Automating Scientific Workflows with AiiDA

Pawel Pomorski
ppomorsk@sharcnet.ca

August 9, 2023

AiiDA - Automated Interactive Infrastructure and DAtabase (https://www.aiida.net/)

AiiDA is an open-source Python infrastructure to help researchers with automating, managing, persisting, sharing and reproducing the complex workflows associated with modern computational science and all associated data.

Initially designed for computational materials science but can be extended to other areas.

Can be installed on Linux, MacOS X (Homebrew) and Windows (WSL).

AiiDA alternatives

- AFLOW - materials science calculations over large database of structures
- Atomic Simulation Environment (ASE) : single calculations only
- FireWorks : workflow software for running high-throughput calculation workflows at supercomputing centers.
- atomate : uses pymatgen, custodian, and FireWorks

Workflows

In research computing one increasing has to automate complex multi-step procedures integrating different simulation software.

In AiiDA, workflows use the full power of Python programming language running on their local computer to launch arbitrary executables on remote compute resources.

Once a workflow is submitted, the AiiDA daemon handles its execution, waiting for pre-requisite jobs to finish before launching subsequent steps.

AiiDA workflows can be chained, enabling users to group their code into functional building blocks and combined them in sophisticated simulations. Workflows can be shared with others by becoming part of a workflow library.

AiiDA plugins

AiiDA plugins will take care of preparing input files for the simulation and parsing the output files into AiiDA data types.

Public AiiDA registry provides a central place to find AiiDA plugins.

Users can develop their own plugins.

Current plugins already include support for popular materials science software, including: Abinit, ASE, CASTEP, CP2K, Gaussian, Gromacs, LAMMPS, NWChem, ORCA, Quantum ESPRESSO, Siesta, VASP, Wien@K

Plugins enabling the use of FireWords and ACE also available.

HPC Interface

AiiDA can interact with multiple HPC clusters.

The system to be used for a calculation can be selected by changing one line of code.

Specifically for Alliance users, this means can easily submit jobs to cedar, graham, niagara, narval and beluga from a single interface, allowing greater job throughput.

Once a calculation is submitted, AiiDA takes care of all the details: uploading the data, submitting the job, downloading the results once the job is done and cleaning up the job directory on the cluster.

Reproducibility

In AiiDA, data provenance is tracked automatically.

Each calculation is represented by a node that is linked to its input and output data nodes.

The resulting provenance graph is stored in a local database and can be queried using a performant high-level Python interface.

Sharing

AiiDA users can export their provenance graph and share it with others.

When uploaded to Materials Cloud, other reserchers can interface with the graph and download it. They can then reproduce the work or continue from where the author left off.

Querying data

As AiiDA is used, its database can grow large. Fortunately, AiiDA provides a powerful querying mechanism to search for data.

For example, for a given node, one can determine which other nodes have used it as input. Thus, if there was a problem with that node and it needs to be recalculated, all the nodes which will also need to be recalculated can be found.

aiida seminar

September 1, 2023

1 AiiDA setup

After installing the required Linux packages and starting the required services, install AiiDA with pip install into a virtual environment.

Run verdi setup utility.

Start notebook in AiiDA kernel previously created.

[1]:
```
! ~/envs/aiida/bin/verdi status
```

```
  version:     AiiDA v2.4.0
  config:      /Users/pawelpomorski/.aiida
  profile:     quicksetup
  storage:     Storage for 'quicksetup' [open] @ postgresql:
//aiida_qs_pawelpomorski_542a35bff468739fefdcff6f3186d68e:***@localhost:5432/qui
cksetup_pawelpomorski_542a35bff468739fefdcff6f3186d68e /
DiskObjectStoreRepository: 2098278fdc7a40eba76163c2dab69685 |
/Users/pawelpomorski/.aiida/repository/quicksetup/container
Warning: RabbitMQ v3.12.2 is not supported and will cause
unexpected problems!
Warning: It can cause long-running workflows to crash and jobs
to be submitted multiple times.
Warning: See https://github.com/aiidateam/aiida-
core/wiki/RabbitMQ-version-to-use for details.
  rabbitmq:    Incompatible RabbitMQ version detected!
Connected to RabbitMQ v3.12.2 as
amqp://guest:guest@127.0.0.1:5672?heartbeat=600
  daemon:      The daemon is not running.
```

2 Nodes and provenance graphs

[76]:
```
from aiida import orm

node1 = orm.Int(2)
node1.store()
node1
```

[76]: <Int: uuid: 4341ecd0-1565-4447-99a6-f558a5d76c7f (pk: 1387) value: 2>

[77]:
```
from aiida import engine

@engine.calcfunction
def multiply(x, y):
    return x * y
```

[78]:
```
x = orm.load_node(pk=1387)
y = orm.Int(3)
multiply(x, y)
```

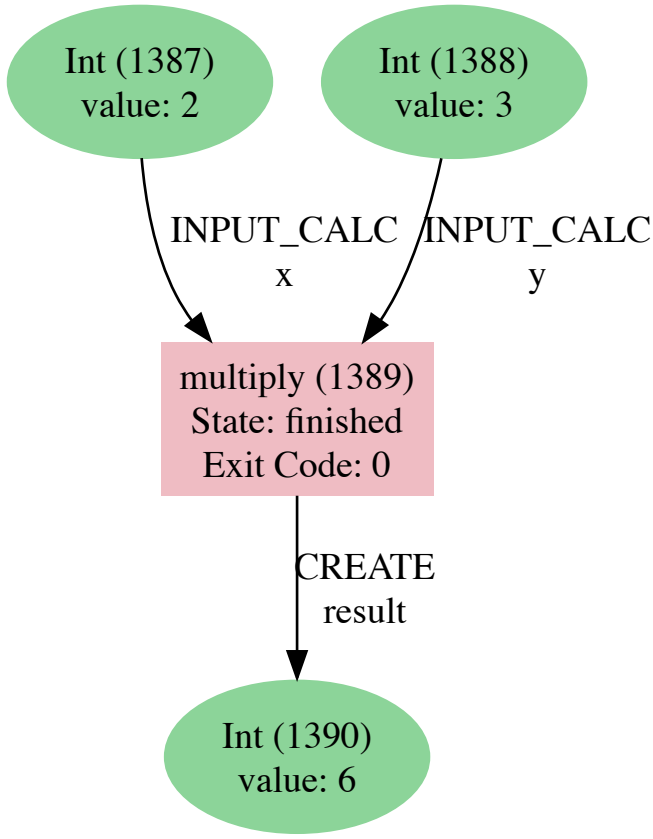
[78]: <Int: uuid: df2d8239-7e50-4787-9bef-1940c7778e7a (pk: 1390) value: 6>

