Proceedings of the 5th WSEAS Int. Conf. on SIMULATION, MODELING AND OPTIMIZATION, Corfu, Greece, August 17-19, 2005 (pp531-534)

Distributed and heterogonous simulation program

Jiří KULHÁNEK, Radim FARANA
Department of control systems and instrumentation
VŠB Technical university of Ostrava
17. listopadu 15, Ostrava 708 33
CZECH REPUBLIC
jiri.kulhanek@vsb.cz, radim.farana@vsb.cz http://www.vsb.cz/~far10

Abstract: - This contribution shows some advantages of distributed and heterogonous simulation program concept. In the paper the developed simulation program SIPRO will be shortly described. Well-established simulation program SIPRO is focused on education of control systems, because it is a low cost solution and it is easy to learn. The used developer's platform is DCOM – cornerstone of the last Windows operating systems version. The DCOM programming technology acts as a connection for heterogonous modules of modular application.

Key-Words: - SIPRO, distributed, heterogonous, simulation, DCOM

1 Introduction

In this contribution the advantages of DCOM programming technology for developing of distributed and heterogonous programs will be shown. An example of such a simulation program will be a SIPRO simulation program, which we are developing continuously on our department.

The SIPRO block oriented simulation program is being developed under hard competitive condition of a large set of other well-known simulation programs. Reasons for its continuous development are in particular: it is the low cost solution of simulation tool for education [2], it has a simply and easy to learn user interface and finally – we maintain precise control of source code and program features [3], [4], [7].

Over the years, the program has changed from one purpose software without user interface into a user friendly version for Windows OS. The user interface in the last program releases didn't change significantly, which is a good thing for simplicity and backward compatibility reasons. But the internal architecture and program possibilities were changed greatly. The whole source code was rewritten with the use of up to date software technologies. The core technology used in the program is the DCOM – Distributed Component Object Model. It allows easy communicating with other programs, it is possible to embed a simulation program in web applications and at last but not least, it allows distributing simulation tasks over computer network.

2 COM/DCOM and other technologies

There are many programming technologies for distributed and heterogonous applications on the market. But we are using the Microsoft Windows operating systems platform, so we are using the developing technologies for WIN32 platform.

The predecessor of all the later mentioned distributed technologies was RPC (Remote Procedure Call). The RPC technology is used in operating system till now, but it is used only for simple purposes or it is used to provide compatibility with older systems. RPC technology was built up on C programming language, and extends it by the new own language – IDL (Interface Definition Language). The IDL describes an interface between distributed applications and it allows communication between different programs across computer networks. From the IDL language Microsoft derives a newer programming technology - the COM (Component Object Model) and its distributed version - DCOM. COM and DCOM technologies are now heavily used in Microsoft's operating systems and in all various types of applications.

2.1 COM/DCOM

The COM technology makes it possible to create reusable and independent software blocks – the COM components. Each of the COM components can contain several objects. The COM objects acts as servers; they provide their functionality to clients – the applications. The applications are built up from their own source code and from several external COM components. This way it makes the developing of applications easier and faster.

The communication between an application and COM objects is provided by a small set of exactly defined points. These sets are called interface, it is the fundamental part of a component technology.

DCOM is a distributed extension of COM technology. The distribution of a COM component is often very easy; it can be realized by a configuration tool or programmatically. DCOM technology is suitable for applications in Microsoft Windows operating systems. Distribution of DCOM objects over network is often limited only to LAN or Intranet networks. There are big

communication problems with connections over internet firewalls; it is caused by the security rules of DCOM.

3 The SIPRO program

The SIPRO simulation program is based on COM objects. The most important objects are concentrated to a computing core. The computing core provides interfaces for various programming languages and technologies (see Figure 1). The interfaces can be consumed by various clients (applications). Clients can create or run simulation tasks.

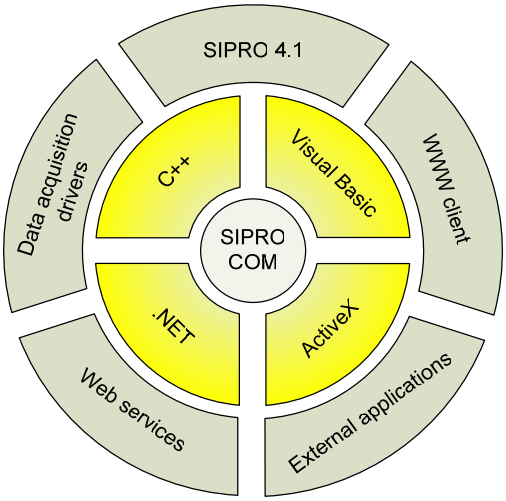

Fig. 1 Centralized architecture of SIPRO program computing core

On Figure 1 the centralized architecture of SIPRO COM object model is shown. The circle in a centre of this picture is the SIPRO computing core. It is programmed in C++ language. Objects and interfaces of computing core are prepared for communication with various applications by the use of various programming languages and technologies (see the middle layer on Figure 1). The exterior layer in the picture is an example set of different applications – clients of computing core [6]. The one of them is the SIPRO 4.1 application.

