A Survey of Distributed Systems in Bioinformatics

Fabrice Bernardi[1] Jean-François Santucci[2]
David Hill[3]

Research Report LIMOS/RR04-02

19 janvier 2004

[1]bernardi@isima.fr
[2]santucci@univ-corse.fr
[3]drch@isima.fr

Abstract

There is no doubts that bioinformatics is and will be one of the major application domains in computational power needs and researches. Very extensive databases and power-consuming algorithms have been designed and the most promising approach to provide solutions for data-mining or computation is to use distributed systems. This term is widely used and englobes various specific approaches. We propose in this article a survey of some distributed systems that are used or applied in bioinformatics. We cover basic Client/Server and Peer-To-Peer systems, Clusters and Grids approaches, but also Distributed Agents systems. There is one main voluntary limitation to this review: we focus only on available/reusable, Open-Source, free or educational solutions.

Keywords: Bioinformatics, Distributed Systems, Review, Grid, Cluster, Agent, P2P

Résumé

Il ne fait aucun doute que la bioinformatique est et sera un des domaines d'application majeurs dans la recherche de puissance de calcul. De très grandes bases de données et des algorithmes très coûteux en temps CPU ont été définis et la meilleure solution pour répondre aux problèmes posés est d'utiliser des architectures distribuées. Ce terme est très couramment employé actuellement et englobe différentes approches. Nous proposons dans cet article un état de l'art de systèmes distribués en bioinformatique. Nous couvrons les systèmes Client/Serveur basiques et Peer-to-Peer, les Clusters, les Grilles, mais également les systèmes d'agents distribués. Il existe une limitation volontaire à ce travail : nous nous concentrons uniquement sur les solutions disponibles/réutilisables, Open-Source, gratuites ou du domaine de l'éducation.

Mots clés : Bioinformatique, Systèmes distribués, État de l'art, Grille, Cluster, Agent, P2P

1 Introduction

Bioinformatics, defined as "the application of information science and technology to the management of biomedical data and information" by [26] is currently a vast research field. Primarily defined by biologists, many computer scientists have been convinced to work on this new science. Actually, some papers specially intended to them can be found in the literature, like [38] or [44].

One of the main characteristics of bioinformatics is the huge amount of available data stored in very large public databases. The problem is so important that many researches are performed to provide effective solutions. More information can be found in [74], which proposes a review for integrating the various existing databases. Thus, one of the most important concerns of bioinformatics is to retrieve and to exploit these databases which contains many information about biological entities and can be used for instance for gene sequence finding and recognition by alignment as shown by [83]. This amount of data and computational needs makes it possible (and necessary) to use distributed or parallel architectures as shown by [80], [86] or [24]. [81] and [49] provides a lot of links of bioinformatics tools and shows how extensive the domain is.

We propose in this article a survey of some distributed systems applied in bioinformatics. There is one main voluntary limitation to this review: we focus only on available/reusable, Open-Source, free or educational solutions. Some proprietary or non-free solutions exist, but do not appear here.

This article is articulated following three sections. In a first one, we provide an historical perspective of the development of computer networks, and especially of the Internet. In a second one, we give a brief taxonomy of distributed systems as they are usually defined. We give the definition of Client/Server systems, Peer-To-Peer systems, Clusters, Grid, Agent-based distributed systems and describe the three main communication models that can be used. The expert reader could note some debatable points in this section (like distinguishing grids and Peer-To-Peer systems), but this is the best way we found to classify the various toolkits able to be used in bioinformatics. This classification is covered by a third section that describes some systems used in the bioinformatics area. We can note that we do not give very precisely how these systems are used in bioinformatics, but we provide many references in this objective.

2 Historical Perspective

If we had to trace what could be the beginning of distributed systems, we have to recall to our mind that at the beginning of the sixties, Leonard Kleinrock a scientist working at MIT, published a paper on packet switching entitled "Information flow in large communication Nets" (see [54]). Still at MIT, J.C.R. Licklider envisioned a global interconnected set of computers where everyone could access programs and information from any sites. The first book on the packet switching theory was published in 1964, again by Kleinrock (see [55]).

In 1965, the first wide area network was built with a low speed telephone line connecting Massachusetts and California time-shared computers that were able to work together. Also in 1965, the US Department of Defense Advanced Research Project Association (DARPA), started to work on ARPANET (the Advanced Research Program Agency Network). In 1966, the concept of "computer network" was introduced at DARPA and published in 1967. In 1969, the File Transfer Protocol (FTP) helped in distributing the RFCs (Request For Comments) which were fast and easy ways to share ideas amongst network researchers. It was also at the end of the sixties that Ken Thompson, Dennis Ritchie and others started to work on what will become Unix at AT&T Bell Laboratories. At the same time in 1969, ARPANET was connecting 4 universities & research institutes (Stanford Research Institute, UCLA, UC Santa Barbara and the University of Utah).

In 1972, the InterNetworking Working Group (INWG) was retained to control the growing network and the first e-mails were introduced to public. The next year saw the introduction of the Ethernet Protocol at Xerox PARC (Palo Alto Research Center). Year 1974 saw both the arrival of Telnet, a commercial version of ARPANET and the rewriting of Unix V.4 with the C language, thus enabling a better portability of this operating system among different computers. In 1979, the USENET newsgroup was introduced by students from North Carolina and Duke Universities. The beginning of the eighties saw the spreading of IBM personal computers and its compatibles.

The term Internet was introduced in 1982, and in 1983, the old NCP (Network Control Protocol) introduced in 1970 was replaced by the current TCP/IP protocol. Five years later, in 1988, the spreading of the Internet Worm showed the vulnerabilities of the Internet, thus leading to the formation of the Computer Emergency Response Team (CERT) to address security issues. At the Palo Alto DEC Research Center, the distribution of computing tasks sent via e-mails to workstations inside and outside their laboratory was successfully achieved in 1988 (results were also collected via e-mails). A few years later, in 1991, the concept of World Wide Web is born and in the same year Linus Torvalds distributed the source code of the Linux kernel as freeware.

In 1993, the first Web browser, named Mosaic is released and the first RSA Security challenge is won by a team of 600 volunteers providing their distributed computing resources. In 1994, version 1.0 of Linux, produced by Linus Torvalds and a world wide team of hackers, is available for download. The annual growth rate of the Web between 1993 and 1994 is over 300%. The following year saw the introduction of an alpha version of Java, an object-oriented language with a portable byte-code able to be transmitted and executed on the Internet. The GIMPS distributed computing project, searching for larger and larger prime numbers is launched in 1996. On August the 26th of 1997, the PI challenge is launched by Fabrice Bellard to find the 1000th billion binary digit of PI, this goal is reached on September 22nd.