[79]:
```
# all calculation inputs automatically stored in database
y
```

[79]: <Int: uuid: 6f7870ee-fd10-46ba-a2d4-33f198afa96c (pk: 1388) value: 3>

[26]:
```
from aiida.tools.visualization import Graph
graph = Graph()
calc_node = orm.load_node(1389)
graph.add_incoming(calc_node, annotate_links="both")
graph.add_outgoing(calc_node, annotate_links="both")
graph.graphviz
```

[26]:



[86]:
```
! ~/envs/aiida/bin/verdi node show 1389
```

```
Property     Value
-----------  ------------------------------------
type         multiply
state        Finished [0]
pk           1389
uuid         4a2fbbdf-3931-4cc9-9d3c-cbb1125b8fd1
label        multiply
description
ctime        2023-08-08 23:20:44.416909-04:00
mtime        2023-08-08 23:20:44.573526-04:00

Inputs      PK  Type
--------  ----  ------
x         1387  Int
y         1388  Int

Outputs      PK  Type
---------  ----  ------
result     1390  Int
```

3 VASP calculation

Simple VASP calculation of Si bulk structure needs just 4 input files.

[24]:
```
!ssh aiida_ctetsass@graham.computecanada.ca ls /project/def-ctetsass/
↪aiida_ctetsass/job_files_aiida/9943617vasp08-06-23-17-24-52/run_direct

# mpirun -np 2 /project/def-ctetsass/aiida_ctetsass/.local/easybuild/software/
↪2020/avx2/MPI/intel2020/impi2019/vasp/6.3.0/bin/vasp_std
```

```
/usr/bin/id: cannot find name for group ID 13200135
INCAR
KPOINTS
POSCAR
POTCAR
```

[49]:
```
!ssh aiida_ctetsass@graham.computecanada.ca cat /project/def-ctetsass/
↪aiida_ctetsass/job_files_aiida/9943617vasp08-06-23-17-24-52/run_direct/INCAR
```

```
/usr/bin/id: cannot find name for group ID 13200135
EDIFF = 0.0001
ENCUT = 200
IALGO = 38
ISMEAR = -5
PREC = NORMAL
SIGMA = 0.1
```

[27]:
```
# output files after run completes
!ssh aiida_ctetsass@graham.computecanada.ca ls /project/def-ctetsass/
↪aiida_ctetsass/job_files_aiida/9943617vasp08-06-23-17-24-52/run_direct
```

```
/usr/bin/id: cannot find name for group ID 13200135
CHG
CHGCAR
CONTCAR
DOSCAR
EIGENVAL
IBZKPT
INCAR
KPOINTS
OSZICAR
OUTCAR
PCDAT
POSCAR
POTCAR
REPORT
vaspout.h5
vasprun.xml
WAVECAR
```

XDATCAR

[28]:
```
!ssh aiida_ctetsass@graham.computecanada.ca  cat  /project/def-ctetsass/
↪aiida_ctetsass/job_files_aiida/9943617vasp08-06-23-17-24-52/run_direct/
↪OUTCAR | grep "free energy"
```

```
/usr/bin/id: cannot find name for group ID 13200135
  free energy    TOTEN  =        -3.18722214 eV
  free energy    TOTEN  =       -10.91141958 eV
  free energy    TOTEN  =       -10.97522910 eV
  free energy    TOTEN  =       -10.97541058 eV
  free energy    TOTEN  =       -10.97541071 eV
  free energy    TOTEN  =       -10.85138207 eV
  free energy    TOTEN  =       -10.79499734 eV
  free energy    TOTEN  =       -10.79589378 eV
  free energy    TOTEN  =       -10.79611292 eV
  free energy    TOTEN  =       -10.79613809 eV
```

4 AiiDA setup

Local database controlled by user, or remote database accessible by research group.

AiiDA-VASP plugin installed via pip.

[98]:
```
! ~/envs/aiida/bin/verdi computer list
```

```
Report: List of configured computers
Report: Use 'verdi computer show COMPUTERLABEL' to display
more detailed information
* graham1
* graham_service
```

[103]:
```
! ~/envs/aiida/bin/verdi computer test graham_service
```

```
Report: Testing computer<graham_service> for
user<ppomorsk@sharcnet.ca>…
* Opening connection… [OK]
* Checking for spurious output… [Failed]:
We detected some spurious output in the stderr when connecting to the computer,
as shown between the bars
================================================================================
=====================
/usr/bin/id: cannot find name for group ID 13200135

================================================================================
=====================
Please check that you don't have code producing output in your ~/.bash_profile,
~/.bashrc or similar.
If you don't want to remove the code, but just to disable it for non-interactive
```

```
shells, see comments
in this troubleshooting section of the online documentation:
https://bit.ly/2FCRDc5

* Getting number of jobs from scheduler… Warning: squeue
returned exit code 0 (_parse_joblist_output function) but non-empty
stderr='/usr/bin/id: cannot find name for group ID 13200135'
[OK]: 8699 jobs found in the queue
* Determining remote user name… Warning: There was
nonempty stderr in the whoami command: /usr/bin/id: cannot find name for group
ID 13200135

[OK]: aiida_ctetsass
* Creating and deleting temporary file… Warning: There
was nonempty stderr in the whoami command: /usr/bin/id: cannot find name for
group ID 13200135

[OK]
* Checking for possible delay from using login shell…
[OK]
Warning: 1 out of 6 tests failed
```

