LOAD BALANCING FOR CORPORATE SOFTWARE SERVICES

AJMAL BEG

The concepts described in this publication are based on European Patent Application EP1564637

ISBN: 978-0-9805610-2-9

Dedicated to my family

Introduction

Use of distributed computing for providing software services has become quite common in large corporations due to availability of low-cost computing hardware and scalable software architectures. It is quite common to have large number of application servers in combination with single or multiple database servers to provide software services to large number of internal and external users. For distributing software services requests execution load among different servers, load balancers are employed. This book introduces an improved method of assigning software service requests to servers using past load characteristics stored in persistence storage for better utilization of the available hardware resources and thus, reducing the execution time of software service request. This method of assigning service request uses categorizes to characterize the software service execution load data and the capacity of the servers executing such service requests, thus empowering an average system administrator to handle complex software services infrastructure.

Chapter 1

Need for better load balancing methods

Distributed computing has become quite common due to availability of low cost servers in the market and also due to the general availability of the software which can run on multiple application servers and/or database servers [[1],[3],[5],[7],[10],[13],[20],[21],[23]]. Many small and large size companies use ERP to provide software services to their employees and customers [[12],[16]]. An ERP in a large organization may have more than a dozen application servers, thus allowing reducing server hardware cost by utilizing low cost server hardware available in the market. Distributed computing also helps increase the availability time of the system as in case of failure of a server, the software service requests can be diverted toward other servers providing similar software services [[6]].

Servers hosting corporate systems provide different software services to internal and external users that contribute to a variety of business applications. The services range from email services to complex business transaction services [[2], [14]]. In a large corporate, large number of users log on, start or stop service and then logout. Some computer use internal services that correspond to the human user. For example, in a distributed system with an application sub-system and a database system, the human user communicates with the application sub-system, and the application sub-system communicates with the database system correspondingly through an intermediate service.

Each service places load on a server. Services are continuously added or removed so that the overall load goes up and down with the time. The system has to accommodate peak loads. In the system, a load balancer periodically determines the load on each server and shifts services between servers. The load balancer periodically determines the load on each server and shifts services between the servers [[8],

[22]]. The generally used load balancing policies are:

Policy A:

By counting the number of services that are executed on that particular servers.

Policy B:

By counting the human users that are logged on

The balancer assigns new services to a server using the above policies. Such policies work perfectly where there is almost equal load per service, the servers are of substantially equal capacity and there is not much difference in load from different users using the same service. However, this approach appears problematic in complex corporate landscapes, where there are multiple servers employed with different hardware capacities and load from software services varies from one type of service to another. Furthermore, due to the complexity of business in the large corporations different users using the same software service may generate different load depending on the business requirement of each user or department, thus making it difficult to assign the users software service request to an appropriate server.

To further explain the problem lets assume the following business scenario involving two servers providing monthly payroll calculations to two customers:

Available Server: S_A and S_B

Available Software Service on SA and SB: Monthly payroll calculation

Capacity of Server S_A: C_A

Capacity of Server S_B: C_B

Users of software service at S_A and S_B: U_A and U_B

Load from user $U_{A:}$ L_A (calculates payroll for 2500 employees)

Load from user U_B: L_B (calculates payroll for 5000 employees)

Let’s assume the case when:

$C_A < L_B < C_B$

$L_A < C_A$

Assume that the request for monthly payroll arrives from both UA and UB at the same time; the load balancer may assign LB to CA if it follows any of the policy A or policy B. Such software service assignment may create a bottleneck on server SA or it may lead to a situation where a monthly payroll is not completed in the allowed time window regardless of having hardware capacity available at another server.

The current software services assignment policies used in the corporations do not consider the time dependency of the workload that is created as a result of software execution on a server. Figure 1 illustrates such business scenario in which a server provides Material Requirement Planning Service to two different companies A and B. The first request from user A is a single step request which creates constant load on the server. The second request from user B is a three step software service request. During the first step execution the server is able to handle the load; while in the second step the total load from user A and user B exceeds the total capacity of the server. It may lead to a hardware bottleneck situation or material Requirement Planning request not being completed within the execution window for both companies.

