IV

Computers and Molecular Biology: Issues and Constraints

22

Network Computing

Restructuring How Scientists Use Computers and What We Get Out of Them

Brian Fristensky

1. Introduction

This article describes the network computer (NC), an alternative to the stand-alone PC. By shifting data storage and most processing to the server, any user can do any task from any NC. NCs provide a reliable, consistent interface for all users, and make it easy to provide group access to resources such as laboratory databases. NCs are intrinsically insulated from obsolescence, and offer economies of scale through shared hardware, software, and administration. In the future, stand-alone PC-driven laboratory equipment could be superseded by Java-based NC robots, controlled and monitored across the network.

1.1. The Problem: The Fat Client

The standalone PC is referred to as a "fat client" in that it must have all the hardware and software necessary for every task. The rapid evolution of processing capabilities drives software development to utilize those capabilities. Consequently, most PCs, and their accompanying hardware and software, must be replaced every 3–5 yr.

System administration is the largest hidden cost of PCs. Because each PC is typically configured by different users for different purposes, each PC is a special case. PCs are becoming increasingly complex, particularly because of their increasing integration in networks. Keeping everything working on all machines in a department, and upgrading smoothly, is becoming an increasingly unrealistic goal even for professional PC/LAN administrators.

From: *Methods in Molecular Biology, vol. 132: Bioinformatics Methods and Protocols*
Edited by: S. Misener and S. Krawetz Jr. © Humana Press Inc., Totowa, NJ

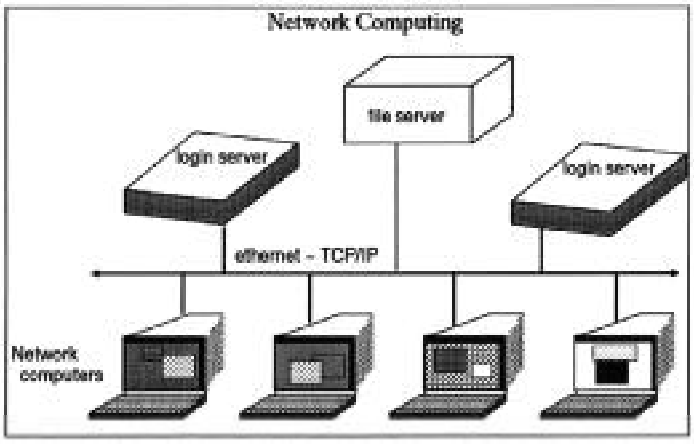

Fig. 1. Network Computing: data and software reside on a central file server, mounted remotely to all servers. Any user can log into any server from any NC, and do any task.

Everyone is familiar with the inconvenience of having to wait to use a program because the only machine on which it is installed is occupied. Again, each PC is a special case. In another example, to put a sequence alignment into a paper, you might first do the alignment on a lab computer that has sequence software, then upload the alignment to a LAN directory, go to your office, download the file, and import it into a word processor. Then you decide you want to present the alignment in a slightly different way. You must then repeat all of these steps.

Fragmentation of data and software among PCs wastes time, causes frustration, and makes it difficult to remember where a specific file is, or where the most recent version of the file is. Also, since shared PCs are seldom backed up, and PC-based operating systems almost entirely lack security features, other users can, through mishap or malice, destroy valuable data.

1.2. The Solution: The Thin Client

Early computers were centrally administered, serving numerous users via remote text-only terminals. Today, systems like UNIX can provide dozens or hundreds of users with a point-and-click desktop. Several variations of network computing are being developed. The *X11*, or *X-Windows* system, pioneered at MIT in the mid 1980s and now developed by The Open Group ***(1)***, is the most stable and widely used protocol. As illustrated in **Fig. 1**, a user connects to a login server, which sends *X11* commands to the NC. The *X11* commands specify the content and location of each window as it appears on the

