WHO Integrated Data Platform (WIDP)

IT Challenges and solutions of a global platform

DHIS2 Annual conference, Oslo, 18 June 2019
Outline

● WHO Integrated Data Platform (WIDP) - Quick overview (5 mins)
● DB server performance challenges (20 mins + 10 mins Q&A)
● Metadata packaging & sync challenges (15 mins + 5 mins Q&A)
● Organisation Unit Tree sync challenges (10 mins + 5 mins Q&A)
● Conclusion (10 mins)
● Remaining Q&A (10 min)
Overcoming challenges of (NTDs) data

Data-driven decision making

Action

Data collection

Data flow

Data validation

Data analysis

Data dissemination

WHO Integrated Data Platform

Standardization

Information systems

Strengthening (Country + Global)

Capacity building

Improved (NTDs) indicators

World Health Organization
Complementary use of DHIS2 by WHO

- Strategic tool for several WHO programmes
  - Country system strengthening
  - Dissemination of standards and data quality practices
  - Monitoring key targets (SDGs, GPW13…)

→ Standards for health data
  - Integrated data use all the way from facility/community
  - Key indicators and disaggregations
  - Harmonized analysis

→ WHO Integrated Data Platform (WIDP)
  - Support countries for data collection and use
  - Global reporting from countries with trends and impact monitoring
  - Validation/verification of disease elimination
Complementary use of DHIS2 by WHO

- Strategic tool for several WHO programmes
  - Country system strengthening
  - Dissemination of standards and data quality practices
  - Monitoring key targets (SDGs, GPW13…)

→ Standards for health data
  - Integrated data use all the way from facility/community
  - Key indicators and disaggregations
  - Harmonized analysis

→ WHO Integrated Data Platform (WIDP)
  - Support countries for data collection and use
  - Global reporting from countries with trends and impact monitoring
  - Validation/verification of disease elimination

Presentation on Thursday
What is the WHO Integrated Data Platform (WIDP)?

Coordination of people and processes across WHO departments & levels

STANDARDS

WHO Integrated Data Platform

dhis2

Infrastructure
What is the WHO Integrated Data Platform (WIDP)?
DB SERVER PERFORMANCE CHALLENGES
WIDP DB server performance challenges

1 - Architecture challenges for WIDP
WIDP DB server performance challenges

1 - Architecture challenges for WIDP
WIDP DB server performance challenges

1 - Architecture challenges for WIDP
WIDP DB server performance challenges

1 - Architecture challenges for WIDP
and set the following properties:

- `max_connections = 200`

Determines maximum number of connections which PostgreSQL will allow.

- `shared_buffers = 3200MB`

Determines how much memory should be allocated exclusively for PostgreSQL caching. This setting controls the size of the kernel shared memory which should be reserved for PostgreSQL. Should be set to around 40% of total memory dedicated for PostgreSQL.
WIDP DB server performance challenges

2 - Tuning PostgreSQL
WIDP DB server performance challenges

2 - Tuning PostgreSQL
WIDP DB server performance challenges

2 - Tuning PostgreSQL
WIDP DB server performance challenges

3 - DB optimization

- Postgresqltuner: script to detect suboptimal configurations and suggest changes
- pgbench: script to check current postgresql capabilities (tps)
- Does not eliminate the need of a first analysis to customize postgresql DB to application needs, but from an initial configuration can optimize
- With direct access to a user with privileges in the system or SSH connection with private key it can go even further and suggest OS memory, users, connections, indexes,
WIDP DB server performance challenges

