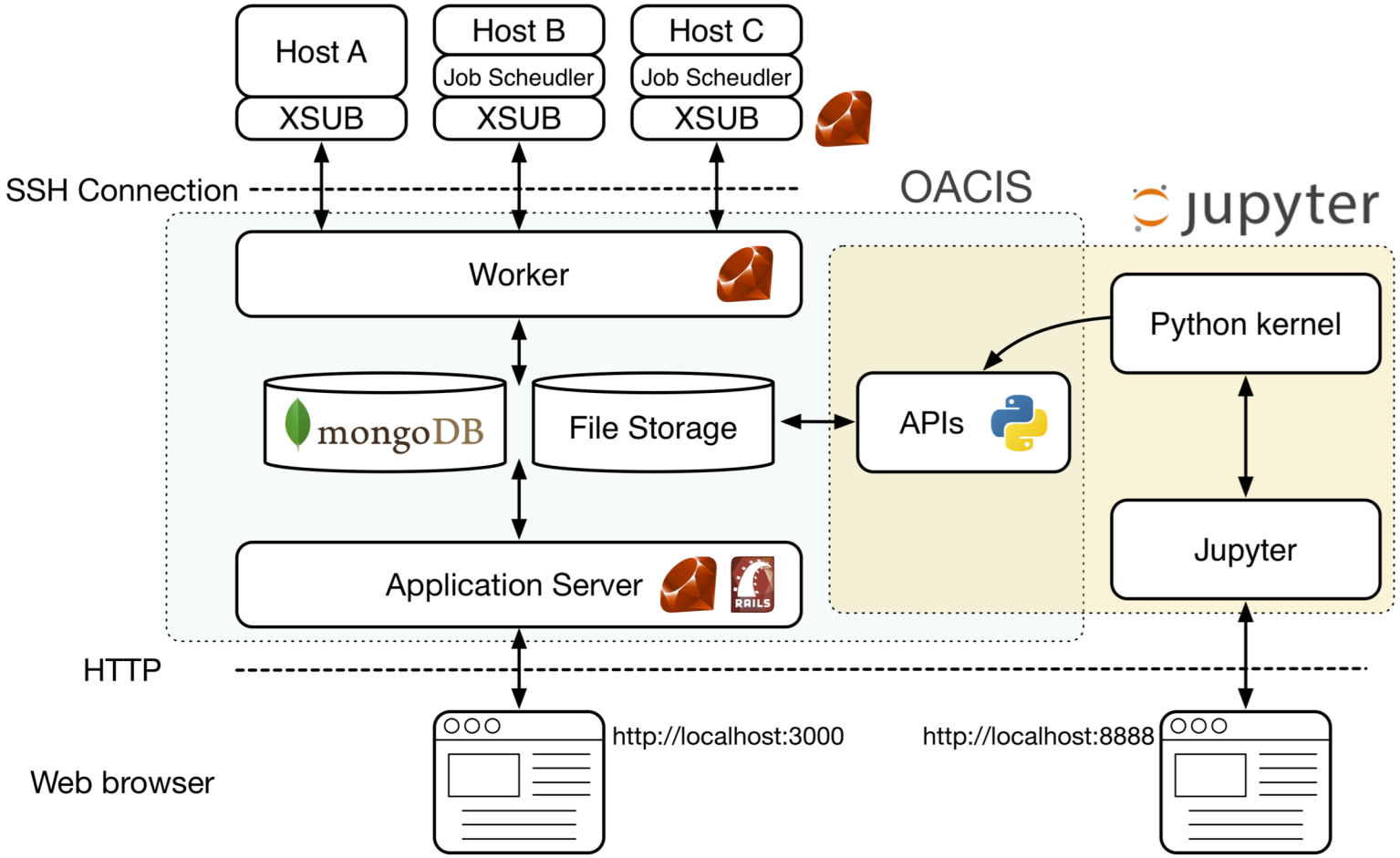
- http://jupyter.org/

- Jupyter is a Python REPL available on a web browser. We can run a python script, and output is shown in the same page.
- We can write texts in markdown format, which lets us to make a "research notebook".



OACIS + Jupyter

We are able to call OACIS Python APIs from Jupyter.