In 1999, the concept of Internet computing was providing at least 3 times more computing power (33 Teraflops) than any other supercomputers with the Seti@home project of Berkeley University, which analyzed data from the famous Arecibo radio-telescope. Clients software were available for PC, Macintosh and Unix. More useful projects appeared with the new millennium. In year 2000, the Casino-21 project goal was to simulate the dynamics of the earth's climate for the next century. In year 2001, Barabasi from Notre Dame University introduced in [11] and [10] the concept of parasitic computing, explaining and showing how Internet protocols could be used to compute complex linear programming problems. More recently at the end of year 2001, the French Decrypton project was setup to compute and compare 50000 protein sequences with a distributed client software developed by Genomining and IBM. More than 75000 Internet users participated to reach the first results in 2 months, 1170 years would have been necessary to achieve the same result on a single computer. The collected data is now publicly available for the international community of scientists working in bioinformatics.

In the next section, we propose a taxonomy of distributed technologies that have emerged following all these events.

3 Taxonomy of Distributed Technologies

Many presentations and definitions of distributed systems can be found in the literature (see [80], [34], [35] or [52] for a quite complete list) but we choose to present them following four axes: classical client/server systems, massive distributed architectures, agent-based systems and the communication models able to be implemented in these kinds of systems.

3.1 Basic Client/Server-based Model

The classical Client/Server approach is the most used nowadays by the Internet community (Web, FTP,...). For [63], this kind of remote operations are “the most basic form in distributed systems”. The term Client/Server was first used in the early 80s in reference to personal computers on a network.

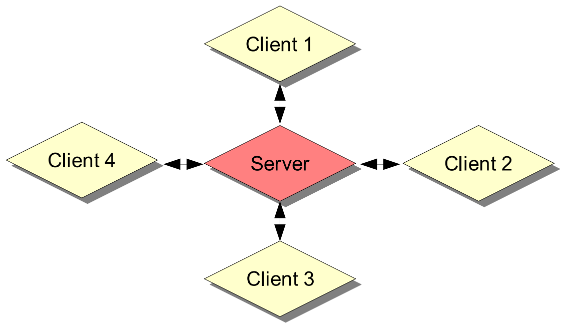

Figure 1: Basic Client/Server Model

As shown in Figure 1, the operating scheme is very simple: one process (the Client) sends a message to another (the Server), asking it to do some work. Once the result has been processed, the Server sends back a message and the result of the work. Most of the classical protocols (HTTP, FTP,...) are based on this concept, and a lot of techniques have been developed in order to improve performances: buffers, caches,... Any entity in such a system can play both roles but for a different purpose, i.e. server and client functionality residing on separate nodes.

This concept of Client/Server, even if very simple, can be seen as a theoretical basis for many (and much more complicated) distributed approaches.

3.2 Metacomputing Architectures

Metacomputing is also known as “Network-based Distributed Computing”. A rough definition of a metacomputing system can be: a networked virtual supercomputer which shares resources. For [48], its main motivation (with respect to supercomputing) was driven by “the recognition that the whole can be greater than the sum of the parts”. The operating scheme is to connect heterogeneous computing resources in order to form a super-computational one. Another motivation was the geographically separated clients and their need to collaborate via the network. In such a scheme, computers or clusters play the similar role as microprocessors within supercomputers.

Peer-To-Peer systems, Clusters and Grids are forms of Metacomputing systems.

3.2.1 Peer-to-Peer, Hybrid Peer-to-Peer Systems and Internet Computing

A Peer is defined as an entity with capabilities similar to other entities on a system. Peer-To-Peer (P2P) applications are essentially used for data storage and exchange as shown by [12] or [61]. Even such systems are very popular nowadays, the concept is not all that new: the underlying technologies have been defined at least at the same time with USENET.

The basic principles of this approach is to avoid network vulnerabilities and to discover resources by diffusion. Data are often associated with meta-descriptions allowing them to be accessed using search engines or Web-based front-ends. We can draw a distinction between "pure" and "hybrids" P2P applications (see Figure 2). In the first case, all participant computers are peers. In this scheme, no central server is used to control, coordinate or manage the exchanges among the peers. In the second case, the application relies on a central server to perform some of the required functions. The degree of involvement varies with the application.

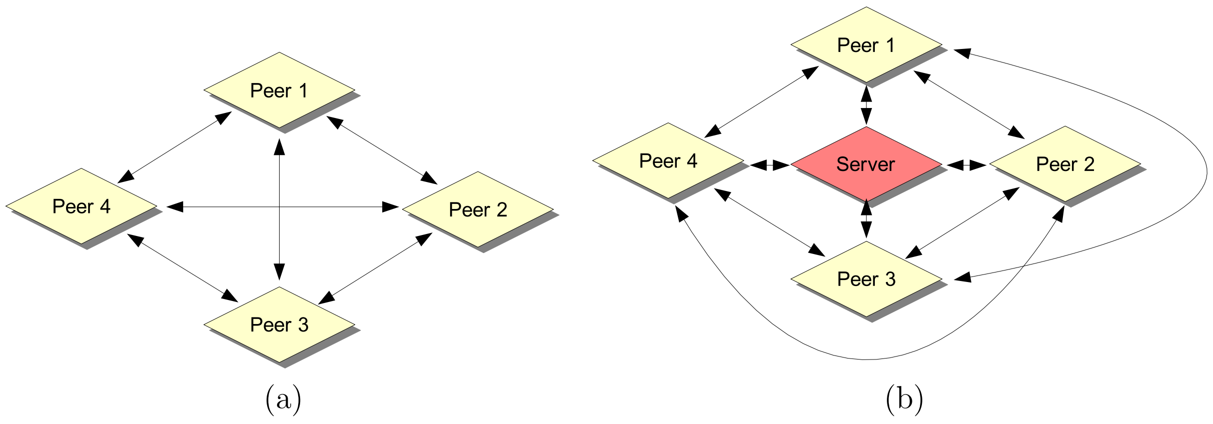

Figure 2: "Pure" (a) and "Hybrid" (b) Peer-To-Peer Models

Several goals can be achieved using P2P. The first one is cost sharing/reduction, since centralized systems that serve many clients typically bear the majority of the cost of the system. Secondly, P2P applications improve scalability and reliability, since peers are autonomous and do not depend on a strong central authority. The third goal is to aggregate resources and to allow interoperability since a centralized approach lends itself naturally to an aggregation of resources. P2P applications increase also autonomy since, in many cases, users of a distributed system do not want to rely on any centralized provider. This autonomy is completed with an anonymity and privacy if one do not want anyone to know about his involvement in the system. The fifth goal is dynamism, since P2P systems assume that the computing resources will be entering and leaving the system continuously. Finally, the last goal of P2P systems is to enable effective communication and collaboration.

Internet Computing applications can be seen as parallelizable P2P applications, where the same task is performed on each peer using different parameters. The principle is to use resources of idled computers on the Internet in order to compute a specific task. As soon as a node computer detects an inactivity it contacts a master client and informs it of its availability for computing some work. Thus this new activity is totally transparent for the user.