[23]:
```
! ~/envs/aiida/bin/verdi computer show graham_service
```

```
---------------------------  ------------------------------------------
Label                        graham_service
PK                           6
UUID                         99ae5bb6-ee3a-4eb3-a670-685731c834ec
Description
Hostname                     graham.computecanada.ca
Transport type               core.ssh
Scheduler type               core.slurm
Work directory               /project/def-ctetsass/aiida_ctetsass/aiida
Shebang                      #!/bin/bash
Mpirun command               mpirun -np {tot_num_mpiprocs}
Default #procs/machine       2
Default memory (kB)/machine  8000
Prepend text
Append text
---------------------------  ------------------------------------------
```

[24]:
```
! ~/envs/aiida/bin/verdi code list
```

```
Full label                      Pk  Entry point
------------------------------  ----  -------------------
vaspgraham@graham_service       864  core.code.installed
vaspgrahamcopy@graham_service   973  core.code.installed

Use `verdi code show IDENTIFIER` to see details for a code
```

[25]:
```
! ~/envs/aiida/bin/verdi code show vaspgraham@graham_service
```

```
-----------------------  ----------------------------------------------------
--------------------------------------------------------------------
PK                       864
UUID                     402aba7a-67e2-4c72-be1a-8a7630b9c467
Type                     core.code.installed
Computer                 graham_service (graham.computecanada.ca), pk: 6
Filepath executable      /project/def-ctetsass/aiida_ctetsass/.local/easybuild/s
oftware/2020/avx2/MPI/intel2020/impi2019/vasp/6.3.0/bin/vasp_std
Label                    vaspgraham
Description              VASP 6.3.0 on graham
Default calc job plugin  vasp.vasp
Use double quotes        False
With mpi
Prepend text             module purge
                         module load StdEnv/2020  intel/2020.1.217
intelmpi/2019.7.217 vasp/6.3.0
Append text
-----------------------  ----------------------------------------------------
--------------------------------------------------------------------
```

Pseudopotential uploaded beforehand via AiiDA-VASP plugin interface.

5 AiiDA - first example

[2]:
```
# run vasp lean

"""
An example call script that performs a single static VASP calculation.

Performs a self consistent electron convergence run using the standard silicon␣
 ↪structure.
"""
# pylint: disable=too-many-arguments
import numpy as np

from aiida import load_profile
from aiida.common.extendeddicts import AttributeDict
from aiida.engine import submit
from aiida.orm import Bool, Code, Str
from aiida.plugins import DataFactory, WorkflowFactory

load_profile()


def get_structure():
```

```
    """
    Set up Si primitive cell

    Si
       5.431
         0.0000000000000000    0.5000000000000000    0.5000000000000000
         0.5000000000000000    0.0000000000000000    0.5000000000000000
         0.5000000000000000    0.5000000000000000    0.0000000000000000
    Si
       2
    Direct
      0.8750000000000000  0.8750000000000000  0.8750000000000000
      0.1250000000000000  0.1250000000000000  0.1250000000000000

    """

    structure_data = DataFactory('core.structure')
    alat = 5.431
    lattice = np.array([[.5, 0, .5], [.5, .5, 0], [0, .5, .5]]) * alat
    structure = structure_data(cell=lattice)
    for pos_direct in ([0.875, 0.875, 0.875], [0.125, 0.125, 0.125]):
        pos_cartesian = np.dot(pos_direct, lattice)
        structure.append_atom(position=pos_cartesian, symbols='Si')
    return structure


def main(code_string, incar, kmesh, structure, potential_family,␣
 ↪potential_mapping, options):
    """Main method to setup the calculation."""

    # First, we need to fetch the AiiDA datatypes which will
    # house the inputs to our calculation
    dict_data = DataFactory('core.dict')
    kpoints_data = DataFactory('core.array.kpoints')

    # Then, we set the workchain you would like to call
    workchain = WorkflowFactory('vasp.vasp')

    # And finally, we declare the options, settings and input containers
    settings = AttributeDict()
    inputs = AttributeDict()

    # Organize settings
    settings.parser_settings = {'output_params': ['total_energies',␣
 ↪'maximum_force']}

    # Set inputs for the following WorkChain execution
```

```
    # Set code
    inputs.code = Code.get_from_string(code_string)
    # Set structure
    inputs.structure = structure
    # Set k-points grid density
    kpoints = kpoints_data()
    kpoints.set_kpoints_mesh(kmesh)
    inputs.kpoints = kpoints
    # Set parameters
    inputs.parameters = dict_data(dict=incar)
    # Set potentials and their mapping
    inputs.potential_family = Str(potential_family)
    inputs.potential_mapping = dict_data(dict=potential_mapping)
    # Set options
    inputs.options = dict_data(dict=options)
    # Set settings
    inputs.settings = dict_data(dict=settings)
    # Set workchain related inputs, in this case, give more explicit output to␣
↪report
    inputs.verbose = Bool(True)
    # Submit the requested workchain with the supplied inputs
    output = submit(workchain, **inputs)
    print(output)

if __name__ == '__main__':
    # Code_string is chosen among the list given by 'verdi code list'
    CODE_STRING = 'vaspgrahamcopy@graham_service'

    # POSCAR equivalent
    # Set the silicon structure
    STRUCTURE = get_structure()

    # INCAR equivalent
    # Set input parameters
    INCAR = {'incar': {'prec': 'NORMAL', 'encut': 200, 'ediff': 1E-4, 'ialgo':␣
↪38, 'ismear': -5, 'sigma': 0.1}}

    # KPOINTS equivalent
    # Set kpoint mesh
    KMESH = [9, 9, 9]

    # POTCAR equivalent
    # Potential_family is chosen among the list given by
    # 'verdi data vasp-potcar listfamilies'
    POTENTIAL_FAMILY = 'pbe'
    # The potential mapping selects which potential to use, here we use the␣
↪standard
```

```
    # for silicon, this could for instance be {'Si': 'Si_GW'} to use the GW␣
↪ready
    # potential instead
    POTENTIAL_MAPPING = {'Si': 'Si'}

    # Jobfile equivalent
    # In options, we typically set scheduler options.
    # See https://aiida.readthedocs.io/projects/aiida-core/en/latest/scheduler/
↪index.html
    # AttributeDict is just a special dictionary with the extra benefit that
    # you can set and get the key contents with mydict.mykey, instead of␣
↪mydict['mykey']
    OPTIONS = AttributeDict()
    OPTIONS.account = ''
    OPTIONS.qos = ''
    OPTIONS.resources = {'num_machines': 1, 'num_mpiprocs_per_machine': 1}
    OPTIONS.queue_name = ''
    OPTIONS.max_wallclock_seconds = 3600
    OPTIONS.max_memory_kb = 10240000

    main(CODE_STRING, INCAR, KMESH, STRUCTURE, POTENTIAL_FAMILY,␣
↪POTENTIAL_MAPPING, OPTIONS)
```

```
Warning: RabbitMQ v3.12.2 is not supported and will cause
unexpected problems!
Warning: It can cause long-running workflows to crash and jobs
to be submitted multiple times.
Warning: See https://github.com/aiidateam/aiida-
core/wiki/RabbitMQ-version-to-use for details.
uuid: ba2b7901-69d1-433e-bbb8-dce3a10a74b4 (pk: 1434)
(aiida.workflows:vasp.vasp)
```

