

MICE: A Mathematical Integrated Computation Environment

Jian Zhan , Lian Li

Computer Science Department, School of Information Science&Engineering,

Lanzhou University, Lanzhou, Gansu, 730000 , China

flysmart@lzu.edu.cn

Abstract: In this paper, we present a runtime environment for mathematical programs, which enables the distributed computation resources to be aggregated into a powerful mathematical computation entity to provide uniform services. A set of system design requirements are developed that cover the mathematical computation protocol for connecting clients and servers, end-user interface, computing engine interface, resource management in Internet, a broker mediated Client-server/server-server interaction, and job scheduling among services. A prototype has been constructed and a representative application is also implemented in java programming language. The end of framework aims to be a powerful, flexible, distributed and dynamic system.

1 Introduction

For the last 3 years, we have been developing a suit of tools named by MICE (Mathematics Internet Computing Environment), The goal of MICE is to construct a new working platform to provide large and complex mathematical computation through Internet. MICE can integrates much of existing mathematic software and provides uniform seamless access to distributed computing nodes. It provides a uniform programming interface to program on various math software deployed on distributed computing nodes, and a submission mechanism to improve computing performance. The ordinary task can be distributed to several ordinary nodes which deploys some common software, such as MatLab, SciLab, etc. Meanwhile, the special task can go with special nodes which have deployed unusual resources, including both the software and the hardware. It will also come with a free easy-to-use front-end interface. This client software manipulates user's display and transports the computation request to MICE server. In order to increase the computational capability, we can simply increase the CSP's number in the Internet and update the mathematical web service [1].

2 Related work

MONET [2], demonstrate the applicability of creating a semantic web to the world of mathematical software, using sophisticated algorithms to match the characteristics of a problem to the advertised capabilities of available services and then invoking the chosen se aspects. Firstly, MICE's message encode in MathML. Secondly, all computational nodes are deployed with famous mathematical software, such as Matlab to achieve high performance. Thirdly, MICE adopts the multi-domain federated server technology and request optimizing technology to

achieve good scalability and availability. rvides through a standard mechanism. The goal of MICE is some similar to MONET. But MICE is much different from MONET in several

RISC [3], a framework for brokering distributed mathematical services, in this project the broker (possibly in cooperation with deduction calculation) determines the suitable services and returns them to the client for invocation. The project is based on Web Service, and puts forward Mathematical Services Description Language.

RIACA [4] has developed a sort of tool. Through TCP/IP socket port that have been built in Server-side, user can remotely access the Mathematica computation on Internet. The protocol in communication is OpenMath.

In Kent State University, similar function can be obtained through Starfish, Dragonfly in project IAMC (Internet Accessible Mathematical Computation)[5]. The primary components of IAMC project include: the Mathematical computation protocol (MCP), a client prototype (Dragonfly), a server prototype (starfish), a mathematical encoding converter (XMEC), and an open mathematical compute engine interface (OMEI).

MathWeb [6] supplies MathWeb-SB (software bus) for web-supported mathematical computation. MathWeb-SB system provides the functionality to turn existing theorem proving systems and tools into mathematical services that are homogeneously integrated into a network proof development environment, its agent-oriented programming. And this group have developed CALCULEMUS, which is based on the communication functions provided by CORBA and the OpenMath [7] standard.

The rest of the paper is organized as below. Section 3 presents the architecture of the MICE system. Section 4 discusses the design and implementation of MICE. Section 5 gives the summary and the future work.

3 Architecture Overview

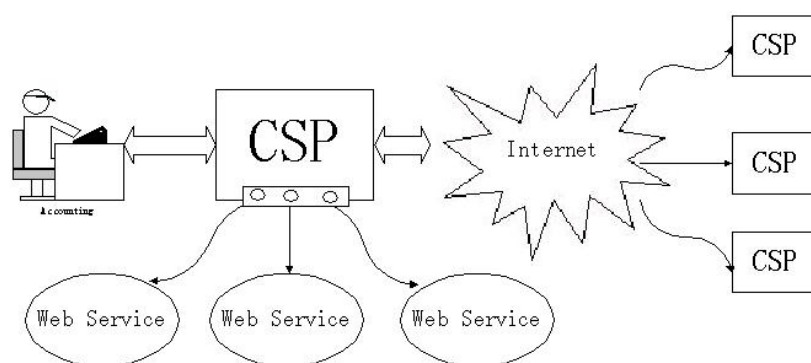


Figure 1. The computing environments of MICE

The computing environments, which essentially describe the user side of a computing system—how users interact via a set of distributed back-end resources. In MICE, with client software, user can connect with a CSP. All CSP constitute CSPD (Computing Service Platform Domain), and the domain provides services for user transparently. For every CSP in domain, the completion of computing task is realized through invoking interior basic function or registered

web services.