In a computing on demand approach, any service should be available after the demand arises. Also, the server should reliably perform the service even if the service increases the load to the server, while execution. There is an ongoing need to alleviate these problems without manually planning and scheduling of the software services.

Chapter 2 in this book describes the improved method of load balancing which suits the core corporate software services. It further details the method to characterize the load and the capacity figures into a format which is easier to understand and use for system administrators. It further describes the flow of load balancing. Chapter 3 summarizes the improved method of software services assignments.

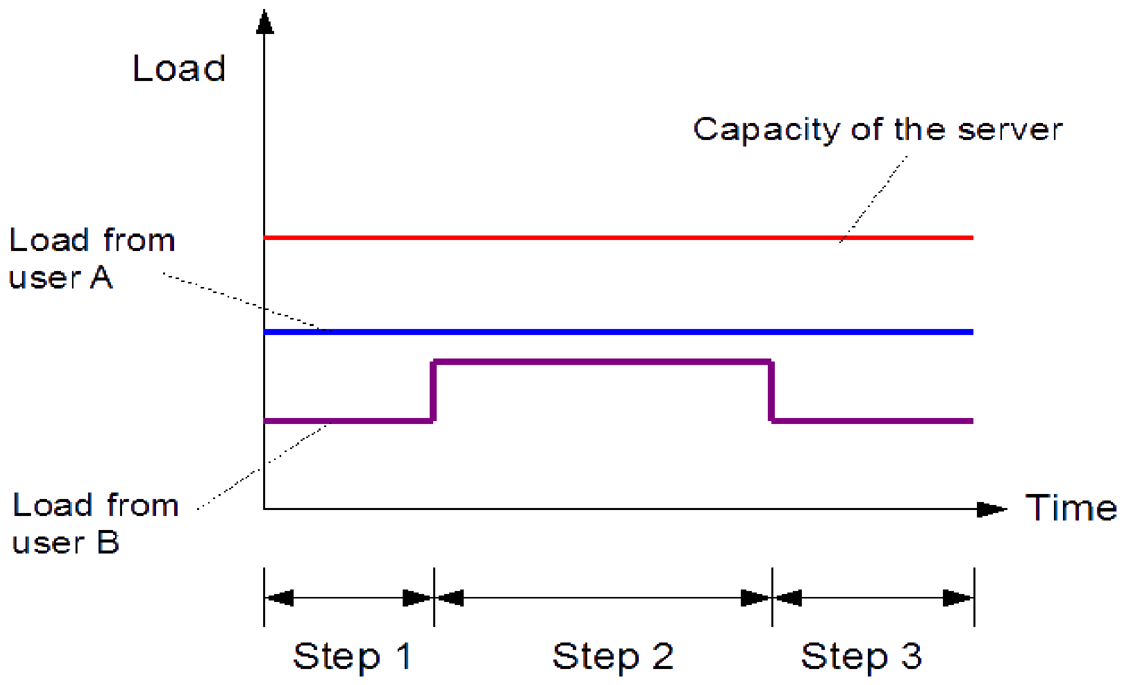

Figure 1 : Load from two independent users on the server

Chapter 2

Improved method of assigning software service

Some of the core software services in the corporate have a fixed pattern of consuming computing resources. For example, the software service which performs calculation of monthly payroll, consumes almost same quantity of the computing resources every month if there is no drastic change in the number of employees. Similarly, in the case of software service which calculates the material requirements for a specific assembly unit in a manufacturing corporation, there is generally no drastic unplanned change in the manufacturing output of the assembly lines. Load balancing is well explored area and different load balancing techniques have been proposed for different application areas with limited success [[3],[11],[15],[18],[19]]. In case of many core corporate software services, the computing resources required can be predicted quite accurately by collecting and utilizing the resource consumption history of the past. The computing capacity of each piece of hardware is also known factor. Utilizing this resource consumption predictions and the already known total capacity of the computing hardware, can help avoid inappropriate service assignment problem described in the introductory section.

