

Computer simulations create the future

OACIS講習会 (session2)

村瀬洋介, 内種岳詞

理化学研究所 計算科学研究機構

2016/05/11 OACIS公開ソフト講習会



- session1
 - 13:00 – 13:20 概要説明
 - 13:20 – 13:50 シミュレーション実行ハンズオン
- session2
 - 14:00 - 14:30 simulator, analyzer, host登録方法の説明とハンズオン
- session3
 - 14:40 – 15:00 各自の環境に合わせたOACISのセットアップ・運用の相談（希望者）

- Simulatorの登録
 - Simulatorの要件
 - パラメータの渡し方
 - 結果のファイルの取り込み方
 - ジョブ投入時の挙動
 - 登録ハンズオン
- Analyzerの登録
 - Analyzerとは？
 - Analyzerの要件
 - ジョブ投入時の挙動
 - 登録実行デモ
- Hostの登録
 - sshログイン
 - xsub
- コマンドラインインターフェース (oacis_cli)
 - コンテナ(仮想環境)へのログイン
 - oacis_cliコマンドの実行デモ

Simulator 登録

Simulatorの要件

以下の要件を満たすコマンドをSimulatorとして登録できる

- 引数またはJSONでパラメータを受け取ること（次スライドで解説）
- 出力ファイルはカレントディレクトリ以下に作られること
- 以下の名前のファイルがあっても干渉しないこと
 - `_input.json`, `_output.json`, `_status.json`, `_time.txt`,
`_version.txt`, `_log.txt`, `_stdout.txt`, `_stderr.txt`
- 正常終了で0、異常終了で0以外の返り値を返すこと

パラメータの渡し方

- 引数渡しの場合
 - 登録したコマンドにパラメータが順番に引数として渡される。ただし最後は乱数の種
 - `~/simulator.out <p1> <p2> <p3> ... <seed>`
- JSON渡しの場合
 - 実行時に `_input.json` というファイルが作成され、その中にパラメータが記述される
 - `{ “p1” : 30, “p2” : 10, “_seed” : 12345 }`

Simulatorの準備

- 入出力形式をOACISの形式に合わせるため、シミュレーション実行をラップするスクリプトを作成し、そのスクリプトをSimulatorとして登録すると良い。

オプション引数でパラメータを渡す場合の例

```
~/my_proj/my_simulator.out -l 8 -v 0.25 -t 1234 --tmax 2000 --seed 1234
```

スクリプトのサンプル

```
#!/bin/bash
set -e
script_dir=$(cd $(dirname $BASH_SOURCE); pwd)
$script_dir/my_simulator.out -l $1 -v $2 -t $3 --tmax $4 --seed $5
```

詳しくはこちらを参照

http://crest-cassia.github.io/oacis/ja/configuring_simulator.html

出力ファイルの取り込み方

- 基本的にはカレントディレクトリ以下にファイルを出力するようにすればよいだけ
- プロットしたい結果がある場合には
 - “_output.json” というファイル名でJSON形式で出力する
 - キーの名前は任意で良い
- bmp, png, jpgなどの形式で出力しておけば一括閲覧できる (epsは不可)

```
{
  velocity: 0.2584166666666667,
  flow: 0.07752499999999968
}
```


Simulatorの登録

New Simulator

Name

Definition of Parameters
_seed seed -
Random number seed

[Add Parameter](#)

Pre process script

Command

Print version command

Input type

Support mpi

Support omp

Description

Executable_on localhost

Simulatorの名前

Parameterの定義
名前、型、デフォルト値

プリプロセスのコマンド
(詳細後述)

Simulation実行コマンド

バージョン取得コマンド

JSON入力か引数入力か

MPI, OpenMP並列化か

どのようなSimulatorか
についてのメモ

実行可能なホスト

Simulator登録時のTips

- コマンドは、ホームディレクトリからの相対パスで書くのがオススメ
 - 絶対パスだとホストごとに異なる場合がある
- ログインノードでしかできないコマンドはプリプロセスとして登録する
 - 入力ファイルの準備など
- Simulatorのバージョンを記録すると一括削除や置換ができる