3 - DB optimization

postgresqltuner.pl version 0.0.8
Connecting to /var/run/postgresql:5432 database testdb with user postgres...
[OK] User used for report have super rights
==== OS information ====
[INFO] OS: Debian GNU/Linux 8 \n \n
[INFO] OS total memory: 15.62 GB
[OK] vm.overcommit_memory is good : no memory overcommitment
[INFO] Running on physical machine
[INFO] Currently used I/O scheduler(s): cfq
----- General instance informations -----
...... Version ......
[WARN] You are using version 9.4.8 which is not the latest version
...... Uptime ......
[INFO] Service uptime : 101d 21h 53m 03s
...... Databases ......
[INFO] Database count (except templates): 2
[INFO] Database list (except templates): postgres testdb
...... Extensions ......
[INFO] Number of activated extensions : 1
[INFO] Activated extensions : plpgsql
[WARN] Extensions pg_stat_statements is disabled
...... Users ......
[OK] No user account will expire in less than 7 days
[OK] No user with password-username
[OK] Password encryption is enabled
...... Connection information ......
[INFO] max_connections: 100
[INFO] current used connections: 6 (6.00%)
[INFO] 3 are reserved for super user (3.00%)
[INFO] Average connection age : 1d 11h 31m 18s
...... Memory usage ......

> info: [https://github.com/jfcoz/postgresqltuner](https://github.com/jfcoz/postgresqltuner)
WIDP DB server performance challenges
3 - DB optimization

- Things we used postgresqltuner to:
  - Detect that we needed to disable the overcommit memory feature to avoid postgresql to be killed by the OS
  - Identify that we needed to take advantage from SSD specific parameter from postgresql
  - Check that the work_mem and rest of memory-related parameters were properly configured
4 - End-User performance tests

- **Apache JMeter:**
  - Open source tool to test Web Applications
  - Quite useful to get a sense of the user experience and some test scenarios depending on the values assigned to certain configuration parameters
  - We used it to check the impact of different DB parameters in some control scenarios like:
    - Bulk events import
    - Bulk aggregated data import
    - Analytics run
    - Dashboards rendering
  - We used avg response delay and % of failure as the variables to do the assessment
  - We incremented concurrent threads to represent users increment
WIDP DB server performance challenges

4 - End-User performance tests

+info: https://jmeter.apache.org/
WIDP DB server performance challenges

4 - End-User performance tests

- +info: https://jmeter.apache.org/
### WIDP DB server performance challenges

#### 2 - Tuning PostgreSQL

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>100</td>
<td>200</td>
<td>125</td>
<td>200</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>1024MB (131072 * 8kB)</td>
<td>3200MB (2000MB)</td>
<td>2000MB (4000MB)</td>
<td>2000MB (4000MB)</td>
</tr>
<tr>
<td>work_mem</td>
<td>16MB (16336 * 1kB)</td>
<td>2MB (2MB)</td>
<td>16MB (2MB)</td>
<td>2MB (2MB)</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>256MB (262144 * 1kB)</td>
<td>512MB (200MB)</td>
<td>200MB (200MB)</td>
<td>200MB (200MB)</td>
</tr>
<tr>
<td>effective_cache_size</td>
<td>4096MB (524288 * 1kB)</td>
<td>4096MB (200MB)</td>
<td>200MB (4000MB)</td>
<td>4000MB (4000MB)</td>
</tr>
<tr>
<td>checkpoint_completion_target</td>
<td>0.5</td>
<td>0.0</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>synchronous_commit</td>
<td>on</td>
<td>off</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>wal_writer_delay</td>
<td>200ms (200 * 1ms)</td>
<td>10000ms (10000ms)</td>
<td>10000ms (10000ms)</td>
<td>10000ms (10000ms)</td>
</tr>
<tr>
<td>random_page_cost</td>
<td>4</td>
<td>1.1</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>autovacuum_max_workers</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>checkpoint_timeout</td>
<td>5min (300 * 1s)</td>
<td>5min (300 * 1s)</td>
<td>10min (300 * 1s)</td>
<td>10min (300 * 1s)</td>
</tr>
<tr>
<td>wal_buffers</td>
<td>8MB (8MB)</td>
<td>16MB (2048 * 8kB)</td>
<td>16MB (16MB)</td>
<td>16MB (16MB)</td>
</tr>
</tbody>
</table>

