ICES/KIS Project No. 1544516

Title: **The Virtual Lab-AM RTS**

Authors: **VLAM-G-AM Group**

Date: **18th April 2001**

Document Code: **/UvA/VLAM-G-AM/TN01**

Version: **1.1**

Status: **Proposal**

Technology & Scientific Center
WATERGRAAFSMEER

The Virtual-Lab Amsterdam RTS (Design document)

Adam Belloum, Zeger W. Hendrikse and David Groep

document version 18th April 2001

Note to the readers:

- The status of this document is a proposal, *i.e.* it may be considerably changed in the future due to remarks of its readers. It provides a starting point for discussion on the design of the VLAM-AM RTS and may point to future challenges.
- This report aims also at providing enough references needed for the implementation of the VLAM-AM RTS.
- While reading this short report the reader is invited to give his comments and remarks. We especially welcome comments from VLAM developers on how they see their developments integrated in the VLAM AM-RTS, or which services they need from the VLAM AM-RTS.
- Notation: if *?? sentence/word ??* is found in the text it means that this sentence/word might be not appropriate to describe the idea being discussed.
  The color used in the figures represent different components
  - Yellow: Globus toolkit components;
  - Red: packages developed for Globus;
  - Gold: Database components;
  - Light Blue: Module developed
  - Brown: Module to be developed
  - Cyan: shared elements, domains etc.;
  - Green: Potential software for the implementation;
- Web pages where the reader can find more in formation on the VLAM Abstract Machine developments:
  - VLAM-AM Module skeleton API User Guide:
    http://www.science.uva.nl/˜adam/psfiles/VlAPI.ps
  - http://www.nikhef.nl/˜davidg/vlab/
  - http://www.science.uva.nl/˜zegerh/work.html
  - http://www.science.uva.nl/˜adam/currentResearch.html
- This document is the result of VLAM-AM design meetings. During this project, many people have been contributed. This makes it almost impossible to mention all of them. However, we do want to name A.W. van Halderen, F. van der Linden, V. Klos, D. Groep, A. Belloum and Z.W. Hendrikse.

Abstract

This document introduces the Virtual Laboratory Amsterdam (VLAM-G) Abstract Machine Run-Time System (VLAM-G-AM RTS) and its design. The VLAM-G-AM RTS is middleware, which enables the end-user to run experiments within the VLAM-G. As such, it acts as an intermediate layer between the grid infrastructure (Globus) and the VLAM-G users, hiding details on grid technology which are irrelevant to the users carrying out experiments. The VLAM-G-AM RTS provides services such as scheduling and running modules developed by VLAM-developers and updating internal parameters on the fly.

1 Introduction

The aim of the Virtual Laboratory is to develop a platform for scientists and engineers to carry out experiments making optimum use of modern Information Technology. This platform makes use of the most recent developments in Grid-computing, and should provide the end-users a flexible and transparent modularized data-flow based architecture. In particular, the VLAM-G will focus on three application domains:

- physics,
- biology
- and systems engineering.

In principle, the modules processing the data-flow have a strong analogy to those used in IRIS explorer^{TM}, see `http://www.nag.co.uk/Welcome_IEC.html`. Thus the topology of an experiment may be represented by a data-flow graph (DFG).

In the context of the Virtual Laboratory, experiments are defined by selecting a number of processing elements (also referred to as modules) and are connected in a such a way to represent a data-flow. Processing elements[*] are independent of each other and have been developed by different persons in general. Therefore the VLAM-G-AM RTS has to offer

- a transparent layer which communicates data between these processing elements and
- an platform enabling the modules to execute their data processing and computations.

In terms of a multi-tier architecture, the VLAM-G-AM RTS is just another specialized piece of middleware, which enables the end-user to run VLAM-G specific experiments. Let us for the moment assume a generic 4-tier architecture, then the VLAM-G-AM RTS can be positioned in the third tier, as indicated in Figure 1.

Typical issues which need to be considered during design and implementation of this VLAM-G-AM RTS are:

- security,
- domain restrictions,
- QoS (reliability issues).

