The VL-Abstract Machine: a Data & Process Handling System on the Grid

A. Belloum[1], Z.W. Hendrikse[1], D.L. Groep[2], E.C. Kaletas[1], A.W. van Halderen[1], and L.O. Hertzberger[1]

[1] University of Amsterdam. Computer Science Department,
Kruislaan 403, 1089 SJ Amsterdam, The Netherlands
{adam,zegerh, kaletas, berry, bob}@science.uva.nl
[2] National Institute for Nuclear and High Energy Physics NIKHEF,
P.O. Box 41882,1009 DB Amsterdam, The Netherlands
{davidg}@nikhef.nl

Abstract. This paper presents the architecture of the *Virtual Laboratory Abstract-Machine* (VL-AM), a data and process handling system on the Grid. This system has been developed within the context of the Virtual Laboratory (VL) project, which aims at building an environment for experimental science. The VL-AM solves problems commonly faces when addressing process and data handling in a heterogeneous environment. Although existing *Grid technologies* already provide suitable solutions most of the time, considerable knowledge and experience outside of the application domain is often required. The VL-AM makes these grid technologies readily available to a broad community of scientists.
Moreover, it extends the current data Grid architecture by introducing an object-oriented database model handling complex queries on scientific information.

1 Introduction

Grid technology provides a persistent platform for high-performance and high-throughput applications using geographically dispersed resources. It constitutes a potential solution for effective operation in large-scale, multi-institutional, wide area environments, called *Virtual Organizations* [1, 2].

Various research groups are currently investigating Grid technologies, focusing on scientific applications. In an effort to outline requirements on Grid technology, Oldfield [3] has gathered information on a large number of Grid projects. These projects cover a wide range of scientific applications, including (bio)medics, physics, astronomy and visualization. Intensive use of large data sets was a common feature of all projects involved. Therefore a scalable and powerful mechanism for data manipulation is required, known as *Data-grid*.

The Data-Grid is defined as a layer offering services related to data access and data manipulation. Within the grid-community Data-Grid systems are generally agreed to contain: Data handling, Remote processing, Publication, Information discovery, and Analysis [3].

One of the advantages of Data-Grid systems is their ability to support location-independent data access, *i.e.* low-level mechanisms are hidden for the user. Five levels of "transparency" have been identified by the Data Access Working Group[1]: name, location, protocol, time and the single sign-on transparency [4].

A proposal for a generic Data-Grid architecture is currently being studied by the Data Access Working Group, offering protocol-, location- and time-transparency. The so-called "Data Handling System" provides an interface connecting the separate components of this data access system and offering access to it, see Figure 1.

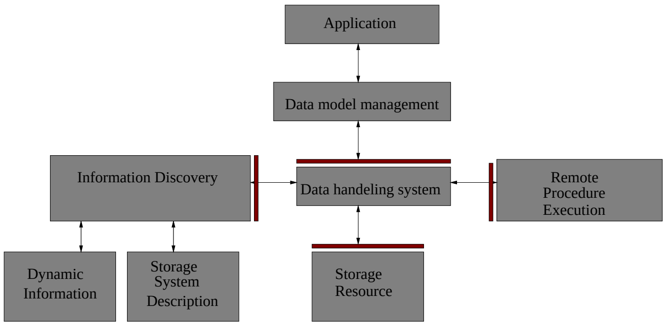

Fig. 1. The general Data-Grid architecture as proposed by the Data Access Working Group[5]. Note the central position of the Data handling system.

At this moment, the Globus Toolkit[2] provides the leading Grid-technology and is becoming a *de facto* standard in this area. It offers a variety of tools which may be used to build an efficient grid testbed quickly [5]. However, since the Globus toolkit constitutes a low-level middle-tier, it is not intended to be harnessed directly by a broad scientific community. An additional layer is needed to bring Globus technology to scientists, the most important group of Grid users thus far.

The purpose of the VL-Abstract Machine (VL-AM) is to bridge this gap. If we envisage a general Grid architecture as a four-tier model (application, application toolkit, Grid-services and Grid-fabric tier), the VL-AM is located in the third — the application toolkit — tier.

The VL-AM will initially focus on four application domains:

- chemo-physical analysis of surfaces at micrometer scales,
- bio molecular genome expression studies,
- a biomedical visualization workbench, and
- an analysis work bench for electronic fee collection.