Here, we describe which resource consumption parameters can be collected and used in case of 3-tier enterprise resource planning system. Figure 2 shows a typical Enterprise Resource Planning system. The persistence layer is a single database system or a cluster of multiple database instances storing large amount of business related information. The application layer is the layer which executes the business logic which uses the data stored in the persistence layer and consists of multiple application servers and a load balancer. The load balancer assigns the users to the multiple application servers. The presentation layer is the computing layer, which presents the calculated results from the business logic to the users in the corporate. The requests from the users are sent to the load balancer and the load balancer distributes these requests among the application servers. Persistence layer and the business logic layer both can have Performance Statistics Collectors to collect resource consumption indicators:

Database Time

Database time is the time that can be described here as the spent on read and write operation to the database. Database time can be observed and recorded on the database by observing the difference in the timestamp at which the SQL was received at the database and the timestamp at which the result of the SQL execution was sent back to the application server. Similarly the database time can be observed by the performance statistics collectors on the application servers by calculating the difference in the timestamp at which the SQL request was sent to the database and the timestamp at which the results were received. The network time can be excluded from this figure to have more accurate database time figure.

CPU Time

CPU time is the time that is spent on executing the business logic. The CPU time excludes the database time. The CPU time can be recorded at the business logic layer, where executing the database time from the time spent on executing the business logic.

Operation per time

The number of standard size services that can be performed simultaneously by the database or the application server. This parameter can be calculated by the Performance Statistics Collectors running on the application server or the database.

Memory Consumption

Memory consumption can be calculated by calls to the operating system by the Performance Statistics Collectors running on the database or the application servers.

Maximum Parallel Processing

The number of standard size processing that can be performed simultaneously by the database or the application server.