MICE propose a three-layer architecture for network computing, as illustrated in Fig.2.

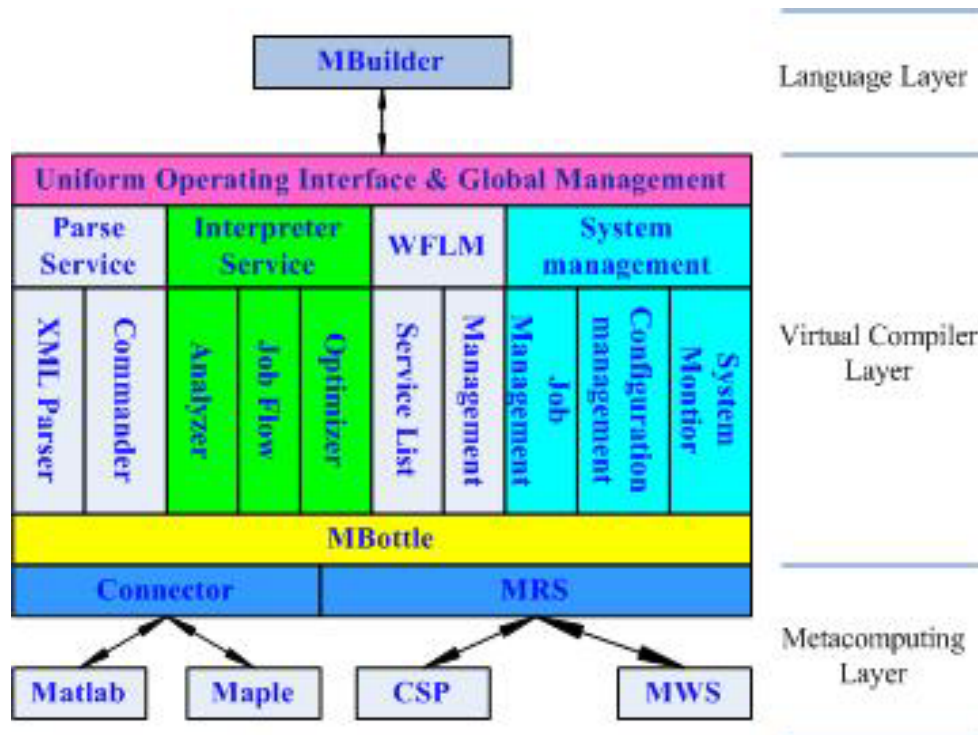


Fig.2 MICE Architecture

The first layer, Language Layer, is a representation way for computing job to user. We provide a pascal-like language name by M language for describe mathematical computing tasks in MICE [8].

The second layer, Virtual Compiler layer, is mainly for job scheduling on network computing. We implement it as CSP (Computing Service Platform).

The third layer, Meta-computing Layer, will encapsulation all computation entity in a uniform interface. We also implemented it as a part of our system. We called it UCS (Universal Computation Service).

4 Design and Implementation

The main components of the MICE include Client tools (MBuilder), CSP servers and UCS, as illustrated in Fig 2.

4.1 MBuilder

MBuilder is a environment which can edit the mathematical computational program by M language, view all the information of CSP, list and update the mathematical functions supplied by the MICE system. The Client tools receive the requests from users or applications and submit them to CSP. Meanwhile it accepts the processing results from CSP and displays them to users.

Other issues have been considered in Mbuilder including:

- Commands template

It's very common in practice that a series of simple commands make up a group of complicated commands. If these commands are formed in fixed order, users often don't want to repeat the same operation when they are doing similar tasks. Commands template provides users the ability by which they can build up corresponding template according to their requirement to resolve similar tasks.

- Setting of system parameters

This part includes the setting of windows style (background color, foreground color, menu style, button style), the style setting of result display (this means the result outputs by command line or standard mathematical formula), the setting of character (character format, character size), the setting of article style, etc.

- Converter

Converter's main task is to deal with the grammar check-up and convert the format of some content before they are transported to the CSP.

- Help

Users can know how to use the software by the help and some advanced content not involved in the interface: the technique of programming, communication with external application, etc.