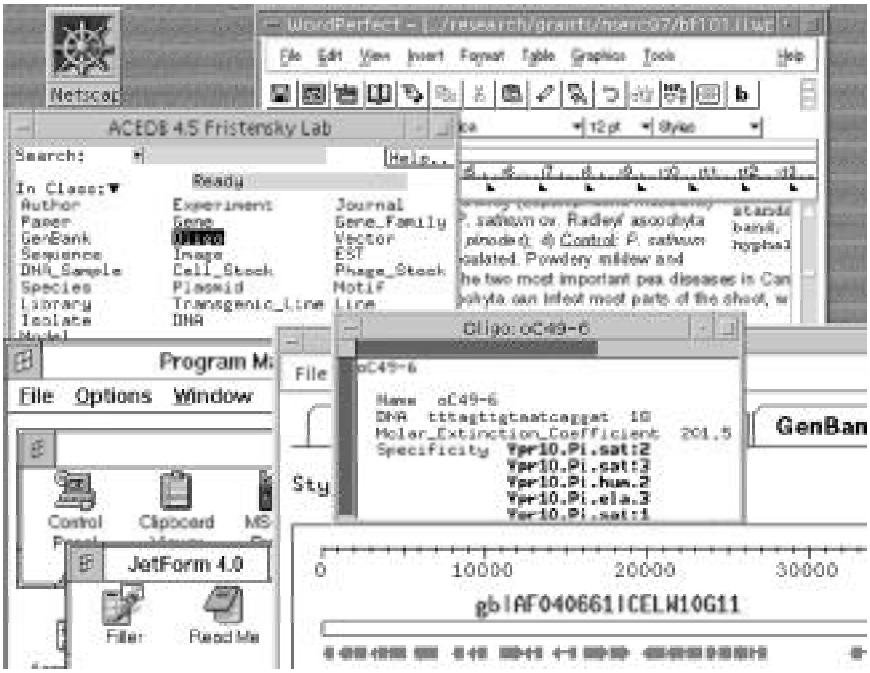

Fig. 2. Screen shot of a typical UNIX session, illustrating the wide range of desktop and scientific software available.

screen. Typically, most of the work done by a program involves redrawing of windows as you scroll, cut and paste, or open or close windows. *X11* offloads these tasks to the NC, reducing load on the server. Because all other tasks are done at the server, the NC is referred to as a "thin client." *X*-terminals are diskless, and do not themselves run programs.

Where multiple servers are available, all software and files are centralized on a file server, whose file systems are remotely mounted across the network to all login servers. Regardless of which server you log into, you will have the same desktop and the same home directory. Put simply, the central concept of network computing is that any user can do any task at any NC.

Figure 2 illustrates the capabilities available to users on our Sun/UNIX system at the University of Manitoba. At top, a grant proposal is being written using *WordPerfect* for UNIX ***(2)***. Background information on oligonucleotides is obtained from our *AceDB* ***(3)***-driven lab database. Information on genes to be used in the project is accessed from *GenBank* using the *NEntrez* network client ***(4)***, in the background at bottom right. At lower left, the *MS-Windows Program Manager* runs via Sun's *Wabi* ***(5)***, making it possible to fill out the

spread-sheet part of the grant using *JetForm*. Finally, a Web-browser can be opened at the click of an icon when information is needed from the Web.

On such a system software is usually installed by the system administrators, but there is nothing to stop the user from installing or writing their own software. To illustrate, all development and management of our university-wide sequence analysis facility, *BIRCH*, was done on a regular user account without root privileges ***(6)***.

The main point I wish to make is that NCs allow you to do all tasks commonly done on PCs. I have used UNIX exclusively for all my computing since 1990 and my lab does not even have a PC. Our first *X*-terminal, purchased in 1993, still lets us run the latest software on the server. Upgrades to the operating system and applications software, and even an upgrade to a 64-bit server, were entirely transparent, requiring no reconfiguration of our terminals.

2. Hardware, Software, and Know-How

2.1. Hardware

You will need a UNIX-based workstation, such as a Sun/Solaris, IBM/AIX, or Intel/Linux. Avoid buying a 32-bit workstation. Most versions of UNIX are either completely adapted to 64-bit architecture, or will be in the next few years. If your institution provides centralized UNIX servers, you may not even need to buy a server. You will also need an *X*-terminal, or a PC running *X-Windows* emulation software (*see* **Subheading 3.5.**). The most important single factor to ensuring smooth performance is RAM. The more memory the server has, the less frequently it will need to swap programs and data to disk. Additional memory can often be purchased cheaply from third party vendors who routinely advertise on Internet newsgroups. Additionally, most versions of UNIX can support two or more CPUs.

