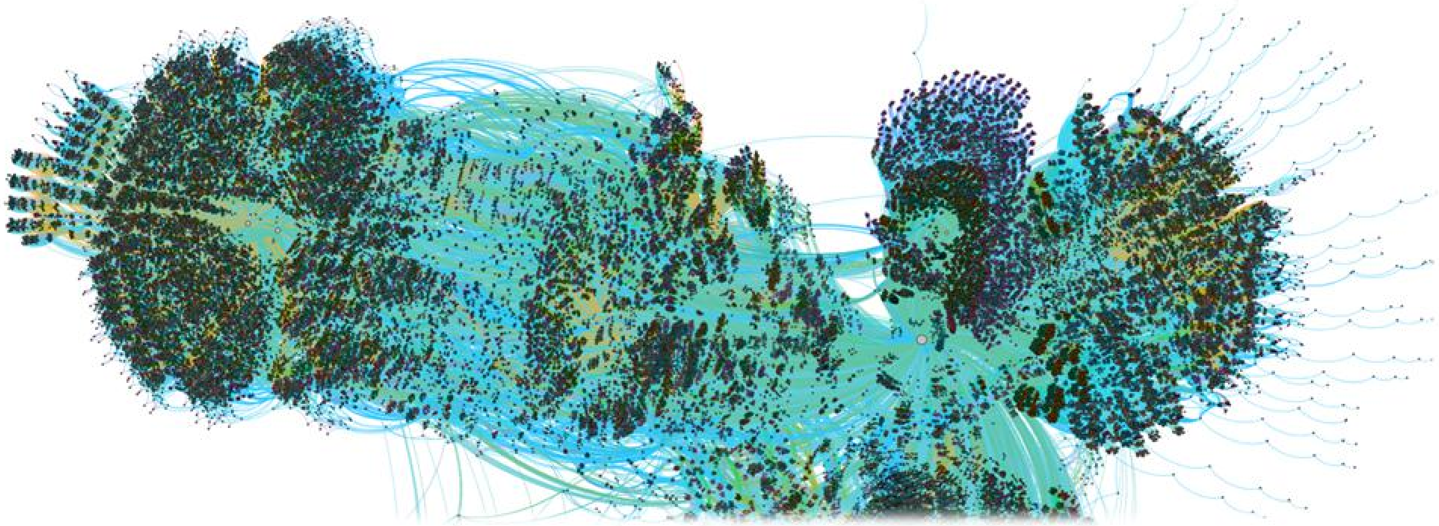


Aiida Provenance graph of a database (> 100K nodes).

SISC LAB PROJECT 7

ANALYSIS TOOL OF A MATERIALS DESIGN DATABASE

Supervisors: Prof. Dr. Stefan Blügel, Dr. Daniel Wortmann, Jens Bröder, Johannes Wasmer
Quantum Theory of Materials (PGI-1/IAS-1), Forschungszentrum Jülich

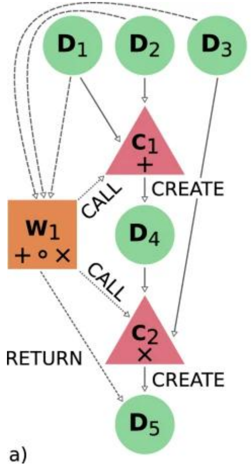
2 March 2021 | Group 10: Sijie Luo, Miao Wang, Zhipeng Tan

Outline

- Problem description & AiiDA introduction
- Deliverable 1 (D1): Statistical birds eye view of the contents in an AiiDA database
- Deliverable 2 (D2): Structure property visualizer
- Structure of the report
- Conclusion & outlook

Problem description

- AiiDA, an infrastructure developed for computational materials science.
- AiiDA can record any calculation or workflow in provenance graph and store them in database automatically, enabling **reconstruct the complete history of each calculation** or scientific result.:



- However, for larger database in real life…

Motivation

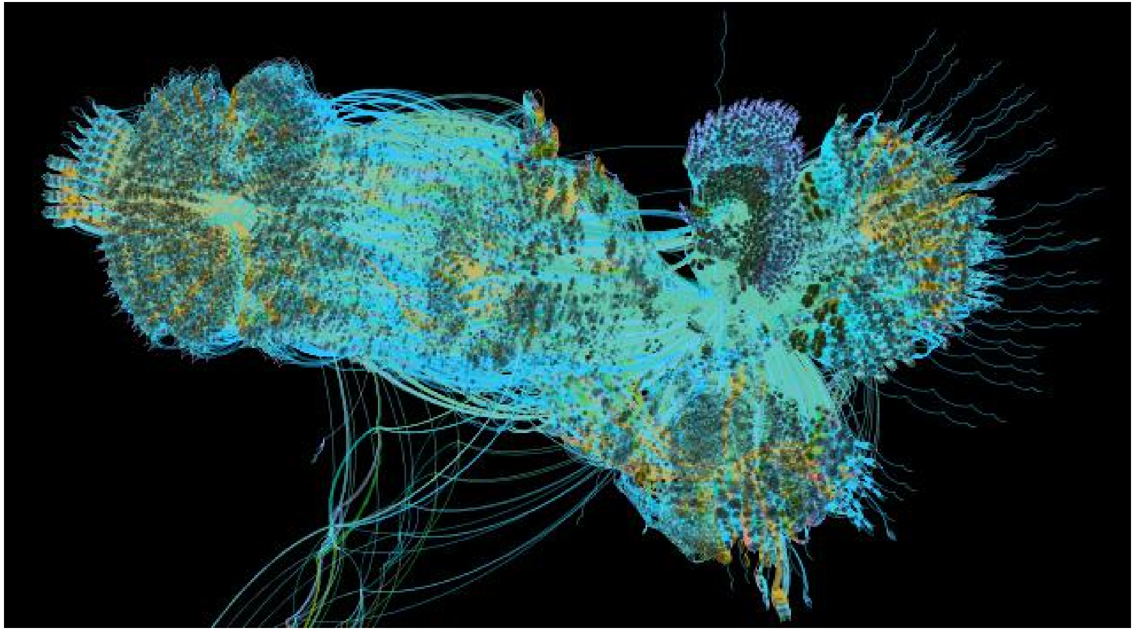


Figure 1: Aiida provenance graph, ~1000 FLEUR calculations with >100 K nodes.

Authors: Jens Bröder, Daniel Wortmann, Stefan Blügel. Using the AiiDA-FLEUR package for all-electron ab initio electronic structure data generation and processing in materials science, IAS Series 40, p 43-48 (2019).

- **When the database is large, a useful analyse and visualize tool for statistics becomes crucial. But AiiDA doesn't have those functions yet.**

Motivation and requirements

- **Aims** for the Analysis Tool: Given an AiiDA database, we can extract important statistical visual information from the database using the tool.
- **Requirements**:
  - **Interactive Plots:** To get extra information when hovering over the plot.
  - **Performance**: Running time of each subtask or total task.
    - **Serialization & Deserialization:** Read the DB only once and then store the desired data into output file, which avoids tedious retrieving for the next time.
- **Deliverables:**
  - **Python Helper packages for processing and visualization**
  - **2 Jupyter notebooks (D1 and D2), 1 python file (bokeh application)**
  - **json/excel output files containing data**
  - **Plots**

Team Introduction & Task Distribution

- **Miao Wang:** Deliverable 1, report, slides
- **Zhipeng Tan:** Deliverable 1, report, slides
- **Sjie Luo:** Deliverable 2, report, slides

- Cooperated over github: https://github.com/JuDFTteam/aiida-jutools/tree/SiscLab2020

Outline

- Problem description & AiiDA introduction
- **Deliverable 1 (D1): Statistical birds eye view of the contents in an AiiDA database**
- Deliverable 2 (D2): Structure property visualizer
- Structure of the report
- Conclusion & outlook

Deliverable1 Statistical birds eye view of the contents in an AiiDA database

MiaoWang, Zhipeng Tan

- **Querying in database**: When using Aiida to calculate, the database is usually very large, when we query the database, we need to use an efficient search tool--QueryBuilder. Through QueryBuilder we can find the specified content.

- **QueryBuilder**: A python interface to query the database in a hybrid relational / graph - query fashion.

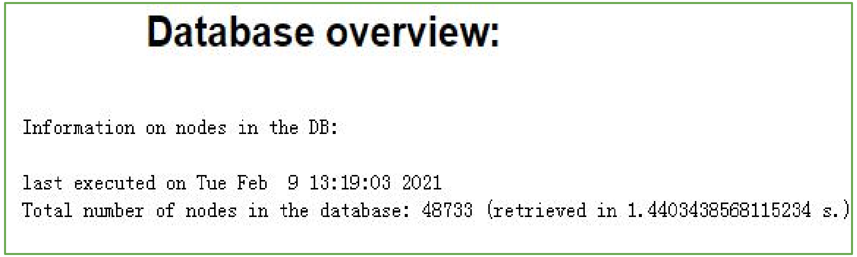
D1 Statistical birds eye view of the contents in an AiiDA database

- **Database overview**: Get last executed time and total number of nodes in the database.
- **User Information**: Print the list of Users in database and how many nodes belong to them.
- **Nodes types distribution**: Show data nodes,process nodes with lowest classes name and dict nodes with incoming link label in pie plots(interactive).
- **Database time evolution**: Total and each user ctime & mtime of all nodes over time in line plots (interactive).
- **Codes**: List Code name and sorted by calcjobs.

D1 Task a: Database overview

Query for all nodes in database

- **Nodes**: In AiiDA, the Node class is the base class to represent any node in the graph.
- **Include**:
  - The user who created it.
  - The creation and last modification times.
  - An optional computer on which it was run or stored.
  - A human-readable label and description.



Database overview:

```
Information on nodes in the DB:

last executed on Tue Feb  9 13:19:03 2021
Total number of nodes in the database: 48733 (retrieved in 1.4403438568115234 s.)
```

Figure3: Database overview

D1 Statistical birds eye view of the contents in an AiiDA database

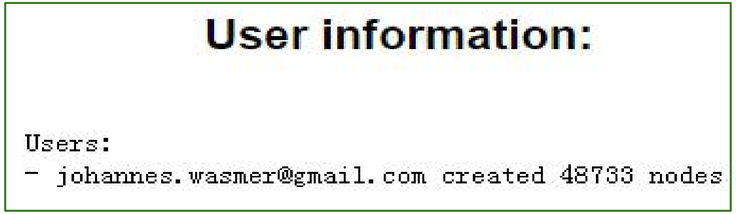
- **Database overview**: Get last executed time and total number of nodes in the database.
- **User Information**: Print the list of Users in database and how many nodes belong to them.
- **Nodes types distribution**: Show data nodes,process nodes with lowest classes name and dict nodes with incoming link label in pie plots(interactive).
- **Database time evolution**: Total and each user ctime & mtime of all nodes over time in line plots (interactive).
- **Codes**: List Code name and sorted by calcjobs.

D1 Task b: User Information

Details about user information

- **User**: The node was created by a user.
- Show a list of all users in database.

```
for count, email in sorted((v, k) for k, v in users.items())[::-1]:
        print("* {} created {} nodes".format(email, count))
```



User information:

```
Users:
- johannes.wasmer@gmail.com created 48733 nodes
```

Figure 4: User Information

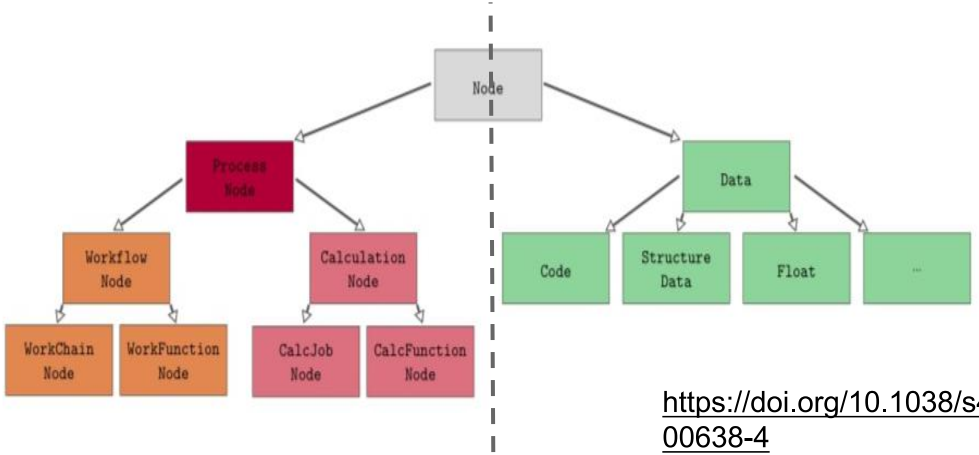
D1 Statistical birds eye view of the contents in an AiiDA database

- **Database overview**: Get last executed time and total number of nodes in the database.
- **User Information**: Print the list of Users in database and how many nodes belong to them.
- **Nodes types distribution**: Show data nodes,process nodes with lowest classes name and dict nodes with incoming link label in pie plots(interactive).
- **Database time evolution**: Total and each user ctime & mtime of all nodes over time in line plots (interactive).
- **Codes**: List Code name and sorted by calcjobs.

D1 Task c: Node types distribution

Get node types

- The data in the AiiDA database is stored as a graph of connected entities.
- **Nodes**: As vertices of a directed graph.
- **Links**: Graph edges connect the nodes.



https://doi.org/10.1038/s41597-020-00638-4

Figure5: The hierarchy of the node types in AiiDA

D1 Task c: Node types distribution

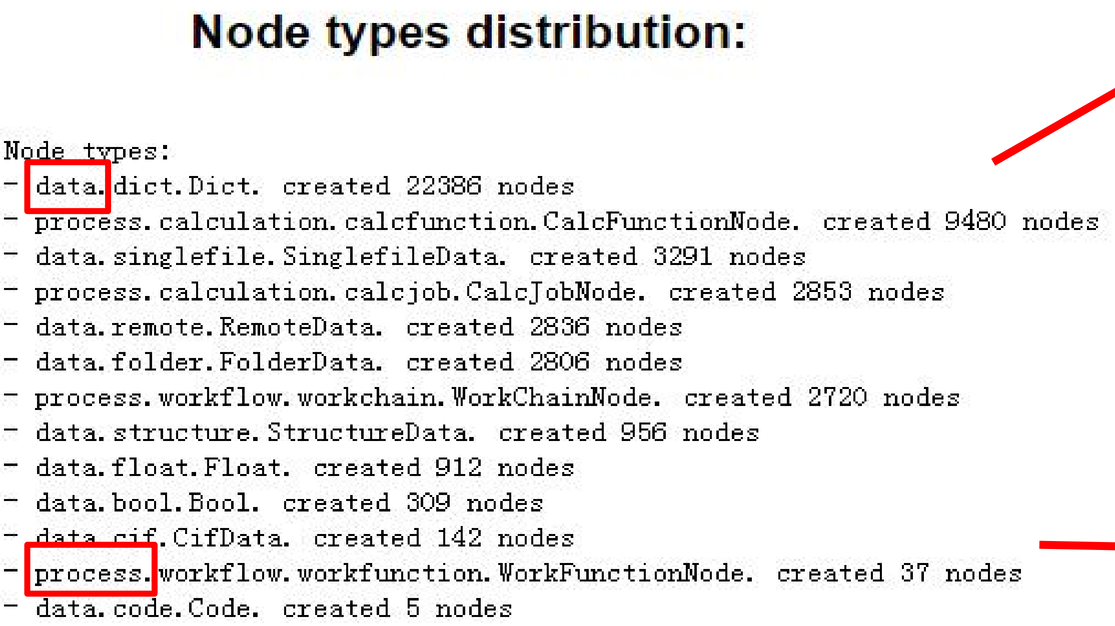
Split data nodes and process nodes

- **Data Nodes:** Int, Float, Dict, ArrayData, StructureData, FolderData, …(Here, 'structure' means 'crystal structure data’).

- **Process Nodes:** ProcessNode serves as a mere record in the database of what actually happened during execution.
  - **Calculation**: Create data,orchestrate other processes,return data produced by calculations.
  - **Workfunction**: Work function calling other process through Python.(**easier**)
  - **Workchain**: Work chains are used to implement **complex** workflows calling many long-running calculation jobs.

