Running FakeQuakes over OSG remotely with Python
Python Scripts, their descriptions, and how to run them:

Submit a job (launch a Fakequakes run on OSG):

Scripts: **dag_v2_4_prepare_from_vdc.py, send_vars_launch_from_vdc.py**

How to run:

So dag_v2_4_prepare_from_vdc.py is the main Python script which SSH's into OSG, sends in the files, and launches the FakeQuakes DAGMan workflow over OSG. And send_vars_launch_from_vdc.py is a wrapper file where you change the variables (strings) into what the user enters in to customize their run (number of rupture-jobs, input files, and 5 important parameters).

To run things, you first need to be in an activated environment with 'Paramiko' - the library that I use to SSH into OSG and transfer files. Then you will edit send_vars_launch_from_vdc.py so that its variables match what you want to send in (make sure the file names match the files uploaded for FakeQuakes and the other parameters are in the correct format). Here are all the variables/parameters and their descriptions. They should all be set as strings in the wrapper script):

numruptures: The number of 'rupture jobs' to launch. Each rupture jobs generates 16 ruptures, and then each rupture will have a set of waveforms generated for it. So, if you wanted to make about 1,000 waveforms, you would set numruptures="63" because 63*16=1,008. So, this is what determines how many jobs occur.

These next 5 files are necessary for running FakeQuakes.

modfile: The name of the velocity structure model. Must end in .mod and match the file being used. (e.g. "vel1d_chile.mod")

faultfile: The name of the fault geometry. Must end in .fault and match the file being used. (e.g. "chile.fault")

xyzfile: The name of the Slab 1.0 Ascii file (used for 3D faults). Must end in .xyz and match the file being used. (e.g "chile.xyz")

mshoutfile: The name of the GMSH output file (used for 3D faults). Must end in .mshout and match the file being matched. ("chile.mshout")

gflistfile: The name of the file with Station information; lists all the stations and used for generating waveforms. Must end in .gflist and match the file being used. (e.g. "chile_gnss_small.gflist")

utmzone: UTM zone that defines (roughly) the center of the fault model to help project geographic coordinates to cartesian coordinates. (e.g. “19J”)

timeepi: The UTC date and time at which the synthetic events will originate

targetmw: Input parameters of numpy.arrange() which returns an array of values within a given interval. Decides of what approximate magnitudes FakeQuakes are generated for. Enter in 3 numbers (start, stop, step) that are separated with command and have no space in between. For Fakequakes we are trying to get data for high magnitude earthquakes. Because they don’t happen often, we need to simulate them. (For example: “8.5,9.2,0.2”)

maxslip: The maximum slip (meters) allowed in the model. A reasonable integer. (e.g “100”)

hypocenter: Defines the specific hypocenter location (if force_hypocenter=True which the fault is not). The hypocenter is the starting location of a FakeQuake simulation. (e.g “0.8301,0.01,27.67”)

After editing the wrapper file with the proper variables and being in an environment where Paramiko is active, you want to upload the files that you are using for FakeQuakes into the current directory that the scripts are ran from. You are required to upload and have the 5 geometry files (.mod, .fault, .xyz, .mshout, .mgflist) in directory of the scripts. Optionally, you can also upload distance matrices, Green’s function matrices, and rupture files to be used/recycled. The distance matrices (2 files that end in .npy) are made at the same time ruptures are being generated. They take a long time, so if a user has pre-made them before-hand, they can be uploaded and recycled automatically (or else they’ll be made over OSG). (e.g. planar_subduction.dip.npy)

The Green’s function matrices (6 files ending in .mseed) are made during the same time as the waveforms. They also take a long time to generate, so if the user pre-makes them they can be uploaded and recycled. Both the distance and Green’s function matrices need to be generated with the same geometry files being used. To recycle them, you simply upload the matrix files themselves (.npy and .mseed files) into the current directory, and the scripts will compress them and send them to OSG to be recycled.

If a user has premade ruptures, they can also be sent it, handled by OSG, and will have waveforms generated for them. For using rupture files there is two files per rupture (.rupt and .log) and there needs to be a file called ruptures.list which is a list of all the .rupt files. So, to recycle these, upload all the. rupt and .log files and their corresponding rupture.list to the current directory, and the python script will compress them all and send them to the OSG.

So, after uploading all the files being used to the current directory, editing the wrapper file so the variables are proper, and having Paramiko active in the environment, you simply need to run the send_vars_launch_from_vdc.py Python script. This can be done with the bash command:

“python send_vars_launch_from_vdc.py”

or by importing the script into another python script and running it:

"python send_vars_launch_from_vdc"

The wrapper will call the actual script which launches FakeQuakes on OSG, and then all of the files uploaded to the current directory (to be sent) will be cleaned up, deleted, and left the way that the directory was found.

Cancel a job (cancel a FakeQuakes run on OSG):

Scripts: **cancel_fakequakes_from_vdc_v2_4.py**, **cancel_fakequakes_wrap.py** (use optional)

How to run:

To cancel a FakeQuakes run, you need the 'Cluster ID' for the specific DAGMan run. The Cluster ID is given back in the output after launching a FakeQuakes run. The cluster ID encompasses all jobs in a run, so you can use it to cancel the entire thing.

