Python Reader for ADF Data Description

Dr. Aleksandar Jelenak
Allotrope Connect Workshop
September 2021
Outline

- *h5ld*, Python package for reading linked data in HDF5
- Working with ADF Data Description using *h5ld*
- Reading Data Description from ADF files in Amazon S3
- Storing ADF Data Description graph for reuse
h5ld: HDF5 Linked Data
**h5ld**: HDF5 Linked Data Reader

- Open-source Python package from The HDF Group.
- Community code contributions are welcome.
- Dependencies: Python 3.7 or later; *h5py*, *rdflib packages*; and HDF5 library v1.10.6 or later.
- Goal is to provide independent readers for HDF5-based formats with Linked Data
- Currently supported: Allotrope Data Format (ADF).
- Command-line and programmatic interface.
- On GitHub: [https://github.com/HDFGroup/h5ld](https://github.com/HDFGroup/h5ld)
Working with ADF Data Description Using h5ld
Command-Line Interface

Example:

    python -m h5ld -f json-ld -o output.json example.adf

- Read linked data from the ADF file and write it out in the JSON-LD format to output.json file.
- Output RDF formats: Turtle, JSON-LD, N-Quads, TriG.
- Omitting an output file will print the RDF content out for ingest by another command-line tool.

- Full description:
  python -m h5ld --help
Programmatic Interface

In [3]: ```python
with h5py.File("../..../Allotrope/examples/R180735_PQTEST_small.adf") as f:
    g = AllotropeDF(f).get_ld()
```

In [4]: ```python
g
```
Out[4]: `<Graph identifier=Nc5781f4de4d445a59725c047f267fd48 (<class 'rdflib.graph.ConjunctiveGraph'>)>

In [5]: ```python
len(g)
```
Out[5]: `894`
### Allotrope RDF Namespaces

```
In [6]: namespaces = dict((pre, str(iri)) for pre, iri in g.namespaces())
print(namespaces)

{'adf-dc': 'http://purl.allotrope.org/ontologies/datacube#',
 'adf-dc-hdf5': 'http://purl.allotrope.org/ontologies/datacube-hdf-map#',
 'adf-dd': 'http://purl.allotrope.org/ontologies/datadescription#',
 'adf-dp': 'http://purl.allotrope.org/ontologies/datapackage#',
 'af-a': 'http://purl.allotrope.org/ontologies/audit#',
 'af-c': 'http://purl.allotrope.org/ontologies/common#',
 'af-e': 'http://purl.allotrope.org/ontologies/equipment#',
 'af-m': 'http://purl.allotrope.org/ontologies/material#',
 'af-p': 'http://purl.allotrope.org/ontologies/process#',
 'af-q': 'http://purl.allotrope.org/ontologies/quality#',
 'af-r': 'http://purl.allotrope.org/ontologies/result#',
 'af-s': 'http://purl.allotrope.org/shape#',
 'af-sh': 'http://purl.allotrope.org/ontologies/shapes#',
 'af-x': 'http://purl.allotrope.org/ontologies/property#',
 'afs-hdf5': 'http://purl.allotrope.org/shapes/hdf5#',
 'dct': 'http://purl.org/dc/terms/',
 'hdf5': 'http://purl.allotrope.org/ontologies/hdf5/1.8#',
 'obo': 'http://purl.obolibrary.org/obo/',
 'owl': 'http://www.w3.org/2002/07/owl#',
 'pav': 'http://purl.org/pav/',
 'qb': 'http://purl.org/linked-data/cube#',
 'qudt': 'http://qudt.org/schema/qudt#',
 'qudt-quantity': 'http://qudt.org/vocab/quantity#',
 'qudt-unit': 'http://qudt.org/vocab/unit#',
 'rdf': 'http://www.w3.org/1999/02/22-rdf-syntax-ns#',
 'rdfs': 'http://www.w3.org/2000/01/rdf-schema#',
 'sh': 'http://www.w3.org/2001/sw/shacl#',
 'skos': 'http://www.w3.org/2004/02/skos/core#',
 'xml': 'http://www.w3.org/XML/1998/namespace',
 'xsd': 'http://www.w3.org/2001/XMLSchema#')
```
In [7]: list(g.contexts())

Out[7]: [<Graph identifier=urn:x-adf:TechnicalGraph (<class 'rdflib.graph.Graph'>)>,
        <Graph identifier=urn:x-adf:UserDataGraph (<class 'rdflib.graph.Graph'>)>,
        <Graph identifier=urn:x-adf:MetaGraph (<class 'rdflib.graph.Graph'>)>]
Visualization with *kglab* and PyVis

