



# Distributed Database Access and Data Stream Management

## Requirements Specification and Architectural Design<sup>1</sup>

Deliverable	WG4-D1
Authors	Working Group 4
Editors	Tobias Scholl
Date	May 12, 2006
Document Version	1.0.0
Current Version	1.0.0
Previous Versions	0.1.12, 0.1.8, 0.1.0

### **A: Status of this Document**

Working draft for the requirements and design document of working group 4.

### **B: Reference to project plan**

First deliverable of working group *Distributed Database Access and Data Stream Management*.

---

<sup>1</sup>This work is part of the D-Grid initiative and is funded by the German Federal Ministry of Education and Research (BMBF).

**C: Abstract**

Database Management and Data Stream Management are components of the AstroGrid-D middleware. This document provides an overview of the functional requirements for these components. The requirements are derived from scientific use cases collected within the project. The initial architectural design is motivated by these requirements.

**D: Change History**

<b>Version</b>	<b>Date</b>	<b>Name</b>	<b>Brief summary</b>
0.1.0	27.02.2006	Tobias Scholl	Working Draft Creation.
0.1.8	25.03.2006	Tobias Scholl	Added a glossary and scope. Notes on DMC, FRINGE, and GAIA-DB. Adapted deliverable template.
0.1.12	03.04.2006	Tobias Scholl	Clarified relationship to deliverable on metadata management.
1.0.0	03.05.2006	Tobias Scholl	Release as deliverable.

E:

## Contents

<b>Abstract</b>	<b>2</b>
<b>Change History</b>	<b>3</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 Scope</b>	<b>5</b>
<b>3 Requirements</b>	<b>5</b>
3.1 Astrometric Matching (UC1) . . . . .	6
3.2 Clusterfinder (UC2) . . . . .	7
3.3 Millennium Simulation Queries (UC3) . . . . .	7
3.4 Integration of robotic telescopes (UC5, UC6) . . . . .	7
3.5 Planck Process Coordinator (UC7) . . . . .	9
<b>4 Architectural Design</b>	<b>9</b>
<b>5 Database Management Component</b>	<b>11</b>
<b>6 Data Stream Management Components</b>	<b>12</b>
6.1 Core Data Stream Management . . . . .	14
6.2 Query Optimization Service . . . . .	14
6.3 (Mobile) Operator Repository . . . . .	15
6.4 Interaction with other AstroGrid-D Components . . . . .	15
<b>Glossary</b>	<b>17</b>
<b>References</b>	<b>18</b>

## 1 Introduction

The participating institutes of the AstroGrid-D project collected use cases about their current scientific challenges. These use cases describe the concrete challenges and how the institutes currently approach them. Furthermore, it is described what functionality is expected from AstroGrid-D. This is necessary to decide which functionality needs to be provided to the applications in order to gain benefits from using the Grid to tackle the challenges described in the use cases. This document describes the functional requirements captured by the use cases as of the end of February 2006 and outlines an architectural design derived from these requirements.

## 2 Scope

This section summarizes the scope of the *Database Management* (DBM) and *Data Stream Management* (DSM) services.

- This document will not define new protocols or data formats. We use specifications currently under review by standardization bodies or which are widely-accepted (XML, XQuery, WSRF, SQL). We evaluate whether these are stable enough to guarantee a solid foundation for our architecture. Existing standards or third-party implementations are only extended or adapted, should they be insufficient for either service.
- The initial implementation focuses on key functionality.
- This document will not define a security architecture for the AstroGrid-D middleware.
- *Data Stream Processing* in the context of this document always implies a filtering step in the processing of data. It will not define an architecture for pure (communication) data streams. Such data streams will not be handled by the DSM service.
- This document will not address any user interface specific design issues (portal, graphical user interface, command-line). The application developers can accommodate users with varying experience levels.

## 3 Requirements

Intensive cooperation within the AstroGrid-D federation resulted in several use cases. These describe the current approaches for solving scientific challenges with current technologies and how these applications would benefit from AstroGrid-D. More details on AstroGrid-D use cases are available on the AstroGrid-D intranet.<sup>2</sup>

---

<sup>2</sup><http://www.gac-grid.de/intranet/usecases/index.html>

Identifier	Use Case (Contributors)
UC1	Astrometric Matching (MPE, TUM)
UC2	Clusterfinder (AIP, MPE)
UC3	Millennium Simulations Queries (MPA, MPE)
UC5	STELLA (AIP)
UC6	PLANET (ZAH)
UC7	Planck Process Coordinator (MPA)

Table 1: The Use Cases analyzed in this document.