Comment: at this point the job is launched but of course it might take a while before it is done. We cannot really work in the notebook right away to analyze the data. We could of course come back to the notebook once the job is done.

orm - object relational mapping

[4]:
```
# show verdi process output
! ~/envs/aiida/bin/verdi process list
```

```
  PK  Created    Process label    Process State    Process status
----  ---------  ---------------  ---------------
-----------------------------------
1401  15m ago    VaspWorkChain      Created
1412  13m ago    VaspWorkChain      Created
1423  9m ago     VaspWorkChain      Created
1434  1m ago     VaspWorkChain      Waiting        Waiting for child processes:
1436
```

```
1436  9s ago     VaspCalculation    Waiting        Waiting for transport task:
upload

Total results: 5

Report: last time an entry changed state: 9s ago (at 10:46:38
on 2023-08-09)
Report: Checking daemon load… OK
Report: Using 1% of the available daemon worker slots.
```

[7]:
```
# show that job was submitted and ran
# sacct command
!ssh aiida_ctetsass@graham.computecanada.ca  sq
```

```
/usr/bin/id: cannot find name for group ID 13200135
             JOBID     USER              ACCOUNT           NAME  ST  TIME_LEFT
NODES CPUS TRES_PER_N MIN_MEM NODELIST (REASON)
```

[ ]:
```
# querying result
```

[ ]:
```

```

[8]:
```
! ~/envs/aiida/bin/verdi process report 1434
```

```
2023-08-09 10:46:38 [193 | REPORT]: [1434|VaspWorkChain|run_process]:
launching VaspCalculation<1436> iteration #1
2023-08-09 10:50:40 [195 | REPORT]: [1434|VaspWorkChain|results]: work chain
completed after 1 iterations
2023-08-09 10:50:44 [196 | REPORT]: [1434|VaspWorkChain|on_terminated]: cleaned
remote folders of calculations: 1436
```

[9]:
```
! ~/envs/aiida/bin/verdi node show 1434
```

```
Property     Value
-----------  ------------------------------------
type         VaspWorkChain
state        Finished [0]
pk           1434
uuid         ba2b7901-69d1-433e-bbb8-dce3a10a74b4
label
description
ctime        2023-08-09 10:45:36.485977-04:00
mtime        2023-08-09 10:50:40.179558-04:00

Inputs               PK  Type
-----------------  ----  -------------
clean_workdir      1433  Bool
code                973  InstalledCode
kpoints            1425  KpointsData
```

```
max_iterations     1432  Int
options            1429  Dict
parameters         1426  Dict
potential_family   1427  Str
potential_mapping  1428  Dict
settings           1430  Dict
structure          1424  StructureData
verbose            1431  Bool

Outputs          PK  Type
-------------  ----  ----------
misc           1439  Dict
remote_folder  1437  RemoteData
retrieved      1438  FolderData

Called          PK  Type
------------  ----  ---------------
iteration_01  1436  VaspCalculation

Log messages
----------------------------------------------
There are 3 log messages for this calculation
Run 'verdi process report 1434' to see them
```

[11]:
```
! ~/envs/aiida/bin/verdi node repo ls 1438
```

```
CONTCAR
DOSCAR
EIGENVAL
OUTCAR
_scheduler-stderr.txt
_scheduler-stdout.txt
vasp_output
vasprun.xml
```

