AIP Leibniz-Institut für Astrophysik Potsdam

German Astrophysical GAVO Virtual Observatory

Data publication at AIP

Data sets, data curation, tools

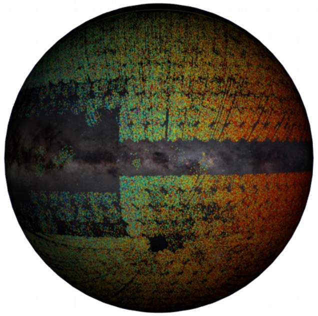
ASTERICS European Data Provider Forum

June 15, 2016, Heidelberg

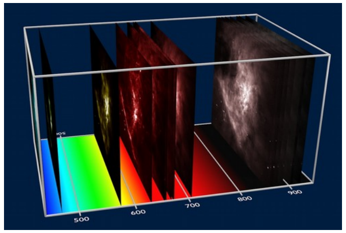
Kristin Riebe, AIP, GAVO

SPONSORED BY THE
Federal Ministry
of Education
and Research

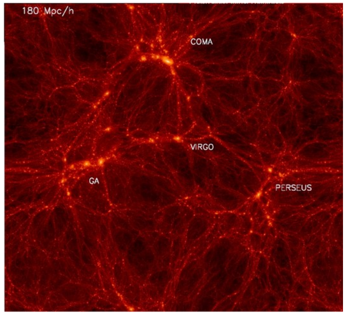
Example data at AIP



- Observations:
  - RAVE
    - radial velocities survey
      - catalogs of stellar properties, spectra
  - Plates archive:
    - archive of digitized plates from AIP, Hamburg, Bamberg, Tartu (Est)
      - images (scans of plates, log books and envelopes), catalogs of identified objects
  - Gaia data
    - so far only simulated data (GUMS10, GOG11, GDR0)
  - MUSE
    - 3D spectroscopy (data cubes)

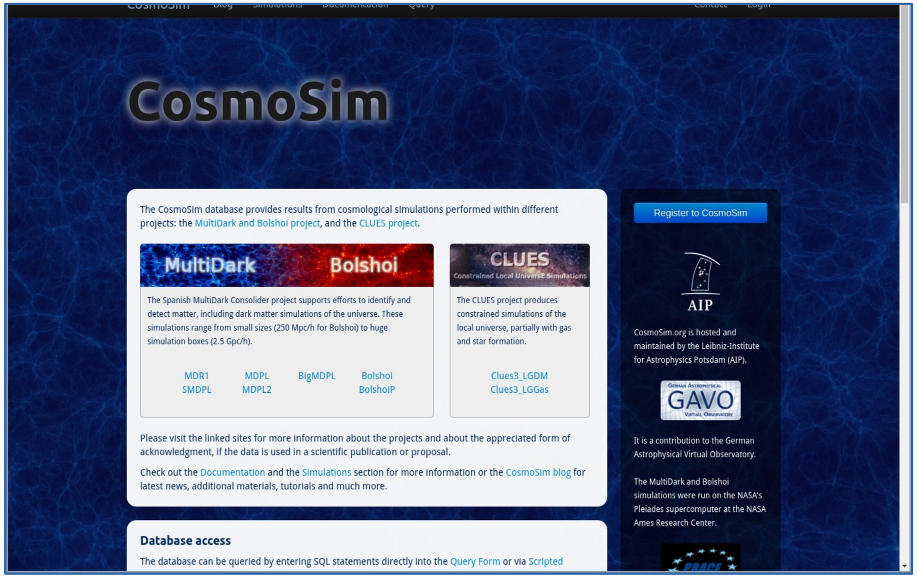


- Simulations:
  - magnetohydrodynamical simulations
  - cosmological simulations
      - raw snapshots, halo catalogs, merger trees, galaxy catalogs



Example: CosmoSim Database

- computer simulations of the evolution of the universe
- 9 different simulations with different resolution, box size
- in total currently about 30 TB public data, ~ 10 TB in preparation
- sometimes it's a long way to publish the data ...



The CosmoSim database provides results from cosmological simulations performed within different projects: the MultiDark and Bolshoi project, and the CLUES project.

The Spanish MultiDark Consolider project supports efforts to identify and detect matter, including dark matter simulations of the universe. These simulations range from small sizes (250 Mpc/h for Bolshoi) to huge simulation boxes (2.5 Gpc/h).

MDR1 SMDPL MDPL MDPL2 BigMDPL Bolshoi BolshoiP

The CLUES project produces constrained simulations of the local universe, partially with gas and star formation.

Clues3_LGDM Clues3_LGGas

Please visit the linked sites for more information about the projects and about the appreciated form of acknowledgment, if the data is used in a scientific publication or proposal.

