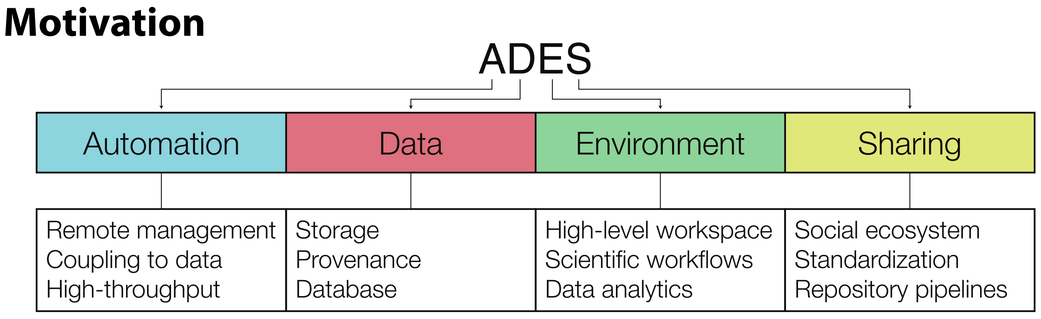
The AiiDA platform: Recent developments

S. Zoupanos[1], M. Uhrin[1], S. P. Huber[1], R. A. Häuselmann[1], L. Kahle[1], N. Mounet[1], D. Gresch[5], S. Kumbhar[1], T. Müller[7], L. Talirz[1], J.Boullier[1], A. Cepellotti[6], F. Gargiulo[1] , A. Merkys[4]
B. Kozinsky[2,3], N. Marzari[1], G. Pizzi[1]

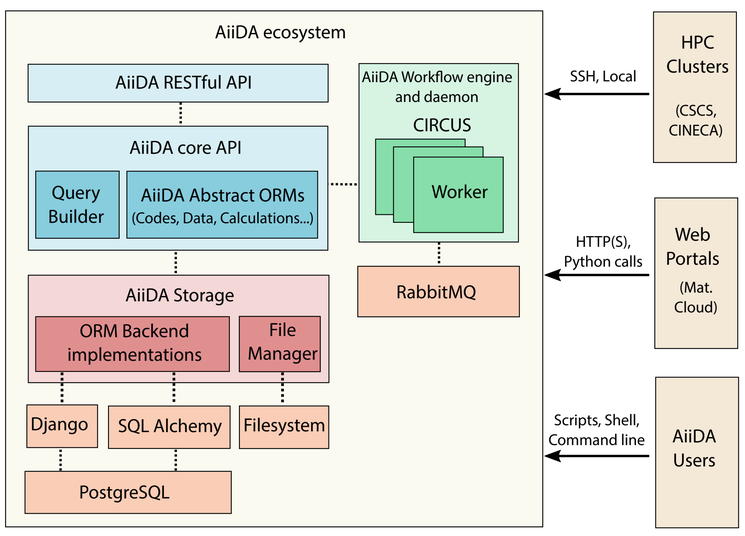
1 Theory and Simulation of Materials (THEOS), and National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
2 Robert Bosch LLC Research and Technology Center, Cambridge, MA 02139, USA
3 Harvard University, Cambridge, MA 02138, USA
4 Vilnius University, Vilnius 01513, Lithuania
5 Institute for Theoretical Physics, ETH Zurich, 8092 Zurich, Switzerland
6 University of California, Berkeley, CA, USA
7 Institute for Physical Chemistry, University of Zurich, 8006 Zurich, Switzerland

1. Motivation



Four pillars of an infrastructure for computational science [1]

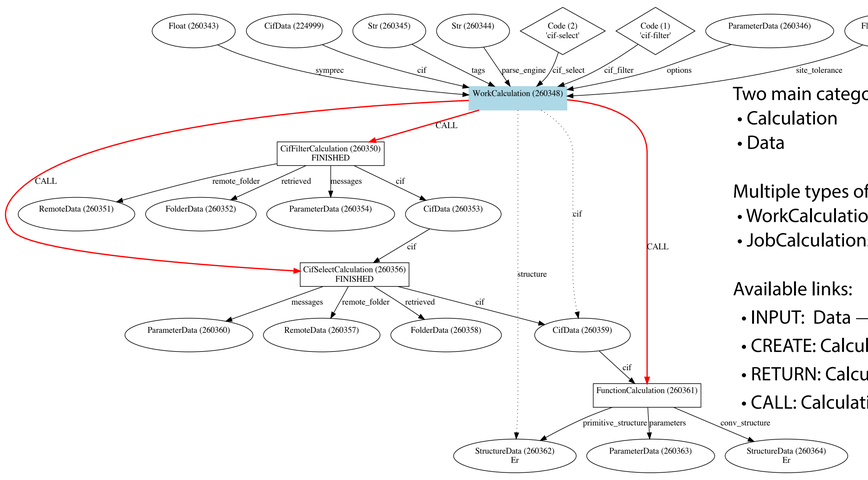
2. AiiDA architecture



Main features

- AiiDA supports Python 2.7 and very soon Python 3.5 (Pull request #1682 - Collaboration between MARVEL groups)
- 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. OPTiMaDe implementation ongoing.
- Plugin management system and extended code support
- Easy sharing of the results with other users in the community
- New high-throughput workflow engine based on RabbitMQ message broker following an event based approach and daemon based on CIRCUS