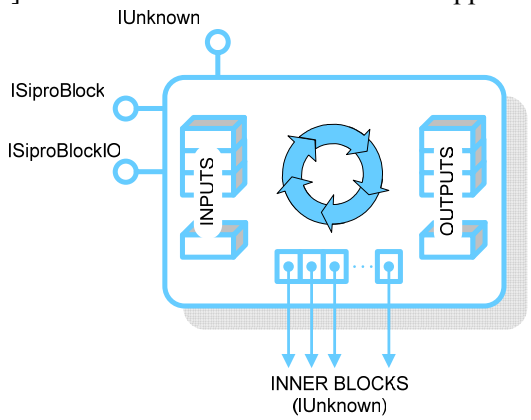

Fig. 2 Common structure of the SIPRO block objects

Architecture of the SIPRO 4.1 application is greatly affected by computing core objects and interfaces, for example: if the core is appended with a new simulation block, the SIPRO 4.1 client application automatically implements this change to user interface.

The component core consists of many COM objects (and the whole SIPRO application too). The smallest ones are simulation blocks – each simulation block is separate DCOM object. This feature allows distributing of each simulation block to a different computer. A scheme of one simulation block is in Figure 2. The block has a set of inputs and set of outputs, a macro block can own a set of compounded blocks. Each block has two proprietary interfaces – ISiproBlock and ISiproBlockIO.

There is another very important COM object in SIPRO, the compute object (see Figure 3). The compute object realizes the computation of simulation task.

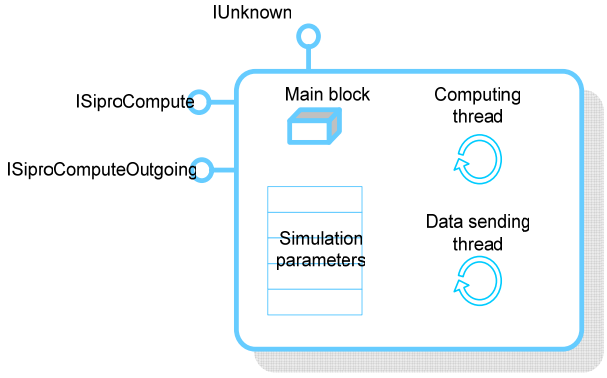

Fig. 3 Compute object's structure

Each compute object has its own copy of simulation (main block) and it has a set of simulation parameters (for example solution step, end time etc.) Due to the distributed manner of the SIPPRO application, the compute object has two parallel threads. The main thread computes a simulation task; generated solution outputs are stored to resizable memory structure. The second thread pops the stored data and it sends them to consumers. Thanks to such architecture, the compute thread isn't dependent on speed of output data consumers (charts, files etc.).

An interesting question is the speed of COM and DCOM technology. We have realized the tests of performance for various configurations. The result of COM performance was very similar to performance of a classical monolithic program (without COM technology). In the next chapter testing of performance for DCOM technology is described.

4 The computing speed analysis

The DCOM technology uses the identical components and objects as the COM technology, but the speed of distributed object are greatly affected by computer networks [5].

Distributed communication slows down by three main reasons. When the distributed application starts, it has to connect to remote computer(s) and has to provide security authorization. This process is dependent on many factors; it differs for various types of OS and versions of service packs. After the security negotiation, the distributed communication can be started.

The distributed communication itself is slowed down by two time constants. The first constant is delay of a distributed function call. The second constant is time for transmition function parameters.

For evaluation of the two mentioned time constants we prepared a testing application. This application makes millions of distributed calls for a function without parameters and it makes millions of distributed calls for a function with parameters of different size.