- MPI並列の場合、 `mpiexec -n $OACIS_MPI_PROCS ...` というコマンドを実行コマンドとして登録する
- OpenMP並列の場合、 `OMP_NUM_THREADS`環境変数に指定した値が入る

Create a new parameter set on: echo

p1 (Integer)	<input type="text" value="0"/>
p2 (Float)	<input type="text" value="1.0"/>
Target # of Runs	<input type="text" value="1"/>
Submitted to	<input type="text" value="localhost"/>
Priorities of Runs	<input type="text" value="normal"/>
MPI procs	<input type="text" value="4"/>
OMP threads	<input type="text" value="2"/>
	<input type="button" value="Create"/> <input type="button" value="Cancel"/>

Run作成時に指定可能に

ジョブスクリプトの例

```
#!/bin/bash
export LANG=C
export LC_ALL=C

# VARIABLE DEFINITIONS -----
OACIS_JOB_ID=5624ae1364663800e7d40000
OACIS_IS_MPI_JOB=true
OACIS_MPI_PROCS=4
OACIS_OMP_THREADS=2
OACIS_PRINT_VERSION_COMMAND="echo 'version 1'"

# PRE-PROCESS -----
if [ `basename $(pwd)` != ${OACIS_JOB_ID} ]; then # for manual submission
  mkdir -p ${OACIS_JOB_ID} && cd ${OACIS_JOB_ID}
  if [ -e ../${OACIS_JOB_ID}_input.json ]; then
    ¥mv ../${OACIS_JOB_ID}_input.json ./_input.json
  fi
fi
echo "{" > ../${OACIS_JOB_ID}_status.json
echo " ¥"started_at¥": ¥"date`¥", " >> ../${OACIS_JOB_ID}_status.json
echo " ¥"hostname¥": ¥"hostname`¥", " >> ../${OACIS_JOB_ID}_status.json

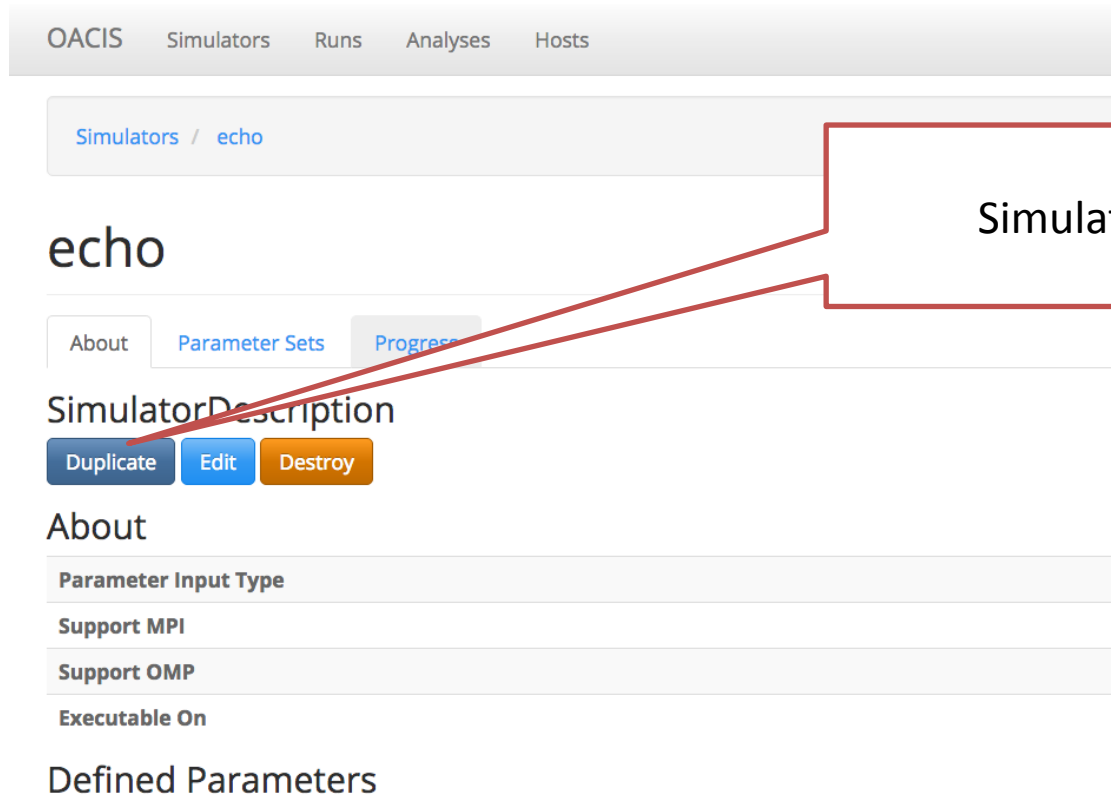
# PRINT SIMULATOR VERSION -----
if [ -n "$OACIS_PRINT_VERSION_COMMAND" ]; then
  (eval ${OACIS_PRINT_VERSION_COMMAND}) > _version.txt
fi

# JOB EXECUTION -----
export OMP_NUM_THREADS=${OACIS_OMP_THREADS}
{ time -p { { echo 3 1.0 1473031699; } 1>> _stdout.txt 2>> _stderr.txt; } } 2>> ../${OACIS_JOB_ID}_time.txt
RC=$?
echo " ¥"rc¥": $RC, " >> ../${OACIS_JOB_ID}_status.json
echo " ¥"finished_at¥": ¥"date`¥" >> ../${OACIS_JOB_ID}_status.json
echo "}" >> ../${OACIS_JOB_ID}_status.json
```

