Journal of Parallel and Distributed Computing **62**, 669–695 (2002)
doi:10.1006/jpdc.2001.1809, available online at http://www.idealibrary.com on IDEAL®

From Design Patterns to Parallel Architectural Skeletons

Dhrubajyoti Goswami, Ajit Singh, and Bruno R. Preiss

Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1
E-mail: dgoswami@uwaterloo.ca, asingh@uwaterloo.ca, brpreiss@uwaterloo.ca

Received September 15, 2000; revised July 10, 2001; accepted August 20, 2001

The concept of design patterns has been extensively studied and applied in the context of object-oriented software design. Similar ideas are being explored in other areas of computing as well. Over the past several years, researchers have been experimenting with the feasibility of employing design-patterns related concepts in the parallel computing domain. In the past, several pattern-based systems have been developed with the intention to facilitate faster parallel application development through the use of preimplemented and reusable components that are based on frequently used parallel computing design patterns. However, most of these systems face several serious limitations such as limited flexibility, zero extensibility, and the ad hoc nature of their components. Lack of flexibility in a parallel programming system limits a programmer to using only the high-level components provided by the system. Lack of extensibility here refers to the fact that most of the existing pattern-based parallel programming systems come with a set of prebuilt patterns integrated into the system. However, the system provides no obvious way of increasing the repertoire of patterns when need arises. Also, most of these systems do not offer any generic view of a parallel computing pattern, a fact which may be at the root of several of their shortcomings. This research proposes a generic (i.e., pattern- and application-independent) model for realizing and using parallel design patterns. The term "parallel architectural skeleton" is used to represent the set of generic attributes associated with a pattern. The Parallel Architectural Skeleton Model (PASM) is based on the message-passing paradigm, which makes it suitable for a LAN of workstations and PCs. The model is flexible as it allows the intermixing of high-level patterns with low-level message-passing primitives. An object-oriented and library-based implementation of the model has been completed using C++ and MPI, without necessitating any language extension. The generic model and the library-based implementation allow new patterns to be defined and included into the system. The skeleton-library serves as a framework for the systematic, hierarchical development of network-oriented parallel applications.

0743-7315/02 $35.00

Key Words: design patterns in parallel computing; parallel programming environments; skeleton-based parallel programming; high-performance computing models; software tools for parallel programming.

Contents

1. INTRODUCTION

The term *design pattern* has been extensively used in the context of object-oriented software design. Patterns in this context describe strategies for solving recurring design problems in systematic and general ways [11]. Similar ideas are being explored in other disciplines of computing as well. For instance, ACE (the Adaptive Communication Environment) [27] is an object-oriented toolkit that implements various network-level patterns to simplify the development of concurrent, event-driven communication software. The "Pattern Languages of Program Design" series of books [17] is a good reference for anyone interested in pattern-related topics covering a diverse range of disciplines.

In the parallel computing domain, (parallel) design patterns specify recurring parallel computational problems with similar structural and behavioral components, and their solution strategies. Examples of such recurring patterns are static and dynamic replication, divide and conquer, data parallel computation with various topologies, compositional framework for irregularly structured control-parallel computation, pipeline, and singleton pattern for single-process (single- or multi-threaded) computation. This paper presents a generic, design-pattern based model and system for building parallel applications.

1.1. Pattern-based Approaches in the Past

As an important difference from the usages of the design-level patterns in the object-oriented domain [11], patterns in parallel computing are often employed not only at the design level but also at the implementation level. That is, the design-level patterns are often preimplemented as reusable components. This is similar in concept to a *framework* [18] in the conventional software engineering terminology.

Starting in the late 80s, several pattern-based systems were built with the intention to facilitate the rapid development of parallel applications through the use of preimplemented, reusable components. Some of the earlier systems include *Code* [4] and *Frameworks* [28]. Some of the recent systems based on similar ideas are *Enterprise* [26], *Code2* [5], *HeNCE* [5], *Tracs* [2], and *DPnDP* [30].

Concurrent to the above thread of development, another group of researchers started exploring parallel design patterns from a different perspective. This direction of research concentrates on the idea of replacing explicit parallel programming