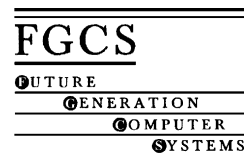




ELSEVIER

Future Generation Computer Systems 17 (2001) 783–793



www.elsevier.nl/locate/future

A parallel/distributed architecture for hierarchically heterogeneous web-based cooperative applications

Thomas L. Casavant, Terry A. Braun*, Sureshkumar Kaliannan,
Todd E. Scheetz, Kyle J. Munn, Clayton L. Birkett

*Parallel Processing Laboratory, Department of Electrical and Computer Engineering, and the Genetics Program,
University of Iowa, Iowa City, IA 52242, USA*

Abstract

A new class of applications is described which requires cooperation among diverse users in multiple data and problem instance domains. The hierarchy of parallelism includes heterogeneity within a single instance of the problem, homogeneity among subsets of users within a problem domain, and multiple problem domains which share computational resources. The particular problem of genetic linkage analysis is used to illustrate and implement the architecture. GenoMap, the first implementation of this system is being deployed for several groups of cooperating users at multiple institutions in a study to isolate the genomic locus of the controlling gene(s) in several diseases including autism. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Distributed; GenoMap; Genetic; Linkage; Socket Server

1. Introduction

This paper describes a general software architecture for a class of applications made possible by the WWW. The motivation for our study is a medical research application in the Human Genome Project (HGP)—genetic linkage analysis. Genetic linkage analysis is the primary method used to determine the association between a genetically linked trait (of interest to us are disease traits such as cancer, autism, etc.) and a region (locus) of the human genome (a search space of some three billion base pairs). This particular problem requires cooperation among a diverse collection of researchers including geneticists, clinical

physicians, statisticians, disease specialists, laboratory technicians, and computer scientists/engineers. The scale of this problem would clearly indicate the need for large-scale informatics support for a diverse group of heterogeneous users. Historically, however, relatively rare, simple traits have been studied and ad hoc methods for data storage and analysis have been successfully employed to isolate only a few hundred disease genes (from a set of approximately 100 000 candidates) to date. The experience gained in these studies, coupled with the application of distributed/parallel computing methods, is making possible the systematic approach of the isolation of all the disease genes in the human genome. Below, we characterize the primary parallel/distributed characteristics of this class of applications. Section 2 gives a brief introduction to the concepts and requirements for a linkage analysis. Sections 3 and 4 describe GenoMap and its various components. The current implemen-

* Corresponding author. Tel.: +1-319-335-6432;
fax: +1-319-335-6028.
E-mail addresses: tabraun@eng.uiowa.edu, genomap@eng.uiowa.edu (T.A. Braun).

tation of GenoMap is briefly discussed in Section 4, for more information the reader is directed to [1]. To date, the first internal release of GenoMap has been in use for approximately 2 years and is presently being employed in the analysis of more than 400 genetic loci with data describing disease traits from more than 300 individuals with respect to the disease autism.

The class of applications of interest have parallelism present at three distinct levels. We describe them in a hierarchy, and also distinguish between multiplicity of users (sets of clinicians) and multiplicity of problems (diseases).

Level 1. Heterogeneous parallelism within a single problem instance. These components represent functional parallelism.

Level 2. Multiple instances of each functional component referred to in Level 1.

Level 3. Multiple instances of the entire problem class, potentially with multiple instances of each functional component in Level 1.

Level 1 illustrates a key attribute of an application of this class. In the genetics linkage setting, this refers to the need to support the computational demands of each of the diverse members of the team of users—physicians, statisticians, laboratory technicians, etc. Each user needs to see the system through the “window” of their part of the collaboration. In Level 2, we are describing the oft-present attribute of scale. As a problem instance becomes large, not only do computational demands grow, but so do the number of project personnel and their geographical dispersion. In the case of our autism study, the scale of the project dictated that distinct groups (one at the University of Iowa, one at Tufts University, and one at Vanderbilt University) cooperate in the labor-intensive gathering of phenotypes (patient status w.r.t. autism), genotypes (the states of the 400 different genome loci), and conducting the many analyses required to determine which (if any) of the loci were primary candidates for autism. The increasing computational demands yet dispersion of computational resources is a natural scenario for a load balancing approach. At the third level of the hierarchy, it is desirable to manage the computational resources, in addition to the data resources, in a cooperative way. This involves the sharing of software and hardware. Users in many problem domains (such as medicine and biology) do not have easy

access to the kind of support for cooperative computing as dictated in this class of applications. Thus, the solution to such a problem requires the ability to support sharing of data, software, and computational resources, while assuring secure isolation of data from multiple distinct instances of the problem. Clearly, in handling information regarding genetic traits, and diseases, the need for such security is clear.