Check out the Documentation and the Simulations section for more information or the CosmoSim blog for latest news, additional materials, tutorials and much more.

CosmoSim.org is hosted and maintained by the Leibniz-Institute for Astrophysics Potsdam (AIP).

It is a contribution to the German Astrophysical Virtual Observatory.

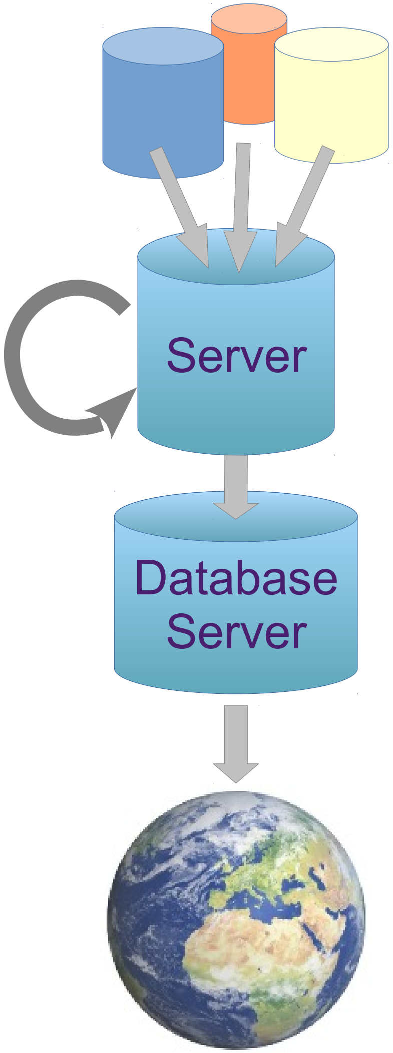
The MultiDark and Bolshoi simulations were run on the NASA's Pleiades supercomputer at the NASA Ames Research Center.

Database access

The database can be queried by entering SQL statements directly into the Query Form or via Scripted

Example: Data flow for ComoSim

- Extract:
  - Cosmologists produce data worldwide, copy them to a central server at AIP
- Transform:
  - We check data and reading routines,
    data curation: corrections, additions, convert format
- Load:
  - Ingest data into database
- Check and test:
  - Check the data for completeness, consistency
  - Create Peano-Hilbert keys *(Spatial3D, T. Budavari, G. Lemson)*
  - Create DB indexes
- Publish:
  - Using Daiquiri framework
  - Write/update documentation; update admin tables of the database
  - Inform users (blog)



Data curation

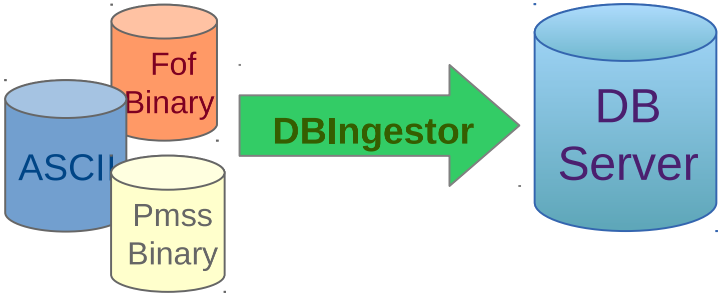
- Check completeness of data sets
  - no missing snapshots, corrupted files
  - restarted simulations => some snapshots may be duplicated
- Create homogeneous data sets, common (standard) formats
  - different names for the same physical properties (e.g. spheroidMassGas vs. Mgas_bulge, Mvirs vs. Mass)
  - different coordinate systems (e.g. physical/comoving coordinates)
  - different units
  - different counts for snapshot numbers
- Add identifiers, grid indexes etc. for faster queries & for representing relations in the database
- Cross-link data with other catalogues (DB indexes)
- unsufficiently documented data structures require lots of research and communication with data creators

Wishlist to data creators

- documentation
  - provide good and extensive documentation for their data and also for their data format (not just “my code is my documentation”)
- write/read routines, architecture information
  - provide a write and read routine for their data (along with architecture dependent information like little/big endian, 32/64-bit, any compiler setting regarding byte alignment)
- HDF5 format for binary data
  - provide binary data in HDF5 format (e.g. Galacticus: 2000 pages of documentation (pdf), HDF5-format => only need to know the data path, types are given automatically)