In the next section, we are going to discuss some design issues, introducing the Globus Toolkit as a starting point. Thereafter we will focus on the Grid-aspects of the VLAM-G-AM RTS and we will end with a summary and conclusions.

[*]Separate modules may be grouped together, to form a larger so-called super-module.

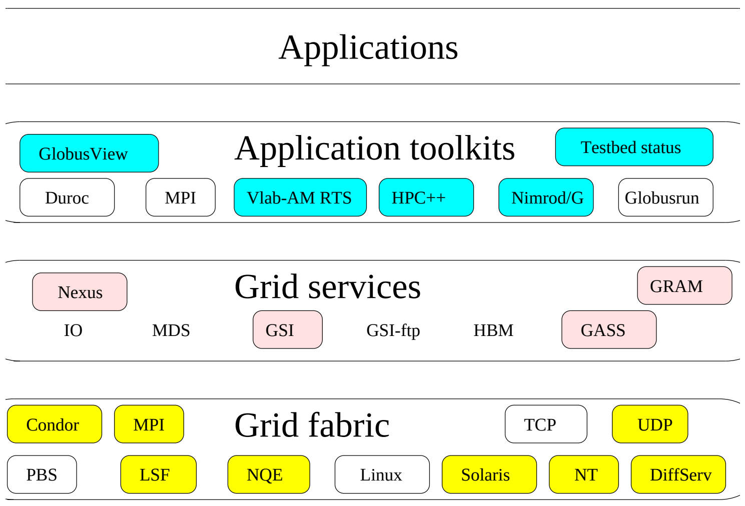


Figure 1: *Schematic overview of the position of the VLAM-G-AM RTS in the commonly agreed on 4-tier Globus/Grid architecture.*

2 Design and implementation considerations

When studying a real scientific experiment (for instance, the microbeam experiment at NIKHEF), it becomes clear that communication mechanisms needed for the VLAM-G-AM RTS have to support at least three types of interactions:

1. interactions among processing elements located on the same machine,
2. interactions within a local area network and
3. interactions with external devices (Figure 2).

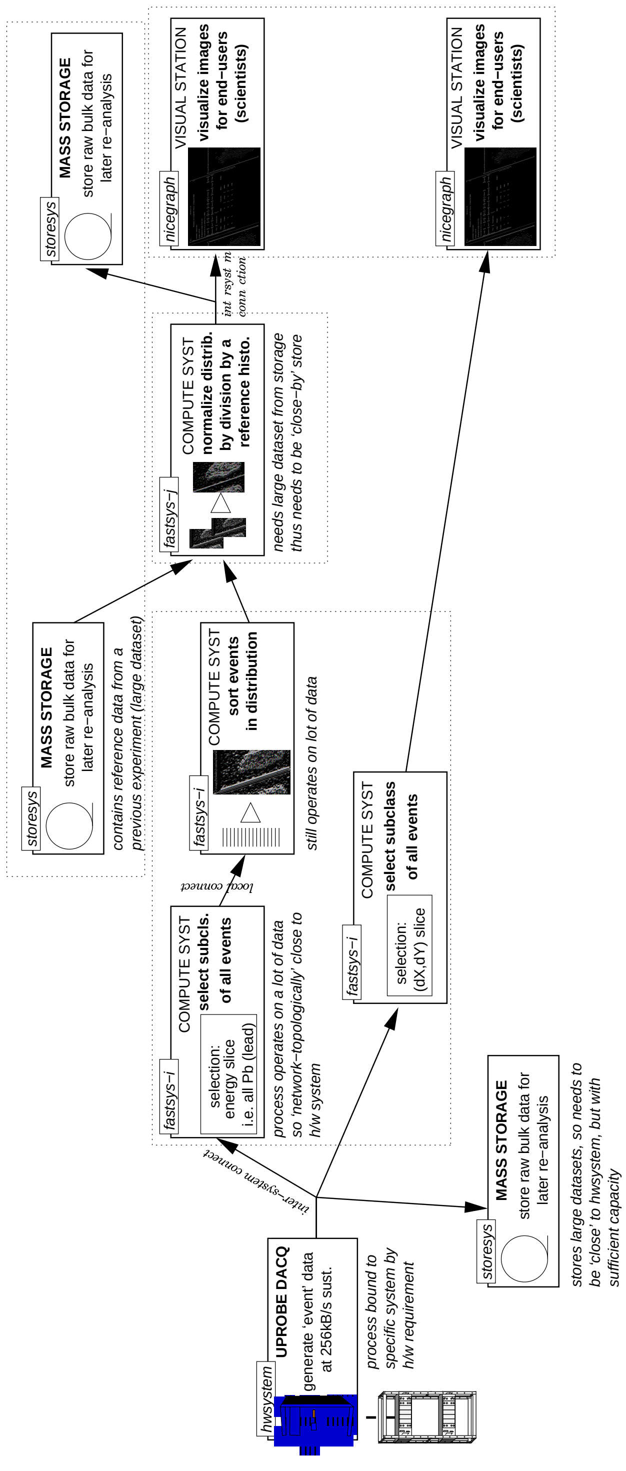
In this particular experiment, data are generated at the microbeam (256 Kb/s) and analyzed real-time by various processing elements distributed on heterogeneous computing resources. The microbeam experiment has not only shown the need for point-to-point connections, but also the need for connections shared by several modules, such as a fan-in and fan-out.