D1 Task c: Node types distribution

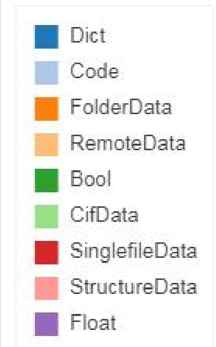
Aim: Split data nodes and process nodes



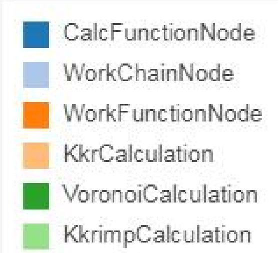
Node types distribution:

```
Node types:
- data.dict.Dict. created 22386 nodes
- process.calculation.calcfunction.CalcFunctionNode. created 9480 nodes
- data.singlefile.SinglefileData. created 3291 nodes
- process.calculation.calcjob.CalcJobNode. created 2853 nodes
- data.remote.RemoteData. created 2836 nodes
- data.folder.FolderData. created 2806 nodes
- process.workflow.workchain.WorkChainNode. created 2720 nodes
- data.structure.StructureData. created 956 nodes
- data.float.Float. created 912 nodes
- data.bool.Bool. created 309 nodes
- data.cif.CifData. created 142 nodes
- process.workflow.workfunction.WorkFunctionNode. created 37 nodes
- data.code.Code. created 5 nodes
```

Figure 6:Node types



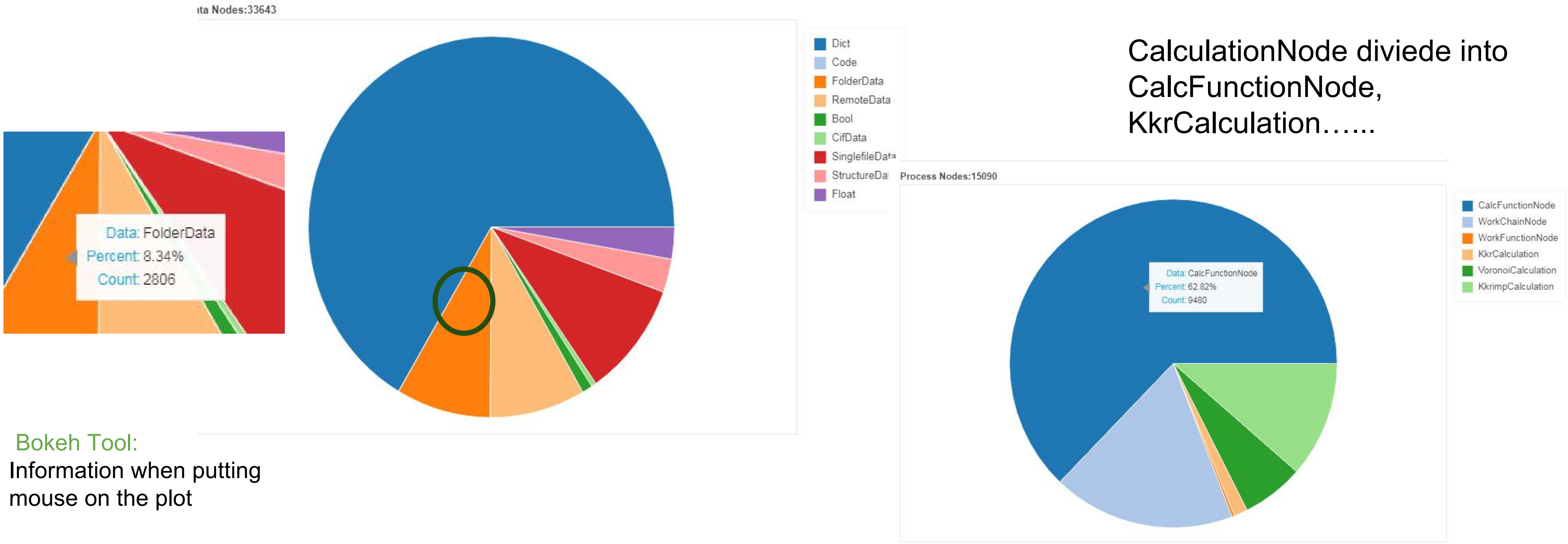
🟦 Dict
🟦 Code
🟧 FolderData
🟧 RemoteData
🟩 Bool
🟩 CifData
🟥 SinglefileData
🟥 StructureData
🟪 Float



🟦 CalcFunctionNode
🟦 WorkChainNode
🟧 WorkFunctionNode
🟧 KkrCalculation
🟩 VoronoiCalculation
🟩 KkrimpCalculation

D1 Task c: Node types distribution

Plots nodes information in pie charts(interactive)



CalculationNode diviede into CalcFunctionNode, KkrCalculation……

Bokeh Tool:

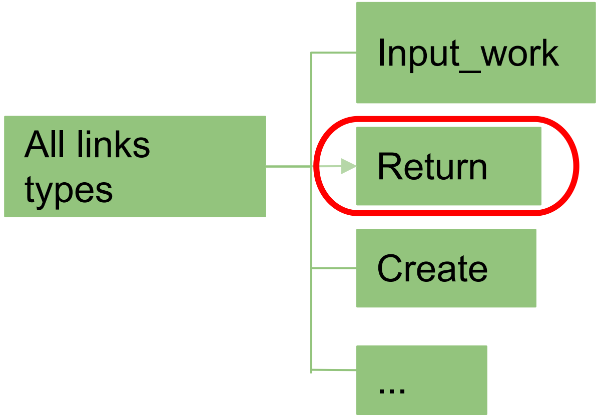
Information when putting mouse on the plot

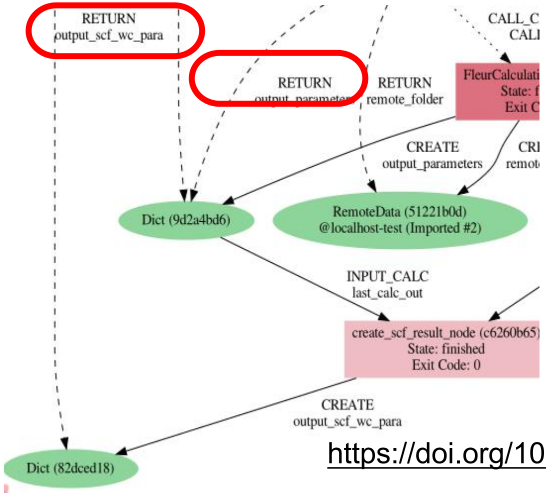
Figure 7: Node types in pie plots

D1 Task c: Node types distribution

Dict nodes with incoming link label

- Links have a label that can be used, given a node, to distinguish nodes connected to it with the same link type.
- **Dicts nodes:**a dictionary of key-value pairs ,the dicts with ‘return link types’ have a label in the incoming link.



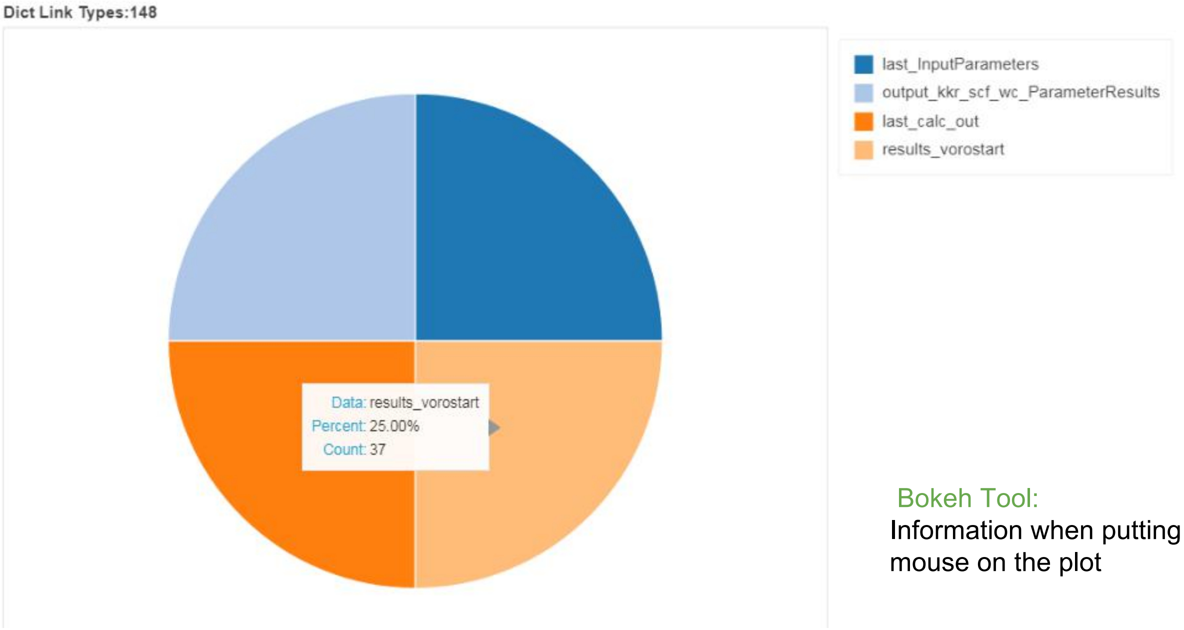


https://doi.org/10.1038/s41597-020-00638-4

Figure 8: Return link

D1 Task c: Node types distribution

Plots nodes information in pie charts(interactive)



Bokeh Tool:

Information when putting mouse on the plot

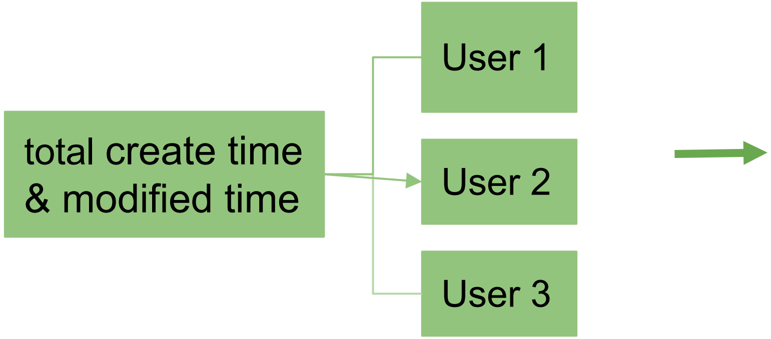
Figure 9: Dict link types in pie plot

D1 Statistical birds eye view of the contents in an AiiDA database

- **Database overview**: Get last executed time and total number of nodes in the database.
- **User Information**: Print the list of Users in database and how many nodes belong to them.
- **Nodes types distribution** : Show data nodes,process nodes with lowest classes name and dict nodes with incoming link label in pie plots(interactive).
- **Database time evolution**: Total and each user ctime & mtime of all nodes over time in line plots (interactive).
- **Codes**: List Code name and sorted by calcjobs.

D1 Task d: Database time evolution

Total and each user ctime & mtime in line plots(interactive)


total create time & modified time

User 1

User 2

User 3

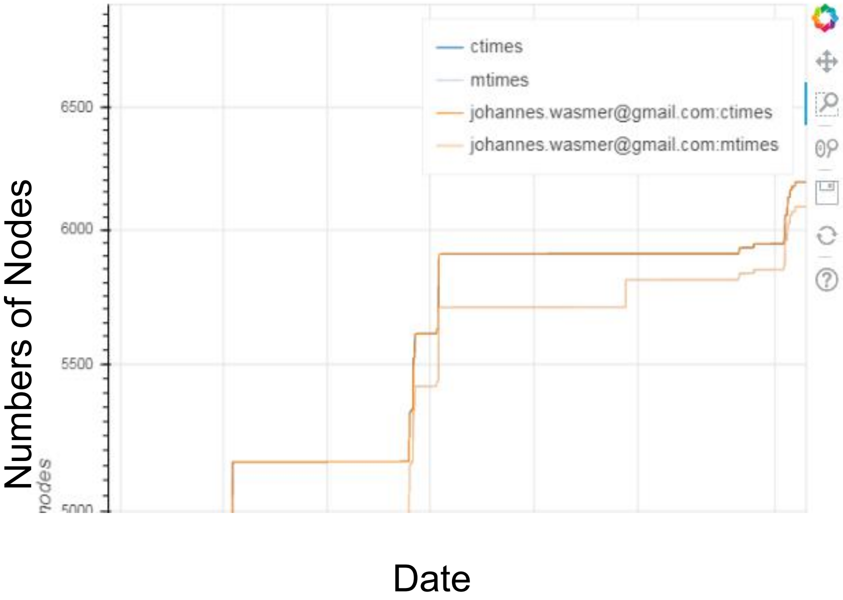


Figure 10: Ctime & mtime in line plot

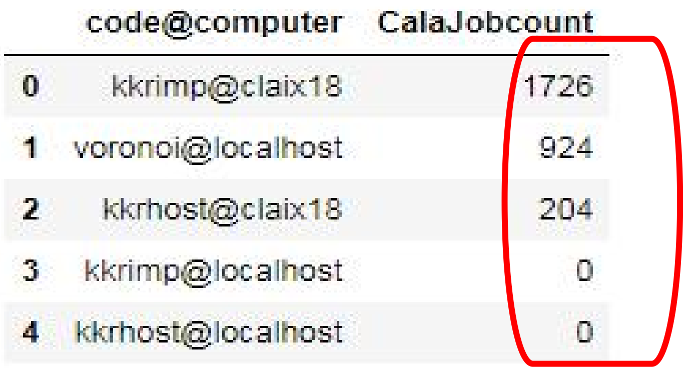
D1 Statistical birds eye view of the contents in an AiiDA database

- **Database overview**: Get last executed time and total number of nodes in the database.
- **User Information**: Print the list of Users in database and how many nodes belong to them.
- **Nodes types distribution** : Show data nodes,process nodes with lowest classes name and dict nodes with incoming link label in pie plots(interactive).
- **Database time evolution**: Total and each user ctime & mtime of all nodes over time in line plots (interactive).
- **Codes**: List Code name and sorted by calcjobs.

D1 Task e: Codes

Aim: List Code name and sorted by calcjobs

- **Calculations jobs:** As Code nodes formed by some calculations, we want to retrieve the calculations between them, commonly run via a job scheduler and optionally on a remote machine.



	code@computer	CalaJobcount
0	kkrimp@claix18	1726
1	voronoi@localhost	924
2	kkrhost@claix18	204
3	kkrimp@localhost	0
4	kkrhost@localhost	0

Figure 11: Codes

D1 Database Overview and Data provenance Health indicator

Tasks:

- **Groups Analysis**
- **Structure Analysis**
- **Process Analysis**
- **Provenance Analysis**

D1 Database Overview and Data provenance Health indicator

- **Other requirements:**
  - **Interactive Plots:** Plots should be interactive, so that we can check extra informations when we put mouse on them.
  - **Performance**: Running time of each subtask or total task.
    - **Serialization & Deserialization:** Read the DB only once and then store the desired data into output file. Then next time when we want to read the data again, we don’t need to process the data again.

D1 Task requirements: Performance

- **Serialization: We preprocess the data, and then serialize the data to a file. Then all the other parts will be quick.**

```
try:
    filepath = './output/Struct_Element.json'
    x = Serializer.deserialize_from_file(filepath,Node_type = 'StructureElement')

except:
    qb = QueryBuilder()
    qb.append(StructureData)
    StructData = qb.all()
    serializer = Serializer.Serializer(StructData)
    filepath = './output/Struct_Element.json'
    serializer.to_file(filepath,'StructureElement')
    x = Serializer.deserialize_from_file(filepath,Node_type ='StructureElement')
    ShowElements(x)
```

Different **preprocessing** methods by specifying **Node_type**

D1 Task requirements: Performance

- **Serialization: Running time comparison.**
- DB infomation: Size: 431 MB. Nodes: 48950. Process nodes: 15166. Data nodes: 33784
- Database description: 800 Impurity (defect atoms) embeddings into different elemental host crystals with aiida-kkr.