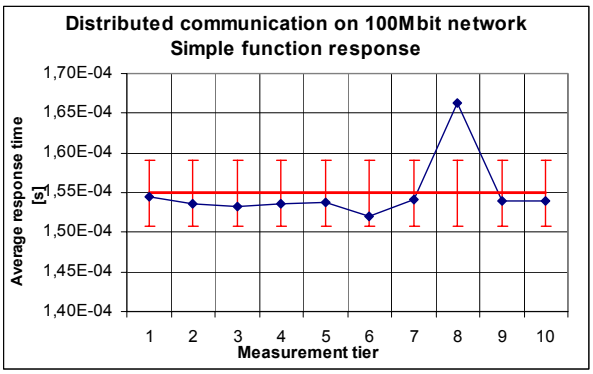

Fig. 4 Distributed response of function without parameters

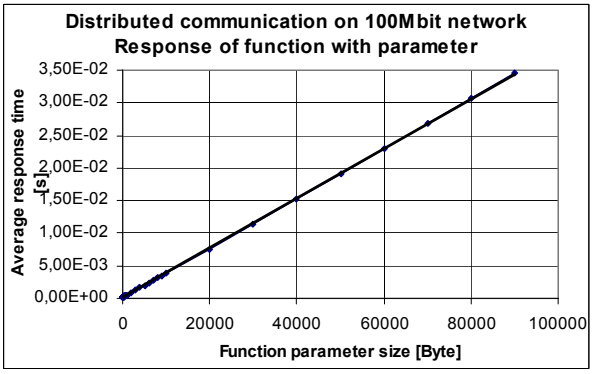

Fig. 5 Distributed response affected by parameter size

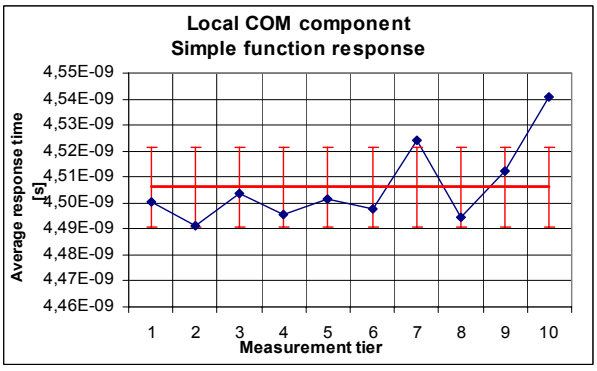

Fig. 6 Response time of a local function call

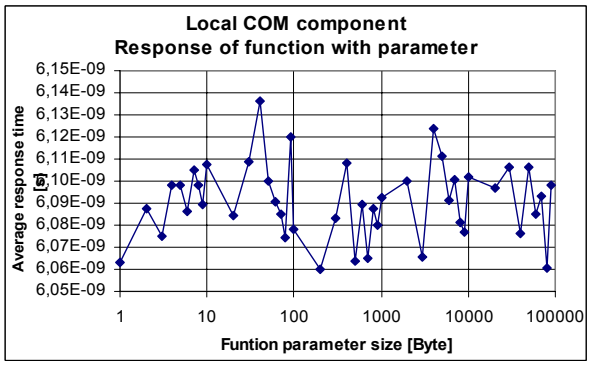

Fig. 7 Response time of a local function is not affected by parameter size

The Figures 4, 5, 6 and 7 show results of performance measurement. In Figure 4 the chart of distributed function response is shown, the distributed calls were made over unloaded 100 Mbit/s Ethernet network. The average response time was 155±4 μs. In Figure 5 the function response time depending on a function parameter size is shown. The dependency is linear. The time constant for transfer of one byte of parameter is 0.38 μs.

In Figure 6 the chart of a local function call response is. It looks similar to the distributed call response, but it has a very different average time. The average time is 4.51±0.02 ns.

In Figure 7 response time of a local function call for various sizes of a parameter is shown. The parameter size doesn't affect the local function call response (as was expected).

The response time of a distributed function call is about 35 thousand times slower than response of a local function call. The distributed function call is significantly dependent on a function parameters size. They are important factors, which must be considered upon design of a distributed simulation.

3 An example of a distributed simulation

The distributed function calls are much slower than the local calls. Because of that, the distributed simulation has to eliminate frequent distributed calls. A typical simulation task consists of tens or hundreds blocks. Each block is called in each step of a simulation. Distribution of separate blocks to different computers will lead to great slowdown of a simulation.

Another way is to distribute the whole simulation instead of the separate blocks. This way it is affected by a number of simulation outputs. There is not a significant slowdown of simulation.

The SIPRO simulation program has the possibility of parallel (or distributed) computing of identical simulations; the simulations differ only in one parameter.

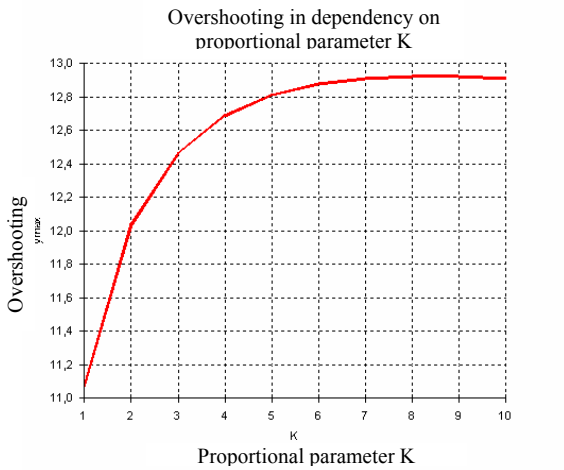

Fig. 8 Dependency of a system overshot on proportional parameter K.

