香港中文大學
The Chinese University of Hong Kong

AiiDA: Automated Interactive Infrastructure and Database for Computational Science

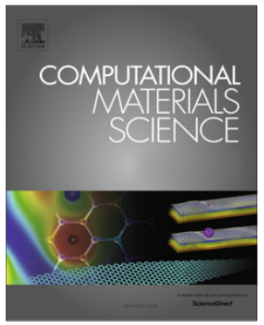
Huiwen Tan
9 Apr 2026

Computational Materials Science 111 (2016) 218–230

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci



AiiDA: automated interactive infrastructure and database for computational science

Giovanni Pizzi [a,*,1], Andrea Cepellotti [a,1], Riccardo Sabatini [a], Nicola Marzari [a], Boris Kozinsky [b]

[a] *Theory and Simulation of Materials (THEOS), and National Centre for Computational Design and Discovery of Novel Materials (MARVEL), École Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland*

[b] *Research and Technology Center, Robert Bosch LLC, Cambridge, MA 02139, USA*

The Current State: Artisan Workshop Model

> Computational science is often still performed using the renaissance model of individual artisans gathered in a workshop, under the guidance of an established practitioner

01 Reproducibility
Fragmented scripts, inconsistent versions, no traceable computation history

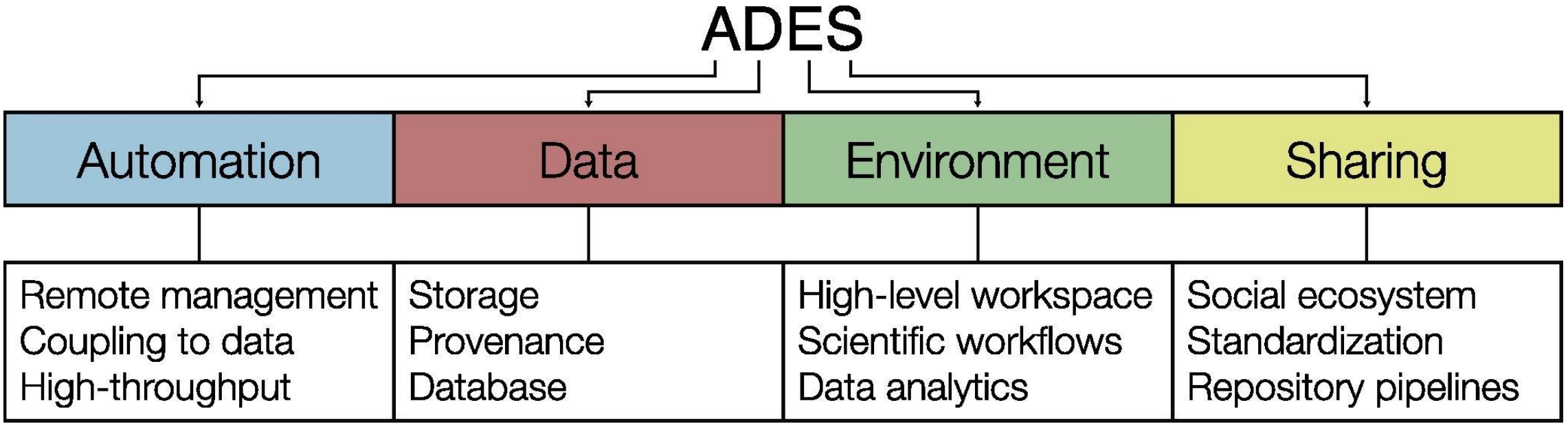
02 Data Management
Ad-hoc file naming, arbitrary directory structures, difficult to search and retrieve

03 Automation
Manual job submission, monitoring, and retrieval, unreliable and not scalable

04 Sharing
Inconsistent data formats, difficult to share and reuse in public databases

We need systematic infrastructure, not just better scripts!