2.2. Software

Because almost everything related to the Internet was invented under UNIX, typical configurations come standard with full Internet services, including telnet, ftp, e-mail programs, and a Web browser such as *Netscape*. UNIX systems typically also include the *X11 Display Manager* (xdm) that serves *X*-terminals. If the computer center at your institution already operates networked UNIX servers, they may be willing (perhaps for a small fee) to administer your server as a clone of their other machines. This has the advantage that if your server needs to be down, you can log into any other server. To cite an example, in 1996 I moved my home directory from my personal workstation to our campus system. During a 3-mo interim prior to purchasing a new server for several lab groups in our department, I used the publicly-available dual-CPU

Sun Sparc20 servers, which typically had 30–60 simultaneous users. With few exceptions, I seldom noticed any degradation in performance.

2.3. Know-How

If your computer center administers your server as described above, you reap the greatest benefit of the NC model: Let the experts do the work. This is particularly important with respect to security. Most professionally administered systems have someone who keeps up on the latest security holes, and installs the appropriate patches in a timely fashion. Alternatively, someone in your lab or department will need a working knowledge of UNIX, some previous experience in programming (preferably C or Java) and an ability to write html. All other users will benefit from an introduction to UNIX ***(7)***, but by and large, the skills and concepts learned on Windows-based systems will easily transfer to UNIX desktops.

3. Practical Matters

Network computing is still evolving, and there will probably be many different implementations of the thin client model. As the UNIX/*X-windows* approach to network computing has been fairly stable for almost a decade, and is likely to continue to be one of the major network computing models, I will discuss some of the things you need to know to do all your computing on an *X-windows* platform.

3.1. Make a Complete Switch

When you move to the NC platform, do it all the way. If you divide your computing and datafiles between NCs and PCs, you will actually make things more complicated. Also, you will learn your way around server-based computing faster if you use it for everything.

3.2. The Third Party Software Problem

The chief impediment to the growth of network computing is the perceived lack of third party software for server-based platforms. At one level it is true that the majority of desktop software vendors write specifically for PCs. Whereas use of Java may result in platform independence in the future, at present it is often more difficult to find applications for server-based systems such as UNIX, VMS, or AS400. On the other hand it is surprising how much software is available for these platforms. In many cases, server-based versions of programs such as *WordPerfect* ***(2)*** or *Adobe PhotoShop* ***(8)*** are available. In other cases, comparable applications exist that are specifically targeted to client/server platforms.

3.3. Using Windows Applications under UNIX

Generally, it is best to use native UNIX applications wherever possible. When no UNIX version is available, Windows applications can be displayed from a *WindowsNT* server to an *X-Windows* desktop via NCD's *Wincentre* ***(9)***. This approach has the advantages that applications run on native *Intel* architecture, and some tasks are offloaded to the NT server. The disadvantages are that a dedicated NT server is needed, and the NT server must be configured to work with the UNIX file server. On Sun systems, *Wabi* ***(5)*** can run *Windows 3.1* in software emulation.

3.4. X11 Programs from Remote Servers

Sometimes it is desirable to run an *X-Windows* application on a server other than the one you are logged into currently. For example, an application might be licensed to run on only a few servers. This problem can be solved easily by logging into the licensed server and setting the *X11* display to your terminal or workstation.

If a user named raven is logged into marigold.uofm.ca, but wants to run *SAS*, which is installed on petunia.uofm.ca. Log into petunia using telnet:

```
{marigold:/home/raven}telnet petunia
Trying 130.122.36.48 . . .
Connected to petunia.uofm.ca.

UNIX (r) System V Release 4.0 (petunia) (pts/18)

login: raven

Password:

{petunia:/home/raven}
```

Next, set the environment variable **DISPLAY** to the name of your terminal or workstation.

```
{petunia:/home/raven}setenv DISPLAY ncd12.uofm.ca:0.0
```

This command will cause all subsequent *X11* programs launched in this shell to display on the screen of the *X*-terminal called ncd12.

