TOP-C: Task-Oriented Parallel C for Distributed and Shared Memory

Gene Cooperman*
College of Computer Science
Northeastern University
Boston, MA 02115
gene@ccs.neu.edu

Summary. The “holy grail” of parallel software systems is a parallel programming language that will be as easy to use as a sequential one, while maintaining most of the potential efficiency of the underlying parallel hardware. TOP-C (Task-Oriented Parallel C) attempts such a model by presenting a task abstraction that hides much of the details of the underlying hardware. DSM (Distributed Shared Memory) also attempts such a model, but along an orthogonal direction. By presenting a shared memory model of memory, it hides much of the details of message-passing required by the underlying hardware. This article reviews the TOP-C model and then presents ongoing research on combining the advantages of both models in a single system.

1. Introduction

This paper proposes the TOP-C model as a way to easily organize computations on DSM systems with many processors, while maintaining high concurrency. The proposed model allows the application writer to implicitly declare segments of his environment that correspond to the program objects that he is using. The segments are implicit in that the application writer need only declare to TOP-C which segments are modified by a given routine.

TOP-C has been successful in executing many large, parallel applications [4, 8, 10, 11, 12, 17]. TOP-C is implemented as a C library, and does not require a modification of the programming language of the application. As with any C library, the TOP-C library can also be used by a C++ program. One can choose any of three TOP-C libraries to choose between: SMP (Symmetric MultiProcessing, or shared memory) architectures, distributed memory architectures, and a sequential architecture. The application writer may continue to use his or her favorite programming language as long as that language has an interface to C libraries.

It should be noted that current high-end SMP architectures (many processors) are quite similar to DSM systems with hardware support. Hence, there appears to be a gradual progression from low-latency SMP through medium-latency DSM systems, with no sharp dividing line. Accordingly, we talk about the SMP version of the TOP-C model with the intention that this also applies to DSM.

* Supported in part by NSF Grant CCR-9732330.
Section 2. describes the TOP-C model. Section 3. then motivates why the model needs to be extended when the environment uses a lot of memory. Section 4. then describes a natural way to enhance the TOP-C model by providing an application abstraction of segments. If the application program is an object-oriented C++ program, then each segment will often correspond to an object.

Section 5. then describes how the enhanced TOP-C model maps onto a DSM architecture. In particular, there is an important issue of how the multiple segments of the TOP-C environment map onto the multiple pages of a DSM system. We are still in the process of obtaining a suitable DSM, and so we have not had the opportunity to test TOP-C in this environment. Nevertheless, a paper analysis describes many of the DSM features that we expect will be necessary for TOP-C to run efficiently on top of DSM.

2. The TOP-C Model

The TOP-C model has been described in [7]. The model is sufficiently flexible to also be easily ported to interactive languages [5, 6]. The model has also been applied to metacomputing [9], due to the ease of checkpointing the current state and sending a copy of that state to a new process joining the computation. The model has been successfully used in a variety of applications [4, 8, 10, 11, 12, 17].

The model allows a single file of application code to be executed as a sequential, SMP, or distributed memory application, by simply linking with a different library. Portability is emphasized by building on top of a POSIX threads library (for SMP) or MPI [14] (for distributed memory). MPI was chosen as a widely available message-passing standard, with good efficiency. The TOP-C distribution also contains its own small, unoptimized subset implementation of MPI, allowing one to quickly set up a small, self-contained application. Further, the portability of TOP-C makes it easy to re-target to another message-passing platform, such as PVM. TOP-C is freely distributed at ftp://ftp.ccs.neu.edu/pub/people/gene/top-c/.

The programming style is SPMD (Single Program, Multiple Data). This is executed in the context of a master-slave architecture and an environment or global state. This environment receives lazy, incremental updates, in a fashion that will be made clear later.

The user interface has purposely been kept simple by restricting the user interface to a single, primary system call: master_slave(). That function requires as parameters, four application functions declared by the user: set_task_input(), do_task(), get_task_output() and update_environment(). The philosophy is to present the higher-level task abstraction to the application. This should be contrasted to lower level interfaces that present either a message-passing abstraction or a shared memory abstraction.