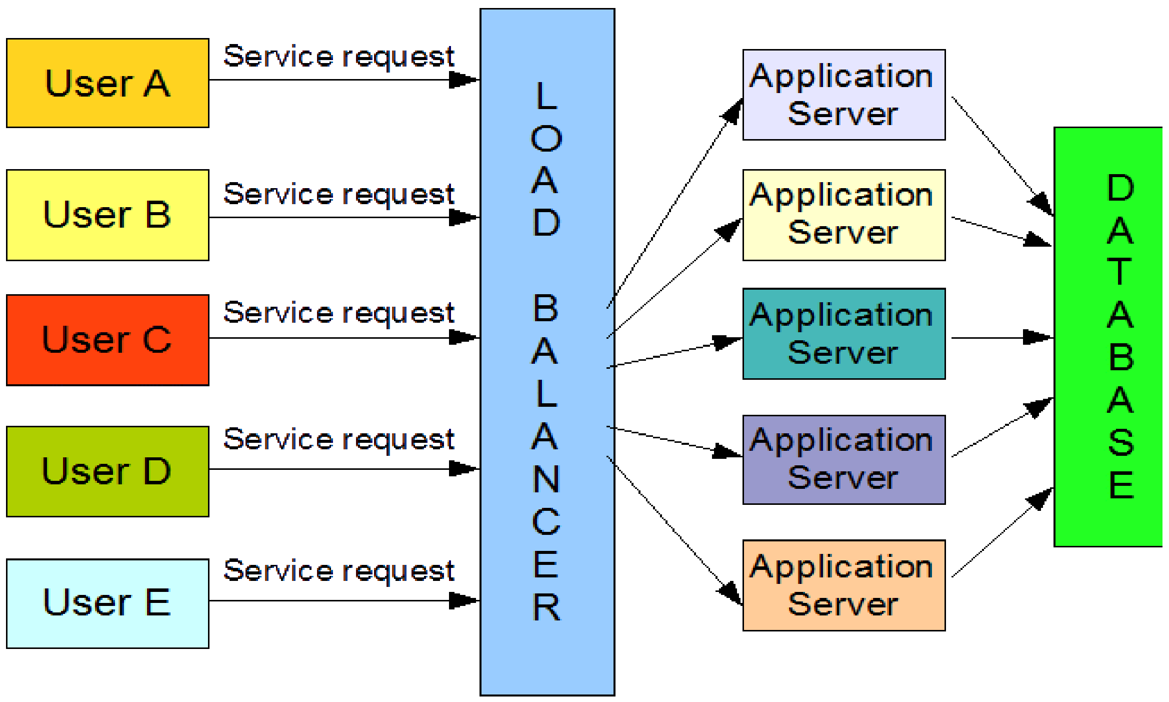

Figure 2: Typical Architecture of Enterprise Resource Planning System

2.1 Conversion of raw performance data into easy to use data

The performance data collected on the multiple application servers and database needs to be converted into easy to use and understand format for the operators of the software service provider. The balancer converts the data collected by the difference Performance Statistics Collectors in a useful form and stores in a record for easy access. The balancer uses categories to describe the load of the services. Categories such as LOW, MEDIUM, HIGH, VERY HIGH and MAX ease the processing. There is no need to process exact numerical values such as measurement values. For example, balancer categories those are product of a factor M with a standard quantity. The standard quantity is an absolute measurement term. For example, taking the category LOW as standard, Type MINIMAL corresponds to 1 LOW or 1000 operations per second; Type MEDIUM corresponds to 4LOW; Type HIGH corresponds to 8LOW; Type VERY HIGH corresponds to 40LOW; and TYPE MAXIMUM corresponds to 300LOW. The multiplication factors 1, 4, 8, 40 and 300 are chosen for the convenience of explanation, other factor can be used as well.

Re-categorizing load with time is advantageous for users known to potentially increase the service load. For example,

- Users of monthly payroll calculation service turn the load from LOW to HIGH category, as soon as the names of employee are selected for monthly payroll calculation and actual payroll calculations start.
-
- In the first step in Material Request Planning Service, where the list of material used in different plants is retrieved, the load is LOW. As soon as the material requests are calculated for a specific period of time, the load from user changes to HIGH category.

The capacity of the servers is also described using the above categories. It allows easy to use and understand capacity to load ratio during the load balancing.

2.2 Flow of software service assignments

Figure 3 shows the typical flow of the load balancing procedure:

- User initiates a service request and sends it to balancer
- Balancer obtains the user history of the load for this specific service from the record. The load history can be described using categories defined and stored in the database.
- Balancer obtains the current available capacity of the servers which is described using categories defined and stored in database.
- Balancer assign the service request to the server which has the capacity to perform the task
- Balancer subtracts the historical load of the service request from the available capacity record at the start of the service initiation.
- During the execution of the service, the actual load history is recorded for updating the characteristics of the load history of the user.
- At the end of the service execution the current available capacity record is updated again to record that the available capacity at the server has increased.