Runの画面から実際に生成されたスクリプトを確認することも可能

Simulatorの複製

- ちょっと仕様の違うSimulatorを新たに登録する場合、既存のSimulatorを複製して作成すると楽。



OACIS Simulators Runs Analyses Hosts

Simulators / echo

echo

About Parameter Sets Progress

SimulatorDescription

Duplicate Edit Destroy

About

Parameter Input Type

Support MPI

Support OMP

Executable On

Defined Parameters

Simulatorの複製

Simulator 登録ハンズオン

- シミュレータの確認 (cli)
 - コンテナ内にログイン (利用者の環境へログイン)

```
docker exec -it -u oacis oacis_tutorial bash -l
```

- シミュレータの実行

```
~/nagel_schreckenberg_model/run.sh 200 5 0.3 0.1 100 300  
12345
```

Key	Description
l	Road length
v	Maximum velocity
rho	Car density
p	deceleration probability
t_init	thermalization steps
t_measure	measurement steps

Simulator登録ハンズオン

- 登録 (web) : OACIS トップページ 「New Simulator」 ボタンから内容入力

設定項目	内容
Name	MySimulator
Definition of Parameters	L, v, rho, p, t_init, t_measure
Command	~/nagel_schreckenberg_model/run.sh
Input Type	Argument
Executable_on	localhost

Key	Type	Default Val
l	Integer	200
v	Integer	5
rho	Float	0.3
p	Float	0.1
t_init	Integer	100
t_measure	Integer	300

画面内の
Add Parameter
をクリックして
パラメータ追加

- Simulatorの実行 (web)
 - Settion1と同じ結果が得られることを確認する
 - ParameterSet作成
 - Run作成
 - 結果の確認
 - Plotの表示

Analyzer 登録

Analyzerとは

- Runの実行後に行うポスト処理をAnalyzerという形で登録することができる
 - リモートホストでは実行できない処理、追加で実行したくなった処理をAnalyzerとして登録するとよい
- OACISには2種類のAnalyzerを登録可能。
 - Runに対するAnalyzer
 - 結果の可視化
 - ParameterSetに対するAnalyzer
 - ParameterSet配下のすべてのRunに対する統計処理

Analyzer登録

- Simulator登録とほぼ同様
- 異なるのは実行時に既存のRunの結果が `_input/` にコピーされること
- どのような形式で入力ファイル (Runの結果ファイル) が準備されるかはドキュメントを参照

Edit Analyzer

Name:

Type:

Definition of Parameters

Pre process script:

Command:

Print version command:

Input type:

Support mpi:

Support omp:

Auto run:

Description:

Executable on: localhost

Host for Auto Run:

自動実行

- Runの終了時にAnalyzerを自動的に作成することができる (auto_runフラグ)
 - ParameterSetに対するAnalyzer : Yes/No から選択
 - あるPSに属するRunがすべてfinishedになる度実行
 - Runに対するAnalyzer: Yes/First run only/No から選択

- Analyzer登録
 - Analyzer準備
 - 登録に必要な内容を入力する
- Analyzer実行
 - RunやParameterSetの参照方法を示す
- Analyzer自動実行
 - 自動実行方法を示す

Analyzer登録実行デモ

- Analyzer準備 (cli)
 - ディレクトリ構造を調べる`tree`コマンドの準備

```
docker exec -it oacis_tutorial bash
apt-get install -y tree
exit
```

- Analyzer登録 (web)
 - Runに対するAnalyzerを登録する
 - Simulator Aboutタブページの最下部「New Analyzer」ボタンを押す
 - 内容入力

設定項目	内容
Name	Tree
Type	on_run
Command	tree
Input Type	Argument
Auto run	yes
Executable_on	localhost

- Analyzer実行

OACIS Simulators Runs Analyses Hosts Logs Document


Simulators / Nagel_Schreckenberg / Param:56e10f6263646200dc060000 / Run:56e10f6263646200dc070000

Run

(l=200, v=5, rho=0.3, p=0.1, t_init=100, t_measure=300)
 /home/oacis/oacis/public/Result_development/5625a5533939360088030000/56e10f6263646200dc060000/56e10f6263646200dc070000

About Results **Analyses**

List of Analyses

Show entries 

AnalysisID	analyzer	parameters	status	version	updated_at
No data available in table					

Showing 0 to 0 of 0 entries Previous Next

Create Analysis

Analyzer

submitted_to

Priority

Analyzer手動実行
 (自動実行は新たに作成されたRunのみ)

Analyzer登録実行デモ

- Analyzer実行結果（作業ディレクトリの構造）

– _stdout.txt

```

.
|-- 56e1115863646200dc0d0000_xsub.sh
|_ _input
|   |-- 56e10f6263646200dc070000_xsub.sh
|   |-- _output.json
|   |-- _status.json
|   |-- _stderr.txt
|   |-- _stdout.txt
|   |-- _time.txt
|   |-- _version.txt
|   |-- initial_time_series.dat
|   `-- traffic.png
|-- _input.json
|-- _stderr.txt
`-- _stdout.txt

1 directory, 13 files
  
```

_inputディレクトリにRunの内容がコピーされる

実行時のパラメータ値がファイルで渡される

```

{"analysis_parameters": {}, "simulation_parameters": {"l": 200, "v": 5, "rho": 0.3, "p": 0.1, "t_init": 100, "t_measure": 300, "_seed": 1458507618}}
  
```


Analyzer登録実行デモ

- Analyzer登録(web)
 - ParameterSetに対するAnalyzerを登録する
 - Simulator Aboutタブページの最下部「New Analyzer」ボタンを押す
 - 内容入力

設定項目	内容
Name	TreePS
Type	on_parameter_set
Command	tree
Input Type	Argument
Auto run	yes
Executable_on	localhost

- Analyzer実行

OACIS Simulators Runs Analyses Hosts Logs Document

[Simulators](#) / [Nagel_Schreckenberg](#) / [Param:56e10f6263646200dc060000](#)

Parameter Set

/home/oacis/oacis/public/Result_development/5625a5533939360088030000/56e10f6263646200dc060000

[About](#) [Runs](#) [Analyses](#) [Plot](#)

List of Analyses

Show entries [refresh](#)

AnalysisID	analyzer	parameters	status	version	updated_at
No data available in table					

Showing 0 to 0 of 0 entries [Previous](#) [Next](#)

Create Analysis

Analyzer

submitted_to

Priority

Analyzer手動実行
(自動実行は新たに作成されたParameterSetに限る)

Results

Analyzer登録実行デモ

- Analyzer実行結果（作業ディレクトリの構造）

– _stdout.txt

```

.
|-- 56e1155663646200dc130000_xsub.sh
|-- _input
|   |-- 56e10f6263646200dc070000
|       |-- 56e10f6263646200dc070000_xsub.sh
|       |-- _output.json
|       |-- _status.json
|       |-- _stderr.txt
|       |-- _stdout.txt
|       |-- _time.txt
|       |-- _version.txt
|       |-- initial_time_series.dat
|       |-- traffic.png
|-- _input.json
|-- _stderr.txt
|-- _stdout.txt
2 directories, 13 files
  
