VXDL: Virtual Resources and Interconnection Networks Description Language

Guilherme Koslovski[1,2], Pascale Vicat-Blanc Primet[1], and Andrea Schwertner Charão[2]

[1] INRIA - École Normale Supérieure (ENS) Lyon
[2] Laboratório de Sistemas de Computação (LSC)
Universidade Federal de Santa Maria (UFSM)

Abstract. Data grid applications require often an access to infrastructures with high performance data movement facilities coordinated with computational resources. Other applications need interconnections of large scale instruments with HPC platforms. In these context, dynamic provisioning of customized computing and networking infrastructure as well as resource virtualization are appealing technologies. Therefore new models and tools must be studied and developed to allow users create and handle such on-demand virtual infrastructures within grid platforms or even within the Internet. This work presents VXDL, a language for virtual resources interconnection networks specification and modeling. Besides allowing end resources description, VXDL lets users describe the desirable virtual network topology, including virtual routers and timeline. In this paper we motivate and present the key features of our modeling language. We explore typical examples to demonstrates the expressiveness and the pertinence of it. Then we detail experimental results based on the execution of NAS benchmark on virtual infrastructures, conforming different VXDL specifications.

Key words: virtual grids, network virtualization, description language

1 Introduction

Data grid applications require often an access to infrastructures with high performance data movement facilities coordinated with computational resources, while other applications need interconnections of large scale instruments with HPC platforms. Computational grids offer a distributed infrastructure of hardware and software, comprising computers and clusters of computers interconnected and geographically distributed [12]. The resources available in a grid can be shared by multiple users for different computational purposes, such as high-throughput computing, on-demand computing, data intensive and distributed supercomputing. In this context, dynamic provisioning of customized infrastructure combined with end resources and network virtualization are appealing technologies.

To allow efficient and flexible resources sharing in grids, projects like VGrADS [14] Virtual Workspace [20], VioCluster [19] are exploring the use of resources virtualization. On an other hand projects like G-Lambda [22], Phosphorus [3] or UCLP [4] are exploring end-user level dynamic lambda path or bandwidth provisioning for the creation of customized high speed networking environments.

However, modeling and specifying these type of resource interconnection is still an open issue. In such context users should be able to specify what are the resources needed to run their applications. When environments are complex, a language, which exposes a set of parameters for describing resources to be composed, is required. Several languages for computer resources description and selection like ClassAd, vgDL or SWORD [18, 9, 17] have been proposed. These languages differentiate by their grammar, the proposed parameters and the implementation specificities. But the interconnection network which plays a central role in the distributed computing infrastructure, cannot be finely modeled by these languages. On an other hand, in the context of networking capacities provisioning, few languages have been proposed, such as the NDL [7] based on XML/RDF. But they do not include fine end resource description nor virtualization constraints specification. Standardisation work is also ongoing in groups like OGF NML-WG [21]. To include virtualization constraints, some extra parameters are required in the specification. For example, virtualization enable to split the same physical resource (computational or network resource) and shared it in different virtual entities. Thus, the user has to be able to specify the maximum number of virtual resources on the same physical entity he wants.

We propose *VXDL*, a language for virtual resource interconnection network specification, which allows a detailed and extensible definition of all components of a virtual infrastructure. The first level goal of *VXDL* is to integrate the *network interconnection*, the *virtualization constraints* and the *timeline* with the classical resources description. *VXDL* has been defined for the HIPCAL [2] and CARRIOCAS [1] projects.

This paper is organized as follows. Section 2 presents an example and the characteristics of virtual infrastructures. In the section 3 the language VXDL is described, the grammar is illustrated on uses case. To evaluate the ease of use, we performed an analysis of the available specification properties and compare them with those of other existing languages. We then investigate the improvement obtained with the specification of network topology and parameters such as latency and bandwidth, through the analysis of the total execution time of the benchmark NAS [6]. Section 4 shows the results obtained with the execution of some experiments on the Grid5000 platform. Section 5 reviews related works. The conclusions and perspectives are given in section 6.

2 Virtual Infrastructure definition

We define an infrastructure as the structured aggregation of computing resources. Thus, a virtual infrastructure represents the aggregation of virtual resources through an organized interconnection network. The virtualization layer enables an efficient separation between the applications specifications and physical resources. OS-level virtual machines paradigm is becoming a key feature of Grid as it provides a powerful abstraction. It has the potential to simplify management of resources and to offer a great flexibility in resource usage. Each VM a) provides a confined environment where non-trusted applications can be run, b) allows to establish limits in hardware resource access and usages, through isolation techniques, c) allows adapting the runtime environment to the application instead of porting the application to the runtime environment (this enhance the application portability), d) allows using dedicated or optimized OS mechanisms

(scheduler, virtual memory management, network protocol) for each application, e) applications and processes running within a VM can be managed as a whole. The abstraction of the hardware enables to create multiple, isolated and protected virtual clusters on the same set of physical resources by sharing them in time and space. In other words, with representation in Virtual Machines (VM), it is possible that a physical resource (node) hosts VMs of different virtual clusters. A virtual cluster [2] is defined as a group of machines (physical or virtual) configured for a common purpose. As Grid computing is about using multiple sites spread over wide area, the virtual cluster abstraction is not sufficient to solve the network QoS and secure communication channels issues raised in Grids.