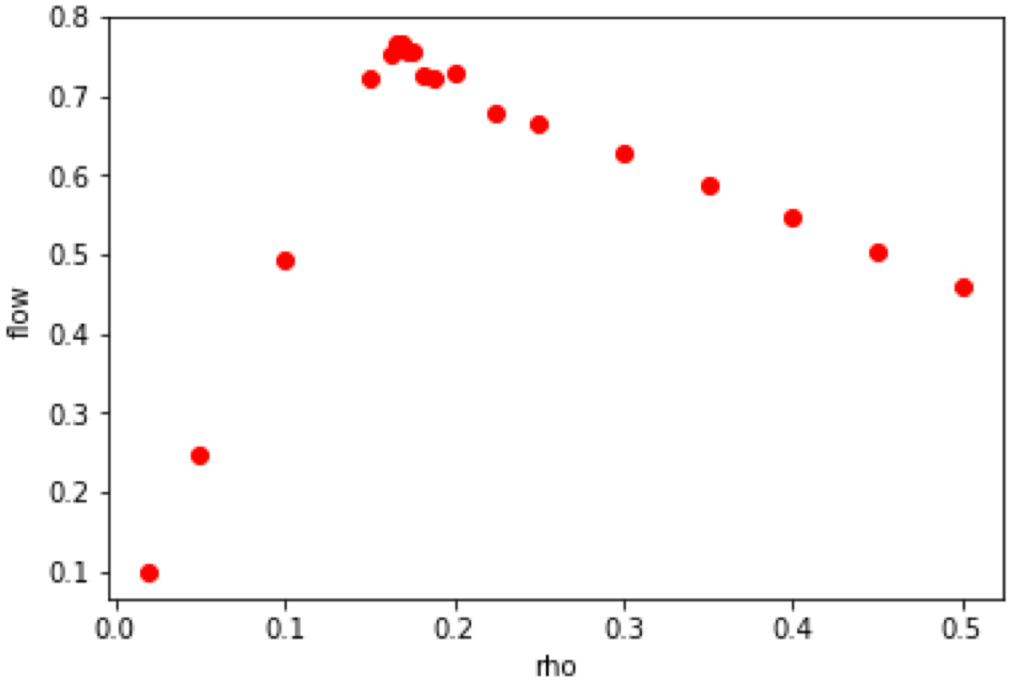
[Hands-on]

1. Import "oacis" module.
2. Find "NS_model" simulator.
3. Create a new ParameterSet.
4. Create a new Run.
5. Check the OACIS web interface and confirm that a new ParameterSet was created.
6. Search ParameterSets where rho=0.2

```
import os,sys
sys.path.append( os.environ['OACIS_ROOT'] )
import oacis

sim = oacis.Simulator.find_by_name("NS_model")

param = {"l":200,"v":10,"rho":0.2,"p":0.1,"t_init":100,"t_measure":300}
ps  =  sim.find_or_create_parameter_set( param )

host = oacis.Host.find_by_name("localhost")
runs = ps.find_or_create_runs_upto(3, submitted_to=host)

found = sim.parameter_sets().where( {"v.rho": 0.2} )
for ps in found:
   print( ps.v() )
```

