Profile Oriented User Distributions in Enterprise Systems with Clustering*

Ping-Yu Hsu[1] and Ping-Ho Ting[1,2]

[1] Business Administration Department, National Central University, 300 JongDa Rd, ChungLi, Taiwan 320
pyhsu@mgt.ncu.edu.tw

[2] Information Management Department, ChungChou Institute of Technology , 6, Line 2, Sec 3, Shan-Chiao Rd.,Yuanlin Changhwa., Taiwan, 510
ding@dragon.ccut.edu.tw

Abstract. As enterprises world-wide racing to embrace real-time management to improve productivities, customer services, and flexibility, many resources have been invested in enterprise systems (ESs).All modern ESs adopt an n-tier client-server architecture that includes several application servers to hold users and applications. As in any other multi-server environments, the load distributions, and user distributions in particular, becomes a critical issue in tuning system performances.
Although n-tier architecture may involve web servers, no literatures in Distributed Web Server Architectures have considered the effects of distributing users instead of individual requests to servers. The algorithm proposed in this paper return specific suggestions, including explicit user distributions, the number of servers needed, the similarity of user requests in each server. The paper also discusses how to apply the knowledge of past patterns to allocate new users, who have no request patterns, in a hybrid dispatching program.

1 Introduction

All modern ESs share a common IT foundation, namely, the n-tier client-server architecture. The architecture has a database server in the storage layer, multiple application servers in the service layer, several web servers in the interface layer and browsers or other access devices in the presentation layer. In the architecture, programs, applications, or transactions are held and executed in the application servers. When users logging on a system, he/she either selects an application server or is being assigned to one by the system.

It is a vital issue to keep response time under control for most system administrators. When the increases of memories and CPUs reach the hardware limitation, adding more application servers to a system is a reasonable alternative. When an ES has multiple application servers, distributing users with

* This study is supported by the MOE Program for Promoting Academic Excellence of Universities:Electronic Commerce Environment, Technology Development, and Application(Project Number:91-H-FA08-1-4)

H. Jin et al. (Eds.): NPC 2004, LNCS 3222, pp. 230–237, 2004.

similar applications to the same application servers plays an important role in tuning system performance[3], as pointed out by documents of a major ES system [1, 10].

Commercial products, such as SAP R/3, equipped with a simple dispatching algorithm considers only user numbers and server response time [1]. The task of grouping users is left to system administrators [1, 10]. In addition to the rough guideline of grouping financial users into one server and logistic users into another, system administrators need specific suggestions, such as explicit user distributions, the number of servers needed, and the similarity of user requests in each server. To address the needs, the paper shows a set of algorithms to collect transaction patterns, establish pattern prediction rules, associate patterns with users, group users into clusters with patterns, select clusters to form distributions, and dispatch users with hybrid methodology that can dispatch users with or without patterns. Patterns can be collected from system logs or traces because the transactions run by each user in daily operations are specified and set in implementation phases and are seldomly changed after system going live.

2 Finding Users' Regular Transactions

To record system and user statuses, most enterprise systems include various tracing mechanisms. Among the various recordable data are user sessions and applications executed in sessions. For the purpose of the paper, these data are transformed into user profiles. A user profile is a set of $\{\langle$ user-id, transaction-set $\rangle\}$, where user-id is the account name of a user and transaction-set is the set of transactions accessed by the user in a session.

To compute or estimate regular transactions for each user, three steps are employed. The first one computes large itemsets with any existing set oriented pattern discovering algorithm, such as [3, 11]. In the second algorithm, each large 1-itemset is examined against each user to form users' regular transactions. For new users who do not have accumulated enough entries to computer personal regular transactions, the paper propose to predicate their regular transactions with the association rules computed with known Aprori algorithms. Assume the regular transactions is shown in Table 1.

New users do not have any records in the user profiles and do not have associated regular transactions. However, dispatching programs still need to dispatch them in run-time. Therefore, help for dispatching programs to guess the patterns of new users are in order.

If each new user provides one of the transactions she/he wishes to access after logging on, the dispatching program can check if the transaction has high association with any large itemsets. If so, the union of the large itemsets denote the user's Predicted Regular Transaction set is the third step.

[3] An application in an ES corresponds to an atomic and unbreakable transaction. In this paper, transactions and applications are used interchangeably.