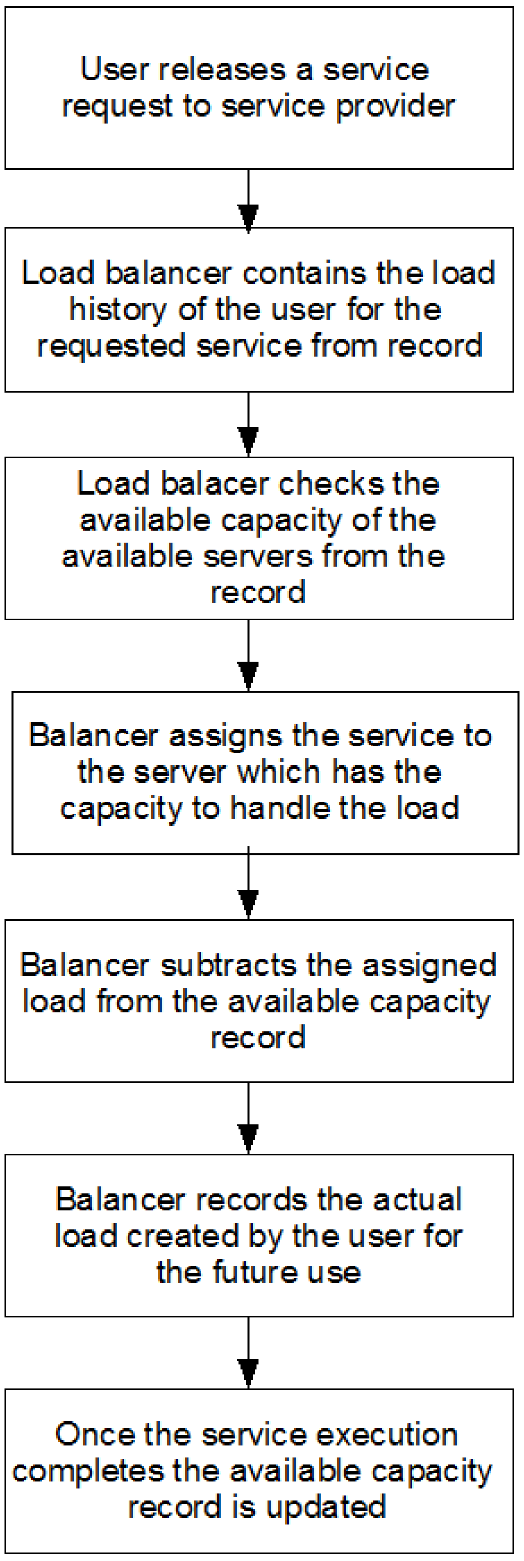
User releases a service request to service provider
Load balancer contains the load history of the user for the requested service from record
Load balacer checks the available capacity of the available servers from the record
Balancer assigns the service to the server which has the capacity to handle the load
Balancer subtracts the assigned load from the available capacity record
Balancer records the actual load created by the user for the future use
Once the service execution completes the available capacity record is updated

Figure 3: Flow of load balancing

The different types of information that can be used for implementing the described load balancing procedure is described in Figure 4.

User authentication data

This data is used to identify the users and decide whether to allow the access to the requested service or not based on the user authorization profile.

Server identification and capacity data

This data is used to determine which application servers are available for providing the services to the servers and what is their hardware capacity.

Services and their load characteristics data

This data describes which services are available on which application servers and what are the general load characteristics of each service.

Performance monitors configuration data

This data describes what type of performance monitor is configured on which server and how there performance collection parameters are configured.

Load categorization data

This data describes how to categorize different types of data in categories like HIGH, LOW and MEDIUM.

Currently available capacity and load

This data contains the information about how much of the available capacity has been assigned to which service request on each server. This data is updated at the time of service assignment and the completion of the assigned service.