[1] a subgroup of the Global Grid Forum, see `http://www.gridforum.org/`
[2] `http://www.globus.org/`

In the context of the VL, experiments are defined by selecting processing elements (also referred to as modules) and connected these in such a way as to represent a data-flow; the topology of an experiment may be represented by a data-flow graph (DFG). As such, the modules comprising the data-flow have a strong analogy to those used in IRIS explorer[3]. Processing elements[4] are independent of each other and in general may have been developed by different persons. Therefore the VL-AM Run Time System (RTS) should offer an implementation of (most of) the transparency layers and a platform enabling modules to perform data processing and computations.

In addition to the VL-AM RTS, a VL-AM front end has been designed to form the main interface by which external users can access VL compute resources, data elements and software repositories. The VL-AM front end queries other VL components on behalf of the end-user to collect all information needed to facilitate his work within the VL environment. The collected information is presented to the end-users through a graphical user interface (GUI).

The remainder of this paper is organized as follows: Section 2 describes the way in which experiments are performed. The architecture for the VL-AM designed to support experimentation is addressed in 3. In Section 4, the main components of the VL-AM front end are presented. Section 5 focuses on the design of the VL-AM RTS. Thereafter Section 6 highlights some important features of related projects. Finally Section 7 presents some conclusions and future prospects.

2 Experimenting in VL

The VL [6] designs and develops an open, flexible and configurable laboratory framework. This includes the hardware and software to enable scientists and engineers to work on their problems collectively by experimentation. Developments are steered initially by the aforementioned scientific application domains.

Studying these applications domains has allowed us to identify a set of generic aspects valid for all of these scientific domains. By combining these generic features that characterize scientific experiments with the characteristics of case-studies, it has been possible to develop a database model for the Experimentation Environment (EE). The EE data model [7] focuses on storage and retrieval of information to support the generic definition of the steps and the information representing a scientific study, in order to investigate and reproduce experiments. The EE data model allows any ordering of processes and data elements, which enable one to construct an arbitrary process-data flow. The EE data model has been applied to build the information management system for the chemo-physical surface analysis lab (MACS) [8] and the EXPRESSIVE [9] (genome expression) application databases.

Within the context of the VL, experiments are thus embedded in the context of a *study*. A study is a series of steps intended to solve a particular problem

[3] `http://www.nag.co.uk/Welcome_IEC.html`

[4] Separate modules may be grouped to form a larger so-called aggregate-module.

in an application domain, and corresponds to a process flow. As such, a study generally comprises experiments performed using the VL-AM RTS, process steps performed outside the scope of the RTS – *e.g.* manually – and the description of data elements related to objects or process steps.

A *process flow* is the result of investigating the established work flow in a specific application domain, *e.g.*, space-resolved surface analysis or genome expression studies. Such a process flow is shown in Fig. 2. This particular flow describes chemo-physical analysis of samples using space-resolved techniques: material analysis of complex surfaces (MACS) [8]. A similar process flow exists for bio genetic experiments [9]. The context of the operational steps (represented by ovals) is provided by entity meta data (represented by boxes). For one-time experiments, a trivial process flow should be defined, *i.e.*, consisting of one operational step only, and limited contextual meta data associated with the result.

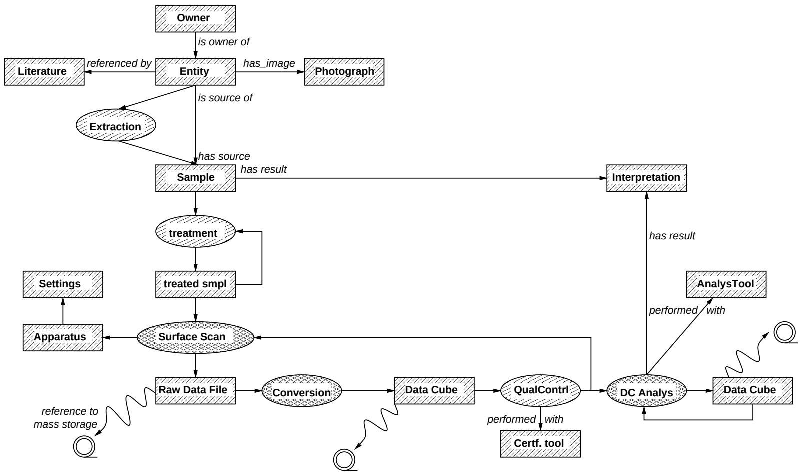

Fig. 2. Typical steps in a surface-analysis study, as represented by the MACS [8] process flow. The hatched rectangles represent meta data associated with objects (either physical objects or bulk data on mass storage). Hatched ovals represent operations: cross-hatched ovals are performed using the VL-AM Run Time System, the single-hatched ovals represent meta data associated with manually performed process steps.