This inability to enforce network QoS may prevent this approach from being useful in a range of scenarios. A virtual private network is classically provisioned over a public network infrastructure to provide dedicated connectivity to a closed group of users. Resource-guaranteed VPNs can be obtained from a carrier but are generally static and require complex service level agreements (SLAs) and overheads. Tunneled VPNs such as those in use over the Internet are more lightweight but offer low performance and no assured quality of service. Functionality necessary for automating dynamic provisioning of lightpath or virtual private network capacities are emerging. The originality of our approach is to combine OS-level machine virtualization with network virtualization and bandwidth reservation through service overlays which can be based on router virtualization.

Grid applications are characterized by the fact they may transfer large subsets of data across network for processing while exchanging very small and urgent messages. In most cases, grid resource scheduler allocates precious computing and storage resources first, then generates output as bulk data transfer requests. The volume of dataset is determined from task specification, and a deadline may be specified to enforce the efficient use of expensive grid resources (CPUs, disks) as well as to shorten the task completion time. Low latency interprocess communications (MPI), sharing the same wide area networking infrastructure meet also strong QoS requirements and may benefit from end to end bandwidth reservation and isolation service. Such a service can be implemented with the overlay network and router virtualization approach. An overlay network has an ideal vantage point to monitor and control the underlying physical network and the applications running on the VMs. The overlay, being a virtual network can run on a network that provides reservations or ligh-path setup and teardown in an optical network. Router virtualisation enable the customization of packet routing, packet scheduling and traffic engineering for each virtual network crossing it. The substrate to build such dynamic, predictable and large-scale computing environments is under development within the HIPCAL project [2]. We concentrate here on the virtual infrastructure description language defined within this project and demonstrate the importance of interconnection network description and control.

3 VXDL grammar

VXDL has been defined for specifying an interconnection of virtual resources into a virtual infrastructure. In this context, the process of resource specification and selection

has some particular features, when compared to conventional grids. It is expected that a language should allow the user to describe:

– individual resources and groups;
– the elementary function assigned to the resource, which can be attributed to a single component or a cluster, for example: request of *computing* nodes, *storage* nodes, *visualization* nodes, etc.;
– the network topology, including virtual representations for routers and characterization of the necessary links, such as node-node, intra-cluster, cluster-node, node-router, etc. in terms of QoS metrics;
– the applications and tools needed for each component (operating systems and programming tools, for example);
– the executing timeline of the application (this parameter should help in resources co-scheduling).

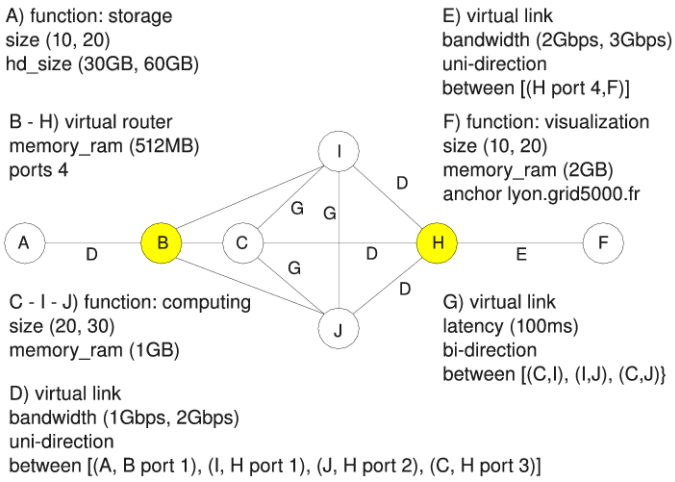


Fig. 1. Graph representing an infrastructure.

Figure 1 shows a graph representing a typical infrastructure, composed by two virtual routers that perform the interconnection among multiple virtual clusters, with different configurations. The representation of the network topology define the shape of the infrastructure. In the figure 6 the grammar is divided in *virtual grid, virtual topology* and *virtual timeline*. With the first part, *virtual grid*, it is possible describe all necessary components (nodes and clusters) and create groups among them. Here, was used the idea of identification by *elementary functions* of a component, where the user can describe what is the use of the resource, for example: *computing, storage, network_sensor, visualization, aquisition, router and endpoint*. An example of its use can be seen in the request of a cluster for *computing* and *storage*, or the definition of a node for *visualization* and *network endpoint*.

A specified virtual infrastructure can be instanciated by the allocation of geographically distributed resources. The option *anchor* allows the user to specify the physical location where the component (node or cluster) should be reserved. This parameter helps in modeling environments where there is a local dependency for execution, such as specific components for result visualization or a particular input database.

