DMP Implementation Status of IPP

2024-03-20

Since we are committing to UDA …

… I thought it worthwhile to perform some timing tests

None of the data considered in the following slides is particularly large

- equilibrium : 16.200 MB is the largest
- core_profiles < 1 MB
- summary < 0.04 MB

Slowdowns of 580.12, 210.02, 367.15 for “get: of summary, core_profiles and equilibrium

Timing tests (Gateway accessing test data at IPP)

```
./uda.py -u 'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/test/3/1000/1000&backend=hdf5'

Accessing data from imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/test/3/1000/1000&backend=hdf5
Time data for IDS edge_profiles
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]


Timing information
    DBentry = 30.486
       open =  0.146
        get = 23.105
      close =  0.041
```

Timing tests (Gateway accessing summary data at IPP)

```
./uda.py -u 'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug-dmp/3/41570/0&backend=hdf5' --case
summary

Accessing data from imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug-dmp/3/41570/0&backend=hdf5
Time data for IDS summary
[0.158 0.258 0.358 0.458 0.558 0.658 0.758 0.858 0.958 1.058 1.158 1.258
 1.358 1.458 1.558 1.658 1.758 1.858 1.958 2.058 2.158 2.258 2.358 2.458
 2.558 2.658 2.758 2.858 2.958 3.058 3.158 3.258 3.358 3.458 3.558 3.658
 3.758 3.858 3.958 4.058 4.158 4.258 4.358 4.458 4.558 4.658 4.758 4.858
 4.958 5.058 5.158 5.258 5.358 5.458 5.558 5.658 5.758 5.858 5.958 6.058
 6.158 6.258 6.358 6.458 6.558 6.658 6.758 6.858 6.958 7.058 7.158 7.258
 7.358 7.458 7.558 7.658 7.758 7.858 7.958 8.058 8.158 8.258 8.358 8.458
 8.558 8.658 8.758 8.858 8.958 9.058 9.158]


Timing information
    DBentry = 30.432
       open =  0.159
        get = 29.006
      close =  0.049
```

Timing tests (Gateway accessing core_profile data at IPP)

```
./uda.py -u 'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5' --case
core_profiles

Accessing data from imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5
Time data for IDS core_profiles
[0.31600001 0.41600001 0.51600001 0.61600001 0.71600001 0.81600001
 0.91600001 1.01600001 1.11600001 1.21600001 1.31600001 1.41600001
 1.51600001 1.61600001 1.71600001 1.81600001 1.91600001 2.01600001
 2.11600001 2.21600001 2.31600001 2.41600001 2.51600001 2.61600001
 2.71600001 2.81600001 2.91600001 3.01600001 3.11600001 3.21600001
 3.31600001 3.41600001 3.51600001 3.61600001 3.71600001 3.81600001
 3.91600001 4.01600001 4.11600001 4.21600001 4.31600001 4.41600001
 4.51600001 4.61600001 4.71600001 4.81600001 4.91600001 5.01600001
 5.11600001 5.21600001 5.31600001 5.41600001 5.51600001 5.61600001
 5.71600001 5.81600001 5.91600001 6.01600001 6.11600001]


Timing information
    DBentry = 30.443
       open =  0.156
        get = 39.903
      close =  0.053
```

Timing tests (Gateway accessing equilibrium data at IPP)

```
./uda.py -u 'imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5' --case
equilibrium

Accessing data from imas://uda.ipp.mpg.de:56565/uda?path=/root/public/imasdb/aug/3/17151/1&backend=hdf5
Time data for IDS equilibrium
[0.31600001 0.41600001 0.51600001 0.61600001 0.71600001 0.81600001
 0.91600001 1.01600001 1.11600001 1.21600001 1.31600001 1.41600001
 1.51600001 1.61600001 1.71600001 1.81600001 1.91600001 2.01600001
 2.11600001 2.21600001 2.31600001 2.41600001 2.51600001 2.61600001
 2.71600001 2.81600001 2.91600001 3.01600001 3.11600001 3.21600001
 3.31600001 3.41600001 3.51600001 3.61600001 3.71600001 3.81600001
 3.91600001 4.01600001 4.11600001 4.21600001 4.31600001 4.41600001
 4.51600001 4.61600001 4.71600001 4.81600001 4.91600001 5.01600001
 5.11600001 5.21600001 5.31600001 5.41600001 5.51600001 5.61600001
 5.71600001 5.81600001 5.91600001 6.01600001 6.11600001]


Timing information
    DBentry = 30.425
       open =  0.159
        get = 95.459
      close =  0.052
```