There are many implementations of such systems: KaZaa, FreeHeaven, JXTA, Gnutella, Freenet ("Pure" P2P), Napster ("Hybrid" P2P), SETI@Home, RSA-135 (Internet Computing).

3.2.2 Clusters

A cluster connects complete computers (including processor, memory, I/O units) and its component computers are loosely connected typically by a LAN as shown in Figure 3. It is used as a single, unified computing resource, but is not distributed geographically. Component computers of a cluster are typically workstations or PCs and are generally homogeneous, and the first so-called "Beowulf" cluster has been designed in 1994 for the NASA by Becker and Sterling (see [75]).

A cluster architecture is divided in three non-hierarchical logical tiers: the Access Tier providing access and authentication services to the users; the Management Tier responsible for providing the basic cluster services (file service, backup management,...); the Compute Tier supplying the compute power for the cluster. Jobs submitted through upper tiers in the architecture are scheduled to run on one more nodes in the Compute Tiers.

The two critical cluster services are the Batch Queuing System and the Job Scheduler. The first one is a service that receives job submissions from users and executes them on the cluster. This software (without supplemental software) runs jobs on a first come, first serve basis. The second one is a priority based scheduling service that determines the order of execution for jobs based on a wide range of criteria: average CPU usage, average resources usage,...

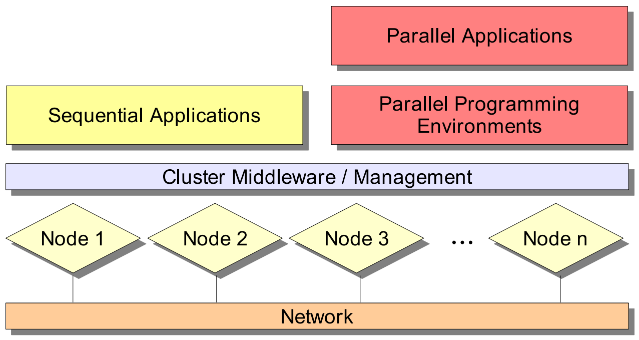

Figure 3: Clusters Architecture

3.2.3 Grid Computing

Grid computing is defined as “coordinated resource sharing and problem solving in large, multi-institutional virtual organization” by [8]. The term Grid suggests a computing paradigm similar to an electric power grid. This is a very new concept and is currently under a heavy phase of Research and Development. A quite complete review can be found in [2], [18] provides an economical perspective of this kind of systems, while [69] provides a review of Grids in the health sector.

Grid applications couple resources that cannot be replicated at a single site or may be globally located for other reasons. Thus, a Grid is an infrastructure that can unify globally and diverse resources. The main issues for computational grids are heterogeneity since a grid involves a multiplicity of heterogenous resources; scalability since a grid might grow from few resources to millions; dynamicity or adaptability since the probability of some resource failing is high with so many resources.

There are four components necessary to form a grid (see Figure 4): a Grid Fabric comprising all the resources (computers, clusters,...) geographically distributed and accessible from anywhere on the Internet; a Grid Middleware offering core services such as remote process management, co-allocation of resources, storage access,...; a Grid Development Environment offering high-level services that allows programmers to develop applications and brokers that act across global resources; some Grid Applications and Portals developed using Grid-enabling languages (such as MPI).

There are many grid projects worldwide. We can cite Globus, Legion, CERN Data Grid, UNICORE,...

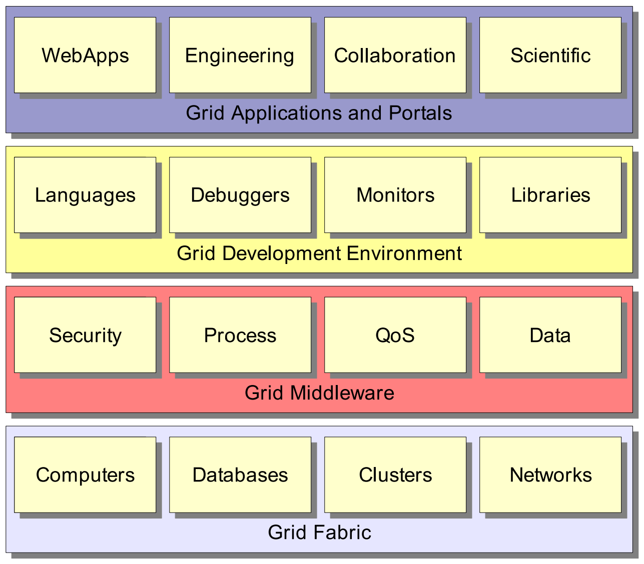

Figure 4: The Four Grid Components

3.3 Agent-based Distributed Systems

The goal of this kind of systems is to implement processing algorithms in parallel using a family of mobile agents as described by [40]. At the system level, task parallelism represents an Object-Oriented design strategy, whereby both data and the operations on those data are encapsulated in a single entity. Thus each task has the ability to operate independently, and yet communicate and exchange data with other tasks. In this context of parallelism, an Agent is a natural task abstraction. The Task Agent encapsulates the task data, operations to be performed on that data and the necessary communication mechanisms. Figure 5 presents a Distributed Agent-based architecture.

Task Agent are associated with two components: the Agent Link serving as the Agent's communication interface and the Agent Monitor performing fault detection by monitoring the parent and the children. They are supported by an infrastructure composed by various service components. Each service is available on each node in the system so as to maintain a completely distributed, fault tolerant execution environment. An example of one such service is resource supervision that integrates availability and load balancing components. A resource directory (Registry) is maintained on each node that contains load and state information. The directory is updated via a discovery process when new resources are added to the system. When a node becomes unavailable, the local registry is updated by the first agent that has

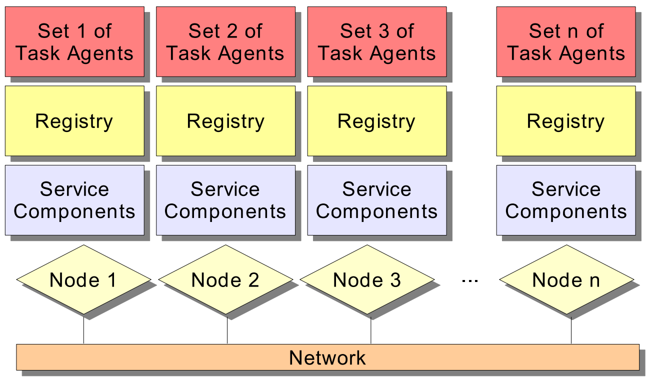

Figure 5: Agent-based Distributed Systems Architecture

attempted to communicate with an agent on that remote node.

As implementations,we can cite the NZDIS Project, ANAISoft, FIPA or DReAMS.

3.4 Communication Models

3.4.1 Message-Passing Models

The message passing model relies on the idea that independent, sequential processes running in parallel can communicate with one other via messages in order to jointly solve a parallel algorithm. Two *de-facto* standards have arisen: the Parallel Virtual Machine (PVM, described in [37]) and the Message Passing Interface (MPI, fully documented in [62]), and represent two models.

