

## Virtual Telescope

---

### 1. Scenario Overview

---

#### 1.1 Background and Purpose

In "Theory VO" (VO=Virtual Observatory !) parlance, a "Virtual Telescope" is a piece of software that mimics an astronomical observatory (telescope/instrument/satellite/spectrograph etc etc) which can "observe" the result of a simulation and produce a synthetic observation that can be directly compared to the output of the instrument that is being simulated. The idea is that the software includes lots of instrumental details which allow a user insight in the biases introduced by such effects in real observations.

In this scenario we propose to make available a large set of simulation results of galaxy clusters together with a virtual telescope that simulates an X-Ray satellite. Users will be enabled to search a database for simulations of interest to them, which are stored somewhere on the data grid. They can also find a virtual telescope (grid) service that can be applied to the selected simulation result. They can then execute the observation somewhere on the grid where sufficient compute and data storage resources are available to them. The results will in general be relatively small images that should be returned to the user.

This use case may be subdivided into (a) searching a database of simulated clusters according to some criteria, (b) finding or defining a VT service, and (c) applying a VT to selected clusters to generate images.

#### 1.2 More information

- For an example of such a service see <http://www.g-vo.org/hydrosims/>?. This application does not allow construction not of real images, but the idea is the same. A real virtual telescope will take more resources to run in a reasonable time than these.
- For literature about such a virtual telescope see, for example: Gardini et al., "Simulating Chandra Observations of Galaxy Clusters" Monthly Notices of the Royal Astronomical Society, Volume 351, Issue 2, pp. 505-514

---

### 2. Current Scenario description

---

Currently users go to a web page of GAVO where they can choose from 2 simulations, then for each from about 90 results. This (meta-)data is obtained by querying a meta-data repository. They can then choose one of two visualisation services, set parameters and execute. The result can be downloaded via a hyperlink.

#### 2.1 Environment

##### 2.1.1 Hardware

Describe the hardware resources that are currently used.

- Processing

- . single PC

- Storage

- . local disk only

- Network

- Only as much as required to download the result which in general is at most a 512x512 image.

- Describe special hardware or other hardware resources that are relevant for the scenario.

NA

## 2.1.2 Software

- Describe used software such as operating system, software libraries, e.g. HDF5-plugin for GridFTP, ...
  - . list non-standard software with min. version

Linux

- What programming language is used and what compiler/linker version is required?

- F90/C/C++ is required for the code. The F90 compiler must be compatible with linux. (gcc will compile C and C++, but not (yet) F90.)

- How is the program deployed?
  - . source code, pre-compiled binaries, ...

- Source code

- How is the program compiled?
  - . Ant, make, ...

- make

- State the program license and any commercial 3rd party licenses.

- Both the visualisers (Splotch and SMAC) are or will be available under GPL. (SMAC uses HEALPix, which is also GPL.)

## 2.2 User Interaction

### 2.2.1 Initiation

- Describe how the program is started and any steps needed before the actual initiation.

Via web portal, described above.

- compilation (cf. Section 2.1.2),  
retrieve/copy/generate data/files (cf. Section 2.3.2),  
parameters (cf. Section 2.3.1)

By hand as part of deployment of web app.

- Where is the program executed?  
. single PC
- How is the program initiated?  
. web portal

### 2.2.2 Monitoring/Steering/Visualization during the run-time of the program

Definitions:

Monitoring - Information retrieval regarding the state of the program. For example, "Program is running" or more application specific information such as "Current simulation iteration is 42".

Steering - Remote alteration of the programs state. For example, program stop or application specific information like "at iteration 42, set parameter x = 78".

Visualization - Remote access of the application data needed for visualization of, for example, a simulation.

- What type of data is produced by the program during run-time used for monitoring/steering/visualization?  
. log-file, available for download after execution.
- What methods/tools exists for accessing data produced by the program during run-time?  
. download of log file produced by code
- Does your application support any standard for monitoring/steering?

no

- Describe any security measures related to program access for monitoring/steering/visualization.

Parameters are restricted by HTML dropdowns to prevent parameter values that would jeopardize the process. Then there is a timeout of 1-2 min per execution.

- Who can access the running program OR run-time produced monitoring data?

GAVO admin

- From where can run-time produced monitoring data be accessed?  
. specific IP/netmask, anywhere, ...

NA

- How is the program termination detected?

Result web page.

- How much monitoring data and how often is monitoring data transferred during a program run (min/max/avg)?  
. X MB every time a buffer is full, X GB when the program finish, ...  
. all data is transferred, partial data is transferred, last N records are transferred, ...

NA

- Does your program generate metadata and stores this externally (e.g. in a catalog)?  
. when [start, end, periodically, specific events]?  
. where is the catalog?  
. how is it accessed [protocol/API, access controlled]?

No

- Who accesses this metadata? From where? Does your program access metadata generated by other programs?

NA

- How many executions/jobs must be monitored/steered in parallel? By how many users?

No data available as currently program was run as prototype only.

## 2.3 Input

### 2.3.1 Parameters

Describe the program parameters in detail.

### 2.3.2 Input data

- How is the input data prepared?

