Global Computation

Luca Cardelli

Digital Equipment Corporation, Systems Research Center
<http://www.research.digital.com/SRC/personal/Luca_Cardelli/home.html>

Abstract
Computation over planet-wide structures is hindered by administrative, architectural, and physical constraints. These problems are surmountable, but must be addressed by developing new models of programming and of computation.

Global Information Structures

The Internet communication protocols synthesize global information structures out of large collections of processors and networks. There are many kinds of such global structures. For example, the FTP protocol realizes a global file system, the Telnet protocol realizes a global multiprocessor, and the HTTP protocol realizes a global hypertext domain (the World-Wide Web) [3]. We would naturally like to have global information structures that are programmable. That is, we would like to use some global structures as *global computers*. A number of novel research issues arise if we want to program global computers.

Global Computers

The main characteristic of a global computer is its geographical distribution. Because of the slowness of light, a planet-wide computer cannot be usefully regarded as a localized computer. For example, a procedure call to the antipodes takes at least a noticeable 0.13 seconds; therefore, a widely-dispersed program is easily distinguishable from a localized one. This physical limit has drastic consequences for programming. It implies that we need control over the locality, mobility, and distribution of computation. Latency and bandwidth, not CPU speed and memory size, become the limiting factors and must be directly addressed.

The Web is evolving rapidly towards programmability. A single global structure, however, will not satisfy all needs for global computation. Every large company will soon have its own internal global computer connecting its geographically distributed resources reliably and securely. The Web itself may be split into high-quality and low-quality services. Different global computers may be based on different instruction sets (virtual machines), and may be separated by administrative boundaries (firewalls). Therefore, we will have to think of how multiple global computers, each characterized by uniform guarantees of service, can interact effectively.

ACM Sigplan Notices, 32(1):66-68, 1997.
ACM Computing Surveys, 28A(4), 1996.

Models of Computation

What models of computation are appropriate for global structures? Traditionally, we have used sequential, functional, object-oriented, relational, concurrent, and distributed models. Although these models have been formally characterized, comparatively little formal work has been carried out on locality and other global computation issues. These issues are going to become central.

In order to program a global computer we first need to understand its model of computation. For example, does computation on the Web correspond naturally to a traditional model? There are indications that it does not; a common experience exemplifies this point. When browsing, we actively observe the reliability and bandwidth of certain connections (including zero or time-varying bandwidth), and we take action on these dynamic quality-of-service observables. These observables are not part of traditional models of computation, and are not handled by traditional languages. What models of computation and programming constructs can we develop to automate behavior based on such observables?

Programming Languages

Different global computers may provide different guarantees. Therefore, different programming languages may be appropriate in each case. As an example, consider the following programming languages that support mobile computation.

Telescript [4] is an agent-based language that explicitly deals with locality, mobility, and finiteness of resources. Telescript agents may migrate to new locations while active, but cannot engage in distributed communication. Telescript agents run only on a dedicated global computer that guarantees the integrity and security of agents.

Obliq [2] is an object-based language that encourages distribution and mobility. Mobile computations maintain distributed connections as they move. Obliq can run effectively on any single reliable global computer, but does not deal with administrative domains or widespread unreliability.

Java [1] deals with security and multiple global computers. (Its programs are allowed to cross administrative domains.) Mobility, however, is restricted to transmitting program sources, in preprocessed form, and not computations. As a consequence, Java works satisfactorily in unreliably-connected environments, since passive sources do not maintain connections. (Java is evolving to support active distributed computation, and will confront the same problems as Telescript and Obliq.)

Each of these three languages is better suited than the others to a particular kind of global computer. However, none of them is particularly well suited for carrying out general computation over the World-Wide Web. No language yet has the Web as its "run-time system": no language covers the full spectrum of Web behavior. Therefore, there is a need to develop more general semantic frameworks for global computation, in order to understand the assumptions and requirement of various languages and to answer important questions such as "Who runs what, when, and where?". How do we reason about global programs?

Programming Issues

Programming issues acquire new facets in a global context. Among these are:

Typing. Global computation, being highly decentralized, needs to rely on some common notion of typing for the data that is exchanged. The Internet already has a rather sophisticated system of data types, called MIME; unfortunately, Web servers provide mostly HTML data that is poorly structured. CORBA defines a rich type system for distributed computation, but it is disjoint from that of the Web. To enable global programming, data will have to be transmitted over well-defined typed links. Moreover, link types will have to be mapped to (or identified with) program types.

Security. A model of security is critical for mobile computation. The cryptographic underpinnings of security are well understood. Unfortunately, it is not clear how to effectively and flexibly integrate security into programming languages and mobile computations. Currently, paranoia rules. What should be the syntax, static checking, semantics, and logic of security?

Reliability. Some global structures will always be unreliable, it seems. Unfortunately, like the World-Wide Web itself, these may also be the most interesting global structures. Therefore, we need to find programming constructs and methodologies ("quality-of-service abstractions") that can increase the reliability of the substrate to tolerable levels.

Modularity. An appealing possibility, pursued by Java, is that software components will be fetched dynamically over the network, whenever the need arises. This approach requires modularity guarantees stronger than ever before, as well as novel approaches to software production, distribution, and maintenance. It also suggests a notion of interfaces and modules as dynamic entities that is rarely found in current languages.

Resource Management. Time, space, bandwidth, and services need to be managed. Moreover, substantial new challenges are offered by the handling and transfer of money in locally minute and globally huge quantities, within an open environment.

Directions

Today the standard computing infrastructure consists of locally networked personal computers with poor global connections. Gradually, this infrastructure will be complemented and replaced by network terminals that rely heavily and transparently on global resources. This transformation will be associated with the development of new computation and programming paradigms.

References

[1] Campione, M. and K. Walrath, **The Java tutorial: object-oriented programming for the Internet**. Addison-Wesley, 1996.

[2] Cardelli, L., **A language with distributed scope**. *Computing Systems*, **8**(1), 27-59. MIT Press. 1995.

[3] Internet Engineering Task Force, **Internet standards**. The Internet Society, <http://www.isoc.org>. 1996.

[4] White, J.E., **Telescript technology: the foundation for the electronic marketplace**. White Paper. General Magic, Inc. 1994.