A process flow template (PFT) is used to represent a process flow of the study to the scientist. The scientist is guided through the process flow using context-sensitive interaction. each PFT has a starting point; in the case of MACS this is the 'Entity', see figure 2. At first, interaction is possible with the 'Entity' box only. The object characteristics (meta data) should be supplied by the end user, *e.g.*, via a fill-in form. Once the 'Entity' data is supplied, the directly related items (here: Literature, Extraction, Sample, Photograph and Owner) become available for interaction. Boxes require object meta data to be supplied, process steps require a description of the procedure applied to particular objects. In the case of sample extraction, the process is performed outside of the VL-AM scope and details should be supplied (by means of a fill-in form).

Cross hatched ovals represent process steps performed by the VL-AM RTS. Selection of such a process step by a user will initiate a VL experiment, refer to section 5. Any resulting meta data from a process step performed by the VL-AM RTS is captured. This context consists of the VL-AM RTS experiment, constituting a process step, input and output data objects and position of a box within the PFT. This meta data is optionally added to by the scientist by means of a fill-in form. Note that meta data are stored in the application database, whereas any resulting 'base data sets' may be stored outside of this context, depending on the application.

3 VL-AM Architecture

The VL-AM is the heart of the VL. It should support the functionality provided to the VL to the users, hiding details related to the middleware layer that require non-application specific knowledge. Its proposed architecture is represented in Figure 3. It consists of four major components:

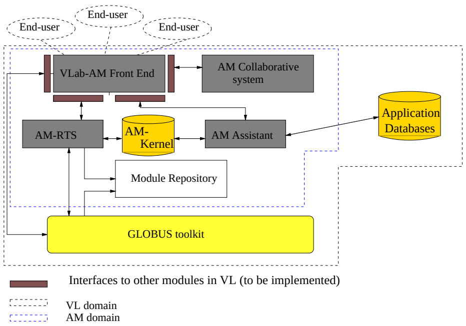

Fig. 3. Representation of the VL-AM architecture and its components

– a front end,
– a run time system (VL-AM RTS),
– a collaboration system (VL-AM Collaboration), and
– an assistant (VL-AM Assistant).

These components are supported by an information management system that includes the Kernel DB and application databases.

The VL-AM front end and the VL-AM RTS form the core of the VL-AM. Together they take care of the execution of tasks. The VL-AM front end is the only component with which the VL-users interact: it provides the user interface (UI) to the VL-AM. It delegates VL users commands to the various components and relays the results back to the user. The VL-AM RTS is responsible for scheduling, dispatching and executing the tasks comprising an experiment. These components are discussed in Sections 4 and 5.

The two remaining are not directly involved in the execution of the VL-experiments: they support a user in composing an experiment by providing the templates and information about the available software and hardware resources, and by providing communication mechanisms during execution.

- *VL Collaboration* The VL Collaboration system may operate in a synchronous or asynchronous mode. Certain VL-experiments require a real-time collaboration system that allows users to exchange ideas, share working space, and monitor ongoing experiments simultaneously. On the other hand, experiments may last over extended time scales, the VL collaboration system supports a dynamic number of scientists, *i.e.* users may join or leave an experiment while it is running. In this case, VL collaboration system operates in an asynchronous mode, delivering urgent messages to disconnected members of ongoing experiments.
- *The VL-AM Assistant:* The VL-AM Assistant is a subsystem that assists users during the design of a VL experiment, *e.g.*, it may provide module definitions or PFT's (of previous experiments) to the VL-AM front end. Decisions of the VL assistant are supported by knowledge gathered from the VL-AM Kernel DB, the application database, and the Grid Information Service (GIS).

The VL-AM Kernel DB stores administration data, such as the topology of experiments, user sessions and module descriptions. The information in the VL-AM Kernel DB is accessed by the *VL assistant*, but it is presented to the user by means of the VL-AM front end.