We summarize the requirements of our solution as follows:

- The solution must support seamless sharing and cooperation of users at Levels 1 and 2.
- The solution must assure that parallelism at Level 3 is supported with secure isolation of distributed components.

GenoMap, a distributed, web-based system for the support of genetic linkage analysis, illustrates the class of applications, and the key elements of our approach to solving the problem.

2. Background: genetic linkage analysis

This section serves as a brief introduction to some of the integral concepts of the genetics and molecular biology underlying the application of linkage analysis with an intuitive explanation of linkage analysis.

A genome is the entire genetic material of an organism, meaning all of the DNA that makes up the organism (which is replicated in every cell). The human genome is divided into 22 separate somatic (non-sex) chromosomes as well as the X and Y chromosomes. Within the genome are genes—regions of chromosomes that serve as templates for the production of proteins. Thus genes code for proteins. A dysfunctional gene and/or gene product can have a harmful effect on the organism (e.g., Duchenne Muscular Dystrophy). However, the entire genome of higher organisms is not dedicated only to encoding for proteins. In fact, only a small portion (genes) is used as a template for protein production. Much of the intervening areas are still of use for analysis through the use of genetic markers. A genetic marker is a unique site within the genome that can be used to determine the “state” of a region of the genome, known as the “genotype.” The markers, or genotypes, are the mechanisms/data by which the actual inheritance patterns of chromosomes

Table 1
GenoMap components

Name of software component	Description
Subject Log	Recording and searching for subject specific information
Marker Log	Recording and searching for marker specific information
Trait Log	Recording and searching for trait specific information
Linkage Experiment Editor	Creating, viewing, and updating linkage experiments
Genotyping Assistant	Creating, viewing, and updating genotyping experiments
GenoScape	Viewing and genotyping of genotyping gel images
GenoScape Launcher (client/server)	Front-end interface to genotyping tool
Verification (client/server)	Interface to display all recorded genotypes together
Linkage Analysis (client/server)	Interface to multiple linkage analysis packages
Socket Server	Manages communication between clients and servers, plus handles load balancing

between generations is observed. Now, given a set of polymorphic (informative) genetic markers, we can collect genotypes for an individual across all of his/her chromosomes. Another piece of information required for analysis is the phenotype of an individual—the status of that subject with respect to the disease (or trait) being studied. Intuitively, an affected individual for a disease is likely to contain in their genome the defective gene responsible for that disease.

Formally, linkage analysis is a statistical analysis, attempting to correlate a suspected genetic disease to a region within the genome. When evidence exists for the localization of a disease gene to a region of the genome, this is referred to as “linkage,” i.e., the disease gene is “linked” to a region (genetic marker) on a chromosome. Linkage analysis requires a set of genetically related subjects (a family), the phenotypes of the subjects with respect to a disease, and the genotypes of the subjects for a genetic marker. The likelihood that the data (family structure, phenotypes, and genotypes) correlates with a model of inheritance for the disease and with the actual inheritance patterns (observed by genotypes) provides a measure of the evidence for “linkage.” Thus, a linkage analysis calculates the likelihood that the disease-causing gene is located near a genetic marker based on the given set of subjects, phenotypes, and genotypes.

In the context of GenoMap, we provide the following definitions. A *linkage experiment* is a grouping of subjects, genetic markers and a trait to be analyzed through linkage analysis. A *linkage study* is a set of linkage experiments, all pertaining to the same trait, but with possibly different subjects and/or markers. For example, the first linkage experiment of a linkage

study might be to screen for the position of a trait’s locus across the entire genome, such that three candidate loci are identified. Then a more focused linkage experiment could be designed, to more efficiently probe the suspected loci. Lastly, a *genotyping experiment* is a set of genotyping gels associated with a given linkage experiment, where a *gel* is a tool used by molecular geneticists to obtain genotypes.

3. The GenoMap system from the user view

GenoMap is a suite of independent, yet inter-related tools primarily developed in Java. These tools manipulate data from a domain-specific networked database, allowing sharing of information among multiple distributed clients without replication and coherency problems. The main goal of GenoMap is to provide a portable, intuitive interface for managing the information associated with the gene location/discovery process. Table 1 briefly describes the various services provided by GenoMap. A more complete description of each of these components appears elsewhere [1,9].