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

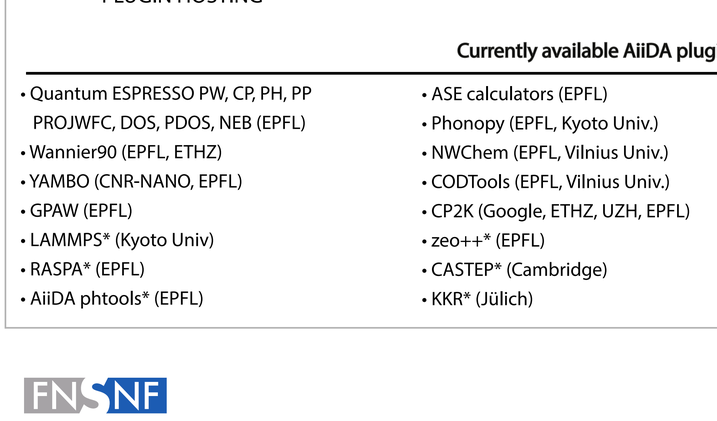


Two main categories of AiiDA nodes:
- Calculation
- Data

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 (sub-step in workflow)

On going work on link types and entities Finalization by Nov 2018



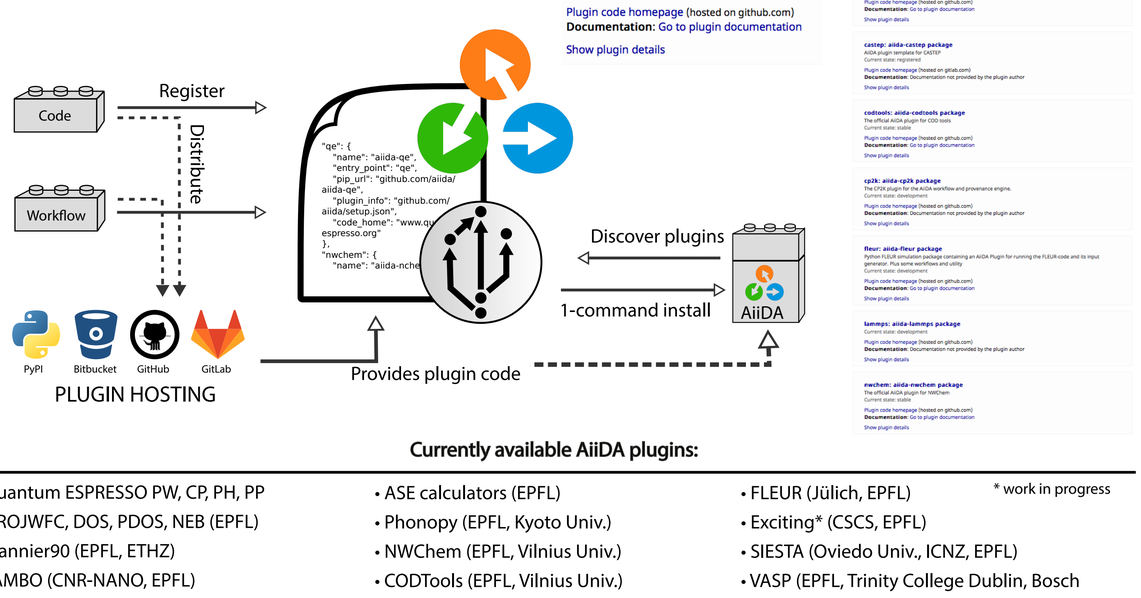
Aim: Formalization of AiiDA's provenance model

Type-dependency tree

Entities	*Sub-Entities*
Node	Data, Calculation
Data	Code, StructureData, ParameterData, …
Calculation	WorkCalc, RunCalc
WorkCalc	WorkChainCalc, WorkFunctionCalc, …
RunCalc	RunJobCalc, RunFunctionCalc, …

4. AiiDA plugin registry

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



Currently available AiiDA plugins:

* work in progress

- 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)
- AiiDA phtools* (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)
- KKR* (Jülich)
- FLEUR (Jülich, EPFL)
- Exciting* (CSCS, EPFL)
- SIESTA (Oviedo Univ., ICNZ, EPFL)
- VASP (EPFL, Trinity College Dublin, Bosch ETHZ, SCITAS, Gent Univ)
- DDEC* (EPFL)
- Gollum* (Oviedo Univ.)
- Gudhi* (EPFL)

