國立中山大學資訊工程研究所
碩士論文

指導教授： 楊竹星 教授

設計並實作一個有效率的應用伺服網站

Design and Implement an Efficient Web Application Server

研究生：吳志鴻 撰

中華民國八十九年六月

設計並實作一個有效率的應用伺服網站

指導教授：楊竹星教授　　學生：吳志鴻

國立中山大學資訊工程系研究所

摘要

因為電子商務所能獲得的龐大利益，使得應用伺服網站已經迅速成為在競爭上佔取優勢的重要資源。　然而所有的利潤可說是都來自於消費者，所以能夠吸引越多的消費者就可以獲得越多的利潤。　因此一個可以提供更有效率的應用伺服網站將可以吸引更多的人潮。　而為了獲得最大的優勢，我們研究了網際網路進而發現了一個新的方法。　一個應用伺服網站意味著它將會有十分頻繁的交易服務。　而且它將必須透過 CGI (Common Gateway Interface)程式去資料庫存取資料和產生新的網頁以回覆給消費者。　在這篇論文中我們發表了一個經由節省網路的頻寬、減輕伺服器的負載和縮短使用者的等待時間以改進應用伺服網站效率的新方法。　這方法主要是將原本網頁的原始碼分開成純粹的動態資料和共通的原始碼兩部份。　藉著節省重覆下載共通的原始碼所花費的時間和減少對伺服器發出請求的次數使得網路的頻寬、伺服器的負載和使用者的等待時間能實際的改善整體效率。　而這方法也可以容易的和其它現存用來解決網際網路效率問題的技術共同使用。　此外我們提供了以新方法去設計和實作一個有效率的應用伺服網站的途徑。　最後我們測量了以傳統方法和新方法實作的應用伺服網站在各方面的數據，用以證實這個方法的的確確可以節省網路的頻寬、減輕伺服器的負載和縮短使用者的等待時間。　希望藉由這個方法可以改善目前在網際網路普遍存在的問題。

關鍵字: 應用伺服器，動態網頁，網站效率。

Design and Implement an Efficient Web Application Server

Chu-Sing Yang and Jr-Houng Wu

Department of Computer Science and Engineering,

National Sun Yat-Sen University, Kaohsiung, Taiwan, ROC

Abstract

Web application servers are rapidly becoming the essential resources for competitive advantage, because e-businesses can gain amount of revenues. However, benefits are coming from consumers, more consumers cause more benefits. So web application server, which can provide more efficient services, will attract more people. To gain the biggest advantage, we research in Internet to find out a new method. A web application server means it will have frequent transactions. And it will access data in database and produce new web pages to respond consumers through the CGI (Common Gateway Interface) programs. In this paper, we present a new method that can improve performance of web application server through saving network's bandwidth, reducing web server loading and cutting wait-time of end user. The method primarily uses divided dedicated data and common information of the source web page. With saving the multiple download time of the common information, the efficiency of network's bandwidth and the user waiting time can be improved effectively. The method also can be easily combined with other existed technologies

that can solve Internet efficiency problems. And we developed an approach to design and implement an efficient web application server. Finally, we measure the result of the traditional method and the new method to prove that our method really can save network's bandwidth, reduce web server loading and cut wait-time of end user. We hope the method can improve general existed problems in Internet.

Keyword : Application Server, Dynamic Page, Web Performance.

Acknowledgements

To my advisor: Chu-Sing Yang,

I am sincerely grateful for his guidance.

To C.Y. Liu,

I especially thank for his advices.

To my parents and Ying-Chieh Lin,

I would like to express my gratitude to them.

To my classmates in the Parallel and Distributed System Laboratory,

Thanks for their comments and help.

Content

Chapter 1 Introduction

1.1 About Internet and WWW

The World Wide Web has continued to grow exponentially as Internet's popularization. People can search what they want or show their thoughts through Internet by WWW. The primary web server almost offers only static pages. Nowadays, as business appears, web server's services have changed its focus from static pages to dynamic pages. And the way of providing dynamic pages is CGI (Common Gateway Interface). One high performance web server can typically deliver up to several hundred static files per second on a uniprocessor. Contrastingly, the rate at which dynamic pages are delivered is often orders of magnitude slower. Because dynamic pages are constructed by programs that execute at the time a request is made, those can seriously reduce web server performance.

