MARVEL

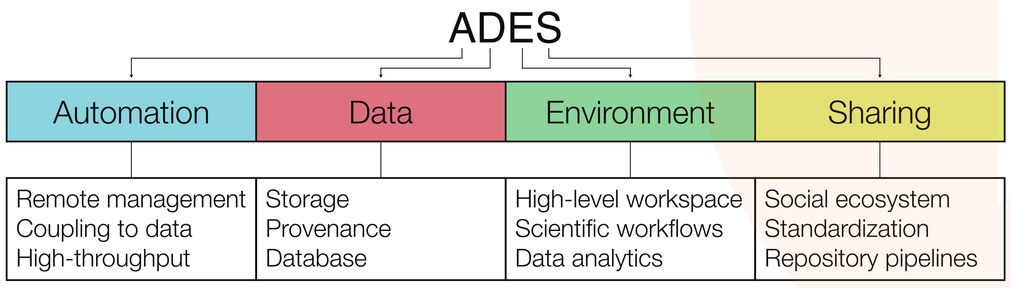
NATIONAL CENTRE OF COMPETENCE IN RESEARCH

AiiDA - Automated interactive infrastructure and database for computational science

S. Zoupanos[1], L. Kahle[1], S. P. Huber[1], M. Uhrin[1], N. Mounet[1], R. A. Häuselmann[1], D. Gresch[5], S. Kumbhar[1], L. Talirz[1], A. Cepellotti[6], F. Gargiulo[1] , A. Merkys[4]
B. Kozinsky[2,3], N. Marzari[1], G. Pizzi[1]

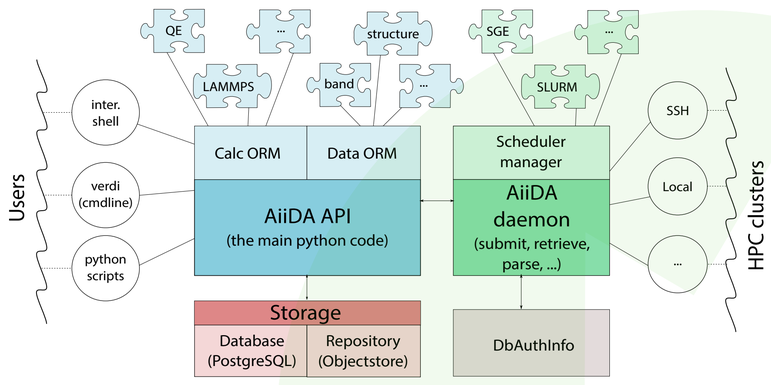
[1] Theory and Simulation of Materials (THEOS), and National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
[2] Robert Bosch LLC Research and Technology Center, Cambridge, MA 02139, USA
[3] Harvard University, Cambridge, MA 02138, USA
[4] Vilnius University, Vilnius 01513, Lithuania
[5] Institute for Theoretical Physics, ETH Zurich, 8092 Zurich
[6] University of California, Berkeley, CA, USA

1. Motivation



Four pillars of an infrastructure for computational science

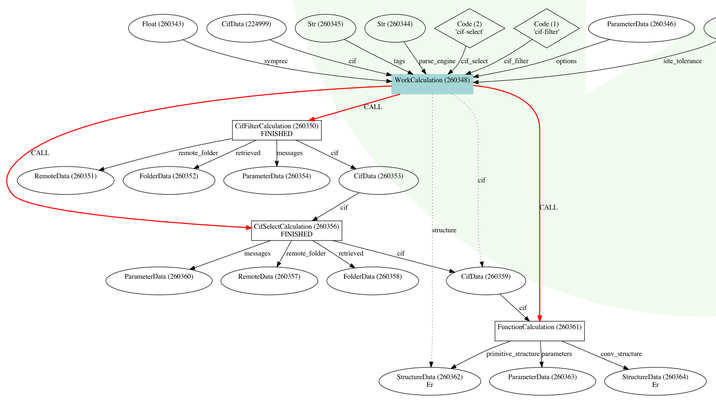
2. Infrastructure



Main features

- Written in Python 2.7
- SQL backend with access through a Python ORM
- Local connection to clusters, or via ssh
- Interface to various job schedulers
- Daemon with remote management and workflow execution manager
- REST APIs using Flask and a translation layer for AiiDA data
- Plugin management system and extended code support
- Easy sharing of the results with other users in the community