The second part of the grammar, *virtual topology*, allows the user to specify the network topology desirable to connect all components of the infrastructure. The specification can detail internal parameters in clusters or between clusters, applying parameters as latency and bandwidth to all links needed. Through the identification of source and destination, the same component may have different communication channels. The goal of detailing the network topology is to allow users requesting a virtual infrastructure for communication-sensitive applications, i.e. applications that require transmitting a high data volume or that benefits from low latency. Using the topology specification, such applications can ask for an environment with skillful network links. Is important to note that VXDL allows to express the virtual network components and topology, but is not expressing explicitly any mechanism for physical provisioning or of any information on the pricing scheme. These *physical* and *economical* parameters are defined as external parameters and can be joined to the specification.

Figure 2 shows two graphs, *G(V, e)*, which represent different infrastructures for execution. In these graphs, the vertices represent the necessary components and the edges represent the communication links between them. Vertex can be nodes (endpoints), clusters or routers. The graph in Figure2.a represents a topology where components communicate directly with each other. This figure may represent the communication required between four clusters, for instance. In Figure 2.b, we represent a topology where there are routers (in yellow) between components. In both examples, the user can specify separate parameters for each communication link.

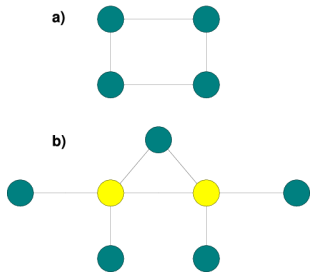


Fig. 2. Graphs of the network topologies.

The specification of a *virtual timeline* helps in identifying the moment when the resources are needed. This description asks resources for a certain time, or wait the execution of a stage to start another. This type of notation is usual in workflows, which are used to identify and to map the resources needed for executing an application in a grid. Definitions as *start*, *after* and *until* allow the creation of a timeline for execution, requesting resources for certain time intervals. The approach implemented is the nominal identification of time variables to allow reference in future rules.

3.1 Example

To study and elaborate a specification using VXDL, we select an application named VISUPIPE (High Performance Collaborative Remote Visualisation), which is used to view images in real time. This use case is developed within the CARRIOCAS project. This application generate images of 20 Mpixels x 32 bits x 30 images/s x 2 for stereo. This represents a bandwidth requirement of 38,4 Gb/s when a researcher wants to explore the data and images on the visualisation wall (12). Low latency is also required for interactivity. Figures 10 and 11 shows a pipeline and an environment necessary to make a view session. The five clusters described perform the storage, initial computing (filters), mapping, image formation and final visualization. In this pipeline the network is a critical part, so there is a clear need for specifying network topology requirements.

Figure 7 shows the corresponding resource query using VXDL. In addition to parameters for composition and elementary functions identification, there is a request for a time to use the resources and specific clusters interconnection. Using the *start* and *for* parameters, the user informs the time for execution and the period for make the resources reservation. In the network topology requested, we described the configuration for the links *bandwidth1* and *bandwidth2*, informing the minimum and maximum bandwidth for each. Yet, this request informs the maximum tolerable latency on some connections, intra and inter clusters.

In the execution timeline for VISUPIPE, we indicate virtual times (t0, t1, t2 and t3) that may be referenced in future rules. The values specified for computing time and data transfer volume come from the application description. In this timeline, features such as *Bandwidth2* and *Cluster_Visual* should be available only after the execution of previous tasks. This kind of description can help the schedule in the resources distribution among the existing virtual infrastructures, allowing to make reservations only when necessary.

4 Evaluation

The evaluation we performed aimed a quantitative analysis of the benefit obtained with a detailed virtual infrastructure specification, more precisely the network topology parameters provisioned by VXDL. For this, we selected the NAS benchmark [6] for a execution in different virtual infrastructures allocated on the test environment Grid5000 [1]. The Numeric Aerodynamic Simulation (NAS) parallel benchmarks is a set of applications developed by NASA, which have been grouped in order to form a benchmark for parallel architectures. For this analysis, we selected only four applications from the package (cg, is, lu and mg), due to their different characteristics of communication, memory occupation and processing [11].

In the virtual environment, we used Xen version 3.1. The virtualized operating systems were based on Debian Sid Linux distribution. For benchmarks execution, we used MPICH [10]. All results were obtained from the average of 10 executions of each test. The resource selection in a Grid5000 was performed manually: starting from informed requests, the resources that meet with the request were selected and allocated for execution.

[1] https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home

4.1 Latency specification

We prepared four queries requesting the virtual cluster creation:

Query without network topology, 1 VM per node: This query, represented by Figure 8.a in the annex, asks for a virtual environment composed of at least 16 and no more than 20 nodes, classified as computing nodes. It also asks for a minimum of 1GB of memory RAM. The resource selection resulted in the allocation of 16 nodes, 8 per each physical cluster.

Query with latency (0.100ms), 1 VM per node: In this request, Figure 8.a, we used the same nodes parameters defined in the description above, adding the network topology. It asks for interconnected resources in a network with maximum latency of 0.100ms, resulting in the selection of resources with geographic proximity, allocated in the same physical cluster.

