Parallel Computation Still Needs a Unifying Model to Play in the Mainstream

Domenico Talia

ISI-CNR

c/o DEIS, Università della Calabria, Rende, Italy

Email: `talia@si.deis.unical.it`

Towards the next century

Parallel computation in the last twenty years has been effectively used to solve many significant problems in several application areas of applied science and engineering. The research and development work both in parallel hardware and software showed as parallel computers allow the implementation of high-performance applications that cannot efficiently implemented in sequential computers [1]. However, until today parallel computing did not become a methodology used *in all the areas* of computer science applications. It has not yet one of the most significant portion of the mainstream of computing. Even if parallelism met important goals, to reach its maturity this goal should be a major challenge of parallel computing in the next years for assuring itself a future in the next century.

A major factor that limits parallel computation to be part of the mainstream of computer science is the lack of *general-purpose parallel computing models*. In fact, there is a large attention in industry and academy towards the parallel computing field. But, unfortunately there is no generally accepted model for parallel computing both from the software and hardware point of view. Parallel computers is a class of computing systems that includes a large number of very different architectures, from SIMD machines to distributed-memory MIMD computers. Therefore it seems difficult to define a unifying architectural model for parallel computing, as the von Neumann model is the unifying model for sequential computing. Several specialists think that finding a unifying model is not possible so they develop parallel software without a special attention to portability.

The architecture differences among parallel computers correspond on the software side to a large set of different parallel models and languages that often are architecture dependent and then they offer a partial solution to the problem of programming portable parallel applications as instead occurs in sequential computing using standard languages like C, Pascal, or Fortran. Many parallel programming languages used today are low-level languages that require the programmer to face the architectural issues of the parallel machine where the application is running. On the other hand, high-level parallel languages abstract from architectural issues but show unpredictable performance on different architectures. Thus porting the same program on different parallel computers, for example from a message passing multicomputer to a shared-memory multiprocessor, its performance may change dramatically.

A realistic strategy

To find a solution to the outlined problems and limits of parallel computation the following actions should be performed

- on the hardware side, the design and implementation of *general-purpose parallel computers* [2] that are able to support a wide range of programming models providing predictable performance;
- on the software side, the definition of programming models that are architecture independent allowing abstraction and portability across different parallel computers; at the same time these models should be quite simple and expressive.

In the 1980s and 1990s several general-purpose MIMD computers have been developed using microprocessors connected by a communication network or sharing a memory space. These systems, such as the Intel iPSC and Paragon, the Sequent Symmetry, the Transputer-based multicomputers, and the CRAY T3D, were effectively used to develop high-performance applications. Although they do not completely hide the architectural issues to the programmer, they showed a practical way to be followed to reach the goal of high-level general-purpose parallel computation. Along this direction new high-performance parallel computers composed of a large number of general-purpose microprocessors are going to be developed.

An important step to make these new parallel computers successful is the definition of high-level languages that are architecture-independent and demonstrate that parallel programming is not more difficult than sequential programming. Low-level approaches such as *PVM* and *MPI* [3] are driven by the *heterogeneous parallel computing* approach that tries to offer, on different computers, library primitives for parallelism and communication.

These approaches partially meet the portability goal but are based on low-level library functions that are tedious to be used and do not free the programmer from concurrency, communication and synchronization issues. In fact, even if PVM and MPI are two de facto standards in parallel programming, their programming style in many aspects looks like to assembler-level programming of sequential computers.

On the other hand, high-level approaches have been proposed in the latest years that may represent good candidates for architecture-independent programming models on general-purpose computers. They are the *Bulk Synchronous Parallel* (*BSP*) [4], the *LogP* [5], and the *Bird-Meertens Formalism* [6] models. Furthermore, other interesting models are the *skeleton-based* [7] and the *actor-based* [8] languages.

Although these models suffer from low performance, they represent an interesting starting point towards architecture independence because they abstract from architectural issues and allow predictable performance. If the parallel computing community will convince itself that with a clear strategy based on high-level languages could be possible to find a unifying model for parallel computation, in the next years those models can be used to drive this process. Adopting this strategy high-level programming and generality can be conjugated with high performance leading parallel computation to be a significant part of the mainstream of computing.

Conclusions

The issues addressed here are not simple to be solved and in same case a trade-off between different requirements must be found. However, in our opinion the general-purpose approach outlined above might represent a significant step towards a very large use of parallel computing in many application areas for developing portable and standard parallel software. Along these directions could be possible enlarging the community of parallel computing users to the majority of computer science users both in research and industry and making parallel computers the standard computing platforms of next century.

References

[1] D. B. Skillicorn and D. Talia, eds., *Programming Languages for Parallel Processing*, IEEE Computer Society Press, U.S.A., 1994.

[2] D. May, Towards general-purpose parallel computers, *Proc. CRAI Spring International Seminar on Highly Parallel Processing*, Capri, Italy, May 1990.

[3] J.J. Dongarra, S.W. Otto, M. Snir, and D. Walker, "A Message Passing Standard for MPP and Workstations", *Communications of the ACM*, vol. 39, no. 7, pp. 84-90, 1996.

[4] L. G. Valiant, A bridging model for parallel computation, *Communications of the ACM*, 33 (8), 103-111, 1990.

[5] D. Culler et al., LogP: toward a realist model of parallel computation, *Communications of the ACM*, 39 (11), 1996.

[6] D. B. Skillicorn, Architecture-independent parallel computation, *IEEE Computer*, 23 (12), 38-51, 1990.

[7] M. Cole, *Algorithmic Skeletons: Structured Management of Parallel Computation*, Research Monographs in Parallel and Distributed Computing, Pitman, 1989.

[8] G. Agha and C.J. Callsen, ActorSpace: an open distributed programming paradigm, *Proc. ACM SIGPLAN Symp. on POPP*, 23-32, May 1993.