Proceedings of the
International Workshop on Real-Time Ada Issuses
Moretonhampstead, Devon, UK
13-15 May 1987

Sponsored by Ada UK in cooperation with ACM SIGAda

Organizing Committee:

John Barnes
Benjamin Brosgol
Mark Dawson
Anthony Gargaro
Richard A. Volz

Units of Distribution for Distributed Ada

Trevor Mudge
Robotics Research Lab & Advanced Computer Architecture Lab
The University of Michigan
Ann Arbor, Michigan 48109, USA

Background

Since the beginnings of Ada over a decade ago there have been dramatic developments in computer hardware which have led to new possibilities for the design of large parallel or distributed computer systems, particularly those intended for real-time embedded applications. Developments have occurred on two fronts: 1) the level of integration possible on a single chip has followed "Moore's Law" [1] and has continued to almost double every year; and 2) fiber-optics and the availability of LSI chips that implement standard network protocols have simplified the hardware needed to construct local/small area networks.

The first of these developments has made sophisticated 32-bit microprocessors and 256 K-bit memory chips commonplace. This development together with developments in network hardware make it preferable to construct many real-time embedded systems as networks of microprocessor-based systems, where the processing can be placed close to the site of sensors or effectors. In many cases this placement of processing power simplifies system design, and improves the response times and the throughput rates. From a traditional viewpoint this may seem wasteful, but the increased cost of the hardware (if any), is greatly outweighed by performance, reliability, and ease of design. The latter is especially true if the alternative is one in which the real-time tasks must time-share a cpu; scheduling can make system design a nightmare, particularly when hard deadlines have to be met.

The developments in hardware have also resulted in the emergence of "massively parallel" computers [2] that are made up of a regular array of hundreds of processors in a non-shared memory configuration. Although programming these machines may seem to be a problem unrelated to that of programming an embedded real-time system, the fact that both have distributed memory structures means that they share many problems and can, therefore, both

benefit from any solutions to their common problems. Furthermore, Ada with its support for parallel and scientific programming, would seem to be a natural candidate for these new supercomputers.

To take system-wide advantage of strong typing, packages, tasking, exceptions and other features of Ada, these distributed systems—both the embedded ones and the massively parallel ones—should be programmed as a single Ada program. Given the need to do this, what is the appropriate "unit of distribution" ?

Recommendation

Within the current definition of Ada, it is not possible to define a unit of distribution that is without some shortcomings. Our recommendation is that library subprograms and library packages be the only distributable units. They represent a reasonable level of granularity for distribution, and they provide a reasonable unit for structuring distributed programs. Furthermore, as shown in [3], they do not require cross-machine dynamic scope management. This fact greatly simplifies compiler implementation and run-time support, allowing efficient cross-machine communications. Supplementary recommendations are: 1) Data objects created from remotely defined types should be placed with the unit creating them, with implicit and basic operations being replicated. User defined operations should remain on the unit elaborating the corresponding type definition. 2) Task objects should be placed with the unit initiating their creation.

The above recommendations could be presented as guidelines for the programming of distributed systems without requiring changes in the language definition. However, compiler and run-time support would still be needed for those situations that may arise should the user decide to ignore the guidelines. To avoid this, the recommendations should be part of the language definition. They represent a compromise that will require a minimum of reinterpretation and augmentation to the present language definition, while greatly facilitating the programming of distributed systems.

A more radical departure from the current language, and one that we would also like to see, is to have package types added to the language specification [4]. This would allow the further recommendation that task objects created from task type definitions be restricted to the units where the corresponding type definitions are elaborated. This would simplify distributed task termination without restricting tasks of the same type to the same processor. Package types would also simplify the programming of massively parallel machines. They are currently programmed in much the same way as distributed systems—each processor is programmed separately and communication is performed by a run-time system or simple operating system [5]. The package type would provide a natural abstraction for the "single code" model* of

*In the single code model, multiple copies of the same program execute asynchronously. Thus, different execution paths may be being taken by different processors. Each copy of the code is also a function of the position in the array of the processor executing it.

programming which is currently the way in which most applications are programmed for these machines [2].

References

[1] G. Moore, "VLSI: Some fundamental challenges," *IEEE Spectrum* **16**, (Apr. 1979), 30–37.

[2] J.P Hayes, T.N. Mudge, Q.F. Stout, S. Colley, and J. Palmer, "Architectures of a hypercube supercomputer", *Proceedings of the 1986 International Conference on Parallel Processing,* August 1986, pp. 653–660.

[3] R.A. Volz, T.N. Mudge, G.D. Buzzard, and P. Krishnan, "Translation and execution of distributed Ada programs: Is it still Ada?" to appear in the special issue on Ada of the *IEEE Transactions on Software*, Winter/Spring 1987.

[4] G.D. Buzzard and T.N. Mudge, "Object-based computing and the Ada programming language", *Computer,* March 1985, pp. 11–19.

[5] T.N. Mudge, G.D. Buzzard, and T.S. Abdel-Rahman, "A high performance operating system for the NCUBE", *Proceedings of the 1986 Conference on Hypercube Multiprocessors,* (to appear).