Table 1. Regular Transactions

User-Id	Regular Transactions
1	{A, B, E, F, H}
2	{A, B, E, F, H}
3	{A, B, E, F, H}
4	{I, J, K}
5	{B, I, J, K}
6	{B, I, J, K}
7	{P, Q, R}
8	{P, Q, R}
9	{P, Q, R}
10	∅

Definition 1 *The Associated Regular Transactions of a transaction, t, under a set of large itemsets, P, a user profile, U, is*

$$AT(t) = \cup_{p \in P \,\wedge\, t \in p} CP_U(p|t) \geq \textit{confidence threshold},$$

$$\textit{where } CP_U(p|t) = \frac{|\{s|s \in U, p \in s.transaction\ set\}|}{|\{s|s \in U, t \in s.transaction\ set\}|}$$

3 The Definitions of Similarity Measure, Clusters, and Distributions

Load balancing programs utilizes the benefit of multiple servers at the cost of wasting memories in keeping duplicated programs and data. In sophisticated application servers with hundreds of people on-line, the memory needed for transactions are considerable [2]. Therefore, users with similar regular transactions should be grouped into one cluster, which are then assigned to a server. This section defines the measure of similarity and proves related properties. Formal definitions of clusters and distributions are also included.

Definition 2 *A Cluster is a set of users that share similar regular transactions.*

The similarity of users in a cluster is measured by AR, Application Reusability. The AR of a transaction in a cluster is defined as the percentage of users in the cluster evoking the transaction, and the AR of a cluster is defined as the average AR of transactions in the cluster.

Definition 3

- *The Regular transactions, T, of a user, u are defined as T(u)*
- *AR of a transaction, t, in a cluster, c, is defined as*

$$\boldsymbol{R}(t,c) = \frac{|\{u \,|\, u \in c \textit{ and } t \in \boldsymbol{T}(u)\}|}{|c|}$$

.

- *The AR of a cluster, c, is defined as the average AR of regular transactions in the cluster.*

$$\boldsymbol{R}(c) = \frac{\sum_{t \in \boldsymbol{T}(u) \ and \ u \in c} \boldsymbol{R}(t, c)}{|\{t \,|\, u \in c \ and \ c \in \boldsymbol{T}(u)\}|}$$

.

Theorem 1 Conditional Anti-Monotonicity of AR ***R****(c) decreases with new user added to c, if the number of transactions accessed by the new user is fewer than or equal to the average number of transactions accessed by original users.*

Proof
ommited to save space.

Therefore, $\boldsymbol{R}$(c) has the property of Conditional Anti- Monotonicity, which allows POCA to prune hapless user groups.

Definition 4

- *A cluster whose AR exceeds a given ARThreshold is called qualified cluster.*
- *A set of clusters is comprehensive under a user profile, U, if the union of the clusters includes all and only all users with regular transactions.*
- *A set of clusters is disjoined if the intersection of any two clusters in the set is empty.*
- *A set of qualified clusters is a distribution under a user profile, U, if they are comprehensive under U and disjoined.*

The clusters of {1, 2, 3}, {4, 5, 6}, and {7, 8, 9} have ARs of 100%, 11/12, and 100%, respectively, under the running example. If the ARthreshold is set at 55% then all three are qualified clusters. The three clusters are both comprehensive and disjoined in the example, and therefore form a valid distribution.

4 Clustering and Distributing by POCA

POCA returns distributions that satisfy administrator constraints and has the fewest number of clusters, and the rules associating single transactions to predicted regular transactions. The constraints include an AR threshold, min-support, rule confidence threshold. The recommendations guarantee that when all frequent users logging on the system and accessing all regular transactions, each server still has an AR above the given AR Threshold. Information included in the recommendations are number of servers, user distribution and associate rules of predicting patterns from transactions.

POCA relies $\geq_U$, which is a chain, to hold Conditional Anti-Monotonicity and to form each user combination at most once.

Definition 5 *Let S be the set of users in a user profile, U. The order $\geq_U$ is defined on S such that for any $u_1, u_2 \in S$, $u_1 \geq_U u_2$ if*

– $\boldsymbol{T}(u_1) > \boldsymbol{T}(u_2)$, *or*
– $\boldsymbol{T}(u_1) = \boldsymbol{T}(u_2)$ *and user-id of* $u_1 \leq$ *user-id of* u_2.

POCA includes two major steps in computing the recommendations – computing the set of qualified clusters and selecting clusters to form server distribution. The main steps are listed as following:

Initialization: for each user with regular transactions, turns the user into a single-user cluster. These clusters form C_1, the 1-user cluster set.