The script cancel_fakequakes_from_vdc_v2_4.py takes in a single argument which is that Cluster ID. You could also optionally use the wrapper file (cancel_fakequakes_wrap.py) by editing the variable inside of it, and then running the wrapper which calls the actual script and sends in the Cluster ID as an argument. The cancel script SSHs into OSG so it has the Paramiko dependency.

To run this file, you first need to know the Cluster ID of the run you are trying to cancel. With this, you can call the python script and send in the Cluster ID as its only argument to stop the run. You can either import the cancel_fakequakes_from_vdc_v2_4.py script into a python file, and use the line:

'cancel_fakequakes_from_vdc_v2_4.cancel(<Cluster-ID>)'

to cancel it, or you can run the command:

'python cancel_fakequakes_from_vdc_v2_4.py <Cluster-ID>'

in a bash shell with the cluster-ID as an argument. If you want to use the wrapper, edit the wrapper file with the proper cluster ID and either import the script into a python script and call it, or call the script in a shell with no argument.

Get the output of a job (get a tarball containing output from OSG):

Script: **ssh_getFakeQuakeOutput.py**

How to run:

To get the output of a finished FakeQuake job, you will need the name of the directory that the FakeQuake simulations were ran in. This name is given to you when you first submit that job; it will look something like "chile_2022-9-28_1.45.50". It is derived from the fault file along with the date and time that the simulations were launched. So once a job is done you can call the ssh_getFakeQuakeOutput.py script with the directory-name as an argument.

The script takes in just a single argument which is that directory name. The python script SSHs into OSG (so it depends on Paramiko), compresses a folder containing all of the FakeQuake output into a tarball (.tar.gz), and retrieves it bringing it back to the current directory that the script was ran it. When you decompress the tarball you need to use the -m flag to set the timestamps correctly (because the tarball's timestamps will be different across machines.) To decompress the tarball, use the command in the format:

'tar -xzmf fakequakes_output<dir-name>.tar.gz'

This will decompress the output into a folder containing the output. It will be called "fakequakes_output<dir-name>".

To run the script, you can either call it from the shell by being in an environment with Paramiko and using the line:

'python ssh_getFakeQuakeOutput.py <dir-name>'

Or you can import the file in a python script and use the .main() method with the call:

'python ssh_getFakeQuakeOutput.main(<dirname>)'

Get the status of a job (get the time statistics for running FakeQuakes on OSG):

Scripts: **ssh_getStatsDuring.py**

How to run:

To get the status of a FakeQuake run, you will need the name of the directory that the FakeQuake simulations are running in. This name is given to you when you first submit that job; it will look something like "chile_2022-9-28_1.45.50". It is derived from the fault file along with the date and time that the simulations were launched. So, while it is running you will call this python script with the directory-name as its one and only argument.

This script SSHs into the OSG (so it depends on Paramiko), runs a script to get time statistics (consisting of average job time, average submission times, longest/shortest times, total run time, and more), and retrieves a text file from the OSG containing the output with the time statistics. The output text file will be in format:

TimeStatsDuringOutput<dir-name><date-time>.txt

It labeled uniquely with the date and time of when the status was retrieved, so you can run it multiple times on the same FakeQuakes run and things will work.

To run the script, you can either run in a bash shell by calling the script with the directory-name as an argument (in an environment with Paramiko):

'python ssh_getStatsDuring.py <directory-name>'

Or you can import the script into another python script and use the .main() method with the directory name as an argument:

'ssh_getStatsDuring.main(<directoy-name>)'

Get the throughput of a job (get the time statistics for running FakeQuakes on OSG):

Scripts: **ssh_getThroughputDuring.py**

How to run:

To get the throughput of a FakeQuake run, you will need the name of the directory that the FakeQuake simulations is running in. This name is given to you when you first submit that job; it will look something like “chile_2022-9-28_1.45.50”. It is derived from the fault file along with the date and time that the simulations were launched. So, while it is running you can call this python script with the directory-name as its one and only argument.

This script SSHs into the OSG (so it depends on Paramiko), runs a script to get the current throughput (jobs/minute) and some other info (consisting of the number of jobs and runtime in detail), and retrieves a text file from the OSG containing the throughput and job info. The output text file will be in format:

ThroughputDuringOutput<dir-name><date-time>.txt

It labeled uniquely with the date and time of when the throughput was retrieved, so you can run it multiple times on the same FakeQuakes run and things will work.

To run the script, you can either run in a bash shell by calling the script with the directory-name as an argument (in an environment with Paramiko):

‘python ssh_getThroughputDuring.py <directory-name>’

Or you can import the script into another python script and use the .main() method with the directory name as an argument:

‘ssh_getThroughputDuring.main(<directoy-name>)’

Note: In dag_v2_4_prepare_from_vdc you will have to enter in your own OSG host, username, and password, along with having the FDW source code in your OSG home directory. You also need to create a SSH key for your remote machine to add into your OSGconnect profile.