The central concept of the PVM model is the notion of Virtual Machine that enables the usage of a heterogeneous system containing different types of computing nodes. PVM provides a dynamic programming environment where both processes and computing nodes (called hosts) can be dynamically added or deleted either from the application program or from a system console. The other important issue in PVM is the highest possible level of support for interoperability both at the programming language level and in the communication system. PVM provides also the necessary message format transformation to hide differences in computer architectures.

The fundamental innovation in the MPI communication concept is the introduction of the notion of communicator which is a binding between a communication context and a group of processes. In the MPI approach, point-to-point process communication is allowed only within a group and for each group there is a unique communicator allocated by MPI. The two main advantages of MPI over PVM are that its implementations are more efficient since it support native communication layers of parallel computers, and that it provides a much richer set of library functions for point-to-point and collective communication operations.

The lack of interoperability of MPI excludes different MPI application programs to communicate by message passing. The current PVMPI described in [32] solves this problem by proposing a combination of MPI and PVM process groups in a way that MPI groups can dynamically join and leave PVM groups that serve as communication relays for the MPI application process groups.

3.4.2 Object-Oriented Middlewares

For [9], "middlewares are a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems". Object-Oriented Middlewares are used as an implementation layer over the network layer for many distributed systems. They are defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system (see Figure 6). They resolve heterogeneity of hardware, operating systems, networks or programming languages and, thus, can be considered as software that make a distributed system programmable.

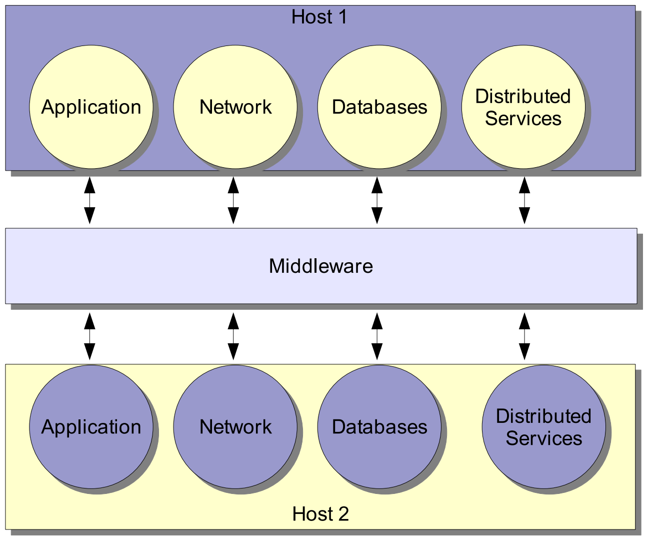

Figure 6: Principle of Middlewares

There are six fundamental middleware subcomponents: database gateways, network gateways, transaction processing monitors, message-oriented middleware, object request brokers and distributed services. Database and network gateways serve as translators to connect dissimilar processes, protocols or platforms. Transaction processing monitors, message-oriented middlewares and object request brokers distribute and manage transactions and messages. Distributed services deliver access to specialized system resources such as print, file, security and directory services.

The basic Object-Oriented concepts of data encapsulation, inheritance and polymorphism provide a solid basis to separate the specification of computation from the implementation. Currently, three major Object-Oriented systems are used: Java Remote Method Invocation (RMI), the Common Object Request Architecture (CORBA) and the Distributed Common Object Model (DCOM/COM+).

3.4.3 Web Services

Web Services are founded on the concept of "Service-Oriented Architectures" which enables the greater use of existing applications by creating services that can be reused. Thus a Service-Oriented Architecture minimizes the need for extensive rewriting of code. Web Services are services offered across a network using existing Web infrastructure. They are based on industry standards and are interoperable whitin company boundaries, or across many companies. They are also cross-platform and cross-language. Their main benefits are that they are rapid to deploy and that they increase value of existing applications.

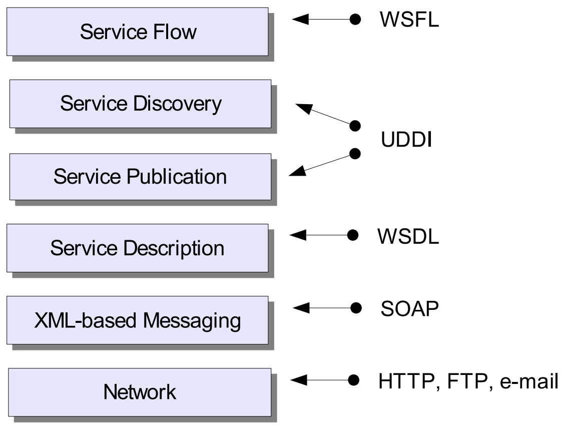

Figure 7: Layers of Web Services

Web Services are constituted of six layers as shown in Figure 7: Service Flow, Service Discovery and Service Publication, Service Description, XML-based Messaging and finally Network. These layers are implemented using five main de-facto standards. WSFL (Web Services Flow Language) is an XML-based language for the description of Web Services compositions as part of a business process definition (Service Flow). UDDI (Universal Description, Discovery and Integration) is an XML-based registry for businesses worldwide, which enables them to list themselves and their services on the Internet (Service Discovery and Service Publication). WSDL (Web Services Description Language) is an XML-based language used to describe the services a business offers and to provide a way to other businesses to access these services (Service Description). SOAP (Simple Object Access Protocol) is a way for a program running on one kind of operating system to communicate with a program running on the same or on another kind of operating system by using HTTP and XML as mechanisms for information exchange (XML-based Messaging). Finally, HTTP, FTP or e-mail are the choices for the networking layer. This kind of architecture is well described by [45].

Implementations and applications are currently being developed. We can cite Microsoft's .Net and its Open-Source implementation MONO, or Sun's Open Net Environment (ONE).

4 Distributed Technologies for Bioinformatics

In this section, we present systems classified following the section 3 (in the alphabetical order) that are employed in the bioinformatics field. We can note that some of them are not specially defined for this research field but they all have been used in this area.

4.1 Classical Client/Server Systems

Classical Client/Server applications are the most used in the bioinformatics world. The main reason is that the available public databases can be accessed using Web-based frontends and a simple Web browser. Even if the hidden framework of these databases is large and complex, researchers can use them in a very easy and user-friendly way. This kind of use is certainly the optimal one for classical Client/Server systems since they can not provide computational resources.