There are additional projects which are not covered in this document, but will be discussed with regards to the components under discussion. The *Data Management Component* (DMC) from the MPA could be used by the services. The FRINGE project at MPA and the GAIA-DB project of ZAH are additional projects which will be considered for further requirements.

The functional requirements of the individual use cases are numbered according to the following scheme: **[FR-UC1-10]** denotes a functional requirement (FR) originating from the first use case. The mapping of the identifiers for the use cases is shown in Table 1.

### 3.1 Astrometric Matching (UC1)

The middleware should enable the astronomer to distribute data analysis tasks across several nodes if applicable to achieve load balancing. By removing irrelevant data as early as possible (early filtering), the researchers address the issues of combinatorial explosion. As a side-effect, network-traffic and processing time are reduced. Exchanging and processing data between computational nodes in the form of data streams can result in increased overall performance and reduce memory requirements.

**[FR-UC1-10] Provide access to distributed archives.** Many astrophysical projects publish their results on their own in widely distributed federated archives. The AstroGrid-D middleware should provide access to these archives.

**[FR-UC1-20] Distribute computation across several nodes.** To achieve better load balancing and higher flexibility the processing of the data catalogs should not happen only at the machines of the end-users but also inside the network.

**[FR-UC1-30] Support early filtering of irrelevant data.** To increase the number of archives that can be compared, unreasonable matches must be filtered out as soon as it can be determined that they will not contribute to the final result. This kind of filtering should be supported by the services.

**[FR-UC1-40] Overcome memory limitations of the current approach.** Federated data archives can be correlated by a single client. However, the number of catalogs is limited by the available memory of that machine if an in-memory approach is used [1, 6].

**[FR-UC1-50] Provide early feedback to intermediate results.** The sooner users are provided with (intermediate) results, the better they can compare the results to their expectations and, if necessary, adjust or terminate the job.

### 3.2 Clusterfinder (UC2)

**[FR-UC2-10] Support flexible storage of intermediate results.** Intermediate results are currently stored in files. It may be useful to store these in a database. Databases provide better support for semantic retrieval.

**[FR-UC2-20] Increase application responsiveness.** Caching frequently used archives within the community grid may speed up the application.

**[FR-UC2-30] Support transparent archive access.** It may be suitable just to specify a logical descriptor of a catalog and not a special physical location (e.g., *ROSAT* instead of *ROSAT at MPE Garching*). Then AstroGrid-D middleware can decide which server to use.

**[FR-UC2-40] Support efficient parallelism.** To create the likelihood maps for galaxy clusters the sky is divided up into several patches. These patches are not disjoint and may be seen as several overlapping data windows. *Window-based queries* are a technique used in data stream management systems and may increase parallelism in the application.

### 3.3 Millennium Simulation Queries (UC3)

**[FR-UC3-10] Support distributed clients.** To enable distributed researchers to access the data of the Millennium runs they should be available within the AstroGrid-D community.

**[FR-UC3-20] Optimize database-based filtering for queries.** The data volumes of the simulation runs are huge. Existing database indexing techniques ([10]) or special purpose indices ([4, 5, 13]) may provide the desired acceleration.

**[FR-UC3-30] Buffer intermediary results in databases.** Complex queries sometimes require additional post-processing. Performing these operations directly in a database can speed up these operations [9].

**[FR-UC3-40] Provide means for result-sharing.** Projects are typically realized as cooperations of various institutes. Sometimes, it is important to differentiate access rights of fellow researchers within or outside of a specific group [9], for example with regards to material which is not published yet.

### 3.4 Integration of robotic telescopes (UC5, UC6)

Networks of robotic telescopes facilitate automatic follow-up observations of extraordinary events, and promise to prolong the average duration of observations.