#### TEST

<table>
<thead>
<tr>
<th></th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 users in 16 seconds - 2 users/sec</td>
<td>71,282s</td>
<td>-</td>
<td>62,398s</td>
<td>-</td>
<td>74,345s</td>
<td>100%</td>
<td>60,138</td>
<td></td>
</tr>
<tr>
<td>60 users in 16 seconds - 4 users/sec</td>
<td>126,464s</td>
<td>-</td>
<td>126,464s</td>
<td>-</td>
<td>125,464s</td>
<td>100%</td>
<td>112,067</td>
<td></td>
</tr>
<tr>
<td>120 users in 16 seconds - 8 users/sec</td>
<td>246,915</td>
<td>-</td>
<td>235,418s</td>
<td>-</td>
<td>219,556s</td>
<td>Some failing</td>
<td>222,237</td>
<td></td>
</tr>
<tr>
<td>180 users in 16 seconds - 12 users/sec</td>
<td>372,829</td>
<td>Only one call failing</td>
<td>337,439s</td>
<td>Some failing</td>
<td>326,422s</td>
<td>Some failing</td>
<td>317,229</td>
<td></td>
</tr>
<tr>
<td>240 users in 16 seconds - 16 users/sec</td>
<td>333,299</td>
<td>Some failing</td>
<td>283,255s</td>
<td>Some failing</td>
<td>297,802s</td>
<td>Almost all failing</td>
<td>296,770</td>
<td></td>
</tr>
</tbody>
</table>

#### Aggregated data import

<table>
<thead>
<tr>
<th></th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 user in 1 second</td>
<td>56,109</td>
<td>100%</td>
<td>41,543</td>
<td>100%</td>
<td>39,034</td>
<td>100%</td>
<td>42,172</td>
<td>100%</td>
</tr>
<tr>
<td>2 users in 1 second</td>
<td>67,710</td>
<td>100%</td>
<td>43,517</td>
<td>100%</td>
<td>42,191</td>
<td>100%</td>
<td>54,548</td>
<td>75%</td>
</tr>
<tr>
<td>3 users in 1 second</td>
<td>71,156</td>
<td>100%</td>
<td>52,004</td>
<td>100%</td>
<td>53,046</td>
<td>75%</td>
<td>51,074</td>
<td>75%</td>
</tr>
<tr>
<td>4 users in 1 second</td>
<td>73,002</td>
<td>100%</td>
<td>58,765</td>
<td>100%</td>
<td>57,131</td>
<td>75%</td>
<td>57,363</td>
<td>75%</td>
</tr>
<tr>
<td>5 users in 1 second</td>
<td>62,947</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6 users in 1 second</td>
<td>71,036</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7 users in 1 second</td>
<td>78,007</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8 users in 1 second</td>
<td>91,197</td>
<td>3 calls fail,</td>
<td>71,333</td>
<td>100%</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>9 users in 1 second</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>10 users in 1 second</td>
<td>93,975</td>
<td>1 call fail,</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Individual data import

<table>
<thead>
<tr>
<th></th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
<th>Results</th>
<th># Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 user in 1 second</td>
<td>31,001</td>
<td>100%</td>
<td>27,404</td>
<td>100%</td>
<td>34,172</td>
<td>100%</td>
</tr>
</tbody>
</table>
### WIDP DB server performance challenges

