12. Y. F. Zhang, Acceleration compensation for biped robots to reject external disturbance. IEEE Transaction on System, Man, and Cybernetics, 2006, 10, p.p. 106-116.
13. S. Yashika, W. Ryujin, A. Chiaik. The intelligent ASIMO: Systems Overview and integraton. International Journal Intelligent Robots and Systems, 2013, 22, p.p. 1455-1469.
14. B. Espiau, M. Celaiappevos. A Joint project for the development of an anthropomorphic Biped robot. International Advanced Robotics, 2011, 8, p.p.1-12.
15. J. Y. Wong, C. F. chiang. A general theory for skid steering of tracked vehicles on firm ground. Mechanical Engineers, 2013, 26, p.p.343-356.
16. H. T. Yaislalles. Analysis of carrier robot vehicle grade ability. Mechanical & Electrical Engineering, 2013, 20, p.p.49-56.
17. Q. Huang, Y. Nakamura. Sensory reflex control for humanoid walking. IEEE Transaction on Robotics Automation, 2005, 21, p.p. 977-984.

Research on the Distributed Big Data Management of Bank Based on the Three-tier C/S model

Jiqin Jiang*

Chongqing City Management College, Chongqing, 401331, China

Abstract
Designing and implementing an effective three-tier client/server system to support the new business paradigm is the focus of this document. To fully realize the potential of the three-tier architecture, this paper makes the middle layer easily accessible to multiple front-ends. The middle tier is running on a separate machine. This allows it to be accessible to multiple clients and it provides for the maximum performance. A prototype system was built using the business transaction of a certain Commercial Bank as the case studies. The experiment result shows that the three-tier client/server system has better performance in distributed big data management.
Keywords: DISTRIBUTED BIG DATA MANAGEMENT, THREE-TIER C/S MODEL, DISTRIBUTED SYSTEM FOR BANK.

1. Introduction

The term client/server originally applied to software architecture that described processing between two programs: an application and a supporting service. At that time, the client program and the server program did not have to be physically separated – they could be calling and called programs running on the same computer. Thus, originally, the client/server

discussions were limited to interactions between one client and one server [1]. In the 1980s, the term client/server was used in reference to personal computers (PCs) on a network. The actual client/server model started gaining acceptance in the late 1980s. The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability, and scalability as compared to centralized, mainframe, time sharing computing [2].

A client is defined as a requester of services and a server is defined as the provider of services. A single machine can be both a client and a server depending on the software configuration. The client/server computing model implies a comparative processing of requests submitted by a client, or requester to the server, which processes the requests and return the results to the client. The client/server-computing model covers a wide range of functions, services, and other aspects of distributed environment. Relevant issues include local and wide area networking, data distribution and replication, distributed processing, transaction management, software distribution and system management, data warehousing, middleware, standards and open systems. This section briefly discusses the evolution of distributed systems [3].

By legacy system we mean such a machine as a mainframe or minicomputer. With mainframe software architectures all intelligence is within the central host computer. Users interact with the host through a terminal that captures keystrokes and sends that information to the host. Mainframe software architectures are not tied to a hardware platform. User interaction can be done using PCs and UNIX workstations. A limitation of mainframe software architectures is that they do not easily support graphical user interfaces or access to multiple databases from geographically dispersed sites. In the last few years, mainframes have found a new use as a server in distributed client/server system.

2. The Main Framework and Theory Foundation

Host-based application processing is performed on one computer system with attached unintelligent, "dumb" terminals. A mainframe with attached character-based display terminals is an example of the host-based processing environment. From an application processing point of view, host-based processing is totally non-distributed. Figure 1 shows host-based processing environment.

Here the slave computers are attached to the master computer and perform application processing-related functions as directed by their master. Application processing in a master-slave environment is somewhat distributed, even though the distribution of the processing tends to be unidirectional-from the master computer to its slaves. Figure 2 shows master-slave processing environment [4-5].

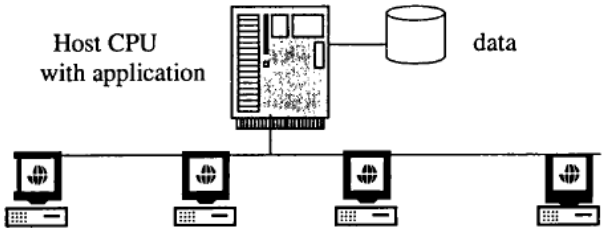

Figure 1. Host-based processing environment

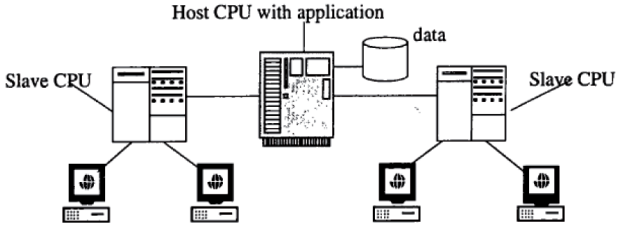

Figure 2. Master-slave processing environment

Typically, slave computers are capable of some limited local application processing, such as on-screen field validation, editing, or function-key processing. An example of this is a mainframe (host) computer such as the IBM 30XX, used with cluster controllers and intelligent terminals.

With this system, PCs are attached to a system device that allows those PCs to share a common resource-a file on a hard disk and a printer is typical examples. Such shared devices are called servers (a file server or a printer server in our example). These shared-devices are used to receive to receive requests for service from the PCs for generic, low-level functions. In a typical LAN based, shared-device processing, theses PC requests are usually limited to services related to shared file or print processing. The drawback of such an approach is that all application processing is performed on individual PCs, and only certain functions (print, file I/O) are distributed. Therefore, an entire file has to be sent to a PC that issues a READ request against this file. If a file has to be updated, the PC that issued the update request locks the entire file.

The shared-device system works if shared usage is low, update contention is low, and the volume of data to be transferred is low. In the 1990s, PC LAN computing changed because the capacity of the file sharing was strained as the number of online user grew

(it can only satisfy about 12 users simultaneously) and graphical user interfaces (GUIs) became popular (making mainframe and terminal displays appear out of date) [6]. Examples of shared-device system are Novell's NetWare or Microsoft's LAN Manager, which allow a LAN to have a system dedicated exclusively to file and/or print services [7]. Figure 3 shows shared-device processing environment.

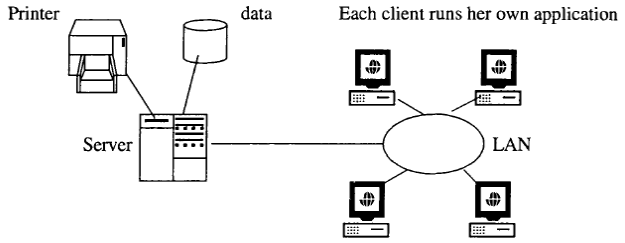

Figure 3. Shared-device processing environment

This system is a natural extension of shared device system. As LANs grew in size and number of supported workstations, the shared device system also grew in capacity and power. The main reason for the change was that, in a large LAN environment, sharing of file and print services among the workstations represented only a fraction of a typical application. The significant part of the functionality was also good candidate for sharing among LAN users. Therefore, some of the application processing was distributed to a new server-the server that receives that requests from applications running on workstation (clients) and processes them for each of its clients. In this system, application processing is divided (not necessarily evenly) between the client and the server. The processing is actually initiated and partially controlled by the service requester-the client [8].

Instead, both the client and the server cooperate to successfully execute an application. Database servers such Sybase SQL Server, Microsoft SQL Server, Oracle and so on are examples of the client/server system. Figure 4 shows shared-device processing environment.

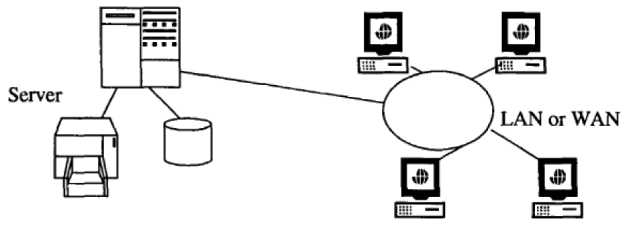

Figure 4. Shared-device processing environment

Architecturally, client/server processing requires:

Reliable, robust communication between clients and server

1 Client/server cooperative interactions that are initiated by a client

2 Application processing distribution between a client and its server

3 Server-enforced control over what services or data clients can request

4 Server-based arbitration of conflicting clients' requests.

3. Centralized Database Model

In the centralized model, the application processing components, the database software, and the database itself all reside on the same processor. For example, a PC user might run application programs that use Oracle database software to access a database residing on the PC's internal hard disk. Since the application components, the database software, and the database itself all reside on the same PC, the application conforms to the centralized model. Much of the mainstream information processing performed by many organizations still conforms to the centralized model. For example, a mainframe processor running IBM'S IMS or DB2 database software may provide terminal operators at widely distributed locations with fast access to a central pool of data. However, in many such systems, all three components of the database application execute on the same mainframe. Thus this configuration also conforms to the centralized model [9-10].

In the file-server model, the application processing components and the database software reside on one computing system, and the physical files making up the database reside on some other computing system. Such a configuration is often used in a LAN environment in which one or more computing systems play the role of file servers that store data files to which other computing systems have access. In the file-server environment, networking software is employed that makes the application software and the DBMS running on an end-user system think that a file or database stored on a file server is actually attached to the end-user machine. The files implementing the database reside on a different machine from the application components and the DBMS. Such an environment can be more complex than the centralized model because the networking software may implement concurrency mechanisms that permit more than one end user to access the same database without those users interfering with one another.

With the database extract processing model, the user, possibly at a PC, makes a connection with the

remote computing system where the desired data is located. The user then interacts directly with the software running on the remote machine and formulates a request to extract the required data from the remote database. The user then transfers the desired data from the remote machine to the user s own computer and hard disk. The user then processes the local copy of the data using local database software. With this approach, the user must know where the data is located and also how to access and interact with the remote computer that maintains the database. Complementary application software must be available on both computing systems to handle database access and the transfer of data between the two systems. However, the database software running on the two machines need not be aware that remote database processing is taking place, since the user interacts with them separately.

In the true client/server database model, the database again resides on a machine other than those that run the application processing components. But the database software is split between the client system that runs the application program and the server system that stores the database. In this model, the application processing components on the client system make requests of the local database software. The local database software component in the client machine then communicates with complementary database software running on the server. The server database software makes requests for accesses to the database and passes results back to the client machine. In this model, it is common to refer to front-end software and back-end software.

Front-End Software. This runs on a PC or desktop workstation and serves the computing needs of a single individual [11]. The front-end software typically plays the role of the client in a client/server database application and performs functions that are oriented to the needs of the end user. Front-end software generally falls into one or more of the following categories: End-User Database Software, Simple Query and Reporting Software, Data Analysis Software, Application Development Tools, and Database Administration Tools.

Back-End Software. Back-end software consists of the client/server database software and network software that runs on a machine playing the role of database server.

The file-server model and the client/server database model both assume that the database resides on one processor and that the application program that accesses the database resides on some other processor. The true distributed database model assumes that the database itself can reside on more than one machine.

Traditionally, most large organizations developed applications to reside on central, mainframe computers, which also house the data those applications are to access. However, the use of large, centrally located computers for such centralized application processing became more and more expensive, especially when compared with the price-performance advantages of microcomputers. Today, businesses are becoming more and more interested in implementing distributed databases. Some of the reasons for doing this include:

Businesses' desire to reduce operating costs and to decentralize operations in order to be more competitive and more responsive to Customer demands

The advances in the area of distributed and client/server computing.

The availability of enabling technology for distributed processing, including products implementing distributed databases in client/server architecture.

Given the ever-pressing need to ensure high data availability, integrity, and correctness, distributed data management poses a challenge to developers and database administrators alike. The main purpose of distributed data management is the ability to access data distributed to multiple sites (network nodes) in a fashion transparent to end-users. Given the complexity and relative inefficiency of existing solutions for providing end-user data access to the (often remote) locations at which data originates, distributed data management also has the responsibility of keeping most data local to the sites that actually use it. The distributed data management environment is characterized by the two- way distribution:

1. The data and DBMS are distributed among multiple nodes, including the node with the application logic.

2. Data-management functions are distributed between a front-end system (data manipulation logic and language [DML]), and the back-end server (database functions [DBMS]).

Distributed data management deals with data distributed among multiple nodes. Distributing data among multiple sites offers the following benefits:

1. Placement of data closer to its source wherever appropriate

2. Higher data availability by placement of multiple copies of critical; data at different locations, thus eliminating a potential single point of failure and enforcing local autonomy

3. More efficient data access, thus improving data-management performance

4. Application load balancing as it relates to data access

5. Facilitation of growth in applications and end-user demands

There are, of course, some drawbacks to distributing data. Among them are the complexity of the distributed data management and a relatively high potential for the loss of data synchronization. In fact, the decision on centralized versus distributed data is largely dominated by the degree of data sharing and local autonomy required by an organization-the higher the degree of data sharing, the more attractive is the option that includes centralized data storage. Obviously, centralized data is easier to monitor, manage, and control. However, organizations, which are geographically dispersed, and require a high degree of local autonomy, cannot be satisfied with centralized data solutions. Then data distribution becomes a viable alternative.

In the management system, linear equation can be expressed into the following simplified forms:

$$L(\nabla,\omega)f(x,\omega)=0,$$
$$L(\nabla,\omega)=T(\nabla)+\omega^2\rho\mathrm{J} \quad (1)$$

In which,

$$T(\nabla)=\begin{Vmatrix} T_{ik}(\nabla) & t_i(\nabla) \\ t_k^T(\nabla) & -\tau(\nabla)\end{Vmatrix},\ \mathrm{J}=\begin{Vmatrix}\delta_{ik} & 0\\ 0 & 0\end{Vmatrix},$$

$$f(x,\omega)=\begin{Vmatrix}u_k(x,\omega)\\ \varphi(x,\omega)\end{Vmatrix} \quad (2)$$

Consider delay, the L can be expressed as:

$$L^0=\begin{Vmatrix}C^0_{ijkl} & e^0_{kij}\\ e^{0T}_{ikl} & -\eta^0_{ik}\end{Vmatrix} \quad (3)$$

These functions can be expressed in the following form:

$$C(\mathrm{x})=C^0+C^1(\mathrm{x}),\ e(\mathrm{x})=e^0+e^1(\mathrm{x}),$$
$$\eta(\mathrm{x})=\eta^0+\eta^1(\mathrm{x}),\ \rho(\mathrm{x})=\rho_0+\rho_1(\mathrm{x}) \quad (12)$$

The value with superscript of 1 represents the difference below:

$$C^1=C-C^0, e^1=e-e^0,$$
$$\eta^1=\eta-\eta^0, \rho_1=\rho-\rho_0 \quad (4)$$

The whole function can be simplified into the following integral equation set:

$$f(x,\omega)=f^0(x,\omega)+\int_V S(x-x')(\mathrm{L}^1F(\mathrm{y}')$$
$$+\rho_1\omega^2\mathrm{g}(R)\mathrm{T}_1f(\mathrm{y}')]S(\mathrm{y}')\mathrm{dy}' \quad (5)$$

In addition, we can introduce the abbreviated formula:

$$g(x,\omega)=\begin{Vmatrix}G_{ik}(x,\omega) & \gamma_i(x,\omega)\\ \gamma_k(x,\omega) & g(x,\omega)\end{Vmatrix},$$

$$s(x,\omega)=\begin{Vmatrix}G_{ik,l}(x,\omega) & \gamma_{i,k}(x,\omega)\\ \gamma_{k,l}(x,\omega) & g_{,k}(x,\omega)\end{Vmatrix},$$

$$L^1(x,\omega)=\begin{Vmatrix}C^1_{ijkl} & e^1_{kij}\\ e^{1T}_{kij} & -\eta^1_{ik}\end{Vmatrix},$$

$$F(x,\omega)=\begin{Vmatrix}u_{(i,j)}(x,\omega)\\ \varphi_{,\ i}(x,\omega)\end{Vmatrix} \quad (6)$$

In these expression, $G_{ik}(x,\omega)$, $\gamma_i(x,\omega)$, $g(x,\omega)$ can be represented as:

$$g(x,\omega)=\frac{1}{(2\pi)^3}\int g(k,\omega)\exp(-i\mathrm{k_g x})d\mathrm{k} \quad (7)$$

$$g(k,\omega)=\begin{Vmatrix}G_{ik}(k,\omega) & \gamma_i(k,\omega)\\ \gamma_k^T(k,\omega) & g(k,\omega)\end{Vmatrix} \quad (8)$$

Where $G_{ik}=(\Lambda_{ik}+\frac{1}{\lambda}h_ih_k^T)^{-1}$,

$g=-(\lambda+h_i^T\Lambda_{ij}^{-1}h_j)^{-1}$, $\gamma_i=\frac{1}{\lambda}h_k^TG_{ki}$.

4. Experiment Results

This provides a blueprint for coding and gives glimpse of the basic procedure of the software. Figure 5 shows flowchart for the transaction subsystem and figure 6 shows flowchart for the accounts creation subroutine. Figure 6 shows flowchart for the accounts creation subroutine and figure 7 shows flowchart for the loans subroutine.

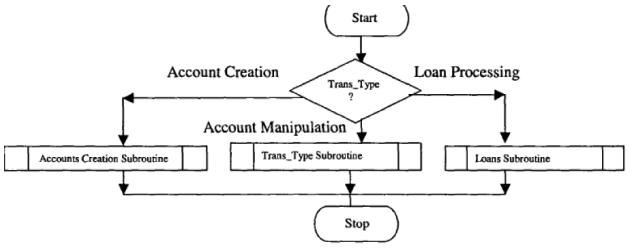

Figure 5. Flowchart for the Transaction Subsystem

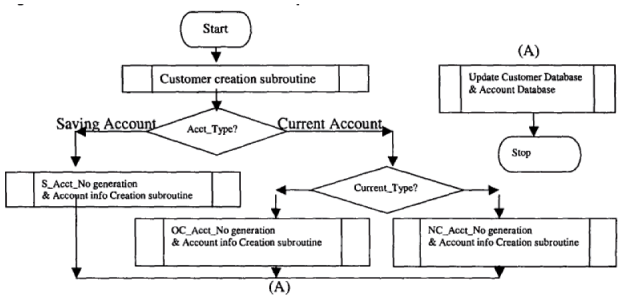

Figure 6. Flowchart for the Accounts Creation Subroutine

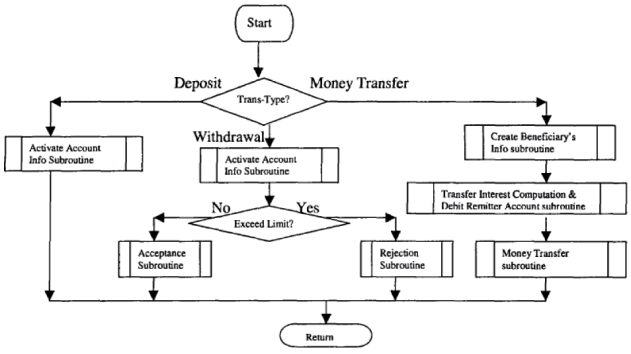

Figure 7. Flowchart for the Account Manipulation Subroutine

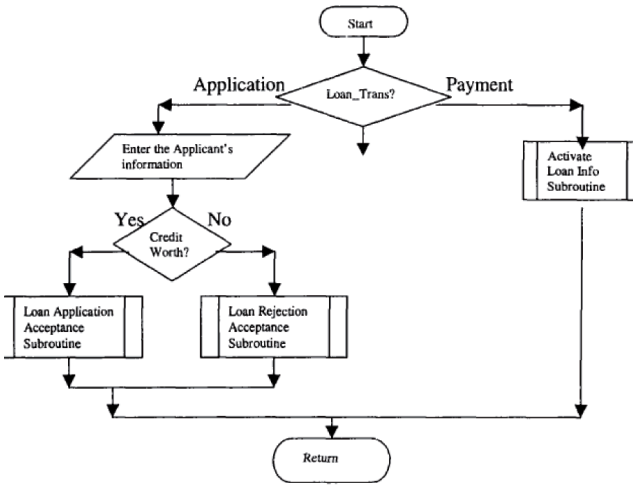

Figure 8. Flowchart for the Loans Subroutine

The comparison before and after using three-tier client/server system in task scheduling algorithm can be seen from figure 9. The result shows that in the same experimental time, after using three-tier client/server system it can achieve better performance in calculating time than before using it.

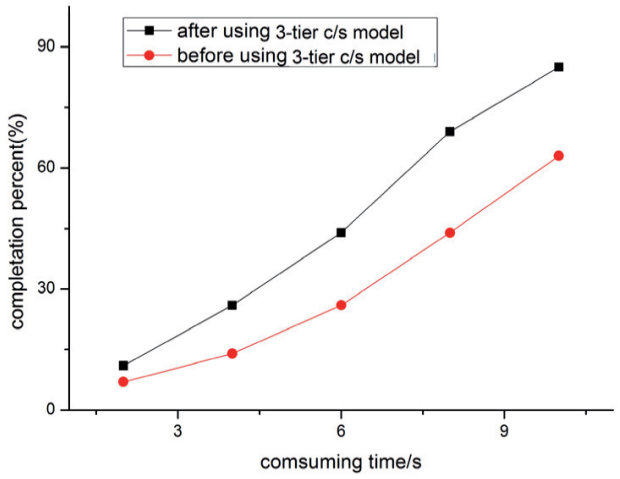

Figure 9. The comparison before and after using fractal algorithm in calculating time

The accuracy comparison before and after using three-tier client/server system can be seen from figure 10. The result shows that in the same experimental time, after using three-tier client/server system it can achieve better performance in accuracy than before using it.

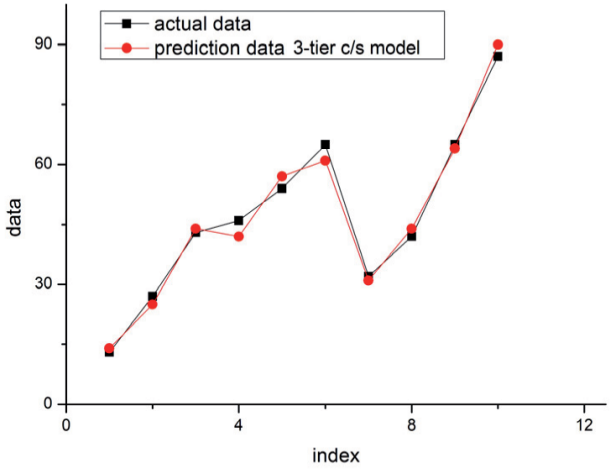

Figure 10. The comparison before and after using fractal algorithm in accuracy

5. Conclusion

One of the great truisms of application development is that there are many ways to do everything. One of the hardest things to do when building an application is to decide on an approach and to know if it is better than the other approaches. The client/server application developer's dilemma lies in knowing how to split up the work between the front and back-ends. However, with structured programming techniques and proper software engineering guidelines can help separate these components. Moreover, proper use of object-oriented techniques, especially encapsulation and inheritance, can facilitate these often-intuitive efforts.

Acknowledgements

This work is supported by Chongqing Municipal Education Commission scientific and technological research project, China (No.KJ1503209).

References

1. Mary C. Lacity,Leslie P. Willcocks,Ashok Subramanian. A strategic client/server implementation: new technology, lessons from history[J]. Journal of Strategic Information Systems, 1997, 62.
2. G.I. Papadimitriou, A.I. Vakali, A.S. Pomportsis. A learning-automata-based controller for client/server systems[J]. Neurocomputing, 2003, 61.

3. Jin Hyun Son, Myoung Ho Kim. An analysis of the optimal number of servers in distributed client/server environments[J]. Decision Support Systems,2002, 363.
4. Shiyuan Wang,Divyakant Agrawal,Amr El Abbadi. Towards practical private processing of database queries over public data[J]. Distributed and Parallel Databases, 2014, 321.
5. Suchendra M. Bhandarkar, Lakshmish Ramaswamy, Hari K. Devulapally. Collaborative caching for efficient dissemination of personalized video streams in resource constrained environments[J]. Multimedia Systems, 2014, 201.
6. Jeff Offutt, Vasileios Papadimitriou, Upsorn Praphamontripong. A case study on bypass testing of web applications[J]. Empirical Software Engineering, 2014, 191.
7. Heyoung Lee, Heejune Ahn, Samwook Choi, Wanbok Choi. The SAMS: Smartphone Addiction Management System and Verification[J]. Journal of Medical Systems, 2014, 381.
8. Zhi-Wei Xu. Cloud-Sea Computing Systems: Towards Thousand-Fold Improvement in Performance per Watt for the Coming Zettabyte Era[J]. Journal of Computer Science and Technology, 2014, 292.
9. G. L. Laing, J. L. Bruce, D. L. Skinner, N. L. Allorto, D. L. Clarke,C. Aldous. Development, Implementation, and Evaluation of a Hybrid Electronic Medical Record System Specifically Designed for a Developing World Surgical Service[J]. World Journal of Surgery, 2014, 386.
10. Bulent Tugrul, Huseyin Polat. Privacy-Preserving Inverse Distance Weighted Interpolation[J]. Arabian Journal for Science and Engineering, 2014, 394.
11. Manfred F. Buchroithner, Guido Ehlert, Bernd Hetze, Horst Kohlschmidt, Nikolas Prechtel. Satellite-Based Technologies in Use for Extreme Nocturnal Mountain Rescue Operations: a Synergetic Approach Applying Geophysical Principles[J]. Pure and Applied Geophysics, 2014, 1716.

Research on the Improvement of Proactive Workload Management Based on the Multi-Objective Genetic Algorithms

Guangqing Shan*

Chongqing City Management College, Chongqing, 401331, China

Abstract

In this paper, the author researched the improvement of proactive workload management based on the multi-objective genetic algorithms. The authors were irresolute between building a new simulation environment or utilizing an existent one. The answer for this question was to find a suitable simulation which provide most of the