Finally, launch *SAS*:

```
{petunia:/home/raven}sas &
```

Note the ampersand (&) after the command. Putting an ampersand after any UNIX command causes it to run in the background. This does two things. First, it frees up the command window to let you do other things as the application is

running. Secondly, if you wanted to, you could log out from petunia without causing *SAS* to terminate.

For applications that frequently need to be run from remote servers, those steps could be automated by putting these commands into a shell script, which could be launched from the workspace menu.

3.5. Turning PCs into NCs

Many labs have made large investments in PC hardware. As well, most departments are full of older PCs that can no longer run the latest software (e.g., 486s that can not run *Windows'95*). Finally, it is usually not feasible to purchase a large number of NCs at one time. For all these reasons, a number of *MS Windows* and/or Mac-based programs allow a PC to act as an *X*-terminal.

Some of the pluses of *X11* emulation software include:

- Inexpensive.
- The software has been around long enough to be reasonably reliable and easy to use and install.
- Even 486s will often perform almost as quickly as an *X*-terminal. Also, the task of drawing windows on a screen always remains about the same, so once an old 486 works, it should always work.
- Typically includes network transport protocols such as *SLIP* and *PPP*, making it possible to run an *X-Windows* session over a fast modem from home.

There are some very good reasons why *X* emulation software is an interim fix, rather than a permanent solution:

- You still have to keep *MS Windows* working. Any time you upgrade *Windows*, or install software, or alter the PC networking software, on the PC or on the LAN server, you risk affecting the *X* emulation program.
- *X* emulators are not perfect. Because the PC and Windows add a layer of complexity, there is never a guarantee that the *X11* software will do everything that an *X*-terminal is supposed to do.
- If you are using a PC with an *X* emulator, there is a temptation to do some things on the *X* desktop, and some things on the PC. Thus, you fragment your files and necessitate uploading and downloading of information, and have less incentive to really learn how to use the *X* desktop. Things are simpler if you stick to one system.

The strategy should therefore be to upgrade existing PCs to *X* emulation in the short term, and in the long term, buy new *X*-terminals, rather than new PCs. PC *X* emulation programs include Hummingbird's *eXceed* ***(10)***, White Pine Software's *eXodous* ***(11)***, and NCD's *PC-Xware* ***(12)***. Many of these vendors offer free downloads of a trial copy of the program.

4. The Future: How Network Computers Will Change the Way We Work

The purpose of this section is to show the kinds of developments that become possible once network computing becomes commonplace. I will stick to things that are already in development or are already being implemented, with the exception of **Subheading 4.4.**, which is a synthesis of these developments.

4.1. Work the Same Way Anywhere

Today, each researcher and student is wedded to a single PC, or worse, to several PCs specialized for different tasks. We have to copy files to diskette to work at home, or carry a laptop when we travel.

Network computers are even now beginning to appear in hotels, airports, and university computer centers and libraries. In the near future, you will be able to use a network computer at home to work on papers, run resource-intensive computations, or even check on an experiment in progress (*see* **Subheading 4.4.**). Because NCs are cheap, they will be everywhere. During airport layovers, in your hotel room, or even on sabbatical half way around the world, you can do anything you can do in your lab or office, *in exactly the same way*. You will no longer be limited to the files, software, and computing power that you can carry with you.

To simplify the use of NCs by the traveling public, companies such as Network Computer Inc. ***(13)*** have developed smart cards that carry the information required to find your home server across the Internet and to connect you to your account.

4.2. Electronic Seminars, Presentations, Teaching

Classroom and symposium presentations are already being transformed by Web-based presentations. However, Web browsers are still limited compared to NCs. For example, I present all lectures for my cytogenetics course using an *X* terminal from which the screen output is sent to a 1024×768 projector. Whereas most of the lecture is on the course Web site ***(14)***, I often use graphics applications for simple demos. The server-based nature of the terminal guarantees that the demo will work in class exactly the way it worked in my office.

