Module Interconnection Frameworks for a Real-Time Spreadsheet

Progress Report
August 27, 1993

Program: SBIR N92-112

Scientific Officer: Ralph Wachter, Office of Naval Research

Principal Investigator: Richard Clarke, RTware, Inc.

1. Overview

RTware is working on a proposal for an SBIR Phase II software research and development project for distributed control systems (DCS) using Module Interconnect Frameworks (MIF). SBIR Phase I work includes research into the feasibility, design and benefits of such a system, resulting in a detailed Phase II proposal. The proposed software package will be based on RTware's ControlCalc real-time spreadsheet control system and the Polylith MIF system from the University of Maryland.

The prior progress reports have identified the main outlines of the proposed Distributed Control System (DCS), and described key benefits to the Department of Defense (see the May 30th Progress Report). The second progress report (July 29, 1993) divided the system into two major phases, DCS-I and DCS-II. DCS-I provided external access to module services defined by a ControlCalc application. A detailed specification of DCS-I was provided in the July progress report. DCS-II will allow the largely transparent distribution of compiled ControlCalc code segments across remote, and heterogeneous, platforms.

This progress report presents a more detailed description of DCS-II, and a summary of the work done in the last month.

2. Work in Progress Description

Work in the last month included:

1) Installation of the new Sun computer and testing of Polylith.

2) Initial phases of a port of ControlCalc to the Sun.

3) Porting Polylith to OS-9.

4) Exploration of specific issues for DCS-II.

3. Discussion in Detail

3.1. Polylith on Sun

The Sun computer was received and installed during this phase. The system was installed on RTware's network, and the professional Sun C development system was loaded. Polylith was loaded and built on the Sun and our test programs were verified. As expected, our Polylith test programs which had been run locally on the Encore machine were successfully run remotely, invoking services on the Encore side while running on the Sun and vice-versa. The system uses TCP-IP, which is the currently supported network protocol for Polylith.

3.2. ControlCalc on Sun

The ControlCalc source code (approximately 150,000 lines of C) was moved to the Sun and initial porting work was started. It is expected to take another 2 weeks to complete the initial port. The major work required to port ControlCalc to the Sun platform is the implementation of a Sparc code generator for the internal ControlCalc compiler. This work is expected to take 10 weeks, and will be included in the Phase II proposal. The initial port will allow ControlCalc to run in a multi-tasking interpreted mode which supports about 90% of the run-time functions. ControlCalc also includes a graphical user interface option that is based on X-Windows and Motif. During the next two months we will be looking into Sun's OpenLook system to determine how practical it is to support both.

3.3 Polylith on OS-9.

Since ControlCalc runs on the OS-9 and OS-9000 real-time operating systems in a wide variety of applications, it would be useful to have Polylith running on these platforms. Within the Distributed Control System, OS-9 running of VME 68040 processors would provide hard real-time processing while UNIX-based platforms would provide user interface, supervisory control, data logging, communication processing, etc. OS-9000 provides an identical environment running on PC-AT compatible systems. Both operating systems provide an Internet support package that supports TCP-IP, sockets, ftp, and NFS. While not POSIX-compatible, these systems support the standard C library.

RTware has now successfully built and run Polylith on OS-9, and used it to communicate with UNIX systems.

3.4 Exploration of DCS-II Issues.

DCS-II is defined as the transparent distribution of ControlCalc compiled control "scans" to slave processors. ControlCalc is already a multi-tasking system, where each page of the three-dimensional spreadsheet evaluates in a real-time compiled mode. Each scan is controlled by a separate process, with its own trigger mode (cyclic, event-driven, interrupt service). The compiled code directly references the shared memory spreadsheet data structure, allowing cells calculated in one scan to be referenced by another at any time. Counting semaphore functions allow synchronization between scans, through, for example, the construction of monitors. Although scans are implemented as processes, the system is conceptually a threaded system, where the spreadsheet data area is the common data space of a process and each scan is a thread of control running within that data space. In order to understand the issues involved in DCS-II, it is useful to review the mechanisms underlying ControlCalc's real-time scans.

The basic mechanism is to compile the formulae entered in the spreadsheet into C-callable subroutines, one for each page of the spreadsheet. Natural spreadsheet evaluation order (row-major) provides the

basic flow of the program. GOTO and GOSUB functions allow programming branches, loops and subroutines, similar to a BASIC program. Unlike BASIC, however, the normal result of each statement is to compute and return a value which becomes the current value of the cell containing the statement. In a very real sense, this type of spreadsheet evaluation can be considered a rule-solving system, where sets of rules (logical, text or arithmetic functions) are scanned to recompute the state of a system. This has a strong correspondence to control systems programming, which is often considered a set of control laws that are applied to a set of inputs in order to generate a set of outputs. Many functions, of course, operate primarily through side-effects, which we will consider below. One important feature: the spreadsheet does not allow address computation, except through indexing operations into ranges, or tables of data in the spreadsheet.

Disregarding side effects, the rule-solving evaluation mode results in a simple model of distributed evaluation, based on each scan being assigned to a separate, remote or tightly coupled processor. In downloading a page of the spreadsheet to a remote processor for scanning, one is also downloading the data space associated with all writes by that page, since a function's data and formula (or rule) are two attributes of a spreadsheet expression cell. Since no address computation is allowed on writes, inter-page references can be captured at compile time and packaged for remote access. In ControlCalc, range references are restricted to intra-page only. That means that ranges can only be two-dimensional and must therefore be fully contained within one page.