```

_inputディレクトリに
Run IDディレクトリが
作成される

各Runの内容は
Run IDディレクトリに
コピーされる

実行時のパラメータ値
とRunのidリストが
ファイルで渡される

```

{"analysis_parameters": {}, "simulation_parameters": {"l": 200, "v": 5, "rho": 0.3, "p": 0.1,
"t_init": 100, "t_measure": 300}, "run_ids": ["56e10f6263646200dc070000"]}
  
```

Analyzer登録実行デモ

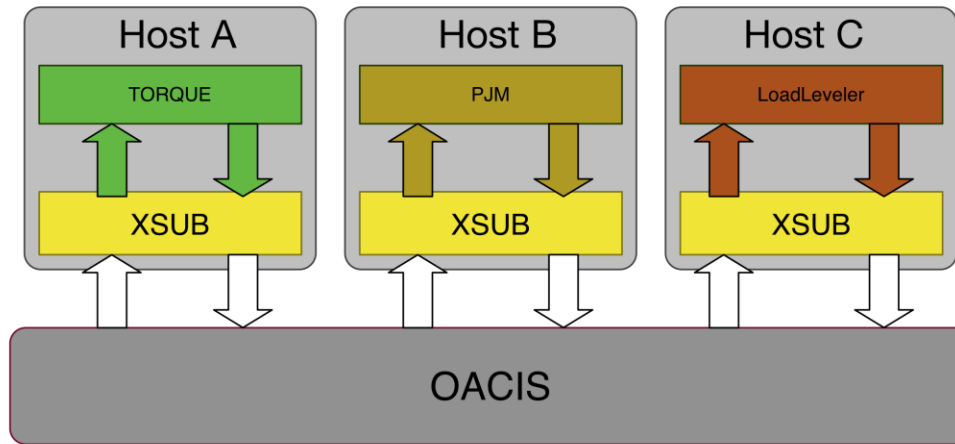
- Analyzer自動実行
 - 自動実行はAnalyzer登録後，新たに作成されたParameterSetやRunに対して実行される
- TreeとTreePSの自動実行デモ
 - 新たなParameterSetとRunを作成
 - Analyzerが自動実行されたことを確認

Host登録

ホストの登録は以下の手順で行う

- 鍵認証でリモートログインできるようにする
 - パスフレーズが必要な場合は、ssh-agent を利用してパスフレーズを入力した状態でOACISを再起動
 - Macであればキーチェーンアクセスにパスフレーズを入力すればOK
- xsubを導入する
- OACISにホストを登録する

xsubとは？



- ジョブスケジューラは仕様が様々で、方言も存在
- その違いを吸収するためのスクリプトをリモートホストにおいて、OACISからは `xsub`, `xstat`, `xdel` という統一したコマンドを呼べるようにしている
- 現在はRubyで実装されている (が、将来はpythonで実装したい…)

Create New Runs

# of Runs	<input type="text" value="1"/>	
Submitted to	<input type="text" value="k"/>	
Priorities of Runs	<input type="text" value="normal"/>	
MPI procs	<input type="text" value="1"/>	
elapse	<input type="text" value="1:00:00"/>	Format: / [^] \d+:\d{2}:\d{2}\$/
node	<input type="text" value="1"/>	Format: / [^] \d+(x\d+){0,2}\$/
shape	<input type="text" value="1"/>	Format: / [^] \d+(x\d+){0,2}\$/

Runの作成時にジョブカードに必要なパラメータが求められるようになる

xsubの導入

- リモートホストで
 - git clone <https://github.com/crest-cassia/xsub.git>
- .bash_profile を編集。(OACISはリモートログインするときにはbashログインシェルからコマンドを実行する)

```
export PATH="$HOME/xsub/bin:$PATH"  
export XSUB_TYPE="torque"
```

- 現状、サポートされているのは、none, torque, FX10, K, SR16000, FOCUSスパコンのみ。必要に応じて追加するので、ご相談ください