The VL-AM Kernel DB is used to extend the GIS-based resource discovery subsystem, currently implemented as a set of LDAP directory services. It is based on object-oriented database technology. Therefore it improves on support for complex queries (as opposed to a hierarchical data model, inherent to LDAP).

A database model for the VL-AM Kernel DB has been developed to keep track of the necessary information for the VL-AM RTS to execute properly. Amongst others, this includes user profile data, processing-module information and experiment definitions [10].

As stated above, an experiment may be represented by a DFG, composed of modules that are connected by their I/O ports. This implies the definition of two main data types for the VL-AM Kernel DB data model:

- the *Experiment Topology* defines a specific set of modules and their interconnects, which make up a VL-experiment. The topology is stored in the VL-AM Kernel database to be reused and to analyze results of a particular experiment at another time.
- the *Modules* represent the processing elements and are centrally registered by a VL administrator. Every module contains a number of *ports*. A developer

should summarize its functionality, and specify the data-types of the ports and the resource requirements needed during execution.

4 VL-AM Front End

Since the VL-AM front end provides the interface between end-users and the constituent components of the VL middleware, it is required to be user friendly and to provide seamless access to the available features of the VL. Given the wide range of scientific disciplines, it is essential that the VL-AM front end is designed such that it can easily be adapted to include either a new scientific domain or a new group of users.

Its modular design therefore combines *Grid technology* components for authorization, resource discovery, and monitoring with existing *commodity components* that are more focused on the collaboration, editing and science-portal areas.

Commodity Grid Toolkits (CoGs) provide an appropriate match between similar concepts in both technologies [11]. The tools already developed for Java [12] provide an apt basis for supporting the diverse current and future end-user groups and applications.

The VL-AM front end consists of:

1. *Resource discovery component*, enabling users to make an inventory of all the available software, hardware and information resources. The information needed to provide such a service will be offered by a collection of meta-data stored either in the AM-Kernel database or in the Globus GIS or meta data replica catalogue. A connecting layer contains interfaces to the subsystems managing the meta data directories.
2. *VL execution and resource monitoring*: Two ways of monitoring are defined: on the one hand, application-level parameters and module state may be monitored [13], on the other hand process and resource status by using Grid monitoring tools. The monitoring is carried out using the VL-AM front end, although, in the case of resource monitoring, Grid monitoring tools can also be used directly.
3. *VL Experiment editing and execution*: Editing is performed using an appropriate science portal. All portals provide a drag-and-drop GUI (like the one shown in Figure 4). Processing elements, selected from a list supplied by the VL Assistant appear on the editing pane as a box with pending connection endings. Processing elements that are specific to a certain scientific discipline appear in a particular science portal only, generic processing elements may be selected using any portal. Connections are established by drawing lines between output and input ports. In addition, the experiment editing interface may automatically generate a module skeleton appropriate to the users needs, if the user want to add new functionality. Thereafter a user can add his own code to this skeleton (module skeletons are described in section 5).

   The VL Experiment editing component generates a graph like representation of an experiment and forwards it to the VL-AM RTS. While editing an

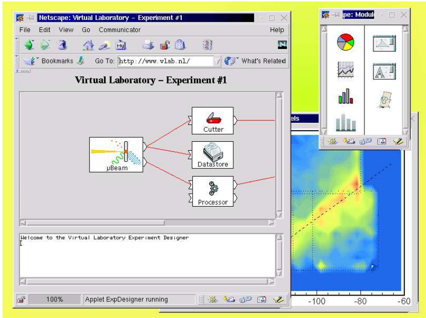

Fig. 4. VL-AM front end editing “ergonomics”

experiment, a VL end-user is assisted by the system. Depending on the study context, a relevant topology of a previously conducted experiment may be suggested by the VL Assistant.

5 VL-AM RTS

The VL-AM RTS executes the graph it receives from the VL-AM front end. It has to provide a mechanism to transfer data between modules. These data-streams are established between the appropriate modules. The modules are instantiated and parameter and state values cached or propagated by the VL-AM RTS.

5.1 Design

From case studies (*e.g.* the chemo-physical surface laboratory [14]), it becomes clear that at least three connection types have to be supported:

– interactions among processing elements located on the same machine,
– interactions among two or more machines and
– interactions with external devices (where some of the processing elements are bound to a specific machine).

