Getting started

February 5th, 2019 Satoshi ITO

1. Install

If you can use git commands:

```
$ git clone https://github.com/SatoshiITO/VGE.git
$ cd VGE
$ python setup.py --user
```

If you download VGE as a zip archive:

```
$ unzip VGE.zip
$ cd VGE
$ python setup.py --user
```

2. How to use?

2.1 Sample codes

In order to understand how use VGE, we prepared a very simple test codes and small data sets. Sample codes and data are stored in `example` directory. The directory structure is below:

```
example/
  ├──bwa_mapping.sh
  ├──make_commands.py
  ├──mysplit.sh
  ├──prep_ref.sh
  ├──run_shell.sh
  ├──run_vge.sh
  ├──samplecode.py
  ├──small1.fastq
  └──small2.fastq
```

In this example, we prepared two job-submit scripts. One is an ordinary shell scripts for grid engines (`run_shell.sh`) and the other is the scripts that describes the same processing details as the former one using VGE (`run_vge.sh`).

2.2 Run with a Grid Engine

In this section, we explain the first sample script `run_shell.sh`. This script contains two tasks, one is `mysplit.sh` that divides input files (`small1.fastq, small2.fastq`) and the other is `bwa_mapping.sh` that aligns split fastq files onto a reference genomes by Burrows-Wheeler Aligner (`bwa`). Figure 1 shows the workflow of `run_shell.sh`.

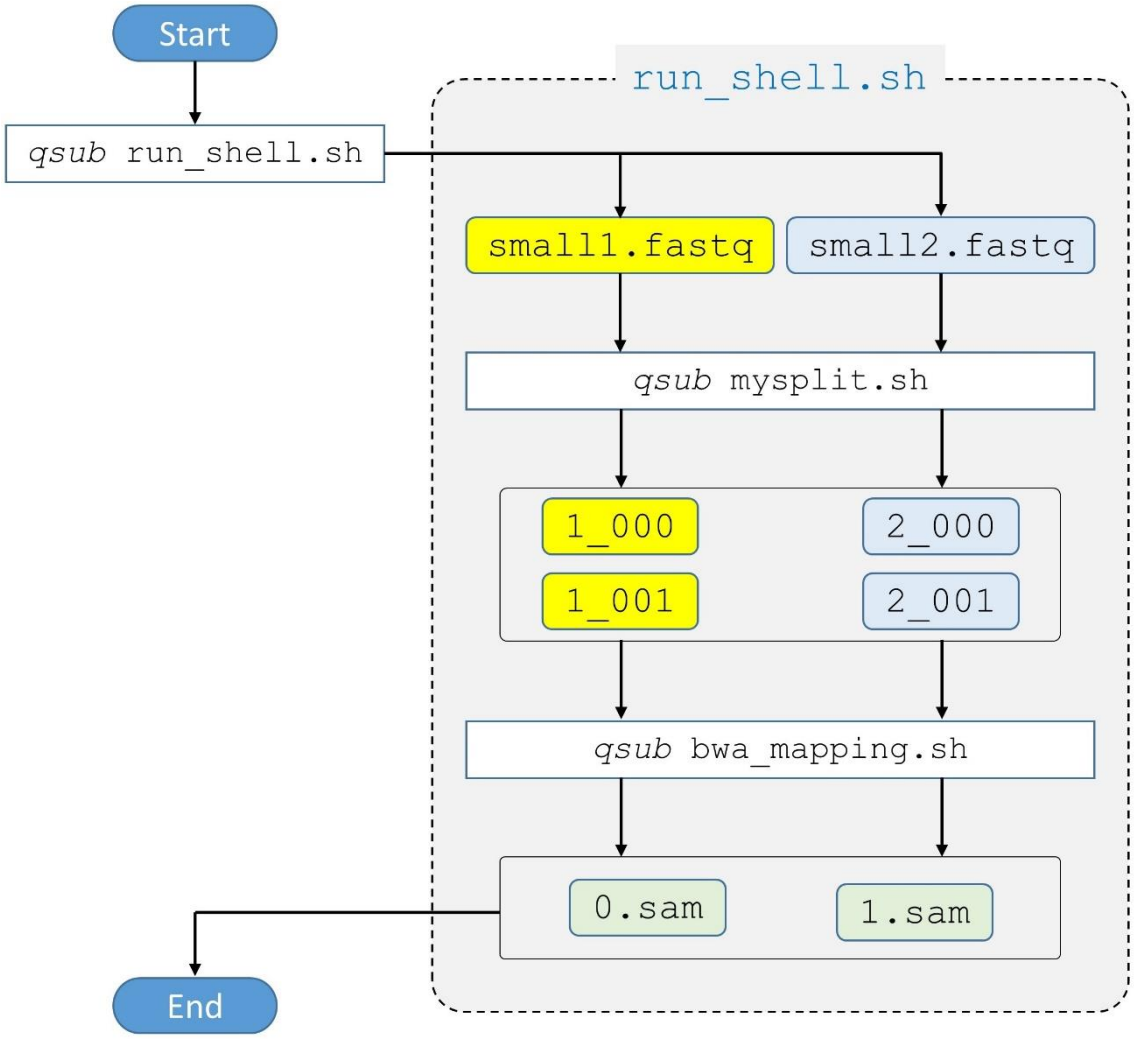

Fig. 1 The workflow of `run_shell.sh`.