The ADES Model: Four Pillars



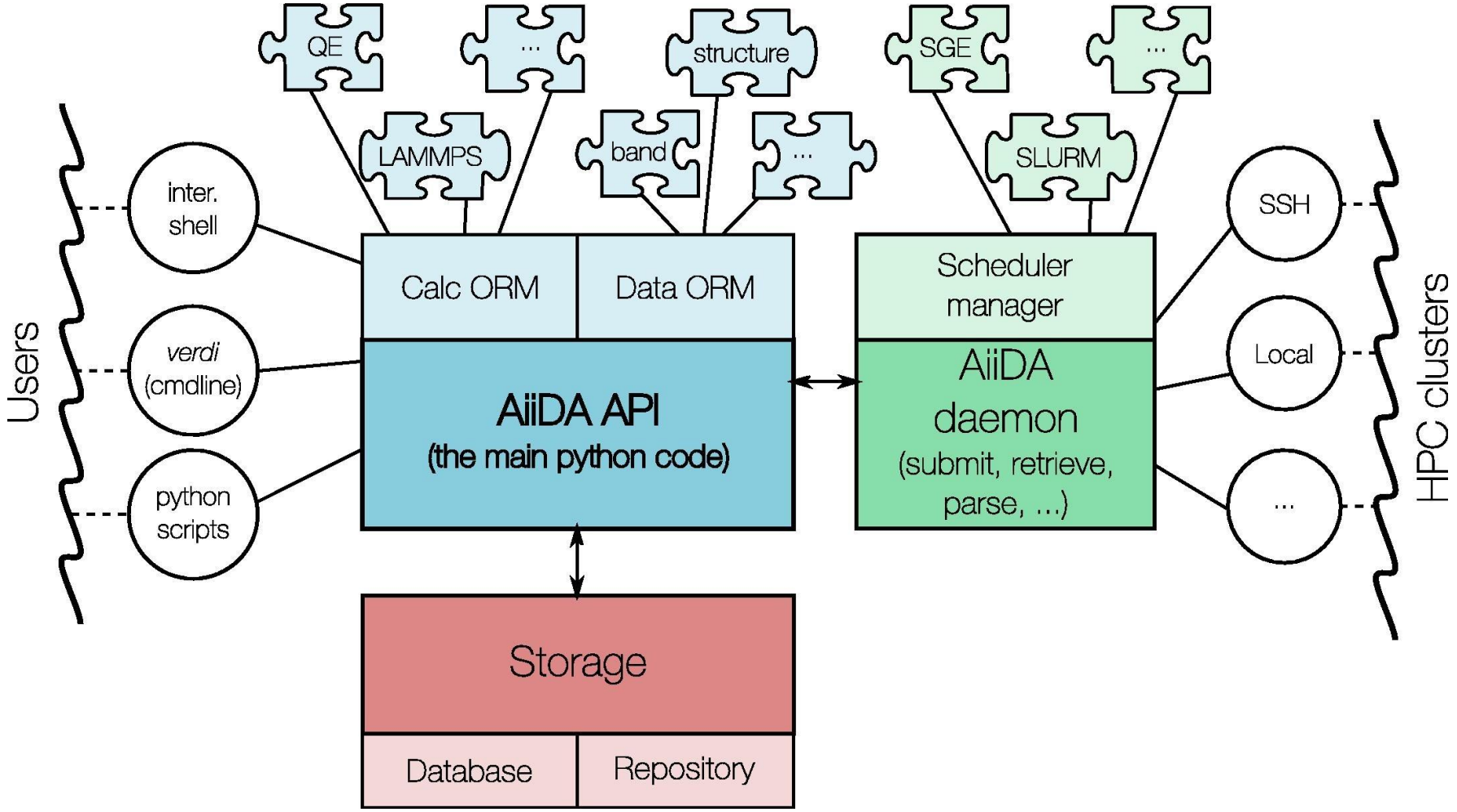
Abstract away the low-level tasks to prepare, submit, retrieve and store automatically large numbers of calculations

Management and persistence of heterogeneous simulation data; database search and query; reproducibility

Natural, high-level environment to encode complex sequences of low-level codes into scientific workflows and turnkey solutions

Social ecosystem to foster interactions, share codes, data and scientific workflows in open repositories, and promote standardized formats

AiiDA System Architecture



Architecture Components

User Layer

`verdi` command line tool

Inspect running / past jobs: `verdi calculation list`
Inspect inputs & outputs: `verdi calculation show <pk>`
Visualize full provenance graph: `verdi graph generate <pk>`

Python scripts / iPython shell

AiiDA API (ORM)

