

Integrating Parallel Programming into Distributed Computing

Chia-Chu Chiang^a, Gary Anderson^b, Sally Robison^c

^aDepartment of Computer Science

^bDepartment of Applied Science

^cDepartment of Mathematics and Statistics

University of Arkansas at Little Rock

2801 South University Avenue, Little Rock, Arkansas 72204-1099, USA

cxchiang@ualr.edu

Abstract: This on-going research addresses the concern of introducing the concepts of distributed and concurrent programming to the non-computer science students who, in the future, will most need to understand this skill to solve problems in their respective fields. To accomplish this, we are proposing a programming paradigm that allows students in a variety of majors in science, technology, engineering, and mathematics (STEM) to efficiently learn the concepts of distributed and concurrent programming. These concepts proceed from the abstract level to the low-level language details.

Keywords: Distributed computing, heterogeneity, parallel programming

Introduction

The goal of this research is to provide a simple way for students in a variety of majors in science, technology, engineering, and mathematics (STEM) to learn distributed and concurrent programming. In addition, those who do not have formal training in the field can be quickly introduced the concepts and skills of distributed and parallel computing.

1. The Proposed Programming Paradigm

We have developed an architecture, shown in Figure 1, to support the proposed distributed and concurrent programming paradigm [1-2]. The Common Object Request Broker Architecture (CORBA) is a middleware that allows computer applications to work together and exchange data across different platforms. An adapter was developed to manage the interactions among programs through CORBA.

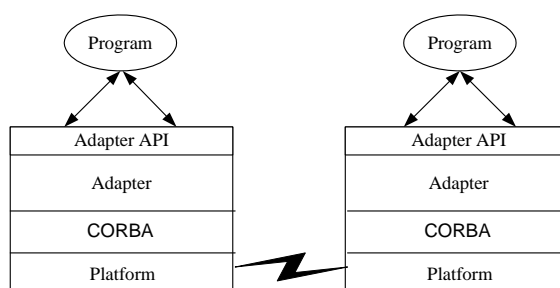


Figure 1. Layering of Architecture

The process students will use to develop programs using this paradigm is depicted in Figure 2. A student first defines the interface in an Interface Definition Language (IDL) file, which is a plain text file that declares modules and interfaces in a special language designed for this purpose. The IDL file will be automatically translated into a program template that students will fill in with code to complete the program. The template will contain a set of data definitions in the particular sequential programming language that is being used by the student (i.e., C or Java). The program is then compiled and linked with the CORBA run time library and the Adapter library into an executable file.

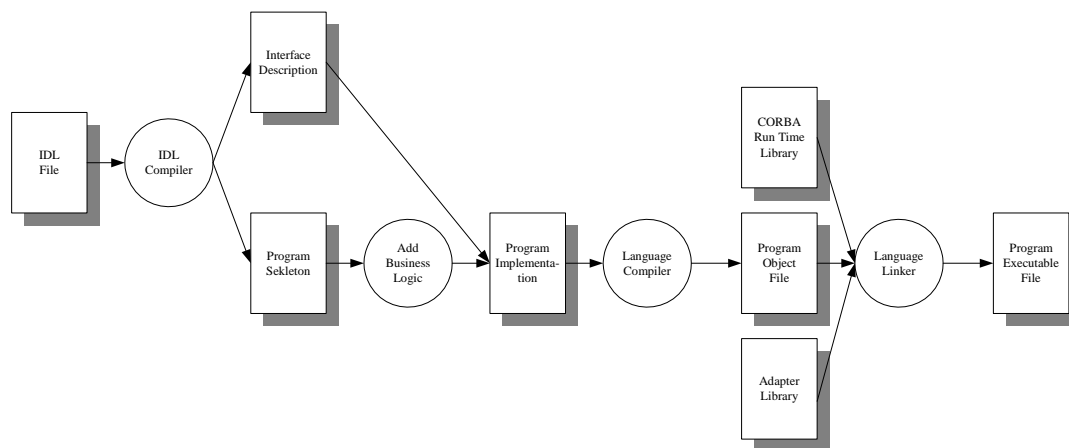


Figure 2. Program Creation

2. Summary

The proposed work will provide a programming paradigm for exposing undergraduates from a variety of backgrounds to a burgeoning high-tech method of solving complex problems, which will increase their capabilities when they join the workforce. Making the development of distributed and concurrent applications simple makes the topic accessible to a wide variety of undergraduates. Introducing the concepts of distributed and concurrent computing in an accessible manner will allow students to gain confidence in their abilities to use these techniques. Afterwards, the low-level details of code encapsulated in a library can be taught to introduce the concepts of abstraction, encapsulation, communication, concurrency, and software interoperability. Moreover, students from many STEM majors will have the opportunity to collaborate in interdisciplinary teams with students from other fields, improving their abilities to communicate effectively with people who have different technical backgrounds. In addition, having a programming paradigm that eases the development of the applications to solve complex problems helps retain STEM students, including under-represented groups in their fields.

References

- [1] C.-C. Chiang, "The Use of Adapters to Support Interoperability of Components for Reusability," *Information and Software Technology*, Vol. 45, No. 3, March 2003, pp.149-156.
- [2] C.-C. Chiang, "Development of CORBA Components in COBOL for Reusability and Interoperability", *Proceedings of the 20th International Conference on Software Maintenance (ICSM 2004)*, September 11-17, 2004, pp. 521.