



Computer simulations create the future

# OACIS Hands-on (session2)

Yohsuke Murase

Discrete-Event Simulation Research Team

RIKEN R-CCS

OACIS Hands-on

2019/6/28 @Tokyo



# Agenda of Session 2

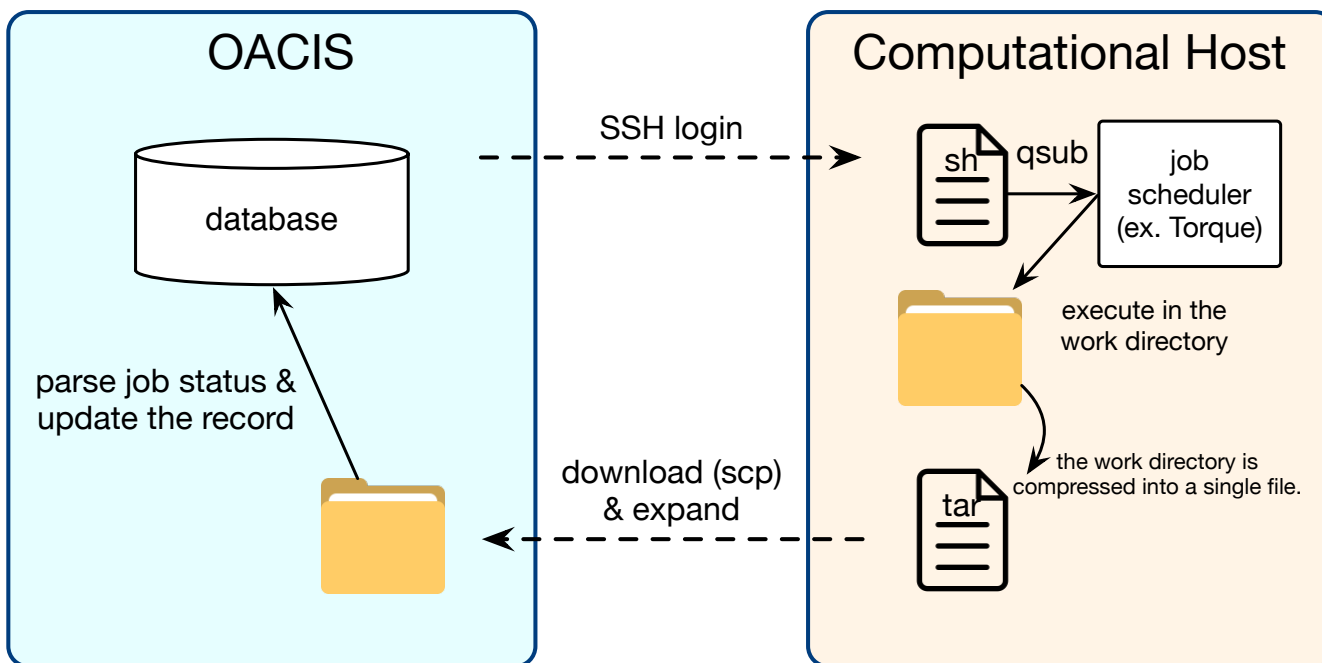
- setting up a Simulator
  - how jobs are executed by OACIS
  - the requirements for Simulator
    - format of input parameters
    - format of output files
  - Hands-on
- setting up Host and HostGroup
  - how OACIS are connected to hosts
  - demonstration: setting up a Host
    - SSH login and authentication using key
    - xsub
    - register a Host on OACIS
  - demonstration: defining a HostGroup
- Q&A

# setting up a Simulator

- Reference
  - [http://crest-cassia.github.io/oacis/en/configuring\\_simulator.html](http://crest-cassia.github.io/oacis/en/configuring_simulator.html)
  - [http://crest-cassia.github.io/oacis/ja/configuring\\_simulator.html](http://crest-cassia.github.io/oacis/ja/configuring_simulator.html)

# how jobs are executed

1. Work directory and shell script are created.  
Jobs are submitted to the scheduler.



3. Results are downloaded to OACIS server.  
The files are parsed and the records are updated.

2. Jobs are executed by the job scheduler.  
Current directory is set to the work directory.

# requirements for Simulator

Simulator must satisfy the following requirements.