Moreover, the chemo-physical case study has not only shown the need for point-to-point connections, but also the need for connections shared by several modules, in the form of a fan-out. Since interactions with external devices are involved, QoS has to be dealt with properly.

Processing elements are developed independently. Communication is established using uni-directional ports. In order to make sure that only ‘proper’ connections exist, processing elements communicate via a strongly typed communication mechanism. Data types constitute the interface with which developers of modules must communicate with the ‘outside world’, *i.e.* with which modules must exchange their data. This will eliminate the possibility of an *architectural mismatch*, a well know problem in software composing design [15, 16].

The sole use of unidirectional, typed streams intentionally precludes the possibility for 'out-of-band' communication between modules. Such communication is only allowed between the VL-AM RTS and a module. Predefined, asynchronous events can be sent from the VL-AM RTS to any module (parameters) and from any processing module to the VL-AM RTS (state).

Modules are continuously active, *i.e.* once instantiated they enter an continuous processing loop, blocked by waiting for arrival of data on the input ports. The communication mechanism allows modules to communicate in a transparent way: they do not address their peers directly, but merely output their data to the abstract port(s).

The implementation of the VL-AM RTS would potentially suffer from a tedious development path, need it be built from scratch. Fortunately, existing Grid technologies provide suitable middleware for such a development. In particular, the Globus toolkit [1] provides various secure and efficient communication mechanisms that can be used for an implementation of many of the VL-AM RTS components. The use of Globus is further motivated by the fact that it is a *de facto* standard toolkit for Grid-computing.

Using Globus toolkit components, we take advantage of all the new software currently being developed around it. For instance, the mapping of an experiment data-flow graph (DFG) onto the available computing resources can be alleviated by using the Globus resource management module GRAM and information from the Grid Information Service GIS [5]. In addition, tools developed for forecasting the network bandwidth and host load such as the *Network Weather Service* [17] can be incorporated.

5.2 Implementation

Implementation of the VL-AM RTS on a stand-alone resource could be based on several inter-process communication (IPC) mechanisms such as sockets, streams or shared memory. Within a Grid environment, the Globus toolkit offers two possibilities to implement wide-area IPC: Globus IO[6], (mimicking socket communication) and GridFTP[18].

We have chosen to use the latter for the VL-AM. Although GridFTP is a more extensive protocol for data transfer, its merits outweigh its liabilities.

GridFTP is based on the FTP protocol[7]. It focuses on high-speed transport of large data sets, or equivalently a large number of small data sets. Key features include a privacy-enhanced control connection, seamless parallelism (multiple streams implementing a single stream), striped transfers and automatic window and data-buffer size tuning. Data transfer is memory-to-memory based, and optionally authenticated and encrypted.

Using GridFTP as a data transport mechanism in the VL-AM RTS has significant advantages:

---

[5] For information on the GIS, the reader is referred to `http://www.globus.org/mds/`

[6] `http://www.globus.org/v1.1/io/globus_io.html`

[7] IETF Internet Standard STD0009

- efficient, high-performance data transfer across heterogeneous systems,
- third party arbitrated communication inherent in the protocol (in our case, the AM RTS acts as an arbitrator in the inter-module communication protocol; ports are assigned symbolic names corresponding to quasi-URIs for GridFTP access),
- uniformity between modules and external storage resources like HPSS's producing data (the latter running GridFTP as a standard service).
- an elaborate API exists together with an implementation as part of the Globus toolkit.

However, for small packets of data exchanged, the use of GridFTP may induce additional overhead in the communications channel.

6 Related Work

Currently several projects are aiming at a redefinition of the scientific computing environment by introducing and integrating technologies needed for remote operation, collaboration and information sharing. The Distributed Collaboratory Experiment Environments (DCEE) program of the US Department of Energy involves a large set of such projects. In the following paragraphs we summarize briefly some of the important features of the DCEE projects:

The Spectro-Microscopy Laboratory [8] In this collaboration scientists at the University of Wisconsin-Milwaukee remotely operate the synchrotron-radiation instruments at the Berkeley Lab Advanced Light Source, Beamline 7.
A remote experiment monitoring and control facility has been developed within this project. It utilizes a server that collects requests on data from experiment and interfaces to the hardware specialized in collecting data. Client programs provide a user interface and allow researchers to design an experiment, monitor its progress and analyze results. The client software may be both accessed locally or remotely to conduct experiments. The prototype includes a.o. remote teleconferencing system that allows users to control remote audio-visual transmissions and a Electronic Notebook.