Generally, various fields care as follows:

The people who study in Internet focus on the network bandwidth's usage.

The services provider focus on the web server's performance.

The end users at the client side focus on wait-time in downloading pages.

As continued growth of the World Wide Web motivates techniques to improve various performances. For example, proxy caching can improve all above statements, but only in static page. FastCGI can only improve that server constructs dynamic pages performances [5]. Hence, if we can improve all mentioned practices in dynamic pages, it will have great contributions.

1.2 About Web Application Server

In traditional business environment (Figure 1), many inter-company processes are performed using paper documents, such as purchase orders and invoices. This is usually time-consuming and costly when the volumes are large. And those processes of document exchanges invite extensive manual processes (data entry and re-entry), manual intervention, interpretation, resulting in time delay, labor costs, and error. To solve these problems, Electronic Data Interchange (EDI)(Figure 2) is consequentially.

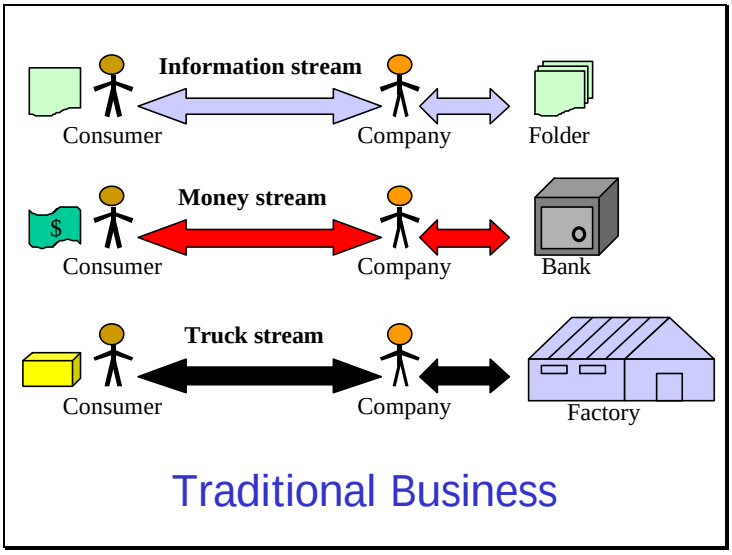

Figure 1

Electronic Data Interchange (EDI) allows companies to exchange their documents in a structured and computer-processable format. It is a set of specifications for formatting documents that is designed to automate business flow among businesses by replacing paper documents with paperless ones. EDI documents, unlike paper documents, are processed electronically by application programs with no human intervention, saving time and costs by reducing or eliminating paper transactions and phone calls, compressing document turnaround

times, and improving data accuracy by reducing error introduced while entering data manually [7].

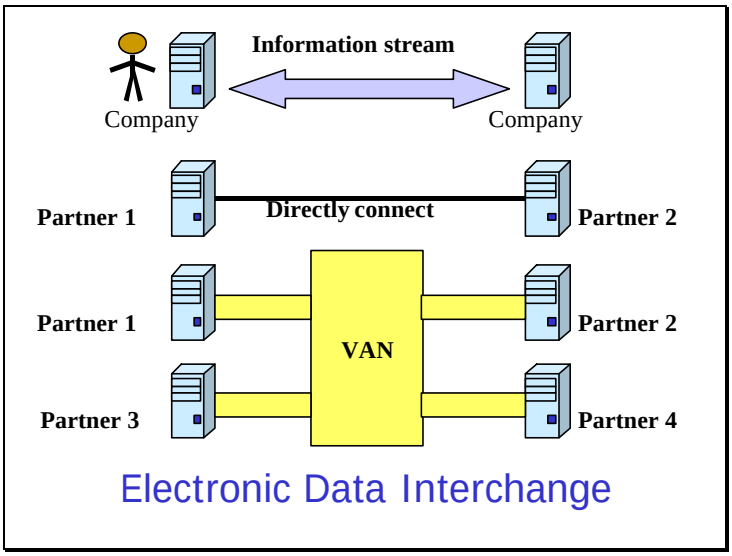

Figure 2

Although EDI has been successfully employed in specific industries (such as retail) and in some large enterprises, it has not been widely adopted. The primary reasons are the high costs of implementation and the costs of communication, which is frequently done using Value-Added Networks (VANs) [7]. So most medium- and small-size companies do not have ability to construct the EDI system. And the EDI system only focused on business-to-business transactions over proprietary networks.