4. GenoMap architecture

GenoMap is a large-scale, distributed, heterogeneous, client–server application to support the systematic exploration of the genome to narrow, and ultimately identify, the locus of a particular gene (or set of genes) involved in a disease or trait. In contrast to many applications developed in support of the Human Genome Project (HGP) [4] to date, GenoMap

does not involve gathering or analysis of any DNA sequence data. Rather, the fundamental informational components of this *functional genomics* [7] applications are (1) familial relationships (pedigree information), (2) clinical observations of disease, or trait (phenotype), (3) sets of known polymorphic genome loci (genetic markers), and (4) information about the state of candidate loci for the individuals being studied (genotypes). There are two primary ramifications of these characteristics that distinguish this problem from most existing network-based genome analysis applications.

1. the need for privacy regarding pedigrees, genotypes, and phenotypes, and
2. the need to support a diverse collection of cooperating individuals in the gene isolation process.

The first implication requires protection of the data being used to perform analyses, and the second naturally suggests a heterogeneous, distributed solution. However, if these are the primary requirements of the solution, then they present an immediate conflict. Distribution of data to heterogeneous computation sites inherently involves taking risks with the security of data. The approach taken by GenoMap provides for security by:

- Verifying identities of individuals in the granting of *access* to sensitive data through the network:
 - Passwords protect access to the tools themselves.
 - Database contents can only be retrieved in the context of a single functional tool of the system. Thus users are prevented from making “custom” queries to extract information that would compromise individual privacy.
 - The database structure itself hinders the association of individual identities with their clinical, or genetic information.
- Databases containing sensitive data can be kept localized with the computations accessing them, thus allowing local administrative control, and allowing physical restrictions to be placed on the set of IP addresses allowed to access a particular database.

Most of GenoMap has been implemented in Java with a socket-oriented, client–server design employing recent applet security features [2]. The gene identification application supported by GenoMap is characterized by a two-stage process of data gathering

and verification, followed by an analysis phase known as *genetic linkage analysis* [6,8]. Java applets provide interfaces for specifying *linkage experiments*, support for management of the data collection and verification process, and interacting with statistical linkage analysis packages [3]. One of the data collection tools is a large C/X-windows application—GenoScape [9], for scoring gels with repeat markers [5], and many of the analysis packages are pre-existing and run in an UNIX environment (originally written in various languages—Fortran, Pascal, etc.). However, to support both security, interoperability, and sharing of the software among multiple laboratories and projects, a novel component of our system is a *socket server process* (SSP) that provides a naming service to the applets and applications, controls access to applications and data, and provides a load-balancing function.

Fig. 1 shows the main interface window that provides not only the common user view of the GenoMap system, but also serves as a central starting point for launching all GenoMap functions, and validating a user’s identity.

4.1. Requirements

The GenoMap system is designed to support a diverse collection of users in a wide-area network environment. The data objects being managed are of the most sensitive nature—usually identifying family relationships among members, some of whom may carry stigmatizing genetic diseases.

An additional requirement is that GenoMap be portable and sharable among multiple research laboratories. Due to the difficulties and costs associated with updating and distributing copies of software, we have decided that GenoMap must be a web-based system, i.e., whenever possible, the only static copies of GenoMap applets and applications will be stored on the web server at the University of Iowa. While appearing to be a potential bottleneck in several ways, in fact, this allows our group to make full use of the system locally, while not having to expend efforts supporting users outside of our local site with updates, releases, and copies of documentation. In fact, these tasks represent the primary reason that much university-based software is rarely used outside the domain in which it was created. In order to support the use of GenoMap at multiple, geographically