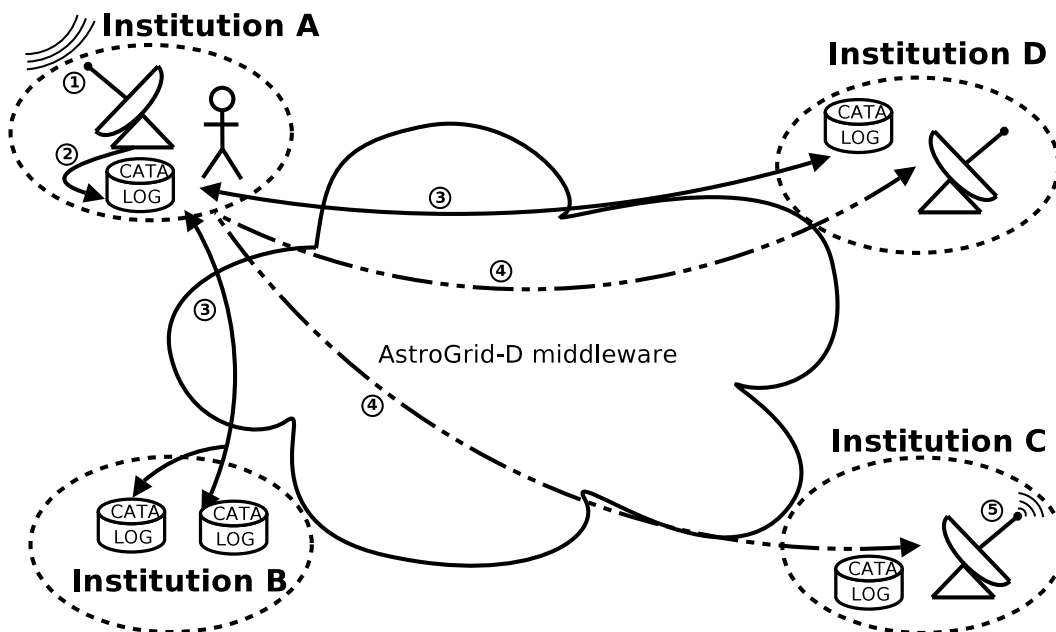


Figure 1: An alerter network. Distributed organizations are interconnected by a notification system which runs on AstroGrid-D middleware. When new data arrives (1) and comparisons with own (2) and distributed (3) data federations show that it is an interesting observation, an event is sent across the notification system (4) and potentially automatic follow-up observations are performed (5).

To interconnect telescopes and archive sites, we envision an alerter network (see Figure 1). Currently, manual changes and human-based decisions are predominant. Both projects, STELLA and PLANET, aim at replacing these workflows to a great fraction by intelligent software-driven steering and processing components.

**[FR-UC5-10] Interconnect distributed telescopes.** Telescopes should be allowed to *publish* important data, and to *subscribe* to data published by other telescopes or in persistent data archives to integrate telescopes seamlessly into an alerter network.

**[FR-UC6-20] Enable observations triggered by events.** If an interesting event (e. g., a gamma-ray burst, or a gravitational lensing anomaly) occurs, other telescopes should be notified in order to join in follow-up observations.

**[FR-UC6-30] Use interoperable messaging format.** There already exist several standards for communication between robotic telescopes. Remote Telescope Markup Language (RTML) and VOEvent are two XML based examples for such standards.

**[FR-UC6-40] Allow scalable and customizable event subscriptions.** Jobs at restricted resources like telescopes need to be prioritized in order to support scheduling decisions. Depending on the various event types an observatory

can support, it may restrict its subscription to those events. This avoids that telescopes receive notifications about events, they cannot observe.

**[FR-UC6-50] Support correlation of streaming and persistent data.** New observational data recorded by a telescope is compared with local existing archives and remote archives. This is needed to classify incoming data (such as “new data”, “found locally”, “found elsewhere”). Therefore processing of streaming and persistent data storages within the same workflow should be supported.

### 3.5 Planck Process Coordinator (UC7)

**[FR-UC7-10] Support in-network processing.** The Planck Process Coordinator will distribute a workflow in the AstroGrid-D network. *In-network* processing enables load balancing across several network nodes. This approach is more flexible than systems where computation only is possible either at the data source or at the users’ machines.