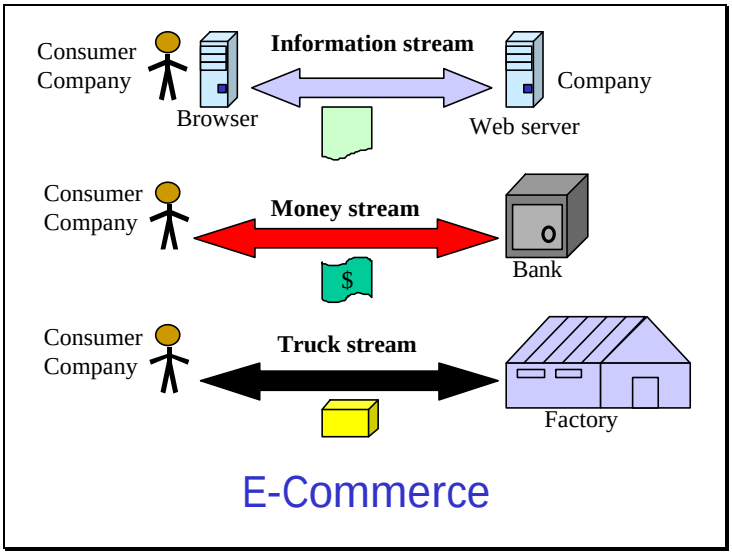

Figure 3

Electronic commerce (Figure 3) has attracted a great deal of attention recently. The current general definition of e-commerce is the ability to do business on-line via the Internet. How different is e-commerce from other commerce? It implies more, such as:

Using a non-proprietary open network, the Internet, with its associated issues of security and reliability.

Not requiring proprietary client software, that is, any browser should do.

Server that is 24 hours a day, 7 days a week, and its associated system reliability requirement.

Customers and suppliers can be geographically distributed worldwide.

A change in the relationships between trading parties. Since it becomes simpler to locate and compare products, the role of 3rd party brokers and middlemen changes. They must provide some significant value added service if they do not want to be bypassed.

The need to establish the identities of parties without requiring physical contact, and the resulting authentication problems.

The need to establish off-line contact between parties through email, voice, fax, or other means.

The ability to collect data and profile parties. Vendors can know more about their customers' habits and needs and so provide more precise marketing.

Although they have many differences, these just only change with the commercial surroundings of development. Most business objects and analysis patterns for e-commerce are the same as those for usual business system. [10]

Internet based electronic commerce (e-Commerce) is flourishing, but mostly in the Business-to-Consumer (B2C). The lack of well-accepted standards is hindering the success in promoting Business-to-Business (B2B) electronic commerce solutions. As to the solutions are Open Buying of the Internet (OBI) of IBM [12], Application Framework of University of Alberta Edmonton [10] and etc.. They offer the probable and reusable construction to e-commerce system of single company.

From above statements of business development, we can know that web application server which is like e-Commerce is replacing the simple client-server application server that is like EDI. And these web application servers will go into amount business. So one efficient web application server will bring a great deal of advantage.

1.3 Motivation

Web application servers are rapidly becoming an essential resource for competitive success, but one business system without benefits is valueless. However, benefits come from consumers, more consumers more benefits. So one who can provide more efficient services will attract more people. To gain the biggest advantage, we propose the way to approach that saving networks bandwidth, reducing server loading and cutting wait-time of client.

Web application servers with the new technology supplies better efficiency for existent web application servers. And it will combine easily with other technologies that can solve Internet efficient problems.