As conference centers adapt to network computing, symposium speakers will have all figures and programs from their home server available at the podium. Even unanticipated data or figures could be presented in response to questions.

4.3. Java: Write Once, Run Anywhere

Java ***(15)***, the new programming language from Sun Microsystems, was specifically designed to run on any computer platform. This is how it works: Java

programs are run within a shell called the Java virtual machine. Because the Java virtual machine has been ported to virtually all computer platforms (e.g., UNIX, Windows, OpenVMS, IBM mainframes, Macintosh, and so on) all Java programs should run on all platforms. All you need is the Java virtual machine. Software written in Java is therefore platform independent. Software developers need only write and maintain one Java version of the software, rather than many versions for many platforms.

The Java *Molecular Biology Workbench* ***(16)*** is an example of a suite of Java programs. In this case, the programs are run as applets, which are downloaded from the remote server at runtime. Java programs can also be downloaded and run as standalone applications on one's local server, workstation or PC.

One of the main advantages of Java is that it is modular. Applications written in traditional languages are single entities taking up many megabytes of memory. Java applications are packages of small objects, each carrying out a single function. When a NC runs a Java application, only the objects needed at a given time are downloaded from the server. Consequently, Java-based NCs don't need large amounts of memory and processing power, which provides some protection from obsolescence.

4.4. Clean integration of Computing Platform, Network, Lab Notebook, and Lab Equipment

Java was conceived originally as a hardware-independent language to allow electronic devices to be programmed, rather than having their capabilities hardwired. That aspect of Java may lead to networked laboratory equipment that is far more versatile and upgradeable than the machines currently in our labs. Today, laboratory devices such as fluorescent imagers, DNA sequencers, and even plant growth chambers each require a PC to operate them. Because software for analysis and data acquisition both usually reside on the dedicated PC, you cannot analyze your data if the device is in use by someone else. In the future, many devices will be networked Java NCs. One obvious result is that it will no longer be necessary to purchase and configure a PC for each device. Dedicated hardware such as monitors, printers, or LCD displays will also be unnecessary, making Java devices smaller and cheaper. A Java chip ***(17)*** resident in each device will perform all operations. Each device can be controlled and monitored from any NC. At the completion of the experiment, data can be uploaded directly to the user's directory for analysis.

More exciting is the concept of the virtual robot. In principle, virtual robots could be created on the desktop by linking Java devices together in an equipment control program. In this example, the equipment control program represents each real device by a screen icon. Note that the pipetting robot, which is

called by each device, does not need to be represented in the control program. (This is analogous to class inheritance in object-oriented languages like Java). In **Fig. 3**, a DNA sequence analysis program is used to design primers, which are sent to the equipment control program. A DNA sequencing robot is created by linking a DNA synthesizer, thermal cycler, and DNA sequencer in succession. Results are relayed from the sequencer via the control program, into a DNA sequence analysis program. The electronic lab notebook is also a Java device, and can be used to tell the program which DNA samples to use for sequencing, or where to store samples generated by the thermal cycler.

Whereas multicomponent robots could be created using existing PCs and operating systems, each virtual device is a special case requiring extensive programming to implement. A software-based control program would make it possible to tailor virtual robots for each task. For example, if you wanted to quantitate your PCR product before loading onto the sequencer, a fluorescence detector might be linked between the thermal cycler and the sequencer, and only samples that successfully amplified would be sequenced.

5. Looking Back at Today from Tomorrow's Perspective

When we look back at the PC era in five or ten years, we will be amazed at the things we took for granted. Most ridiculous will be the notion that every user was a system administrator. In the NC era, servers will be professionally administered. By their nature, there is essentially nothing to administer on an NC. All the user will do is use them.

The economics of computing will be changed, breaking the obsolescence cycle. Users will still spend money on computing, but rather than personally buying RAM, disk drives, coprocessors, and software, we will pay our service providers a monthly fee to provide these things for us. Professionally maintained centralized resources will be more stable and reliable.