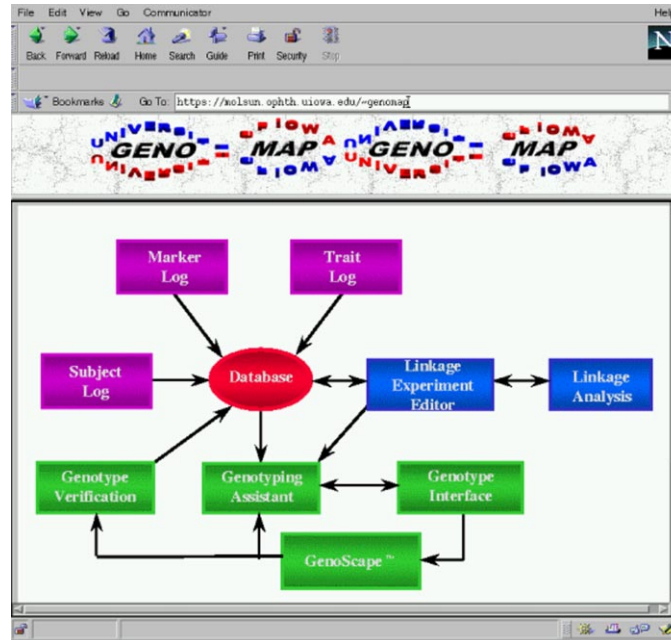


Fig. 1. User's WWW view of GenoMap.

dispersed sites we define the notion of a *database domain* or simply *domain*. Users register with the GenoMap system and are assigned to a domain. The domain corresponds to a database that is most likely stored and served on a machine within the administrative domain of the user. This greatly reduces the chances that security violations with respect to privacy of data will occur. However, it complicates the overall design of GenoMap, requiring that applets and applications be restricted to access only the domain in which they were originally instantiated.

Finally, GenoMap must interact with extant software. The software ranges from data collection packages on PC/Mac-based systems, to legacy codes in FORTRAN for genetic linkage analysis. For the immediate future, it is a requirement for GenoMap to directly be able to “wrap” the interface details of such packages into a Java client/server interface that makes them appear to be a Java applet.

4.2. Component organization

A typical *domain* (see Fig. 2) consists of the following components.

4.2.1. Services

GenoMap is a set of services to gather and manage data for, and to perform, genetic linkage analysis. A service in GenoMap consists of two or more components/resources; an applet which implements the user interface, a backend server which performs most of the computationally intensive operations, implements all file accesses, and the GenoMap Database which stores all the information required for the service. Not all services require a backend server.

4.2.2. User interface

The applets provide an interface for the users to access the services. The applets allow the users to provide information, connect to the GenoMap Database after proper authentication, conduct analysis on the gathered data, and connect to the backend server, if required.

4.2.3. Backend server

Most of the services require different modes of file accesses. The applets may have to store some information (e.g., results after analysis) in a file with a known name. The applet may need to load

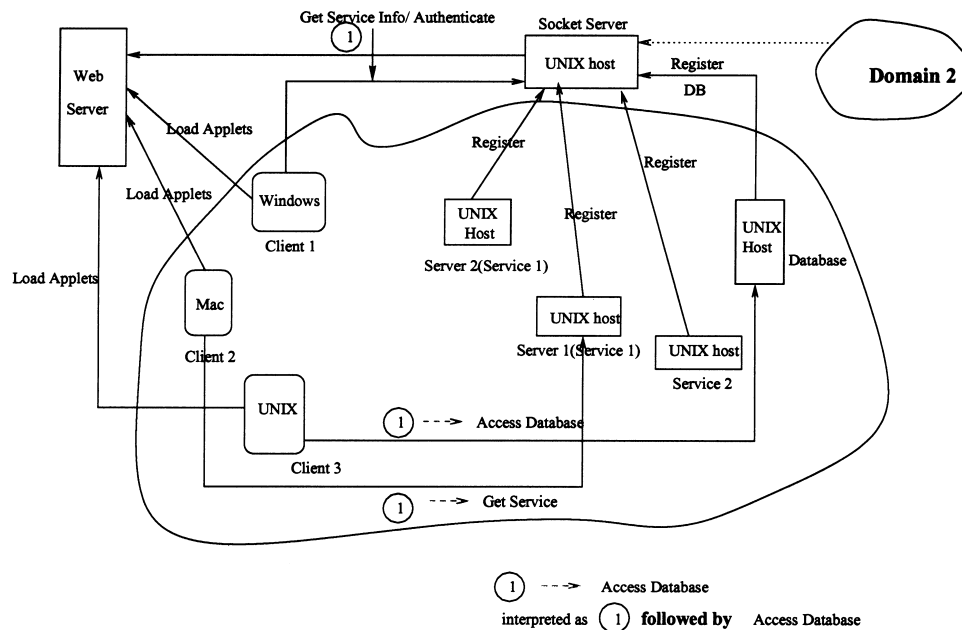


Fig. 2. Domain components.

a file (e.g., an image for processing). The backend server implements all the file operations required for GenoMap. The server also performs computationally intensive tasks for the clients.