#### 2 - Tuning PostgreSQL

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>max_connections</td>
<td>100</td>
<td>320</td>
<td>200</td>
</tr>
<tr>
<td>shared_buffers</td>
<td>1024MB (131072 * 8 kB)</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>work_mem</td>
<td>16MB (16384 * 1 kB)</td>
<td>20MB</td>
<td>16MB</td>
</tr>
<tr>
<td>maintenance_work_mem</td>
<td>256MB (262144 * 1 kB)</td>
<td>512MB</td>
<td>200MB</td>
</tr>
<tr>
<td>effective_cache_size</td>
<td>4096MB (524288 * 1 kB)</td>
<td>4000MB</td>
<td>3200</td>
</tr>
<tr>
<td>checkpoint_completion_target</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>synchronous_commit</td>
<td>off</td>
<td>off</td>
<td>off</td>
</tr>
<tr>
<td>wal_writer_delay</td>
<td>200ms (200 * 1 ms)</td>
<td>10000ms</td>
<td>10000ms</td>
</tr>
<tr>
<td>random_page_cost</td>
<td>4</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>auto vacuum max_workers</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>checkpoint_timeout</td>
<td>5min (300 * 1 s)</td>
<td>5min (300 * 1 s)</td>
<td>10min</td>
</tr>
<tr>
<td>wal_buffers</td>
<td>-1 (X * 8 kB)</td>
<td>16MB</td>
<td>16MB</td>
</tr>
</tbody>
</table>

- Individual data import
  - 1 user in 1 second: 31,001
  - 2 users in 1 second: 39,067
  - 3 users in 1 second: 43,146
  - 4 users in 1 second: 49,625
  - 5 users in 1 second: 54,754

- pgbench
  - tps including connections establishing: 350,768741
  - tps excluding connections establishing: 437,081043

- tps including connections establishing
  - 30 clients: 870,693810
  - 60 clients: 1018,124818

- tps excluding connections establishing
  - 60 clients: 896,363187

- 90 clients
  - 120 clients
    - tps including connections establishing: 635,097630
    - tps excluding connections establishing: 909,538788

- 120 clients
  - 0 time updated: 00:08:20
  - 0 time updated: 00:06:29.531

- Table updated: 00:05:23.736

---

World Health Organization

DHIS2 Annual Conference | Oslo, June 18th 2019
WIDP DB server performance challenges

5 - DB connection pool

- DB connection pool:
  - Jira issue: [https://jira.dhis2.org/browse/DHIS2-5691](https://jira.dhis2.org/browse/DHIS2-5691)
  - c3p0 software. Used by DHIS2
  - Optimization:
    - Leave 30% of max connections for connections made outside the pool manager
    - Configure max_connections in dhis.conf
    - Configure the rest of the pool properties in a file called c3p0.properties in WEB-INF/classes inside the webapps folder of the DHIS2 war already deployed
    - Main idea, maximize the usage of the DB connections configured in the postgresql DB
    - Important: Postgresql servers shared by multiple instances need to configure their pools so that they can coexist
      - +info: [https://www.mchange.com/projects/c3p0/](https://www.mchange.com/projects/c3p0/)
      - +info: source code links [1](#) & [2](#)
Takeaways for DB server performance

● It would be desirable to get official UiO recommendations on:
  – Best practices on postgres/DHIS2 architecture installation
  – Postgresql parameters tuning (for multiple-instances or purpose-oriented)
  – Complete DHIS2 benchmark to understand the limits of the system

● Is there a good explanation on how DHIS2 is managing database connections?

● We would like that the community opens the discussion and share experiences
  https://community.dhis2.org/t/database-dhis2-benchmark/36250
METADATA PACKAGING & SYNC CHALLENGES
WIDP: Metadata packaging & sync challenges

- Multiple instances, different users, different purposes, a single metadata
WIDP: Metadata packaging & sync challenges

- Packaging Tools: Advanced Metadata Export
  https://github.com/WISCENTD-UPC/Advanced-Metadata-Export
WIDP: Metadata packaging & sync challenges

- Packages repositories:
  - WHO Country support packages
    - Check the presentation on Sched: [Implementing the aggregate WHO metadata packages. Lessons learned](#)
  - WIDP packages:
    - Goal → Store comparable json files with version tracking
    - Git repository
    - Now in process of standardization
WIDP: Metadata packaging & sync challenges

- **WIDP packages:**

  WHO metadata packages

  - 124 commits
  - 4 branches
  - 0 releases
  - 2 contributors
  - GPL-3.0

  This branch is 123 commits ahead, 1 commit behind master.