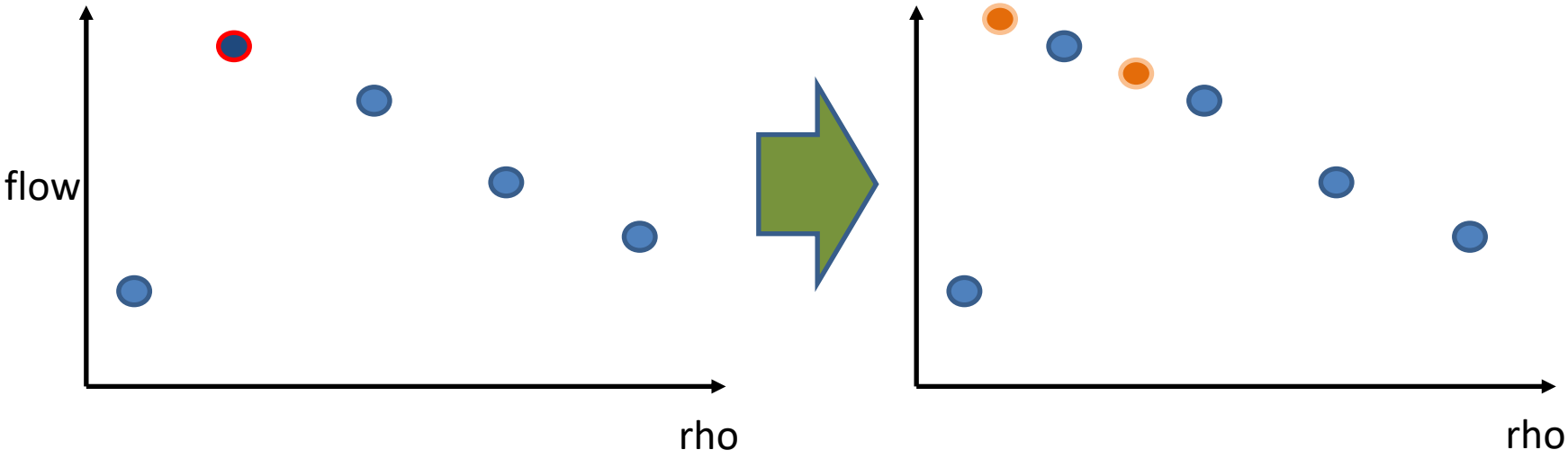
[DEMO] searching optimum rho

- Let us search optimum density for NS model
  - Amount of traffic flow shows non-monotonic behavior against rho.



algorithm

1. Take the PS having the largest "flow".
2. Create two ParameterSets at the centers between its neighboring PSs.
3. Go back to 1 if we do not have an enough resolution.



Async & Await

- We want to iteratively determine parameters based on the results of finished jobs.

http://crest-cassia.github.io/oacis/en/api_watcher.html

```
import oacis
w = oacis.OacisWatcher()

def f1():
    # --- (1)
    oacis.OacisWatcher.await_ps( ps1 )
    # --- (2)
w.async( f1 )

def f2():
    # --- (3)
    oacis.OacisWatcher.await_all_ps( ps_list )
    # --- (4)
w.async( f2 )

w.loop()
```

Use “OacisWatcher” class for asynchronous calls

“f1” and “f2” are asynchronously executed

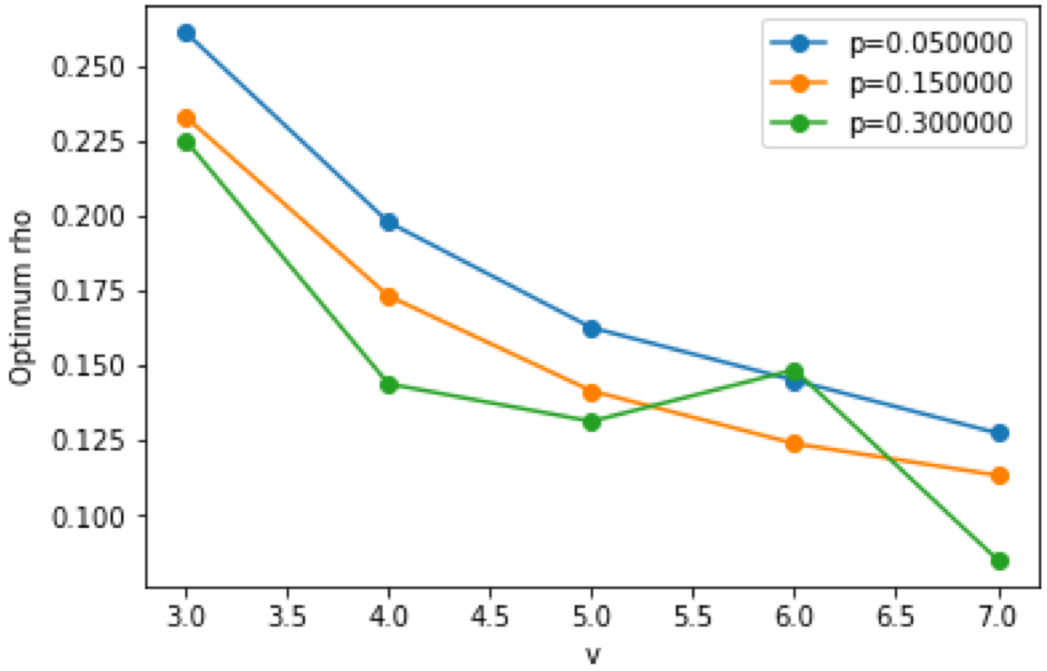
(2) is executed when “ps1” have finished