Captured on web page

turned into configuration file

The configuration file is placed in a unique directory for the web server session where the job is executed.

- Where is the input data stored? Describe all central and distributed locations.

. local file system

- Are file-names known in advance (before the program is started)?

. Are retrieved from a metadata catalogue which is used to populate the web pages

- Are data locations (directory, server, ...) known in advance?

. Through metadata catalogue.

- Describe the different ways data is accessed.
  - . POSIX read
- Non-file based data access (XML, database, ...) should include description of
- How much data is accessed at each run?
  - . number of files/data sets: min/avg/max, 1 file
  - . total-size: min/avg/max, 0(50Mb)
  - . retrieved-size: min/avg/max )(0.5Mb)
- Is it possible that a data set/file is accessed multiple times over a short period of time?
  - . For example by different "threads" of the program. Then, replicating and/or caching might be interesting.

Yes, could be if scenario is used to do multiple observations of same dataset from different angles.

- How many users are using the same data simultaneously?
  - Are these users geographically distributed?

It is a VO application so geographical distribution is likely.  
How many can not be predicted as of yet.

- Elaborate on the use of metadata related to input data.
  - . amount, limited, a few rows in a relational database per simulation output.
  - . how it is accessed,

Currently through Java code using the Hibernate Object-Relational mapping toolset.

- . security restrictions,
  - Currently none. Users have web access without requiring authentication as the execution times and result sets are limited.

- . metadata format [key/value ?],
  - Relational database queried by Java-based OR mapping tool Hibernate
  - . life-cycle: creation, usage time, used by multiple program runs, deletion, ...

When new datasets are added a special purpose Java program is run that maps these to the metadata repository. The underlying model is rather complex, hence the use of the OR mapping tool.

### 2.3.3 Additional Notes

Describe any additional information regarding the input data which has not yet been covered.

## 2.4 Output

This covers what data products are generated (INTERMEDIATE and FINAL results), where they are generated and how they are handled after the program finished (transferring data or removing it, ...).

### 2.4.1 Output data

- Where is the output data stored? Describe all centralized or distributed locations.
  - . local file system

- How is the output data structured?
  - . a number of files, one data, some logging and parameter files
  - . data formats: depending on visualisation service: logging and parameter files plain text, results: FITS and JPEG
- Describe what happens when the program finishes? How are the results used?
  - . remain at the output location until "clean-up" daemon throws them away after a certain time (24 hours),
  - . moved/copied/deleted: can be manually copied via hyperlink off result web page
- Describe the different ways data is created/changed.
  - . POSIX write
- Non-file based data access (XML, database, ...) should include description of
  - . name of the database management system,
  - . how the database is accessed (ODBC, JDBC, WWW interface, command line, ...),
  - . typical create patterns (bursty, continuous, ...),
  - . physical location/distribution (local, externally, ...),
  - . possibility to replicate the data through some mechanism,
  - . any security related restrictions when data is written,
  - . ...

Currently the metadata is accessed via a rather complex mechanism using Hibernate. This can likely be simplified and maybe should. The problem is that for users to make an informed decision on which datasets they want to visualise they need astronomical information. For this we started developing a data model, though eventually this will be taken up by the IVOA. The more physical data about storage etc can likely be taken up from some Data Grid standard.

- How much data is written by the program at each run?
  - . size: min/avg/max, < 1Mb
  - . number of files/data sets: min/avg/max about 3 (data+log+parameter)
- Describe the parameters which influence the amount of data and number of files/data sets generated.
 

The number of files per run is fixed. There are parameters specifying how large the image should be, this sets the size.
- Elaborate on the use of metadata related to output data.
  - . amount,
  - . how it is accessed,
  - . security restrictions,
  - . metadata format [key/value ?],
  - . life-cycle: creation, usage time, used by multiple program runs, deletion, ...

Currently all metadata about the contents of the results is stored in the parameter and log file.

The temporary location of the results is given as hyperlinks on the result web page.

#### 2.4.2 Additional Notes

Describe any additional information regarding the output data which has not yet been covered.

## 2.5 Information resources

Give a summary of each information resource that is accessed by the program. Include information about data input/output, locations, access methods (XQuery, SQL, ...), security related restrictions, search of metadata (exact key search i.e. "ABC", range queries i.e. "AB\*", ...)

Only information query done by users is browsing a web page which is filled in by a query to the metadata repository.

This repository is stored in a Postgres database and accessed via SQL generated by Hibernate from HQL (OQL-like Hibernate Query Language).

## 2.6 Data Stream Management

Definitions:

Data Stream - intermediate results can be processed by the subsequent processing module before the current module has processed the last element of the input.

- Can single operations be performed on any compute node or do they need special hardware or software?
- Are data exchanged between distributed parts of the application? does this happen at the beginning, during run-time or at the end?
- Are operations compute intensive?

NA

## 2.7 Resource Security and Access Restriction

Describe all security related information that considers access of resources. User based, Group based, by IP-address/netmask, certificates, nodes/resources within a private network, firewall restrictions, ...

Currently no security issues are taken into account explicitly.

