Grids: Harnessing Geographically-Separated Resources in a Multi-Organisational Context

Anand Natrajan, Marty A. Humphrey, Andrew S. Grimshaw
Department of Computer Science, University of Virginia, Charlottesville, VA 22904, USA
{anand, humphrey, grimshaw)@cs.virginia.edu

1 Introduction

Grids are becoming ubiquitous platforms for high-performance computing and distributed collaboration. A grid benefits users by permitting them to access heterogeneous resources, such as machines, data, people and devices, that are distributed geographically and organisationally. It benefits organisations by permitting them to offer unused resources on existing hardware and thus reclaim otherwise lost costs. Although worldwide grids can be constructed today, issues regarding heterogeneity, security and failures must be resolved especially if the participating resources are controlled by different organisations. A grid infrastructure that harnesses the power of distributed resources for computing and collaboration must respect the autonomy of organisations to choose policies for using their resources.

Legion is a grid infrastructure that presents users a view of a grid as a single virtual machine [GRIM97]. This view reduces the complexities a user encounters before running applications or collaborating on a grid. The functions performed by Legion on a grid are similar to the functions performed by a traditional operating system on underlying hardware. The design principles of object-basedness and integration have enabled Legion to be extended and configured in a number of ways, while ensuring that the cognitive burden on the grid community is small.

1.1 Grid History

Grids are the next step in a logical progression beginning with the Internet and the World Wide Web. The internet enabled connecting previously-isolated islands of computing resources to one another. With internet tools, a user could connect to a machine remotely, without being physically present at the machine. After connecting to a remote machine, the user could utilise a small set of services, such as transferring data or issuing limited commands.

The World Wide Web improved over the internet in two ways. First, it made the internet more accessible by making the tools more usable. Second, it enabled a richer form of sharing among users. Previous internet tools transferred raw, uninterpreted data. However, a web browser interprets data, thus giving users a better interface and enabling more abstract collaboration, such as sharing a picture rather than transferring kilobytes. The web showed that for computing infrastructure to be considered useful, it must enable collaboration.

A grid extends the notions of collaboration while preserving the traditional role of computers as resources used for computing. In essence, computing is collaboration, where a resource provider and a consumer collaborate using a job or task as a unit of collaboration. A large number of applications are starved for computation resources (searches for extra-terrestrial intelligence, studies of protein folding, genomics, stock market models, etc.), whereas an overwhelming majority of computers are often idle. This disconnect can be bridged by permitting computation-intensive applications to be run on otherwise idle resources, no matter where the resources are located. Running Java applets on the web is a form of computing-as-collaboration; however, it is still not a grid because the model for running applets merely extends the basic web model. The sophisticated collaboration enabled by a grid is desirable; scientific users expect to share more than images, financial users expect to share more than periodically-updated tables, and all users expect to control who accesses whatever they choose to share. Since a grid is a first-class step in the evolution of computational infrastructures, a design from first principles is indicated strongly to satisfy and anticipate current and future demands.

1.2 Legion History

The Legion project evolved from the experience gained from an earlier project, Mentat, and the guidance of multiple professors of Computer Science at the University of Virginia. The domain expertise of each contributor — distributed systems, networks, architecture, security, programming languages and information retrieval — led to an integrated infrastructure for managing grids [GRIM94]. This design process is a reflection, on a much smaller scale, of the design process that resulted in the sophisticated operating systems available today. Mentat, the precursor of Legion, was a data-parallel language that added a small number of keywords to the vocabulary of C++. Mentat programs were parsed by a compiler which determined data dependencies, placed data accordingly, and extracted as much parallelism as possible from the program [GRIM96]. Exploiting fine-grained parallelism is expensive; therefore, the designers of Mentat focussed on exploiting coarser-grained parallelism for grids.