  **AA_common**
  - 1.0.3 - New userRole Individual Data Use and Authority added to Dashboard
  - Date: 20 days ago

  **ETA**
  - update eta version
  - Date: 2 days ago

  **HFP**
  - Merge branch '2.30' of github.com/fysfjord/WHO-packages into 2.30
  - Date: 9 days ago

  **HWF**
  - Implement changes in Other indicators in Module 1 tab
  - Date: 6 days ago

  **NTD**
  - NTD / Leish / Stock / updated BGD and NPL HFs
  - Date: 4 days ago

  **ZZ_common**
  - common divided into AA_common and ZZ_common for files that have to be...
  - Date: 22 days ago

  **LICENSE**
  - Initial commit
  - Date: 3 months ago

  **README.md**
  - Initial commit
  - Date: 3 months ago
WIDP: Metadata packaging & sync challenges

● Major challenges
  – some actions are considered as modifications should not be considered as modifications
  – how to have different configurations of same metadata
  – missing dependencies map

● Minor challenges
  – metadata exports does not have a specific order
  – several incoherences importing metadata
    • maps, indicatorGroups, (indicators & mapViews have to be imported first)
  – translations getting duplicated in each modification
WIDP: Metadata packaging & sync challenges

- Command-line tools
  - jcat → python CL utility to work with json files
    - Main features:
      - json files join
      - Ordering per level (for plain text packages comparison)
      - Replacing UIDs and strings
      - Filtering to extract structures
      - std output can be piped and chained
  - change_user_groups → python script to alter sharing settings on jsons
    - Main features:
      - Change user accesses and user group accesses to certain elements of a package
      - Remove OU references
      - Replace

+info: https://github.com/EyeSeeTea/ESTools
WIDP: Metadata packaging & sync challenges

Sync:
- Instances cloning:
  - python script with actions and post-clone actions that can be customized on a per-instance basis.
  - from a subset of embedded actions or shell scripts, SQL scripts and API modifications
- Weekly:
  PROD → TRAINING (no data)
  PROD → PREPROD
  DEV → DEV-CONT (build upgrade)
- Examples of post-actions:
  - Bring last war file from builds repository
  - Add run analytics user role to some users
  - Import a new version of a metadata package
WIDP: Metadata packaging & sync challenges

- **Sync:**
  - Instances cloning:
WIDP: Metadata packaging & sync challenges

- Sync:
  - Instances cloning:
WIDP: Metadata packaging & sync challenges

- Metadata repository and instance synchronization
  - Different users are changing different parts of metadata in different instances (e.g., PROD and DEV)
    - indicators, indicator groups, dashboards,...
  - Keep metadata of all instances in synch
    - Pushing metadata from DEV to PREPROD and PROD - on demand
    - Pushing new/changed metadata from PROD back to DEV - periodically
  - Keeping a centralized repository of master (production) metadata available.
WIDP: Metadata packaging & sync challenges

- User management across instances
  - Advanced-Export App for “propagating” new users to the different WIDP instances
  - **User Extended App** for users management and sync
  - Cloning script to activate/deactivate/modify users
Takeaways for metadata packaging & sync

- A complex, but common scenario is to sync only parts of the metadata
- Lack of appropriate dependency map for the different metadata objects
- Metadata versioning would make our life easier
- Metadata audit logic should be split
  - Sharing Settings or OU assignment shouldn’t probably be recorded as a metadata update
- Non IT admins don’t have interfaces in core DHIS2 to manage packaging and sync individual pieces of metadata
- We would like the community to open discussion and share experiences [https://community.dhis2.org/t/metadata-packaging-sync/36253](https://community.dhis2.org/t/metadata-packaging-sync/36253)
- We seek for the creation of a community repository to share tools and scripts
WIDP OU TREE: SYNC CHALLENGES
WIDP OU Tree: Challenges

- Having a global and shared DHIS2 instance like WIDP raises additional challenges for the management and synchronization of the OU.