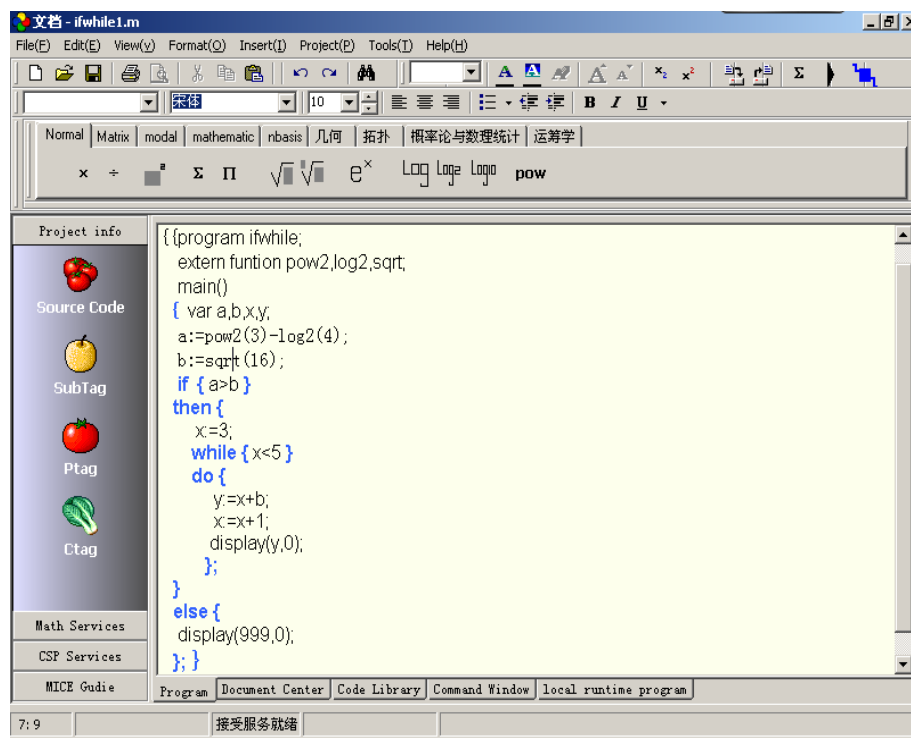


Fig.3 Mbuilder

4.2 CSP

The single CSP is consisted of six core components, including Uniform Access Interface, Parse Service, Interpreter Service, WFLM(Web Function List Management), MBottle, and System Monitor.

- The Uniform Access Interface provides a uniform access between MBuilder and CSP.
- The Parse Service is used to parse the job based XML encode.
- The Interpreter Service takes charge of analyzing, dispatching the job to small code piece, and optimizing the executing sequence of these codes.
- The WFLM provides mathematics function list in the CSP. It provides query and maintenance for CSP mathematical function list.
- The MBottle is mainly for scheduling the job on CSP.
- The system Monitor is a management tools based web, it mainly includes the creation and deletion of MICE system's users, system configuration and deployment and job status monitoring for the whole MICE system..

Now we introduce some detail mechanisms in CSP. CSP's workflow and framework are shown as Fig 4. Once the request is obtained from the user, CSP will work as follow [9]:

The first step is receiving messages, and then putting the receiving messages in a job pool. Then, The Parse Service's will work, which could fetch the mathematical program described from job pool and parse it to a program tree. The Parse Service also chooses the next action to execute according to the type of the instruction. Finally, the WFLM and Mbottle determine the suitable services for invocation. This mechanism thus hides from the client the coordination of mathematical services. If it is a simple operation like two numbers' sum, it will be executed immediately by java mathematics lib; if it is a Matlab operation, it will be send to Matlab connector to execute; if it is a mathematical web service operation, it will be send to the relevant web service; or if it is a small code piece it will be sent to other CSP.

