MOCHA: A Database Middleware System Featuring Automatic Deployment of Application-Specific Functionality*

Manuel Rodríguez-Martínez , Nick Roussopoulos, John M. McGann,
Stephen Kelley, Vadim Katz, Zhexuan Song, Joseph JáJá
Institute for Advanced Computer Studies and Department of Computer Science
University of Maryland, College Park

Introduction

MOCHA[1] is a novel database middleware system designed to interconnect data sources distributed over a wide area network. MOCHA is built around the notion that the middleware for a large-scale distributed environment should be *self-extensible*. This means that new application-specific data types and query operators needed for query processing are deployed to remote sites in automatic fashion by the middleware system itself. In MOCHA, this is realized by shipping Java classes implementing these types or operators to the remote sites, where they can be used to manipulate the data of interest. All these Java classes are first stored in one or more *code repositories* from which MOCHA later retrieves and deploys them on a "need-to-do" basis. A major goal behind this idea of automatic code deployment is to fulfill the need for application-specific processing components at remote sites that do not provide them. MOCHA capitalizes on its ability to automatically deploy code to provide an efficient query processing service. By shipping code for query operators, MOCHA can produce efficient plans that place the execution of powerful data-reducing operators (filters) on the data sources. Examples of such operators are aggregates, predicates and data mining operators, which return a much smaller abstraction of the original data. In contrast, data-inflating operators that produce results larger that their arguments are evaluated near the client. Since in many cases, the code being shipped is smaller than the data sets, automatic code deployment facilitates query optimization based on data movement reduction, which can greatly reduce query execution time.

The architecture for MOCHA consists of four major components: **Client Application** - an applet, servlet or Java application used to pose queries to the system.; **Query Processing Coordinator** (QPC) - provides services such as query parsing, query optimization, query operator scheduling, catalog management and query execution, and is also responsible for deploying all the necessary functionality to the client and to those remote sites from which data will be extracted.; **Data Access Provider** (DAP) - provides the QPC with a uniform access mechanism to a remote data source, and executes some of the operators in the query, namely those that filter the data being accessed.; and **Data Server** - the server application that stores the data sets for a particular data site.

Description of the Demo

We have implemented MOCHA using Sun's Java Development Kit 1.2 and the system is operational at the University of Maryland. Our demo will showcase MOCHA interacting with an Earth Science client application. Users will be able run queries that extract satellite images and other GIS data (e.g. land features) from various sites, which will be combined and presented through a GUI. The client application and a QPC will be run from a computer located at the SIGMOD 2000 Conference venue. The QPC will connect over the Internet to the remote DAPs for two different data sources located at the University of Maryland. This demonstration shows the following unique aspects of MOCHA:

- Automatic deployment of the Java classes for the user-defined data types and operators used in a query to the client and remote data sites.
- Dynamic push of the code and computation of data-reducing operators (filters) to the remote data sites.
- Evaluation of data-inflating operators in a query near the client site.
- Easy and seamless specification and use of metadata for user-defined types and operators using XML and RDF.

In summary, our demo showcases a state-of-the-art database middleware system that is efficient, scalable and easy to maintain.

*This research was sponsored by DOD/Lucite Contract CG9815.

[1] Stands for **M**iddleware Based **O**n a **C**ode S**H**ipping **A**rchitecture.

MOD 2000, Dallas, TX USA