As mentioned before, processing elements are developed independently in general. Communication takes place via connections, which in principle consist of mere abstract locations at the side of a module. These locations may either be assigned input or output data streams. In order to make sure that only proper connections can be established in such a scheme, the the processing elements need to communicate via a strongly typed communication mechanism. When using a strongly type communication scheme, each module may have defined a number of input/output ports: only a predefined type of input/output stream is allowed to connect to a particular input/output port.

Summarizing, the data type constitutes the interface with which developers of the modules can communicate with the outside world, *i.e.* with which their module is going to interact with other processing elements. This will eliminate the possibility of an *architectural mismatch*, a well know problem in software composing design [1, 2].

The implementation of the VLAM-G-AM RTS would potentially suffer from a rather tedious and long time needed for development, in case we would have to start from scratch. Fortunately, existing Grid technologies provide a suitable low-lever layer for such a development. In particular, the Globus toolkit provides different secure and efficient communication mechanisms which can be used as basic building blocks for an implementation of the VLAM-G-AM RTS communication scheme (see Section 3 for more details). The choice for the

Globus toolkit is further motivated by the fact that nowadays it is the *de facto* standard in Grid-computing.



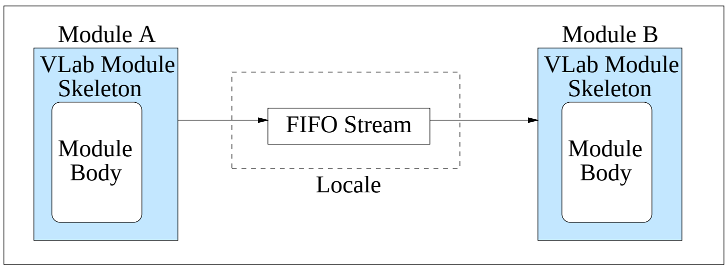
storesys
MASS STORAGE
store raw bulk data for later re-analysis
nicegraph
VISUAL STATION
visualize images for end-users (scientists)
nicegraph
VISUAL STATION
visualize images for end-users (scientists)
int rsyst m conn ction
fastsys-i
COMPUTE SYST
normalize distrib. by division by a reference histo.
needs large dataset from storage thus needs to be 'close-by' store
storesys
MASS STORAGE
store raw bulk data for later re-analysis
contains reference data from a previous experiment (large dataset)
fastsys-i
COMPUTE SYST
sort events in distribution
still operates on lot of data
local connect
fastsys-i
COMPUTE SYST
select subcls. of all events
selection: energy slice i.e. all Pb (lead)
process operates on a lot of data so 'network-topologically' close to h/w system
fastsys-i
COMPUTE SYST
select subclass of all events
selection: (dX,dY) slice
storesys
MASS STORAGE
store raw bulk data for later re-analysis
stores large datasets, so needs to be 'close' to hwsystem, but with sufficient capacity
inter-system connect
hwsystem
UPROBE DACQ
generate 'event' data at 256kB/s sust.
process bound to specific system by h/w requirement