(4) is executed when all PS in “ps_list” have finished

start an event-loop to monitor the completion of jobs

Code

- See https://gist.github.com/yohm/f01ce95973acc8a6632a56efcb87c73c



[optional] Sample 2

- try candidates parameters iteratively until we get an expected result.
  - e.g. convergence calculation

```
{
  "base": {"p1": "foo"},
  "candidates": [
    {"p2": 1.0, "p3": 1.0},
    {"p2": 1.5, "p3": 2.0},
    {"p2": 2.0, "p3": 2.0},
    {"p2": 2.5, "p3": 3.0}
  ]
},
```

1st candidate

try 2nd candidate if 1st candidate fails

try 3rd candidate if 2nd one fails

Code

- https://github.com/yohm/oacis_sample_iterative_trial_on_candidates

```
def f():
    for cand in candidates:
        param = base_param.copy()
        param.update( cand )
        ps = create_ps_and_run( param )
        oacis.OacisWatcher.await_ps( ps )
        if is_result_satisfactory( ps ):
            print("Found a satisfactory PS : %s" % ps.v() )
            break
w.async( f )
```

iterate over candidates

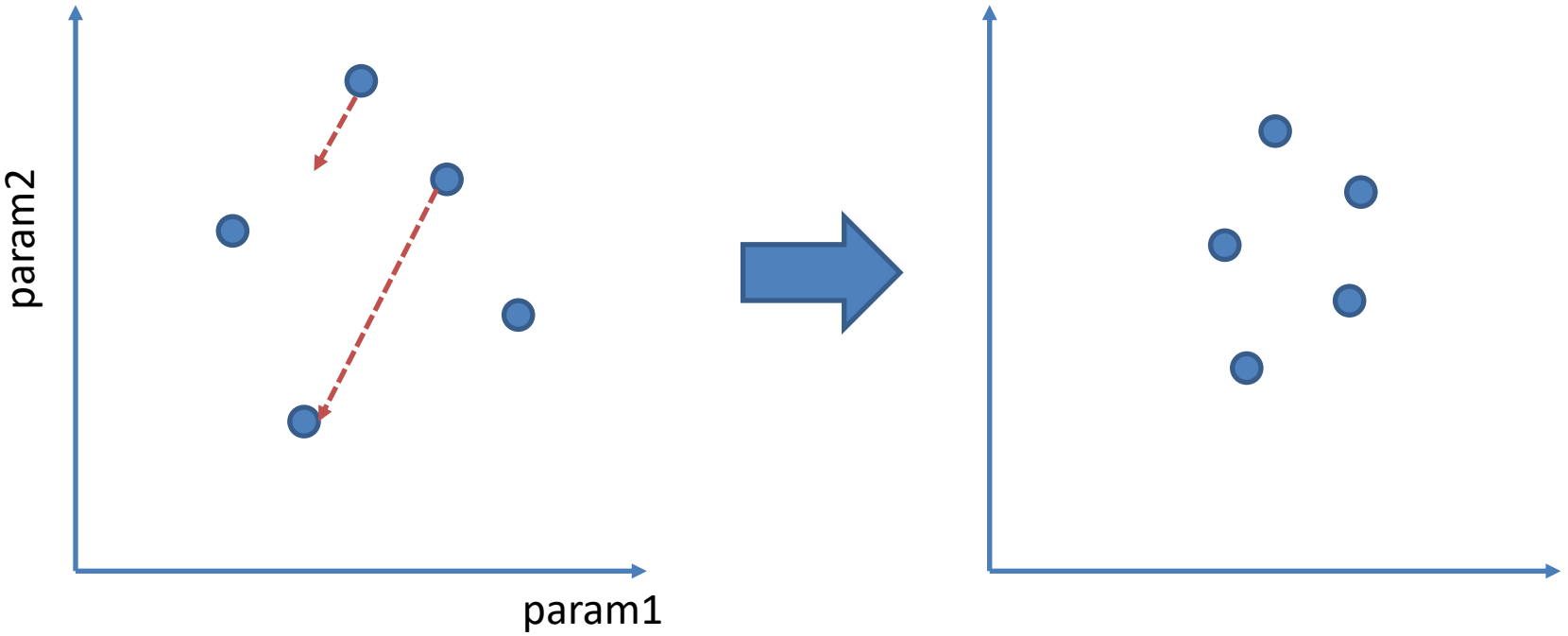
create PS and Run

wait until job finishes

stop iteration if the result is OK

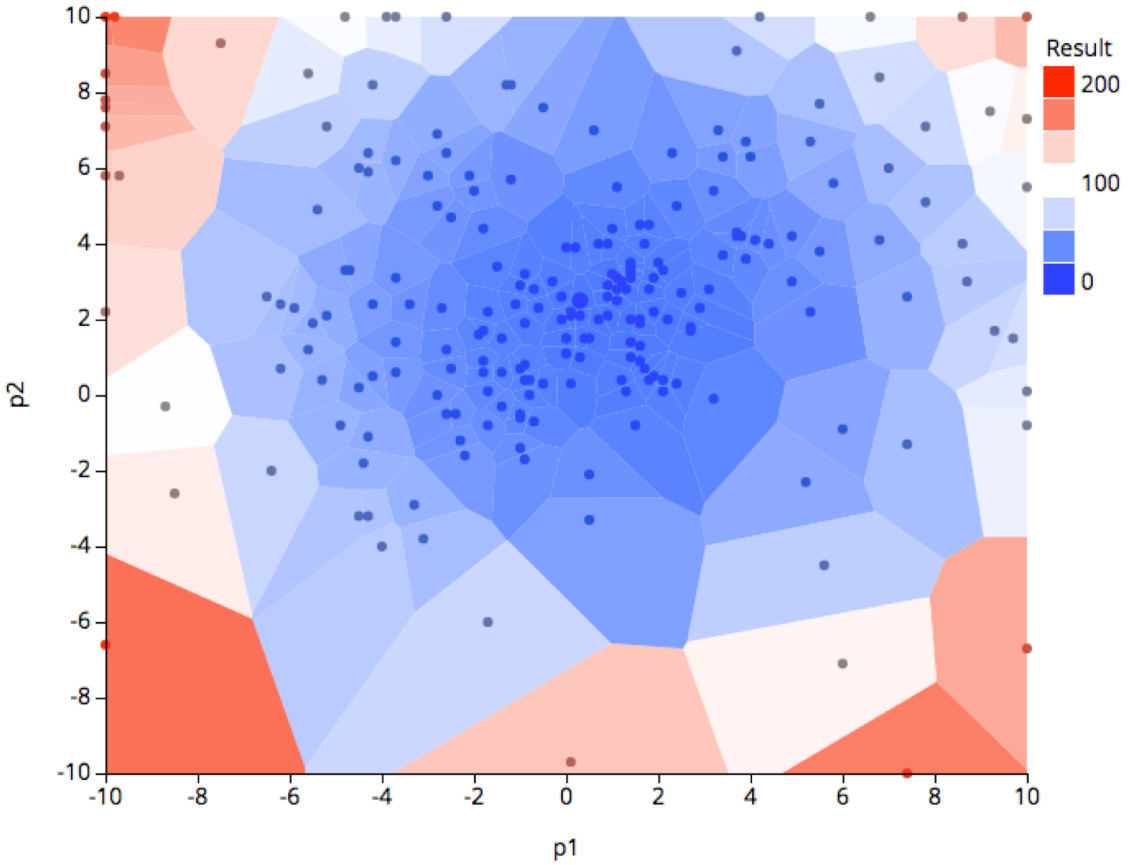
[optional] Sample 3

- Optimization using Differential Evolution Algorithm.
  - DE is a metaheuristic method that optimizes a problem by iteratively trying to improve a candidate solution.