The backend server listens for connections from applets, retrieves required files and performs the required computations. Each server offers exactly one service and the servers are independent. Multiple copies of the same server may be actively running for better performance. It is the job of the Socket Server to perform load balancing (described in Section 4.2.7) among multiple instances of a server.

4.2.4. Database

An important characteristic of any genome-related experiment is the need for a large volume of sensitive data. The data is usually stored in a database and the applets query the GenoMap Database for information pertaining to the service.

The GenoMap Database is an essential component and is used to form an administrative domain; i.e., only users belonging to that domain can access the data. Database access requires a separate level of authentication.

4.2.5. Web server

The web server is the starting point for the users to access all the services. It hosts all the user interface applets for the services offered by GenoMap. Fig. 1 shows the main interface for accessing the services of GenoMap. All the software is hosted in the web server and it is not local to each domain. Therefore, there is no need for software updates as it is maintained in the development site.

4.2.6. Clients

Clients are the users who access the services through WWW browsers. Typically, the users have widely different backgrounds. They can be geneticists, clinical physicians, statisticians, disease specialists, laboratory technicians, or computer scientists/engineers. They do not directly interact with the server or the database. Each user has privileges which determine which services he/she can access.

In general, GenoMap distinguishes three class of users:

- *Root*. Root has the highest privileges and the responsibilities include creation of domains, deletion of domains and changing domain parameters.

- *Domain Administrator.* The Domain Administrator (DA) is responsible for adding new users to, deleting users from, changing access rights for, and maintaining the domain. The DA also starts the backend servers for the services offered in the domain.
- *Users.* The people who actually use the services of GenoMap.

Fig. 2 depicts the different components explained above and the various activities that take place in a typical domain. Once a domain is created by the Root, the DA starts the backend server for each service offered in the domain. Depending on the need for a particular service, multiple instances of the server can be started. The backend servers register their availability with the Socket Server. The database location for the domain is also registered with the Socket Server.

The users access the services through their browsers from anywhere and from a variety of platforms (e.g., Windows, Mac, or UNIX). First, a user has to login to his/her domain and receive authentication from the server. The user is given an identity which is used to access the service through the applets. The applets may connect to the database and/or the backend server depending on the type of the service.

4.2.7. Socket Server

The Socket Server is the key element of GenoMap that combines the rest of the components, provides naming service to applets and servers, controls access to services and data, and provides a load-balancing function.

The Socket Server primarily performs the following functions: *Storage of information.* One of the important functionalities of the Socket Server is to store information about the various components of the system. It keeps track of the following:

- Identity of each backend server which includes:
 - the domain to which the server belongs,
 - the type of service it offers, and
 - the location information tuple $\{host, port\}$ upon which it listens for client requests.
- Database information includes the tuple $\{host, port\}$, the type of the database and its domain.
- Domain information such as the number of registered domains and per domain parameters.
- User information.

Load balancing. The backend servers are independent (possibly multiple) instances of the same server which can be started for a particular service. Depending on the need, the DA starts the required number of instances. The goal of load balancing in GenoMap is to evenly distribute the requests for servers among the many server instances. This is achieved by providing the appropriate server location information to applets. The Socket Server selects the appropriate server depending on the following parameters:

- number of client requests currently being serviced by each server,
- individual server parameters such as the maximum number of requests the server can handle, and
- idle time since last serviced request.

A LoadValue (LV) is calculated using the following equation:

$$LV = \sum_{i=1}^n W_i X_i, \quad (1)$$

where n is the number of factors used to calculate LoadValue (LV); W_i the weight (importance) assigned to that factor; X_i the normalized value of the parameter.

The initial Socket Server implementation considers the following simple factors:

$$X_1 = \frac{MaxNoOfRequests - NoOfRequests}{MaxNoOfRequests},$$

$$X_2 = \frac{IdleTime}{MaxIdleTime}$$

W_i are values ranging from 0, . . . , 1. For example, if $W_1 > W_2$, the Socket Server favors selecting the server whose LV is the highest.

Management of resources. The Socket Server manages addition/removal of domains. It allows for adding and removing users from a domain and changing their privileges. The database and all servers can be dynamically registered and unregistered. The Socket Server also accounts for failures during a transaction between a client and a server.