Figure 2: *The Micro-beam experiment*

Using the Globus toolkit components as the core technology for the design of the VLAM-G-AM RTS, we take advantage of all the new software currently being developed around it. For instance, a typical task the VLAM-G-AM RTS has to perform is the mapping of data-flow graph (DFG) onto the available computing resources. Such a task can be elevated using the Globus resource management module MDS (Meta Directory Service [†]). In addition, tools developed for forecasting the network bandwidth and host load such as the *Weather Forecast service* [5] may be incorporated.

As a first step towards the implementation of the VLAM-G-AM RTS, it has been decided to use a *port*-oriented approach. In this approach each processing element (PE) must specify a number of ports. When an output port of PE A is connected to an input port of PE B, the VLAM-G-AM RTS assigns a data-stream channel to transfer data from A to B (Figure 3).

The basic mechanism used to connect ports is a simple FIFO data channel. The FIFO streams are sequential data streams with one producer and one consumer. Ports and communication channels are typed, in order to assure a single data type per port or channel only. Connections are established between ports of the same or equivalent type, see [4].



The experiment as described by the end-users (Location of the resource is hiden)

Figure 3: *VLAM-G-AM RTS architecture for local communication. Each modules is implemented based on a pre-defined module-skeleton. This enforces uniformity within the VLAM.*

This communication mechanism needs to be extended in order to meet the requirements of the experiments carried out within the VLAM: multi-consumers may access the same communication channel in parallel. To support such a feature, the system has to be extended to a multiple fan-out FIFO. A strait forward solution is to write specialized modules that do nothing but de-multiplexing the data stream they received as input (Figure 4).

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

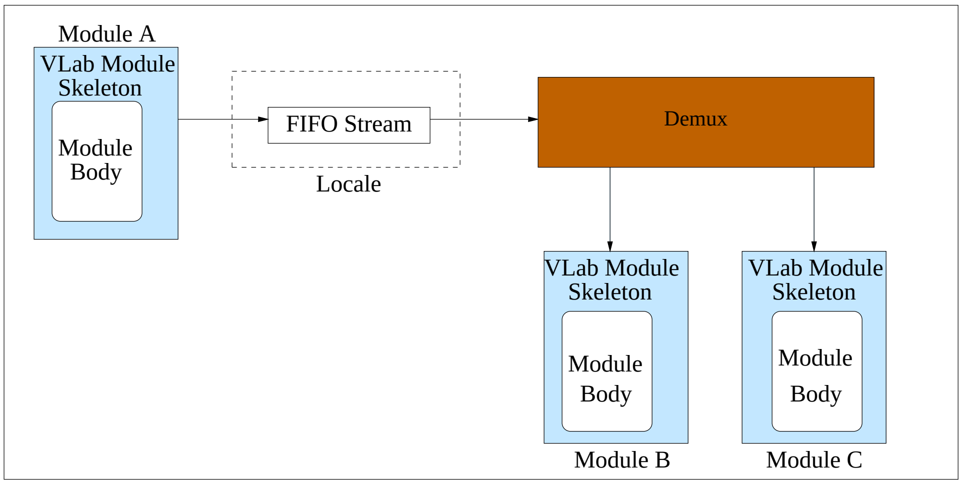


Figure 4: *VLAM-G-AM RTS module used to duplicate a stream of data.*

However, the de-multiplexing solution proposed here has one major drawback. It requires a separate implementation for each data type used. Therefore, this can only be used as a temporary solution.

Properties of connections should be transparent to the end-user *i.e.* there should be no difference between local and remote connections. The VLAM-G-AM *scheduler* maps the DFG representing the VLAM-experiment (PE's and connections) onto the available computing resources (Figure 5), thereby using information gathered form the MDS and the AM-Kernel database [].