In Figure 8 output of a parallel simulation of a simple system is shown. The proportional controller parameter K varies from value 1 to 10 with the step value 1.

A chart in the picture shows the system overshot in dependency on the K parameter. To compute such type of simulation, the SIPRO program runs ten simulations parallelly, each simulation with a different parameter K.

The simulation can be faster on computers with parallel CPUs, but the real speedup occurs in a distributed simulation. The simulation is then faster (see Figure 9) in accordance to a number of involved computers.

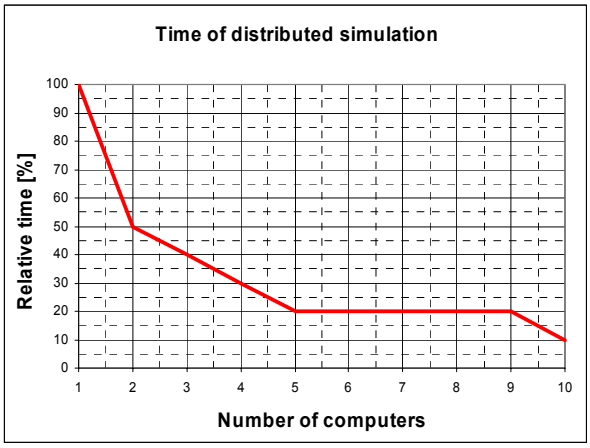

Fig. 9 Relative time of distributed simulation for various numbers of computers

7 Conclusion

The target of this paper was to introduce possibilities of distributed computing technologies for system simulations. For the reason of quick functionality, availability and wide range of possibilities the COM/DCOM technologies were chosen for making distributed applications in LAN.

In the paper the influence of COM/DCOM technology on speed of functions was verified and analyzed.

It shows that separating of user interface and a working core is a prospective use of distributed computing. This trend can be seen in various commercial products (Matlab, Dynast etc.). The user interface in SIPRO program is technically separated from computing core (see Figure 1), but the client application is not separated by default.

Next step of SIPRO program evolution will be a Web user interface. The designed architecture of the Web SIPRO application supposes a computing core stored on an application server side and user interface stored on a Web server [1].

The current version of the SIPRO simulation program is on http://homel.vsb.cz/~kul74/projekty/sipro4.htm. It is an evaluation version for testing purposes in Czech language only. It is tested by various people [8].

References:

[1] BABIUCH, M. 2004. Web Applications of Sensors and Measurement Laboratory. In *Proceedings of XXIX. Seminary ASR '04 "Instruments and Control"*. Ostrava : Katedra ATŘ, VŠB-TU Ostrava, 2004, pp. 9-12. ISBN 80-248-0590-1.

[2] FARANA, R. 1996. *Univerzální simulační program Sipro 3.4*. Uživatelská příručka. Ostrava : KAKI, 1996. 116 p. ISBN 80-02-01087-6.

[3] FOJTÍK, D. *Realizace řízení v reálném čase pod Microsoft Windows 2000/XP*. AUTOMA, vol. 10, č. 11, pp. 43 – 47. ISSN 1210-9592.

[4] KOČÍ, P. Evaluate the Revolutions from Measured Current. In *Proceedings of the 6th International Scientific – Technical Conference Process Control 2004*. Pardubice : University of Pardubice, 8. – 11. 6. 2004, pp. 236-1 – 236-6. ISBN 80-7194-662-1.

[5] KULHÁNEK, J. 2004. The Speed of Component-Based Application in .NET Platform. In *5th International Carpathian Control Conference*. Zakopane, Poland : AGH-UST Krakow, 25. – 28. 5. 2004, pp. 843-848. ISBN 83-89772-00-0.

[6] LANDRYOVÁ, L. SCADA Applications based on .NET Architecture. In *5th International Carpathian Control Conference*. Zakopane, Poland : AGH-UST Krakow, 25. – 28. 5. 2004, pp. 313-318. ISBN 83-89772-00-0.

[7] SMUTNÝ, L. & ŠKUTA, J. University Experimental Education with Laboratory Models of Real Technological Processes. In *5th International Carpathian Control Conference*. Zakopane, Poland : AGH-UST Krakow, 25. – 28. 5. 2004, pp. 509-514. ISBN 83-89772-00-0

[8] WAGNEROVÁ, R. Robust Control Design for Technological Processes. In *5th International Carpathian Control Conference*. Zakopane, Poland : AGH-UST Krakow, 25. – 28. 5. 2004, pp. 289-294. ISBN 83-89772-00-0.