A Simple Guide to Using Pylada

Here is a quick guide to using Pylada on Compute Canada machines.

Installation

To install Pylada on Niagara (or any cc computer) do this

```
module load python cmake intel/2020u4
pip install --user --upgrade pip
pip install --user git+https://github.com/pylada/pylada-light@few_fixes
```

I like to have it in a separate virtual environment:

```
module load python cmake intel/2020u4
python -m venv pylada_env
source pylada_env/bin/activate
pip install --upgrade pip
pip install git+https://github.com/pylada/pylada-light@few_fixes
```

This should take a little while, because it needs to compile from source. Make sure that python is 3.9 or less. After that you need to copy the file `.pylada` in your home directory (this file may be hidden, but it is in this folder!). At line 20 of that file change my username for yours

```
jobs   = Popen(['squeue', '-u', 'your_user_name'], stdout=PIPE, encoding='utf-8').communicate()[0].split('\n')
```

Lines 40-61 contain the slurm script that Pylada will use to launch jobs. Make sure to change L55-58 to the correct modules and environment that you are using if you are on a different computer.

```
###################### PBSSCRIPT ######################

pbs_string =  '''#!/bin/bash -x
#SBATCH --account={account}
#SBATCH --nodes={nnodes}
#SBATCH --ntasks-per-node={ppn}
#SBATCH --exclusive
#SBATCH --export=ALL
#SBATCH --time={walltime}
#SBATCH -o {out}
#SBATCH -e {err}
#SBATCH --job-name={name}
#SBATCH -p {queue} #<- You may need to remove this line

# Go to the directoy from which our job was launched
cd {directory}

# source ~/.bashrc #<- You may need to add this line
module use /scinet/niagara/software/commercial/modules    #<-This line
module load intel/2020u4 intelmpi/2020u4 hdf5/1.10.7 vasp #<-This line
module load python cmake                                  #<-This line
source ~/pylada_env/bin/activate                          #<-This line

{header}
```

```
python {scriptcommand}
{footer}
'''
```

> Note
>
> On some Compute Canada computers there are no partitions, you will need to remove line 35. You may also have to manually source ~/.bashrc

Usage

Pylada can be used for a lot of different things such as manipulating crystal structures and finding primitive cells. Here I will show you how to use it to submit VASP jobs.

Submission script

Let's go through `HT_relax_example.ipy`.

Setting VASP files

At lines 26-30 you can set the location of the VASP executable and the pseudo-potentials:

```
# Program and file location
vasp.program = '/scinet/niagara/software/commercial/vasp-6.1.2/bin/vasp_std'
vasp.has_nlep = False

pseudoDir = '/scinet/niagara/software/commercial/vasp-6.1.2/potpaw_PBE/'
```

Setting pseudopotentials

The next few lines (32-45) set the pseudopotentials that you want to use for each element. This is useful if you want to use ultrasoft PPs or if you want to use a version with d or f orbitals.

```
vasp.add_specie = "O", pseudoDir + "/O" # To use the O PP
vasp.add_specie = "O", pseudoDir + "/O_s" # To use the O ultrasoft PP
```

Setting INCAR parameters

Lines 46-77 set the INCAR parameters. A few things to note:

```
vasp.convergence = 1.0e-3
```

If you are relaxing cell parameters (which is not the case in this file, see below), it is good to do multiple relaxations because as you change the cell shape your basis set does not. Pylada does that automatically and this set the convergence between each relaxation step.

```
vasp.relaxation = "ionic"
```

This is where you set the type of relaxation (ISIF) the options are `static`, `volume`, `cellshape` and `ionic`. If you wanted to relax the volume shape and atoms you would do.

```
vasp.relaxation = "volume ionic cellshape"
```

Most INCAR parameters can be set directly e.g. `NSW` is `vasp.nsw`. If you are using a less common parameter sometimes you need to *manually* add it using this command :

```
vasp.add_keyword("NUPDOWN", 2)
```

Setting the input and output directories

Lines 79-98 go through input files and add them to a list of structures. This is where I set the input and output directories and their naming conventions.

```
structures={}

jobs = get_jobs()
for poscar in iglob('POSCAR_*'): # <- This is where you set the input files     location

    s=read.poscar(poscar)
    s.name=''
    name='output/'+poscar[poscar.rindex('_')+1:] # <- This is where you set the     output file location
    if name not in jobs: # Making sure it is not already running
        structures[name]=s
```

The rest of the lines just add the structures to the `JobFolder()` object which is the instance that will launch the jobs. This is where you could set `INCAR` parameters that are structure-dependent like the magnetic moment for example or `NPAR`.

Launching jobs

To launch all the jobs on Niagara you can simply uncomment that last line of `HT_relax_example.ipy` and run:

```
ipython HT_relax_example.ipy
```

But I like to do it manually in Ipython so I can make sure everything is ok

```
ipython
In [1]: %run HT_relax_example.ipy
In [2]: %launch scattered --account=rrg-ovoznyy --walltime=24:00:00 --ppn=40 --queue=compute
```

The launch command controls the account you are using the amount of time you are asking for and your queue. If you jobs have already finished Pylada will detect it and they won't be relaunched.

Extracting Data

Pylada makes it easy to extract a lot of information on the jobs once they are finished. To get information in python or Ipython do:

```
python
>>> from pylada.vasp import Extract, MassExtract
>>> allresults = MassExtract("output_folder")
>>> dir(allresults) # To get a list of available quantities
>>> results = Extract("output_folder/specific_run")
>>> dir(results)
```