Query with latency (0.100ms), 2 VMs per node: In the description presented in Figure 8.c we requested for a virtual environment consisting of nodes with 256MB of memory RAM. This allows the allocation of multiple virtual machines on a single physical resource. The network description indicates that the resource allocation occurs on the same physical cluster. It was selected 8 physical computers, allocating 2 virtual systems in each node.

Query with latency (10ms), 2 VMs per node: In this query (Figure 8.d) the parameters for network topology ask for links with maximum latency of 10ms. This way, a selection results in the allocation of physical nodes in different clusters. The RAM and CPU configurations allow the allocation of multiple virtual machines on the same physical resource. The selected virtual environment was composed by 8 nodes, distributed between two clusters. In each node it was allocated 2 virtual operating systems.

Table 1 and Figure 8 show the results obtained with the execution of NAS applications in virtual infrastructures. It is possible to observe in the total execution time the direct influence of resources location. Descriptions that informed the network topology desirable obtained a lower execution time in all applications. Comparing the queries *Without network topology* and *With latency specification, 0.100ms* it is observed that in applications such as cg, is and mg, that perform a high number of communication, the total execution time is 6 times, 26 times and 7 times lower, respectively.

Table 1. Results of NAS benchmark with different latency specifications.

NAS	N.I.	0.100ms	0.100ms	10ms
	1VM/node	1VM/node	2VMs/node	2VMs/node
cg	893s	137s	220s	860s
is	182s	7s	11s	171s
lu	191s	63s	93s	209s
mg	36s	5s	13s	38s

Even in figure 8, it is observed that the allocation of two virtual machines on a unique physical resource results in a overhead (description *With latency specification, 0.100ms, 2 VMs per node*). Is important to discuss the fact that in allocations of multiple

virtual machines on the same physical resource should respect the queries demands, that is, there must be guarantee the provision of solicited resources (nodes and network).

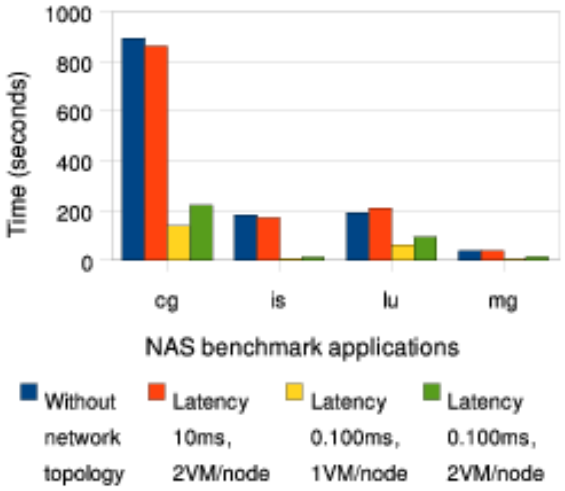


Fig. 3. Results of NAS benchmark with different latency specifications.

4.2 Bandwidth specification

For accomplishment of this tests, a virtual infrastructure was requested resulting in the selection of 18 nodes located one the same physical cluster (as specified for the *anchor* parameter). The VXDL specification (Figure 9) describes the execution infrastructure where we represented two clusters, interconnected by two routers. In the graph in Figure 4), the vertices are represented by resources (green vertices are the computing resources and yellow vertices are the routers) and the edges by the network topology.

The resources selection resulted in the allocation of a virtual machine in each physical computer. The routers were carried in software, through a specific virtual machine (as the specification). For the execution of the tests, we selected three virtual infrastructures with the same specification described, modifying the desirable bandwidth between the routers through the alteration of the parameter *bandwidth* on the link called *Router_to_Router*. The values used had been: 1Gb/s, 100Mb/s and 10Mb/s. To limit the bandwidth between the routers was used the Traffic Control (tc) tool, which informs kernel parameters about the performance of network interfaces. The cross-traffic was generated using the *iperf* tool, performing the total use of the available bandwidth (in the results this scenario is identified by the word ”cross-traffic” on subtitles).

The results of NAS benchmark execution in the three infrastructures elaborated are shown in the figure 5 and in the table 2. Analyzing the execution time taking with base the result obtained in 1Gb/s network, is observed in the execution on 100Mb/s

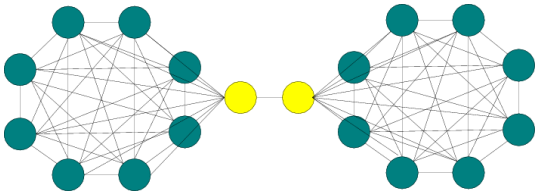


Fig. 4. Network topology for bandwidth specification.

network (cross-traffic 900Mb/s) that the applications CG, IS, LU and MG increased the execution time in 59%, 400%, 90% and 33% respectively. The comparison between configurations 100Mb/s and 100Mb/s (with cross-traffic) can represents a impact of a possible channel division between different virtual infrastructures. It is observed that the execution time increases 1.60, 1.41, 1.79 and 1.90 for the applications CG, IS, LU and MG, respectively.