	Groups analysis	**Structures analysis**	**Processes analysis**	**Provenance analysis**
Running time (unserialized)	0.6411(s)	9.2704(s)	106.0234(s)	1002.4026(s)
Running time (serialized)	0.1500(s)	0.6174(s)	1.4370(s)	4.2827(s)
Speed-Up	4.3	15.1	73.8	234.1

D1 Task requirements: Interactive Plot

- **Interactive Plot: We use bokeh hover tools**

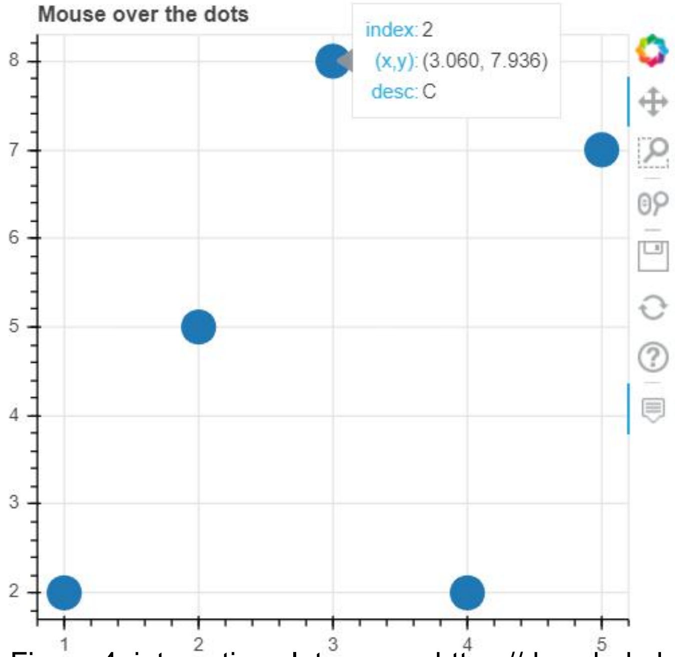


Figure 4: interactive plot. source:https://docs.bokeh.org/

Advantages:

1. Allow zoom in
2. Allow more information when clicking

General Overview of the Database

- **Show the pictures from Johannes**
- **Based on this database, our analysis tool shows the following results.**

D1 Database Overview and Data provenance Health indicator

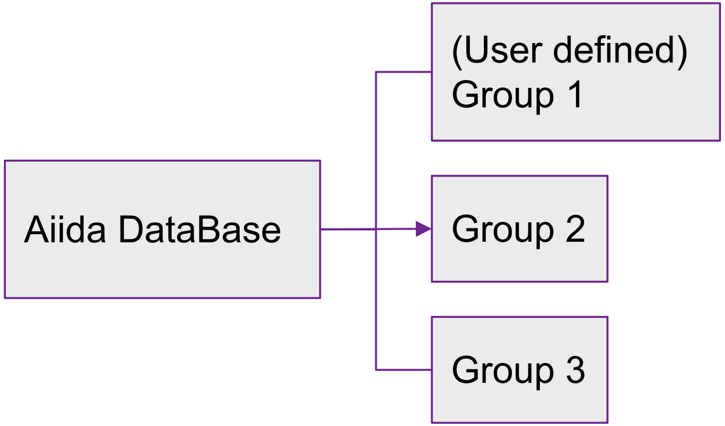
Tasks:

- **Groups Analysis:** Analyse all group names with how many nodes they contain, exclude certain nodes we don’t want to count.
- **Structure Analysis**: Further analyze what structures are in the DB, formulas and compositions.
- **Process Analysis**: Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message and exit code.
- **Provenance Analysis**: Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

D1 Task f: Groups

Analyse all group names with how many nodes they contain

- **Aim: We want to count the number of nodes of each group, and also filter any of them.**



List:

Name 1：number of Nodes

Name 2：number of Nodes

Name 3：number of Nodes

Filter:

Name 1：number of Nodes

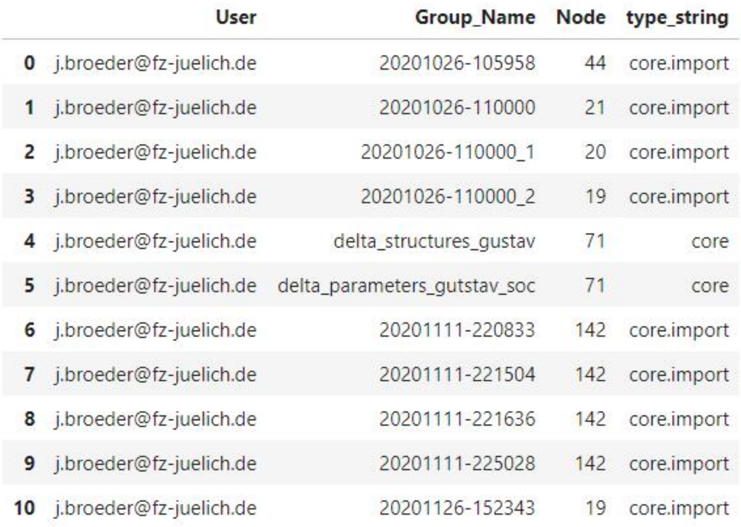
Name 2：number of Nodes

D1 Task f: Groups

Analyse all group names with how many nodes they contain

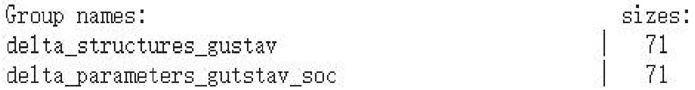
- **Result: Count the Nodes number of each group, and filter**

All Groups



	User	Group_Name	Node	type_string
0	j.broeder@fz-juelich.de	20201026-105958	44	core.import
1	j.broeder@fz-juelich.de	20201026-110000	21	core.import
2	j.broeder@fz-juelich.de	20201026-110000_1	20	core.import
3	j.broeder@fz-juelich.de	20201026-110000_2	19	core.import
4	j.broeder@fz-juelich.de	delta_structures_gustav	71	core
5	j.broeder@fz-juelich.de	delta_parameters_gutstav_soc	71	core
6	j.broeder@fz-juelich.de	20201111-220833	142	core.import
7	j.broeder@fz-juelich.de	20201111-221504	142	core.import
8	j.broeder@fz-juelich.de	20201111-221636	142	core.import
9	j.broeder@fz-juelich.de	20201111-225028	142	core.import
10	j.broeder@fz-juelich.de	20201126-152343	19	core.import

After filtering



```
Group names:                                    sizes:
delta_structures_gustav                       |   71
delta_parameters_gutstav_soc                  |   71
```

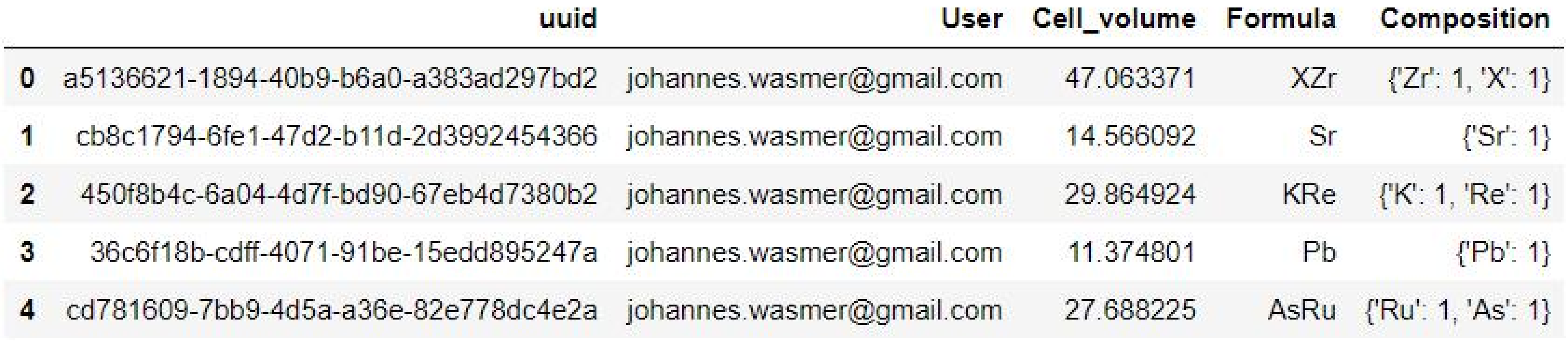
D1 Database Overview and Data provenance Health indicator

Tasks:

- **Groups Analysis:** Analyse all group names with how many nodes they contain, exclude certain nodes we don’t want to count.
- **Structure Analysis**: Further analyze what structures are in the DB, formulas and compositions.
- **Process Analysis**: Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message and exit code.
- **Provenance Analysis**: Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

D1 Task g: Structures

- **StructureData: Special Data containing crystal structure information.**

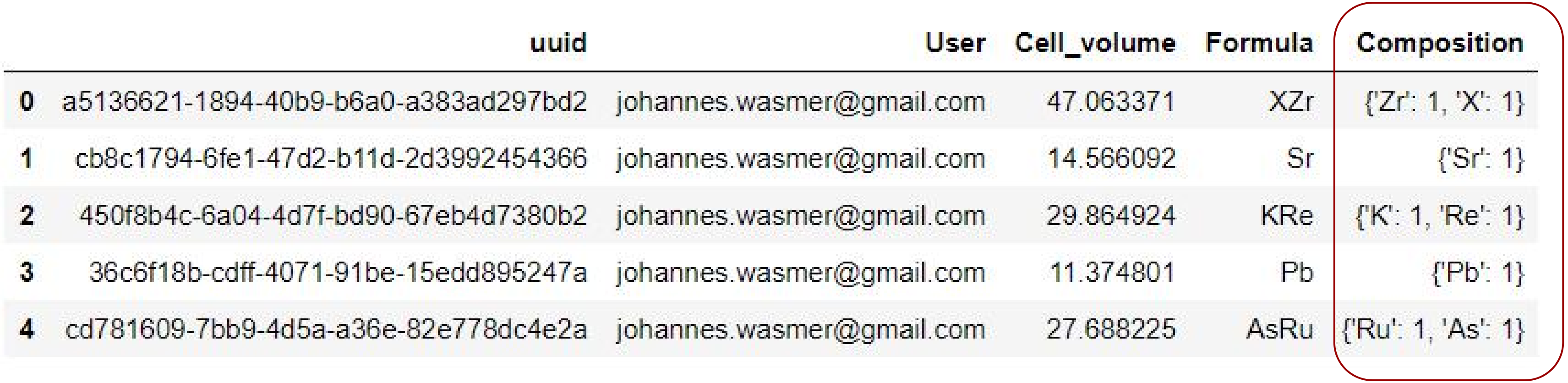


	uuid	User	Cell_volume	Formula	Composition
0	a5136621-1894-40b9-b6a0-a383ad297bd2	johannes.wasmer@gmail.com	47.063371	XZr	{'Zr': 1, 'X': 1}
1	cb8c1794-6fe1-47d2-b11d-2d3992454366	johannes.wasmer@gmail.com	14.566092	Sr	{'Sr': 1}
2	450f8b4c-6a04-4d7f-bd90-67eb4d7380b2	johannes.wasmer@gmail.com	29.864924	KRe	{'K': 1, 'Re': 1}
3	36c6f18b-cdff-4071-91be-15edd895247a	johannes.wasmer@gmail.com	11.374801	Pb	{'Pb': 1}
4	cd781609-7bb9-4d5a-a36e-82e778dc4e2a	johannes.wasmer@gmail.com	27.688225	AsRu	{'Ru': 1, 'As': 1}

D1 Task g: Structures

Further analyze what structures are in the DB, formulas and compositions.

- **Subtask 1: Analyse the number of atoms for different Nodes.**
- **Subtask 2: Count number of different elements of the DB.**



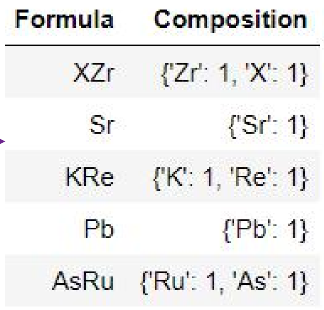
	uuid	User	Cell_volume	Formula	Composition
0	a5136621-1894-40b9-b6a0-a383ad297bd2	johannes.wasmer@gmail.com	47.063371	XZr	{'Zr': 1, 'X': 1}
1	cb8c1794-6fe1-47d2-b11d-2d3992454366	johannes.wasmer@gmail.com	14.566092	Sr	{'Sr': 1}
2	450f8b4c-6a04-4d7f-bd90-67eb4d7380b2	johannes.wasmer@gmail.com	29.864924	KRe	{'K': 1, 'Re': 1}
3	36c6f18b-cdff-4071-91be-15edd895247a	johannes.wasmer@gmail.com	11.374801	Pb	{'Pb': 1}
4	cd781609-7bb9-4d5a-a36e-82e778dc4e2a	johannes.wasmer@gmail.com	27.688225	AsRu	{'Ru': 1, 'As': 1}

D1 Task g: Structures

Further analyze what structures are in the DB, formulas and compositions.

- **Aim 1: Analyse the number of atoms for different Nodes.**

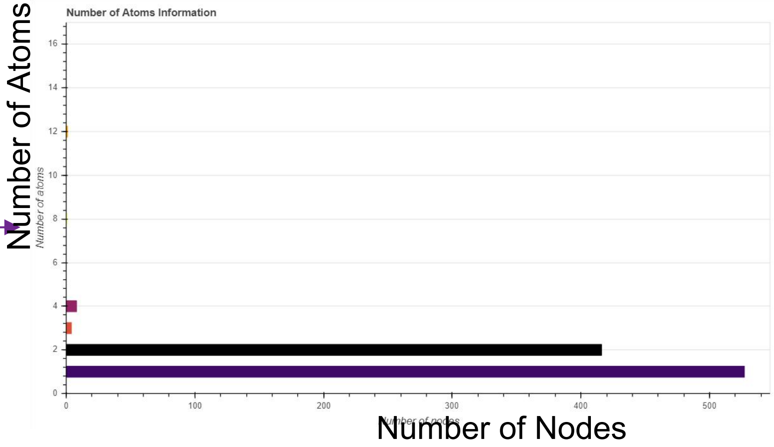
Aiida DataBase



Formula	Composition
XZr	{'Zr': 1, 'X': 1}
Sr	{'Sr': 1}
KRe	{'K': 1, 'Re': 1}
Pb	{'Pb': 1}
AsRu	{'Ru': 1, 'As': 1}

Number of atoms

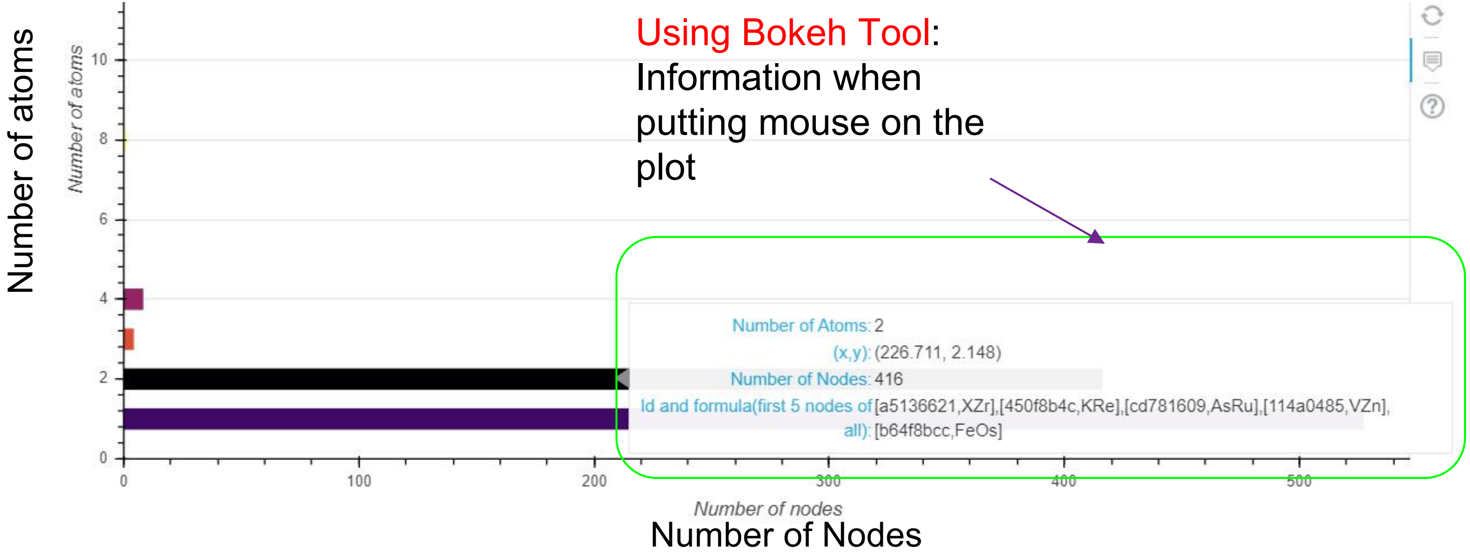
2
1
2
1
2



D1 Task g: Structures

Further analyze what structures are in the DB, formulas and compositions.