**[FR-UC7-20] Support mobile code for user-defined functions.** To support extensibility even after the processing framework has been started, it should be possible to dynamically deploy user-defined functions (or modules) during runtime. These operators should be published in decentralized repositories. This alleviates the maintenance of individual developers, as they only need to update one single repository while making the code publically available.

**[FR-UC7-30] Expand support for streaming across several nodes.** The Process Coordinator already supports *intra-node* stream processing, i. e., streaming data between processes on the same machine. Adding support for *inter-node* (network-wide) data stream processing opens new application areas. In addition, results are delivered faster to users during the execution. Users can early react to (un)expected results.

## 4 Architectural Design

The database management principles and the data stream management concepts are closely interrelated. Our design is influenced by the requirements analysis and experience in the field of query processing in distributed Internet information architectures [2] and data stream management systems [12]. We structure the functionality into network architecture, communication, and processing characteristics.

### Network Architecture

**Flexible Peer-to-Peer (P2P) topology** A classic client-server architecture does not fit the notion of distributed cooperating institutions. A Peer-To-Peer

network, where each participant is server as well as client, reflects better the picture of collaborative “peers”.

**Node classification** Not all participants in such a network are homogeneous. Therefore a distinction can be made between *thin-peers*, *super-peers*, and *speaker-peers*. Thin-peers are the basic clients, only subscribing to data streams, providing data sources and operators. Super-peers are the backbone of the data stream management component and perform all the computation. A speaker-peer is a super-peer which optimizes the query processing.

**Grid Services** Grid Services are well-established in the field of astrophysics and support the realization of loosely-coupled services which will be orchestrated to the AstroGrid-D middleware.

## Communication

**XML data streams** XML is the *de facto* way to exchange data on the web between web services. We therefore concentrate on XML data streams in the first place. When it is applicable to leverage internal structural information within binary data formats for data stream processing, binary data streams may be supported.

**Overlay routing** Nodes within the P2P architecture communicate via the overlay connections. Therefore data flow can be channeled with regards to optimization objectives.

## Processing

**Multi-query optimization** If many queries run concurrently on the system, all relevant queries are analyzed for possible reuse.

**In-network query processing** Support processing within the network opposed to only processing at the data source (query-shipping) or at the clients (data-shipping). This leads to better load balancing and less redundant calculations.

**Framework for external (user-defined) operators** Integration of user-defined operations manifesting in mobile code supports dynamic deployment and increases the extensibility of the data stream management system.

## Technologies Used

The AstroGrid-D middleware, including database management and data stream management, will be built as *service oriented architectures (SOA)*. The individual functionality will be realized as web services either using standard web services on SOAP or stateful web services using the *Web Services Resource Framework*

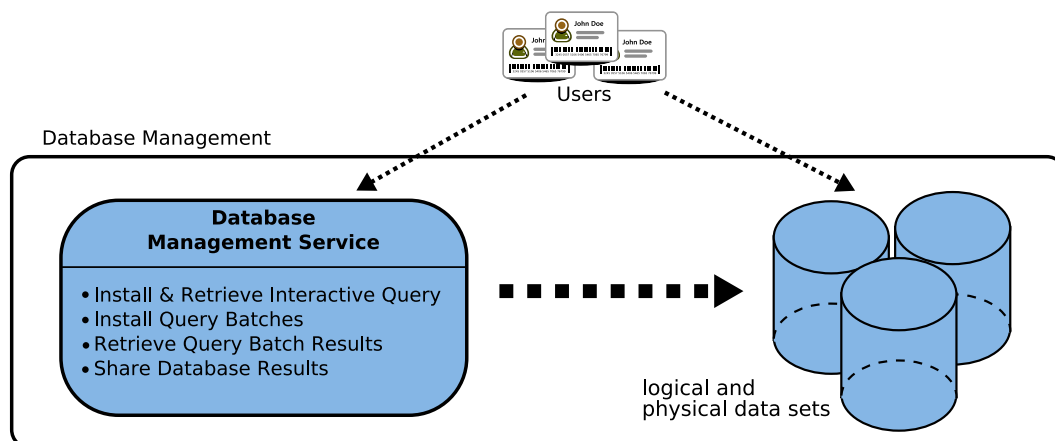


Figure 2: The *Database Management* component. Users access databases either directly or via the interface provided by the web service.