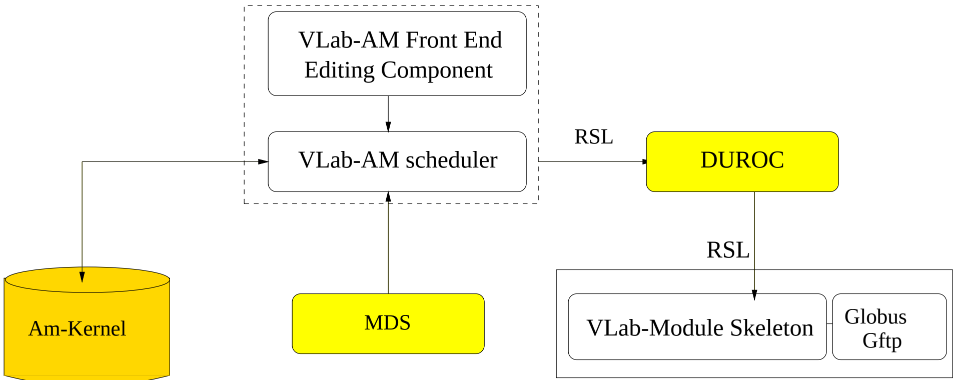


Figure 5: *VLAM-G-AM*

3 VLAM-G-AM RTS on the Grid

Implementation of the VLAM-G-AM RTS on a stand-alone resource can either be based on sockets or on UNIX IPC between processes/threads, *e.g.* shared memory. However, using Globus Toolkit components restricts the freedom to basically two alternatives:

- Globus I/O[‡],
- GridFTP[3].

Let us for a moment elaborate on these two alternatives.

3.1 Globus I/O

Globus I/O closely mimics the behaviour of conventional socket communications (Figure 6). The Globus I/O library takes care of most of the details of socket setup and attribute specification. The Globus GSI (Grid Security Infrastructure) is based on SSL (secure socket layer). This implies that all communication is encrypted and authenticated. Globus I/O provides a familiar abstraction mechanism for handling files[§].

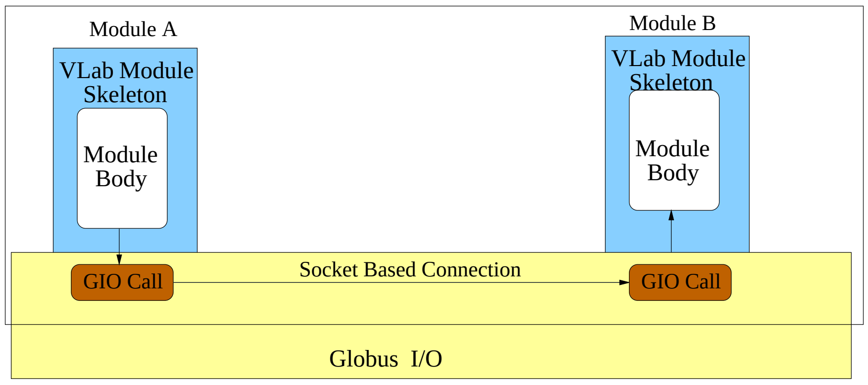


Figure 6: *VLAM-G-AM RTS using GIO (a socket based connection)*

Advantages of Globus I/O are:

- It hardly requires any additional implementation within the modules, in particular in the module-skeleton.
- It requires a minimum amount of overhead.

A obvious disadvantage is the connection to remote databases: a separate protocol has to be introduced at this point. This is actually what leads us to consider in the next section, the GridFTP API.

3.2 GridFTP

Although GridFTP is a more extensive protocol for data transfer, it includes many features from which we might profit. GridFTP is based on the FTP protocol as defined in RFC 959 [¶]. It focuses is on high-speed transport of large datasets or equivalently a large number of small datasets. New features include seamless parallelism (multiple streams implementing a single stream) and automated window size tuning. All data transfer is memory-to-memory based.

[‡]http://www.globus.org/v1.1/io/globus_io.html
[§]Security, socket options and QoS are handled using attributes.
[¶]http://www13.w3.org/Protocols/rfc959/

Using GridFTP as a data transport mechanism in the VLAM-G-AM RTS has the following advantages:

- high-performance data transfer across heterogeneous systems,
- third party arbitrated communication (in our case, the AM RTS acts as an arbitrator in the module-to-module communication protocol),
- uniformity between modules and HPSS's producing data (the latter are supposed to be running GridFTP anyway).
- Starting a data transfer from the AM RTS can be done through a single `globus_gass` call (*globus_gass_url_to_url_copy*).