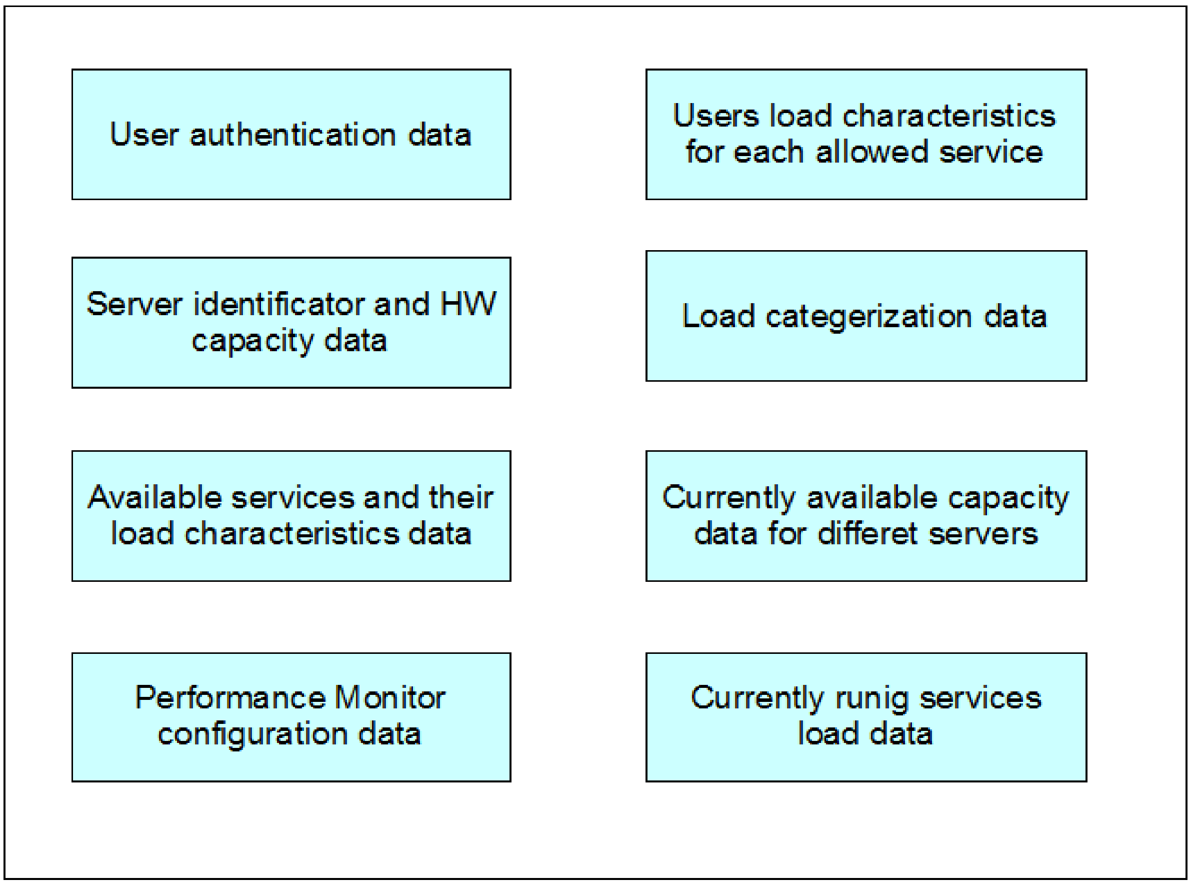
User authentication data
Users load characteristics for each allowed service
Server identificator and HW capacity data
Load categerization data
Available services and their load characteristics data
Currently available capacity data for differet servers
Performance Monitor configuration data
Currently runig services load data

Figure 4: Data used in the load balancing of the service requests

Chapter 3

Summary

This work introduces a load balancing mechanism for corporate service requests. The described load balancing mechanism specially suits the corporate environment where servers of different capacity are used for distributed computing and the load caused by the service varies from user to user. This work pays special attention to the fact that in the case of corporate services, the past load characteristics of the service requests from each user can be used to estimate the future load characteristics quite precisely. The described techniques release the system administrators from the arduous task of performing load distribution configuration manually. The usage of easy to understand categories for describing load and the capacity allows an average system administrator to handle large number of complex software services requests from users.

References

[1] A. G. Ganek, T. A. Corbi, “The dawning of the autonomic computing era”, IBM System Journal, 2003.

[2] A. Gouaich, “Distributed ubiquitous software services”, IEEE/WIC International Conference on Intelligent Agent Technology, 2003. , October 2003, Page: 531-534.

[3] A.Y. Zomaya, T. Yee-Hwei, “ Observations on using genetic algorithms for dynamic load-balancing”, IEEE Transactions on Parallel and Distributed Systems, Sep 2001, Vol 12(9).

[4] A. Umar, “Third Generation Distributed Computing Environment”, Nge Solution Inc., Chapter 2.

[5] D. C. Yen, D. C. Chou, “Intranet for organizational innovation”, “Information Management & Computer Security”, 2001.

[6] F. C. Gartner, “Fundamentals of fault-tolerant distributed computing in asynchronous environments”, March 1999, Vol 31(1), Pages 1-26.

[7] F. Kon, R. H. Campbell, M. D. Mickunas, K. Nahrstedt, F. J. Ballesteros, “2K: A Distributed Operating System for Dynamic Heterogeneous Environments”, Ninth IEEE International Symposium on High Performance Distributed Computing (HPDC-9 '00), Page 201.

[8] F. Kon, T. Yamane, C. Hess, R. Campbell, M. D. Mickunas,“ Dynamic Resource Management and Automatic Configuration of Distributed Component Systems“ , Proceedings of the 6th USENIX Conference on Object-Oriented Technologies and Systems (COOTS'01), January 2001.

[9] G. Bianchi, I. Tinnirello, “Proving load balancing mechanisms in wireless packet networks”, IEEE International Conference on Communications, 2002. ICC 2002, vol.2. Pages 891- 895.

[10] G. Lawton, “Distributed Net Applications Create Virtual Supercomputers”, Computer, June 2000, Vol 33(6), Pages 16-20.

[11] H. Xiao-Dong, L. Ruan, J. Sun, “Multicast routing, load balancing, and wavelength assignment on tree of rings”, IEEE Communications Letters, Feb 2002, Vol 6 (2), Pages 79-81.

[12] H. Klaus, M. Rosemann, G. G. Gable,“What is ERP?”, Information Systems Frontiers, August 2000, Vol 2(2), Pages:141-162.

[13] I. Foster, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of High Performance Computing Applications, 2001 Vol. 15(3), Pages 200-222.

[14] K.H. Bennet, J. Xu,” Software “Software services and software maintenance”, Proceedings of Seventh European Conference on Software Maintenance and Reengineering” , March 2003, Page: 3-12.

[15] K. M. Sim, W. H. Sun, “Ant colony optimization for routing and load-balancing: survey and new directions”, IEEE Transactions on Systems, Man and Cybernetics, Part A, Sept. 2003, Vol 33(5).

[16] L. Hayman, “ERP in Internet Economy”, Information Systems Frontiers, August 2000, Voll 2(2), Pages: 137-139.

[17] M. Cannataro, D. Talia, P. Trunfio,”Distributed data mining on the grid”, Future Generation Computing Systems, October 2002, Vol 18(8) Pages: 1101-1112.

[18] M. R. Pearlman, Z. J. Haas, P. Sholander, S. S. Tabrizi, “On the impact of alternate path routing for load balancing in mobile ad hoc networks”, International Symposium on Mobile Ad Hoc Networking & Computing archive, Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing.

[19] Narula-Tam, E. Modiano, “Dynamic load balancing in WDM packet networks with and without wavelength constraints”, IEEE Journal on Selected Areas in Communications, Oct 2000, Vol 18(10).

[20] T. J. Lehman, A. Cozzi, Y. Xiong, J. Gottschalk, V. Vasudevan, S. Landis, P. Davis, B. K. and P. Bowman, “Hitting the distributed computing sweet spot with TSpaces”, Computer Networks, March 2001, Vol 35(4), Pages 452-472.

[21] T. M. Keane, T. J. Naughton, “DSEARCH: sensitive database searching using distributed computing”, November 2005, Vol 21(8), Pages 1705-1706.

[22] T. Schneider, “SAP Performance Optimization Guide”, SAP Press, chapter 5.

[23] Y. Kim, S. Kang, S. Lee, S. B. Yoo, “A distributed, open, intelligent product data management system”, March 2001, Vol 14(2), Pages 224-235.