(WSRF). *Open Grid Services Architecture (OGSA)* is our Grid middleware of choice, as it is used in the astrophysical community and other D-Grid projects. Its current reference implementation Globus Toolkit 4 is based upon several web standards accepted in the scientific community and industry (WSRF, WS-Notifications).

For database data access, users and services can use SQL queries. For XML based data streams, we use XQuery. Extensions to XQuery may be employed to adhere to the different semantics when processing streaming and persistent data. The Astronomical Data Query Language (ADQL), a standard currently under development in a working group of the International Virtual Observatory Alliance (IVOA), the international standard body for the Virtual Observatory Community may be suitable as query language, too.

## 5 Database Management Component

The integration of DBMSs is realized using web service front-ends and interfaces. Furthermore, knowledge of existing database optimization techniques (such as spatial indices) will influence the design of this component. Existing systems like SkyServer<sup>3</sup> and the data management of AstroGrid (UK)<sup>4</sup> will be evaluated for providing reusable components for the implementation of the database management component.

The database management (see Figure 2) provides a web service interface and optionally transparent archive access to distributed astrophysical data sets.

**Install And Retrieve Interactive Query.** Users mainly access astronomical data sets through interactive interfaces. For such queries, responsiveness is of great

<sup>3</sup><http://www.skyserver.org/>

<sup>4</sup><http://www.astrogrid.org/>

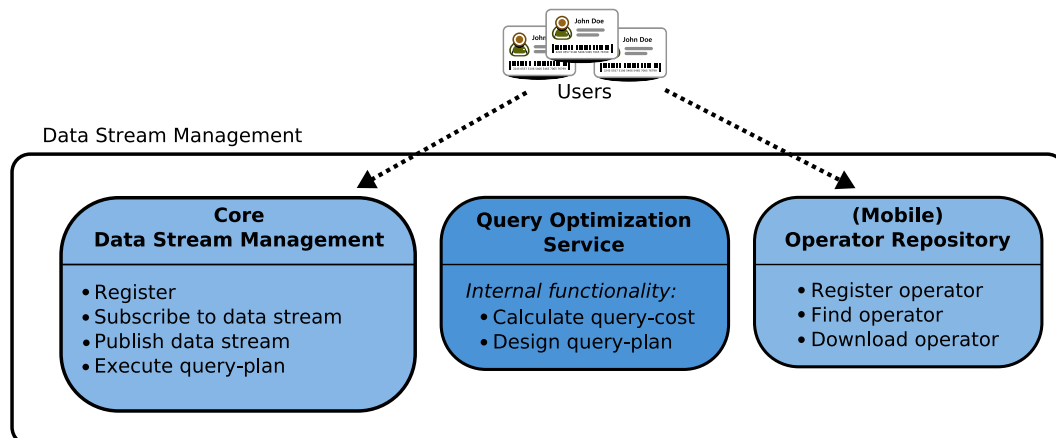


Figure 3: Data Stream Management components.

importance. Having a limited result set size or a threshold of computation time are therefore generally accepted.

**Install Query Batches.** The AstroGrid-D middleware enables users to send long-running queries to data archives. These queries can have unrestricted result size and computing duration. The queries are processed unattended (and therefore are called *query batches*). Users specify queries either using HTML input-forms, configuration files, or direct SQL and specify which catalog to access.

**Retrieve Query Batch Results.** Astronomers, which have installed a query batch, may not be online during the complete processing. However, when the processing has successfully finished, users can retrieve the results.

**Share Database Results.** Users may specify which parts of virtual organizations can read results from its queries. The User and Group Management of the AstroGrid-D middleware validates that the user has the appropriate authentication.

## 6 Data Stream Management Components

Figure 3 offers a general overview of the data stream management system. Certified AstroGrid-D users (visualized by their id-cards) access data stream functionality of the AstroGrid-D middleware by interfaces provided by the *Core Data Stream Management* and *(Mobile) Operator Repository* services. The *Query Optimization Service* only offers internal functionality and is not accessible directly for the users. We describe the individual core components in more detail in the following sections.