3. AiiDA graph, nodes, link types and their properties

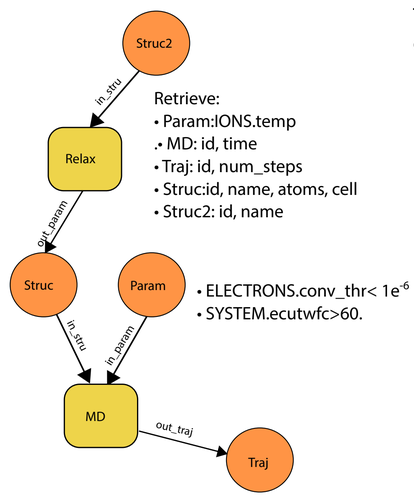


Two main categories of AiiDA nodes:
- Calculation
- Data

Multiple types of calculations:
- WorkCalculations orchestrate workflow execution
- JobCalculations used for simulation submission

Available links:
- INPUT: Data --> Calculation
- CREATE: Calculation --> Data (generated by)
- RETURN: Calculation --> Data (is result of)
- CALL: Calculation --> Calculation (step in workflow)

4. QueryBuilder with full graph traversal



Retrieve:
- Param:IONS.temp
- MD: id, time
- Traj: id, num_steps
- Struc:id, name, atoms, cell
- Struc2: id, name

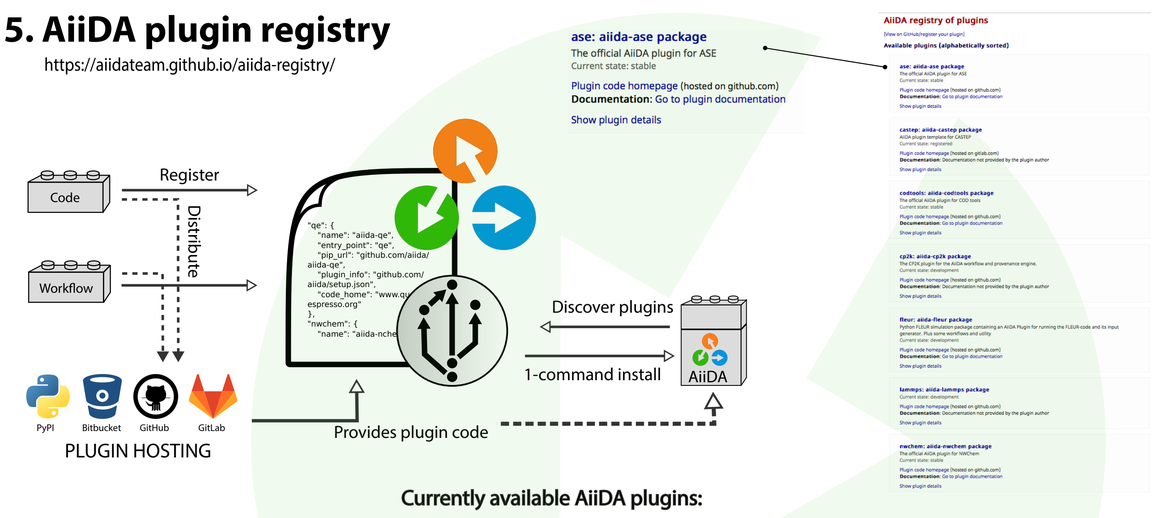
- ELECTRONS.conv_thr< $1e^{-6}$
- SYSTEM.ecutwfc>60.

The AiiDA querying API was redesigned to perform complex graph queries with any combination of filtering and projections on nodes

```
queryhelp = {
    'path':[
        {'cls':StructureData, 'tag':'Struc2'},
        {'cls':PwCalculation, 'tag':'Relax'},
        {'cls':StructureData, 'tag':'Struc', 'output_of':'Relax'},
        {'cls':PwCalculation, 'tag':'MD', 'output_of':'Struc'},
        {'cls':ParameterData, 'tag':'Param', 'input_of':'MD'},
        {'cls':TrajectoryData,'tag':'Traj', 'output_of':'MD'},
    ],
    'project':{
        'Param': ['attributes.IONS.tempw'],
        'MD':    ['id', 'ctime'],
        'Traj':  ['id', 'attributes.array|positions.0'],
        'Struc': ['id', 'label', 'attributes.sites', 'attributes.cell'],
        'Struc2':['id', 'label'],
    },
    'filters':{
        ParameterData:{
            'and':[
                {'attributes.ELECTRONS.conv_thr':{'<':1e-6}},
                {'attributes.SYSTEM.ecutwfc':{'>':60.}}]
        }
    }
}
```