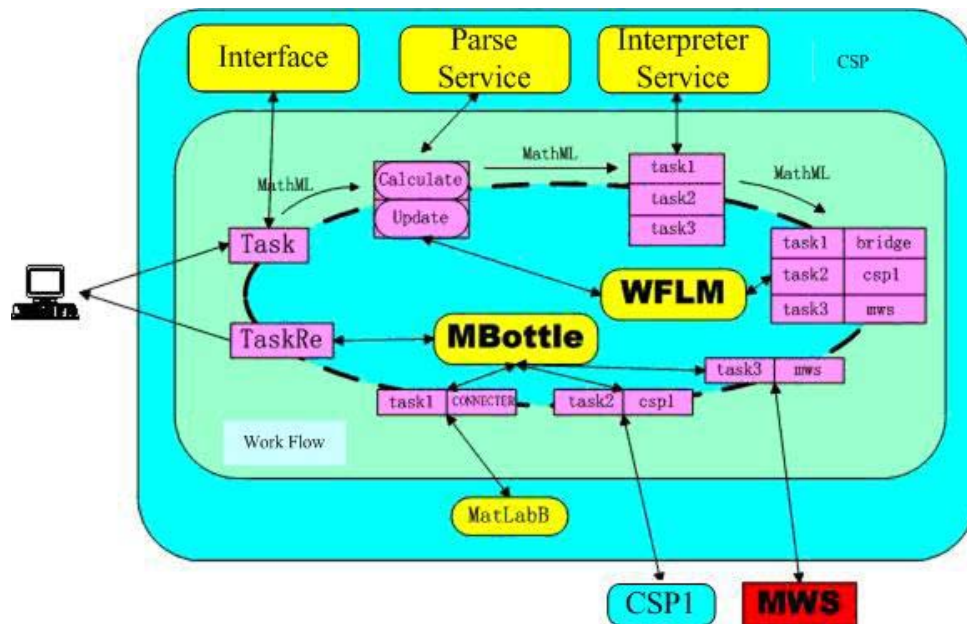


Fig. 4 CSP Work Flow

The major task of WFLM is to manage Functions List. In MICE the functions list will act as a “yellow pages” service. All functions provided by system can be classified and listed in the function list. For convenience, the functions can be, like a menu, shown to user. The function list is collected by the Server-side, and maintained by communication among servers. The function

list's format is considered as an extension of WSDL[10]. Every function takes up a segment for explanation, the content includes: function name (including whole name and shortened form), presentation format, parameter's type, invoking interface explanation (invoking environment, IP address of server), developing time, developer and other necessary explanation, even user's opinion and using records, etc.

The MBottle will choice the right strategy of transmission for every small code piece. There are three strategies in MICE: sequence strategy, load balance strategy and mix strategy. One strategy in our system meets one requirement [11].

Mbottle encapsulations a small code piece to a instruction, the new instruction will be send as following form:

Instruction number	Priority level	Goal server address	Source server address	Command of computation engine launching	The content of instruction

All CSPs are trust in each other, and construct a federation of network computing, the member of this federation is responsible for its services.

The possible communication in system is shown in following:

[1] User-CSP: User gets WFL; user submits computing task; CSP renders the result;

[2] WFLM-WFLM: HeartBeat test; broadcasts the information of WFL's changing. The HeartBeat test (a ping checked) is sure that the service is alive.

[3] Mbottle-Mbottle: task commissioning;

[4] Mbottle-Web Service: invoking;

4.3 UCS

The UCS provides registry and publication of computation entity in our System, it contains two important parts: Connector and MRS(Math Registry Service). The Connector supplies the CSP access to local mathematic function repository, such as Matlab which is installed on this CSP. The MRS provides information services of global mathematical services in the system. It provides query and maintenance for mathematical services information, access and management of computing service resources, security and authorization information.

5 Summary and Future Work

The design of MICE is evolving. But one way to test and refine the design is to proceed with the implementation of key parts of the MICE system to gain feedback and experience. We have finished the master part. Continuing work on MICE include design refinement, a GUI for the client on cross-platform platform, a reference implementation for the MICE protocol, and establishing various demonstration MICE's services.

The ultimate aim is to provide better tools for engineers and scientists who increasingly need access to sophisticated mathematical algorithms to tackle problems. The MICE framework will help guide users towards the most appropriate algorithms for solving their problem, running on the most appropriate hardware platforms.

References

- [1] Zhan Jian, Li Lian. Mathematical Network Computing based on Specification Oriented Programming. Master thesis, Computer Science Department, Lanzhou University, May 2003.
- [2] MONET <http://monet.nag.co.uk/cocoon/monet/index.html>
- [3]. RISC <http://poseidon.risc.uni-linz.ac.at:8080/index.html>
- [4] RIAACA <http://www.riaca.win.tue.nl/products/index.html>
- [5] IAMC <http://icm.mcs.kent.edu/research/iamc.html>
- [6] MathWeb.org <http://www.mathweb.org/>
- [7] OpenMath <http://www.openmath.org>
- [8] Zhenfang Li. Services-Oriented Mathematics Programming Markup Language. Master thesis, Computer Science Department, Lanzhou University, May 2004.
- [9] Zhan Jian, Li Lian. An Workflow Framework for Computational Grids. High Performance Computing and Application (HPCA2004) Workshop.
- [10] WSDL. <http://www.w3.org/2002/ws/>
- [11] Li Liu, Jian Zhan, Lian Li. A Runtime Scheduling Approach with Respect to Job Parallelism for Computational . LNCS 3251 GCC2004, 2004