Science gateways made easy: the In-VIGO approach

A. Matsunaga, M. Tsugawa, S. Adabala, R. Figueiredo, H. Lam and J. Fortes
ACIS Laboratory, University of Florida, Gainesville, FL
email:fortes@ufl.edu

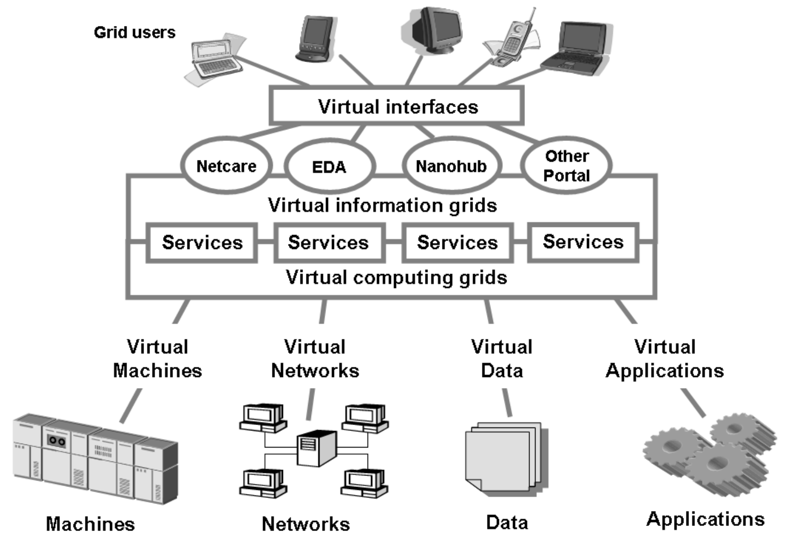
1. Requirements and the In-VIGO approach to grid-computing

According to the definition of a science gateway as a "community-specific set of tools, applications, and data collections that are integrated together via a portal or a suite of applications, providing access to Grid-integrated resources", there are several requirements to be met by its supporting Grid infrastructure [1]. A non-exhaustive discussion of these requirements follows.

Tool requirements: Science gateways must support tools that are diverse beyond differences in the programming languages used for their implementation. Some are sequential while others are parallel codes. Some are open-source research-grade programs for an initially restricted set of users while others are commercial codes - for which only binaries might be available - with a large user base. Some tools can be treated as trusted codes but, in general, they must be assumed to be untrusted. Tool interfaces and usage modes also vary greatly. They can be command-line oriented with text-like inputs and outputs. Some tools subsume other software - such as Matlab - to generate graphical outputs from numeric data. Most commercial tools have relatively sophisticated graphical user interfaces (GUIs). GUIs are often integrated with the computational parts of the tools and do themselves require appropriate software libraries. In some tools the data-input, execution and data-output phases are sequential whereas in others they are concurrent, possibly requiring composition with other tools and workflow control.

User and developer requirements: Many users are interested in using tools whereas some are only interested in demonstration runs. Tool users may be interested in using the codes for educational and/or research purposes. Both batch and interactive modes must be supported. Tool usage may require extensive resources and execution times or may only use a few computational cycles on a single machine. An important class of users consists of those who develop, modify or compose existing tools in order to maintain them, experiment with new models or adapt existing ones to specific needs. Whereas tool users may only require access to tool interfaces, developers need an execution environment that allows access to code and the use of tools for development, testing and debugging. They also need means for easily turning tools into Grid-services.

Shared infrastructure requirements: It should be possible to dynamically add and remove resources across administrative domains, while preserving security and privacy of data, codes and other users' information. Resource owners should never have services denied or compromised by Grid users. These requirements embody the goals of Grid computing and have been addressed by much middleware research and standards development. In the context of these standards, several design decisions must be made to address the performance and usability requirements mentioned above. Users should not experience any functionality limitation, additional overheads and interface complexity when using science gateways instead of their own resources. Ideally, gateways should provide greatly increased capabilities, usage conveniences and efficiencies. Similarly, the creation and management of science gateways, as well as the addition of new tools to them, should require minimal effort by grid-administrators, tool developers or users.



The In-VIGO [2] approach to Grid-computing is unique in that it decouples user environments from physical resources by using existing and novel technologies to virtualize all resources, including machines, networks, applications and data (see figure). In-VIGO creates and aggregates virtual instances of these resources as needed to build virtual computational Grids to serve specific users and applications. Within virtual computational Grids, tools and other utilities can be provided as Web services which can be further aggregated and combined to constitute (virtual) information Grids. This level decouples application interfaces from application implementations thus hiding the kinds of codes and machines used to provide services. At the topmost levels of In-VIGO, users are presented with domain-specific gateways, of which the Nanohub is an example. The first version of In-VIGO has been online since July of 2003; this and newer versions of In-VIGO have been the subject of research and development since August of 2001. The concepts discussed in this paper have been implemented in at least one of these versions.

The purpose of this position paper is not to discuss In-VIGO in detail, but to highlight the following aspects of the In-VIGO approach: how to efficiently turn tools into Grid-services accessible via a user-friendly Web-interface, how to transparently provision the execution environments needed by tools and users, and how to enable single sign-on authentication for different kinds of users without compromising security.

2. Automatically Enabling Unmodified Applications

Automating the process of Grid-enabling applications, i.e., turning unmodified applications into Grid-services that can be integrated into Grid-portals or Grid-workflows, is essential for rapidly deploying a large pool of applications on a science gateway. The process entails the following steps: (1) provision of information about the application; (2) transformation of this information into a Grid-service that can invoke the application; (3) generation of a graphical user interface that provides access to the Grid-service in a user-friendly manner.

The application information is provided by an application *enabler*, a special user who has knowledge of the application, but not necessarily of the underlying Grid infrastructure. The application information consists of system requirements (e.g., necessary machine architecture, operating system, libraries and connectivity) and application command-line interface description. To Grid-enable an application, a grammar for the command-line interface of the application is provided by the application enabler. This grammar allows the enabler to describe each parameter of the command-line in detail, including its description in natural language, while allowing groups and dependencies to be specified. This description is then transformed to the formats required for Grid-service creation and deployment.

In-VIGO provides two solutions for automating the Grid-service creation and deployment: Generic Application Service (GAP) and Virtual Application Service (VAS). The main difference between the two approaches is how the application service interface is exposed. In the Generic Application Service (GAP) solution, a generic Grid-service dynamically configures itself according to the application information, making the interface of the specific application available to the service client using a language developed in the In-VIGO project. The Virtual Application Service (VAS) approach generates one specific Grid-service for each application so that the application interface is fully described using the standard Web Services Description Language (WSDL). The VAS approach is fully based on standards, thus leveraging existing Web-service tools for automated Web-interface creation, workflow construction and Web-service management. The GAP approach has the advantages of (a) being lightweight due to the static number of libraries that it needs and (b) offering the opportunity to easily supply features that are not present in standard specifications (e.g. dynamic modification of the service interface according to user rights). The two approaches also share similarities. The services are responsible for validating job requests taking into consideration each parameter of the application, selecting appropriate Grid resources, submitting the job requests for execution in the selected resources, and monitoring job execution. Typically, any application that runs in batch mode can be handled by both solutions.

For non-batch applications with a GUI, it is not necessary to generate interfaces. Users are given direct access to the tool's GUI through remote display technologies (such as VNC). The above-described services assume that Grid middleware can dynamically provide isolated execution environments as needed for tool execution by different users as described next.

3. Secure services for provision of execution environments

In-VIGO supports dynamic allocation of execution environments per user and per distinct application by using virtual machine technologies (including language-based Java VMs, as well as O/S-based VMs, such as VMware, User-mode Linux) and/or "shadow" accounts. For efficiency and scalability purposes, mechanisms are provided for multiplexing virtual machines and accounts among users and applications without compromising security and customizability. Virtual machines can either be created and destroyed for every In-VIGO session or be made persistent across sessions. Virtual machines used to run applications can also be shared across several applications by using shadow accounts, which are pre-created accounts on machines that In-VIGO can use on behalf of arbitrary users.

User data and codes used in shadow accounts are not persistent across distinct uses of that shadow account by In-VIGO. These accounts have transient access to a user's data – during the intervals defined by the beginning and end of In-VIGO sessions. The Grid Virtual File System of In-VIGO (GVFS) creates on-demand distributed file system sessions that enforce isolation among users and implement the mapping of dynamic virtual machines and associated accounts to the user's identity in an In-VIGO file server, while supporting unmodified binary applications. This is achieved by intercepting, modifying and forwarding distributed file system protocol calls (currently versions 2 and 3 of the Network File System, NFS, are supported) via user-level proxies that are spawned, configured and terminated by In-VIGO's resource manager.