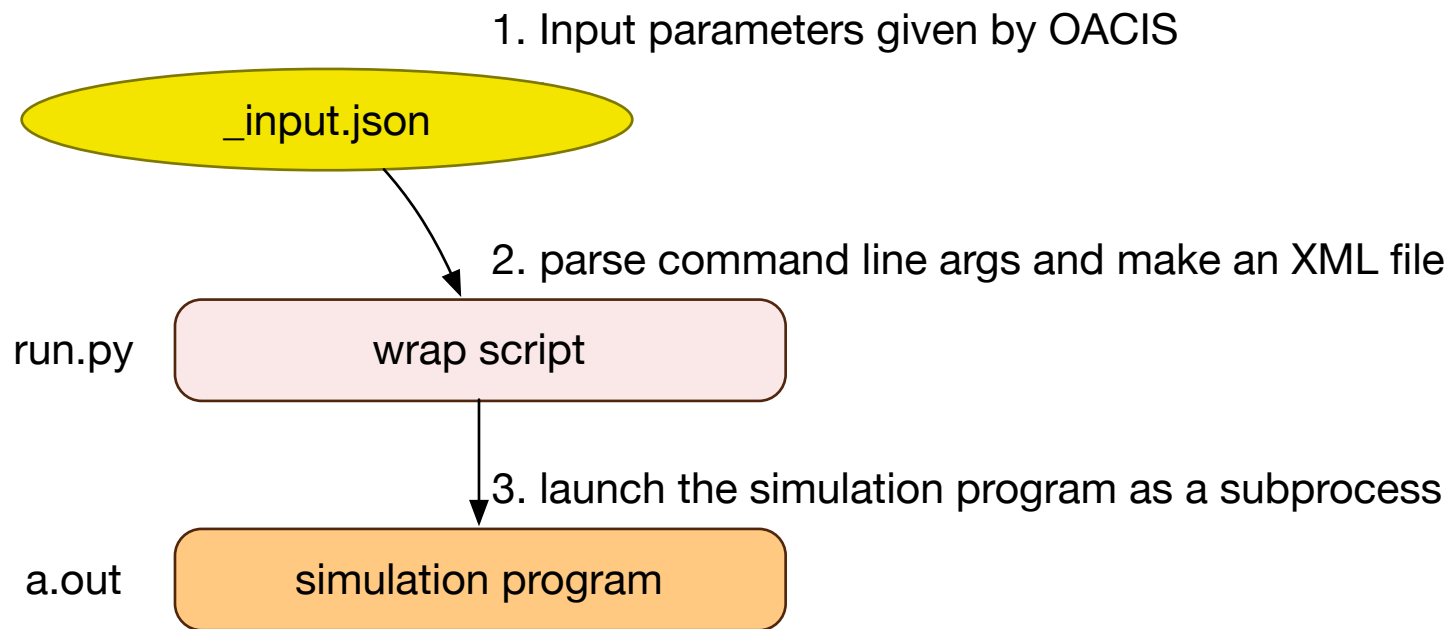
- Input parameters must be given as command line arguments or by a JSON file (Details are shown in the next slide.)
- Output files must be created under the current directory.
- These files are created by OACIS. Simulator must not conflict with these files.
  - `_output.json`, `_status.json`, `_time.txt`, `_version.txt`, `_log.txt`,  
`_stdout.txt`, `_stderr.txt`
- Returns 0 when it finished successfully. Return a non zero code when it has an error.