3D-PSSM: 3D-PSSM is a Web-based tool dedicated to protein fold recognition using 1D and 3D sequence profiles. The website proposes two query submission pages, a simple one and a more advanced one. [51], [14] and [50] present this tool and how it can be employed.
Main URL: `http://www.sbg.bio.ic.ac.uk/~3dpssm/`

BLAST and Variants: BLAST (Basic Local Alignment Search Tool) is a heuristic method to find the highest scoring locally optimal alignments between a query sequence and a database . It is without any doubts the most commonly employed tool in bioinformatics and is usually runned using a Web-based frontend. The most known is at NCBI, but many frontends can be found (like EBI's). Many references on BLAST can be found in the literature. We can cite [5] and [6] as founding articles, and [58] or [53] for more practical information.
Main URL: `http://www.ncbi.nih.gov/BLAST/`

Clustal W: Clustal W is a multiple sequence alignment software for DNA or proteins. As for BLAST, many Web frontends can be found. We can cite EMBL-EBI's, CMBI's or GenomeNet's. This software is presented in [79] or [20], while [56] proposes an MPI-based implementation. As an example of application, we can cite [19].
Main URL: `http://www.ebi.ac.uk/clustalw/`

COMBOSA3D: COMBOSA3D is a tool allowing to color protein structures using sequence alignment information. The Web-based interface uses the Chime or RasMol browser plugins for the visualization. A presentation of this tool can be found in [76].
Main URL: `http://bioinformatics.org/combosa3d/`

DAS: DAS (the Distributed Annotation System) is a client-server system allowing to exchange annotations on genomic sequence data. The website provides an inventory of some DAS public servers. The main reference of this tool is provided by [28].
Main URL: `http://biodas.org/`

HMMER: HMMER is an implementation of profile hidden Markov models for biological analysis. It is distributed as a software package, and some Web-based frontends can be found. We can cite for example the IFOM HMMER Server or NPS@'s. [30] provides a complete manual of this system, while [31] and [88] propose some complementary high-level descriptions and applications.
Main URL: `http://hmmer.wustl.edu/`

Swiss-Model: Swiss-Model is an automated protein structure homology-modeling server. It proposes a set of Web-based tools like Swiss-PdbViewer (for viewing and manipulating protein structures and models) or search engines and is described in [73], [43] or [70].
Main URL: `http://www.expasy.org/swissmod/SWISS-MODEL.html`

T-Coffee: T-Coffee is also a multiple sequence alignment package for DNA or proteins. EMBnet or Notre Dame University provide Web-based frontends to this tool. We can cite [66] and [65] as main references.
Main URL: `http://www.ch.embnet.org/software/TCoffee.html`

4.2 Peer-To-Peer Systems

Peer-to-Peer systems are beginning to be used in the bioinformatics area, mainly in order to provide an easier data access between connected hosts. These data can be classical data of biological databases or even user-defined ones, and are often encapsulated in generic formats (often XML-based). A good starting point for this kind of systems is [68].

BioMOBY: BioMOBY is a workflow framework. Its primary goal is to allow a client to interact with multiples sources of biological data regardless of the underlying format or schema. A unique object format is defined, MOBY-Object, and three entities are used: one MOBY-Central host, one or more MOBY-Server hosts and one or more MOBY-Client hosts. The MOBY-Central receives declarations of services provided by MOBY-Servers and requests from MOBY-Clients that are passed to MOBY-Servers. BioMOBY uses Web-Services technologies (WSDL for service description and XML as MOBY-Objects). BioMOBY is presented in [85] and also in [21].
Main URL: `http://biomoby.org/`

BioPipe: BioPipe is also a workflow framework designed to work intimately with the BioPerl package and is able to handle various input and output data from various databases. Another goal of BioPipe is to provide an explicit protocol-based approach to bioinformatics analysis, in order to facilitate data interpretation and discussion by external parties. Four main components are used: an Input Layer that fetches the inputs to an analysis, an Analysis Layer that specifies the expected inputs from the Input Layer, an Output Layer that stores the output from the analysis layer, and finally a Job Management System (implemented using the MySQL database) that ensures that the job is well maintained as it

goes through each of the three layers.
Main URL: `http://www.biopipe.org/`

GreenTea: GreenTea is a Java-based generic runtime that facilitates the services of P2P computing. It is a total resource aggregation and resource sharing platform including hardware and software resources (in fact, it can be seen also as a Grid OS). While being not specially bioinformatics-oriented, it has been successfully used to run NCBI's standalone BLAST, and serves currently as a basis to the ABSmith project (A Smith-Waterman Algorithm).
Main URL: `http://www.greenteatech.com/`

4.3 Clusters Systems

As expected, clusters are used for their computational power. We can note that the Linux OS is mainly used in these kind of architectures. The main starting point in this optic is [57].

BeoBLAST: BeoBLAST is a package for performing distributed BLAST and psi-BLAST runs on a Beowulf cluster. The integrated Web-based frontend allow users to select multiple databases or to specify multiple queries. This software is based upon the GNU Queue, the GNU load-balancing system, and uses local copies of sequence databases retrieved from NCBI using a Unix cron job. The system is described in [41].
Main URL: `http://bioinformatics.fccc.edu/software/OpenSource/beoblast/beoblast.shtml`

BioBrew: BioBrew is a Linux-based Open-Source bioinformatics cluster solution. It contains all necessary cluster and bioinformatics software. MPI or PVM can be used as communication models, the Sun Grid Engine (SGE) is included as well as all the necessary software to run Ganglia (distributed monitoring and execution system), PVFS (Parallel Virtual File System) or Globus. It has been successfully employed with the NCBI Toolkit, mpiBLAST, HMMER, Clustal W or FASTA. BioBrew is described in [67]
Main URL: `http://www.callident.com/`

4.4 Grid Computing Systems

Grid computing is certainly one of the most exciting and promising improvement of Internet-based tools. All the systems presented here are currently in research and development phases (even Globus which is by far the most advanced and used), but some of them have been employed in very strong and precise researches (see references). More grid projects can be found at [71], at [42] or at [29], and many of the systems presented here are described more precisely in the excellent review provided by [47].

ALiCE: ALice (Adaptive and scalable Internet-based Computing Engine) is a platform-independent, Java based grid middleware. It comprises three types of entities: a resource broker, service producers and service consumers.
Main URL: `http://www.comp.nus.edu.sg/~teoym/alice/alice-tech.htm`

Condor: Condor is defined as a specialized workload management system for compute-intensive jobs. The goal of the project is to develop mechanisms and policies that support high-throughput computing on large collections of computing resources. [78], [13] and [77] present this project.
Main URL: `http://www.cs.wisc.edu/condor/`

Discovery Net System: Discovery Net System is a grid middleware enabling the composition of reusable workflows that can be later redeployed as new services for other uses. This system is described in [72] and [25].
Main URL: `http://www.discovery-on-the.net/new/index.php`

Globus: Globus is the most known and used of grid-enabling systems. It is released as a toolkit proposing a set of components that can be used independently or together. We can cite as components a resource allocation manager, security infrastructure, a monitoring and discovery service or a failure detector as described in [36] or [3]. Globus 3.0 is the first full major implementation of the Open Grid Services Architecture (OGSA well described by [27] and presented also in [4]), an open standard for grid computing. As an application of Globus in the Bioinformatics area, we can cite [33].
Main URL: `http://www.globus.org/`

