Technical Editor:
Marcin Paprzycki
Dept. of Computer Science and Statistics
Univ. of Southern Mississippi
Southern Station 1506
Hattiesburg, MS 39406-1506
m.paprzycki@usm.edu

Structured Development of Parallel Programs

Reviewed by Andrzej Stachurski, Warsaw University of Technology

Structured Development of Parallel Programs
By Susanna Pelagatti
248 pages
$44.95
Taylor & Francis
London
1997
0-7484-0759-6

Structured Development of Parallel Programs presents a structured programming methodology for parallel computations that ensures portability, programmability, and good performance. The book's ultimate goal is to develop a suitable programming language for parallel programming and its compiler. This language is meant to deliver typical parallel constructs (skeletons) and their realizations (templates) on various architectures.

The book's first half presents a critical analysis of the state of the art of parallel software development. It also closely examines several existing approaches to parallel programming, concluding that template-based systems are the best compromise. In this approach, the programmer selects skeletons and their conversion rules, then uses them to build a program. Its performance might not match that of a low-level graph-based approach, but it is predictable and easily ensures programmability and portability.

The book's second half describes the P3L template-based methodology and its realization as the P3L language and its compiler, offering application examples. The author maintains that the template-based system gives rise to accurate performance models for the skeletons library designer as well as for the programmer. The technical and mapping details are left to the skeleton library designer, who can fully exploit specific properties of particular skeletons. The P3L methodology incorporates a small set of basic skeletons and their combination rules. Skeleton selection is based on the analysis of existing approaches. The skeletons reflect typical constructs that parallel program designers use.

The P3L methodology might be a good starting point for developing efficient high-level languages for parallel programming. It suggests how to ensure compromise between performance and portability and programmability. In any case, we should not treat it as something closed and finally established—high-level parallel programming languages continue to develop and improve.

Such high-level languages would let the programmer concentrate less on the details of the machine's architecture and more on the algorithm's design. The lack of high-level languages is one of the major obstacles hampering large, complex software projects and the development of computational algorithms. Currently, the progress of these languages is severely delayed compared to the pure parallel hardware performance. An efficient, high-level language for parallel programming available on computers with parallel processors and on clusters of machines used for distributed computations would be an important tool for people developing general theoretical and application-oriented algorithms.

This book should interest people working on parallel algorithms, but, more importantly, it should interest researchers and software engineers developing languages for parallel computations. It might also be of interest to both undergraduate and graduate computer science students because it does not require any special background. It can supplement material for courses devoted to programming languages and compilation techniques, especially for high-level parallel programming.

In computing, a parallel programming model is an abstraction of parallel computer architecture, with which it is convenient to express algorithms and their composition in programs. The value of a programming model can be judged on its generality: how well a range of different problems can be expressed for a variety of different architectures, and its performance: how efficiently the compiled programs can execute. The implementation of a parallel programming model can take the form of a library invoked Start by marking â€œStructured Development of Parallel Programsâ€ as Want to Read: Want to Read savingâ€¦ Want to Read. Currently Reading. Read. Structured Development by Susanna Pelagatti. Other editions.Â Weâ€™d love your help. Let us know whatâ€™s wrong with this preview of Structured Development of Parallel Programs by Susanna Pelagatti. Problem: Itâ€™s the wrong book Itâ€™s the wrong edition Other. It also addresses the organizational structure. Programming model is the top layer. Applications are written in programming model. Parallel programming models include â˜'.Â Development of programming model only cannot increase the efficiency of the computer nor can the development of hardware alone do it. However, development in computer architecture can make the difference in the performance of the computer. We can understand the design problem by focusing on how programs use a machine and which basic technologies are provided.Â Parallel Computer Architecture - Models. Parallel processing has been developed as an effective technology in modern computers to meet the demand for higher performance, lower cost and accurate results in real-life applications. The pipeline pattern provides for the rst structured parallel programming approach to MCTS. Moreover, we propose a new lock-free tree data structure for parallel MCTS which removes synchronization overhead.Â Much effort has been put into the development of parallel algorithms for MCTS to reduce the running time. The efforts have as their target a broad spectrum of parallel systems; ranging from small shared-memory multicore ma-chines (CPU) to large distributed-memory clusters.Â In this paper, a structured parallel programming approach is used to develop a new parallel algorithm based on the pipeline pattern for MCTS. The idea is to break-up the iterations them-selves, splitting them into individual operations, which are then parallelized in a pipeline. Parallel programs are more difficult to develop and reason about than sequential programs. There are two broad classes of parallel programs: (1) programs whose specifications describe ongoing behavior and interaction with an environment, and (2) programs whose specifications describe the relation between initial and final states. This thesis presents a simple, structured approach to developing parallel programs of the latter class that allows much of the work of development and reasoning to beâ€¦ CONTINUE READING. View PDF.