Object-Relational Mapper:
Python classes ↔ Database Objects

Unified Python interface for all AiiDA objects

Core abstraction: Node (Data / Calculation / Code)

Storage Layer

Data is stored with its full lineage

SQL Database for metadata

File Repository for large data files

Daemon

Background process managing all jobs

Manages calculation lifecycle:

submit → monitor → retrieve → parse → store

`NEW → TOSUBMIT → SUBMITTING → WITHSCHEDULER → RETRIEVING → PARSING → FINISHED / FAILED`

Core Concept: Calculation as Function

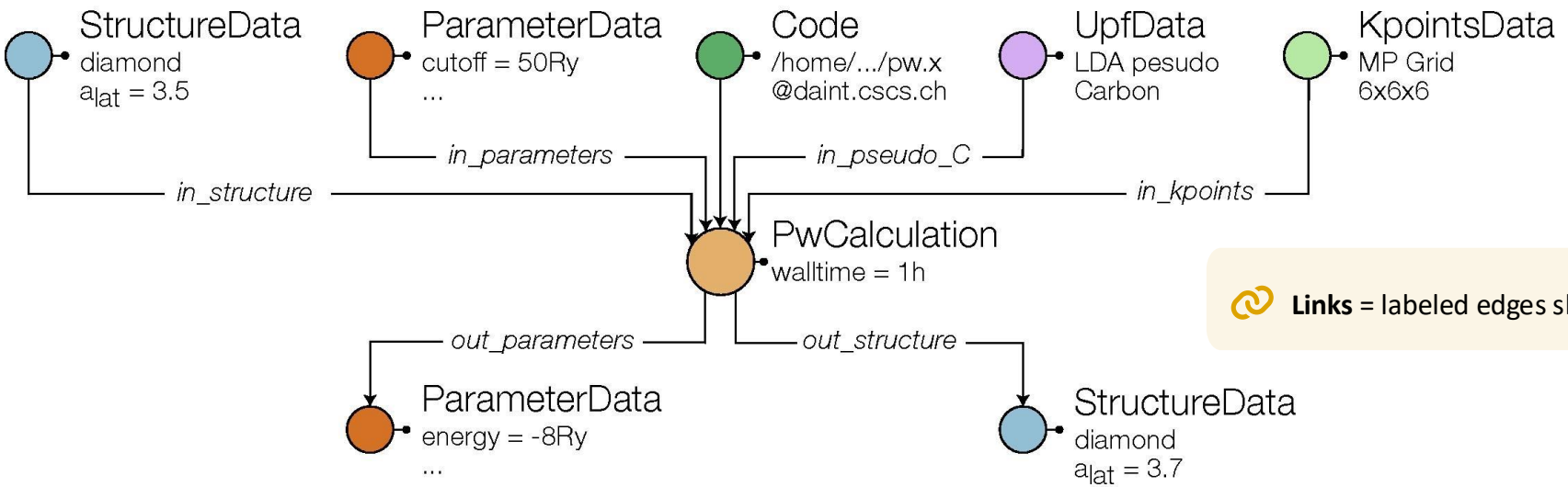
Key Insight

Every calculation is a **function**:

Output Data = **f**(Input Data)

Three Node Types

- **Data nodes** — structures, parameters
- **Calculation nodes** — the "function"
- **Code nodes** — which software was used



Links = labeled edges showing relationships

Directed Acyclic Graph

Directed Acyclic Graph (DAG) is the data model where directed edges capture input–output relationships, recording how results are generated without cycles.