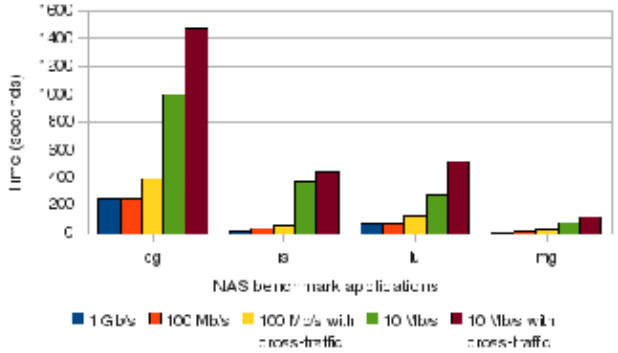


Fig. 5. Results of NAS benchmark with different bandwidth specifications.

The results of these experiments help to note that in the case of virtual channels, often shared among several virtual infrastructures is important to observe that the correct specification of the bandwidth and the ensuring by the manager system is essential to obtain satisfactory results. Is it possible to do the sharing of a communication channel between various environments, respecting the requirements specified by each application.

5 Related works

The subjects of resource description/reservation in grids and description of network topologies have been studied in previous works [18, 17, 9, 16, 5, 15, 13, 7, 8].

To analyze the existing languages more deeply, we selected ClassAd [18], vgDL [9] and SWORD [17]. We used these languages in the same case (VISUPIPE) presented in

Table 2. Results of NAS benchmark with different bandwidth specifications.

NAS	1Gb/s	100Mb/s	100Mb/s	10Mb/s	10Mb/s
cross	0	0	900Mb/s	0	990Mb/s
cg	248s	247s	395s	1006s	1476s
is	11s	39s	55s	380s	448s
lu	65s	69s	124s	270s	522s
mg	9s	11s	21s	81s	118s

the previous section. Using the ClassAd language, we can not easily represent groups of resources (with different specifications) that should run simultaneously. To do so, we represented each cluster as an independent *job*, with all clusters having the same *owner* to allow interaction among the independent *jobs*. Another difficulty is that ClassAd does not allows the network topology specification (parameters such as latency and bandwidth are not supported).

Using SWORD, we were able to specify more precisely the necessary infrastructure and some network parameters, as latency and bandwidth. But in the network topology, SWORD does not offer resources for a full representation, including virtual routers, for example. Moreover, we observed that the specification of the network is held together with other resources. This makes difficult to specify complex scenarios with many parameters.

The vgDL language proposes the abstraction of the network parameters specification, through the use of definitions as *Close, HighBW, ClusterOf and LooseBag* which, according to the language definition, refer to location and type of components. This kind of specification does not allow the user interact directly with the physical resources, such as informing the desired location of a resource (parameter *anchor* in VXDL). As in SWORD, the user can not represent other components of a virtual network, such as routers and switches.

6 Conclusion

This paper discussed the definition of VXDL, a new language for distributed virtual infrastructures specification. With the grammar proposed is possible make a complete environment description, specifying parameters for components classification. It is also possible specify the network topology desirable, applying rules for each link. The main innovations made by VXDL are the possibility of timeline description and a complete specification of the network topology. Moreover, the representation made using the *timeline* helps the scheduling and more efficient resource exploitation. Also in VXDL, applications and workflows represented by graphs can be easily mapped, where the vertex representation are the resources definitions (group and resource specifications) and the mapping of the edges are the network topology.

The evaluation section discussed the results obtained through the NAS benchmark execution in different virtual infrastructure, composed through VXDL specifications. The results emphasize the importance of correct network parameters specifications.

Acknowledgments

This work has been funded by INRIA and the French ministry of Education and Research via the HIPCAL ANR grant and by the CARRIOCAS pôle System@tic grant. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see https://www.grid5000.fr).

References

1. Calcul Réparti sur Réseau Internet Optique à Capacité Surmultiplée (CARRIOCAS), 2007. http://www.carriocas.org/.
2. HIPCAL project, 2007. http://hipcal.lri.fr/wakka.php?wiki=PagePrincipale.
3. Phosphorus project - lambda user controlled infrastructure for european research, 2007. http://www.ist-phosphorus.eu/.
4. UCLP user-controlled lightpaths, 2007. http://www.canarie.ca/canet4/uclp/.
5. Sergio Andreozzi, Stephen Burke, Flavia Donno, Laurence Field, Steve Fisher ans Jens Jensen Ans Balazs Konya, Maarten Litmaath, Marco Manbelli, Jennifer Schopf, Matt Viljoen, Antony Wilson, and Riccardo Zappi. GLUE Schema Specification. Technical Report 1.3, 2007.
6. D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1991.
7. T. Boku, T. Harada, T. Sone, H. Nakamura, and K. Nakazawa. Inspire: A general purpose network simulator generating system for massively parallel processors, 1995.
8. Roberto Canonico, Donato Emma, and Giorgio Ventre. An xml based network simulation description language, 2001.
9. Andrew Chien, Henri Casanova, Yang suk Kee, and Richard Huang. The Virtual Grid Description Language: vgDL. Technical Report TR0.95, VGrADS Project, 2004.
10. William D. Gropp and Ewing Lusk. *User's Guide for* `mpich`*, a Portable Implementation of MPI*. Mathematics and Computer Science Division, Argonne National Laboratory, 1996. ANL-96/6.
11. Ludovic Hablot, Olivier Gluck, Jean-Christophe, and Pascale Vicat-Blanc Primet. Etude d'implémentations mpi pour une grille de calcul. *RenPar'18 / SympA'2008 / CFSE'6, Fribourg, Suisse*, 11 2008.
12. Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufman, 2004.
13. R. Isaacs and I. Leslie. Support for Resource-Assured and Dynamic Virtual Private Networks, 2001.
14. Key Kennedy, Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, S. Lennart Johnsson, Carl Kesselman, John Mellor-Crummey, Daniel Reed, Linda Torczon, and Richar Wolski. The VGrADS Project, 2003. http://vgrads.rice.edu/.
15. A. Kertész, I. Rodero, and F. Guim. BPDL: A Data Model for Grid Resource Broker Capabilities. Technical Report TR-0074, Institute on Resource Management and Scheduling - CoreGRID, 2007.