LabSpace[9]: This project focuses on building a virtual shared space with persistence and history. LabSpace led to a number of developments which have not only allowed remote access to scientific devices over the Internet, but also allowed scientists at Argonne to meet colleagues in virtual rooms. The prototypes produced within LabSpace are composed of a set of software components for communication between multiple networked entities using a brokering system. This system has been used to support collaboration from desktop to desktop over a LAN as well as from CAVE to CAVE over a WAN (a CAVE is an immersive 3D virtual reality environment).

[8] `http://www-itg.lbl.gov/ deba/ALS.DCEE/design.doc/design.doc.html`
[9] `http://www-fp.mcs.anl.gov/division/labspace/`

Remote Control for Fusion[10]**:** Lawrence Livermore National Laboratory, Oak Ridge National Laboratory, the Princeton Plasma Physics Laboratory, and General Atomics are building the "Distributed Computing Testbed for a Remote Experimental Environment (REE)," to enable remote operations at the General Atomics' D-IIID tokamak. This project focuses on the development of web-enabled control and collaboration tools. It has also produced a distributed computing environment (DCE) that provides naming, security and distributed file services. The conventional inter-process-communication system (IPCS) have been converted to use authenticated DCE remote procedure calls (RPC) to provide a secure service for communication among tasks running over a wide area network.

Environmental & Molecular Sciences Collaboratory The testbed developed at Pacific Northwest Laboratories (PNL) is based on instrumentation being developed for the Environmental Molecular Sciences Laboratory project at PNL. The goals of the Environmental Molecular Sciences Collaboratory are:

- to develop an understanding of scientific collaboration in terms of the tasks that distributed collaborators undertake, and the communications required to complete these tasks;
- to develop/integrate a suite of electronic collaboration tools to support more effective distributed scientific collaboration; and
- to use these tools for project analysis, testing, and deployment.

Security for Open, Distributed Laboratory Environments[11] This project developed at LBNL (Lawrence Berkeley National Laboratory) aims at the development of a security model and architecture that is intended to provide general, scalable, and effective security services in open and highly distributed network environments. It specializes on on-line scientific instrument systems. The same level of, and expressiveness of, access control that is available to a local human controller of information and facilities, and the same authority, delegation, individual responsibility and accountability, and expressiveness of policy that one sees in specific environments in scientific organizations. The security model is based on public-key and signed certificate.

The InterGroup Protocol Suite: The project at U.C. Santa Barbara is developing an integrated suite of multicast communication protocols that will support multi-party collaboration over the Internet. The collaboratory environment requires communication services ranging from "unreliable, unordered" message delivery (for basic video-conferencing) to "reliable, source-ordered" message delivery (for data dissemination) to "reliable, group-ordered" message delivery (for coordinating activities and maintaining data consistency) [19].

---

[10] `http://www.es.net/hypertext/collaborations/REE/REE.html`

[10] `http://www.emsl.pnl.gov:2080/docs/collab/CollabHome.html`

[11] `http://www-itg.lbl.gov/SecurityArch/`

7 Conclusions

The VL Abstract Machine (VL-AM) is middleware to make scientific experiments on a computational Grid accessible to a wide range of users. Through its front end, it provides science portals, specifically hiding details related to computing from the end-user scientist. Moreover, it will promote collaboration by offering real time (synchronous and asynchronous) collaboration tools. Since both the VL-AM Run-Time System (RTS) and VL-AM front end are based on tools from Globus, this project can incorporate new Grid developments.

The design of the VL-AM has been presented in this paper. Six main components and their mutual interaction has been outlined. An extensive meta-data driven system (active meta-data) based on application process-flows has been proposed (templates), in order to ease the execution of an experiment and to impose uniformity.

A first prototype of the VL-AM will support filters developed for the initial application domains: chemo-physical micrometer analysis, mass- and database storage, external device controllers and visualization. The basic experimental building blocks, modules, may be aggregated to form complex experiments using visual programming techniques. Once these are instantiated, the VL-AM RTS provides a transparent execution platform. Another important feature of the VL-AM is the data-typing of the I/O-streams.