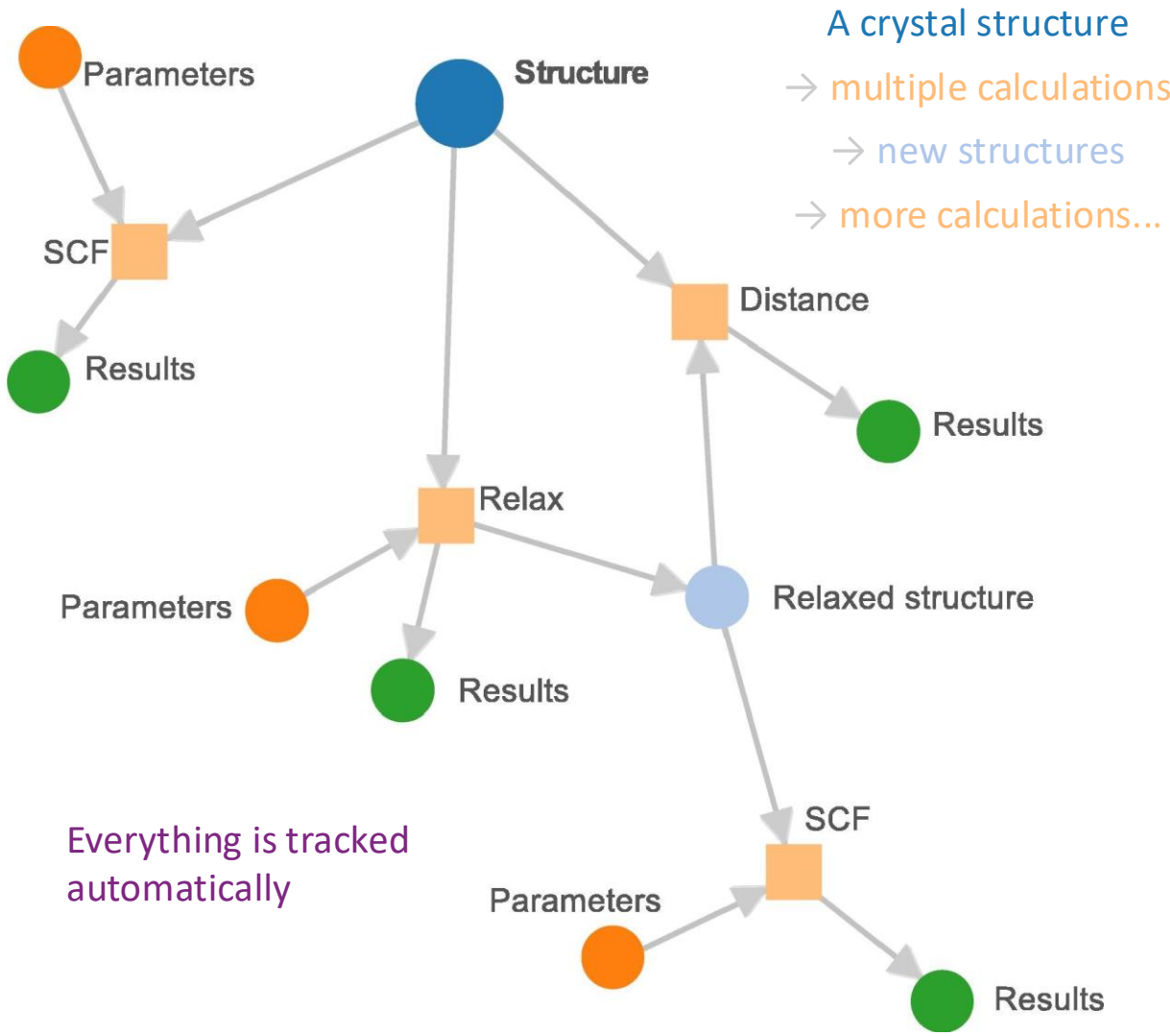
1 Full Provenance
Trace any result back to original inputs