Timing tests (Gateway accessing data at IPP)

Action	test	summary	core_profiles	equilibrium
DBentry	30.486	30.432	30.443	30.425
open	0.146	0.159	0.156	0.159
get	23.105	29.006	39.903	95.459
close	0.041	0.049	0.053	0.052

Notes:

- “close” is completely negligible
- “open” is also small
- “DBEntry” is large but doesn’t seem to scale with data size
  - Why is this so large?
- “get” is large and scales with data size (in some way)
  - Larger than I would like!

Additional notes:

- “rsyncing” the entire 17151 trview data to ITER took less than 1 second
  - sent 7,161,199 bytes received 242 bytes 4,774,294.00 bytes/sec
  - total size is 7,158,729 speedup is 1.00
  - 0.018u 0.032s 0:00.66 6.0% 0+0k 0+0io 0pf+0w
- I think we need:
  - Chunking of the data
  - Possible multiple streams
  - Caching of the data (if allowed by the sender) on the receiving side

uda.py

```
#! /usr/bin/env python

import imas
import argparse
import time

parser = argparse.ArgumentParser(description="Tests UDA access",
                                 formatter_class=argparse.RawTextHelpFormatter)
parser.add_argument("-u", "--uri", type=str, required=True,
                            help="URI to be accessed")
parser.add_argument("-c", "--case", type=str,
                    help="""
Case parameter: "edge_profiles" for the edge_profiles test case
                "summary" for a DMP summary IDS
                "core_profiles" for a constructed core_profiles
                "equilibrium" for a constructed equilibrium"
""", default='edge_profiles',)
args=parser.parse_args()

print(f"Accessing data from {args.uri}")

times = [time.time()]
data = imas.DBEntry(args.uri, 'r')
times.append(time.time())
data.open()
times.append(time.time())
ids = data.get(args.case)
times.append(time.time())
data.close()
times.append(time.time())

print(f'Time data for IDS {args.case}')
print(ids.time)

print(f'\n\n\nTiming information\n    DBentry = {times[1] - times[0]:6.3f}\n       open = {times[2] - times[1]:6.3f}\n        get = {times[3] - times[2]:6.3f}\n      close =
{times[4] - times[3]:6.3f}')
```

Accessing data directly (Citrix server at IPP)

```
analyze_db_sizes -u $USER -d aug-dmp -b HDF5 41570 0

Examining data for dpc/aug-dmp/3/41570/0

Reading 0.001 MB of data for dataset_description/0 took 0.02 seconds
Reading 0.038 MB of data for summary/0 took 0.05 seconds
```

```
analyze_db_sizes -u $USER -d aug -b HDF5 17151 1

Examining data for dpc/aug/3/17151/1

Reading 0.844 MB of data for core_profiles/0 took 0.19 seconds
Reading 0.001 MB of data for dataset_description/0 took 0.02 seconds
Reading 16.200 MB of data for equilibrium/0 took 0.26 seconds
Reading 0.017 MB of data for ic_antennas/0 took 0.03 seconds
Reading 0.018 MB of data for nbi/0 took 0.04 seconds
Reading 0.017 MB of data for pulse_schedule/0 took 0.03 seconds
Reading 0.017 MB of data for summary/0 took 0.04 seconds
Reading 0.001 MB of data for tf/0 took 0.02 seconds
Reading 0.013 MB of data for wall/0 took 0.04 seconds
```

SUMMARY IDS fields

I have reactivated a table I created during FAIR4fusion

- This contains all (at the time of creation) fields of the summary IDS
- Contains the signal names used for mapping AUG and JET (my mapping)
- Added the TSVV11 requests from the table TSV11_0d_signals.txt that Pär sent me
- Link to the table: https://docs.google.com/spreadsheets/d/1lX61xqM6sDMjbwpr-HG5NR8tlbauQ-NkCxZeKhXjYKY/edit?usp=sharing
- Would like to see if there are other fields we would like to have before I re-run the data capture process