Composing C_{i+1} from C_i: Conditional join C_i with C_1 to form C_{i+1}. A cluster c_i from C_i is added by one user in c_1 from C_1 if two criteria are met. The first one states that the use from c_1 has lower rank in $\geq_U$ than any user in c_i. The second criterion asserts that the new cluster has an AR value exceeding the given threshold.

Repeating Last Step Until No New Clusters are Generated: If C_{i+1} is empty then POCA has found all qualified clusters in C_1, ..., and C_i; Otherwise, POCA has to repeat the last step.

Selecting Clusters to Form Distributions: Finding the fewest number of qualified clusters to form distributions. The algorithm includes a loop to check if i clusters can form a distribution where $1 \leq i \leq C_1$. The loop is aborted when distributions are found.

In the running example, the first cluster set is formed by turning each user into a cluster. If setting the AR threshold at 55%, C_2, the set of 2-user clusters, is equal to the join of C_1 and C_1. $C_2 = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{5, 6\}, \{5, 4\}, \{6, 4\}\ \{7, 8\}, \{7, 9\}, \{8, 9\}\ \}$. C_3 is the conditional join of C_2 and C_1. $C_3 = \{\{1, 2, 3\}, \{4, 5, 6\}, \{7, 8, 9\}\}$. C_4 is the conditional join of C_3 and C_1. $C_4 = \{\{1, 2, 3, 7\}, \{1, 2, 3, 8\}, \{1, 2, 3, 9\}\}$. C_5 is empty since potential 5-user clusters have ARs lower than 55%.

Now that all cluster sets are ready, it is time to select clusters to form user distributions. The selection continues by examining 1-cluster distribution, 2-cluster distribution, etc. In the running example, 1-cluster and 2-cluster distributions are empty since there are no 9-user and 5-user clusters. On the other hand, 3-cluster distributions have various alternatives and are all returned to system administrators. 3-cluster distributions and corresponding ARs are listed Table 2. POCA just picks one to examine the comprehensiveness of clusters. The algorithm returns all the distributions that satisfy the requirements and let system administrators to decide which distribution he/she prefers.

Table 2. 3-Cluster Distributions and ARs

User Distributions	ARs
{1, 2, 3}, {4, 5, 6}, {7, 8, 9}	100%, 11/12, 100%
{1, 2, 3, 7}, {4, 5, 6}, {8, 9}	18/32, 11/12, 100%
{1, 2, 3, 8}, {4, 5, 6}, {7, 9}	18/32, 11/12, 100%
{1, 2, 3, 9}, {4, 5, 6}, {7, 8}	18/32, 11/12, 100%

5 An AR Based Hybrid Dispatching Approach

Each ES typically has a dispatching program listening to networks and accepts user requests. The program resides an application server, intercepts user requests, and direct them to application servers.

The distributions suggested by POCA bases on frequent patterns in user profiles. For new and infrequent users, POCA does not suggest their distributions directly but returns association rules, PR (Prediction Rules), in the output to help dispatching program make the decision. To apply the rules, a new user only needs to provide a transaction he/she plan to evoke after logging on the ES. With the association rules, a dispatching program can distribute a user according to its associated predicted regulation transactions. If the first transaction does not lead to any predicted regular transactions, then the single transaction works as the basis for dispatching.

An AR Based Hybrid dispatching algorithm distributes users while keeping the AR of each server as high as possible. In the dispatching procedure, users are distributed to a server by one of the three alternatives:

- If a regular user logging on, then send the user to the recommended server and return to listening mode.
- If an infrequent user logging on with a transaction, then find the predicted regular transactions implied by the transaction. If no entry matched then the single transaction is treated as the predicted regular transaction.
- Compute the potential new AR in each server with the addition of the user with predicted regular transactions. Assign the user to the server with the highest AR, and update the AR in the corresponding server.

The distribution in the running example has ARs of 100%, 11/12, and 100% in the three servers. If a new user with user-id 10 wishes to log on the system and submits an A as the first transaction then the user has a assumed predicted regular transaction of ABE. The ARs after adding ABE to the three servers would be 18/20, 14/24, 12/24. Therefore, the new user is distributed to the first server, and the distribution becomes {1, 2, 3, 10}, {4, 5, 6}, {7, 8, 9}.

