Juicer

Running Juicer on the HPC

Setting up your juiceDir

To get started with Juicer, you will need to have a directory where your input files exist. The juiceDir should contain a folder called `fastq`, a folder called `references`, and a folder called `restriction_sites`. The names should be exactly as stated here as juicer internally looks for folders of these names.

Once you have created these folders and put your data into each one respectively (raw fastq into the `fastq` folder, reference genomes into the `references` folder etc.) you are ready to load the juicer module.

```
$ module load aidenlab/juicer/juicer
```

Loading this module will place a file called `create_juicer.sh` into your home directory. Running that file requires an argument `-j` specify the location of your juicer directory and `-c` specifies the environment, either `cpu` or `slurm`.

The cpu version is much less performance optimized than the slurm version, but is suitable for smaller jobs.

Once `create_juicer.sh` has finished it will create a script file for running hiccups, and for running juicer (the juicer script is either a batch script that you should launch with `sbatch` for the cpu version, or a `.sh` script you can execute directly for the slurm version. The slurm script that you directly execute will spawn multiple slurm jobs on its own).

Running juicer

To run juicer, open the contents of whichever juicer environment version you selected and input the paths to various variables that are set, as well as specificying your account and partition (for the slurm version only).

for example

Slurm version

```
(juicer) [azeezoe@hpc1 juiceDir_tobi]$ ./create_juicer.sh -j test -c slurm
JuiceDir: test
Environment: slurm
Symlink does not exist.

(juicer) [azeezoe@hpc1 juiceDir_tobi]$ cat test/slurm_juicer.sh
```

```
#!/bin/bash

juiceDir=test
module load aidenlab/juicer/juicer
genomeID=
threads=
reference_genome_file=
chrom_sizes_file=
restriction_sites_file=
site=
partition= # Put your partition (eg. compute, debug) here
account= # Put your account (eg. biology) here
compute_tier= # Determines cores in various steps. Select on of either 'low', 'medium', or
'high' (recommended. high)
threads= # Threads for bwa-mem (recommended 24)

$juiceDir/scripts/juicer.sh \
    -D ${juiceDir} \
    -x ${account} \
    -r ${partition} \
    -c ${compute_tier} \
    -z ${juiceDir}/references/${reference_genome_file} \
    -p ${juiceDir}/references/${chrom_sizes_file} \
    -t ${threads} \
    -y ${juiceDir}/restriction_sites/${restriction_sites_file} \
    -g ${genomeID} \
    -s ${site} >> $juiceDir/slurm_juicer.out 2>&1
```

This is the slurm version, and once the necessary values are populated, you kick off the slurm jobs with `./slurm_juicer.sh`

##NOTE FOR SLURM VERSION##

For some reason, the slurm juicer scripts require you to run them twice. If you have large fastq files, the jobs are going to start running, and then the pipeline will fail. This is because a step in the juicer pipeline splits the fastq files so that they will be aligned in parallel more optimally. However, certain variables later on in the script require that these split files already exists. We are looking into how to fix this.

For now, launch `slurm_juicer.sh` and wait for the pipeline to fail. That will usually take around 20-30 mins. You need for it to have created all of the split fastq files in the `splits` folder before you relaunch.

After it fails you can cancel your remaining jobs (which won't launch as they depend on prior jobs to finish. if you run `squeue` and see `dependency never satisfied`, that's what it's talking about) with `scancel -u <your_username>`, and delete the folder that gets created in your juiceDir called `aligned` (check it's empty first, which it should be. If you don't delete it, juicer will ask you to next time you launch). Remember also that `scancel -u <your_username>` will cancel all of your submitted jobs. If you have others running, you can use the `JOBID` instead, to cancel specific jobs only (`scancel <JOBID>`) .