Back to duckdb

- Generated data following the instructions in the git@gitlab.eufus.psnc.pl:g2jwasik/miariadb-performance-test.git repository
- Then read the data into duckdb, creating a persistent duckdb.db file
- Transferred this to the Gateway
- Ran the following queries there
  - Slight modification from the mariadb calls
    - Dropped some “optimizations”
    - Changed “ to ‘
    -

Query 1

```
### Query 1 (get all experiment id that contain values in selected range)
```

```
SELECT DISTINCT ex.id, ex.uri
FROM experiment ex
JOIN request r ON ex.id = r.experiment_id
JOIN entry e ON e.request_id = r.id
JOIN entry_data ed ON ed.entry_id = e.id
JOIN variable v ON v.id = ed.variable_id
WHERE v.variable_name = 'global_quantities/ip/value' AND
ed.variable_value_double BETWEEN 10 AND 20;

--execution time: so long it cannot be measured
```

```
┌──────┬──────────────┐
│      622 rows       │
│     (40 shown)      │
└─────────────────────┘
Run Time (s): real 0.673 user 6.835828 sys 2.418384
```

Query 2

Query 2 (get all experiment id that contain values in selected range)
```
SELECT  DISTINCT ex.id, ex.uri
FROM experiment ex
JOIN request r ON ex.id = r.experiment_id
JOIN entry e ON e.request_id = r.id
WHERE e.id IN(
SELECT DISTINCT ed.entry_id FROM entry_data ed
WHERE
ed.variable_id = 10018 AND
ed.variable_value_double BETWEEN 10 AND 20
);

--execution time: 7.68s

```
```
┌──────────┐
│ 622 rows │
│ (40 shown) │
└──────────┘
Run Time (s): real 0.040 user 1.557309 sys 0.015055
```

Query 3

Query 3 (get all experiment id that contain values in selected range)

```
SELECT DISTINCT ex.id, ex.uri
FROM experiment ex
WHERE ex.id IN
(
    SELECT DISTINCT r.experiment_id FROM request r
    WHERE r.id IN
    (
        SELECT DISTINCT e.request_id FROM entry e
        WHERE e.id IN
        (
            SELECT DISTINCT ed.entry_id FROM entry_data ed
            WHERE
            ed.variable_id = 10018 AND
            ed.variable_value_double BETWEEN 10 AND 20
        )
    )
);

--execution time: 4.1s
```

```
|-------|--------------|
|      622 rows        |
|     (40 shown)       |
|----------------------|
```

```
Run Time (s): real 0.041 user 1.492670 sys 0.025563
```

Next steps, I

- Start exploring the mysteries of authentication and authorization
  - simdb uses
    - A token for the Gateway (seems to work well)
      - Presumably based on the Gateway LDAP
      - This is stored in “~/.config/simdb/simdb.cfg”
    - A username and password for ITER (annoying – one is tempted to hack the code to just store the password in a local file)
  - For uda need
    - A discussion of what options there are
      - Tunnels based on some sort of certificates
      - What hooks are available on the server
    - And a discussion with the local site
      - About what security is needed and how it should be implemented
        - Keycloak authentication against EUROfusion LDAP and authorization by a group maintained in EUROfusion LDAP?
        - Keycloak authentication against EUROfusion LDAP and inclusion in a locally maintained authorization file?
        - Keycloak authentication against the site local LDAP and authorization by a group maintained in LDAP?
        - Locally or centrally maintained X509 certificates?
  - How do the uploaded ssh certificates work in github/gitlab?
    - Could we do something similar?

Next steps, II

- These are the possible options I can see to provide mapped access for AUG data
  - Use the now existing DMZ UDA to map calls to the IPP mdsplus server
    - Not sure how well supported the MDSplus server will be
    - Not sure if we can establish a good security model
  - Use the existing DMZ UDA server to talk to an internal UDA server (with access to the shotfile data storage) that then maps onto direct data access calls (libdd, libww, …)
  - Allow the now existing DMZ UDA server read access to the AUG shotfiles directly
    - Special NFS mount?

- Is there a tutorial somewhere about setting up a private UDA server with support for plugins and IMAS in user space?