16. Chuang Liu and Ian Foster. A Constraint Language Approach to Grid Resource Selection. Technical Report TR-2003-07, Department of Computer Science - University of Chicago, 2003.
17. D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery, 2005.
18. Rajesh Raman, Miron Livny, and Marvin H. Solomon. Matchmaking: Distributed Resource Management for High Throughput Computing. In *HPDC*, pages 140–, 1998.
19. Paul Ruth, P. McGachey, and Dongyan Xu. Viocluster: Virtualization for dynamic computational domains. In *CLUSTER*, pages 1–10. IEEE, 2005.
20. Borja Sotomayor, Kate Keahey, and Ian Foster. Overhead matters: A model for virtual resource management. In *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, page 5, Washington, DC, USA, 2006. IEEE Computer Society.
21. Martin Swany and Paola Grosso. (NML-WG) network mark-up language working group, 2005.
22. Atsuko Takefusa, Michiaki Hayashi, Naohide Nagatsu, Hidemoto Nakada, Tomohiro Kudoh, Takahiro Miyamoto, Tomohiro Otani, Hideaki Tanaka, Masatoshi Suzuki, Yasunori Sameshima, Wataru Imajuku, Masahiko Jinno, Yoshihiro Takigawa, Shuichi Okamoto, Yoshio Tanaka, and Satoshi Sekiguchi. G-lambda: coordination of a grid scheduler and lambda path service over gmpls. *Future Gener. Comput. Syst.*, 22(8):868–875, 2006.

```
<vxdl query> ::= "virtual grid" <name> [<time-to-use>]
        "{" <vg> "}"
<name> ::= <string>
<time-to-use> ::= "start" <date-time> "for" <total-time>
<date-time> ::= <date> " " <time>
<total-time> ::= <number>
<vg> ::= (<resource> | <group>)*
<resource> ::= "resource" "(" <name> ")" "{"
               ["function" <elementary-functions>]
               ["parameters" <resource-parameters>]
               ["software" <software-list>]
               ["anchor" <location>] "}"
<group> ::= "group" "(" <name> ")" "{"
        "size" <value-number>
        ["function" <elementary-functions>
        ["anchor" <location>]
        [<vg>] "}"

<value-number> ::= "(" <number> "," <number> ")" |
    "(" "min" <number> ")" | "(" "max" <number> ")"
<value-freq> ::= "(" "min" <number> "GHz" ")"
    | "(" "max" <number> "GHz" ")"
    | "(" <number> "GHz" "," <number> "GHz" ")"
<value-mem> ::= "(" "min" <number> <men-unit> ")"
    | "(" "max" <number> <men-unit> ")"
    | "(" <number> <men-unit> "," <number> <men-unit> ")"
<value-band> ::= "(" "min" <number> <band-unit> ")"
    | "(" "max" <number> <band-unit> ")"
    | "(" <number> <band-unit> "," <number><band-unit> ")"
<value-lat> ::= "(" "min" <number> <lat-unit> ")"
    | "(" "max" <number> <lat-unit> ")"
    | "(" <number> <lat-unit> "," <number> <lat-unit> ")"
<men-unit> ::= "MB" | "GB" | "TB"
<band-unit> ::= "Kb/s" | "Mb/s" | "Gb/s"
<lat-unit> ::= "us" | "ms" | "s"

<location> ::= <string>
<elementary-functions> ::= <function> ("," <function> )*
<function> ::= "endpoint" | "aquisition" | "storage"
    | "computing" | "visualization" | "network_sensor"
    | "router" "(" "ports" <ports> ")"
<ports> ::= <number>
<resource-parameters> ::= <parameters> ("," <parameters>)*
<parameters> ::= "cpu_frequency" <value-freq>
    | "cpu_mips" <value-number> | "hd_size" <value-mem>
    | "memory_ram" <value-mem>
    | "vms_per_node" <value-number>
    | "cpu_processors" <value-number>
<software-list> ::= <software> ("," <software>)*
<software> ::= <string>

["virtual topology" <name> "{" <links> "}"]
<links> ::= (<link>)+
<link> ::= "link" "(" <name> ")" "{" <link-parameters> "}"
<link-parameters> ::= <link-parameter>
    ("," <link-parameter>)*
<link-parameter> ::= "bandwidth" <value-band>
    | "latency" <value-lat>
    | "between" "[" <components-links> "]"
    | "direction" <direction>
<direction> ::= "uni" | "bi"
<components-links> ::= <pair> ("," <pair>)*
<pair> ::= "(" <component> "," <component> ")"
<component> ::= <name> | <name> "port" <number>

["virtual timeline" <name> "{" (<timeline>)+ "}"]
<timeline> ::= <time-name> "=" (<start> | <after>)
<time-name> ::= <string>
<start> ::= "start" "(" <components-list> ")" [<until>]
<after> ::= "after" "(" <time-name-list> ")" <start>
<components-list> ::= <component-name>
    ("," <component-name>)*
<component-name> ::= <name>
<time-name-list> ::= <time-name> ("," <time-name>)*
<until> ::= (<computation> | <transfer>)+
<computation> ::= "computation" "(" <total-time> ")"
<transfer> ::= "transfer" "(" <value-mem> ")"
```