- **Result 1: Analyse the number of atoms for different Nodes.**



Using Bokeh Tool: Information when putting mouse on the plot

Number of Atoms: 2

(x,y): (226.711, 2.148)

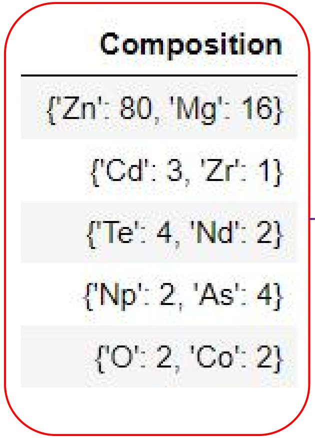
Number of Nodes: 416

Id and formula(first 5 nodes of all): [a5136621,XZr],[450f8b4c,KRe],[cd781609,AsRu],[114a0485,VZn],[b64f8bcc,FeOs]

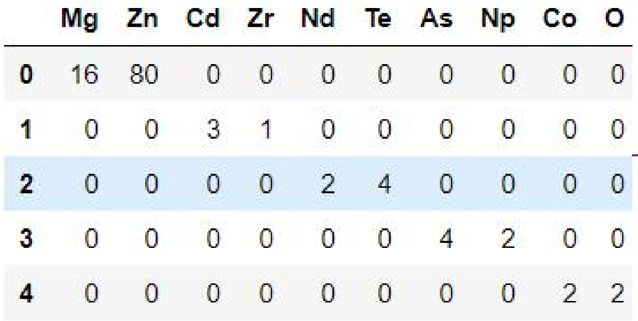
D1 Task g: Structures

Further analyze what structures are in the DB, formulas and compositions.

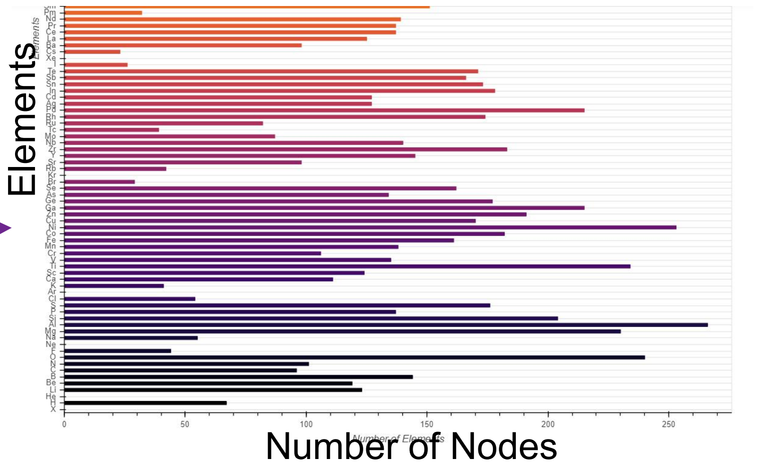
- **Aim 2: Count number of different elements of the DB.**



Composition
{'Zn': 80, 'Mg': 16}
{'Cd': 3, 'Zr': 1}
{'Te': 4, 'Nd': 2}
{'Np': 2, 'As': 4}
{'O': 2, 'Co': 2}



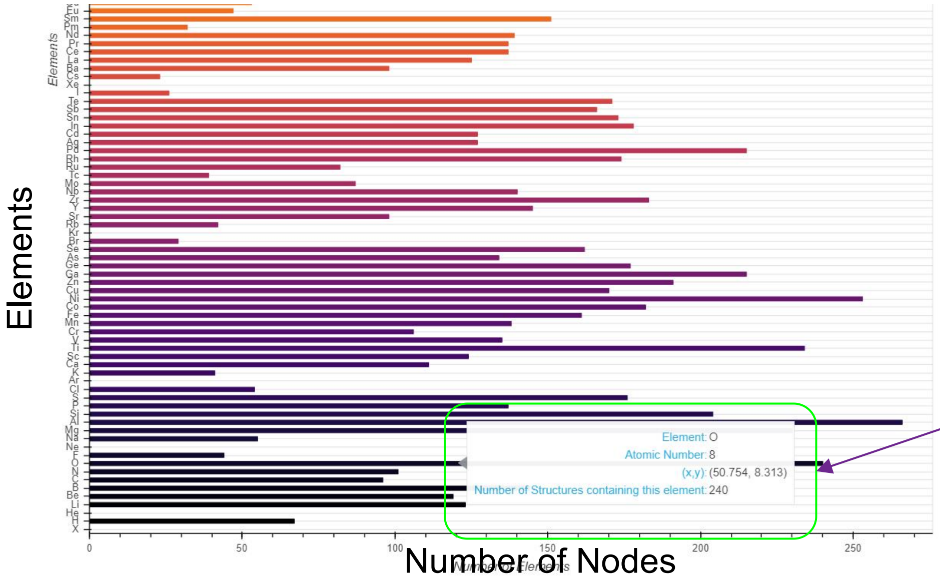
	Mg	Zn	Cd	Zr	Nd	Te	As	Np	Co	O
0	16	80	0	0	0	0	0	0	0	0
1	0	0	3	1	0	0	0	0	0	0
2	0	0	0	0	2	4	0	0	0	0
3	0	0	0	0	0	0	4	2	0	0
4	0	0	0	0	0	0	0	0	2	2



D1 Task g: Structures

Further analyze what structures are in the DB, formulas and compositions.

- **Result 2: Count number of different elements of the DB.**



Element: O
Atomic Number: 8
(x,y): (50.754, 8.313)
Number of Structures containing this element: 240

Using Bokeh Tool: Information when putting mouse on the plot

D1 Database Overview and Data provenance Health indicator

Tasks:

- **Groups Analysis:** Analyse all group names with how many nodes they contain, exclude certain nodes we don't want to count.
- **Structure Analysis**: Further analyze what structures are in the DB, formulas and compositions.
- **Process Analysis**: Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message and exit code.
- **Provenance Analysis**: Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

- **ProcessNode: The Process class contains all the information and logic to tell, whoever is handling it, how to run it to completion.**

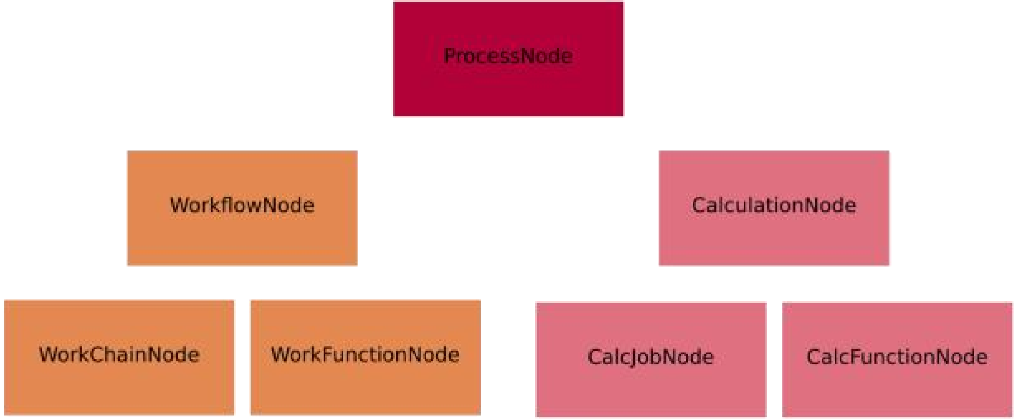
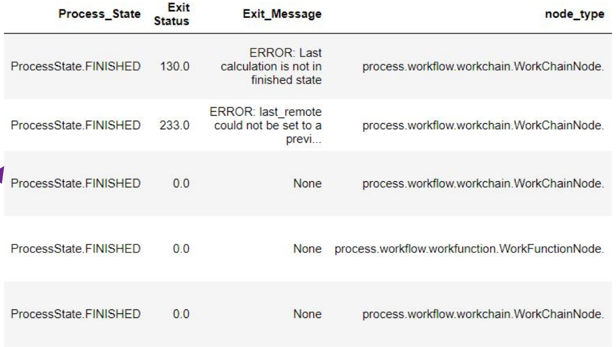


Figure: The hierarchy of the process nodes.
source:https://aiida.readthedocs.io/

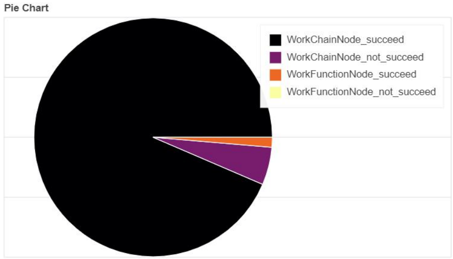
D1 Task h: Process

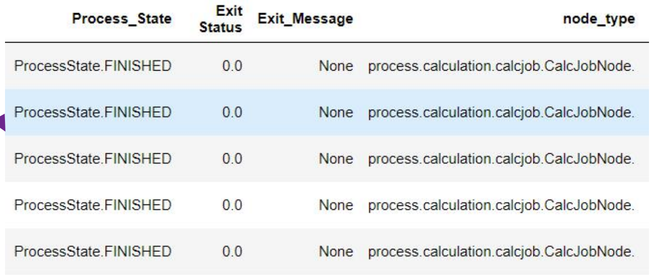
Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

Aiida DataBase

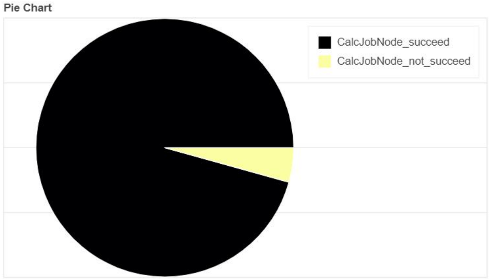


Process_State	Exit Status	Exit_Message	node_type
ProcessState.FINISHED	130.0	ERROR: Last calculation is not in finished state	process.workflow.workchain.WorkChainNode.
ProcessState.FINISHED	233.0	ERROR: last_remote could not be set to a previ...	process.workflow.workchain.WorkChainNode.
ProcessState.FINISHED	0.0	None	process.workflow.workchain.WorkChainNode.
ProcessState.FINISHED	0.0	None	process.workflow.workfunction.WorkFunctionNode.
ProcessState.FINISHED	0.0	None	process.workflow.workchain.WorkChainNode.



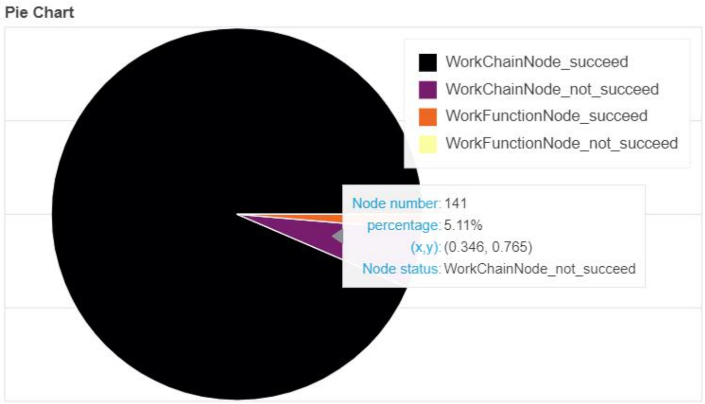


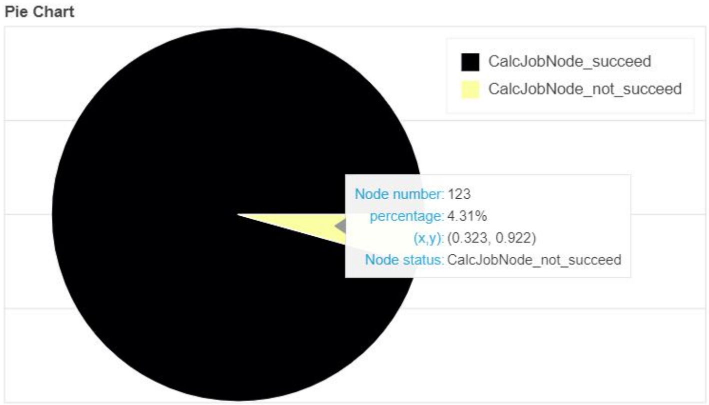
Process_State	Exit Status	Exit_Message	node_type
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.



D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.





D1 Database Overview and Data provenance Health indicator

Tasks:

- **Groups Analysis:** Analyse all group names with how many nodes they contain, exclude certain nodes we don’t want to count.
- **Structure Analysis**: Further analyze what structures are in the DB, formulas and compositions.
- **Process Analysis**: Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message and exit code.
- **Provenance Analysis**: Analyse the health of DB. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

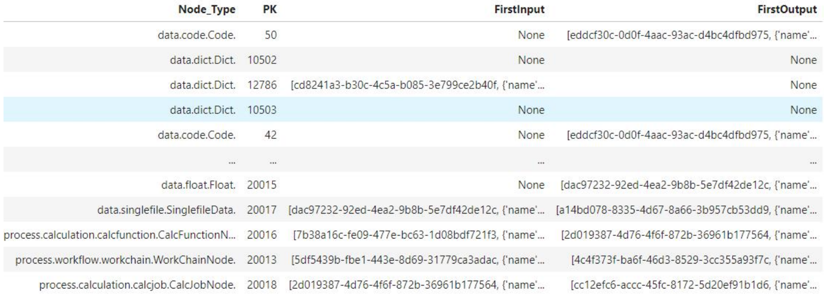
D1 Task i: Provenance

Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

- **Provenance:** AiiDA automatically stores entities in its database and links them forming a **directed graph**. This directed graph automatically tracks the **provenance** of all data produced by calculations or returned by workflows.
- The most important part of workflow is connection. So if nodes don't have connection to other nodes, they were either not used or we have problem with recording.

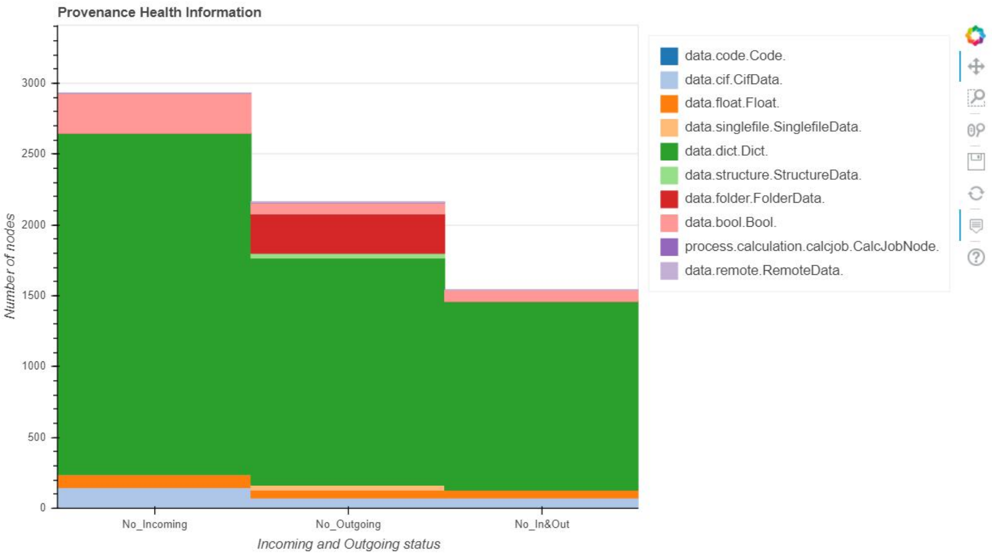
D1 Task i: Provenance

Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.