# how to set input parameters

- By command-line arguments
  - Input parameters and the random number seed are appended to the command.
    - `~/simulator.out <p1> <p2> <p3> ... <seed>`
- By a JSON file
  - When running the command, a file “\_input.json” is created, in which input parameters are written.
    - `{"p1": 30, "p2": 10, "_seed": 12345}`

# Preparation of Simulators

- In order to register your simulator to OACIS, create a script which makes your simulator conform to the input format of OACIS. Register the script as a Simulator of OACIS.



```
import os, sys, json, subprocess
```

```
# Load JSON file
```

```
fp = open( '_input.json' )
```

```
params = json.load( fp )
```

Load the input JSON file

```
# Prepare input file
```

```
f = open('configuration.xml', 'w')
```

```
param_txt = """<configuration>
```

```
    <input>
```

```
        <length value="%d" />
```

```
        <velocity value="%f"/>
```

```
        <time value="%d"/>
```

```
        <seed value="%d"/>
```

```
    </input>
```

```
</configuration>
```

```
""" % (params['length'], params['velocity'], params['time'], params['_seed'])
```

```
f.write(param_txt)
```

```
f.flush()
```

Make an XML file

```
# Execution of the simulator
```

```
simulator = os.path.abspath(os.path.dirname(__file__)) + "/my_simulator.out"
```

```
cmd = [simulator, '-c', 'configuration.xml']
```

```
sys.stderr.write("Running: %s\n" % cmd)
```

```
subprocess.check_call(cmd)
```

```
sys.stderr.write("Successfully finished\n")
```

Spawn a subprocess

Other samples for these scripts are found in our doc page.

[http://crest-cassia.github.io/oacis/en/configuring\\_simulator.html](http://crest-cassia.github.io/oacis/en/configuring_simulator.html)



# Output file format

- As long as output files are created in the current directory, you have nothing to do.
- If you would like to plot scalar values in OACIS, the values should be written in “\_output.json” file.

```
{"flow": 0.235, "velocity": 1.245 }
```

- If you would like to use “figure viewer”, write files in bmp, png, jpg and other figure format. (eps files are not supported.)

# to register a Simulator

OACIS   Simulators   Runs   Analyses   Hosts

## New Simulator

**Name**

**Definition of Parameters**

Parameter Name	Type	Default Value
_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 name**

**Definition of Parameters name, type, default value**

**preprocess command (optional)**

**command of Simulator**

**JSON or Argument input**

**A note of the Simulator**

**List of executable hosts**

# Debugging Tips

- When the set up is inappropriate, runs will fail. In such cases, see “\_stderr.txt” for debugging.

## Parameter Set

/home/oasis/oasis/public/Result\_development/589e8ff1ad229f00ffa9712b/589e8ffcad22