GrADS: GrADS (Grid Application Development Software) is a multi-university project intending to provide tools and technologies for the development and applications in a grid environment. In this system, the user simply presents his parallel application to the framework. This framework is then responsible for scheduling the application on an appropriate set of resources and launching and scheduling the execution. [15] presents

the project while [87] provides an example of use of this system for sequence alignment.
Main URL: `http://nhse2.cs.rice.edu/grads/`

Grid Engine: Sun ONE Grid Engine is available in two versions: SGE and SGE Enterprise Edition. This middleware requires no alterations to applications to be distributed. More information can be found in [22] and [23].
Main URL: `http://wwws.sun.com/software/gridware/`

GriPhyN: GriPhyN is a collaboration of computer science researchers and experimental physicists to enable a grid system following two approaches: apply a methodical and organized data management using the virtual data concept and developing automated request scheduling mechanisms in order to make large grids easy to use as single computers.
Main URL: `http://www.griphyn.org/index.php`

Legion: Legion enables the connection of networks of computers using an independent element approach. There is no central element and the proposed framework permits scheduling and distributing processes as in a single, virtual machine. More information can be found in [64], and an example of use can be found in [84].
Main URL: `http://www.cs.virginia.edu/~legion/`

Metacomputer OnLine: Metacomputer OnLine (known as MOL) is designed as an open, extensible software system comprising a variety of software modules. In contrast to other grid environments, MOL is not based on specific models or tools.
Main URL: `http://www.uni-paderborn.de/pc2/projects/`

NetSolve: NetSolve is a grid framework focusing on three main points: ease-of-use for the end-user, efficient use of the resources and the ability to integrate any arbitrary software component as a resource into the NetSolve system. More information, as well as a grid concept description can be found in [7] or [1].
Main URL: `http://icl.cs.utk.edu/netsolve/`

OBIGrid: OBIGrid is a grid network providing bioinformatics-oriented services: a genome annotation system, a gene network simulator, a bioinformatics dispatcher and a scalable genome database.
Main URL: `http://www.obigrid.org/`

Polder: Polder is a metacomputing environment concerned with the optimal mapping of models onto parallel computer systems. The first studies were based on models using cellular automata, virtual particles and natural solvers. [46] and [82] present this system.
Main URL: `http://www.science.uva.nl/research/scs/SCS4.html`

SRB: SRB (Storage Resource Broker) is a grid middleware providing a uniform interface for connecting to heterogeneous data resources over a network.
Main URL: `http://www.npaci.edu/DICE/SRB/`

4.5 Distributed Agent-based Systems

Distributed Agent-based systems are currently essentially used in the bioinformatics area for data-mining and information use. In fact, they act often as P2P systems plus a data study system.

BioAgent: BioAgent is an agent-based system allowing a complete decentralization of local tasks processing. The system is divided in four layers: the Core layer providing basic distributed features, the Service Agents layer providing a set of services available in a single place, the BioAgents layer providing the set of mobile agents and the Workflow layer providing a definition language. Its main application are to provide an efficient access to biological services accessing local data repositories or local experimental techniques. More information can be found in [60].
Main URL: `http://www.bioagent.net/`

BioMAS/DECAF: DECAF is a multi-agent toolkit and each DECAF agent has been defined as a set of modules that work together to control an agent's life cycle. This toolkit has been extended in BioMAS to define a reusable information gathering system for bioinformatics. Three types of agents have been defined: Information Extraction Agents able to query public databases (using BLAST services), Task Agents responsible for the data processing and the Interface Agents providing a read-write access to a local knowledge-base. DECAF is described in [39] and [59].
Main URL: `http://www.eecis.udel.edu/~decaf/`

GeneWeaver: GeneWeaver is a multi-agent system dedicated to the domain of genome analysis and protein structure prediction. In this system, agents can be concerned with management of primary databases, with

performing sequence analysis using existing tools or with storing and presenting resulting information. The main point of this project is that it does not provide new methods for performing these tasks, but organizes existing ones in an efficient way. This system is described in [16] and [17].
Main URL: `http://www.ecs.soton.ac.uk/~mml/gw.html`

5 Conclusion

We presented in this article a review of distributed systems used in bioinformatics. An historical perspective has shown the evolution of distributed systems since the beginning of the sixties and has led to recent applications of distributed computing to bioinformatics, like the Decrypton project where the 50000 protein sequences known at the end of year 2001 were compared with each other. A taxonomy has been given with definitions to help understanding the differences between Client/Server, Peer to Peer, Clusters, Grids and Agent-based Distributed systems. The communication models have also been discussed, classical message passing interfaces, object-oriented middlewares and the more recent Webservices. We have then listed a set of bioinformatics tools, classified according the previously established taxonomy, and that take benefit of distributed computing. It was out of the scope of this survey to describe and to compare such tools but websites and references are provided for each tool and thus the interested reader will be able to access more information. We did not provide any descriptions of proprietary or non-free solutions since we privileged Open-Source or educational systems. Some such solutions exist but are most of the time association of proprietary software *and* hardware.

The main conclusion that arises is that the bioinformatics research field is a very huge and promising one. “Classical” distributed systems like Client/Server, P2P are currently the most employed, but many researches are directed towards grid computing or agent-based systems. The fact is that these kinds of approaches are extensible to other domains but that bioinformatics fits very well in their objectives of computational power (Grids) or data-mining (Agents).

References

[1] S. Agrawal. Hardware Software Server in NetSolve. Technical report, ICL, 2002.

[2] R.J. Allan and M. Ashworth. A Survey of Distributed Computing, Computational Grid, MetaComputing and Network Information Tools. Technical report, UKHEC, 2001.

[3] W. Allcock, A. Chervenak, I. Foster, L. Pearlman, V. Welch, and M. Wilde. Globus Toolkit Support for Distributed Data-Intensive Science. In *Proceedings of Computing in High Energy Physics (CHEP '01)*, 2001.

[4] M.N. Alpdemir, A. Mukherjee, N.W. Paton, P. Watson, A.A. Fernandes, and A. Gounaris. OGSA-DQP - A Service-based Distributed Query Processor for the Grid. In EPSRC, editor, *Proceedings of UK e-Science All Hands Meeting Nottingham*, 2003.

[5] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[6] S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST - A New Generation of Protein DB Search Programs. *Nucleid Acids Research*, 25(17):3389–3402, 1997.

[7] D.C. Arnold, S.S. Vahdiyar, and J.J. Dongarra. On the Convergence of Computational and Data Grids. *Parallel Processing Letters*, 11(2&3):187–202, 2001.

[8] M. Baker, R. Buyya, and D. Laforenza. Grids and Grid Technologies for Wide-Area Distributed Computing. *Software - Pratice and Experience*, (488), 2002.

[9] D.E. Bakken. Middleware Definition. *Encyclopedia of Distributed Computing*, 2003. Kluwer Academic Press.

[10] A.L. Barabasi. Parasitic Computing - Supplementary Material. Technical report, University of Notre-Dame, 2001.

