Legion

Worldwide Virtual Computer

Table of Contents

The Opportunity

The next several years will see the widespread introduction and use of gigabit wide area and local area networks. These networks have the potential to transform the way people compute, and more importantly, the way they interact and collaborate with one another. The increase in bandwidth will enable the construction of wide area virtual computers, or metasystems. However, just connecting computers together is insufficient. Without easy-to-use and robust software to simplify the environment, the network will be too complex for most users.

The Components

Legion will consist of workstations, vector supercomputers, and parallel supercomputers connected by local area networks, enterprise-wide networks, and the National Information Infrastructure. The total computation power of such an assembly of machines is enormous, approaching a petaflop; this massive potential is, as yet, unrealized. The machines are currently tied together in a loose confederation of shared communication resources used primarily to support electronic mail, file transfer, and remote login. However, the resources could be used to provide far more than just communication services; they have the potential to provide a single seamless computational environment in which processor cycles, communication channels, and data are all shared, and in which the workstation across the continent is no less a resource than the one down the hall.

The Goal

We envision a system in which a user sits at a Legion workstation and has the illusion of a single very powerful computer. When the user invokes an application on a data set, it is Legion's responsibility to transparently schedule application components on processors, manage data transfer and coercion, and provide communication and synchronization. System boundaries, data location, and faults will be invisible.

Enabling Technology

The technological changes that enable the realization of a worldwide virtual computer are

- the introduction of high-speed fiber-optic backbones operating in the gigabit/second range, which effectively "shrink" the distance between geographically separated computers
- the maturation of parallel compilers that support parallel execution on distributed memory machines
- advances in distributed systems software over twenty years which have increased our understanding of how to manage complex distributed environments
- the widespread acceptance in industry of the object-oriented paradigm because of its encapsulation and reuse properties
- advances in cryptography and authentication protocols that give users and service providers confidence that they are protected.

Challenges

These changes challenge the computer science community to provide a solid integrated software foundation on which to build applications, and with which to unleash the potential of so many diverse, powerful resources. The foundation must at least hide the underlying physical infrastructure from users and from the vast majority of programmers, support access, location, and fault transparency, enable inter-operability of components, support construction of larger integrated components using existing components, and provide a secure environment for both resource owners and users. Further, the services must be provided in the context of tens of thousands to millions of autonomous hosts; thus, the software infrastructure must above all else be scalable or it will fail to meet users' performance expectations. It is this software foundation that will make the collection of hosts and fiber connections a virtual computer. We call such software *metasystems software*.

Whether or not metasystems software is explicitly designed, the nation (and the world) will eventually end up with a metasystem. The reason is simple: both individual and organizational users will be required to deal with increasingly obvious shortcomings of the nationwide computer system. The World Wide Web and gopher show that tools will be developed in an ad hoc fashion to paper over the gaps between local systems. The issue is not whether metasystems will be developed; clearly they will. It is, rather, whether they will come about by design in a coherent, seamless system, or painfully and in an ad hoc manner by patching together independently developed systems, each with different objectives, design philosophies, and computation models. To facilitate the design and construction of metasystems, software infrastructures must provide foundations for applications.

Benefits

Legion will provide the illusion of a single virtual machine to users, a virtual machine that provides secure shared object and shared name spaces, application adjustable fault-tolerance, improved response time, and greater throughput. The potential benefits of Legion are enormous. We envision:

- more effective collaboration by putting co-workers in the same virtual workplace
- higher application performance due to parallel execution and exploitation of off-site resources
- improved access to data and computational resources
- improved researcher and user productivity resulting from more effective collaboration and better application performance
- increased resource utilization
- a considerably simpler programming environment for the applications programmers
- a solid foundation upon which to build next generation trans-national applications.