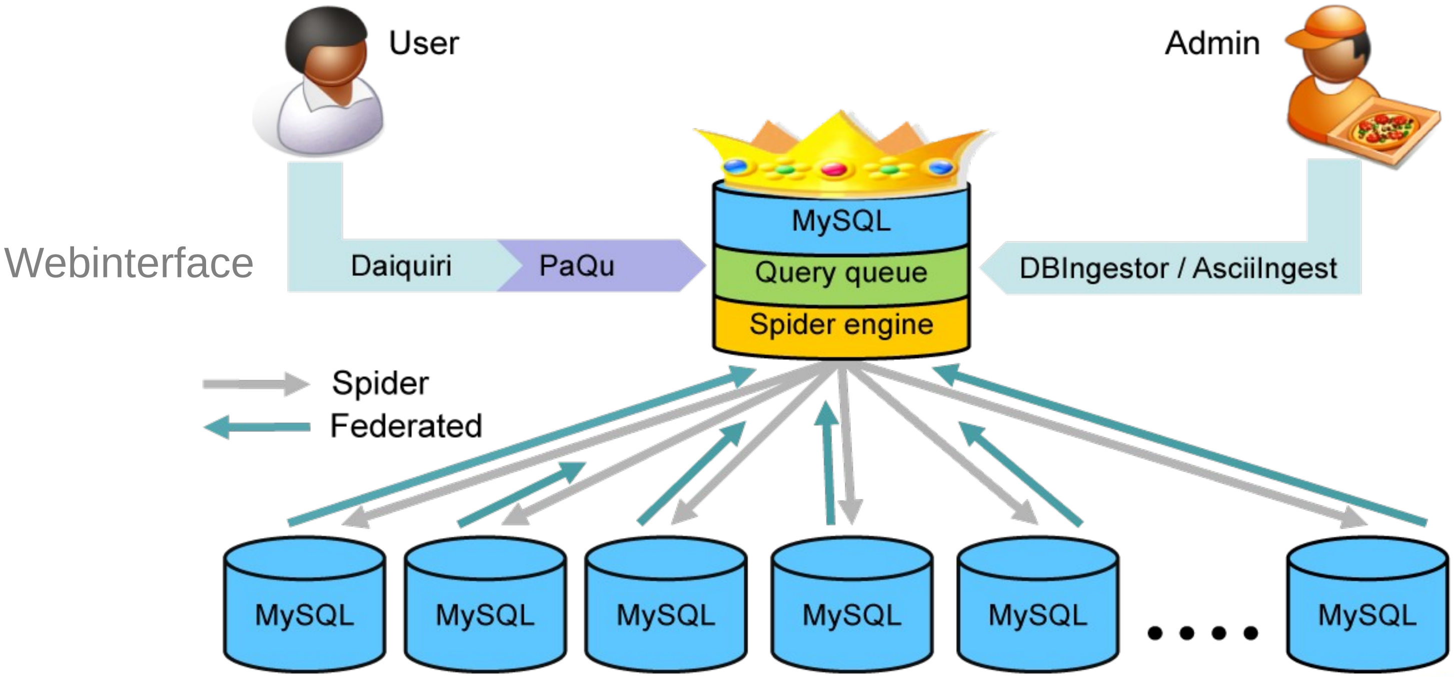
Data upload: DBIngestor

- https://github.com/aipescience/DBIngestor
- adjustable to any database server
- easy to write own file readers
  - e.g. AsciiIngest, FofIngest, PmssIngest, GalacticusIngest
- apply converters during ingestion
  - e.g. unit conversion,
    type conversion (int/real),
    adding identifiers, grid indexes
- apply asserters (not nan, inf, null etc.)
  - => transform and upload in one go
  - => easier to preserve the workflow for later reference



Database technology

- MariaDB + SpiderEdngine
  - use MyISAM engine of MySQL/MariaDB
  - Spider engine (Kentoku Shiba) for distributed queries available
  - => data distributed over 10 nodes, queries much faster!



User

Admin

Webinterface

Daiquiri

PaQu

MySQL

Query queue

Spider engine

DBIngestor / AsciiIngest

Spider

Federated

MySQL

MySQL

MySQL

MySQL

MySQL

....

MySQL

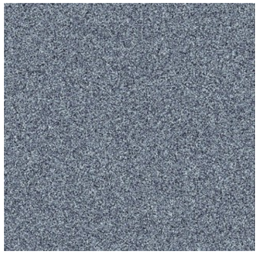
PaQu + QueryQueue

- PaQu (https://github.com/adrpar/paqu):
  - reformulates queries, based on Shard-Query
  - e.g.: aggregate function count
    = count on each node + sum on head node

- QueryQueue (https://github.com/adrpar/mysql_query_queue):
  - allow asynchronous job submission
  - plugin for MySQL, supports priorities
  - control number of executing jobs on server
  - jobs stored in user tables for later retrieval

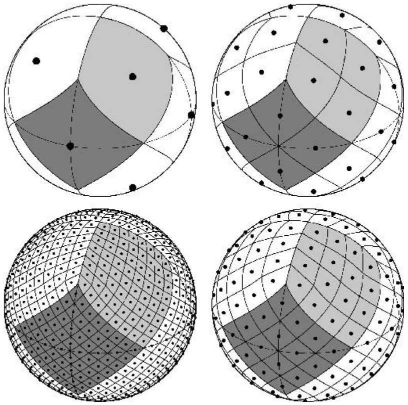
Tools: MySQL

- mysql_sprng (https://github.com/adrpar/mysql_sprng)
  - based on SPRNG library (www.sprng.org)
  - implements randon number generators
  - better random sampling than built-in function



- mysql_sphere (https://github.com/aipescience/mysql_sphere)
  - port of pgsphere to mysql
  - no indexing yet, contributions welcome!