Authentication. GenoMap uses an authentication [10] scheme based on a tree structure (see Fig. 3). There are two categories of rights supported by GenoMap, administrative rights and access rights. *Root* is placed at the root of the authentication tree

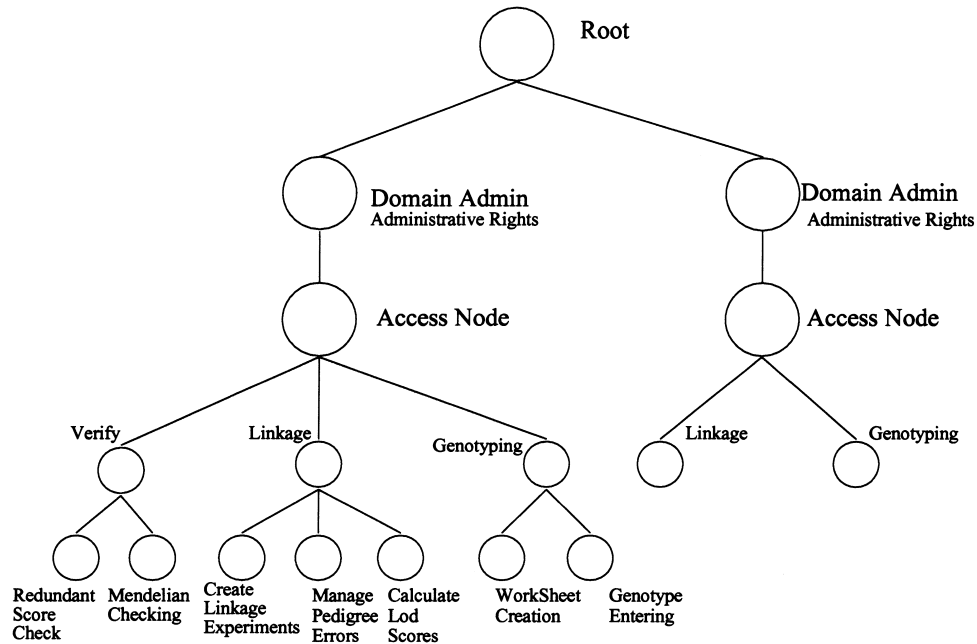


Fig. 3. Authentication hierarchy.

and has the maximum rights. An administrative node is created for each domain as a leaf of the root. The DA is placed in the administrative node and he/she has all administrative rights. Further, an access node is created for each domain which starts the sub-tree for access rights. Each domain is divided into logical sub-domains corresponding to the services offered in that domain. Each service is further divided into sub-domains giving finer control over the service. Users are placed in one or more nodes of the authentication sub-tree rooted at the access node. The user's access rights are assigned according to their position in the hierarchy. The closer the user is to the root, the more rights he/she has. The users at Level 1 have rights to access resources based on that level in addition to the rights of the sub-tree rooted at that node. The users at the leaves have the fewest rights. Each node is named using *prefix coding* and a user's right is just a string representing the user's position in the tree.

4.3. Scenario

Domain creation involves setting up the database and registering it with the Socket Server, starting the

servers (multiple instances to increase the availability) if required, and creating a user group with appropriate privileges. Once created the users can access the services using the client applets. First, the client authenticates itself with the Socket Server and then retrieves the server (hostname, port, etc.) and database information. The client may have to provide further authentication before being allowed to access the database's information.

Adding a new service is very simple and straightforward. All one has to do is to implement the user interface and the server (if required); register the service with the Socket Server and update the privileges for all users who require access to the new service.

4.4. Virtual domain

A virtual domain (Fig. 4) is formed by combining two or more existing domains to offer a new service which utilizes the resources of participating domains. A new user group is formed that has rights to access the services. The backend servers/clients can access the resources of the other domain only after proper authentication from the Socket Server.