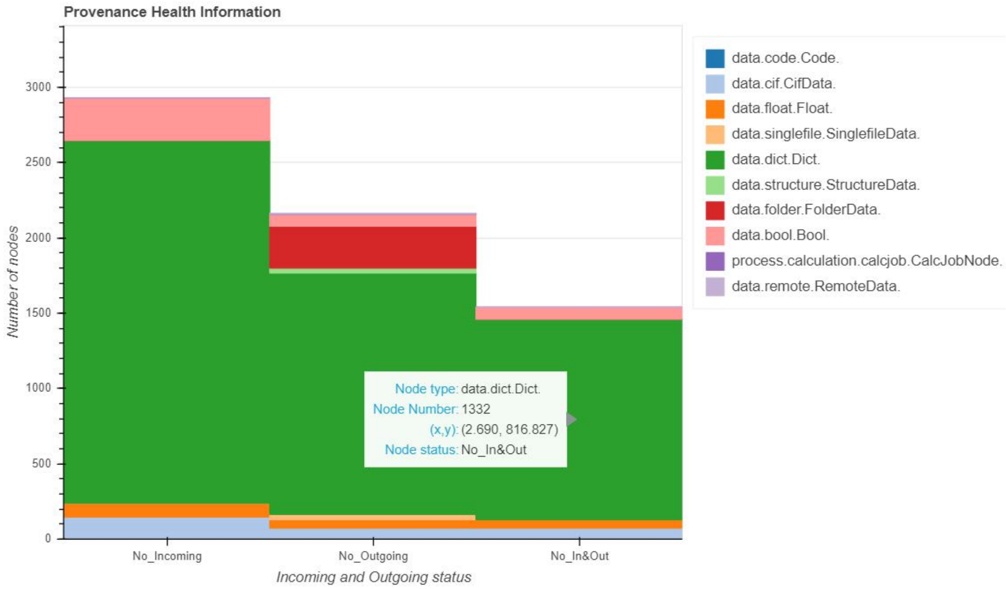
Node_Type	PK	FirstInput	FirstOutput
data.code.Code.	50	None	[eddcf30c-0d0f-4aac-93ac-d4bc4dfbd975, {'name'...
data.dict.Dict.	10502	None	None
data.dict.Dict.	12786	[cd8241a3-b30c-4c5a-b085-3e799ce2b40f, {'name'...	None
data.dict.Dict.	10503	None	None
data.code.Code.	42	None	[eddcf30c-0d0f-4aac-93ac-d4bc4dfbd975, {'name'...
...	...	...	...
data.float.Float.	20015	None	[dac97232-92ed-4ea2-9b8b-5e7df42de12c, {'name'...
data.singlefile.SinglefileData.	20017	[dac97232-92ed-4ea2-9b8b-5e7df42de12c, {'name'...	[a14bd078-8335-4d67-8a66-3b957cb53dd9, {'name'...
process.calculation.calcfunction.CalcFunctionN...	20016	[7b38a16c-fe09-477e-bc63-1d08bdf721f3, {'name'...	[2d019387-4d76-4f6f-872b-36961b177564, {'name'...
process.workflow.workchain.WorkChainNode.	20013	[5df5439b-fbe1-443e-8d69-31779ca3adac, {'name'...	[4c4f373f-ba6f-46d3-8529-3cc355a93f7c, {'name'...
process.calculation.calcjob.CalcJobNode.	20018	[2d019387-4d76-4f6f-872b-36961b177564, {'name'...	[cc12efc6-accc-45fc-8172-5d20ef91b1d6, {'name'...

Aiida DataBase



D1 Task i: Provenance

Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.



Node type: data.dict.Dict.
Node Number: 1332
(x,y): (2.690, 816.827)
Node status: No_In&Out

Outline

- Problem description & AiiDA introduction
- Deliverable 1 (D1): Statistical birds eye view of the contents in an AiiDA database
- **Deliverable 2 (D2): Structure property visualizer**
- Structure of the report
- Conclusion & outlook

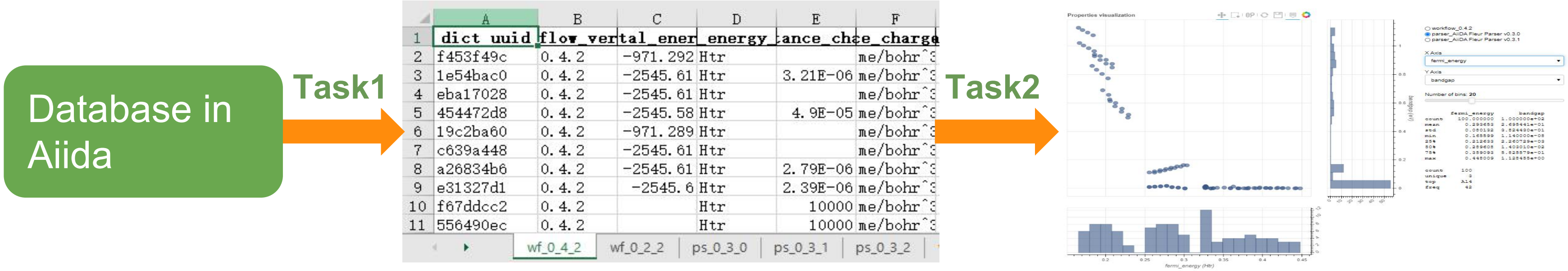
D2: Structure Property Visualizer

Sijie Luo

Aim: Connect the input "structure" to the properties in the output plot.

Task 1: Data acquisition. Extract float data in certain nodes and transform it into pandas objects, which will be then used as datasource.

Task 2: Interactive plot. Implement an interactive scatter plots with linked histograms by bokeh.



	A	B	C	D	E	F
1	dict uuid	flow_ver	tal_ener	_energy_	ance_ch	e_charge
2	f453f49c	0. 4. 2	-971. 292	Htr		me/bohr^3
3	1e54bac0	0. 4. 2	-2545. 61	Htr	3. 21E-06	me/bohr^3
4	eba17028	0. 4. 2	-2545. 61	Htr		me/bohr^3
5	454472d8	0. 4. 2	-2545. 58	Htr	4. 9E-05	me/bohr^3
6	19c2ba60	0. 4. 2	-971. 289	Htr		me/bohr^3
7	c639a448	0. 4. 2	-2545. 61	Htr		me/bohr^3
8	a26834b6	0. 4. 2	-2545. 61	Htr	2. 79E-06	me/bohr^3
9	e31327d1	0. 4. 2	-2545. 6	Htr	2. 39E-06	me/bohr^3
10	f67ddcc2	0. 4. 2		Htr	10000	me/bohr^3
11	556490ec	0. 4. 2		Htr	10000	me/bohr^3

wf_0_4_2 | wf_0_2_2 | ps_0_3_0 | ps_0_3_1 | ps_0_3_2

○ workflow_0.4.2
● parser_AiiDA Fleur Parser v0.3.0
○ parser_AiiDA Fleur Parser v0.3.1

X Axis: fermi_energy

Y Axis: bandgap

Number of bins: 20

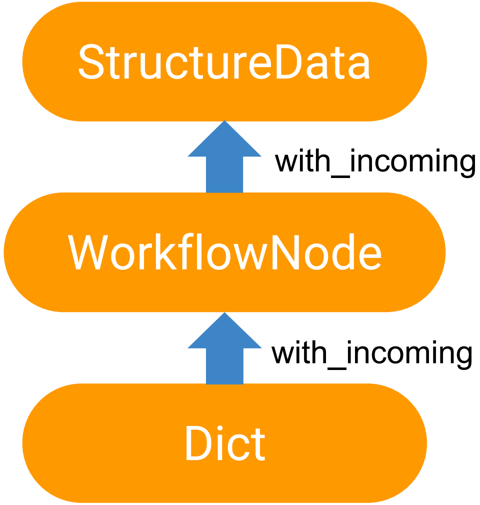
```
        fermi_energy       bandgap
count     100.000000  1.000000e+02
mean        0.293653  2.698441e-01
std         0.080132  3.824430e-01
min         0.165599  1.140000e-08
25%         0.212633  2.260729e-03
50%         0.289608  1.403010e-02
75%         0.359093  5.825579e-01
max         0.448009  1.128455e+00

count      100
unique       3
top        A14
freq        42
```

Figure 11: Structure property visualizer

D2 Task1: Data Acquisition

Core function & basic functions

```
def get_structure_workflow_dict(
    structure_project=['uuid', 'extras.formula'],
    structure_filters=None,
    workflow_project=['uuid', 'attributes.process_label'],
    workflow_filters=None,
    dict_project=['uuid'],
    dict_filters=None,
    timing=False, check_version=False)
```



StructureData

with_incoming

WorkflowNode

with_incoming

Dict

Figure 12: Data structure

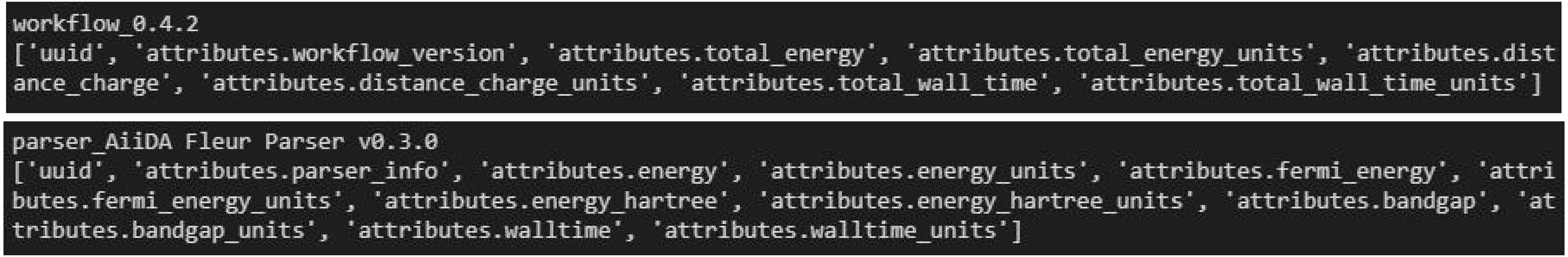
generate_**structure**_property_pandas_source, generate_**dict**_property_pandas_source, generate_**combined**_property_pandas_source

D2 Task1: Data Acquisition

A workflow may contain WorkflowNodes of various versions, which have very different output attributes.

Could **not** just retrieve the **same attributes** from all WorkflowNodes .

```
from helpers import predefined_workflow

for workflow_name, workflow in predifined_workflow.workflow_list.items():

    print(INVMAP[workflow_name])

    print(workflow.projections,'\n')
```



```
workflow_0.4.2
['uuid', 'attributes.workflow_version', 'attributes.total_energy', 'attributes.total_energy_units', 'attributes.dist
ance_charge', 'attributes.distance_charge_units', 'attributes.total_wall_time', 'attributes.total_wall_time_units']
```

```
parser_AiiDA Fleur Parser v0.3.0
['uuid', 'attributes.parser_info', 'attributes.energy', 'attributes.energy_units', 'attributes.fermi_energy', 'attri
butes.fermi_energy_units', 'attributes.energy_hartree', 'attributes.energy_hartree_units', 'attributes.bandgap', 'at
tributes.bandgap_units', 'attributes.walltime', 'attributes.walltime_units']
```

Figure 13: Projections of DictNodes for different workflow version in predefined_workflow

D2 Task1: Data Acquisition

Single workflow version

```
t1 = time.time()
structure_project=['uuid', 'extras.formula']
dict_project = predifined_workflow.get_workflow(MAP[versions[0]]).projections
combinednodes = helpers.generate_combined_property_pandas_source(
        version=versions[0],
        structure_project=structure_project,
        dict_project=dict_project,
        filename=f"combined_properties_{MAP[versions[0]]}.json")
all_times.append(time.time()-t1)
```

Output file

D2 Task1: Data Acquisition

Single workflow version —— Timings

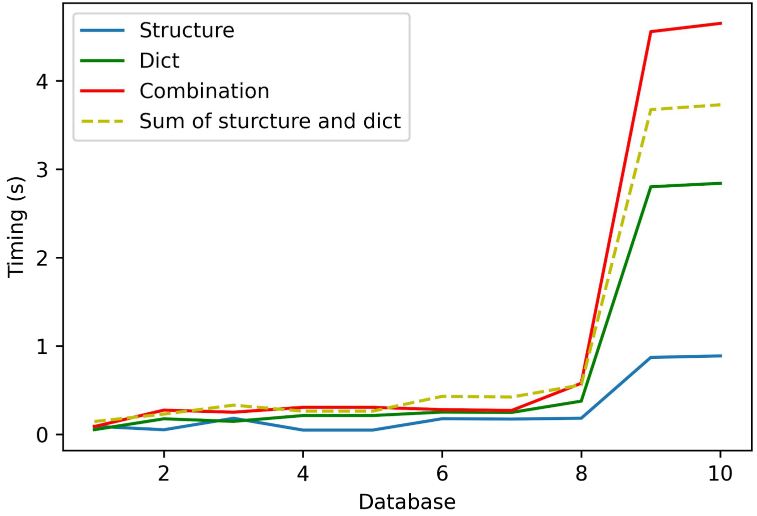


Figure 14: Running time of the three basic functions for the most frequent workflow version in different databases

Size of different databases

Database	1	2	3	4	5
Size(MB)	11	34	48	73	73

Database	6	7	8	9	10
Size(MB)	431	431	1738	1683	1683

D2 Task1: Data Acquisition

Multiple workflow versions

```
t1 = time.time()
filename='combined_properties_all.xlsx
for version in versions:
    structure_project=['uuid', 'extras.formula']
    dict_project = predifined_workflow.get_workflow(MAP[version]).projections
    combined_nodes = helpers.generate_combined_property_pandas_source(
         version=version, structure_project=structure_project,
    dict_project=dict_project)
    combined_nodes.to_excel(excel_writer, sheet_name=MAP[version], index=False)
all_times.append(time.time()-t1)
```

Output file

Save dataframe of each workflow version in each of a single sheet.

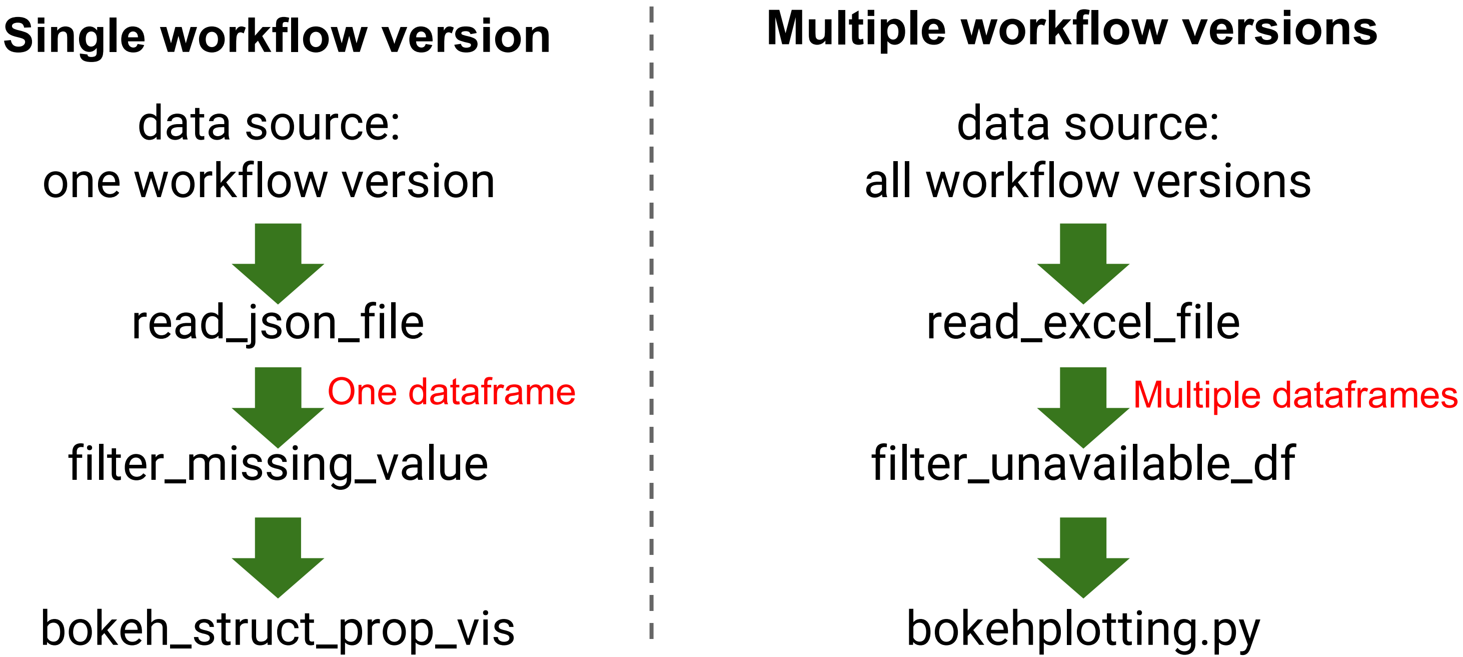
D2 Task1: Data Acquisition

Multiple workflow versions —— Timings

Type	Database size	Nodes	Process nodes	Data nodes	Structure timing (second)	Dict timing (second)	Combine timings (second)
unserialized	431 MB	50168	15529	34639	1.3346	1.0151	0.5829
serialized	431 MB	50183	15534	34649	0.6971	1.0655	0.5114

Database description:
800 Impurity (defect atoms) embeddings into different elemental host crystals with aiida-kkr.

D2 Task2: Interactive Plot



Single workflow version

data source:
one workflow version

read_json_file

One dataframe

filter_missing_value

bokeh_struct_prop_vis

Multiple workflow versions

data source:
all workflow versions

read_excel_file

Multiple dataframes

filter_unavailable_df

bokehplotting.py

D2 Task2: Interactive Plot

Single workflow version —— Result

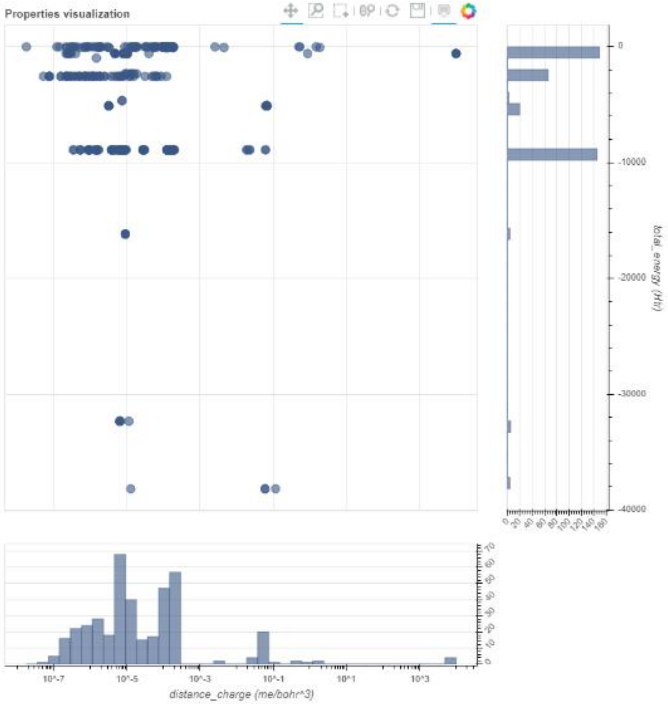


Figure 15: Interactivie plot with bokeh

Workflow version:
AiiDA Fleur Parser v0.3.2
Number of nodes:
502

X axis: distance_charge
X axis unit: me/bohr^3

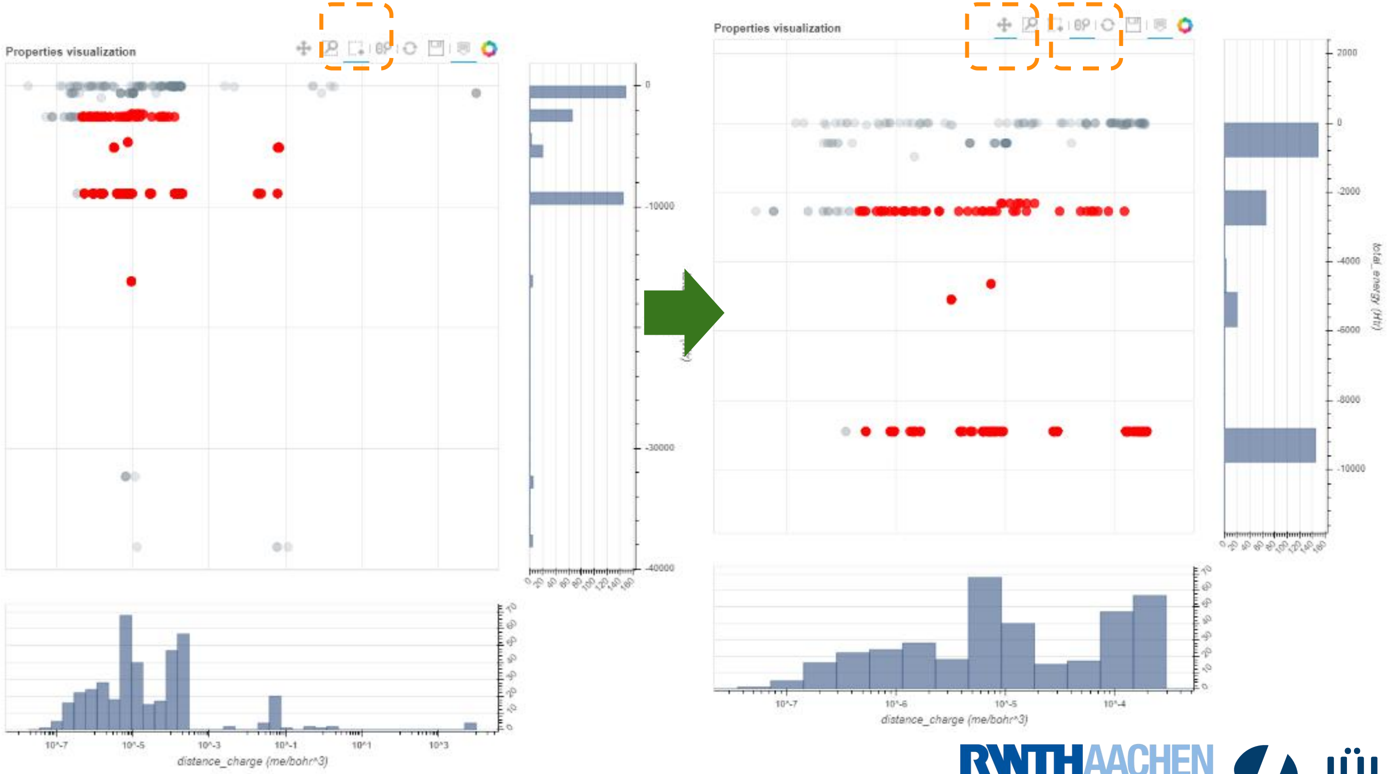
Y axis: total_energy
Y axis unit: Htr

D2 Task2: Interactive Plot

Single workflow version —— Interactiveness

Left:
Box select tool

Right:
Zooming and Pan



D2 Task2: Interactive Plot

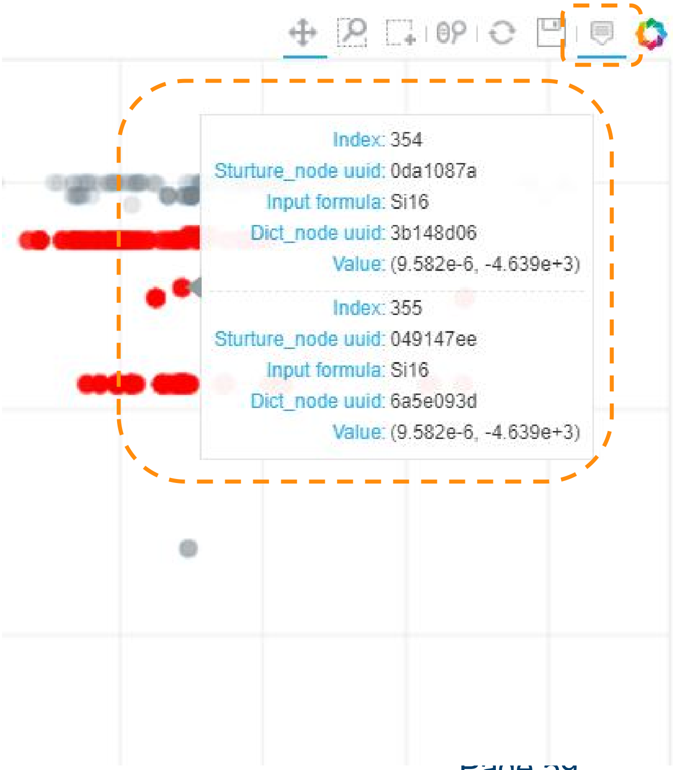
Single workflow version —— Interactiveness

Left:

Hovering on scatter plot

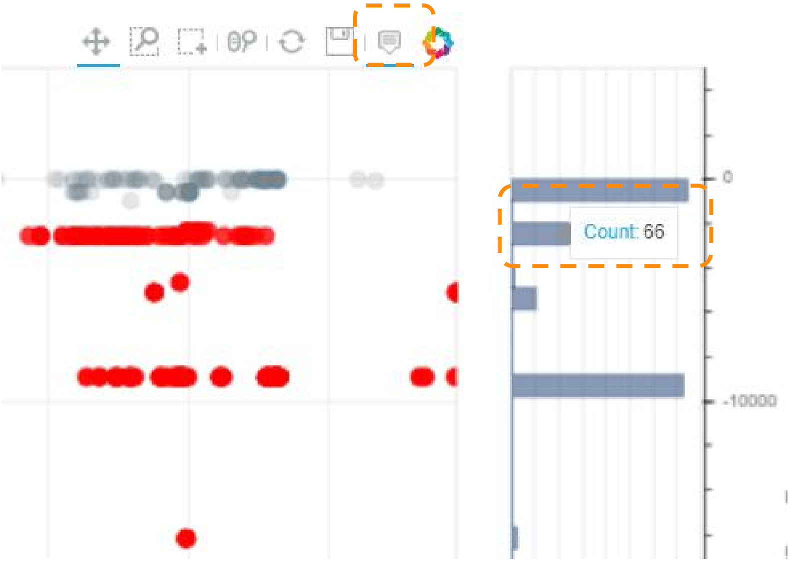
Right:

Hovering on histogram



Index: 354
Sturture_node uuid: 0da1087a
Input formula: Si16
Dict_node uuid: 3b148d06
Value: (9.582e-6, -4.639e+3)

Index: 355
Sturture_node uuid: 049147ee
Input formula: Si16
Dict_node uuid: 6a5e093d
Value: (9.582e-6, -4.639e+3)



Count: 66

D2 Task2: Interactive Plot

Single workflow version —— More plotting examples

Left:

'kkr_startpot_ps_0_3_2' ,

918 nodes

Right:

'AiiDA Fleur Parser v0.1beta',

7862 nodes

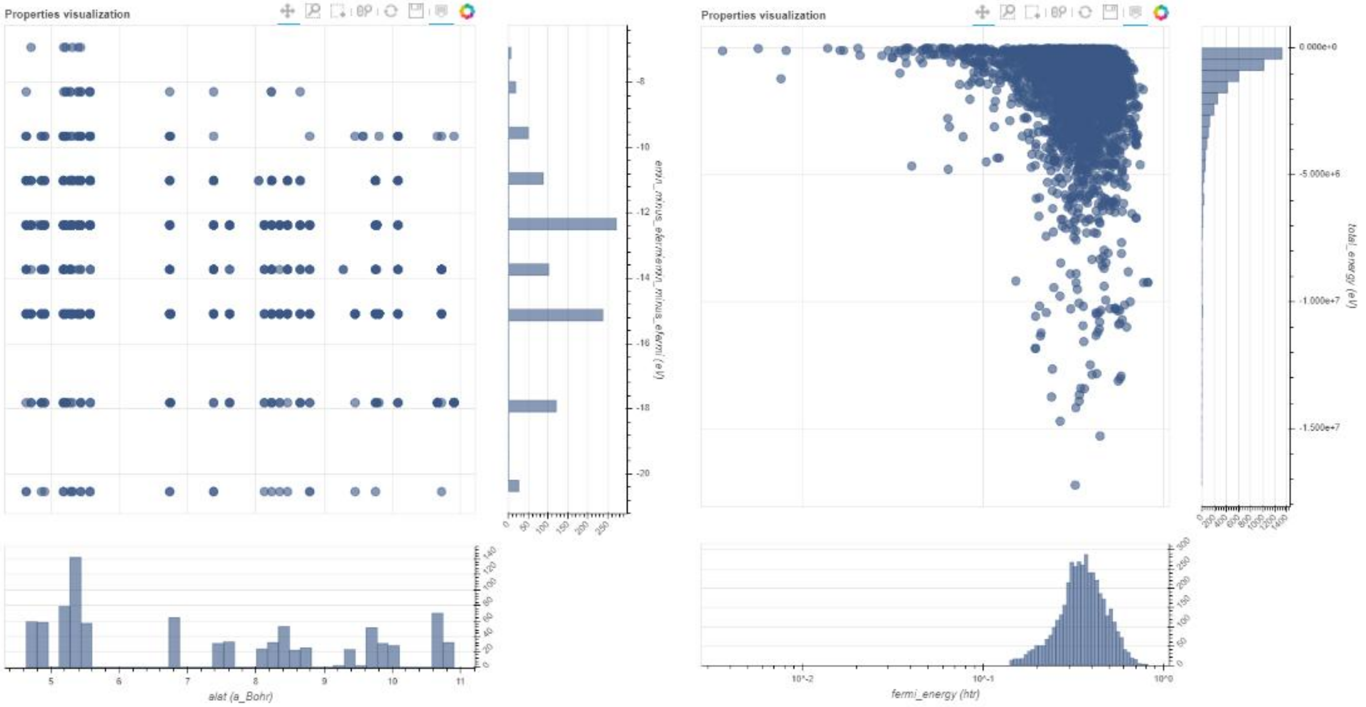


Figure 16: Another plotting examples

D2 Task2: Interactive Plot

Single workflow version —— Problem

Left:

total_energy

vs

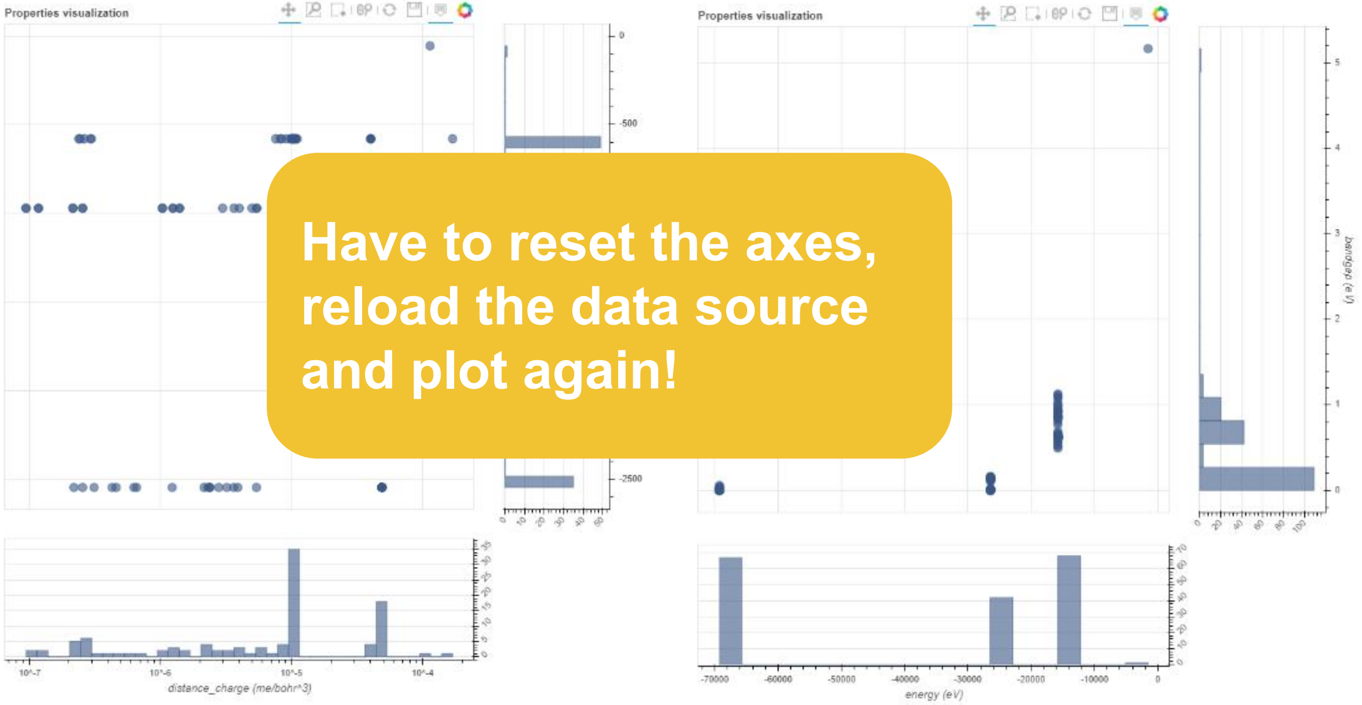
distance_charge

Right:

bandgap

vs

energy



Have to reset the axes, reload the data source and plot again!