Fig. 6. BNF specification of VXDL

```
virtual grid VISUPIPE start 19/02/2008 14:32 for 4 {
 group (Cluster_DB) {
   function storage, aquisition
   size (10, 20)
   resource (Node_Cluster_DB) {
     function storage
     parameters hd_size (30GB, 60GB)
   }
 }
 group (Cluster_Filtrage) {
   function computing
   size (20, 40)
   resource (Node_Cluster_Filtrage) {
     function computing
     memory_ram (512MB, 2GB)
   }
 }
 group (Cluster_Mapping) {
   function computing
   size (20, 40)
   resource (Node_Cluster_Mapping) {
     function computing
     cpu_frequency (1.6GHz, 3GHz)
   }
 }
 group (Cluster_Rendu) {
   function computing
   size (10, 30)
   resource (Node_Cluster_Rendu) {
     function computing
     parameters cpu_frequency (1.6GHz, 3GHz),
     memory_ram (512MB, 2GB)
   }
 }
 group (Cluster_Visual) {
   function computing, visualization, endpoint
   size (20, 40)
   resource (Node_Cluster_Rendu) {
     function computing, visualization
     memory_ram (2GB, 4GB)
     anchor "Paris"
   }
 }
}
virtual topology VISUPIPE_Network {
 link (Low_Latency_Intra_Cluster) {
   latency (max 0.01ms),
   between [ (Node_Cluster_DB, Node_Cluster_DB),
      (Node_Cluster_Mapping, Node_Cluster_Mapping),
      (Node_Cluster_Rendu, Node_Cluster_Rendu)]
 }
 link (Low_Latency_Between_Cluster) {
   latency (max 0.05ms),
   between [(Cluster_DB, Cluster_Filtrage),
     (Cluster_Filtrage, Cluster_DB),
     (Cluster_Mapping, Cluster_Rendu)]
   }
 link (Bandwidth1) {
   bandwidth (7Gb/s, 13Gb/s),
   between [(Cluster_Filtrage, Cluster_Mapping)]
 }
 link (Bandwidth2) {
   bandwidth (13Gb/s, 26Gb/s),
   between [(Cluster_Rendu, Cluster_Visual)]
 }
}
virtual timeline VISUPIPE_Timeline {
 t0 = start (Cluster_DB, Cluster_Filtrage,
     Low_Latency_Between_Cluster)
     until transfer (5GB), computation (30min)
 t1 = after (t0) start (Cluster_Mapping, Bandwidth1)
     until computation (1hour)
 t2 = after (t0) start (Cluster_Rendu)
     until computation (45min)
 t3 = after (t2) start (Cluster_Visual, Bandwidth2)
}
```

Fig. 7. VISUPIPE description using VXDL

```
a)
virtual grid Query_Without_Network_Topology {
 nodes (Cluster_NAS) {
   function computing
   size (16, 20)
   node (Nodes_Cluster_NAS) {
     parameters memory_ram (min 1GB)
   }
 }
}
```

```
b)
virtual grid Query_With_Latency_Specification {
 nodes (Cluster_NAS) {
   function computing
   size (16, 20)
   node (Nodes_Cluster_NAS) {
     parameters memory_ram (min 1GB)
   }
 }
}
virtual topology Query_With_Latency_Specification {
 link (Intra_Cluster) {
   latency (max 0.100ms),
   between [(Nodes_Cluster_NAS,
              Nodes_Cluster_NAS)]
 }
}
```

```
c)
virtual grid Query_With_Latency_Specification {
 nodes (Cluster_NAS) {
   function computing
   size (16, 20)
   node (Nodes_Cluster_NAS) {
     parameters memory_ram (min 256MB)
   }
 }
}
virtual topology Query_With_Latency_Specification {
 link (Intra_Cluster) {
   latency (max 0.100ms),
   between [(Nodes_Cluster_NAS,
                 Nodes_Cluster_NAS)]
 }
}
```

```
d)
virtual grid Query_With_Latency_Specification {
 nodes (Cluster_NAS) {
   function computing
   size (16, 20)
   node (Nodes_Cluster_NAS) {
     parameters memory_ram (min 256MB),
                cpu_frequency(min 1GHz)
   }
 }
}
virtual topology Query_With_Latency_Specification {
 link (Intra_Cluster) {
   latency (max 10ms),
   between [(Nodes_Cluster_NAS,
           Nodes_Cluster_NAS)]
 }
}
```