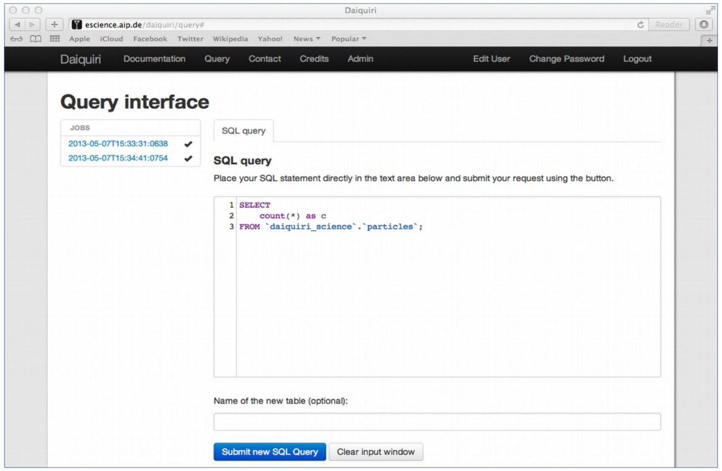
- mysql_dumpvo (https://github.com/adrpar/mysqldump-vo)
  - exports VO-tables directly from MySQL/MariaDB
- mysql_healpix (https://github.com/aipescience/mysql_healpix)
  - function for calculating healpix indexes



- queryparser (https://github.com/aipescience/queryparser)
  - using ANTLR4
  - parsing MySQL and ADQL select statements
  - translation of ADQL geometry functions to mysql_sphere functions

Daiquiri web service

- https://github.com/aipescience/daiquiri
- SQL query interface for querying tabular data
- UWS for non-interactive access:
  - UWS = universal worker service, for asynchronous, job-oriented web services
  - user creates job, job waits in queue until executed
  - results not returned immediately
  - UWS was recently updated to version 1.1



Query interface

JOBS

2013-05-07T15:33:31:0638 ✓
2013-05-07T15:34:41:0754 ✓

SQL query

Place your SQL statement directly in the text area below and submit your request using the button.

```
SELECT
    count(*) as c
FROM `daiquiri_science`.`particles`;
```

Name of the new table (optional):

uws-client (https://github.com/aipescience/uws-client)

- python command line tool for querying VO TAP and UWS services from the command line
  - create job
  - update parameters
  - submit job
  - check execution phase
  - download result
  - remove job
  - abort job
- supports new version UWS 1.1!

uws-validator (https://github.com/kristinriebe/uws-validator)

- for validating UWS-services, including 1.1 features
- can be used for async-endpoints for TAP-services as well
- using behave python module for formulating functional test cases in “human language” (Gherkin syntax)
  - Example test definition:

    Scenario: Ensure user can access UWS endpoint

    When I make a GET request to base URL

    Then the response status should be "200"
  - Each “phrase” is a step that needs to be implemented as a function
- put parameters like basic url to UWS-endpoint, authentication details and test queries into a userconfig-file (json)
-

uws-validator

- Run from command line e.g. like this:
  - Check basic access and authentication:
    - behave -D configfile="userconfig-gaia.json" features/account.feature
  - Test job list, creating veryshort job:
    - behave [...] --tags=basics
  - For UWS 1.0, exclude all 1.1 tests:
    - behave [...] --tags=-uws1_1
  - Do fast tests first (exclude slow and neverending jobs):
    - behave [...] --tags=-slow –tags=-neverending
- still some test cases are quite strict, will fail, if jobs stay in queue for too long (> a few seconds), server returns immediately for WAIT

Summary

- AIP data sets:
  - publishing different data types, but mainly catalogues
- Data curation:
  - can be a pain, especially if data creators are ignorant or uncommunicative
  - necessary to provide consistent data to the user
- Ingestion tools:
  - DBIngestor + readers
- MySQL:
  - using MySQL as backend server
  - Spider Engine for distributed database setup for large data amounts
  - number of plugins for MySQL
- UWS:
  - Daiquiri web framework updated to latest UWS 1.1 version
  - uws-client
  - uws-validator
- check it all out on GitHub:
  - https://github.com/aipescience
  - https://github.com/adrpar
  - https://github.com/kristinriebe