In the future, the VL-AM will be extended with an intelligent agent, assisting scientists in various ways. It will collect data from the Grid layer, the Abstract-Machine layer and the application layer to search for an optimal allocation of the computing resources. Moreover, it will assist both during design and analysis of an experiment (data mining) and perform statistics on module resource consumption. In doing so, it will learn from the successes and failures of previous experiments, and utilize that knowledge to the benefit of other users.

References

1. I. Foster and C. Kesselman. Globus: A toolkit-based grid architecture. In *The Grid: Blueprint for a Future Computing Infrastructure*, pages 259–278. MORGAN-KAUFMANN, 1998.
2. Ian Foster, Carl Kesseleman, and Steven Tuecke. The anatomy if the grid: Enabling scalable virtual organizations. *International Journal on Supercomputer Applications, 2001*, (To be published), http://www.globus.org/research/papers/anatomy.pdf 2001.
3. Ron Oldfield. Summary of existing data grid. Draft http://www.cs.dartmouth.edu/~raoldfi/grid-white/paper.html, Grid Forum Data Acces Working Group, 2000.
4. Reagan Moore. Grid forum data access working group. Draft http://www.sdsc.edu/GridForum/RemoteData/Papers/, Grid Forum, 2000.
5. I. Foster and C. Kesselman. The Globus project: A status report. In *Proceedings of the Heterogeneous Computing Workshop*, pages 4–18. IEEE-P, 1998.

6. H. Afsarmanash and et al. Towards a multi-layer architecture for scientific virtual laboratories. In *Proceedings of High Performance Computing and Networking Conference*, pages 163–176, Amsterdam, The Netherlands, May 2000.
7. E.C. Kaletas and H. Afsarmanesh. Virtual laboratory experimentation environment data model. Technical Report CS-2001-01, University of Amsterdam, 2001.
8. Anne Frenkel, G. Eijkel, H. Afsarmanash, and L.O. Hertzberger. Information management for physics applications in the vl environment. Technical Report CS-2001-03, University of Amsterdam, 2000.
9. E.C. Kaletas and et al. Expressive - a database for gene expression experiments. Technical Report CS-2001-02, University of Amsterdam, 2000.
10. E.C. Kaletas, A. Belloum, D. Group, and Z. Hendrikse. Vl-am kernel db: Database model. Technical Report UvA/VL-AM/TN07, University of Amsterdam, 2000.
11. Gregor Von Laszewski, Ian Foster, Jarek Gawor, Warren Smith, and Steven Tuecke. Cog kits: A bridge between commodity distributed-computing and high-performance grids. In *Proceedings of the ACM Java Grande 2000*, San Francisco, June 2000.
12. Gregor Von Laszewski, et, and al. Cog high-level components. White paper, Argonne National Laboratory, 2000.
13. A.W. van Halderen. User guide for vlab developers. Technical Report UvA/VLab-AM/TN05, University of Amsterdam, 2000.
14. David Groep, Jo Van Den Brand, Henk Jan Bulten, Gert Eijkel, Massi Ferro-Luzzi, Ron Heeren, Victor Klos, Michael Kolsteini, Edmar Stoks, and Ronald Vis. Analysis of complex surfaces in the virtual laboratory. Technical Report -Draft Version-, national Institute for Nuclear and High Energy Physics (The Netherlands), 2000.
15. A. Abd-Allah and B. Boehm. Reasoning about the composition of heterogeneous architecture. Technical Report UCS-CSE-95-503, University of Southern California, 1995.
16. David Garlan, Robert Allen, and John Ockerbloom. Architectural mismatch: Why reuse is so hard. *IEEE software*, Vol. 12(No. 6), November 1995.
17. Rich Wolski, Neil T. Sprin, and Jim Hayes. The network weather service: A distributed resources preformance forecasting services for the metacomputing. Technical Report TR-CS98-599, UCSD, 1998.
18. GlobusProject. Gridftp: Universal data transfer for the grid. White paper http://www.globus.org/datagrid/deliverables/default.asp, Grid Forum Data Acces Working Group, 2000.
19. Kim Potter Kihlstrom, Louise E. Moser, and P. M. Melliar-Smith. The secure ring protocols for securing group communication. In *Proceedings of the IEEE 31st Hawaii International Conference on System Sciences*, pages 317–326, Kona Hawaii, January 1998.