5. AiiDA plugin registry

https://aiidateam.github.io/aiida-registry/



Currently available AiiDA plugins:

- Quantum ESPRESSO PW, CP, PH, PP PROJWFC, DOS, PDOS, NEB (EPFL)
- Wannier90 (EPFL, ETHZ)
- YAMBO (CNR-NANO, EPFL)
- GPAW (EPFL)
- LAMMPS* (Kyoto Univ)
- RASPA* (EPFL)
- ASE calculators (EPFL)
- Phonopy (EPFL, Kyoto Univ.)
- NWChem (EPFL, Vilnius Univ.)
- CODTools (EPFL, Vilnius Univ.)
- CP2K (Google, ETHZ, UZH, EPFL)
- zeo++* (EPFL)
- CASTEP* (Cambridge)
- FLEUR (Jülich, EPFL)
- Exciting* (CSCS, EPFL)
- SIESTA (Oviedo Univ., ICNZ, EPFL)
- VASP (EPFL, Trinity College Dublin, Bosch ETHZ, SCITAS, Gent Univ)

* work in progress

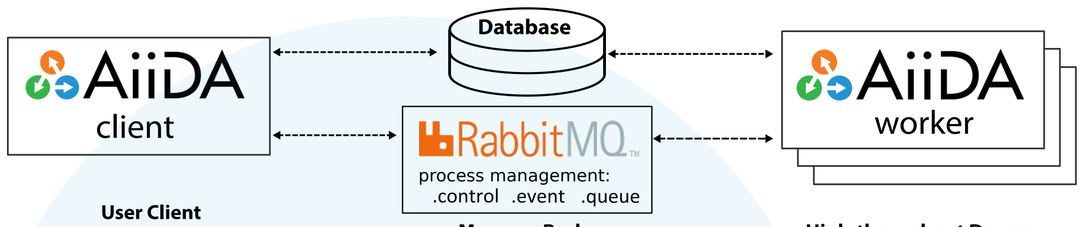
6. Flexible and powerful workflow language and engine

```
class PwRelaxWorkChain(WorkChain):
    def define(cls, spec):
        spec.input('kpoints', valid_type=KpointsData)
        spec.input('structure', valid_type=StructureData)
        spec.input('parameters', valid_type=ParameterData)
        spec.input('options', valid_type=ParameterData, required=False)
        spec.outline(
            cls.setup,
            cls.validate_inputs,
            while_(cls.should_run_relax)(
                cls.run_relax,
                cls.inspect_relax,
            ),
            if_(cls.should_run_final_scf)(
                cls.run_final_scf,
            ),
            cls.results,
        )
        spec.output('output_structure', valid_type=StructureData)
        spec.output('output_parameters', valid_type=ParameterData)
```

Input definition
- Clear overview of inputs
- Automatic validation
- Optional inputs

Workflow logical outline
- Clear definition of logical structure
- Logical operators for maximum flexibility
- Native Python logical syntax

Output definition
- Clear overview of outputs
- Automatic validation



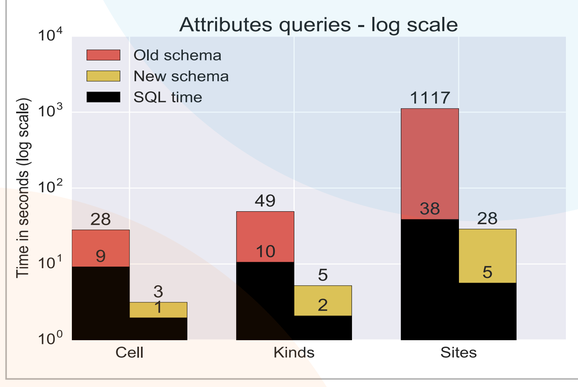
User Client
Client with command line interface as the user endpoint
Provides commands to launch, control and query workflows

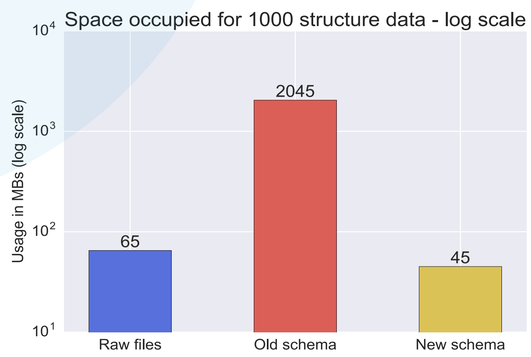
Message Broker
Handles communication, failures and distribution of tasks to workers