6 Related Work

With the Internet rush, many researches have been devoted to distributing user requests with Distributed Web Server Architecture to improve the performance of web servers. Depending on the locations where request distributions happen, these researches are classified into client-based, DNS (Domain Name Server)-based, dispatcher-based, and server-based by [5, 4, 14, 15]. Since current Http protocol is stateless, each request is routed independently to a web server[4]. All of the above researches assume that requests can be independently route to different servers, where as in the application servers of ESs, requests from the same users have to be routed to the same server.

Clustering literatures are classified into partitioning clusterings and hierarchical clusterings [12, 6, 9]. If k clusters are needed, partitioning Clustering choose k centroids initially and gradually tune the constituents of each clusters or centroids with some criteria function until a locally optimized characteristic is met. Hierarchical clusterings can be further divided into agglomerative and divisive clusterings. As the name suggested, agglomerative clusterings gradually merge smaller clusters into larger clusters until k clusters are found. Divisive clustering, on the other hand, splits larger clusters into smaller clusters until k clusters are found. POCA is more close to agglomerative although it does not have predefined cluster numbers.

Most clustering algorithms employ Euclidean distances to compute similarity. The shorter the distances the similar the data points in the clusters are. However, Euclidean distances are not ideal for clustering categorical data. For example, to cluster transaction sets with Euclidean distances, each set has to be translated into a sparse binary vector. Many set oriented algorithms use Jaccard coefficient [12] and ROCK [7]. However, Jaccard coefficient and ROCK along cannot describe the number of elements in each cluster, which are important to calculate the buffer efficiency.

7 Conclusion

Managers in enterprises often add users to ESs as they extend E-business practices to various parts of corporate operations. With the addition of each user, new pressures on performances are brought upon the systems. Yet, system response time is one of the most important factors in measuring user satisfactions.

Since ESs tend to consume many memories, application servers can easily run up all memories set by hardware constraints. When this happens, next step commonly adopted in boosting performance is adding application servers to ESs. With multiple application servers in the scene, distributing users with similar application requirements to the same application servers increases buffer utilization and increase the lead time to next hardware upgrades. POCA provide suggestions of such distributions with the fewest number of clusters. Along with suggestions are Application Reusability in each server for the reference of system administrators.

Several issues require further studies, such as modeling user profiles with sequences, dynamically updating user patterns, incorporating CPU and systems loads into dispatching and distribution algorithms, and improving the efficiency of POCA.

References

[1] SAP AG. *System R/3 Technicale Consultant Training 1 - administration*, chapter R/3 WorkLoad Distribution. SAP AG, 1998.

[2] SAP AG. *System R/3 Technicale Consultant Training 3 - Perf. Tuning*, chapter R/3 Memory Management. SAP AG, 1998.

[3] R. Argawal and R. Srikant. Fast algorithms for mining associations rules. In *Proceedings of International Conference in Very Large Data Bases*, pages 487–499, 1994.

[4] H. Bryhni, E. Klovning, and O. Kure. A comparison of load balancing techniques for scalable web servers. *IEEE Network*, 14:58–64, 2000.

[5] V. Cardellini, M. Colajanni, and P.S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3:28–39, 1999.

[6] R. O. Duda and P. E. Hard. *Pattern Classification and Scene Analysis*. Wiley-Interscience Publication, 1973.

[7] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[8] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*, chapter Mining association rules in large databases. Morgan Kaufmann Publisher, 2001.

[9] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*, chapter Clustersing. Morgan Kaufmann Publisher, 2001.

[10] J.A. Hernándes. *The SAP R/3 Handbook*, chapter Distributing R/3 Systems. McGraw-Hill, 2 edition, 2000.

[11] J. Pei J. Han and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 1–12, 2000.

[12] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[13] P. Mohapatra and H. Chen. A framework for managing qos and improving performance of dynamic web content. In *Proceedings of Global Telecommunications Conference*, volume 4, pages 2460–2464, 2001.

[14] S. Nadimpalli and S. Majumdar. Techniques for achieving high performance web servers. In *Proceedings of International Conference on Parallel Processing*, pages 233–241, 2000.

[15] B. C-P. Ng and C-L. Wang. Document distribution algorithm for load balancing on an extensible web server architecture. In *Proceedings of International symposium on cluster computing and the Grid*, pages 140–147, 2001.

[16] J. Zhang, T. Hamalainen, J. Joutsensalo, and K. Kaario. Qos-aware load balancing algorithm for globally distributed web systems. In *Proceedings of international conferences on Info-tech and Info-net*, volume 2, pages 60–65, 2001.