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

### Runs on (p1=0)

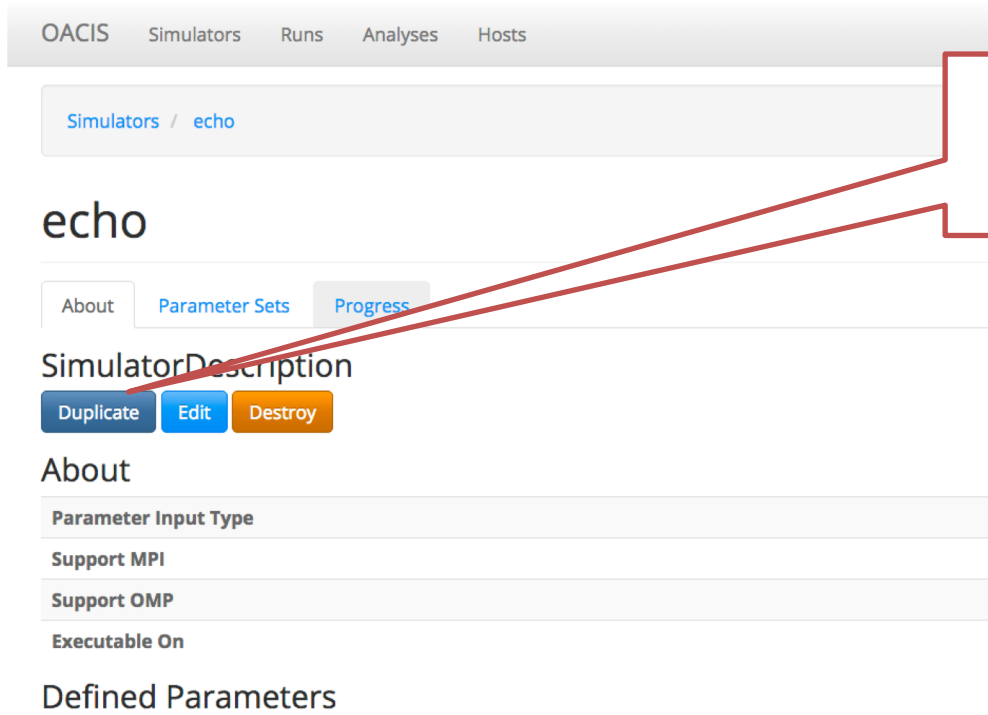
Show  entries 

RunID	status	priority	elapsed	MPI	OMP	version	created_at
<a href="#">ffc132</a>	failed	normal	0.00	1	1		1 min. ago

Showing 1 to 1 of 1 entries

# Duplicating a Simulator

- You may want to try a simulation model which is slightly different from the current one. In such case, you can create a new simulator based on the existing one.



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

Duplicating Simulator

# Hands-on 1

- Let us register “echo” command as a simulator.
- Run a job, and see “\_stdout.txt”

## New Simulator

**Name**  **Simulator name**

**Definition of Parameters**

Parameter Name	Type	Default Value	Action
p1	Integer	100	remove
Description			
p2	Float	0.1	remove
Description			
_seed	seed	-	
Random number seed			

[Add Parameter](#)

**Pre process script**

**Local pre process script**

**Command**  **"echo"**

**Print version command**

**Input type**  **Select "Argument"**

**Support mpi** ☐

**Support omp** ☐

**Sequential seed** ☐

**Description**

**Executable\_on** ☒ localhost **Select "localhost"**

# Hands-on 2

- Let us register the NS\_model simulator by ourselves.

## – Login to the container

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

## – Execute the simulator by yourself.

```
mkdir temp && cd temp  
~/sim_ns_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	initialization steps
t_measure	measurement steps

- From the top page of OACIS, click “New Simulator” button and fill in the form.

Item	Contents
Name	MySimulator
Definition of Parameters	L, v, rho, p, t_init, t_measure
Command	~/sim_ns_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

Click  
“Add Parameter”  
to increase the  
number of  
parameters.



- Execute the newly created Simulator
  - Verify that you get the same result as the previous session.

# registering a Host and HostGroup

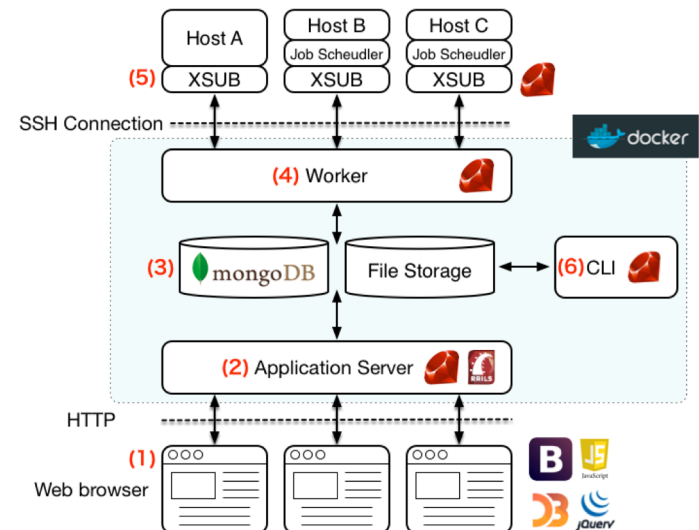
- Reference
  - [http://crest-cassia.github.io/oacis/en/configuring\\_host.html](http://crest-cassia.github.io/oacis/en/configuring_host.html)
  - [http://crest-cassia.github.io/oacis/ja/configuring\\_host.html](http://crest-cassia.github.io/oacis/ja/configuring_host.html)

# Setting up Hosts

A typical sequence for setting up computational host is as follows.

- Setting up SSH authorization key. SSH connections must be established without entering password.
- Install XSUB
- register host information

See  
<http://crest-cassia.github.io/oacis/en/install.html>



# Using SSH-agent

- Setting up SSH authorization key so that SSH connection is available without entering a password or passphrase.

(At OACIS host)