[11] A.L. Barabasi, V.W. Freeh, H. Jeong, and J.B. Brockman. Parasitic Computing. *Nature*, 412, 30 August 2001.

[12] D. Barkai. An Introduction to Peer to Peer Computing. *Intel Developer UPDATE Magazine*, February 2001.

[13] J. Basney, M. Livny, and T. Tannenbaum. High Throughput Computing with Condor. *HPCU News*, 1(2), 1997.

[14] B.A. Bates, L.A. Kelley, R.M. MacCallum, and M.J. Sternberg. Enhancement of protein modeling by human intervention in applying the automatic programs 3D-JIGSAW and 3D-PSSM. *Proteins*, Suppl 5:39–46, 2001.

[15] F. Berman, A. Chien, K. Coope, J Dongarra, I. Foster, and D. Gannon. The GrADS Project - Software Support for High-Level Grid Application Development. *International Journal of High Performance Computing Applications*, 15(4), 2001.

[16] K. Bryson, M. Luck, M. Joy, and D. Jones. Applying Agents to Bioinformatics in GeneWeaver. *Cooperative Information Agents IV*, (1860):60–71, 2000. Lecture Notes in Artificial Intelligence.

[17] K. Bryson, M. Luck, M. Joy, and D. Jones. Agent Interaction for Bioinformatics Data Management. *Applied Artificial Intelligence*, 15(10):917–947, 2001.

[18] R. Buyya. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, 2002. Melbourne, Australia.

[19] U. Catalyurek, E. Stahlberg, R. Ferreira, T. Kurc, and J. Saltz. Improving Performance of Multiple Sequence Alignment Analysis in Multi-Client Environments. In *Proceedings of the First International Workshop on High Performance Computational Biology (HiCOMB)*, 2002.

[20] R. Chenna, H. Sugawara, T. Koike, R. Lopez, T.J. Gibson, and D.G. Higgins. Multiple Sequence Alignment with the Clustal Series of Programs. *Nucleic Acids Research*, 31(13):3497–3500, 2003.

[21] M. Chicurel. Bioinformatics - bringing it all together. *Nature*, (419):751–757, 2002.

[22] J. Coomer and C. Chaubal. Introduction to the Cluster Grid - Part 1. Technical report, Sun BluePrints, 2002.

[23] J. Coomer and C. Chaubal. Introduction to the Cluster Grid - Part 2. Technical report, Sun BluePrints, 2002.

[24] A.F. Coulson, J.F. Collins, and A. Lyall. Protein and Nucleic Acid Sequence Database Searching - A Suitable Case for Parallel Processing. *Computer Journal*, (39):420–424, 1987.

[25] V. Curcin, M. Ghanem, Y. Guo, M. Kohler, J. Syed, and P. Wendel. Discovery Net - Towards a Grid of Knowledge Discovery. In *Proceedings of KDD 2002*, 2002.

[26] S.O. Denn and W.J. MacMullen. The Ambiguous Bioinformatics Domain - A Conceptual Map of Information Science Applications to Molecular Biology. In *Proceedings of the 65th Annual Meeting of the American Society for Information Science and Technology*, 2002.

[27] DeveloperWorks. A visual tour of Open Grid Services Architecture - OGSA, 2003. `http://www-106.ibm.com/developerworks/grid/library/gr-visual/index.htm%l`.

[28] R.D. Dowel, R.M. Jokerst, A. Day, S.R. Eddy, and L. Stein. The Distributed Annotation System. *BMC Bioinformatics*, 2(1), 2001.

[29] DSOnline. Grid Computing - European Projects, 2003. `http://dsonline.computer.org/gc/gcprojects-european.htm`.

[30] S. Eddy. *HMMER's User Guide*. Howard Hughes Institute, Washington University School of Medicine, 2003.

[31] S.R. Eddy. Hidden Markov models and Large-Scale Genome Analysis. *Transactions of the American Crystallographic Association*, 1997.

[32] G.E. Fagg and J.J. Dongarra. PVMPI - An Integration of the PVM and MPI Systems. Technical report, University of Tenessee, 1996.

[33] C. Ferrari, C. Guerra, and G. Zanotti. A Grid-Aware Approach to Protein Structure Comparison. *Journal of Parallel and Distributed Computing*, 63:728–737, 2003.

[34] M.J. Flynn. Some Computer Organizations and their Effectiveness. *IEEE Transactions on Computers*, C-21, 1972.

[35] I. Foster. Designing and Building Parallel Programs, 1994. `http://www-unix.mcs.anl.gov/dbpp/`.

[36] I. Foster and C. Kesselman. Globus - A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2), 2001.

[37] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM - Parallel Virtual Machine - A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press , 1994.

[38] C. Gibas and P. Jambeck. *Developing Bioinformatics Computer Skills*. OReilly, 2001.

[39] J.R. Graham, K.S. Decker, and M. Mersic. DECAF - A Flexible Multi Agent System Architecture. *Autonomous Agents and Multi-Agent Systems*, 2003. to appear but available online.

[40] J.R. Graham, V. Windley, D. McHugh, F. McGeary, D. Cleaver, and K. Decker. A Programming and Execution Environment for Distributed Multi Agent Systems. In *Proceedings of the 4th International Conference on Autonomous Agents*, 2000. Workshop on Agents in Industry.

[41] J.D. Grant, R.L. Dunbrack, F.J. Manion, and M.F. Ochs. BeoBLAST - Distributed BLAST and PSI-BLAST on a Beowulf Cluster. *Bioinformatics*, 18(5):765–766, 2002.

[42] GridComputing.org. Grid Computing Info Centre - GRID InfoWare, 2003. `http://www.gridcomputing.com/`.

[43] N. Guex and M.C. Peitsch. SWISS-MODEL and the Swiss-PdbViewer - an Environment for Comparative Protein Modeling. *Electrophoresis*, 18(15):2714–2723, 1997.

[44] L. Hunter. *Molecular Biology for Computer Scientists*, chapter 1. AAAI Press Book, MIT Press edition, 2002.

[45] Integra. A Definition of Web Services. Technical report, Integra - Genuity, 2002.

[46] K.A. Iskra, R.G. Bellema, G.D. vanAlbada, J. Santoso, P.M. Sloot, and H.E. Bal. The Polder Computing Environment - a system for interactive distributed simulation. *Concurrency and Computation: Practice and Experience*, 14, 2002. Special Issue on Grid Computing Environments.

[47] P. Kacsuk and F. Vajda. Network-Based Distributed Computing - Metacomputing. Technical report, ERCIM, 1999.

[48] M. Karaul. *Metacomputing and Resource Allocation on the World Wide Web*. PhD thesis, New York University, 1998. Department of Computer Science.

[49] H. Kargupta. Distributed Data Mining Bibliography, 2003. `http://www.csee.umbc.edu/~hillol/DDMBIB/`.