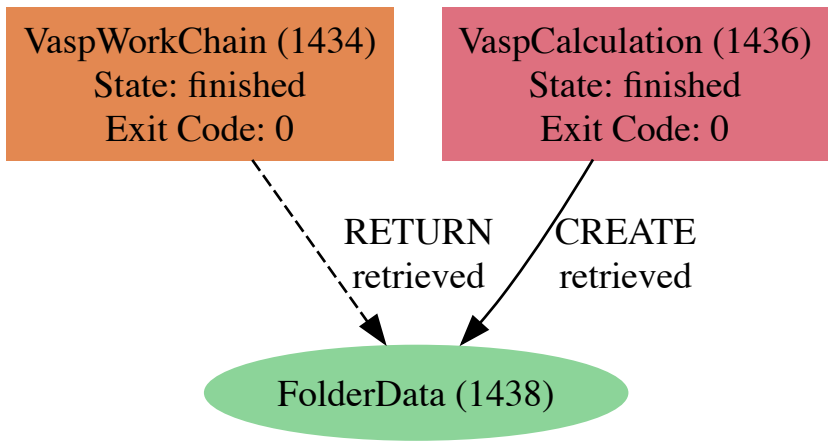
[13]:
```
! ~/envs/aiida/bin/verdi node repo cat 1438 OUTCAR
```

```
 vasp.6.3.0 20Jan22 (build Jul 07 2023 14:56:46) complex

 executed on             LinuxIFC date 2023.08.09  10:49:14
 running on    1 total cores
 distrk:  each k-point on    1 cores,    1 groups
 distr:  one band on NCORE=   1 cores,    1 groups


--------------------------------------------------------------------------------
------------------------
```

```
                    Average memory used (kb):          N/A

                           Minor page faults:        21322
                           Major page faults:          390
                  Voluntary context switches:         2611
```

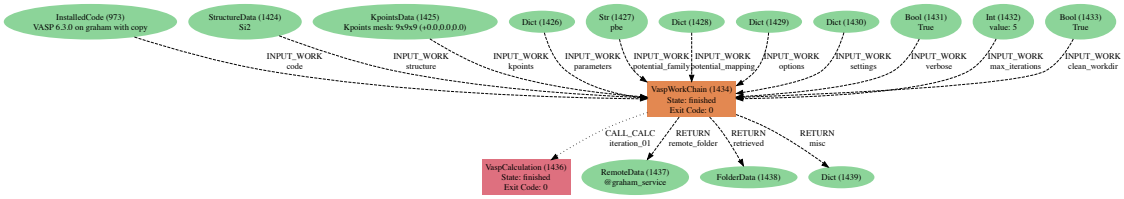
[16]:
```python
from aiida.tools.visualization import Graph
graph = Graph()
calc_node = orm.load_node(1438)
graph.add_incoming(calc_node, annotate_links="both")
graph.add_outgoing(calc_node, annotate_links="both")
graph.graphviz
```

[16]:



[17]:
```python
from aiida import orm
from aiida.tools.visualization import Graph
graph = Graph()
calc_node = orm.load_node(1434)
graph.add_incoming(calc_node, annotate_links="both")
graph.add_outgoing(calc_node, annotate_links="both")
graph.graphviz
```

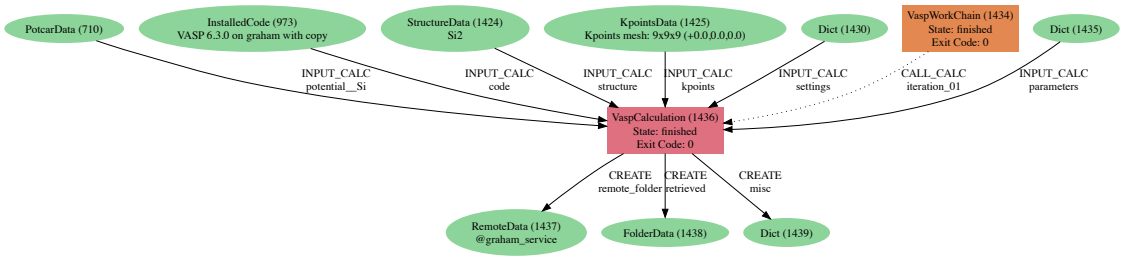
[17]:



[18]:
```python
from aiida.tools.visualization import Graph
graph = Graph()
```

```
calc_node = orm.load_node(1436)
graph.add_incoming(calc_node, annotate_links="both")
graph.add_outgoing(calc_node, annotate_links="both")
graph.graphviz
```

[18]:



[51]:

```
from aiida import orm

node = orm.load_node(pk=1434)
output = node.outputs['misc']

print(output.get_dict())

result = output.get_dict()
print(result["total_energies"])
```

```
{'version': '6.3.0', 'run_stats': {'user_time': 2.683, 'system_time': 0.225,
'elapsed_time': 9.792, 'mem_usage_base': 30000.0, 'mem_usage_grid': 1216.0,
'mem_usage_wavefun': 1285.0, 'mem_usage_fftplans': 800.0, 'average_memory_used':
None, 'maximum_memory_used': 169424.0, 'mem_usage_nonl-proj': 615.0,
'total_cpu_time_used': 2.908, 'mem_usage_one-center': 6.0}, 'run_status':
{'nsw': 0, 'nelm': 60, 'nbands': 8, 'finished': True, 'ionic_converged': None,
'contains_nelm_breach': False, 'electronic_converged': True,
'last_iteration_index': [1, 10], 'consistent_nelm_breach': False},
'maximum_force': 0.0, 'notifications': [], 'maximum_stress': 17.91131062,
'total_energies': {'energy_extrapolated': -10.79613809,
'energy_extrapolated_electronic': -10.79613809}}
{'energy_extrapolated': -10.79613809, 'energy_extrapolated_electronic':
-10.79613809}
```

[46]:

```
# can also print inputs

node = orm.load_node(pk=1434)

input = node.inputs['parameters']
print(input.get_dict())
```

```
{'incar': {'prec': 'NORMAL', 'ediff': 0.0001, 'encut': 200, 'ialgo': 38,
'sigma': 0.1, 'ismear': -5}}
```