```
ssh-keygen -t rsa          # ~/.ssh/id_rsa ~/.ssh/id_rsa.pub are created
scp ~/.ssh/id_rsa.pub [USER]@[HOST_NAME]:~
                           # copy your public key to the remote host
```

(At Computational host)

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys && chmod 600 ~/.ssh/authorized_keys
                           # add your public key to the list of authorized keys
```

(At OACIS host)

```
eval `ssh-agent`          # launch SSH agent
ssh-add ~/.ssh/id_rsa     # add key to agent. Enter your passphrase.
ssh [USER]@[HOST_NAME]   # verify that you don't need to enter the passphrase
```

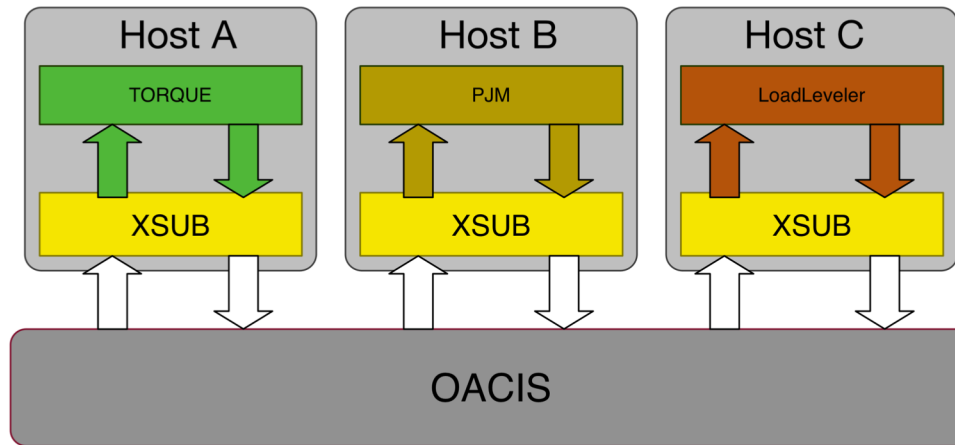
```
bundle exec rake daemon:restart  # launch OACIS
```

# Setting up SSH-config

- OACIS finds an information of remote host from “~/.ssh/config” file.

```
Host my_server
  HostName my_server.example.com
  Port 22
  User murase
  IdentityFile ~/.ssh/id_rsa
Host my_server2
  HostName 127.0.0.1
  Port 22
  User murase
  IdentityFile ~/.ssh/id_rsa
```

# What is XSUB?



- Specification of job schedulers are different for each system.
- XSUB is a small script which absorbs the difference of the specification of job schedulers. OACIS uses XSUB to submit a job. It must be installed in advance.
- Ruby 1.8 or later is required.

Refer to <https://github.com/crest-cassia/xsub>

# Installing XSUB

- On computational host
  - git clone <https://github.com/crest-cassia/xsub.git>

- Edit "~/.bash\_profile"

```
export PATH="$HOME/xsub/bin:$PATH"  
export XSUB_TYPE="none"
```

- Currently, "none", "torque", "FX10", "K", "SR16000", "slurm\_focus", "nqsii", "abci" are supported.