Chapter 2 Background Knowledge

2.1 Internet and World Wide Web

2.1.1 Internet

The Net's roots are in the 1960s. In 1964, the proposal written by Paul Baran was carried out and stated that the new network would have no central authority. The first test network was installed in National Research Laboratory in Great Britain in 1968. The first node (1969) was in UCLA, other nodes were in the Stanford Research Institute, the University of Utah and the UCSB. 1972 was a key year. Ray Tomlinson of BBN invented the first e-mail program. In 1973, the first international nodes were set up. In 1982, the TCP/IP protocol was established for ARPANET. This protocol became standard (instead of NCP) on 1st January 1983. The name "Internet" was first used. In 1984, the number of hosts broke 1.000 and the Domain Name System (DNS) was introduced. In 1987, the number of hosts broke 10.000. In 1989, the number of hosts broke 100.000. In 1992, WWW was released by CERN and the number of hosts broke 1.000.000. In 1993, the first browser, Mosaic, was released. The growth rate of Internet was an incredible 341% and it stills grows and grows now.

The Internet is a packet-orientated network. That means that the data you transfer is divided in packets. The networks are linked by special computers that are called Routers. A Router checks where your packet goes and decides in which direction to send it. Of course not every Router is linked with every other Router, they just decide on the direction your data takes. So if the Routers know where the

data is going, there must be some kind an address. Of course, there is an address, namely the IP - protocol. The data transferred with IP is divided in packets. This is handled by another protocol, the TCP. It was soon discovered that the IP - addresses are of course easy to handle for computers. This was a brief description of how the Internet functions.

2.1.2 World Wide Web

The World Wide Web was invented at CERN, an institute for particle physics situated in Switzerland in 1992. Originally, WWW was developed only for high energy physics (for world-wide communication). In 1994, the first line mode browser was available by anonymous FTP. In 1994, there was the first International WWW Conference, also known as "The Woodstock of the Web". The IW3C2 (International WWW Conference Committee) was founded by NCSA and CERN in Boston. Then it is developed continually from now on.

All WWW pages are written in Hyper Text Markup Language (HTML). While some files may have different file extensions (such as .php or .asp), their core is still HTML. A WWW browser interprets the HTML - code and displays it. HTML is a special version of SGML (is used by big companies for exchange of data) focused on Hypertext. HTML code is written in ASCII - format. This is a big advantage, because ASCII can be read by about any platform (IBM, Macintosh, UNIX,...) thus making the WWW usable for any platform as long as viewer programs, the browsers, exist. The current standard defined by he W3 Consortium is HTML 4. It all started with HTML 1.0. This was no official standard. HTML 1.0 is just what the first real popular browser, Mosaic, could do. The first official version of HTML was 2.0. This is till the most basic standard when it comes to web pages. If you want a page to be readable by any browser, you must use HTML 2.0. A more sensible and newer standard is HTML 3.2. HTML 3.0 was refined because it was not widely accepted.

Farther, the WWW is structured on the basis of client-server that is browser and web server. For example, Microsoft's IE is the browser and Apache Server is the web server. The browser and web server connect with Hypertext Transfer Protocol

(HTTP) in the Internet. In the early, the web server is a passive server that responds to the request that is sent by browser. (Figure 4) Until the method called PUSH is exhibited, but it is also not used generally.

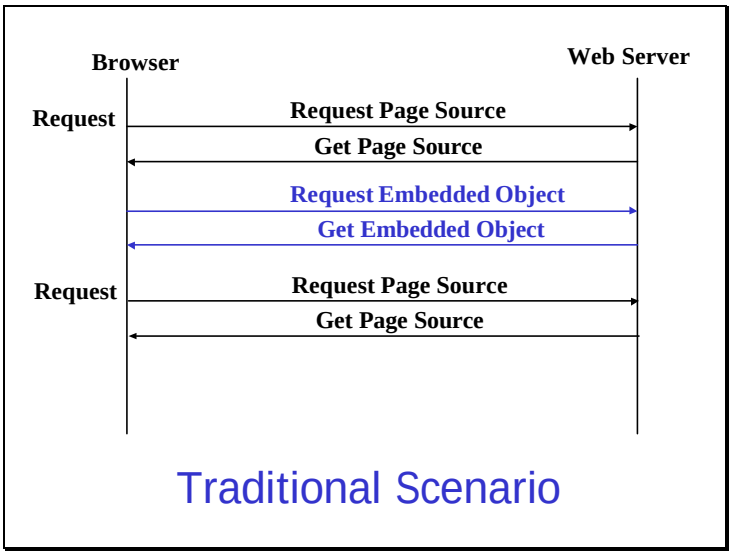

Figure 4

Besides, the web server can support not only static pages that are stored previously on the web server but also dynamic pages that are made up while receiving a request from the browser. The web server must produce the dynamic pages through Common Gateway Interface (CGI)[4]. (Figure 5) For example, both Personal Home Page tools scripts (PHP) and Active Server Pages (ASP) are.

