Jerry ✓
505

M.I.T. Laboratory for Computer Science
Computer Systems Research Division

January 3, 1980
Request for Comments No. 181

Choice of Language for the Distributed Data Storage System Project

by David P. Reed

Since the Distributed Data Storage System project (DDSS in this memo), is to be an implementation project, it is necessary to choose a language for the implementation. Making this choice early has several advantages.

Documentation of the design of the system can be done in terms of the ultimate implementation language. Invention of notations for formats, protocols, interfaces, etc. is thus avoided, and it is more likely that separately designed parts will work together.

Early coding of parts of the system can be carried out to experiment with various design choices. For example, the interface to the network and the interface to the disk storage media will be important to get right.

The implementation language will most likely be the first "client interface" to the data storage system. Rather than talking about some "language in the sky," the designers will have a concrete user interface in mind.

Inevitably, the choice of language is coupled with the choice of the hardware supporting the system. A desirable goal is to avoid putting a language development effort in series with the construction of the initial prototype. The following discussion will explore possible choices of hardware and language, arguing that the most reasonable of the plausible choices is to use the MESA language on the Alto hardware for the first prototype.

Prototype

Nodes in the DDSS play one of three roles—user, repository, and authenticator. *User* nodes are the places where the data objects stored in the DDSS are used in computations. A typical user node is a personal computer. *Repository* nodes are places where large numbers of data objects can be stored reliably. A typical repository node is a computer with a large amount of disk storage, a means for backing up the state of the storage after a crash, and specialized software to use the hardware efficiently. *Authenticator* nodes are trusted intermediaries, used to establish the identity of two communicating nodes to each other. The mechanism for authentication will be based on encryption, so the authenticator nodes will be the generators of encryption keys.

Choice of Language for the DDSS Project

The prototype we initially design and build will be a simplified framework for the eventual DDSS. The prototype can then serve as the basis for further enhancements to improve the characteristics of the DDSS, such as performance, reliability, and ability to support particular language features.

It is not expected that the prototype will provide a service—instead it will be a vehicle for experimentation. Thus the language and hardware used for its construction need not be *the* language and hardware systems of the LCS Distributed Systems Project. We also expect that at some point it will be a good idea to completely rebuild the DDSS—at this point a change of language and hardware may be appropriate.

Possible Hardware Bases

There are four reasonable choices of hardware available to us. These are the Altos, the ν's, some PDP-11's, and the XX TOPS-20. The primary differences between these lie in their availability for use in the project, their suitability for running the resulting system, and the languages and operating system support available on each. Exportability of the ideas is also affected by the hardware base.

The Altos are available now for use in the distributed system project. In addition, we have one Alto equipped with a Trident disk drive interface that can eventually be used as a server. The "personal computer" nature of the Altos is a quite close match to the environment in which the ideas explored in the DDSS ought to be used. A drawback, however, is that the Altos are rather small and weak.

The ν's are not yet available, except for a fragile prototype. They do, however, have some potential advantages. In terms of memory size and performance, they outperform the Altos. Any software we develop on them may be highly exportable, assuming Zenith chooses to market the ν hardware.

A number of PDP-11's could be made available. We have a PDP-11 with a lot of storage on some old Diva drives. Like the Altos, however, the PDP-11's are rather small and weak. It is not really possible to treat the 11's like personal computers, so making them play the user node role would lack verisimilitude.

The XX Tops-20 is available to us, although it is heavily loaded. Some of the issues, particularly reliability and performance, would be difficult to explore in that environment because the machine is being shared with others, so direct access to devices is not feasible.

Since the prototype system will consist of three types of nodes, it certainly would be possible to implement each type of node on a different type of hardware. Such a mixed strategy would have the drawback that the design of low-level software, such as device managers, would have to be done separately for each type of hardware. Another, more serious, drawback is that the networks that are available to each type of hardware are different in characteristics and protocols. Thus, communicating from one to the other may be much more difficult in a mixed strategy.

Languages Available

Possible languages for implementing the DDSS are Clu, C, Mesa, and BCPL. They differ in their current and future availability on the various machines, and in their suitability for the kinds of software to be constructed. Included in the suitability question is the question of support tools, such as debuggers, subroutine libraries, etc.

Clu is currently being transported to the Z-8000, 68000, and Alto. Consequently, at some point in the future, Clu will be available on the ν, the TOPS-20, and the Altos. In this sense it is an attractive choice for the future. Currently, however, the only machine that supports Clu adequately is the TOPS-20. A reasonable estimate of the time until the Clu implementations on the Altos is suitable for a major implementation project is six months.

C is also widely available; implementations exist for the TOPS-20 and PDP-11, and they are planned for the Z-8000 and 68000 processors that are used in the ν. C is a lower-level language than Clu, which may make it a more difficult language to design a large system in. C is also not a good model for the client language, since it is not object based in any sense.

BCPL is available on PDP-11's, the TOPS-20, and the Altos. It would not be hard to implement a BCPL for the ν, although it would still be a number of months before a suitable support system existed. It has the same disadvantages as C, with an additional one due to lack of familiarity with it locally.

Mesa exists only for the Altos (also for Dorados and D-0's, should we ever get those from Xerox). It is not likely that a Mesa will exist for any of the other processors being considered. Mesa does have a quite good programming environment of debuggers, "configuration management" (keeping track of interfaces in a large program), memory management, etc. Choosing Mesa limits the future evolutionary path to two directions—either reimplement the system in another language, or wait for new hardware from Xerox. Reimplementing is not necessarily an onerous burden, since it can be done in parts. Each of the three roles will be played by distinct computers, and assuming that the network protocols define a hard interface, the software that implements each role can be reimplemented separately. Thus reimplementation can be done in a staged manner.

Plan of Action

The choice we have made is to use the Mesa language on the Alto hardware for our prototype implementation. At first, the prototype will be used only for experimentation, and will not provide a service. Once we are satisfied with the basic approach, we will then attempt to reimplement the system "right." Probably the best long-term direction will be to reimplement the software that plays the *user* role in Extended Clu as the first step in reimplementation. Presumably, at that point, a shift to the ν hardware will be possible also.

Because of the experimental nature of the DDSS, we have no immediate plans to make the server we are constructing an LCS-wide resource. In particular, we do not intend for it to supplant immediately the services of a more traditional remote file store, such as the IFS.