Legion was created from a fresh code base in 1993-4, thus closely preceding or concurring with the release of the World Wide Web. In many respects, the World Wide Web is based on principles similar to those in Legion. However, the web's strength is sharing, not high-performance computation, and even its tremendously-popular sharing model, is not as rich as that of Legion. Legion has been funded since inception with grants from the Department of Defence and the National Science Foundation under the National Partnership Alliance for Computational Infrastructure (NPACI). Since inception, Legion has been used to manage a US-wide (and occasionally worldwide) grid almost continuously. This grid, called *npacinet* (originally *vanet*), has been used to demonstrate grid technologies, run scientific applications and conduct numerous demonstrations, especially live ones at SuperComputing and other fora.

In 2000-1, a private company called Avaki Corp. (formerly Applied Metacomputing, Inc.) purchased rights to Legion. The goal of the corporation is to commercialise the technology by addressing the immediate and future needs of organisations intending to be donors and consumers of grid resources. Independently, the research group at the University of Virginia continues to manage *npacinet* for academic and research purposes. Although Legion has made significant strides in addressing many of the complex problems inherent in harnessing distributed resources, research issues still exist, notably in security, fault-tolerance and interoperability with other grid infrastructures.

2 Legion: Philosophy and Architecture

Legion is a grid operating system [GRIM98]. Several of the features provided by Legion for managing a grid, such as a single namespace, a file system, security, process creation and management, interprocess communication, input-output, resource management and accounting, are exactly what traditional operating systems have provided for a single machine. In addition, Legion provides numerous other features, such as complexity management, wide-area access, heterogeneity management, multi-language support and legacy application support, which are required in the context of a grid system. The single virtual machine view of the grid provided by Legion enables users to access and use a grid without necessarily facing the complexity of the components of the grid.

2.1 Object-basedness

In Legion, most important components of a grid are first-class objects [LEWIS96]. Object-based design offers three advantages. First, it leads to a modular design wherein the complexity of a component is managed within a single object. Second, it enables extending functionality by designing specialised versions of basic objects. Third, it enables selecting an intuitive boundary around an object for enforcing security. Although object-basedness is an essential feature in the design of Legion, grid users do not have to conform to object-based or object-oriented design. Legion supports legacy applications without requiring any change to source or object code. Applications do not have to be "Legion-aware", i.e., they need not access Legion objects. For Legion-aware applications, Legion provides a C++, C, Java and Fortran interface.

2.2 Naming & Transparency

Naming services and the transparency usually gained from a good naming service are important concepts in the design of large systems. A traditional OS offers multiple naming domains — for file system components (filenames), for processes (process IDs), for users (user IDs), etc. Likewise, a network system also offers multiple naming domains — for machines (DNS names), for individual connections (port numbers), etc. Clearly, naming is important; useful services and components cannot be identified without naming them. Transparency is also important; a name must mask irrelevant details about the named service or component. Naming and transparency are fundamental concepts in Computer Science — so fundamental that they can be taken for granted easily.

A number of transparencies can be associated with a name. For example, a Unix filename is access-transparent because it masks the storage medium of the file which may be on a disk or tape. The name is migration-transparent as well because it masks changes in the inode set for a file. A DNS name is location-transparent because it masks the physical location of a machine. The URL name for a web site may mask multiple machines serving the same request, an example of replication (or perhaps even concurrency) transparency.

Every object in Legion, be it a machine, a user, a file, a subdirectory, an application, a running job or a scheduler, has a name. Legion unifies the multiple namespaces of traditional systems by providing a single namespace for behaviourally-diverse and geographically-distributed components. Every Legion object has an ID associated with it — its LOID. The LOID of an object is a sequence of bits that identifies the object uniquely in a given grid (and also across different grids) without forcing subsequent accesses to violate transparency. Once the name of an object is known, it can be queried in different ways, such as about its physical location, its current status, the permissions on it, associated metadata and the kind of service it provides (its interface). Once an object's interface is known, it can be requested to perform a desired service, typically by means of an asynchronous remote procedure call. As in dataflow, a procedure call can receive parameters from multiple objects and forward its return values to yet other objects.