Today, most users are locked into the *MS Windows* operating system. In the NC era, even desktops and operating systems will be subject to competition, resulting in more choices and competitive pricing. This article has focused on UNIX, largely because UNIX is particularly well suited to network computing. However, the open nature of the NC model means that other systems such as OpenVMS, AS/400, and possibly WindowsNT (if scalability and security issues can be resolved) could play a role in providing NC services. One possible outcome is that NC-service providers will use a mixture of different servers and operating systems to deliver a complete range of applications to a single desktop, in a fashion that is transparent to the user. Regardless of the changes at the server end, the user's investment in NC hardware will be protected, because the thinner the client, the less there is that can become obsolete.

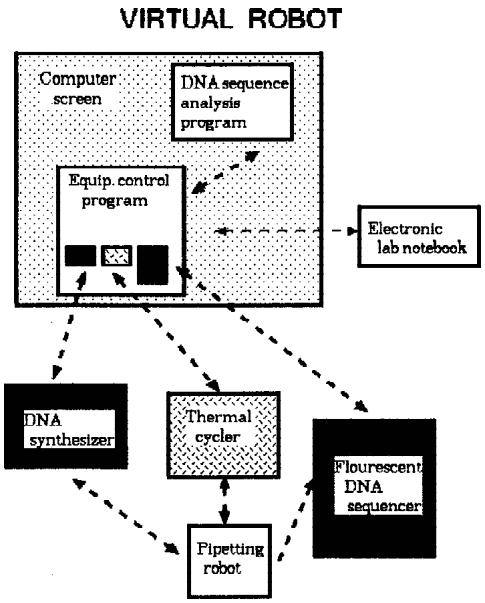

Fig. 3. The virtual robot: Networked devices programmed in Java can be linked together via software to create virtual robots tailored for a specific task, and controlled from the NC.

Our anticipatory retrospective on the PC era was best summarized in 1977 by Ken Olsen, Chairman and founder of DEC: "There is no reason anyone would want a computer in their home" (or lab, B.F.).

For more information on Network Computing, and additional Internet links, *see* `http://home.cc.umanitoba.ca/~psgendb/nc`.

References

1. The Open Group (1998) *X Window System*, `http://www.camb.opengroup.org/`.
2. Corel *WordPerfect* for UNIX, `http://www.corel.com/products/cwp7unix.htm`.
3. Durbin, R. and Thierry-Meig, J. (1991) A C. elegans Database. Code and data available from anonymous FTP servers at `lirmm.lirmm.fr`, `cele.mrc-lmb.cam.ac.uk` and `ncbi.nlm.nih.gov`.
4. Natl. Center for Biotech. Info. Nentrez, `http://www3.ncbi.nlm.nih.gov/Entrez/Network/nentrez/overview.html`.
5. Sun Microsystems, *Wabi*, `http://www.sun.com/solaris/wabi/`.
6. Fristensky, B. (1998) Building a multiuser sequence analysis facility using freeware, this volume, pp. 00–00.

Ed: x-ref

7. Sobell, Mark G. (1995) A Practical Guide to the UNIX System. Addison-Wesley Publishers, Reading, MA.
8. Adobe Corp. *PhotoShop*. http://www.adobe.com.
9. Network Computing Devices Wincenter. http://www.ncd.com/pwin/pwin.html.
10. Hummingbird Ltd., http://www.hummingbird.com/products/exceed/.
11. White Pine Software, http://www.wpine.com/exodus/.
12. Network Computing Devices, http://www.ncd.com/ppcx/ppcx.html.
13. Network Computer Inc. http://www.nc.com/prodcard.html.
14. Fristensky, B. *Introductory Cytogenetics*, Univ. of Manitoba. http://www.umanitoba.ca/afs/plant_science/COURSES/CYTO/.
15. Sun Microsystems. *The Java Platform.* http://java.sun.com/aboutJava/.
16. Toldo, L. (1997) *JaMBW 1.1: Java-based molecular biologists' workbench. Comput. Appl. Biosci.* **13,** 475–476. http://www.embl-heidelberg.de/~toldo/JaMBW.html.
17. Sun Microsystems. *Java Computing.* http://www.sun.com/java/.