# setting up Host

<http://crest-cassia.github.io/oacis/en/tutorial.html#registering-a-host>

OACIS Simulators Runs Analyses Hosts

## New host

Name	<input type="text"/>	} host information for SSH connection
Hostname	<input type="text"/>	
Polling Status	enabled ▾	
User	<input type="text"/>	
Port	22	
Ssh key	~/ssh/id_rsa	} Directory used for running jobs. OACIS will create sub directories under this.
Work base dir	~	
Mounted work base dir	<input type="text"/>	} Maximum number of concurrent jobs.
Max num jobs	1	
Polling interval	60	} Worker checks the status of the remote host with this interval.
MPI processes	1 ~ 1	
OMP threads	1 ~ 1	} The available range of the number of MPI processes and threads.
Executable Simulators	<input type="checkbox"/> NagelSchreckenberg <input type="checkbox"/> echo	
Executable Analyzers		} List of executable simulators on that host.

Save Cancel



Host was successfully created. ×

## Host: desert1

About [Job Information](#)

Name										
Hostname										
Status	enabled									
User										
Port	22									
SSH key	~/.ssh/id_rsa									
Work base directory	~/oasis_work									
Mounted work base directory										
Maximum number of jobs	64									
Polling interval	5									
MPI procs	1 ~ 32									
OMP threads	1 ~ 32									
Host Parameters	<table><tr><th>Name</th><th>Default</th><th>Format</th></tr><tr><td>ppn</td><td>1</td><td>^[1-9]\d*\$</td></tr><tr><td>walltime</td><td>1:00:00</td><td>^\d+:\d{2}:\d{2}\$</td></tr></table>	Name	Default	Format	ppn	1	^[1-9]\d*\$	walltime	1:00:00	^\d+:\d{2}:\d{2}\$
Name	Default	Format								
ppn	1	^[1-9]\d*\$								
walltime	1:00:00	^\d+:\d{2}:\d{2}\$								
Executable simulators	echo									
Executable analyzers										

Parameters required l  
(Host Paramete  
are automatically

Parameters required by XSUB  
(Host Parameters)  
are automatically set.

[Edit](#) [Back to Index](#) [Destroy](#)

OASIS: Version v2.13.1-13-a7b05b5d

## Create a new parameter set on: echo

p1 (Integer)	<input type="text" value="0"/>	
Target # of Runs	<input type="text" value="1"/>	
Submitted to	<input type="text" value="desrt1"/>	
Priorities of Runs	<input type="text" value="normal"/>	
ppn	<input type="text" value="1"/>	Format: /^[1-9]\d*\$/
walltime	<input type="text" value="1:00:00"/>	Format: /^[1-9]\d+:\d{2}:\d{2}\$/
<input type="button" value="Create"/> <input type="button" value="Cancel"/>		

Host Parameters are required when creating a PS.

# demonstration

- From the container where OACIS is working, execute simulation on the host machine.