2.3 Service — Policy *vs*. Mechanism

An important philosophical tenet in Legion is that mechanisms can be mandated but not policies. Users and administrators of a grid must be free to configure a grid and its components in any suitable manner by constructing policies over mechanisms. For example, Legion provides mechanisms for constructing schedulers which can be used to assign machines for jobs. However, Legion neither mandates any scheduling policy, nor requires a single scheduler for a grid. Users and administrators may construct as many different schedulers and instances of any scheduler as they wish. In general, Legion permits users to make trade-offs between multiple types, levels and costs for any service.

2.4 Security

Security in Legion is based on a public key infrastructure (PKI) for authentication and access control lists (ACLs) for authorisation. Legion requires no central certificate authority to determine the public key of a named object because the object's LOID contains its public key. The ACL associated with any object encodes the permissions for that object. When any method of a Legion object is invoked, the protocol stack associated with the object ensures that the security layer is invoked to check permissions before the request is forwarded to the method itself. The security layer is also responsible for decrypting messages encrypted with the public key of the object. Although in the current implementation the security layer is based on PKI and ACLs, it can be retargetted to other authentication mechanisms, such as Kerberos, and other authorisation mechanisms [FERR99].

2.5 Extensibility

A grid infrastructure must be flexible enough to satisfy current as well as anticipated demands of grid users. Legion was designed to extensible for that very reason. Specialised objects can be constructed from basic objects for special functionality. New objects can be constructed and deployed in an existing grid, thus extending the functionality of the grid. Three examples of extensibility are: queue hosts, process control daemon (PCD) hosts and two-dimensional (2D) files.

Ordinary Unix hosts are represented by a Unix host object in Legion. On such hosts, the host object starts jobs by `fork-exec`. However, a queue host object starts jobs by submitting jobs to a queue. The queue host object is a straightforward extension of the Unix host object with the specialisation for job submission, monitoring and termination. Differences between different queuing systems are encoded in a few scripts.

On ordinary hosts or queues, jobs are started under the ID of the local Legion user. However, in order to satisfy stricter demands of security and accounting, a special host, called PCD host was derived from existing host objects. The PCD object ensures that a Legion user can start a job on a host only if she has an account on that host. Alternatively, a relaxed form of the PCD host can enable users to run jobs by mapping them to generic accounts. A daemon associated with such hosts is the only instance of Legion requiring root privileges at a site.

2D files are useful if the data accessed happens to be a two-dimensional matrix. A 2D file can be accessed by rows or columns with arbitrary stripes. 2D files are especially convenient if the entire matrix is too large to fit on one disk; sub-matrices can be stored at different locations and accessed transparently as if they were collocated.

2.6 Interfaces

Grid interfaces determine how users perceive and use the resources on a grid. Legion supports a variety of interfaces such as command-line tools, programmatic interfaces and access through familiar and traditional tools. Each of these interfaces has its own strengths and limitations, but the diversity available enables users to choose the interfaces that they can use best. As grids become more common, we expect interfaces to become more numerous, more sophisticated and less obtrusive. In particular, interfaces for managing and monitoring large numbers of jobs must become more sophisticated so that users can view the progress of bulk runs broadly as well as specifically.

2.7 Integration

A grid is a complex construction because of the diversity of the machines comprising the grid, as well the wide variety of security policies, failures and usage policies associated with them. Much of this complexity must be masked from a user for reasons of relevance and convenience. Legion is designed in order to mask complexity from the user. One of the ways in which Legion masks complexity is by providing an integrated system. A piece-meal approach to providing a grid infrastructure can increase the cognitive burden of grid administrators and users significantly. In contrast, with Legion's integrated approach, administrators can set up a minimal grid within a few minutes by issuing four commands. Adding hosts to a grid involves repeatedly invoking three or four commands. Other examples of integration are Legion's distributed file system and high performance tools.