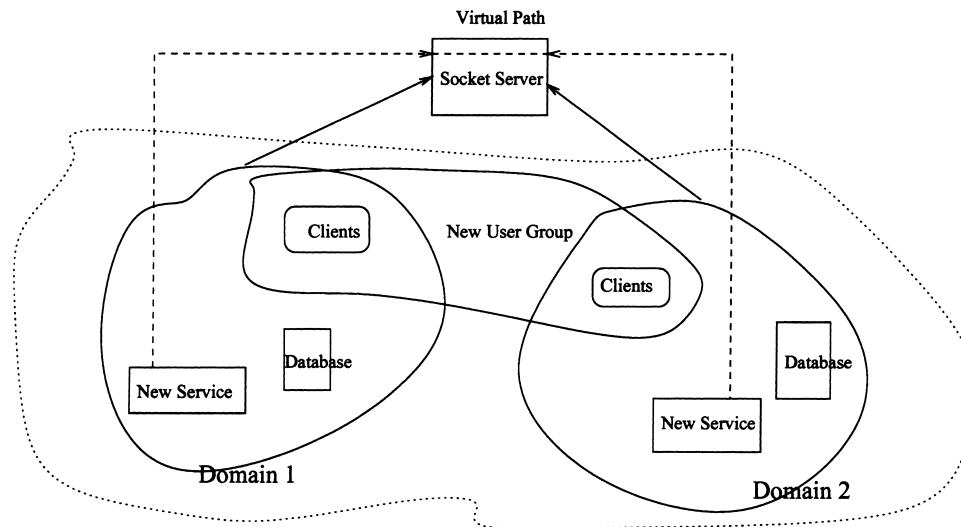


Fig. 4. Illustration of virtual domains.

5. Implementation and usage to date

Apart from specific implementation details, GenoMap is a networked application consisting of a set of interacting (1) Java applets and applications, (2) Java applets/applications and C/X-windows user interfaces, (3) Java and C applications that interact with local files systems, and (4) Java applets that interact with a database server and Java and C applications. Most of the clients are Java applets which provide various functionalities such as verification of genotypes to users. To meet the basic requirements outlined in the previous section, there are two different design approaches to this problem.

1. Clients contact their paired servers individually and registration is done on an applet/server pair basis. Servers then use “well-known” socket addresses to communicate with their client applet(s). This approach is the basis for our first implementation of GenoMap.
2. A separate SSP is implemented and registered with the web server. The Socket Server authenticates valid users of GenoMap, passes the location information for the applet clients needing to contact various backend servers and the GenoMap Database. The Socket Server also performs load balancing among multiple instances of backend servers.

Our first implementation of GenoMap is currently supporting a large autism gene identification study within the University of Iowa, Department of Pediatrics. We are presently expanding the set of users to include individuals and laboratories outside of our direct administrative control (e.g., a cooperating group at Tufts or Vanderbilt University). Thus, while this approach can be made to work in such an environment, the management requirements of this approach, and the limited ability to load-balance application activity, make the approach less attractive and scalable.

A second alternative represents an expandable architecture that naturally supports automatic load balancing and ease of management. We are currently experimenting with various load-balancing schemes and strategies to automatically recover from various failure scenarios [1].

6. Conclusion

The conflicting requirements of security and heterogeneity are at the heart of the approach taken in GenoMap. Our approach provides for security by verifying identities of individuals in the granting of access to sensitive data through the network. The database itself is physically partitionable across lab

boundaries, and access to a database is encapsulated within well-defined APIs.

GenoMap has been implemented primarily in Java with a socket-oriented, client–server design employing recent applet security features. GenoMap is currently being used in the collection of data for, and analysis of, a number of relatively small gene identification studies. It is also being used in one large genome-wide screening for the locus (loci) of the gene(s) involved in autism. The former shows its usefulness in supporting users who want to employ the analysis features of GenoMap, while not needing the large-scale data collection and management facilities, while the latter shows the usefulness in managing gene identification studies that would have been unmanageable without such a system.