OACISへのホストの登録

OACIS Simulators Runs Analyses Hosts

New host

Name

Hostname

Polling Status

User

Port

Ssh key

Work base dir

Mounted work base dir

Max num jobs

Polling interval

MPI processes ~

OMP threads ~

Executable Simulators NagelSchreckenberg
 echo

Executable Analyzers

SSHの接続先の情報

ジョブに使うディレクトリ
(この下にディレクトリを掘って
ジョブが実行される)

ジョブを何本まで投げるか

ジョブの完了確認や投入を何
秒ごとに行うか

このホストで実行が許される
MPI,OpenMPの並列数
Runを作るときにチェック

実行可能シミュレーター

- Docker上で稼働しているOACISからホストOS上でシミュレーションを実行する場合

```
docker exec -u oacis oacis_tutorial cat /home/oacis/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys  
chmod 600 ~/.ssh/authorized_keys
```

接続のテスト

```
docker exec -it -u oacis oacis_tutorial ssh $USER@`hostname`  
# ホストOSに接続できていたら exitで抜ける
```

xsubの導入

```
git clone https://github.com/crest-cassia/xsub.git ~/xsub
```

.bashrcを編集し以下の2行を追加

```
export PATH="$HOME/xsub/bin:$PATH"
```

```
export XSUB_TYPE="none"
```

xsubの導入をテスト

```
docker exec -it -u oacis oacis_tutorial ssh $USER@`hostname` 'bash -l -c xstat'
```

ステージングがある場合

- プリプロセスで実行プログラムなど必要なものをすべて、カレントディレクトリにコピー
- 実行コマンドをカレントディレクトリからのコマンドにする

```
mpiexec -n $OACIS_MPI_PROCS ./a.out
```

- カレントディレクトリのファイルすべてをステージイン、ステージアウトする仕様
 - ランクディレクトリ未対応

- PS一括作成

- 100以上のPSを作成するのはwebからでは不可能

- 150通りxRun4個を作成する場合

- v = [1, 2, 3, 4, 5]

- rho =

- [0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]

- p = [0.1, 0.2, 0.3]

```
docker exec -it -u oacis oacis_tutorial bash -l
cd oacis
./bin/oacis_cli create_parameter_sets -s 5625a5533939360088030000 -i
'{"l":200,"v":[1,2,3,4,5],"rho":[0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5],"p":[0.1,0.2,0.3],"t_init":100,"t_measure":300}' -r
'{"num_runs":4,"mpi_procs":0,"omp_threads":0,"priority":1,"submitted_to":"5625a4703939360088000000","host_parameters":null}'
-o ps.json
```

- Simulator idの調べ方

```
mongo --eval "db = db.getSiblingDB('oacis_development'); db.simulators.find({'name':'Nagel_Schreckenberg'}).map (function(u) {return u._id;})[0]"
#ObjectId(" 5625a5533939360088030000 ")
```

- RunやAnalyzerのリプレイス
 - シミュレータの仕様変更 (バージョンアップ) などの理由で既存のRunを同じ条件で再実行
 - 既存のRunは削除される
 - Nagel_Schreckenbergの全Run, 終了したRun, 失敗したRunを入れ替える

```
docker exec -it -u oacis oacis_tutorial bash -l
cd oacis
# all runs
./bin/oacis_cli replace_runs -s 5625a5533939360088030000 -q simulator_version:version1
# runs where they are finished
./bin/oacis_cli replace_runs -s 5625a5533939360088030000 -q simulator_version:version1 status:finished
# runs where they are failed
./bin/oacis_cli replace_runs -s 5625a5533939360088030000 -q simulator_version:version1 status:failed
```

まとめ

- Simulator登録、Host登録、Analyzer登録のデモを行った
 - これらの登録は煩雑だが、一度登録すれば便利に使うことができる
- メーリングリスト
 - `oacis-users@googlegroups.com`
 - アップデートの通知など
- 開発者への連絡・質問
 - `oacis-dev@googlegroups.com`

⇒ Session 3 では各自の環境に合わせたOACISのセットアップ・運用の相談を受け付けます