There is one important class of side-effect functions which violate the rule-solving nature of the spreadsheet. These are the "SETVAL" functions. Setval and its indexing cousin ISETVAL are the ControlCalc names for the functions which can write a value into a cell other than the current expression cell. These are assignment functions. There is no restriction on page locality for these functions. However, like all cell references, they either reference to a specific cell, determinable at compile time, or they index into a specified range of cells, determinable and restricted to one page. These functions are included in the ControlCalc programming model to simplify programming of state systems. Without these, a spreadsheet is a stateless language.

While stateless languages are quite robust, and easily understood for stateless applications, they required excessive complexity when trying to implement algorithms that are essentially state machines. In control programming, a combination of state and stateless programming is usually required. Stateless programming is used to apply invariant rules to a system. Continuous feedback algorithms are usually stateless, using running integrators and differentiators to measure time-dependent behavior. Safety interlocks are best implemented in a stateless manner. To do that, all output devices should be written at one point in the process. That way, illegal or unsafe conditions can be checked to minimize problems caused by, for example, unforeseen errors in a state sequence. On the other hand, many real-world processes are state machines, involving a sequence of events driven by changing real-world inputs. Implementing these sequences as a set of invariant rules makes the rules quite complicated, while a state sequence implementation would be very simple. Therefore a complete control programming system must include both capabilities. The impact of assignment functions on a distributed processing system is to require support for remote writes as well as reads.

In packaging remote data access, there are two choices: message passing or data mirroring. Message passing means that every time a remote data point is referenced within an expression, the compiler generates a remote read call directly in-line in the code. Data mirroring means that the remote data set is block-transferred at some point, and then accessed locally. While data mirroring is more efficient, it cannot be implemented reliably without a caching mechanism. Such caching would mean that data would be transferred through to remote nodes when written (write-through cache) or only when changed and required by a read (deferred write-through). The overhead of doing a software cache over a network makes it impractical. However, on systems with shared memory hardware, it is very practical. ControlCalc currently runs on such a system: the Reflective Memory System (RMS) used by Encore's Series 91 and Infinity computers.

In the absence of distributed shared memory hardware, we intend to implement a full message-passing system. This approach can achieve the performance of a data mirroring system, if the application were designed with distribution in mind. ControlCalc provides block copy functions (another variation on SETVAL) which lets data be copied between different ranges of the spreadsheet. By setting of the system so that remote data is block-transferred precisely when needed, local references can made to the copied data, greatly reducing the number of remote accesses required.

The explicit block transfer approach fits well with stateless programming techniques. In fact it can be an advantage in the software design because data references are packaged and executed only at one point in the logic, eliminating the possible race condition bus that often occur in multiprocessing. The disadvantage of explicit block transfers is that they must be explicit. The application therefore becomes less portable. More precisely, it becomes potentially less efficient when configured on a shared memory or single-processor system.

Message passing can be considered in two forms: explicit and transparent. Explicit message passing would involve a message from one scan requiring a response from another. This mechanism is already provided using the remote procedure calls running over Polylith. Explicit message passing requires the use of specific remote access functions.

A transparent message passing system would involve a data server process running automatically on each node that is running ControlCalc scans. The compiler would generate a standard message call that would request data from a remote node whenever a remote reference is encountered in the program. Since all scans are compiled at once, such data references could be pre-compiled down to indexed table lookups, reducing the overhead of traversing the spreadsheet sparse matrix to find each data point. The transparent system would also be implemented with Polylith, but without requiring specification by the user.

Finally, in all implementations, it is necessary that counting semaphores be supported across the entire distributed environment. Without this capability, it is difficult or impossible to create tightly coupled applications that work reliably in distributed environments. Once again, ControlCalc does not allow signal address (or i.d.) computation, so the compiler can determine the set of scans that are attaching to specific signals. Signals can therefore be arbitrated by any one of the set of scans, without requiring a central signal arbitrator. This will increase the robustness and performance of the system by allowing signal communication within multiple localized areas of the distributed application.

The key to efficient programming with transparent message passing is for the programmers to understand the mechanism and the cost of remote access. In particular, they should be aware that block transferring data prior to using it, and synchronizing block transfers with signals will both minimize communications traffic and increase software reliability.

3.5 Conclusion.

DCS-II can be implemented using compiler technology to package remotely executing threads and Polylith-level communication routines. The mechanics of communication message passing can be completely hidden from the programmer, who would still work within a shared memory, multi-tasking application model. Two types of remote access are proposed: shared memory and communication bus. Shared memory will use the existing ControlCalc shared memory model, and Polylith will be used for the communication bus.

4.0 Work in the Final Period:

The next two months are the final months of Phase I of this contract. The principle activity will be the preparation of a formal Phase II proposal, which is scheduled to be delivered on October 1. Technical activities will continue the development of the DCS-I concepts presented in the second progress report.

One more technical concept will be explored: distributed multi-user access to a spreadsheet. The basic concept is to split the user-interface part of the program from the spreadsheet data manager and provide low-level locking (transparent or user-defined by region) for editing operations. This will allow shared work-group access to the spreadsheet.

The Phase II proposal will include both DCS-I and DCS-II. Target platforms will include OS-9 (68K processors), OS-9000 (386/486 processors), LynxOS (386/486 and 68K processors), Solaris (Sparc Processors) and Unix SYS V running on Encore Series 91 and Infinity machines (88K processor). Graphical options will include DataViews and an internal RTware X-Windows product currently running on Encore machines. Communication will be provided by Polylith. Finally, the proposal will include enchancements to the function block capabilities of ControlCalc to support more flexible configuration and modular application construction.