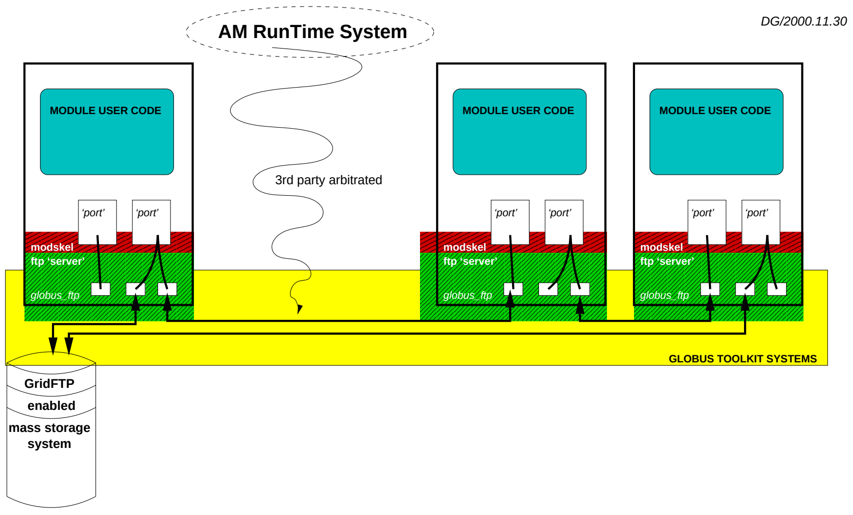


Figure 7: *VLAM-G-AM RTS using Gftp*

On the other hand, some drawbacks are associated to the GridFTP API as well:

- GridFTP induces more overhead for communication on a single resource.
- Every module (skeleton) should include an implementation of a simple GridFTP daemon. This daemon will be contacted by the AM RTS to start inter-module data transfers.

The ports should have symbolic names to refer to using the gsiftp URL. The Globus_FTP API already provides a large number of functions to facilitate setting up such a server, such as predefined listeners and data connections.

4 Conclusion

The design of the VLAM-G-AM RTS presented in this report focuses on the communication mechanisms needed to support data transfers within the VLAM-G working environment.

The problem of the 'fanning-out' of data-streams has been addressed and a temporary solution has been proposed. Since a major drawback has been indicated, this issue still needs to be paid proper attention to.

We also considered two possible mechanisms to implement the inter-module data transfers on a computational Grid. Although the GridFTP alternative requires some more efforts to implement, it is still worth considering, because it is more rewarding in the end. Currently a first prototype is being implemented to get some hands on experience with this technology (Contact `davidgnikhef.nl` for more information) or refer to [3] for more details. If human resources permit, both alternatives might even be implemented, and their performance compared.

At this stage of the design, we did not address any performance issues. Emphasis has been put on the identification of the most prominent problems and on finding simple feasible solutions. Because of the current limited amount of human resources, it is suggested first to focus on a simple and easily extendible (*i.e.* modularized) working implementation, and extend and improve on it afterwards.

As a first step towards an implementation the user could *e.g.* select the RSL (resource specification language) entries in the modules, thereby specifying the computational resources himself. This would make the implementation of the scheduler straight-forward.

References

[1] A. Abd-Allah and B. Boehm. Reasoning about the composition of heterogeneous architecture. Technical Report UCS-CSE-95-503, University of Southern California, 1995.

[2] D. Garlan, R. Allen, and J. Ockerbloom. Architectural mismatch: Why reuse is so hard. *IEEE software*, Vol. 12(No. 6), November 1995.

[3] 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.

[4] A. van Halderen. Typing the data-steams between modules in the vlab. Working Memo No 01, University of Amsterdam, May, 2000.

[5] R. Wolski, N. T. Sprin, and J. Hayes. The network weather service: A distributed resources preformance forecasting services for the metacomputing. Technical Report TR-CS98-599, UCSD, 1998.