6 Multiple lattice constants

[35]:
```
"""
An example call script that performs a relaxation of a structure.

Performs a relaxation of the standard silicon structure.
"""
# pylint: disable=too-many-arguments
import numpy as np

from aiida import load_profile
from aiida.common.extendeddicts import AttributeDict
from aiida.engine import submit, run
from aiida.orm import Code
from aiida.plugins import DataFactory, WorkflowFactory

load_profile()

# parallel arrays
output_nodes = []
lattice_k = []

def get_structure(alat):
    """
    Set up Si primitive cell

    fcc Si:
       alat
       0.5000000000000000    0.5000000000000000    0.0000000000000000
       0.0000000000000000    0.5000000000000000    0.5000000000000000
       0.5000000000000000    0.0000000000000000    0.5000000000000000
    Si
       1
    Cartesian
    0.0000000000000000  0.0000000000000000  0.0000000000000000

    """

    structure_data = DataFactory('structure')
    lattice = np.array([[.5, .5, 0], [0, .5, .5], [.5, 0, .5]]) * alat
    structure = structure_data(cell=lattice)
    for pos_direct in ([[0.0, 0.0, 0.0]]):
        pos_cartesian = np.dot(pos_direct, lattice)
        structure.append_atom(position=pos_cartesian, symbols='Si')
    return structure
```

```python
def main(code_string, incar, kmesh, structure, potential_family,␣
↪potential_mapping, options):
    """Main method to setup the calculation."""

    # We set the workchain you would like to call
    workchain = WorkflowFactory('vasp.relax')

    # And finally, we declare the options, settings and input containers
    settings = AttributeDict()
    inputs = AttributeDict()

    # Organize settings
    settings.parser_settings = {}

    # Set inputs for the following WorkChain execution
    # Set code
    inputs.code = Code.get_from_string(code_string)
    # Set structure
    inputs.structure = structure
    # Set k-points grid density
    kpoints = DataFactory('core.array.kpoints')()
    kpoints.set_kpoints_mesh(kmesh)
    inputs.kpoints = kpoints
    # Set parameters
    inputs.parameters = DataFactory('core.dict')(dict=incar)
    # Set potentials and their mapping
    inputs.potential_family = DataFactory('core.str')(potential_family)
    inputs.potential_mapping = DataFactory('core.dict')(dict=potential_mapping)
    # Set options
    inputs.options = DataFactory('core.dict')(dict=options)
    # Set settings
    inputs.settings = DataFactory('core.dict')(dict=settings)
    # Set workchain related inputs, in this case, give more explicit output to␣
↪report
    inputs.verbose = DataFactory('core.bool')(True)

    # Relaxation related parameters that is passed to the relax workchain
    relax = AttributeDict()
    # Turn on relaxation
    relax.perform = DataFactory('core.bool')(True)
    # Select relaxation algorithm
    relax.algo = DataFactory('core.str')('cg')
    # Set force cutoff limit (EDIFFG, but no sign needed)
    relax.force_cutoff = DataFactory('core.float')(0.01)
    # Turn on relaxation of positions (strictly not needed as the default is on)

    # The three next parameters are for the ISIF setting
```

```
    relax.positions = DataFactory('core.bool')(True)
    # Turn on relaxation of the cell shape (defaults to False)
    relax.shape = DataFactory('core.bool')(True)
    # Turn on relaxation of the volume (defaults to False)
    relax.volume = DataFactory('core.bool')(False)

    # Set maximum number of ionic steps
    relax.steps = DataFactory('core.int')(100)
    # Set the relaxation parameters on the inputs
    inputs.relax = relax

    # Submit the requested workchain with the supplied inputs
    output = submit(workchain, **inputs)
    return output


if __name__ == '__main__':

    global output_nodes
    global lattice_k

    # Code_string is chosen among the list given by 'verdi code list'
    CODE_STRING = 'vaspgrahamcopy@graham_service'

    # POSCAR equivalent
    # Set the silicon structure
    # STRUCTURE = get_structure()

    # INCAR equivalent
    # Set input parameters
    INCAR = {'incar': {'encut': 240, 'ismear': 0, 'sigma': 0.1, 'system': 'test␣
 ↪system'}}

    # KPOINTS equivalent
    # Set kpoint mesh
    KMESH = [9, 9, 9]

    # POTCAR equivalent
    # Potential_family is chosen among the list given by
    # 'verdi data vasp-potcar listfamilies'
    POTENTIAL_FAMILY = 'pbe'
    # The potential mapping selects which potential to use, here we use the␣
 ↪standard
    # for silicon, this could for instance be {'Si': 'Si_GW'} to use the GW␣
 ↪ready
    # potential instead
    POTENTIAL_MAPPING = {'Si': 'Si'}
```

```
    # Jobfile equivalent
    # In options, we typically set scheduler options.
    # See https://aiida.readthedocs.io/projects/aiida-core/en/latest/scheduler/
↪index.html
    # AttributeDict is just a special dictionary with the extra benefit that
    # you can set and get the key contents with mydict.mykey, instead of␣
↪mydict['mykey']
    OPTIONS = AttributeDict()
    OPTIONS.account = 'def-ctetsass'
    OPTIONS.qos = ''
    OPTIONS.resources = {'num_machines': 1, 'num_mpiprocs_per_machine': 2}
    OPTIONS.queue_name = ''
    OPTIONS.max_wallclock_seconds = 3600
    OPTIONS.max_memory_kb = 1024000

    for lattice_constant in [3.5, 3.6, 3.7, 3.8, 3.9, 4.0, 4.1, 4.2, 4.3]:
        STRUCTURE = get_structure(lattice_constant)
        output = main(CODE_STRING, INCAR, KMESH, STRUCTURE, POTENTIAL_FAMILY,␣
↪POTENTIAL_MAPPING, OPTIONS)
        output_nodes.append(output)
        lattice_k.append(lattice_constant)
```