Connectivity between machines (physical or virtual) is provided by In-VIGO virtual networks (ViNe). ViNe creates multiple isolated networking environments with bi-directional communication between any pair of participating hosts, in spite of the presence of limiting devices such as firewalls and NAT gateways. ViNe is based on IP-overlay on top of the Internet, and the architecture resembles a VPN site-to-site configuration. However, ViNe addresses issues that VPN cannot solve such as on-demand creation of isolated virtual networks, inclusion of hosts in private networks with overlapping IP address spaces, and support for large number of participating physical networks with low administration overhead.

4. Secure access to resources, applications and data

In-VIGO users do not have direct access to and are completely decoupled from user accounts on grid resources where jobs are effectively run. In-VIGO middleware has full control of all resources and is responsible for starting jobs as well as maintaining them, with complete freedom on how to dynamically map grid users to local users, and possibly recycle local shadow accounts among grid users. The approach brings advantages for both grid users and resource providers: grid users are freed from the need to manage several credentials; resource administrators are freed from the task of reconfiguring the access control of resources every time a user joins or leaves the grid.

In-VIGO users authenticate themselves by presenting their username and password to the grid portal. After login, user actions resulting in access to a grid resource are handled by the In-VIGO middleware through the use of Role-Based Access Control (RBAC) mechanisms, offering Single Sign-On (SSO) for users. Users are grouped into roles (e.g., regular, Matlab licensed, administrator), while resources are configured by their providers with a set of permission groups which define operations (e.g., a simulator in demo, full and configuration modes). Appropriate mappings between user roles and permission groups are defined, and In-VIGO middleware enforces the mappings when accessing resources on behalf of users. For example, only users in the "Matlab licensed" role would be able to run a Matlab-based simulator in its full operation mode.

Resources, especially shadow accounts, need to be isolated from each other because they are recycled among grid users. To address this need, local accounts are either pre-created by resource providers, or created on-demand for a particular user in VMs where In-VIGO middleware has administrative privileges. In the first case, In-VIGO middleware makes sure that, at any point in time, only one user is mapped to a given shadow account, and also that the account is cleaned when the job finishes. In the latter case, accounts are created and destroyed for one grid user, without the need for recycling. Since a local account does not run processes for two different users simultaneously, user isolation at process level is guaranteed. However, shadow accounts also need to have their data access privileges limited to the current assigned grid user, as isolation is compromised if shadow accounts have access to data of all grid users. GVFS provides the necessary data isolation between grid users. GVFS controls access at the granularity of directories so In-VIGO middleware is able to limit the shadow account's access of data to the home directory of the grid user allocated to it. Further data isolation, among jobs running for the same grid user, can be achieved by limiting the access of the shadow accounts running the jobs to the job working directory, which are subdirectories under the grid user's home directory.

As the In-VIGO middleware has all the necessary credentials to access accounts (i.e., to remotely submit a job, independently of the mechanism – Condor, GSI, PBS, ssh, etc) to run jobs on behalf of the user, providing SSO access to grid resources is trivial. More complex SSO solutions are however required when providing users access to interactive applications that require application level authentication from the user. Examples of such applications currently supported by In-VIGO include VNC sessions and a web-based file-manager. In the case of VNC, In-VIGO remotely starts its server process with a random password in a shadow account. When the user requests access to the VNC desktop, In-VIGO embeds the necessary credential into the VNC client applet and transmits it securely (through SSL) to the user. When the VNC client is run by the user, it authenticates automatically (on behalf of the user) to the server. Adding RBAC to the above process, enables In-VIGO to allow sharing of workspaces among users, i.e., it enables a group of users (belonging to a single user role) to access a given VNC session without the need for users to share credentials and/or passwords.

Acknowledgements

The In-VIGO project is supported in part by the National Science Foundation under Grants No. EIA-9975275, EIA-0224442, ACI-0219925, EEC-0228390; NSF Middleware Initiative (NMI) collaborative grants ANI-0301108/ANI-0222828, SCI-0438246; and by the Army Research Office Defense University Research Initiative in Nanotechnology. The authors also acknowledge two SUR grants from IBM and a gift from VMware Corporation. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Army Research Office, IBM, or VMware.

References

[1] Ian Foster, Carl Kesselman, and Steven Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *Intl Journal Supercomputer App.*, vol. 15, no. 3, 2001.

[2] Sumalatha Adabala, Vineet Chadha, Puneet Chawla, Renato Figueiredo, José A. B. Fortes, Ivan Krsul, Andrea Matsunaga, Mauricio Tsugawa, Jian Zhang, Ming Zhao, Liping Zhu, Xiaomin Zhu, "From Virtualized Resources to Virtual Computing Grids: The In-VIGO System", In *Future Generation Computing Systems*, 04/2004.