To run this example, `bwa` and a reference human genome file (i.e. GRCh37-lite.fa) must be installed on your system. If you don't have a reference genome file, you can use `prep_ref.sh` to download GRCh37-lite.fa file from NIH website.

```
$ bash prep_ref.sh
```

Then, you can run the sample script as follows:

```
$ qsub run_shell.sh
```

You will see two result SAM files (`0.sam` and `1.sam`) if the job successfully finished.

2.3 Run with VGE

Here, we use `run_vge.sh` instead of `run_shell.sh`. This script contains four commands (①-④).

```
(snip)
#
# start VGE
#
mpiexec -n 3 vge &          ―――――――――①
vge_connect -start          ―――――――――②

#
# execute a pipeline
#
time python ./samplecode.py ―――――――――③

#
# stop VGE
#
vge_connect -stop           ―――――――――④
```

Fig. 2 `samplecode.py` (short extraction).

① `mpiexec -n 3 vge &`

This line launches VGE main program by MPI execution command (`mpiexec`). This example runs with two workers, so that the number of MPI processes is set to three (1 master process and 2 worker processes). The last ampersand (`&`) means that VGE main program runs as a background process. It is necessary to execute the latter commands.

② `vge_connect -start` (optional)

Here, VGE has already launched. But, it usually takes a while to be completed (from several seconds to several minutes). `vge_connect` can sends some commands to VGE. This option waits for the completion of VGE boot-up. Note that user pipelines don't need to wait for the boot-up process. This command is useful when you measure the time for computation of your pipeline.

③ `time python ./samplecode.py`

In this line, the script executes the main calculation code (`samplecode.py`). In the previous sample, two tasks (`mysplit.sh` and `bwa_mapping.sh`) were described in the job script (`run_shell.sh`) directly. On the other hand, those tasks are written in `samplecode.py` (Fig.3). This point is the biggest difference between VGE and normal grid engine systems.

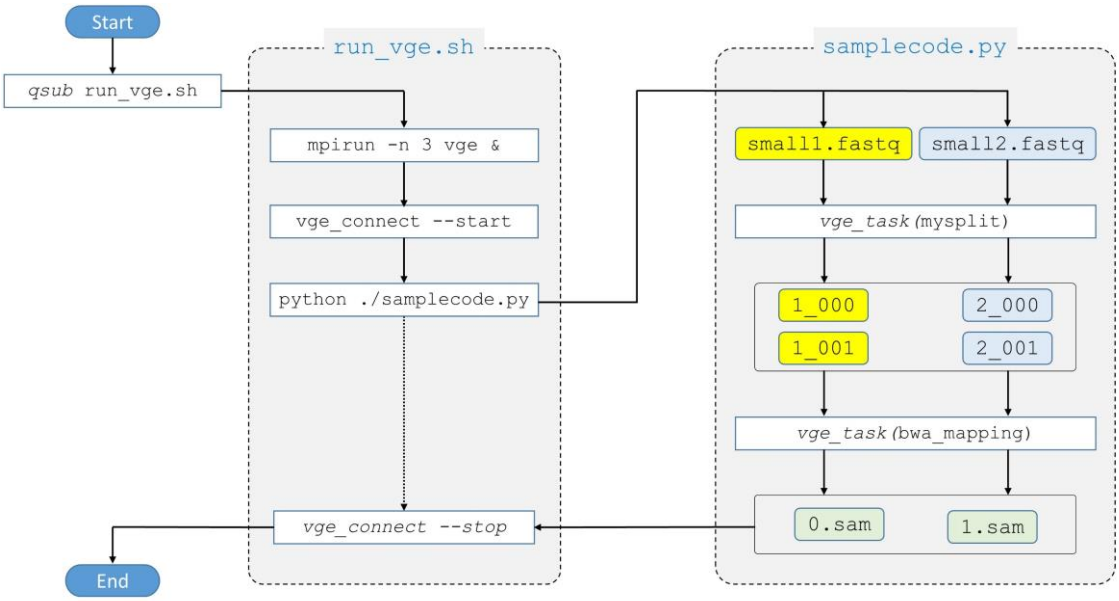

Fig. 3 The workflow of `run_vge.sh`.

④ `vge_connect -stop` (required)

Here, `vge_connect` appears again. First `vge_connect` was optional, but this one must be executed at the end of job scripts. From the specification of VGE, it always runs as a background process. Therefore, you need to terminate it after the calculation of your pipelines. This option (`--stop`) sends a terminate signal to VGE.

To run this example using VGE, type the commands like below:

```
$ qsub run_vge.sh
```

For further information, please see the user manual.