[38]:
```
print(output_nodes)
```

```
[<WorkChainNode: uuid: 4f830012-b6f2-4cd4-a304-a4bbd68c35dc (pk: 1466)
(aiida.workflows:vasp.relax)>, <WorkChainNode: uuid:
5d0cb7d2-9a3e-43f2-a268-a025dffeb2aa (pk: 1493) (aiida.workflows:vasp.relax)>,
<WorkChainNode: uuid: 9fbb0521-70f9-4f47-8f1a-126853de4c0e (pk: 1524)
(aiida.workflows:vasp.relax)>, <WorkChainNode: uuid:
3bbf75b6-a380-4f9d-9ed3-38ad0777b138 (pk: 1553) (aiida.workflows:vasp.relax)>,
<WorkChainNode: uuid: e8cffe85-57f4-434b-9351-4f5f9f85421c (pk: 1585)
(aiida.workflows:vasp.relax)>, <WorkChainNode: uuid:
9567b4a0-8f5c-48bb-9ff6-d1031de87732 (pk: 1617) (aiida.workflows:vasp.relax)>,
<WorkChainNode: uuid: bc88c256-288b-42aa-adbb-bf1032e96d0c (pk: 1648)
(aiida.workflows:vasp.relax)>, <WorkChainNode: uuid:
ea3269c0-3290-4a49-b260-f6f1355c003c (pk: 1676) (aiida.workflows:vasp.relax)>,
<WorkChainNode: uuid: 54939986-34e8-42ff-a062-b532bf86d0d8 (pk: 1709)
(aiida.workflows:vasp.relax)>]
```

[36]:
```
! ~/envs/aiida/bin/verdi process list
```

```
  PK  Created    Process label    Process State    Process status
----  ---------  ---------------  ---------------
----------------------------------------
1401  43m ago    VaspWorkChain      Created
1412  41m ago    VaspWorkChain      Created
1423  37m ago    VaspWorkChain      Created
```

```
1466  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1525
1493  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1526
1524  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1558
1525  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1557
1526  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1616
1553  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1595
1585  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1649
1558  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1646
1557  1m ago     VaspCalculation   Waiting        Waiting for transport task:
submit
1617  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1710
1595  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1678
1616  1m ago     VaspCalculation   Waiting        Waiting for transport task:
submit
1648  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1716
1646  1m ago     VaspCalculation   Waiting        Waiting for transport task:
submit
1649  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1708
1676  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1721
1678  1m ago     VaspCalculation   Waiting        Waiting for transport task:
submit
1709  1m ago     RelaxWorkChain    Waiting        Waiting for child processes:
1715
1708  1m ago     VaspCalculation   Waiting        Monitoring scheduler: job
state UNKNOWN
1710  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1723
1715  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1720
1716  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1725
1720  1m ago     VaspCalculation   Waiting        Waiting for transport task:
submit
1721  1m ago     VaspWorkChain     Waiting        Waiting for child processes:
1727
```

```
1723  1m ago     VaspCalculation    Waiting        Waiting for transport task:
submit
1725  1m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state UNKNOWN
1727  1m ago     VaspCalculation    Waiting        Waiting for transport task:
submit

Total results: 30

Report: last time an entry changed state: 21s ago (at 11:14:12
on 2023-08-09)
Report: Checking daemon load… OK
Report: Using 8% of the available daemon worker slots.
```

[37]:
```
!ssh aiida_ctetsass@graham.computecanada.ca sq
```

```
/usr/bin/id: cannot find name for group ID 13200135
            JOBID     USER              ACCOUNT           NAME  ST  TIME_LEFT
NODES CPUS TRES_PER_N MIN_MEM NODELIST (REASON)
         10010949 aiida_ct     def-ctetsass_cpu     aiida-1708  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010951 aiida_ct     def-ctetsass_cpu     aiida-1725  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010959 aiida_ct     def-ctetsass_cpu     aiida-1557  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010965 aiida_ct     def-ctetsass_cpu     aiida-1616  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010966 aiida_ct     def-ctetsass_cpu     aiida-1646  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010967 aiida_ct     def-ctetsass_cpu     aiida-1678  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010970 aiida_ct     def-ctetsass_cpu     aiida-1720  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010971 aiida_ct     def-ctetsass_cpu     aiida-1723  PD    1:00:00
1    2        N/A   1000M  (Priority)
         10010972 aiida_ct     def-ctetsass_cpu     aiida-1727  PD    1:00:00
1    2        N/A   1000M  (Priority)
```

[43]:
```
! ~/envs/aiida/bin/verdi process list
```

```
  PK  Created    Process label    Process State    Process status
----  ---------  ---------------  ---------------
--------------------------------------
1401  47m ago    VaspWorkChain      Created
1412  44m ago    VaspWorkChain      Created
1423  41m ago    VaspWorkChain      Created
1466  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1525
1493  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
```

```
1526
1524  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1558
1525  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1557
1526  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1616
1553  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1595
1585  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1649
1558  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1646
1557  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1617  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1710
1595  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1678
1616  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1648  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1716
1646  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1649  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1708
1676  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1721
1678  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1709  5m ago     RelaxWorkChain     Waiting        Waiting for child processes:
1715
1708  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1710  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1723
1715  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1720
1716  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1725
1720  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1721  5m ago     VaspWorkChain      Waiting        Waiting for child processes:
1727
1723  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED
1725  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
```

```
state QUEUED
1727  5m ago     VaspCalculation    Waiting        Monitoring scheduler: job
state QUEUED

Total results: 30

Report: last time an entry changed state: 3m ago (at 11:14:36
on 2023-08-09)
Report: Checking daemon load… OK
Report: Using 8% of the available daemon worker slots.
```