2 Querying
Find all calculations that used a specific structure

3 Reproducibility
Re-run any calculation chain

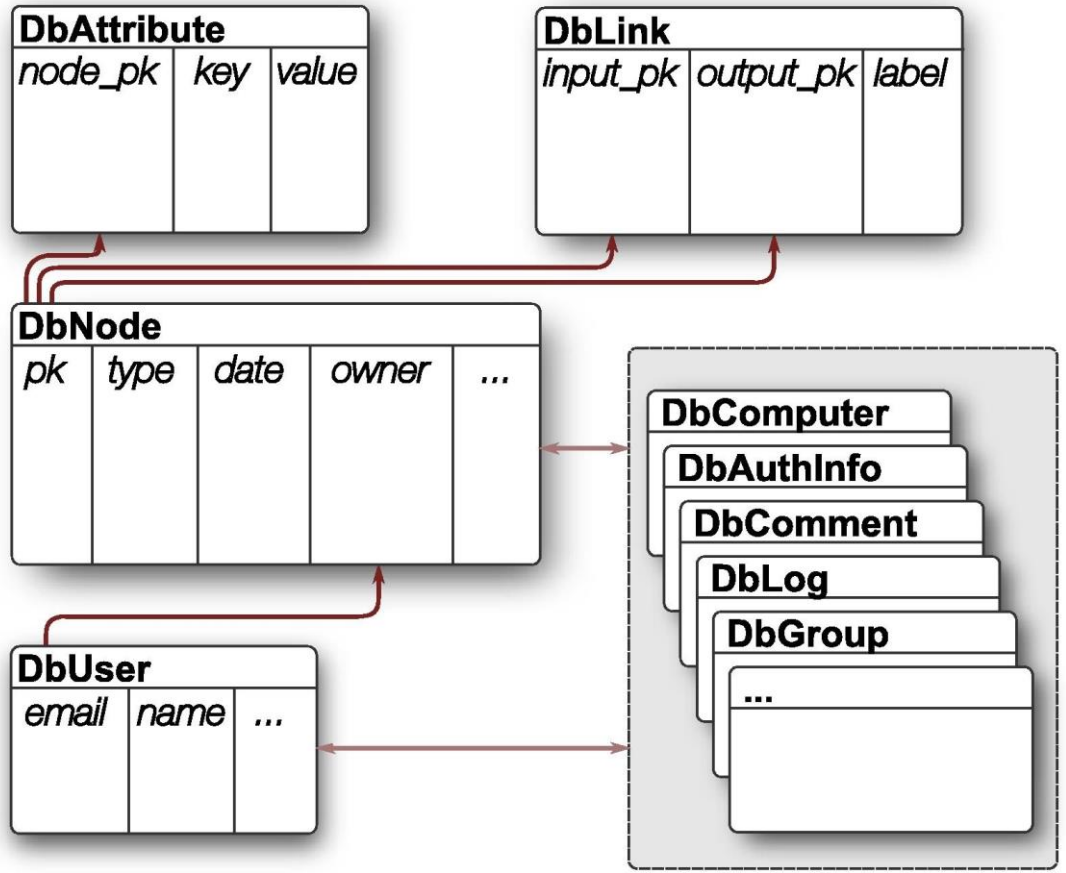
4 Validation
If input is wrong, find all affected results

Provenance Graph



AiiDA Database: Three Core SQL Tables

- **DbNode Table**
  - Stores **nodes** (Calculation / Data / Code)
  - Contains: **ID, type, creation time, owner**
- **DbLink Table**
  - Stores directed connections between nodes
  - Contains: **input node, output node, link label**
  - Defines **input–output relationships**
- **DbAttribute Table**
  - Stores all **node properties** as **key–value pairs**
  - Enables flexible and **queryable** metadata



DbAttribute table

node_pk	key	value
7	num_cpus	48
7	queue_name	"private"
7	submission_time	May 2nd, 2014 13:46:07
8	energy	-13.736229
8	energy_units	"eV"
8	forces	[[4.32, 3.22, 0.73]], [2.23, -1.46, 0.22], ...]
...		

pk=7 Calc

pk=8 Results

AiiDA Ecosystem

Resource	URL
Official Website	https://www.aiida.net/
Documentation	https://aiida.readthedocs.io/
Source Code	https://github.com/aiidateam/aiida-core/
Community Forum	https://aiida.discourse.group/
Plugin Registry	https://aiidateam.github.io/aiida-registry/

