Patternlets — A teaching tool for introducing students to parallel design patterns

Joel C. Adams
Department of Computer Science, Calvin College, Grand Rapids, MI, USA

Abstract

Thanks to the ubiquity of multicore processors, today's CS students must be introduced to parallel computing or they will be ill prepared as modern software developers. Professional developers of parallel software think in terms of parallel design patterns, which are markedly different from traditional (sequential) design patterns. It follows that the more we can teach students to think in terms of parallel patterns, the more their thinking will resemble that of parallel software professionals. In this paper, we present patternlets—minimalist, scalable, syntactically correct programs, each designed to introduce students to a particular parallel design pattern. The collection currently includes 44 patternlets (16 MPI, 17 OpenMP, 9 Pthreads, and 2 heterogeneous), of which we present a representative sample. We also present data that indicate the use of patternlets to introduce parallelism in CS2 produced a modest improvement in student understanding of parallel concepts.

1. Introduction

Virtually every modern desktop, laptop, tablet, and smartphone contains a multicore CPU. Two-, four-, six-, eight-, ten-, twelve-, fourteen-, sixteen, eighteen-, twenty-, and twenty-two-core CPUs are readily available for building systems [4]. Vendors are also selling accelerators, ranging from co-processors with tens of cores [5] to general-purpose graphics processing units (GPUs) with hundreds or thousands of cores [9].

Today's software developers thus have unprecedented computational power at their fingertips, but to take advantage of this power, they must create their software using parallel and distributed computing (PDC) techniques. Unfortunately, very few developers – or CS educators – have received any training in these techniques.

© 2017 Elsevier Inc. All rights reserved.

Keywords: Design patterns, Education, MPI, Multiprocessing, Multithreading, OpenMP, Parallel, Patternlets, Teaching, Threads.
One reason so few developers or educators have any parallel training is that prior to the TCPP [10] and CS2013 [11] curriculum recommendations, parallel computing was an elective topic. As a result, only developers or educators who happened to take an optional course in parallel computing received any training in parallel techniques, and relatively few CS programs offered such courses at the undergraduate level.

However, CS2013 contains a new Parallel and Distributed Computing knowledge area that includes 15 hours in the core CS curriculum. The Systems Fundamentals knowledge area adds additional PDC topics to be covered in the core.

The addition of PDC topics to the core, combined with the low levels of PDC expertise among CS educators, creates a special challenge for CS education. Before CS educators can train their students in parallel techniques, they must themselves develop sufficient expertise in those techniques. This is a challenge because PDC is a complex area, with multiple technologies to understand, at both the parallel hardware and the parallel software levels.

1.1. Parallel hardware technologies

There are three general categories of parallel hardware systems:

1. In shared-memory systems, multiple computational cores share the same primary memory, through which communication can take place. Modern desktops, laptops, tablets, and smartphones are all shared-memory systems. Most accelerator-equipped systems are a special case of shared-memory systems.

2. In distributed-memory systems, multiple computational nodes have their own local primary memories, but share no memory between one another. The nodes generally communicate through a network. Older Beowulf clusters and supercomputers were distributed-memory systems.

3. Heterogeneous systems are distributed-memory systems whose nodes are shared-memory systems. These include newer Beowulf clusters and supercomputers.

1.2. Parallel software technologies

Given the available hardware technologies, the situation is even more complex at the software layer:

1.2.1. Shared-memory software options

Programs for shared-memory systems can be written using either:

- Multithreading, in which a program consists of multiple threads (flows of execution sharing the same address space) that communicate using the shared memory. Such programs can be written in a language with built-in thread support (e.g., C++11, Java), or in a sequential language using an external multithreading system (e.g., C with OpenMP or POSIX threads).

- Multiprocessing, in which a program consists of multiple processes (flows of execution, each having its own address space that is inaccessible to other processes) that communicate with one another by sending/receiving messages. Such programs may be written in a language that supports message passing (e.g., Erlang, Scala), or in a traditional language using an external message-passing library (e.g., C with MPI — the message passing interface).

For high performance computing, C with OpenMP is a commonly used approach, on both CPUs and co-processor accelerators. Nearly all modern C compilers include built-in support for OpenMP.

For the special case of GPU accelerators:

- Nvidia’s CUDA (a dialect of C) is a popular technology, but is limited to Nvidia devices.

- OpenCL is platform independent and can use any of a system’s cores, but it is far more complex than CUDA.

- OpenACC and OpenMP-4 promise to greatly simplify GPU programming, but often at a significant performance penalty compared to CUDA.

1.2.2. Distributed-memory software options

Depending on the problem, programs for distributed-memory systems may be written in:

- A language that explicitly supports message-passing (e.g., Erlang, Scala).

- A traditional language with a message-passing library (e.g., C with MPI).

- Any language supported by the MapReduce/Hadoop framework (e.g., Java, Python, etc.).

For high performance computing, C with MPI is arguably the most commonly used of these approaches. Distributed parallelism is also useful for processing data too big to fit in main memory. For example, the MapReduce/Hadoop framework is popular for “big data” problems in which solutions can be computed using (key, value) pairs.

1.2.3. Heterogenous software options

Many programs for heterogeneous systems are built using C and MPI + X, with MPI being used to distribute processes across the system’s nodes and enable communication between them, and the X depending on the underlying nodes. For example:

- MPI + OpenMP in a system whose nodes contain just multicore CPUs, or CPUs plus coprocessor accelerators.

- MPI + CUDA in a system whose nodes contain CPUs plus Nvidia GPU accelerators.

- MPI + OpenCL in a system whose nodes contain a mix of CPUs and non-Nvidia accelerators.

2. Parallel design patterns

The large number of technologies available at the hardware and software layers can seem overwhelming to CS educators who are new to PDC. Thankfully, a higher-level conceptual framework exists that can provide some relief.

2.1. Parallel patterns background

Software developers have been writing software for parallel systems for over 30 years. During that time, these developers have noticed that certain parallel strategies and techniques were useful in solving many different problems. The professional developers began to name these strategies and techniques, so they could refer to them more conveniently. Eventually, this body of strategies and techniques was given a name of its own: parallel design patterns. Parallel design patterns are thus a collection of named strategies and techniques that industrial software developers have found to be useful in a variety of different contexts. Parallel professionals tend to think and design their software in terms of these patterns;
دریافت فوری
متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات