The Communication in Distributed Client – Server Systems

Valeriu Lupu and Cătălin Lupu

***Abstract*—This paper presents a software for communication in distributed client – server systems designed using Java programming language. The server stores in a PostgreSQL database prismatic and rotation parts needed for processing with numerically controlled machine. The server and the clients can be placed anywhere. The requests are sent from server to clients or vice-versa, each client accessing the part existing on server that must be processed locally. Once the part accessed, the client receives the processing part program using numerically controlled machine, the command program of the industrial robot for unloading the parts (in an intermediate storage on MU1 machine or in the finished goods storage on MU2 machine), geometrical, functional and topological parts' features (necessary for recognition using an artificial vision system placed on an industrial robot).**

***Index Terms* —client, server, web, Java, image.**

I. Introduction in Domain

IN a network environment classic with mainframe computers, an application such as a database running on a powerful central computer is accessed through terminals. The terminal forwards a request to the mainframe computer, finds the information requested and it shows the terminal.

The entire database is transmitted from the network server in the form of messages and landed on the client computer that launched the request. File transfer takes place through the operating system for network and cable, there is a close coordination between the client and server computer to determine the data sought transmitted. The transfer of data between clients and servers increases network traffic and difficult to service other customers.

Development of computer networks has created opportunity for corporations and institutions and to support the business cycle with the information systems on a large scale. Transactions between agencies, known as electronic commerce (e-commerce), led to new patterns of interaction between organizations and individuals. In this environment, movement / transfer of funds is done electronically and must fall within certain deadlines. Business applications that require modern length transactions to be subject to constraints of time in different degrees and are based on the interaction between different computer systems. The real-time Real Times Systems (RTS) and database client-server Client - Server Databases (CS-dbs) are two basic goals that can help said. Although there have been plenty of individual studies in these two areas, a research overview was not conducted. In an RTS, task-ROMs proposed system to be enforced limits were imposed by application requirements. Using modern techniques of programming and use of knowledge about the task envisaged sites, RTSs have been designed so that constraints on Job's individual meet a small percentage as possible deadlines. On the other hand CS-dbs were designed to benefit from the rapid improvement of computer power and the ability of networks to transfer data to offer high rates of audience. This improved performance is derived from the efficient use of resources had available in a network of customers. In this context, this process will be trading in real time in a CS-dbs and will provide indicators on performances and scalability of such systems. In RT-dbs transactions involving the operations of I / O have imposed time constraints on their achievement. Time constraints are specified normally in the form of deadlines (deadlines). A deadline is the last period of time possible for a transaction to be completed in time (be useful). If you do not fall within the time limit is exceeded or when the transaction is abandoned and the new request is made by the server to meet the transfer of data. Therefore, an important measure of the effectiveness of RT-dbs is the percentage of transactions falling within the deadline specified. In this paragraph is to investigate the performance of CS-dbs in view of tasks in real time. Name aggregate of these two systems is Client - Server - databases in real time (CS-RTDBS - Client-Server Real-Time Databases) (Figure II - 21). It will use the resources available at the site and the customer will exploit the data provided by the transaction to support a process in real time. A central database in real time called Centralized Server Based Real Time Databases (EC-RTDBSs) (Figure II - 22) is used as a basic fundamental way. This foundation is used to determine the parameters and operational profit in the configuration of CS-RTDBS may offer promising results in terms of performance expected. I chose not to use a database system in real time as a starting point (foundation) for two reasons:

- the majority of current systems of data in real time have centralized nature
- the absence of connections between nodes multiple databases, the flexibility of such a system is limited.

Although the sites customers often seem to be directly accessed by users, the situation is not always this. A bunch format between client sites and related servers can be used as a configuration of a system kernel that runs applications through a virtual private network VPN (Virtual Private Network). Users subscribe applications on the periphery of the nucleus.

Configuration CS-RTDBS proposed can be used to facilitate the effective development of a wide range of systems applications. This set of software applications include:

- Services database very useful (Highly Available Database Services): Such a database is the heart of many operations in telecommunication operations and their purpose is not only to work with a large volume of data in real time, but and offer customers advanced services and information.
- Architecture for Multimedia Servers (Multimedia-Server Architectures): storage of a large number of multimedia sources can be achieved by using multiple servers cooperate. They may respond to rapid changes in the characteristics considered in parallel with respect to requirements of predefined quality of service.
- Ultra fast delivery of information via the Internet (Ultrafast Internet Content Delivery): There are many current efforts that attempt to use Web technology in all fields. This can be achieved through the use of delegation or multiple features content globally.
- Effective access to large communities of users of electronic commerce (Efficient Access for Massive E-Commerce User Communities): In search of ways to ensure the existence of available resources requirements of the user at any time, sellers remarkable and corporations plan and implement systems multiserver able to isolate class requirements. Organizations may exceed delayed response of the server overload.
- Infrastructure (WAP Wireless Access Protocol - Related Infrastructure): WAP was developed as a mechanism that allows access to information from the Internet to a wide range of users who have limited resources.

The following configuration will present models for databases in real time and will present the algorithms on these models. The transaction may use two models:

- model centralized database
- model of the database client-server .

Here we assume that the database is a collection of objects in uniquely identifiable. In the framework of the centralized database (CE-RTDBS) server database operates all transactions.

Server-RTDBS allocated a thread (thread of execution) to the client / terminal in the system. This Thread maintains a socket with the customer throughout the transaction. Each client sends its transactions initiated by the server using the socket, where all transactions received from all customers are scheduled and executed. Server-RTDBS was designed to be able to process a number of transactions simultaneously. This is done by executing each transaction as an independent thread. The number of transactions that can run competitor depends on memory space available and access to database objects. This is a form of rudimentary communication (query-shipping). Customers (terminals) indicate the figure are points of service and not made any process of trading.

In CS-RTDBS server allocates a thread for each client. Each thread maintains such second socket connection with the customer throughout the transaction. A connection is used exclusively for messages while the other is used to transfer the database. Here too, as in the case of EC-RTDBS for each transaction is allocated a thread or separate but distinct difference is that each customer transactions are scheduled local, independent server. Customers also use the memory of short and long term they have available. The data to the server are stored in memory so that future applications of local data can be met without interact with the server. Coordination of competing requirements is evaluated by the server using a table lock (lock table). Clients are allowed to have two types of "lock" sites: shared-version "read only" (read-only) and exclusive - the version read-write (read-write). The version read-write can be attributed to just one single customer, in turn, while the version of "read-only" can be accessed by more customers simultaneously. The server connects objects given access to customer requirements and responding in the order in which they were made

These features are needed in order to recognize the parts about to be processed using numerically controlled machine. The parts image is captured by two video cameras placed on a robot in perpendicular planes. The image captured is 3D. For recognition it will be used a neuronal architecture such as multilayer perception and the learning is made possible using back propagation.

In this article it is presented only the communication client–server using sockets to transfer the parts (Figure 1).

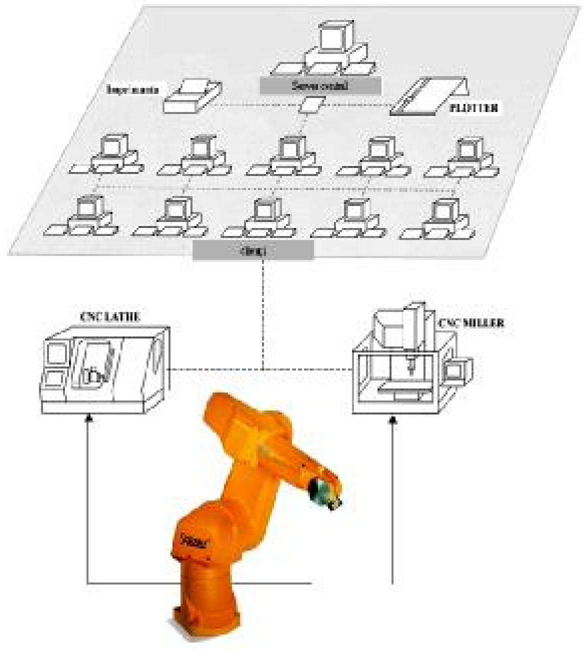

Fig. 1. Example of architecture for a distributed type of client – server

The “client” program contains three classes: „client”, „network” and „writer”. Initially, the program imports three packages: „java.io.*” (used for input/output actions; io = Input / Output), „java.awt.*” (used for graphical objects; awt = Abstract Window Toolkit) and „sun.net.*” (used to establish a connection between client and server).

In order to run the client program, the user will type the following command:

java client <IP or address> <ClientName>

where:

java – the name of the application that launch the classes;

client – the name of the class that will be executed;

IP or address – IP address or server name;

ClientName – the name of the client who wants to connect to the server.

In the following sections will be explained the three classes of the client application.

A. The „client" class

The „client" class extends the *Frame* class, which has a method called *add.* Using this method we can insert in the middle of the frame an object of type *network* [1].

The class contains two methods: *main* and *client.*

In the *main* method it is verified if the number of arguments is smaller than two. The arguments meaning is:

- args[0] – the IP or the server name;
- args[1] – the name of the client who wants to connect to the server.

This function is the first that's executed when the *client* class is started.

In this method it is created an instance of the *client* class, having as parameters the server IP and the client's name.

The *client* method receives as parameters the name or the IP address of the server (*String host*) and the user name (*String username*). Inside the frame we insert a new object, which is in fact a new instance of the *network* class, which has as parameters the same values that this function receives. Then, the frame size is set to (500,500). The method *show()* displays the frame on the main screen.

B. The „network" class

This class extends the Panel class [2]. The meaning of the variables is the following:

public static boolean afis – if it is true, then the messages that are received from server are shown in the green box from the *client*'s main window;

NetworkClient network_client – using this method, we can declare a client. We use the specifications from the sun.net.* package;

DataInputStream net_input – using this variable, the client can receive messages from server;

PrintStream net_output – using this variable, the client can send messages to the server;

static int modif=0 – if it is 1, then the image received from server is shown in the client's main window. This image is called "imagine.gif". If it is 0, then this image isn't shown in the window.

String username – user name;

boolean connected=false – if it is **true**, this means that the client is connected to the server;

writer w – using this variable, we can create a new instance of the writer class, which handles with the read of characters or strings received from server;

String typed_line="" – the message that's typed by client.

In the following sections it will be presented the methods of this class.

C. The class constructor.

The constructor receives two parameters, which represents the name or the IP address of the server and the user name. Then we try to connect to the server, by calling the connect method. If this connection succeeded, the connected will be set on true and also a new instance of the writer class will be created. If the connection doesn't succeed, an error message will be printed on the client's console and the execution of the application will be ended.

The connect method. This method tries to connect to the server, which must be open for this communication on the port 1111. This port can be changed – if it's already associated with another application – but in this situation this number must be changed in the client application and also in the server program. The connection is realized by creating a new instance of the class NetworkClient, which can be found in the sun.net.* package. The arguments of this constructor are the name or IP address of the server and the port number on which we try to establish a communication. If the server is open, thing that's verified by calling the method serverIsOpen, a message will be displayed. This message tells to the user that the connection succeeded. Then, the net_input and net_output variables are initialized with instances of the classes that correspond to their definition. The serverOutput variable is declared inside the NetworkClient class and it is initialized when the constructor of the NetworkClient class is called.

The read_net_input method. This method reads a line from server and returns the string received or null in case that an input/output error occurs. It also returns null if there is nothing available from the server.The write_net_output method. This method writes a string to the server.

The close_server method. This method tries to close the communication between client and server.

The keyDown method. This method is executed when a key is pressed in the client's window. If this key is backspace (so the ASCII code is 8), the last character in the string will be deleted. If the key is ESCape (with ASCII code 27) the application will try to close the communication with server and also to exit the program. Otherwise, in the string that's typed will be inserted the key that was pressed. Also, it will be called the repaint method in order to make a quick repaint of the client's window.

The paint method. This method paints the client's frame. If the writing of messages from server is permitted, it will be written the text „Hello, <UserName> !", then it will be drawn the two boxes in which the text will appear (one is red and the other is green). In the box from the top of the windows (the red one) will be shown the data typed by client and in the other box will be shown the messages received from server. If the modif variable is set, then the image imagine.gif will be shown. This image is received from the server.

D. The „writer” class

This class handles with the reading of data from server. This method extends the Thread class, which is specific for repeated actions. If the keyword “RECV” is received from server, the variable will become true. This string specifies that a data transaction will begin between server and client. This transmission represents the bytes of a “.gif” image. Then the „SIZE nnnn” string is received. This string tells to the application what size the file has. Then we’ll receive nnnn character codes (ASCII code as a 2 character string, so this is the code in hexadecimal). The data are decoded and written in the file imagine.gif. Data are received since the “STOP” string appears. In this case the modify variable from network class will become 1 and the repaint method from the same class will be called.

E. The “server” program

The server program must be started before the users are trying to connect. If the server isn’t started, then the users that want to connect to it will show an error message on their consoles.

Calling the following instructions makes the start of the server class:

for Windows Operating System (9x, Me, 2000, NT, XP):

java server

A complete syntax includes the definition of the path in which the *java* program can find the classes (“.class” files). It’s possible that the environment variable CLASSPATH shall not be defined or shall not include a reference to current directory (specified in this OS by “.” (point)). The complete syntax is:

java –cp %CLASSPATH%;. server

in UNIX based OS:

java –cp $CLASSPATH;. server &

where $CLASSPATH represents the environment variable that contains the path(s) where the “.class” or “.jar” files should be searched.

The symbol “&” specifies to the UNIX OS that this process must be executed in background. This process will be waked-up when something happens on the port 1111. In this way we can realize the multitasking in UNIX.

The server application contains two classes.

The server class extends the NetworkServer class, which is a part of “sun.net” package. In this class we consider that the maximum number of users is 1,000. The meaning of the variables from the server class is the following:

- *String user[]* – this array will contains the name of the users that are currently connected to server;
- *String sir[]* – in this array will be stored the last message from each user;
- *DataInputStream net_input[]* – by using this array, the server can receive data from clients;
- *PrintStream net_output[]* – by using this array the server can send information and data to cliens;
- *reader r[]* – in this vector will be stored the addresses of the „reader” class instances;
- *static int client_counter=0* – the current number of connected clients.

The *main* method creates a new instance of the server class.

The *constructor* of this class initializes the arrays presented above. Then the server is started, by calling the startServer(1111) instruction. This instruction starts the server, specifying that the port on which the server will receive message is 1111. This port number can be changed, but if is changed in the server class it also has to be changed in client class.

If the server was started successfully, a message is printed on the server console:

Waiting for users …

If it’s not possible to open the server on the port 1111, then an error message will be shown and the execution of this application will be canceled.

The most important method in this class is serviceRequest, which is a function of the NetworkServer class. When an event occurs at the server, this function is automatically called. In this function it is created an instance of the *reader* class. This thing happens each time when a client is connecting to the server.

The method read_net_input reads data from clients.

The method write_net_output writes data to clients.

The *reader* class

This class extends the Thread class. The most important method of this class is *run*. This method is called each time when an event occurs (connection of a client, the send of a message from a client, etc.). In the case when an end of sentence punctuation is signaled (point, !, ?), it is verified if the message represents the standard syntax for sending data (images). This syntax is:

SEND <ImageName> TO <user>

where *ImageName* represents the name of the image that’s stored on the server and the *user* is the name of the user that has to receive the image. Examples:

SEND image1 TO machine1.
Send image2 TO machine2.
The syntax is “case-insensitive”.

The server program can have the possibility to search the image and after her index in a PostgreSQL or MySQL database. Using the methods from the package java.sql we can make the interrogation of the databases.

Window capture screen and test pieces are showed in Figure 2.a, Figure 2.b and Figure 2.c.

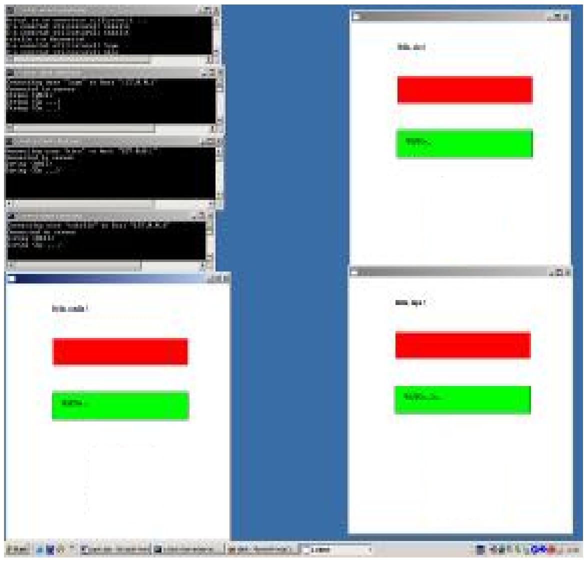
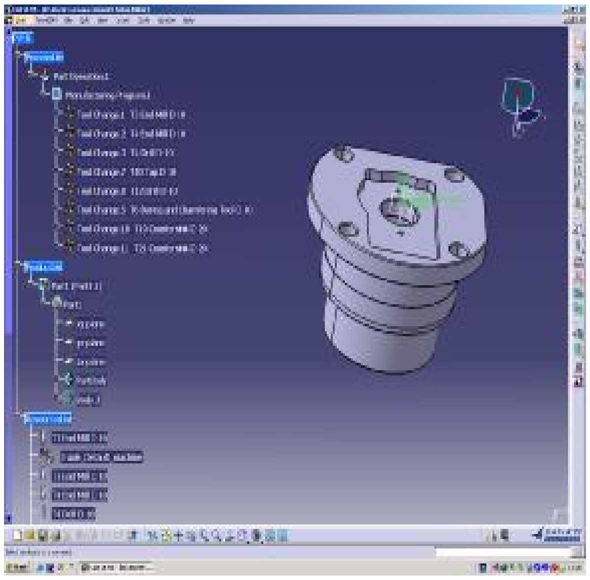
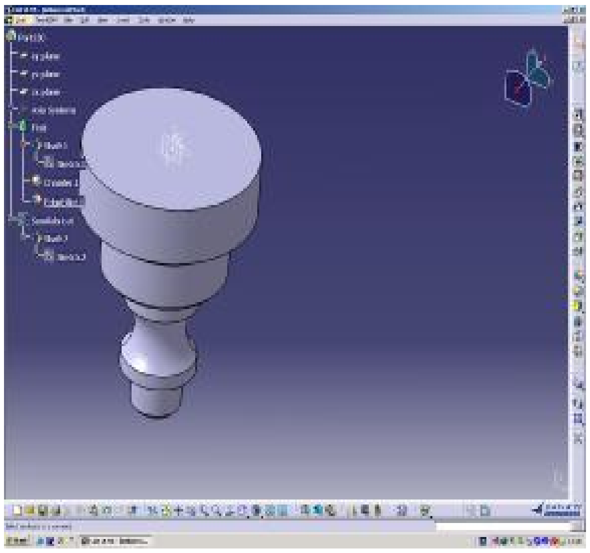
Fig. 2: a) widows screen; b) picture 1 c) picture 2

One or more clients can access the server to upload songs in the PostgreSQL database [1]. Server applications store customers to a queue of priorities. Access to the environment is made according to how he made the request.

The customer can upload programs for machine tool [3] with the leadership of robots or server can send this information in the database. The server can be accessed and transmitting site and the IP address of away. The server may be in Suceava and customers can, for example, be in Bucharest. The laptop (server) is connected with a mobile phone to connect with specialized satellite communications with customers place anywhere. Customers communicate with the server transmitting site, the IP address of the server and the client's behalf. In the event that a client has failed then the owner of client-server application can intervene and fix the fault. System administrator can intervene at any client to view the stage at which they are situated and to intervene in case of breakdown [3].

The following are programs for creating a database in PostgreSQL necessary to describe the parts and communication program for machine tool or robot, the communication in the client - server (the client and the server).

Software client-server provides outstanding economic effects on reducing energy consumption, the price of the entire system cost and the area occupied by the distributed system.

Among the advantages of using software in client-server monitoring and command lines of flexible metal coating can be listed [6]:

- Traceability and registration the operation of the flexible line, even where it is in the operating manual;
- The possibility that a single operator, in run semiautomatic, to order all robots manipulator and carts the flexible line;
- Ensuring the flexibility of the real line, to run automatically, by the admission of any series of types of pieces to the load.

Of all the programs for computer-assisted design, whatever they are CAD, CAM and CAE, CATIA holds supremacy in Chapter opportunities (Figures 3-9) [4, 5].

Among the possibilities are CATIA remember:

- constructive possibilities:
- 2Ddesign;
- 3D design;
- Ansambles
- Achievement;
- Design parts of the board;
- Designing mechanical structures;
- Possibilities for designing equipment, and industrial
  - o networks:
  - o electric networks;
  - o hydraulic;
  - o technologicalϖ possibilities:
    - § finite element analysis
    - § simulation mechanisms;
    - § manufacturing

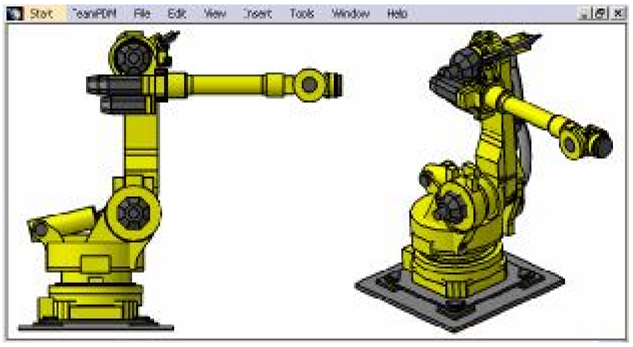
Fig. 3: The design and programming robots.

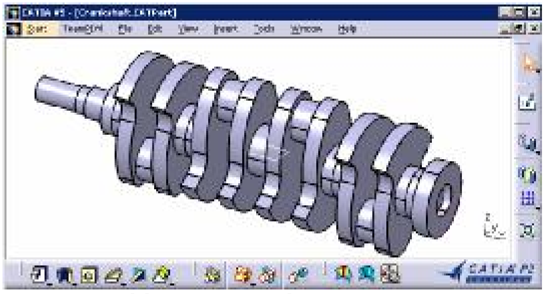
Fig. 4. Modeling a tree meandering

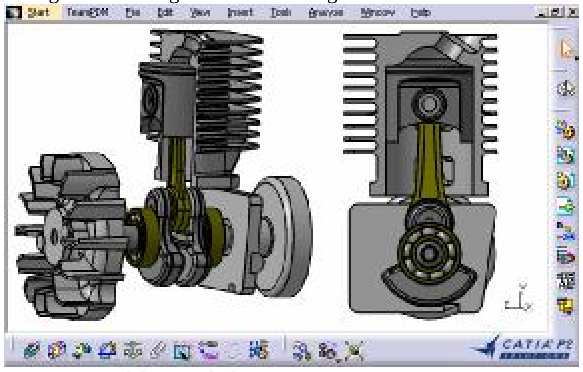
Fig. 5. Modeling a whole

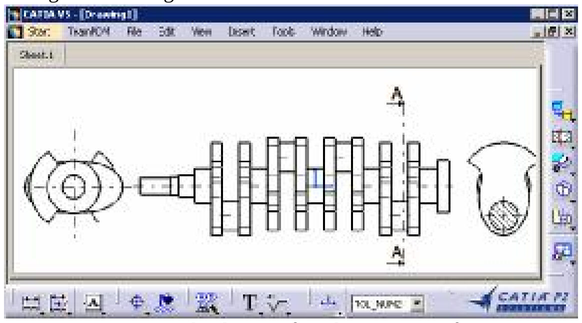
Fig. 6: Extracting the design of implementation of sinuous tree mode

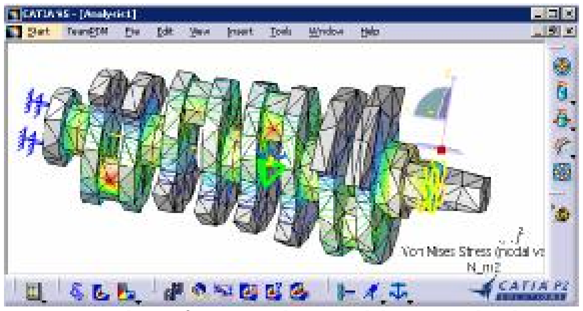

Fig. 7. Analysis of tree meandering with how specialized FEA.

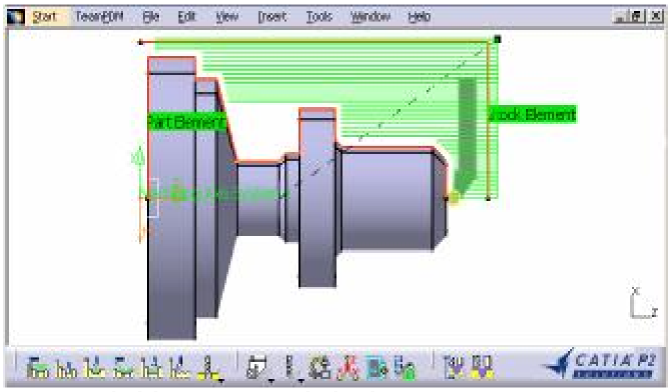

Fig. 8: Schedule processing a cylindrical pieces.

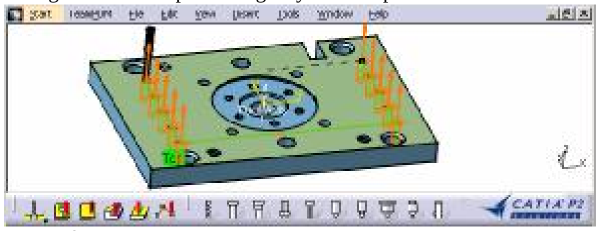
Fig. 9: Programming processing prismatic parts.

II. Conclusion

In this paper, we propose and experimentally evaluate the use of the client-server database paradigm for real-time processing. To date, the study of transaction processing with time constraints has mostly been restricted to centralized or “single-node” systems. Recently, client-server databases have exploited locality of data accesses in real-world applications to successfully provide reduced transaction response times. Our objective is to investigate the feasibility of real-time processing in data-shipping client-server database architecture. We compare the efficiency of the proposed architecture with that of a centralized real-time database system. We discuss transaction scheduling issues in the two architectures and propose a new policy for scheduling transactions in the client-server environment. This policy assigns higher priorities to transactions that have a greater potential for successful completion through the use of locally available data. Through a detailed performance scalability study, we investigate the effects of client data-access locality and various updating workloads on transaction completion rates. Our experimental results show that real-time client-server databases can provide significant performance gains over their centralized counterparts. These gains become evident when large numbers of clients (more than 40) are attached per server, even in the presence of high data contention.

References

[1] S. Lalani, K. Jamsa, „Biblioteca programatorului JAVA”, 1998
[2] Java on-line documentation (http://java.sun.com).
[3] V. Lupu, “Tehnologii informatice moderne utilizate pentru conducerea sistemelor flexibile de fabricaţie” (teza de doctorat), 2004, EDP Bucureşti.
[4] Autocad ****Autocad , www.autodesk.com, aprilie 2003.
[5] CATIA **** www.3DS.com, aprilie 2003.