Next go into the splits folder. The splits folder will contain the fastq files that have correctly been split as well as offending files which caused the workflow to break. The *bad* files have a naming scheme of `'**.fastq.sam'` or `'**.fastq_linecount.txt'`. Here is a look at all these files, and you will need to manually remove the bad ones, but leave the correctly split files alone.

```
[azeezoe@hpc1 test]$ ls splits
 DTG_MicroC_755_R1.fastq000.fastq   DTG_MicroC_755_R1.fastq008.fastq
DTG_MicroC_755_R2.fastq007.fastq
 DTG_MicroC_755_R1.fastq001.fastq   DTG_MicroC_755_R2.fastq000.fastq
DTG_MicroC_755_R2.fastq008.fastq
 DTG_MicroC_755_R1.fastq002.fastq   DTG_MicroC_755_R2.fastq001.fastq  '**.fastq_abnorm.sam'
 DTG_MicroC_755_R1.fastq003.fastq   DTG_MicroC_755_R2.fastq002.fastq  '**.fastq_linecount.txt'
 DTG_MicroC_755_R1.fastq004.fastq   DTG_MicroC_755_R2.fastq003.fastq
'**.fastq_norm.txt.res.txt'
 DTG_MicroC_755_R1.fastq005.fastq   DTG_MicroC_755_R2.fastq004.fastq  '**.fastq.sam'
 DTG_MicroC_755_R1.fastq006.fastq   DTG_MicroC_755_R2.fastq005.fastq  '**.fastq.sort.txt'
 DTG_MicroC_755_R1.fastq007.fastq   DTG_MicroC_755_R2.fastq006.fastq  '**.fastq_unmapped.sam'
```

The files we need to remove are the ones with quotations (`'`) around the filenames. We can remove them by running:

```
[azeezoe@hpc1 splits]$ rm '**.fastq_abnorm.sam' '**.fastq_linecount.txt'
'**.fastq_norm.txt.res.txt' '**.fastq.sam' '**.fastq.sort.txt' '**.fastq_unmapped.sam'
[azeezoe@hpc1 splits]$ ls
DTG_MicroC_755_R1.fastq000.fastq  DTG_MicroC_755_R1.fastq006.fastq
DTG_MicroC_755_R2.fastq003.fastq
DTG_MicroC_755_R1.fastq001.fastq  DTG_MicroC_755_R1.fastq007.fastq
DTG_MicroC_755_R2.fastq004.fastq
DTG_MicroC_755_R1.fastq002.fastq  DTG_MicroC_755_R1.fastq008.fastq
DTG_MicroC_755_R2.fastq005.fastq
DTG_MicroC_755_R1.fastq003.fastq  DTG_MicroC_755_R2.fastq000.fastq
DTG_MicroC_755_R2.fastq006.fastq
DTG_MicroC_755_R1.fastq004.fastq  DTG_MicroC_755_R2.fastq001.fastq
DTG_MicroC_755_R2.fastq007.fastq
```

```
DTG_MicroC_755_R1.fastq005.fastq  DTG_MicroC_755_R2.fastq002.fastq
DTG_MicroC_755_R2.fastq008.fastq
```

Then you can relaunch the pipeline again with `./slurm_juicer.sh` and since the new split fastq files have been created (saved to `splits` directory) the variables will be set correctly.

##END NOTE##

for the `slurm_juicer.sh` version, the standard output (`stdout`) will be saved to a file in your juiceDir called `slurm_juicer.out`. There you can look at all of job ids that get created and which steps of the juicer pipeline they refer to. This makes debugging and checking on jobs with `scontrol show job <JOB_ID>` much easier.

CPU version

The cpu version is a lot easier to submit as it doesn't have these issues. Simply modify the `sbatch_juicer_cpu.sh` file and include the variables that you need to launch the pipeline, and launch it using `sbatch sbatch_juicer_cpu.sh`

While juicer is running, modify the `sbatch_hiccups_cpu.sh` script. Only submit that script with `sbatch sbatch_hiccups_cpu.sh` **after** the juicer pipeline has finished. We have to do this manually as the default script looks for GPUs attached to the cluster, which we don't have. We can run the cpu version manually, which does take some time, but works.

Running Juicebox

ssh with X11 enabled

In order to use Juicebox on the HPC, you have to ssh with X11 enabled, which allows for GUI apps to use your local computer graphics even while the software is installed on the hpc. To do this simply execute:

```
ssh -X <your_username>@hpc1.its.appstate.edu
```

Launching Juicebox

To launch juicebox, simply load the juicer module with `module load aidenlab/juicer/juicer` and then type `juicebox` directly in the terminal. The following window should appear

To load your data, click on 'file/open...'

Then click `local` at the bottom.

Browse to the directory where your `.hic` files exist and then click open. Give it some time to load, afterwhich you should see your Hi-C Map appear