Fig. 8. VXDL specification for the latency analysis: a) No network topology. b) 0.100ms, resulting in 1 VM per node. c) 0.100ms, resulting in 2 VMs per node. d) 10ms, resulting in 2 VMs per node.

```
virtual grid Test start 15/05/2008 14:00:00
                                  for 48:00:00 {
 group (Cluster1) {
   function computing
   size (min 8)
   anchor bordemer.bordeaux.grid5000.fr
   resource (Nodes_Cluster1) {
     parameters memory_ram (min 512MB)
     software debian, mpi
   }
 }
 resource (Router1) {
   function router (ports 16)
   anchor bordemer.bordeaux.grid5000.fr
 }
 resource (Router2) {
   function router (ports 16)
   anchor bordemer.bordeaux.grid5000.fr
 }
 group (Cluster2) {
   function computing
   size (min 8)
   anchor bordemer.bordeaux.grid5000.fr
   resource (Nodes_Cluster2) {
     parameters memory_ram (min 512MB)
     software debian, mpi
   }
 }
}
virtual network topology Test {
 link (Node_to_Node) {
   latency (max 0.200ms), bandwidth (min 1Gbps),
   between [(Nodes_Cluster1, Nodes_Cluster1),
       (Nodes_Cluster2, Nodes_Cluster2)]
 }
 link (Node_to_Router) {
   bandwidth (min 1Gbps),
   between [(Nodes_Cluster1, Router1),
            (Nodes_Cluster2, Router2)]
 }
 link (Router_to_Router) {
   bandwidth (min 100Mbps),
   between [(Router1 port 1, Router2 port 1)]
 }
}
virtual timeline Test {
 time0 = start (Cluster1, Cluster2, Router1,
           Router2, Node_to_Node,
           Node_to_Router, Router_to_Router)
}
```

Fig. 9. VXDL query with network topology.

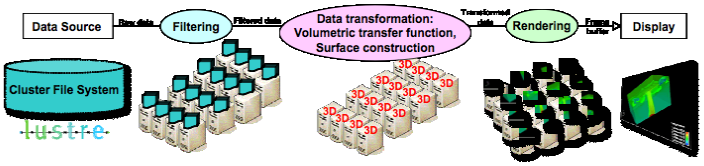


Fig. 10. VISUPIPE execution pipeline.

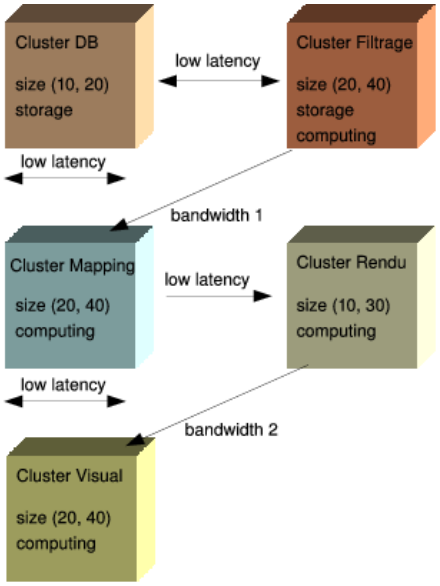
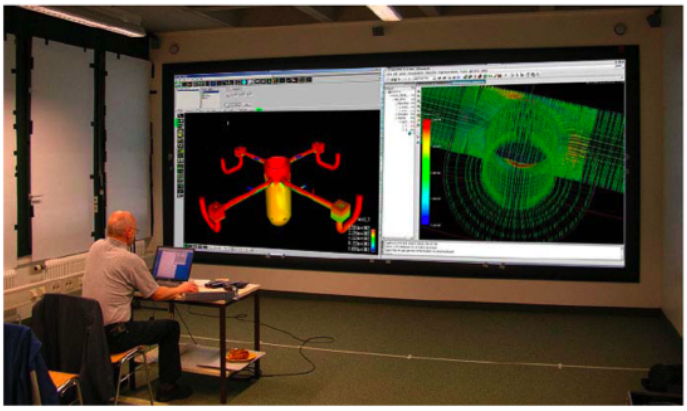


Fig. 11. VISUPIPE execution infrastructure.



Exemple de mur d'images ayant une résolution de plusieurs millions de pixels.

Fig. 12. Example of VISUPIPE visualization resources.