## 2.8 Additional Information

Give additional information not covered by the sections above.

- How are workflow/pipeline steps interrelated to each other?

NA

- Is the application executed in several phases where each phase may have different resource requirements or may be executed at a different resource?

No.

- How long (avg) does the scenario execute (minutes, hours, days)?

< minute

- How often will the scenario be executed?

Hard to guess since the users are unknown.

- Are the executions time-critical?

No, unless limited by the patience of the user.

---

### 3. Future Scenario and AstroGrid-D Usage

---

Describe the future scenario and envisioned usage of AstroGrid-D as detailed as possible. It is not assumed that the questions can be answered as detailed as in Section 2. Focus on what is expected by the Grid environment and how this new functionality can be used.

Note, there is no special section to describe workflows/pipelines or details about a phased execution of a program. If your scenario is a workflow/pipeline OR your application is executed in several phases, describe EACH step/part covering the sections 3.1 - 3.5. In addition you must describe how these steps/parts are interrelated to each other (in Section 3.6).

#### 3.0 General goals

The goal is to extend this framework to

1. larger result sets
2. Larger input datasets
3. more complex visualisation tools, in particular the Chandra simulator (see the reference to Gardini et al under section 1.2).
4. parameter studies

Such as:

- use more compute resources,
- use other data resources,
- provide data to other users,
- completely new scenario,
- overcome deficiencies of current approach,
- ...

#### 3.2 Environment

- Are there any constraints due to your participation in other projects or international collaborations?
  - . specific Grid middleware, hardware, standards, virtual organizations

Eventually we want the users to be able to select simulations based on IVOA standard metadata models, metadata repository designs and query languages. These should be used in conjunction with the underlying grid infrastructure.

#### 3.3 User Interaction

- Which parts should be automated?
  - . resource selection,

It would be nice if the Grid could be used to do the calculations.

- . data transfer before initiation and after termination,

The input data may have to be moved to wherever the Grid is going to execute the job, results should be made available to the user in a transparent manner.

- . . . .

- Which user interface are you planning to use?

- . WS [SOAP], API, WWW portal, ...

portal

- Are you planning to use any standard for application monitoring/steering?

- . Do you want to use such standards in collaboration with the DGI or the other communities OR will you develop your own methods?

If available and applicable, yes.

- Aspects of a Portal / WWW based interface:

- . Which portal features are mandatory/optional (e.g. credential management, job management, job monitoring/steering, data transfer, ...)?

likely all of these.

- . How are user managed? Where is information about users defined / stored?

Currently not, should be done through registration through some central, GAVO based mechanism.

- . Which authentication/authorisation methods are needed ?

None, as long as traffic is not too high.

- . Do you want to access specific data services (web services, databases, etc.) via a portal?

Scenario is based on access through a portal.

- . Are there any existing programs, on which the user interface should be based OR which should be replaced by the portal?

If portal services exist for metadata repository browsing and querying they may be used instead of the current GAVO portal.

- . Should there be a central AstroGrid portal OR do you want to set up a portal server for each scenario/application ?

I can see a portlet per application. Ultimately the owners of the data and software must decide how/where they want to control the access.

- . Does the scenario require any special interfaces OR is it sufficient to use generic interfaces ?

No unusual functionality is required.

- Aspects of a generic Grid Application Programming API (GAT)
  - . Which GAT functionality would you like to make use of (eg. job submission, file handling, resource brokering, etc.) ?

Possibly all that are relevant.

- . What programming languages must be supported ? Which platforms ?

Fortran, C, C++, Java, Linux, Unix, Windows

- . Which Grid Middleware should be supported (Globus, Unicore, gLite, etc.) ?

No preference or restriction.

- . For specific GAT functionality, which protocols/packages/tools should be supported ?  
eg. for job management: clusters with PBS, SGE, Condor

We'll need batch processing. Condor might be good enough ? Globus ?

No preference or restriction to particular software.

### 3.4 Input

- Do you handle input data manually or do you need an automated management of data?

New input data will become available but will likely be registered and published manually.

### 3.5 Output

- Do you handle output data manually or do you need an automated management of data?

Would be good if results can be sent to user in automated fashion. Alternatively the user could get some "My Space" account on the Grid where the data goes to and can then download the ones that are desired.

### 3.6 Additional Information

- How long (avg) does the scenario execute (minutes, hours, days)? Do you aim at a specific speedup?

Run time on a single node may be about an hour for large simulations and realistic virtual telescope configurations.

- How often will the scenario be executed?

Likely often.

- Which restrictions of the current approach (as described in section 2) do you want to overcome?

Limitation on the service so that the more complex and computationally intensive realistic virtual telescopes can be run in reasonable time.

#### 4. Bigger Picture for the far future

Later

##### 4.1 Organization of Multiple Runs

Maintain a list of all simulations, to repeat simulations with a different binary, with different input data, to check if a program was already executed with a certain set of parameters/input data, ... .

##### 4.2 Handling relationships between data products

For example, store metadata on how a data product was generated (from which input data, by which program, with which parameters) and how it can be used by others.

##### 4.3 Constructing More Complex Runs

For example, combine existing single programs.