References

- [1] T.L. Casavant, K.J. Munn, T.A. Braun, T.E. Scheetz, S. Kaliannan, An illustration of a parallel/distributed architecture for a hierachially heterogeneous web-based cooperative applications, University of Iowa, Iowa City, IA, Technical Report TR-ECE-981213. <http://www.eng.uiowa.edu/~tomc/papers/genomapLong.tar.gz>.
- [2] G. Cornell, C.S. Horstmann, Core Java, Prentice-Hall, Upper Saddle River, NJ, 1997.
- [3] R.W. Cottingham, R.M. Idury, Faster sequential genetic linkage computations, *Am. J. Human Genet.* 53 (1993) 252–263.
- [4] Department of Energy, Five Years of Progress in the Human Genome Project, *Human Genome News*, Vol. 7, Nos. 3–4, September–December 1995. Available via the WWW from [www.ornl.gov](http://www.ornl.gov/TechResources/Human_Genome/publicat/hgn/v7n3/04progre.html) in TechResources/Human_Genome/publicat/hgn/v7n3/04progre.html, September, 1997.
- [5] J.F. Gusella, N.S. Wexler, A polymorphic DNA marker genetically linked to Huntington's disease, *Nature* 306 (1983).
- [6] J. Lalonel, R. White, Analysis of Genetic Linkage, Emery & Rimoin's Principles and Practice of Medical Genetics, Churchill Livingstone, New York, 1996, pp. 111–125.
- [7] M.Y. Galperin, E.V. Koonin, Who's your neighbor? New computational approaches for functional genomics, *Nature Biotechnology* 18 (16) (June 2000) 609–613.
- [8] J. Ott, Analysis of Human Genetic Linkage, Johns Hopkins University Press, Baltimore, MD, 1991, pp. 108–141.
- [9] T.E. Scheetz, K.J. Munn, T.A. Braun, V. Sheffield, E.M. Stone, T.L. Casavant, GenoMap: a distributed system for unifying genotyping and genetic linkage analysis, *Parallel Comput.* 24 (1998) 1567–1592.
- [10] A.S. Tanenbaum, Modern Operating Systems, Prentice-Hall, Englewood Cliffs, NJ, July 1996, pp. 181–200.



Thomas L. Casavant received the BS degree in Computer Science, the MS degree in Electrical and Computer Engineering, and the PhD degree in Electrical and Computer Engineering from the University of Iowa in 1986. From 1986 to 1989 he was with the School of Electrical Engineering at Purdue University. In 1989, he joined the faculty of the University of Iowa where he is presently a Professor of Electrical and Computer Engineering, and Director of the Coordinated Laboratory for Computational Genomics and the Parallel Processing Laboratory. In 1993–1994 he was a Guest Professor with the Department of Informatik at the ETH in Zurich, Switzerland. Dr. Casavant has published over 50 technical papers on parallel and distributed computing, and computational genomics, and has presented his work at tutorials, invited lectures, and conferences in the United States, Asia and Europe. Dr. Casavant is a Senior Member of the Institute of Electrical and Electronics Engineering (IEEE), has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Processing* and *The Journal of Parallel and Distributed Computing* (JPDC). He has served as Guest Editor and/or Program Chair for numerous other journals and conferences. His current research interests include large-scale methods for gene discovery, mapping, and disease gene identification, parallel computer architecture, scheduling, trace recovery, and visualization.



Terry A. Braun received the BS degree in Electrical Engineering (1993), and the MS degree in Electrical and Computer Engineering (1995) from the University of Iowa. Currently he is a PhD candidate in Genetics at the University of Iowa. He has co-authored several papers in the area of high performance computer architecture before research interests switched to genetics. Current research interests include disease gene identification and discovery using sequence analysis, and linkage analyses and association techniques for complex diseases.



Todd E. Scheetz received the BS degree (1993) in Electrical Engineering and the MS degree (1995) in Electrical and Computer Engineering from the University of Iowa. He is currently enrolled in the PhD program in Genetics at the University of Iowa. He has co-authored several papers in the area of high performance computer architecture and parallel systems. Research interests include bioinformatics, parallel and distributed processing, and operating systems.



Kyle J. Munn received the BS degree in Biomedical Engineering, and the MS degree in Electrical and Computer Engineering from the University of Iowa in 1999. Currently he is an interdisciplinary PhD candidate in Computational Genetics (Electrical & Computer Engineering) at the University of Iowa. He has

co-authored several papers on computer-aided diagnosis and informatics for supporting large-scale, high-throughput linkage analysis studies. Current research interests include informatics support for large-scale gene discovery studies and novel expression data analysis.



Clayton L. Birkett received the BS degree in Electrical Engineering from Virginia Tech in 1985. He worked as a Biomedical Engineer for 2 years at the Veterans Administration Hospital of Richmond, Virginia. In 1987 he moved to Iowa City, IA where he worked for 8 years as a Biomedical Engineer in the Department of Cardiology at the University of Iowa Hospital. In the Cardiology Department he did research in nerve recording analysis, cardiac defibrillation, and intra-arterial imaging. In 1991 he received a MS in Biomedical Engineering from the University of Iowa. Since 1997 he has worked as a Senior Programmer Analyst at the University of Iowa in the Department of Electrical Engineering. He is currently working in the Computational Genomics Lab on gene discovery and mapping projects.