IPDPS 2003 Tutorials
17th International Parallel and Distributed Processing Symposium
Tuesday, April 22nd & Saturday, April 26th – Nice Acropolis Convention Center – Nice, France

Tutorial 1 – Tuesday Morning
April 22, 2003

Introduction to Performance Tools Development: Build Your Own Tools!

Luiz DeRose
IBM T.J. Watson Research Center, ACTC
Bernd Mohr
Research Centre Julich, ZAM

%Introductory–20 • %Intermediate–60 • %Advanced–20

Description: Application developers are facing new and more intricate performance tuning and optimization problems as parallel architectures become more complex. Hence, users of high-performance computing systems normally complain that it is difficult to achieve a good performance for their application and are constantly asking for more and better performance analysis tools. However, the availability of tools development resources across the industry is shrinking rapidly because development of performance tools is complex and generally not profitable. In summary, more performance tools are needed, but fewer tools can be created by the industry. In this tutorial we present an overview of the major issues, techniques, and resources in performance tools development. Our goals are twofold: first, we will provide enough information, such that users who need specialized tools could attempt to do in-house development, in order to fulfill their needs. Second, we will discuss open problems in the area for researchers and students interested in working in the field of performance tools. Areas covered will include instrumentation, performance measurement, performance data representation, analysis, and visualization techniques.

Tutorial 2 – Tuesday Afternoon
April 22, 2003

Object-Oriented Middleware and Components for the Grid: Java, Corba Techniques and Tools

Denis Caromel
University Nice Sophia Antipolis, INRIA-CNRS-IUF
Christian Perez
IRISA INRIA

%Introductory–20 • %Intermediate–60 • %Advanced–20

Description: Object technology and middleware are increasingly used within parallel and distributed computing. At the same time, components are becoming a very effective tool for the composition and deployment of business applications. While the difficulties to actually program and deploy on the Grid have been, to a great extend, acknowledged, components technology is in the process of bringing many advances for such parallel programming. The aim of this course is to explain and detail this evolution.

The course objectives are:

- to explain the main principles of component technology,
- to explain how object-oriented middleware can be used for parallel and distributed programming,
- to state and to detail how object-oriented middleware, together with components turn out to be very effective for the Grid.

The course is based on widespread, industrial languages and frameworks, such as Java, Corba, CCM.

Tutorial 3 – Saturday, Full Day
April 26, 2003

Programming in the Distributed Shared-memory Model

Tarek El-Ghazawi
George Washington University
Bob Numrich
Minnesota Supercomputing Institute
Dan Bonachea
University of California, Berkeley

%Introductory–30 • %Intermediate–50 • %Advanced–20

Description: The distributed shared-memory programming paradigm has been lately getting rising attention. Recent developments have resulted in viable distributed shared memory languages that are gaining vendors' support, and several early compilers have been developed. This programming model has the potential of achieving a balance between ease-of-programming and performance. As in the shared-memory model, programmers need not to explicitly specify whether accesses are local or remote. Meanwhile, programmers can exploit data locality in distributed memory systems using an abstract model that can enable program portability.

In this tutorial, we present the fundamental concepts associated with the distributed shared-memory programming model. These will include execution models, synchronization, workload distribution, and memory consistency. We then introduce the syntax and semantics of three parallel programming language instances with growing interest. These are the Unified Parallel C or UPC, which is developed by a consortium of academia, industry, and government; Co-Array FORTRAN, which is mainly developed at Cray Inc.; and Titanium, a JAVA implementation from UCB. It will be shown through experimental case studies that optimized distributed shared memory programs can perform at least as good as message passing codes, without much departure from the ease of programming of the shared memory model.