```python
kg = kglab.KnowledgeGraph(
    import_graph=g,
    namespaces=namespaces)

VIS_STYLE = {
    "hdf5": {
        "color": "red",
        "size": 30
    },
    "adf-dc-hdf5": {
        "color": "orange",
        "size": 30
    },
    "adf-dc": {
        "color": "purple",
        "size": 30
    },
    "adf-dp": {
        "color": "green",
        "size": 30
    },
    "adf-dd": {
        "color": "yellow",
        "size": 30
    },
    "qb": {
        "color": "black",
        "size": 30
    }
}

subgraph = kglab.SubgraphTensor(kg)
pyvis_graph = subgraph.build_pyvis_graph(notebook=True, style=VIS_STYLE)
pyvis_graph.force_atlas_2based(damping=2.0)
pyvis_graph.show("graph-vis.html")
```
SPARQL with kglab

```
In [14]: sparql = SPARQL_PRFX + """";
SELECT ?srcgrp ?srname target ?tgtype ?tgname {
   ?srcgrp a hdfs:Group;
   hdfs:member target;
   hdfs:name ?srname .
   target hdfs:name ?tgname .
   target a ?tgtype .
} """
kg.query_as_df(sparql)
```

```
Out[14]:

<table>
<thead>
<tr>
<th>srcgrp</th>
<th>srname</th>
<th>target</th>
<th>tgtype</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="">hdfs:/</a></td>
<td><a href="">hdfs:/data-cubes</a></td>
<td>hdfs:Group</td>
<td></td>
</tr>
<tr>
<td><a href="">urn:uuid:5c946658-8915-4473-ad88-40a27a23bd54</a></td>
<td>measures</td>
<td><a href="">urn:uuid:0be46729-ec9d-4a15-bca8-29e778de389f</a></td>
<td>hdfs:Dataset</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes/1</a></td>
<td>1</td>
<td><a href="">urn:uuid:23c3067c-280c-4438-a574-7c1a9a4df8b</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes/1</a></td>
<td>1</td>
<td><a href="">urn:uuid:aa38f230-7b34-4425-97af-55696a8e9f6</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">urn:uuid:1300633b-7b02-454d-85ef-74cd0c287085</a></td>
<td>scales</td>
<td><a href="">urn:uuid:6438948d-d0fe-4211-a8af-d01e939545</a></td>
<td>hdfs:Dataset</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes</a></td>
<td>data-cubes</td>
<td><a href="">hdfs:/data-cubes/2</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes</a></td>
<td>data-cubes</td>
<td><a href="">hdfs:/data-cubes/0%20%281%29</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes</a></td>
<td>data-cubes</td>
<td><a href="">hdfs:/data-cubes/1%20%281%29</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes</a></td>
<td>data-cubes</td>
<td><a href="">hdfs:/data-cubes/0</a></td>
<td>hdfs:Group</td>
</tr>
<tr>
<td><a href="">hdfs:/data-cubes</a></td>
<td>data-cubes</td>
<td><a href="">hdfs:/data-cubes/1%20%281%29</a></td>
<td>hdfs:Group</td>
</tr>
</tbody>
</table>
```

query_graph = kg.visualize_query(sparql, notebook=True)
query_graph.force_atlas_2based()
query_graph.show("query-vis.html")
Reading Data Description from ADF files in Amazon S3
Local File

- This is the baseline, traditional file access.
- Using an ADF file with dummy Data Description.
- File: QueryTest.adf; size: 63.89 MB; number of RDF triples: 588,082
- Operations: (1) Read HDF5 datasets holding RDF triple data; (2) Decode RDF subject, predicate, and object information; (3) Form N-Quads statements; (4) Create an rdflib graph object by parsing the N-Quads statements.

```python
In [23]: %time
with h5py.File("../Allotrope/examples/QueryTest.adf") as f:
    g = AllotropeDF(f).get_id()

CPU times: user 1min 7s, sys: 547 ms, total: 1min 7s
Wall time: 1min 8s
```
S3 Object with HDF5 ROS3 VFD

- Read-only S3 Virtual File Driver (VFD) is available with the HDF5 library (build time option).
- Its performance is greatly influenced by the creation properties of the HDF5 file and its datasets.