- Same OU tree shared by programs working in different parts of the world -
  - Require coordination and prioritisation
  - Increase the size of the OU tree

- Different programs may be sharing the same parts of the tree
  - Require consensus on the org unit subtree per territory

- Different programs work at different levels
  - Global, national, 1/2/3 subnational levels, villages, health facilities

- Results in having currently ~110 000 Org Units (and growing)

- Constantly changing subnational levels of countries (e.g., Ghana, France)

- Correct and consistent org unit tree is essential for proper data management in DHIS2
WIDP OU Tree: Request procedure

- Issue tracking system (Redmine) to support for the process
- Training of WIDP users
WIDP OU Tree: External sources

- Variety of sources for countries’ subnational levels
  - **Org unit tree provided by the country** if the country has a national DHIS2 instance.
    All information required, but at least name, code, uid and coordinates, if available
  - UN Office for the Coordination of Humanitarian Affairs (OCHA) - https://www.unocha.org/
  - Polio geodatabase (http://polioboundaries-who.hub.arcgis.com/) - outdated
  - Document (e.g. Excel and/or shapefiles) from an official source in the country: programme focal point, HMIS team, national institute of statistics...
  - Document (e.g. Excel and/or shapefiles) from another source after validation by an official from the country
  - Document (e.g. Excel and/or shapefiles) from any other source

- Considered with priorities
WIDP OU Tree: Extraction scripts

- Variety of formats from different sources require specific processing
  - JSON, XML, Excel
  - Different level of details (available attributes)
  - Different subnational level depth (region, province, municipality…)

- Conversion scripts to import org units to WIDP
  - Format conversion
  - Attribute mapping
  - parent-child relationships

- +info:
  - OCHA: [https://github.com/WISCENTD-UPC/OCHA-2-DHIS2](https://github.com/WISCENTD-UPC/OCHA-2-DHIS2)
  - Polio GeoDB: [https://github.com/WISCENTD-UPC/polioDB-2-DHIS2](https://github.com/WISCENTD-UPC/polioDB-2-DHIS2)
WIDP OU Tree: Use cases

- Uganda
  - Recollected from the country’s DHIS2 instance for Malaria
  - Integrated into WIDP org unit tree shared with other programs
  - Facility lists provided per program
- Italy - Chagas
  - Italian institute of statistics - https://www.istat.it
  - Excel file to the municipality level
  - Focal point to validate & provide health facility list linked with the tree
WIDP OU Tree: Spatio-temporal support

- Country subnational levels structure and boundaries **do change** over time
- Spatio-temporal support needed
- DHIS2 should allow users to analyze data through geographies with temporal reference
  - Org units with temporal validity
Takeaways for OU Tree sync

- Establish a protocol and best practices on collaborative/shared orgUnit Tree management
- Community-maintained list of OU sources (hosted by UiO)
- Community-maintained repository of source-specific conversion scripts (Polio GeoDB, OCHA,..)
- Spatio-temporal solution for org unit and boundaries evolution/versioning

DHIS2 community discussion: https://community.dhis2.org/t/shared-org-unit-tree-management/
CONCLUSIONS & IDEAS FOR THE FUTURE
WIDP: Conclusions & ideas for the future

Conclusions:

- Sync is one of the biggest issues in WIDP
- Deciding what to sync sometimes falls on a higher level role, but the sync process is still missing appropriate fancy interfaces
- We need ideas from the community to optimize DB performance
WIDP: Conclusions & ideas for the future

- Ideas for the future:
  - Advanced-Export App: Auto-packaging configurations
  - Instances cloning: Data anonymization
  - Trainings: Assess and migrate to UiO docker scripts
  - User sync: User migration sync feature for User-Extended App
  - OU Tree sync: OU Master list auto-pulled from all instances