[50] L. Kelley, R. MacCallum, and M.J. Sternberg. Recognition of remote protein homologies using three-dimensional information to generate a position specific scoring matrix in the program 3D-PSSM. In ACM, editor, *RECOM99 - Proceedings of the Third Annual Conference on Computational Molecular Biology*, 1999. New-York.

[51] L. Kelley, R. MacCallum, and M.J. Sternberg. Enhanced genome annotation using structural profiles in the program 3D-PSSM. *Journal of Molecular Biology*, 299(2):499–520, 2000.

[52] Kent-University. Internet Parallel Computing Archive, 2003. `http://wotug.kent.ac.uk/parallel/`.

[53] H.S. Kim, H.J. Kim, and D.S. Han. Hyper-BLAST - A Parallelized BLAST for Speedup of Similarity Search. Technical report, Information and Communications University, Korea, 2003.

[54] L. Kleinrock. Information Flow in Large Communication Networks. *RLE Quarterly Progress Report*, July 1961.

[55] L. Kleinrock. *Communication Nets - Stochastic Message Flow and Delay*. McGraw-Hill, 1964. later re-issued by Dover Books.

[56] L. Kuo-Bin. ClustalW-MPI - ClustalW Analysis Using Distributed and Parallel Computing. *Bioinformatics*, 19(12):1585–1586, 2003.

[57] LCIC. Linux Clustering Information Center, 2003. `http://www.lcic.org/`.

[58] S. Markel and D. Leon. *BLAST*, chapter 7. OReilly, Sequence Analysis in a Nutshell edition, 2003.

[59] F. McGeary and K. Decker. DECAF Programming - Agents for Undergraduates. In *Proceedings of the 5th International Conference on Autonomous Agents*, 2001. Workshop on Infrastructure for Scalable Multi-Agent Systems.

[60] E. Merelli, L. Culmone, and L. Mariani. BioAgent - A Mobile Agent System for Bioscientists. In *Proceedings of NETTAB02*, 2002. Agents in Bioinformatics.

[61] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, and B. Richard. Peer to Peer Computing. Technical report, HP Laboratories Palo Alto, 2002.

[62] MPI-Forum. *MPI Standard 2.0*. MPI, 2003.

[63] Sape Mullender. *Distributed Systems*. Addison-Wesley, 1993. University of Twente.

[64] A. Natrajan, M. Humphrey, and A. Grimshaw. Capacity and Capability Computing in Legion. In *Proceedings of the 2001 International Conference on Computational Science*, 2001.

[65] C. Notredame, D.G. Higgins, and J. Heringa. T-Coffee - A novel method for multiple sequence alignments. *Journal of Molecular Biology*, 302:205–217, 2000.

[66] C. Notredame, L. Holme, and D.G. Higgins. COFFEE - A New Objective Function For Multiple Sequence Alignmnent. *Bioinformatic*, 14(5):407–422, 1998.

[67] G. Otero. Biopackage Repository and BioBrew Linux Projects. In *Proceedings of the OReilly Bioinformatics Technology Conference*, 2003.

[68] P2P4B2B. P2P4B2B - Non-Commercial Peer-to-Peer Efforts, 2003. `http://www.stratvantage.com/directories/p2pworkgroups.htm`.

[69] Y.P. Paindaveine. HealthGRID - Old Wine in New Bottle. In Healthgrid, editor, *Proceedings of the First European HealthGrid Conference*, pages 20–27, 2003. Lyon, France.

[70] M.C. Peitsch. ProMod and Swiss-Model - Internet-based Tools for Automated Comparative Protein Modelling. *Biochemical Society Transactions*, 24(1):274–279, 1996.

[71] Globus Project. Globus Related Projects, 2003. `http://www.globus.org/about/related.html`.

[72] A. Rowe, D. Kalaitzopoulos, M. Osmond, M. Ghanem, and Y. Guo. The Discovery Net System for High Throughput Bioinformatics. *Bioinformatics*, 19:i225–i231, 2003.

[73] T. Schwede, J. Kopp, N. Guex, and M.C. Peitsch. SWISS-MODEL - An automated protein homology-modeling server. *Nucleic Acids Research*, pages 3381–3385, 2003.

[74] L.D. Stein. Integrating Biological Databases. *Nature Reviews Genetics*, (4):337–345, 2003.

[75] T. Sterling, D. Savarese, D.J. Becker, J.E. Dorband, U.A. Ranawake, and C.V. Packer. BEOWULF - A Parallel Workstation for Scientific Computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages I:11–14, 1995. Oconomowoc, WI.

[76] P.M. Stothard. COMBOSA3D - combining sequence alignments with three-dimensional structure. *Bioinformatics*, 17(2):198–199, 2001.

[77] T. Tannenbaum, D. Wright, K. Miller, and M. Livny. *Condor - A Distributed Job Scheduler*, chapter 15. The MIT Press, 2002. Beowulf Cluster Computing with Linux.

[78] D. Thain, T. Tannenbaum, and M. Livny. *Condor and the Grid*, chapter 11. John Wiley, Grid Computing - Making The Global Infrastructure a Reality edition, 2003.

[79] J.D. Thompson, D.G. Higgins, and T.J. Gibson. CLUSTAL W - improving the sensitivity of progressive multiple sequence alignment through sequence weighting position specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[80] O. Trelles. On the Parallelisation of Bioinformatics Applications. *Briefings in Bioinformatics*, 2(2):181–194, 2001.

[81] Utyx. Bioinformatics Resources, 2003. `http://utyx.com/bioinformatics/bioinformatics-tools.html`.

[82] A.W. vanHalderen, B.J. Overeinder, P.M. Sloot, R. vanDantzig, D.H. Epema, and M. Livny. Hierarchical Resource Management in the Polder Metacomputing Initiative. *Parallel Computing*, 24(12/13):1807–1825, 1998.

[83] V. Veeramachaneni, P. Berman, and W. Miller. Aligning Two Fragmented Sequences. In *Proceedings of the First International Workshop on High Performance Computational Biology (HiCOMB)*, 2002.

[84] A. Waugh, G.A. Williams, L. Wei, and R.B. Altman. Using Metacomputing tools to facilitate Large-Scale Analyses of Biological Databases. In *Proceedings of the 2001 Pacific Symposium on Biocomputing*, 2001.

[85] M.D. Wilkinson and M. Links. BioMOBY - An Open Source Biological Web-Service Proposal. *Briefings in Bioinformatics*, 3(4):331–341, 2002.

[86] T.K. Yap, O. Frieder, and R.L. Martino. Parallel Computation in Biological Sequence Analysis. *IEEE Transactions on Parallel and Distributed Systems*, 9(3):283–294, March 1998.

[87] A. YarKhan and J.J. Dongarra. Biological Sequence Alignment on the Computational Grid using the GrADS Framework. *Bioinformatics Computing*, 2003. submitted but available online.

[88] Z. Zhang and W.I. Wood. A profile hidden Markov model for signal peptides generated by HMMER. *Bioinformatics*, (19):307–308, 2003.