High-throughput Daemon
Workers can be added to deal with increased workloads
Up to 10,000 simultaneous jobs

7. 65x performance boost and 45x space improvements

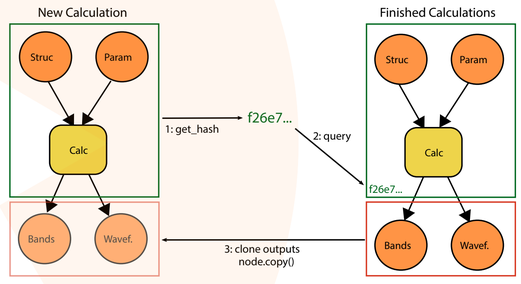
- AiiDA API abstracted to support multiple backends, two backends currently implemented (SQLAlchemy, Django)
- SQLAlchemy (≥0.9) and PostgreSQL (≥9.4) with native support of JSON queries and indexing
- Up to 65x performance boost on queries and command line operations related to JSON encoded information comparing to Django backend
- Up to 45x space improvements comparing to backend and 30% improvement comparing to raw files for structure data
- Stability tests based on Quantum ESPRESSO benchmark revealed accuracy improvements comparing to Django backend





8. Caching

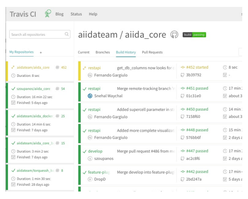
Collaboration between MARVEL groups

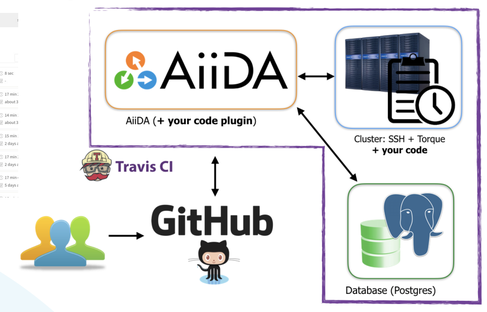


New calculation caching mechanism ensuring computational resources savings.
- For every executed calculation, a unique key is created based on the calculation inputs
- The outputs of the calculation can be referenced by the created unique key
- When a new calculation comes with the same unique key then the results are automatically copied from the previously executed calculation

9. Extended testing and continuous integration

- Extended test code coverage and code stability with continuous integration
- Advanced orchestration for using Docker Compose and Travis. Useful for plugin testing
- More than 450 tests in total and more than 350 backend specific tests
- Different testing levels: Unit tests, Integration tests, System tests
- Tests run per pull request and commit. Pull requests can not be merged without test success





10. Knowledge transfer

Workshops for users and developers

2018 - 1 AiiDA workshop: CINECA (IT) - May 2018 - 25 part
2017 - 4 AiiDA workshops: Lausanne (CH) - May 2017 - 50 part., Lausanne (CH) - Mar 2017 - 50 part., Trieste (IT) - Jan 2017 - 75part
2016 - 3 AiiDA workshops: Trieste (IT) - Jul 2016 - 100 part., Lausanne (CH) - Jun 2016 - 40 part., Kyoto (JAP) - Mar 2016 - 20 part
2015 - 3 AiiDA workshops: Trieste (IT) - Dec 2015 - 10 part., Lausanne (CH) - Nov 2015 - 40 part., Berlin (DE) - Feb 2015 - 40 part
2014 - 1 AiiDA workshop: Zurich (CH) - Oct 2014 - 30 part

Releases

- Latest release 0.12 (1.0.0a1 planned)
- New release every ~2 months
- Backwards-compatibility ensured

Support and community

- Website: http://www.aiida.net
- Documentation: https://aiida-core.readthedocs.io
- Mailing list: aiidausers@googlegroups.com
- Issue tracker: https://github.com/aiidateam/aiida_core/issues

Coding sprint weeks

2018: Lausanne (CH) - Feb 2018 - 10 part.
2017: Leukerbad (CH) - Oct 2017 - 15 part.
2016: Leysin (CH) - Dec 2016 - 20 part.

Reference

[1] G. Pizzi, A. Cepellotti, R. Sabatini, N. Marzari, and B. Kozinsky, AiiDA: automated interactive infrastructure and database for computational science, Comp. Mat. Sci 111, 218-230 (2016)

FNSNF
Swiss National Science Foundation
The National Centres of Competence in Research (NCCR) are a research instrument of the Swiss National Science Foundation