5. Flexible and powerful workflow language

```
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)
```

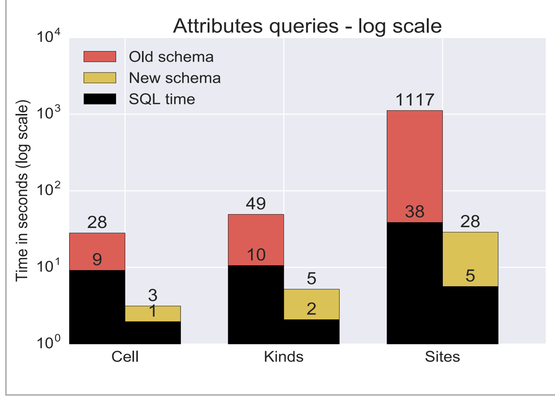
Input definition
- Clear overview of inputs
- Automatic validation
- Optional inputs

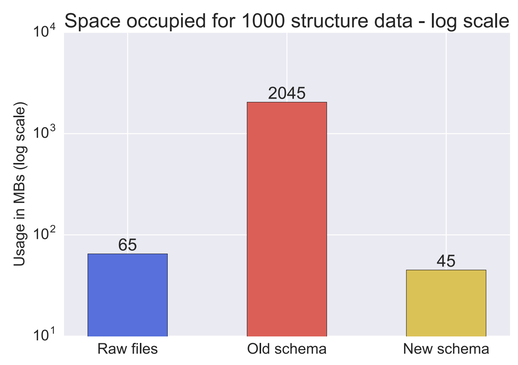
Workflow logical outline
- Clear definition of logical structure
- Logical operators for maximum flexibility
- Native Python logical syntax

Output definition
- Clear overview of outputs
- Automatic validation

6. 65x performance boost and 45x space improvements

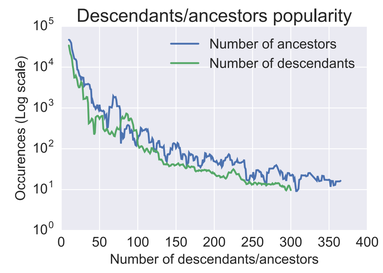
- AiiDA API abstracted to support multiple backends, two backends currently implemented (SQLAlchemy, Django)
- SQLAlchemy (≥0.9) and PostgreSQL (≥9.4) with native support of JSON queries and indexing
- Up to 65x performance boost on queries and command line operations related to JSON encoded information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



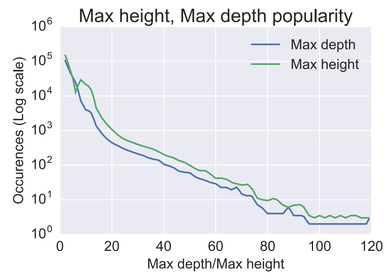


7. On-the-fly graph traversal vs transitive closure table

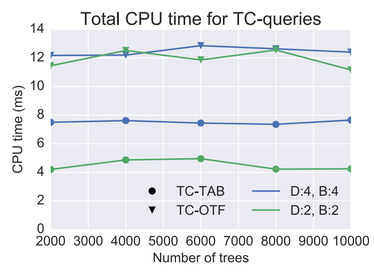
- Population of the transitive closure table (TC-TAB - contains all the paths to ancestors and descendants for every avalable graph node) leads to space explosion even for moderate-sized AiiDA graphs
- Efficient solution implemented for on-the-fly graph traversal (TC-OTF) covering the majority of the cases - no disc space needed
- The lookup of 340 descendant nodes per tree (for 50 trees) using TC-OTF is 50% slower than TC-TAB. However, this case has a **very low probability** (left figure). All more frequent on-the-fly traversals are faster than this and last a **few milliseconds**



Frequencies of the number of the ancestors and descendants of nodes for an 1M node AiiDA graph.

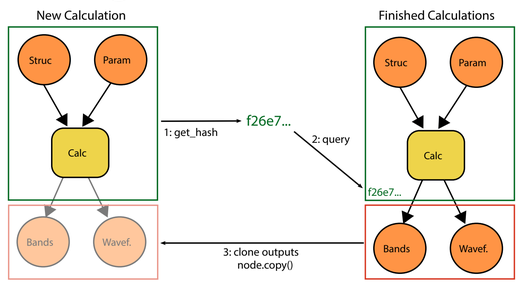


Frequency of different maximum depths/heights for an 1M node AiiDA graph



Lookup of 50 top-level tree nodes using either the on-the-fly traversal (TC-OTF) or the transitive closure table (TC-TAB) for trees of depth D and branching factor B

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. Knowledge transfer

Workshops for users and developers

2018 - 1 AiiDA workshop: CINECA (IT) - May 2018 - 40 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 0.12.2
- 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)

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