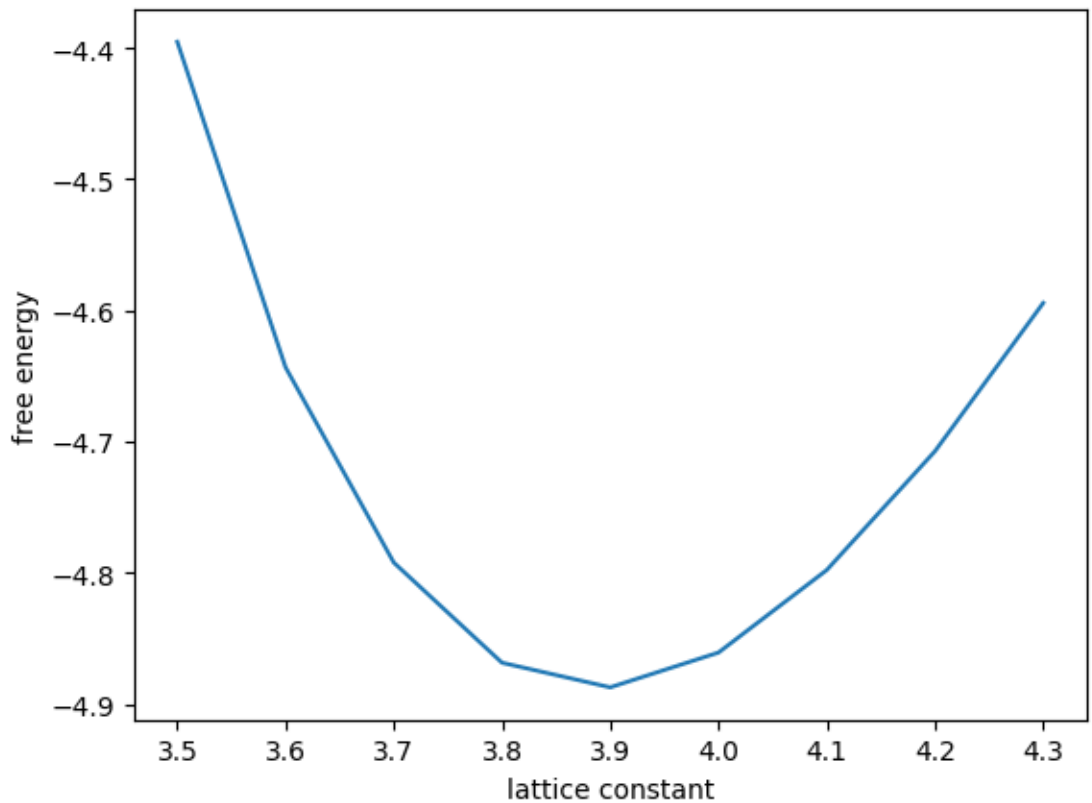
[47]:
```
from aiida import orm

#EOS = []
for i in range(len(output_nodes)):

        pk = output_nodes[i].pk
        lk = lattice_k[i]

        node = orm.load_node(pk=pk)
        misc = node.outputs['misc'].get_dict()

       # EOS.append([lattice_constant,␣
 ↪misc['total_energies']['energy_extrapolated']])
        print(str(lk) + ': ' +␣
 ↪str(misc['total_energies']['energy_extrapolated']))

    # Write volume and total energies to file
    #with open('eos', 'w', encoding='utf8') as file_object:
    #    for item in EOS:
    #        file_object.write(f'{item[0]} {item[1]}\n')
```

```
3.5: -4.39519434
3.6: -4.64331877
3.7: -4.79221948
3.8: -4.86861581
3.9: -4.88738127
4.0: -4.86085152
4.1: -4.79794631
4.2: -4.7073369
4.3: -4.59430816
```

[48]:
```
from matplotlib import pyplot as plt

energies = []
for i in range(len(output_nodes)):

        misc = orm.load_node(pk=output_nodes[i].pk).outputs['misc'].get_dict()
```

```
        energy = misc['total_energies']['energy_extrapolated']
        energies.append(energy)
        lk = lattice_k[i]

plt.xlabel('lattice constant')
plt.ylabel('free energy')
plt.plot(lattice_k, energies)
```

[48]: [<matplotlib.lines.Line2D at 0x115fedf90>]



7 Querying data

[39]:
```
from aiida.orm import QueryBuilder
from aiida.orm import load_node, Node, Group, Computer, User, CalcJobNode, Code

query = QueryBuilder()
query.append(Node)
print("count of Nodes ",query.count())

query = QueryBuilder()
```

```
query.append(CalcJobNode)
print("count of calculation nodes", query.count())
```

```
count of Nodes  1707
count of calculation nodes 35
```

[40]:
```
from aiida.plugins import CalculationFactory, DataFactory

StructureData = DataFactory('structure')

query = QueryBuilder()
query.append(StructureData)
query.limit(5)
for structure, in query.iterall():
    print(structure.get_formula())
```

```
Si2
Si2
Si
Si2
Si2
```

[42]:
```
node = orm.load_node(pk=1434)
a=node.get_incoming().all_nodes()
print(a)
```

```
[<InstalledCode: Remote code 'vaspgrahamcopy' on graham_service pk: 973, uuid:
7e36a338-22bd-47fa-8cfa-24876082e271>, <StructureData: uuid:
c61fd150-4cf5-4771-9939-3bbd228eeb95 (pk: 1424)>, <KpointsData: uuid:
42f4652a-7364-41f7-9b21-8d3d843ee6b3 (pk: 1425)>, <Dict: uuid:
31c93a4f-7e49-423c-9e7a-9c115a3c5830 (pk: 1426)>, <Str: uuid: 1ac31daf-
ef94-4b16-b21d-d40e67379027 (pk: 1427) value: pbe>, <Dict: uuid:
02007c7f-55ec-463d-b69a-a039f9d90d2d (pk: 1428)>, <Dict: uuid:
371d9eb6-4676-4c4c-a127-f2b4f0be9351 (pk: 1429)>, <Dict: uuid:
f18c4c40-9926-4cb5-a5b0-866a95b89e75 (pk: 1430)>, <Bool: uuid:
659aa029-e718-4f0f-b8d2-e463d65e6a19 (pk: 1431) value: True>, <Int: uuid:
4b3bd133-bb00-4b40-9846-aca3bf2cffd1 (pk: 1432) value: 5>, <Bool: uuid:
1768df3f-9b09-4d96-b4e2-60aba068f51d (pk: 1433) value: True>]
```

[ ]:
```

```