Figure 17: Interactivie plot with different attributes

D2 Task2: Interactive Plot

Multiple workflow versions

Interactive plotting with Bokeh server application:

```
bokeh serve --show --port 5001 bokehplotting.py
```

Outline

- Problem description & AiiDA introduction
- Deliverable 1 (D1): Statistical birds eye view of the contents in an AiiDA database
- Deliverable 2 (D2): Structure property visualizer
- **Structure of the report**
- **Conclusion & outlook**

Structure of the report

- Introduction
- Theoretical Background
- Implementation
  - Deliverable 1
  - Deliverable 2
- Application
  - Deliverable 1
  - Deliverable 2
- Conclusion & Outlook
- Appendix

Conclusion & Outlook

So far...

- Successfully created deliverables that manage to (1) have a statistical birds eye view of the contents in an AiiDA database, (2) perform structure properties visualizer.
- A tool to retrieve, analyze and interactively visualize large database on AiiDA.
- Reduce the runtime by the serializer.

Next steps...

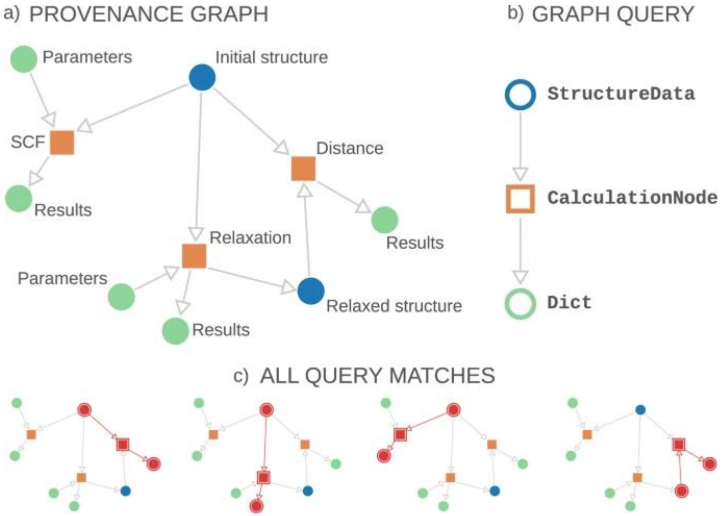
- Improve the performance on large databases.
- Implement more complex data analyses (e.g. clustering, dimension reduction...).
- Implement interactiveness customizability of plotting (e.g. log scaling).

Thanks for your attention!

Any further questions?

Motivation and requirements

- More information on Aiida:
- Furthermore, the workflows are queryable by users, which helps researcher to get the matching data easily:



D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

- **WorkflowNode:** Represent python code that orchestrates the execution of other workflows and calculations, that optionally return the data created by the processes they called.

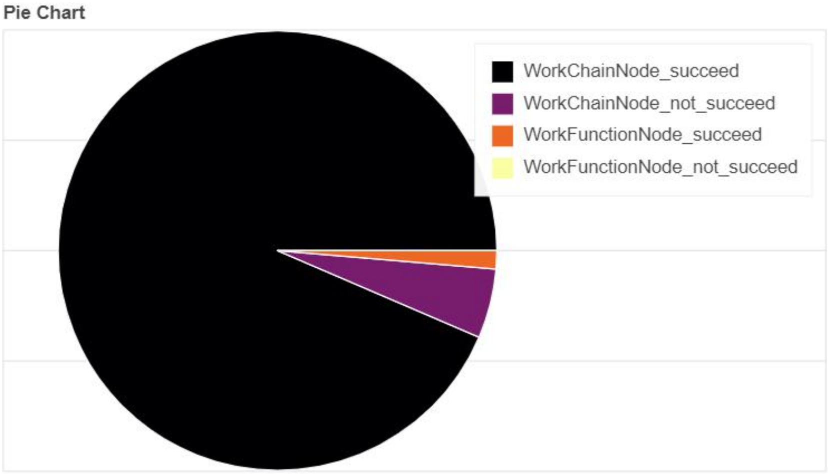


Figure 8

D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

- **CalculationNode:** Represent code execution that creates new data.

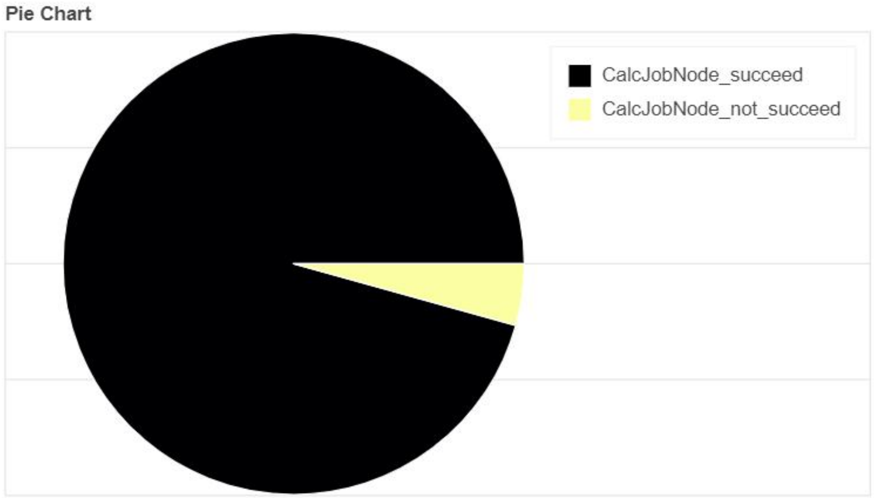
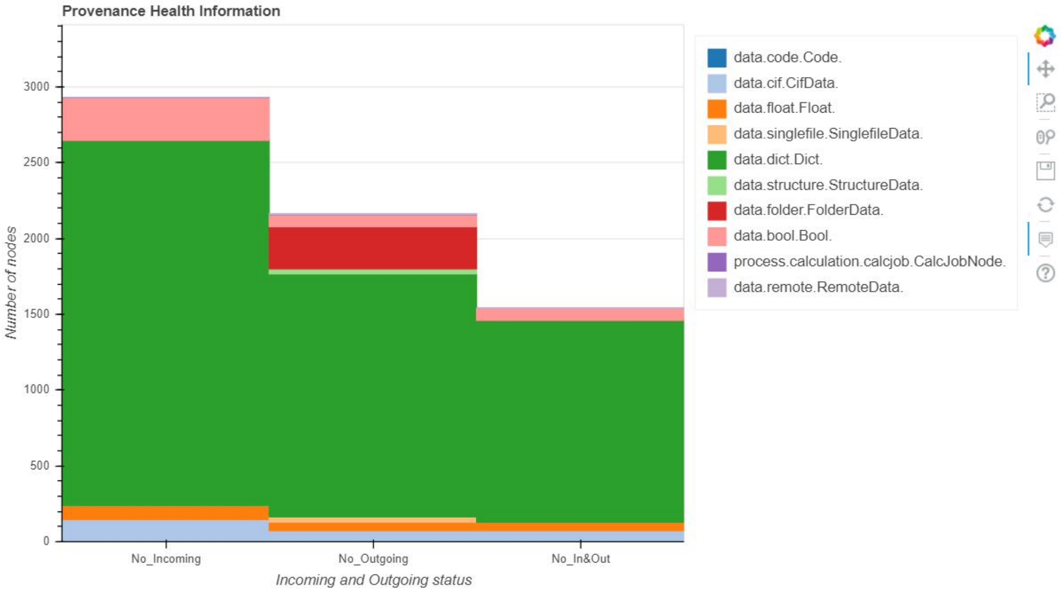


Figure 9

D1 Task i: Provenance

Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.



D2: Plugins

- AiiDA plugins will typically take care of preparing input files for a simulation code and parsing its output files into the AiiDA data types.

- **FLEUR**, a feature-full, freely available FLAPW (full-potential linearized augmented planewave) code, based on density-functional theory.

- The Jülich **KKR** suite contains codes for electronic structure calculations within density functional theory based on Korrings-Kohn-Rostoker (KKR) Green function method.

D2: Workflow version

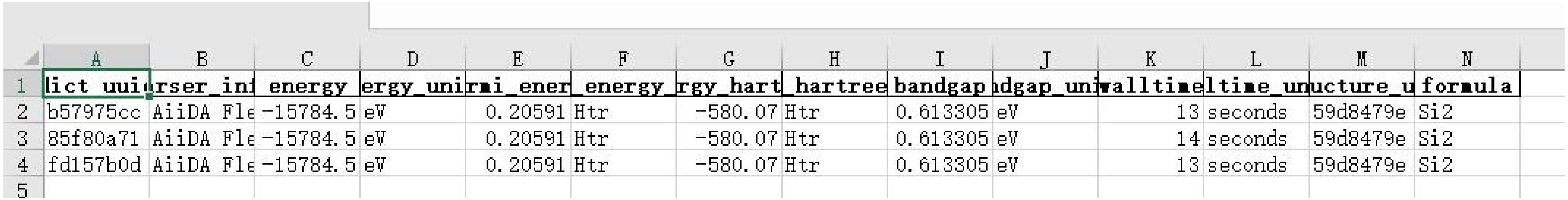
- To uniquely identify a workflow version, we need
  - attribute workflow_version, parser_version, or parser_info, which together we called **“version”** here,
  - attribute process_type, which we called **“type”** here, (checked by “verdi plugin list aiida.workflows”).
- So the **workflow version** we talked about is the tuple of **(type_name, version).**
- For examples,

Workflow	Workflow versions
'fleur_scf_wc'	('fleur_scf_wc', ‘workflow_0.4.2’), ('fleur_scf_wc', ‘workflow_0.3.0’), ('fleur_scf_wc', ‘AiiDA Fleur Parser v0.3.2’), …
'kkr_scf_wc'	('kkr_scf_wc', ‘workflow_0.10.4’), ('kkr_scf_wc', ‘workflow_0.12.0’), …

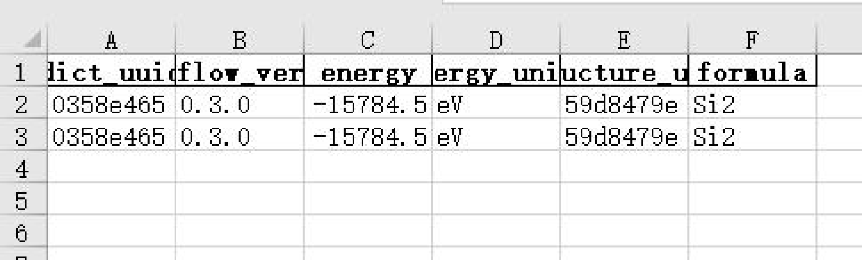
D2: filter_unavailable_df

After filtering_missing_value...

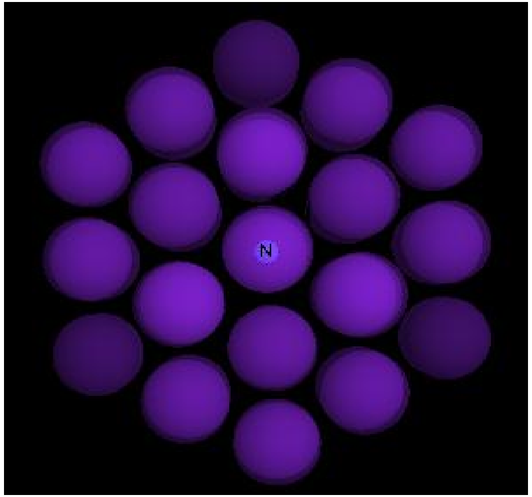
- Dataframe with too little nodes.
- Datafame with only one attribute.

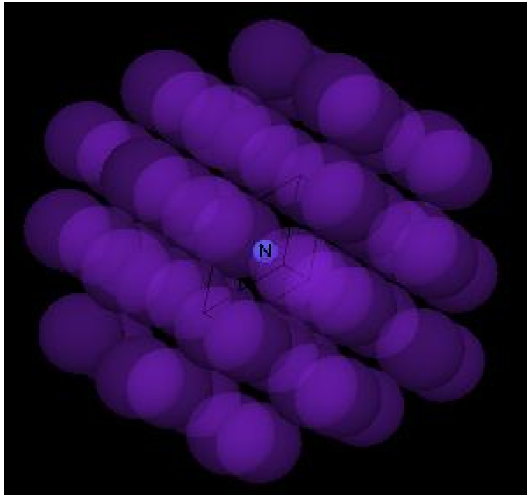


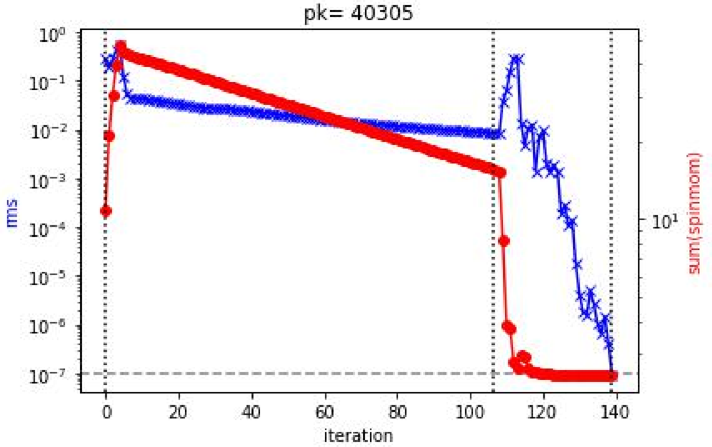
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	lict_uui	rser_inf	energy	ergy_uni	rmi_ener	_energy_	rgy_hart	_hartree	bandgap	dgap_uni	walltime	ltime_un	ucture_u	formula
2	b57975cc	AiiDA Fle	-15784.5	eV	0.20591	Htr	-580.07	Htr	0.613305	eV	13	seconds	59d8479e	Si2
3	85f80a71	AiiDA Fle	-15784.5	eV	0.20591	Htr	-580.07	Htr	0.613305	eV	14	seconds	59d8479e	Si2
4	fd157b0d	AiiDA Fle	-15784.5	eV	0.20591	Htr	-580.07	Htr	0.613305	eV	13	seconds	59d8479e	Si2
5														

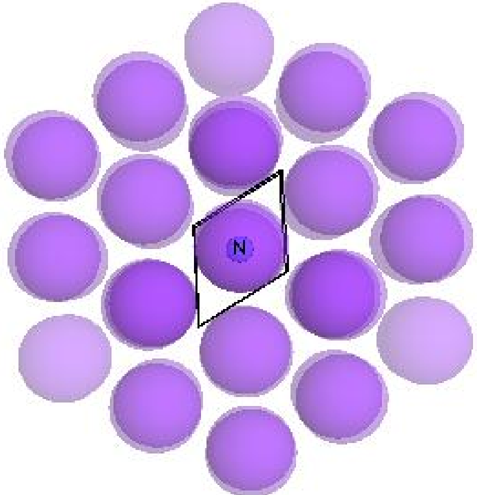


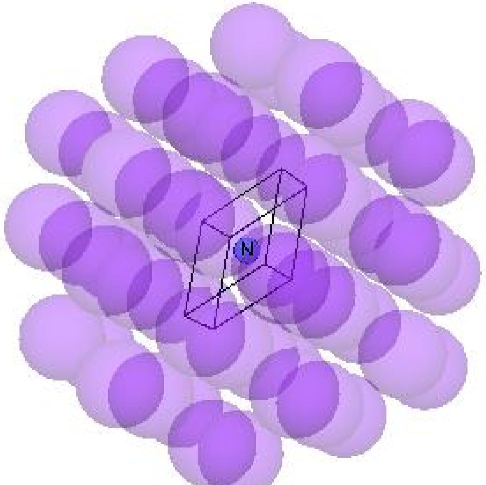
	A	B	C	D	E	F
1	lict_uui	flow_ver	energy	ergy_uni	ucture_u	formula
2	0358e465	0.3.0	-15784.5	eV	59d8479e	Si2
3	0358e465	0.3.0	-15784.5	eV	59d8479e	Si2
4						
5						
6						