Objectives

From our Legion vision we have distilled the following ten design objectives that are central to the success of the project:

- **Site autonomy:** Legion will not be a monolithic system. Legion will be composed of diverse resources owned and controlled by an array of different organizations. These organizations, quite properly, insist on having control over their own resources, e.g., specifying how much resource can be used, when it can be used, and who can and cannot use the resource. Legion is like a cellular automata in that its collective behavior arises out of the individual behaviors of its components.
- **Extensible core:** We cannot know the future or all of the many and varied needs of users. Therefore, mechanism and policy must be realized via extensible, replaceable, components. This will permit Legion to evolve over time and will allow users to construct their own mechanisms and policies to meet specific needs.
- **Scalable architecture:** Because Legion will consist of thousands of hosts, it must use a scalable software architecture. This implies that there must be no centralized structures, and that the system must be totally distributed.
- **Easy-to-use, seamless computational environment:** Legion must mask the complexity of the hardware environment and of communication and synchronization of parallel processing. Machine boundaries should be invisible to users. As much as possible, compilers, acting in concert with run-time facilities, must manage the environment for the user.
- **High performance via parallelism:** Legion must support easy-to-use parallel processing with large degrees of parallelism. This includes task and data parallelism and their combinations. Because of the nature of the interconnection network, Legion must be latency tolerant.
- **Single, persistent name space:** One of the most significant obstacles to wide area parallel processing is the lack of a single name space for file and data access. The existing multitude of disjoint name spaces makes writing applications that span sites extremely difficult.
- **Security for users and resource owners:** Because we cannot replace existing host operating systems, we cannot significantly strengthen existing operating system protection and security mechanisms. However, we must ensure that existing mechanisms are not weakened by Legion.

Further, we must provide mechanism for users to manage their own security needs; Legion should not define the security policy or require a "trusted" Legion.

- **Management and exploitation of resource heterogeneity:** Clearly, Legion must support inter-operability between heterogeneous hardware and software components. In addition, Legion will be able to exploit diverse hardware and data resources. Some architectures are better than others at executing particular kinds of code, e.g., vectorizable codes. These affinities, and the costs of exploiting them, must be factored into scheduling decisions and policies.
- **Multiple language support and inter-operability:** Legion applications will be written in a variety of languages. It must be possible to integrate heterogeneous source language application components in much the same manner that heterogeneous architectures are integrated. Inter-operability also means that we must be able to support legacy codes written before the advent of Legion.
- **Fault-tolerance:** In a system as large as Legion, it is certain that at any given instant, several hosts, communication links, and disks will have failed. Thus, dealing with failure and dynamic re-configuration is a necessity. Current parallel processing systems do not address fault-tolerance. As is the case in the management of heterogeneity, there is an extensive literature in reliable distributed computing systems. We will consider two different fault tolerance problems-fault tolerance of Legion itself, and application fault-tolerance.

Constraints

In addition to the goals above, two constraints restrict our design-we cannot replace host operating systems, and we cannot legislate changes to the interconnection network.

- **We cannot replace host operating systems:** Organizations will not permit their machines to be used if their operating systems must be replaced. Operating system replacement would require them to rewrite many of their applications, retrain many of their users, and possibly make them incompatible with other systems in their organization. Our experience with Mentat indicates that it is sufficient to layer a system on top of an existing host operating system.
- **We cannot legislate changes to the interconnection network:** We must initially assume that the network resources, and the protocols in use, are a given. Much as we must accommodate operating system heterogeneity, we must live with the available network resources. However, we can layer better protocols over existing ones, and we can state that performance for a particular application on a particular network will be poor unless the protocol is changed.

In addition to the purely technical issues, there are also political, sociological, and economic ones.

Philosophy

Legion is an object-oriented system. The principles of the object-oriented paradigm are the foundation for the construction of Legion; we exploit the paradigm's encapsulation and inheritance properties, as well as benefits such as software reuse, fault containment, and reduction in complexity. The need for the paradigm is particularly acute in a system as large and complex as Legion. Other investigators have proposed constructing application libraries and applications for wide area parallel processing using only message passing services. Using such tools requires the programmer to address the full complexity of the environment; the difficult problems of load balancing, scheduling, fault management, etc., are likely to overwhelm all but the best programmers.

Complementing our use of the object-oriented paradigm is one of our driving philosophical themes: we cannot design a system that will satisfy every user's needs, so we must design an extensible system. This philosophy manifests itself throughout Legion's design. For example, consider security. There is a trade-off between security and performance (due to the cost of authentication, encryption, etc.). Rather than providing a fixed level of security, with the result that no one is completely satisfied, users may choose their own trade-offs by implementing their own policies or by using existing policies via inherit ance. Similarly, users can select the level of fault-tolerance that they want, and pay for only what they use. By allowing the user to implement her own services or to inherit them from library classes, we provide her with flexibility complemented by a menu of existing choices.

legion@Virginia.edu
http://www.cs.virginia.edu/~legion/