Legion provides a global, distributed file system for every grid it manages. This file system, a.k.a., *context space*, is similar to a Unix/Windows file system, except that its components are distributed across the machines in the grid. Directories in context space are called *contexts*. A contexts can contain any Legion object, such as other contexts, files, machines, users, console objects and applications. Since users logged on to a grid from different machines have the same view of context space, Legion enables collaboration. Moreover, since any Legion object can be a member of context space, and since arbitrary objects can be created in Legion, context space enables richer collaboration than the internet or the web. Since fine-grained permissions can be set for any object in context space, the collaboration can be as restrictive as the owner of the object desires. Context space can be accessed *via* (i) command-line interfaces (analogues to Unix tools, e.g., `legion_ls`, `legion_cat`, `legion_cp`, etc.), (ii) programmatic interfaces (analogues to Unix calls, e.g., `BasicFile_open`, `BasicFile_read`, etc.), (iii) NFS interfaces (Legion's NFS daemon `lnfsd`) and (iv) Samba, FTP and Web interfaces [WHITE00].

The primary use for a grid presently and in the near future is for high-performance computation (HPC). Legion provides a suite of HPC tools for running legacy applications, MPI applications, PVM applications and parameter-space studies. Legion also provides three schedulers: a default scheduler based on random placement, a round-robin scheduler and a performance-oriented scheduler. Multiple instances of any scheduler can be created and configured. Moreover, schedulers with novel algorithms may be created and used at any time in the lifetime of the grid.

3 Future Directions

Grids are likely to become commonplace in the near future. Initially, HPC applications will drive the demand for grids, especially in academic and industrial environments. However, as the potential for collaboration on the grid becomes more prominent, we expect grids to be deployed on a wider scale. In particular, we expect it to be common for multiple organisations to participate in a single grid, or to coalesce and separate their individual grids as required. We expect several grid infrastructures to co-exist with Legion, for example, Globus [FOST99], Condor [LITZ88], Nimrod [ABR95], NetSolve, etc.

Future trends in Legion involve continuation of the research program and commercialisation. As mentioned earlier, Avaki Corporation is commercialising Legion (known as Avaki in that sector). The project at the University of Virginia will continue to explore research issues with grids, such as interoperability with other grid infrastructures, better interfaces for grid administration and diverse security models.

4 References

ABR95 Abramson, D., *et al*., *Nimrod: A Tool for Performing Parameterised Simulations using Distributed Workstations*, Proc. of the 4th IEEE Intl. Symp. on High Performance Distributed Computing, August 1995.

FERR99 Ferrari, A. J., Knabe, F., Humphrey, M. A., Chapin, S. J., Grimshaw, A. S., *A Flexible Security System for Metacomputing Environments*, High Performance Computing and Networking Europe, 1999.

FOST99 Foster, I., Kesselman, C., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999.

GRIM94 Grimshaw, A. S., Wulf, W. A., French, J. C., Weaver, A. C., Reynolds, P. F. Jr., *Legion: The Next Logical Step Toward a Nationwide Virtual Computer*, Tech. Rep. CS-94-21, Dept. of Computer Science, Univ. of Virginia, 1994.

GRIM96 Grimshaw, A. S., Ferrari, A.. J., West, E, *Mentat*, Parallel Programming Using C++, The MIT Press.

GRIM97 Grimshaw, A. S., Wulf, W. A., *The Legion Vision of a Worldwide Virtual Computer*, Comm. of the ACM, Vol. 40, No. 1, January 1997.

GRIM98 Grimshaw, A. S., Ferrari, A. J., Lindahl, G., Holcomb, K., *Metasystems*, Comm. of the ACM, Vol. 41, No. 11, November 1998.

LEWIS96 Lewis, M. J., Grimshaw, A. S., *The Core Legion Object Model*, Proc. of the 5th Intl. Symp. on High Performance Distributed Computing, August 1996.

LITZ88 Litzkow, M, Livny, M., Muttka, M., *Condor — A Hunter of Idle Workstations*, Proc. of the 8th Intl. Conf. of Distributed Computing Systems, June 1988.

WHITE00 White, B. S., Grimshaw, A. S., Nguyen-Tuong, A., *Grid-Based File Access: The Legion I/O Model*, Proc. of the 9th Intl. Symp. on High Performance Distributed Computing, August 2000.