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

# testing connection

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

# edit (~/.ssh/config file)

```
ssh my_host
```

exit # logout from computational host

exit # logout from container

# installing xsub on computational host

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

# edit ~/.bash\_profile. Add the following two lines.

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

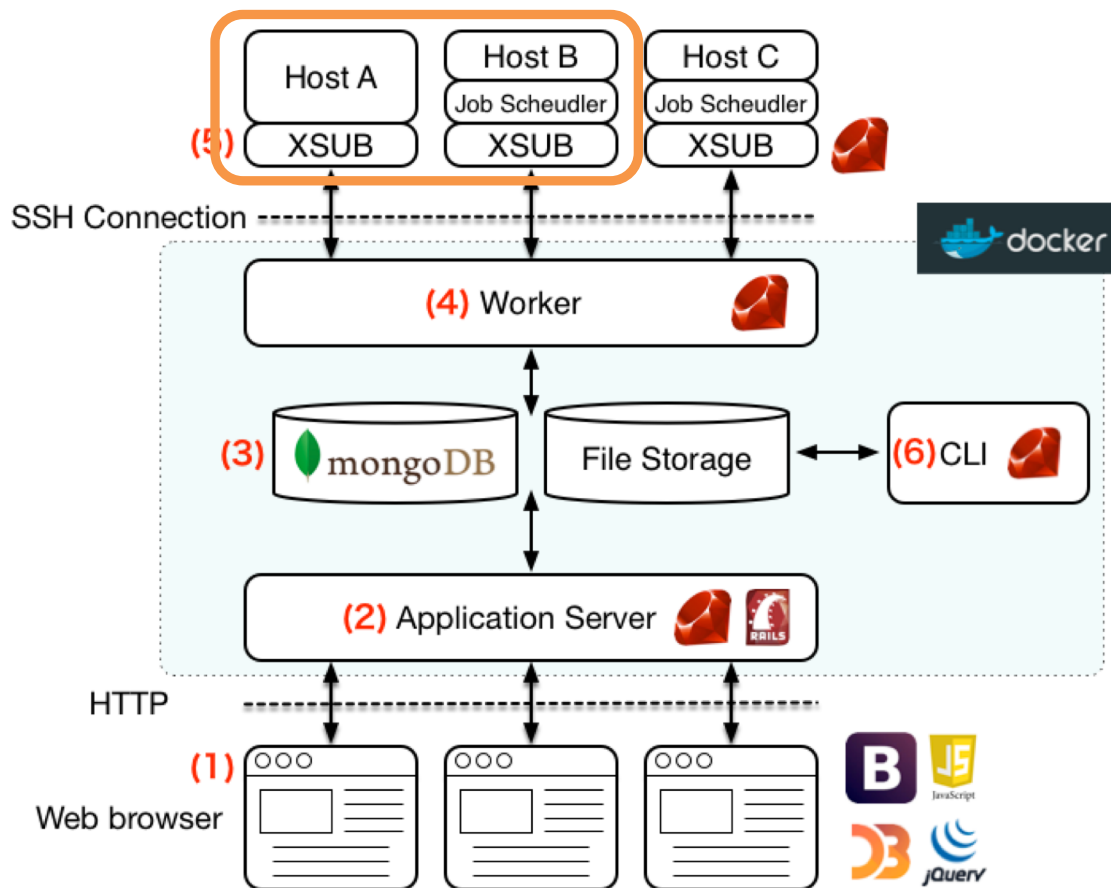
```
export XSUB_TYPE="none"
```

# Defining a HostGroup

We can define a HostGroup, a group of Hosts.

When creating a job, HostGroup can be specified as a destination of the submission.

The jobs are submitted to one of the hosts in the HostGroup.



OACIS

Simulators

Runs

Analyses

Hosts

Logs

click Hosts

click "New Host Group"

# Hosts

Name	Hostname	Status	User	Port	Ssh key	Workdir
localhost	localhost	enabled	oacis	22	~/ssh/id_rsa	pt

New Host

# Host Groups

my\_host\_group

New Host Group

## Create a new parameter set on: Na

l (Integer)

200

v (Integer)

5

rho (Float)

0.3

p (Float)

0.1

t\_init (Integer)

100

t\_measure (Integer)

300

Target # of Runs

1

Submitted to

✓ localhost

HostGroup:my\_host\_group (manual submission)

Priorities of Runs

We can select a HostGroup as the place to which jobs are submitted.

# Conclusion

- We have shown how to register Simulators, Hosts and HostGroups.
    - This might be a bit complicated, but once you have done these registration, the remaining workflow becomes much more productive.
  - If you have any question and feedbacks
    - [oacis-dev@googlegroups.com](mailto:oacis-dev@googlegroups.com)
- ⇒ In the next session, we will demonstrate how to use APIs to automate parameter-space exploration.