Designing the data stream management system based on Peer-To-Peer (P2P) technology reflects the structure of the AstroGrid-D collaboration. Some machines (or

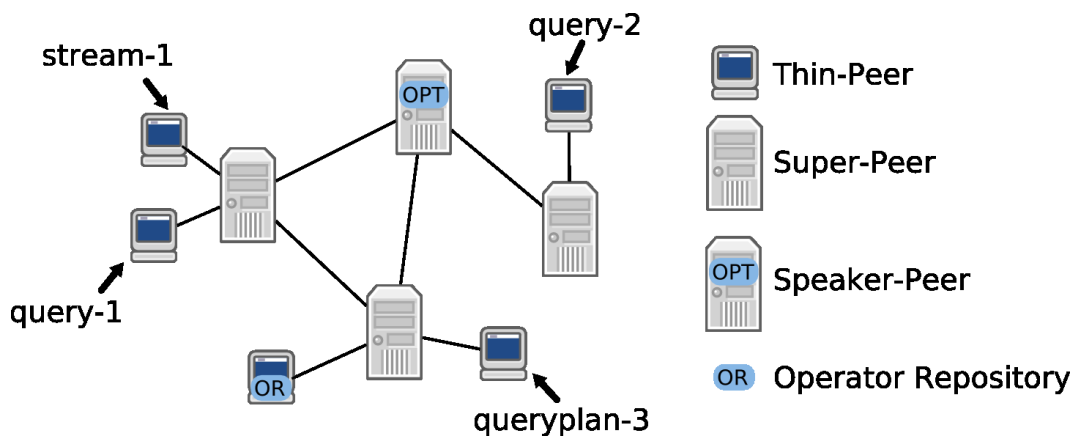


Figure 4: A data stream network. *Thin-peers* publish and subscribe to data streams. These are processed at *super-peers* and a dedicated *speaker-peer* optimizes the query-processing according to a given cost-model.

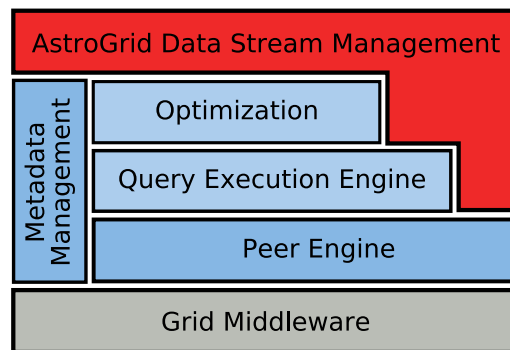


Figure 5: Data Stream Management Protocol Stack.

nodes) have a high availability and constitute a stable backbone of so-called *super-peers*. A distinguished super-peer, a so-called *speaker-peer* runs an optimization service to perform the multi-query optimization in the network. *Thin-peers* are subscribers and publishers of data streams or repositories of stream processing operators (see Figure 4). These repositories provide user-defined functions which can be used by any super-peer in the data stream network. While *query-1* and *query-2* in Figure 4 will be distributed by the internal *Query Optimization Service*, stream operators of *queryplan-3* will be installed as the user has decided.

The protocol stack of the Data Stream Management is depicted in Figure 5. The Grid Middleware layer (gray layer) is an interface which hides details of specific Grid middleware implementations. Thus the architectural design is more concise and is aware of other Grid middleware providers.

## 6.1 Core Data Stream Management

**Register.** This function allows users to interface with the data stream management system. This is required to avoid redundant calculations and transmissions of data streams. For this purpose users register with a super-peer, e. g., the physically closest. Those super-peers are the gateway for users to publish or subscribe to data streams.

**Publish Data Stream.** To publish data inside the AstroGrid-community by using the data stream management system, users can insert several kinds of data. To make multi-query optimization work, some structural data needs to be provided, when a new stream is introduced into the architecture.

**Subscribe To Data Stream.** After having registered with a super-peer, data streams can be subscribed by specifying a subscription in a suitable *subscription language*. This subscription language needs to provide *content-based* (selections) and *structure-based* (projections) filtering techniques [7, 8].

**Execute Query-Plan.** In some situations users have additional knowledge about the capabilities of the individual nodes. This interface provides users with the option to decide individually where the data processing occurs in the network. In addition to that, this interface can be seen as a manual optimizer, when the operators are not supported by built-in techniques. The system defines an XML based format for query-plan description, regardless whether they are user-defined query-plans or query-plans generated by the optimizer [11].