```python
In [29]: %time

s3url = ("https://s3.us-west-2.amazonaws.com/
        "hdf5.sample/data/Allotrope/QueryTest.adf")

with h5py.File(s3url, mode="r", driver="ros3",
               aws_region=b"us-west-2",
               secret_id=os.environ["aws_access_key_id"].encode(),
               secret_key=os.environ["aws_secret_access_key"].encode()) as f:
    g_vfd = AllotropeDF(f).get_id()

CPU times: user 1min 8s, sys: 1.5 s, total: 1min 10s
Wall time: 4min 48s
```
S3 Object with `fsspec` as HDF5 VFL

- `h5py` can use a Python file-like object as an HDF5 virtual file layer
- `fsspec` is a Python package with support for many local, remote and embedded file systems and object stores.

```python
In [16]: s3fs = fsspec.filesystem(protocol='s3')

In [24]: %time

    with s3fs.open("s3://hdf5.sample/data/Alлотrope/QueryTest.adf", mode="rb") as adf_s3:
        with h5py.File(adf_s3, mode="r") as f:
            g_s3 = AllotropeDF(f).get_td()

CPU times: user 1min 11s, sys: 2.23 s, total: 1min 13s
Wall time: 2min 10s
```
Downloaded S3 Object as Local File

- Download from object store, then use as a local file.
- Specific to ADF files with reasonable download times.

```python
In [30]: %time
s3fs.get("s3://hdf5.sample/data/Allotrope/QueryTest.adf", "./"
with h5py.File("QueryTest.adf", mode="r") as f:
    g_dwnld = AllotropeDF(f).get_ld()

CPU times: user 1min 7s, sys: 866 ms, total: 1min 8s
Wall time: 1min 13s
```
HSDS with HDF cloud-native file format

- The original ADF file was ingested into an HDF Highly Scalable Data Service (HSDS) system.
- HSDS ingest converts an HDF5 file into the HDF cloud-native format and creates HDF5 datasets with larger chunks.
- This access method requires the h5pyd package, and a user account on the HSDS instance.

```python
In [33]: %time

with h5pyd.File("/home/ajelenak/Allotrope/hsds/QueryTest.adf", mode="r") as f:
    g_hsds = AllotropeDF(f).get_ld()

CPU times: user 1min 7s, sys: 591 ms, total: 1min 7s
Wall time: 1min 13s
```
HSDS with Original ADF File

- HSDS can also work with the original ADF file.
- HSDS ingest in this case only derives information about file locations of the HDF5 dataset chunks.

```python
In [36]: %time

with h5py.File("/home/ajelenak/Allotrope/file/QueryTest.adf", mode="r") as f:
    g_hsds = AllotropeDF(f).get_id()

CPU times: user 1min 7s, sys: 626 ms, total: 1min 8s
Wall time: 1min 23s
```
Storing ADF Data Description Graph for Reuse
Parquet File

- *kglab* supports storing and loading graph data in Parquet.
- Parquet file size: 31.4 MB.
- Downside: The Parquet file holds only N-Triple statements.

```
In [39]: %time

    kg_s3.save_parquet("s3://hdf5.sample/Trash/graph.parquet")

    CPU times: user 14.4 s, sys: 538 ms, total: 14.9 s
    Wall time: 37.9 s

In [44]: %time

    kg_s3 = kglab.KnowledgeGraph()
    kg_s3.load_parquet("s3://hdf5.sample/Trash/graph.parquet")

    CPU times: user 2min 26s, sys: 17.8 s, total: 2min 44s
    Wall time: 2min 52s

Out[44]: <kglab.kglab.KnowledgeGraph at 0x16f64a610>
```
Compressed N-Quads Text File

- Text file with the ADF file’s Data Description N-Quads statements.
- File size: 20.1 MB.

```
In [41]: %time
    
    nquads = io.BytesIO()
    g_s3.serialize(nquads, format="nquads")
    nquads.seek(0, io.SEEK_SET)
    
    with s3fs.open("s3://hdf5.sample/Trash/QueryTest.nquads.deflate", mode="wb") as outf:
        outf.write(
            zlib.compress(
                nquads.getbuffer(),
                zlib.Z_DEFAULT_COMPRESSION
            )
        )
    
    nquads.close()
```

CPU times: user 17.4 s, sys: 366 ms, total: 17.8 s  
Wall time: 34.9 s

```
In [46]: %time
    
    g_s3 = rdflib.ConjunctiveGraph()
    with s3fs.open("s3://hdf5.sample/Trash/QueryTest.nquads.deflate", mode="rb") as outf:
        buf = io.BytesIO()
        zlib.decompress(
            outf.read()
        )
    
    g_s3.parse(buf, format="nquads")
    buf.close()
```

CPU times: user 44.8 s, sys: 520 ms, total: 45.3 s  
Wall time: 48.7 s
THANK YOU!

Questions & Comments?
ajelenak@hdfgroup.org
help@hdfgroup.org