Key Numbers

1000+
citations across three main AiiDA papers

105
registered plugins

25+
tutorials, workshops, and trainings worldwide

Extensibility: The Plugin System

Everything is a plugin

Calculation Plugins

- Quantum ESPRESSO
- VASP
- GPAW
- Gaussian
- CP2K
- SIESTA
- FLEUR

Data Plugins

- Crystal structures
- Band structures
- Pseudopotentials

Transport Plugins

- SSH
- Local

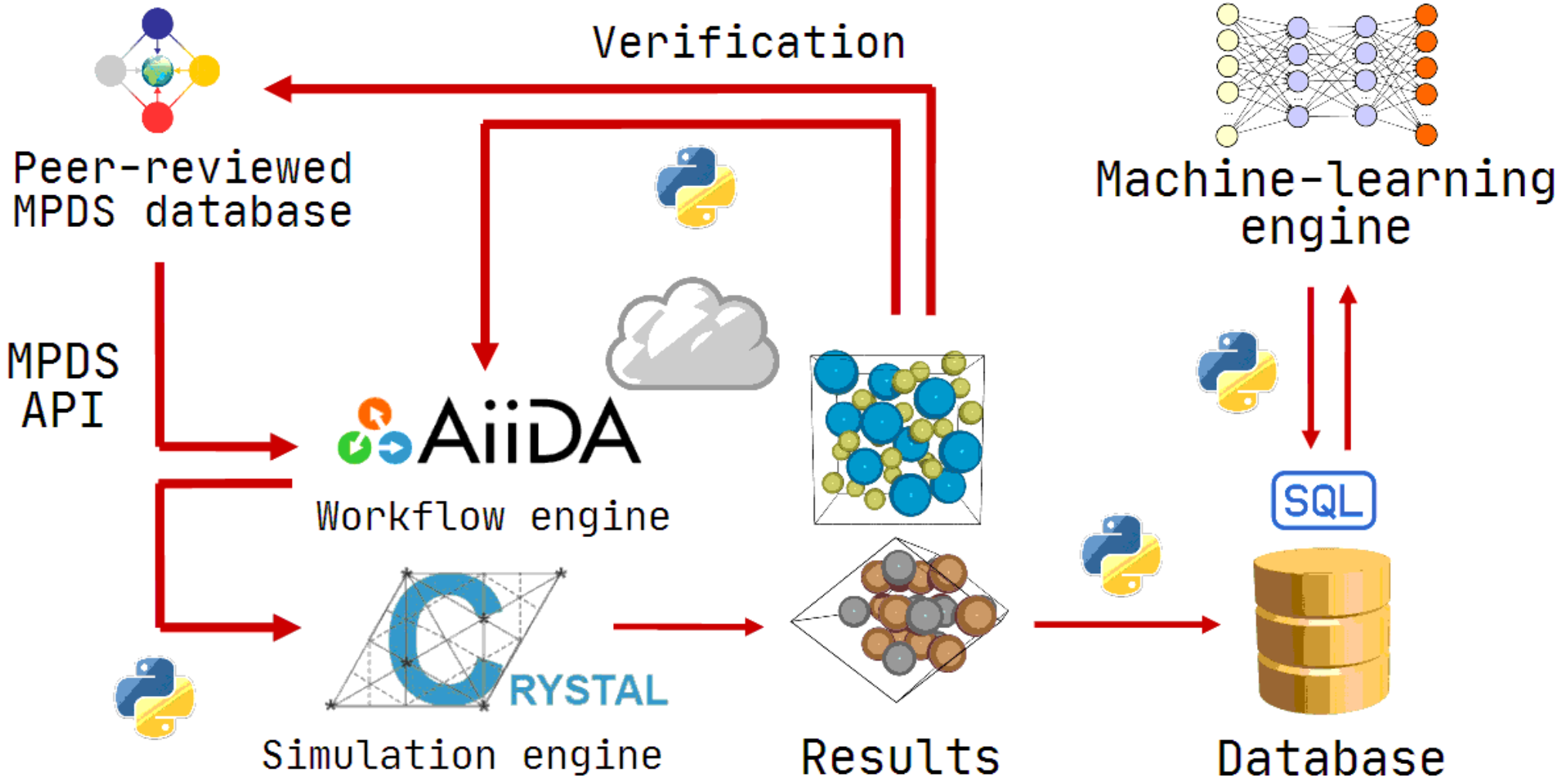
Scheduler Plugins

- SLURM
- PBS
- SGE
- Torque

Workflow Plugins

- PwRelaxWorkChain
- PhononWorkChain
- Wannier90 workflows

Application - MPDS-AIIDA



Conclusions

- Provenance tracking (DAG) makes reproducibility automatic, not manual
- Workflows, data, and computations are unified into one system
- The database + ORM design enables scalable, queryable scientific data
- A plugin architecture allows the system to evolve with new tools and codes

What can we learn from AiiDA?

- Design for **reproducibility from day one**
- **Abstract** complexity, expose simple interfaces
- Make **extensibility** a core feature (plugins)
- Build **community** and **ecosystem**: documentation, tutorials, forums, workshops

Thank You