*Execute Query-Plan* is also used by the *Design Query-Plan* interface of the Query Optimization Service.

## 6.2 Query Optimization Service

This service optimizes data stream processing with regards to a cost model. A newly issued query will be analyzed by extracting properties, which then can be compared with streams already available in the system and possible sources for *reuse* can be spotted. The Query Optimization Service is the integral part of achieving load balancing by intelligent operator distribution and stream dissemination.

Cost models will be transparent to the components using them, and therefore they can be configured in a flexible fashion and enable comparative research with different strategies [3, 8].

As far as the current use cases are concerned, it is reasonable to use a single query optimization instance for optimizing the query processing. In a larger environment, however, global optimization could be a bottleneck. For this scenario partitioning the complete network into several subnets with an individual optimization component offers additional scalability.

**Calculate Query-Cost.** Using an initial query-plan, the optimization component iterates over every possible query-plan and compares the plans based on their cost.

**Design Query-Plan.** When a subscription is issued to the data stream management system (via *Subscribe Data Stream*), a query-plan for this query needs to be generated. This query-plan contains information about which part of the query processing is done at which peers. The Query Optimization component then generates a query-plan which is considered optimal according to the cost model (by subsequent calls of *Calculate Query-Cost*). This optimal query-plan is eventually installed in the data stream management system via *Execute Query-Plan*.

### 6.3 (Mobile) Operator Repository

Some basic functionality for data stream processing will be provided by the data stream management architecture. As many of the use cases described above suggest, support for user-defined functionality is crucial for acceptance by the astrophysical community. Some functionality can be installed at any node. This kind of operator will be called *mobile operator* or *mobile code*. Code which has specific hardware or software requirements is considered non-mobile in the sense specified above.

The basic idea behind this service is the sharing of implementations with fellow researchers. The functionality is enriched with specific metadata and then made accessible, e. g., on a web server or the local file system.

**Register Operator.** Users provide their self-defined functionality along with a metadata description. The metadata model should be designed in a fashion which allows the optimization of subscriptions containing mobile operators at a later stage in the project.

**Find Operator.** If users are looking for a specific functionality, they need to describe the functionality they want. This description can be compared with existing operator descriptions. Matching operators can then be used in the data stream processing.

**Download Operator.** When a query-plan is installed in the system, an operator can be retrieved from an existing operator repository and then dynamically installed on the system.

### 6.4 Interaction with other AstroGrid-D Components

Figure 6 gives an abstract overview of the AstroGrid-D middleware components that interact with the management services for databases and data streams.

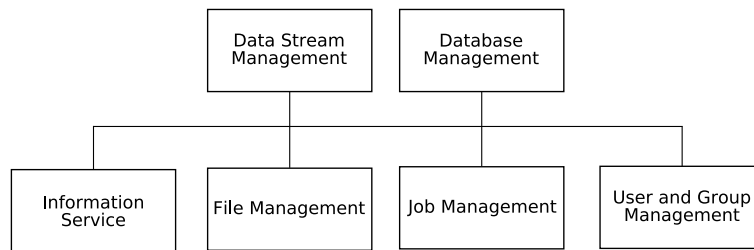


Figure 6: Overview of interfacing components for Database Management and Data Stream Management.

*User and Group Management* provides the verification of user credentials either before connecting to either of the components or when the integrity of mobile code needs to be verified.

The Database Management and Data Stream Management components will interface with *File Management* as soon as results of interactions have materialized as files.

The query processing at the databases and the data stream operators are consumers of AstroGrid-D resources. To integrate seamlessly with other Grid-Jobs, the *Job Management* will be informed of these activities and will update the resources' state, accordingly.

Finally, the *Information Service* will provide access to general metadata within AstroGrid-D or metadata specific to the individual applications. Working group 4 (the authors) and working group 2 (which designs the information service) are currently discussing more details on the interaction between the information service and the services for database and data stream management.

## Glossary

**ADQL** Astronomical Data Query Language.

**Continous query** Queries in a data stream management system. These queries are sustained in a DSMS until the subscriber cancels the query.