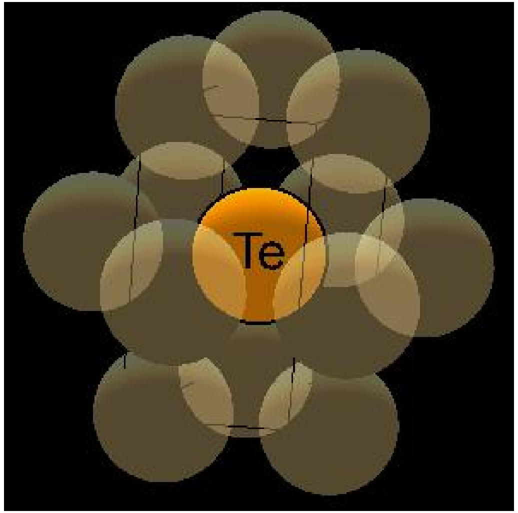


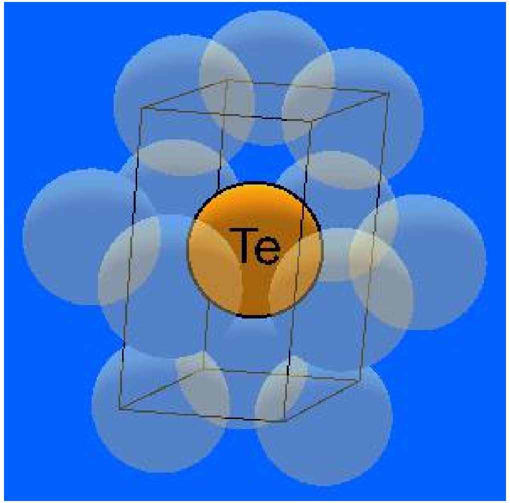


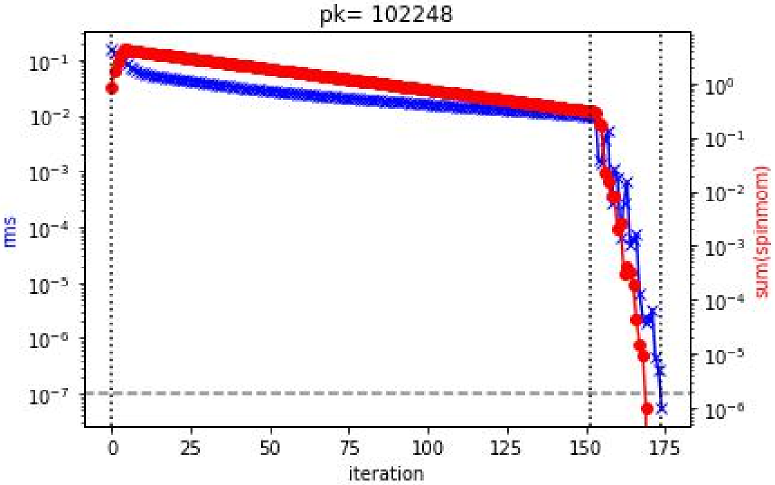




N_Cs from the database we used.





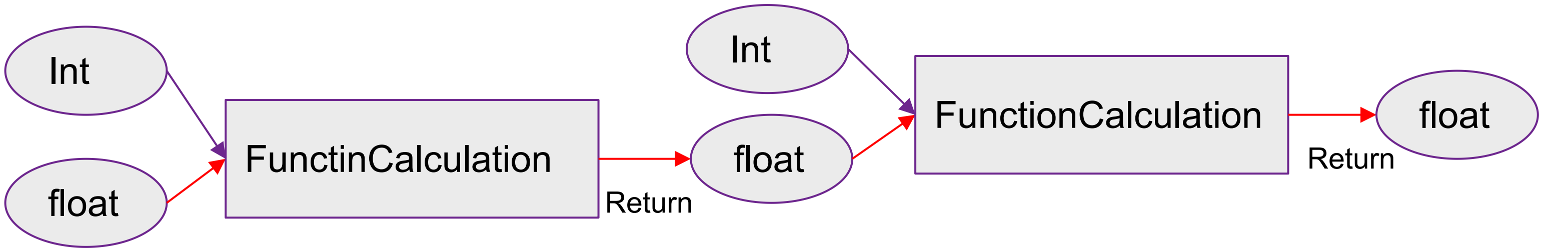


Te_Cd from the database we used.

- The example database holds a few hundred impurity workflows (kkr_imp_wc) for calculating the electronic structure of 'impurity embeddings' into different elemental crystals.
- Here two such systems are depicted, directly from the calculations: Te:Cd (meaning tellurium defect atom inside a cadmium crystal), and N:Cs (nitrogen defect in a caesiu crystal).
- The database when completed holds the cartesian product of embeddings of 60 different elemental impurities in 40 different elemental crystals from the periodic table. The aiida plugin which with these are calculated is aiida-kkr.

Problem description

- AiiDa is widely applied in Computational Science, which can record the computation as follows:



- The provenance together with the results stored in AiiDa database form a graph as above. From the graph one can trace results through simulation steps to their inputs.
- However, when the database is too large…

Motivation and requirements

- **When the database is too large, a useful analyse and visualize tool becomes crucial. There are already some tools.**

- **Aiida,** an infrastructure developed for data science by EPFL, automatically preserve and store the full data provenance in a relational database making it queryable and traversable.
- Aiida aims to help researchers manage the workflows concerned with Computational Science.

AiiDA

Automated Interactive Infrastructure and Database for Computational Science

Motivation and requirements

- More information on Aiida:
- Any calculation or workflow that is run by the engine will be automatically recorded in the provenance graph and stored in a database.Thus it becomes possible to **reconstruct the complete history of each calculation** or scientific result.

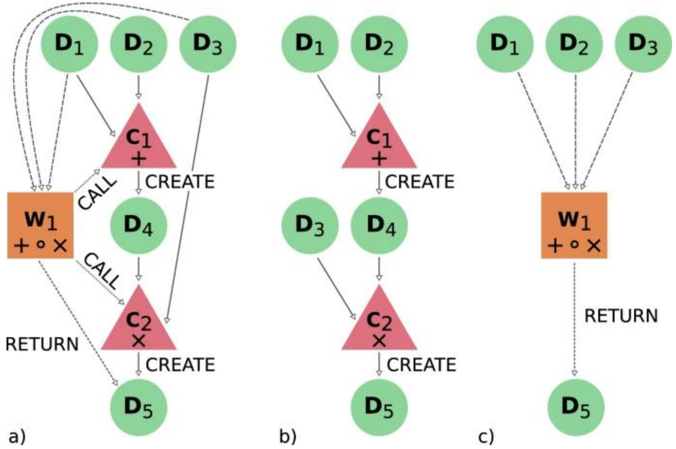


Figure 2: Aiida Provenance

D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

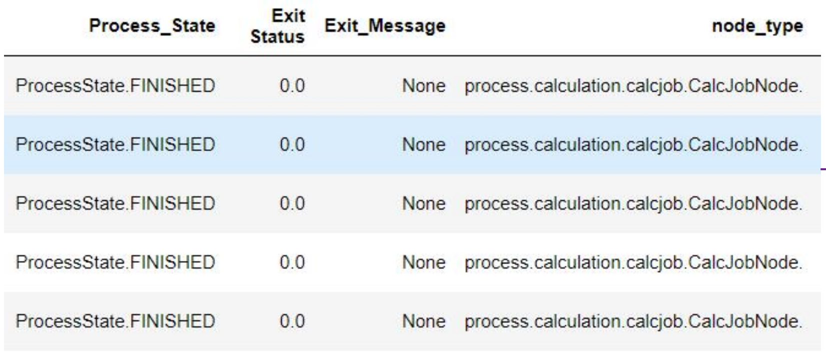
- **WorkflowNode:** Represent python code that orchestrates the execution of other workflows and calculations, that optionally return the data created by the processes they called.

- **CalculationNode:** Represent code execution that creates new data.

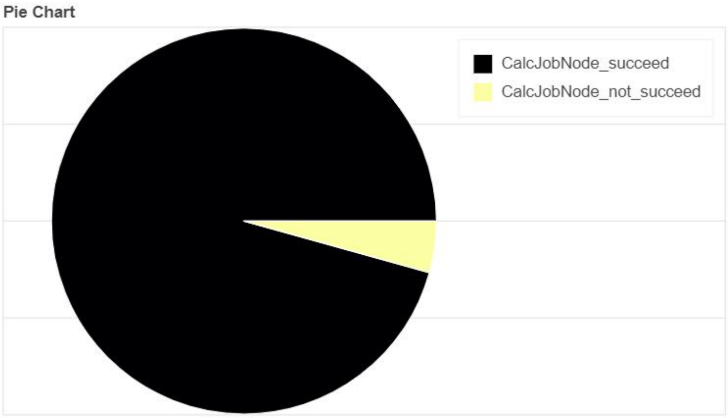
D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

- **CalculationNode:** Represent code execution that creates new data.



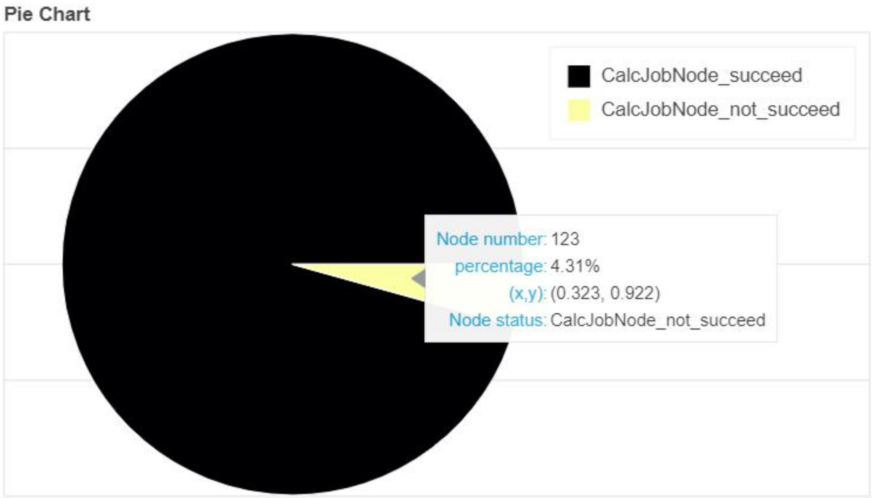
Process_State	Exit Status	Exit_Message	node_type
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.
ProcessState.FINISHED	0.0	None	process.calculation.calcjob.CalcJobNode.



D1 Task h: Process

Detail analysis of Calculations and Workflow. We want to analyse their exit status, exit message.

- **CalculationNode:** Represent code execution that creates new data.



Node number: 123
percentage: 4.31%
(x,y): (0.323, 0.922)
Node status: CalcJobNode_not_succeed

Motivation and requirements

- **When the database is too large, a useful analyse and visualize tool becomes crucial.**

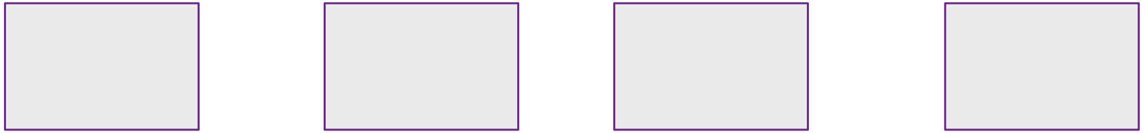
- With the help of **Aiida,** researchers can manage and access the database more easily. But we want to get a statistical, interactive visualization of a given database, which was not included in AiiDa before.

- We have developed a Database Analysis tool based on Aiida.

D1 Task i: Provenance

Analyse the health of workflow. Display the number of nodes that have no incoming links (any number outgoing), no outgoing links (any number incoming), and neither.

- **Provenance:** AiiDA automatically stores entities in its database and links them forming a **directed graph**. This directed graph automatically tracks the **provenance** of all data produced by calculations or returned by workflows. By tracking the **provenance** in this way, one can always fully retrace how a particular piece of data came into existence, thus ensuring its reproducibility.



SISC LAB PROJECT 7

ANALYSIS TOOL OF A MATERIALS DESIGN DATABASE

Supervisors: Prof. Dr. Stefan Blügel, Dr. Daniel Wortmann, Jens Bröder, Johannes Wasmer
Quantum Theory of Materials (PGI-1/IAS-1)

Forschungszentrum Jülich

Group 10: Sijie Luo, Miao Wang, Zhipeng Tan

2 March 2021

RWTHAACHEN
UNIVERSITY

JÜLICH
Forschungszentrum

Motivation

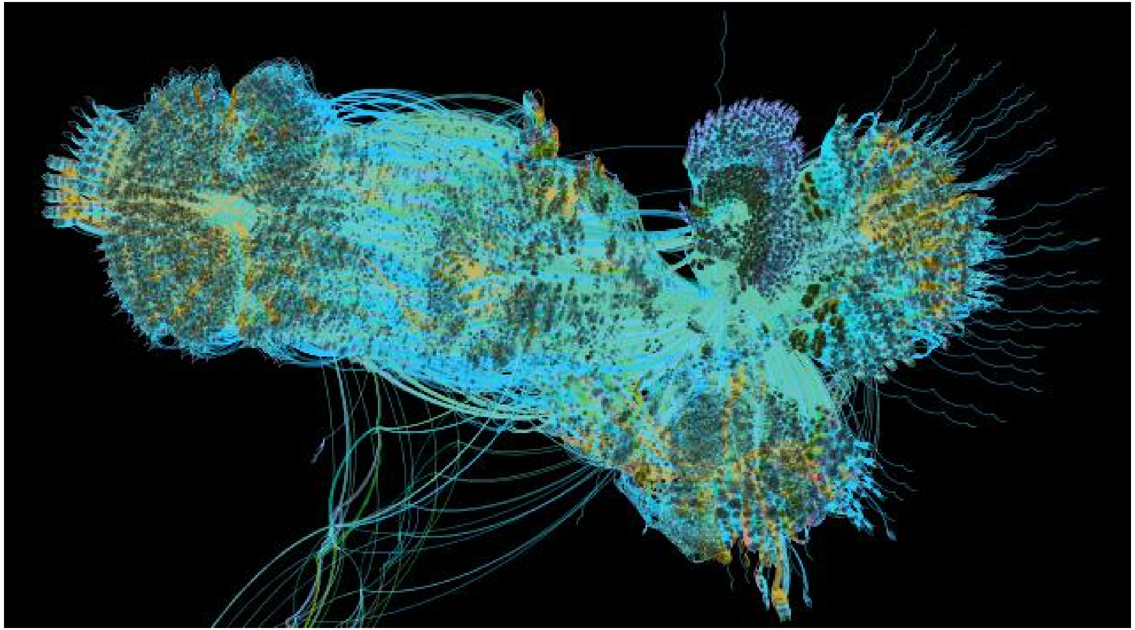


Figure 1: Large database visualization

~1000 FLEUR calculations with >100 K nodes. Every black dot is a node, clustering around the nodes represents the FLEUR code and its input generator on different machines and versions. Nodes far out represent crystal structures.

Authors: Jens Bröder, Daniel Wortmann, Stefan Blügel. Using the AiiDA-FLEUR package for all-electron ab initio electronic structure data generation and processing in materials science, IAS Series 40, p 43-48 (2019).

- **When the database is large, a useful analyse and visualize tool for statistics becomes crucial. But AiiDA doesn't have those functions yet.**