Code

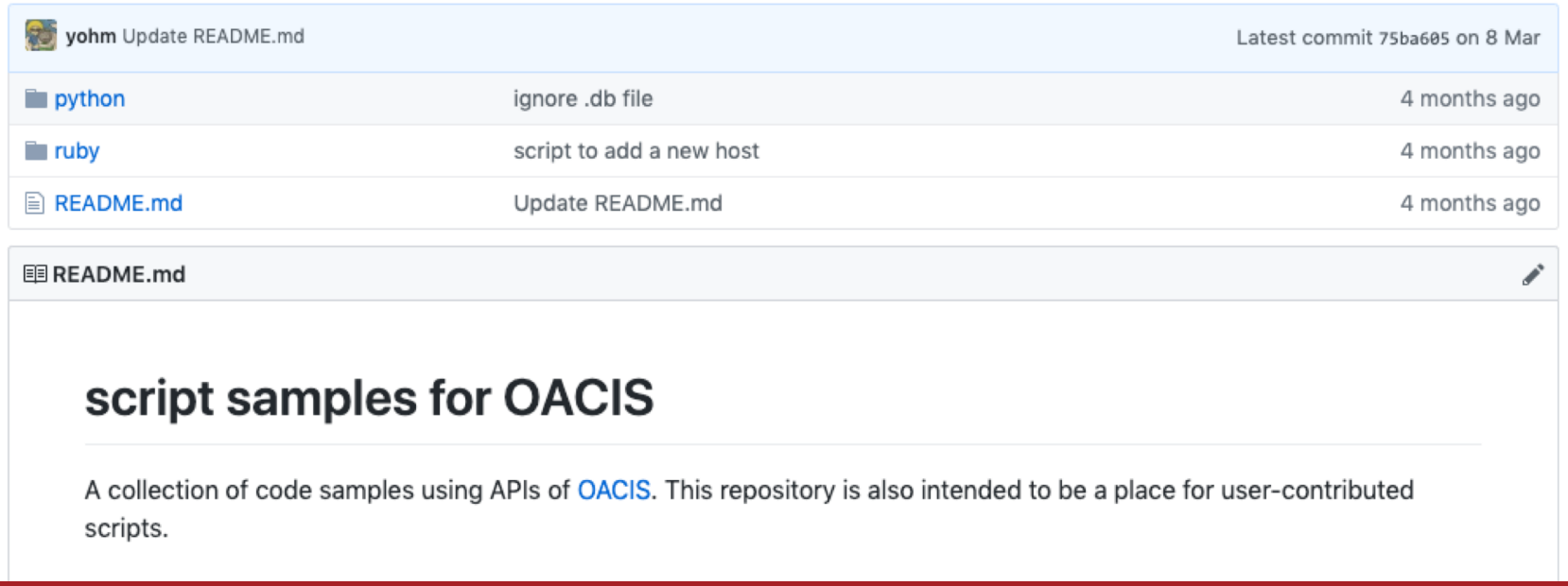
- https://github.com/yohm/oacis_sample_optimize_with_de



Other samples

- A repository collecting OACIS scripts.
  - https://github.com/crest-cassia/oacis_scripts

- We highly welcome your contributions to this repository.



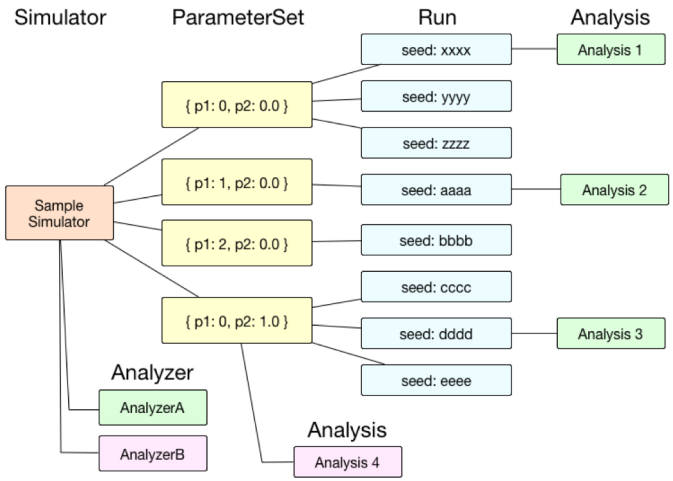
Conclusion

- We briefly demonstrated how to use APIs to automate the workflow.

- Possible applications include parameter sweep, optimization of parameters, sensitivity analysis, Monte-Carlo sampling in parameter space, and applying machine learning to simulation results.

What we skipped

- Installation
  - http://crest-cassia.github.io/oacis/en/install.html
- Analyzer
  - http://crest-cassia.github.io/oacis/en/configuring_analyzer.html
- Backup & Restore
  - http://crest-cassia.github.io/oacis/en/tips.html



Final Remark

- OACIS is an on-going project.
  - We are looking for users, collaborators, and contributors!

- Try using it in your research.
  - If you have any questions or suggestions, please send a mail to
    - oacis-dev@googlegroups.com

- New versions of OACIS are released every two or three months.
  - Please subscribe to our mailing list.
  - https://groups.google.com/forum/#!forum/oacis-users