**Distributed Database Access** Distributed clients access one or more distributed database archives.

**DSMS** Data Stream Management System.

**IVOA** International Virtual Observatory Alliance.

**Mobile code** Code which can be loaded from a (remote) location. It is deployed into a running system and eventually executed.

**OGSA** Open Grid Services Architecture.

**PLANET** Probing Lensing Anomalies NETwork.

**RTML** Remote Telescope Markup Language. RTML is an XML dialect meant to enable the transparent use of remote and/or robotic telescopes. Link: <http://www.uni-sw.gwdg.de/~hessman/RTML/>.

**SOAP** Simple Object Access Protocol.

**Speaker-peer** A *super-peer* which optimizes the data stream usage within a DSMS.

**Super-peer** A highly available backbone node of the Grid.

**Thin-peer** Client of a DSMS which publishes data streams and/or subscribes to those. Can also provide user-defined operators.

**Window-based query** Queries with different semantics in contrast to ordinary continous queries. Time-based or element-based windows are specified to make these queries feasible over long-lasting data streams.

**XQuery** Query Language for XML documents standardized by W3C.

## F: References / Bibliography

### References

- [1] H.-M. Adorf, G. Lemson, and W. Voges. The GAVO Cross-Matcher Application. In *Conf. on Astronomical Data Analysis Software and Systems*, San Lorenzo de El Escorial, Spain, October 2005.
- [2] R. Braumandl, M. Keidl, A. Kemper, D. Kossmann, A. Kreuz, S. Seltzsam, and K. Stocker. Objectglobe: Ubiquitous query processing on the internet. *The VLDB Journal*, 10(1):48–71, August 2001.
- [3] Bo Feng. Optimierung und Bewertung der Anfrageauswertung auf Datenströmen in P2P Netzwerken. Diploma thesis, Technische Universität München, Germany, 2005.
- [4] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelman. HEALPix – a Framework for High Resolution Discretization, and Fast Analysis of Data Distributed on the Sphere. *Astrophysical Journal*, 622:759–771, April 2005.
- [5] Jim Gray, Alexander S. Szalay, Gyorgy Fekete, William O’Mullane, Maria A. Nieto Santisteban, Aniruddha R. Thakar, Gerd Heber, and Arnold H. Rots. There Goes the Neighborhood: Relational Algebra for Spatial Data Search. Technical Report MSR-TR-2004-32, Microsoft Research, Microsoft Cooperation, Redmond, WA, USA, April 2004.
- [6] Sebastian Huber. Grid-based Processing of Multiple Data Streams in E-Science. Master’s thesis, Technische Universität München, Germany, 2005.
- [7] Franz Häuslschmid. Infrastrukturen und Verfahren zur adaptiven Anfragebearbeitung und Optimierung in Datenstrom-Management-Systemen. Diploma thesis, Universität Passau, Germany, 2005.
- [8] Richard Kuntschke, Bernhard Stegmaier, and Alfons Kemper. Data stream sharing. Technical Report TUM-I0504, Technische Universität München, 2005.
- [9] William OMullane, Nolan Li, Maria Nieto-Santisteban, Alex Szalay, Ani Thakar, and Jim Gray. Batch is back: CasJobs, serving multi-TB data on the Web. Technical Report MSR-TR-2005-19, Microsoft Research (MSR), February 2005.
- [10] Ina Schmitz. Optimierung von Datenbankanwendungen aus der Astronomie am Beispiel des ROSAT-Datenkatalogs. Diploma thesis, Universität Passau, Germany, 2006.
- [11] T. Scholl. Distributed Processing of Data Streams in P2P Networks. Diploma thesis, Universität Passau, Germany, 2005.

- [12] B. Stegmaier, R. Kuntschke, and A. Kemper. StreamGlobe: Adaptive Query Processing and Optimization in Streaming P2P Environments. In *Proc. of the Intl. Workshop on Data Management for Sensor Networks*, pages 88–97, Toronto, Canada, August 2004.
- [13] Alex Szalay, Jim Gray, Gyorgy Fekete, Peter Kunszt, Peter Kukol, and Ani Thakar. Indexing the Sphere with the Hierarchical Triangular Mesh. Technical Report MSR-TR-2005-123, Microsoft Research (MSR), September 2005.