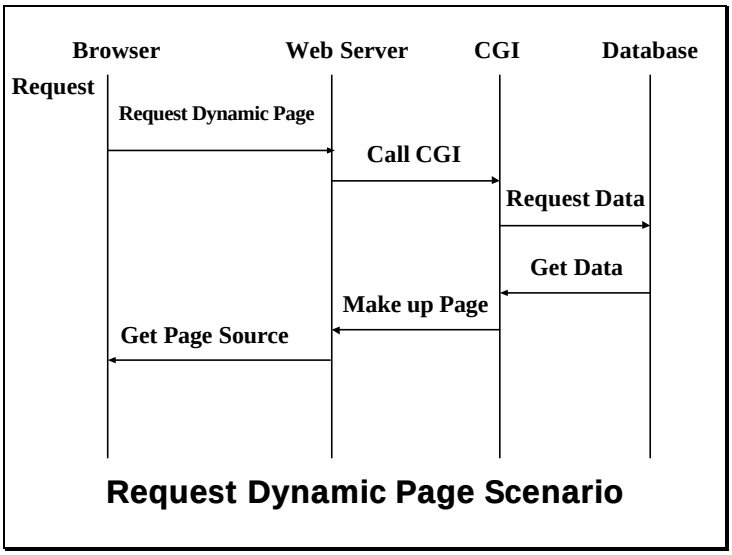

Figure 5

2.1.3 Problem of WWW and Internet

The continued growth of the WWW motivates techniques to improve its performance[1][2][3]. These techniques want to solute the problems that are as follows:

To reduce the user-perceived wait-time(Figure 6) associated with obtaining web documents.

To lower the network traffic from the web server.

To increase the service demands on content providers.

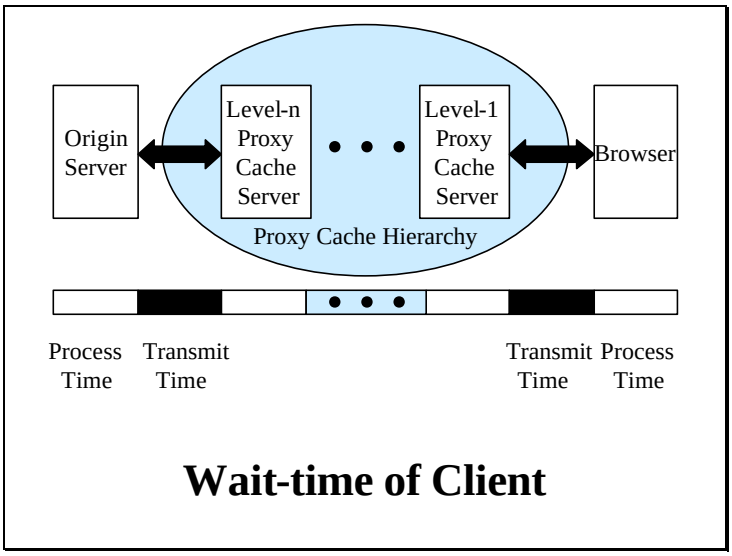

Figure 6

One popular technique is proxy caching, in which one or more computers act as a cache of documents for a set WWW client. These clients are configured to send HTTP requests to proxy. If possible, the proxy servers requests from its cache. Otherwise, the proxy forwards the request to the content provider, that is, to the server containing the source copy of the requested data [1].

How the proxy caching improve the mentioned problems is:

Wait-time of client can be reduced because the proxy cache is typically closer to the client than the content provider.

Network load can be lowered because documents that are served from the cache typically traverse less of the network than when they are served by the content provider.

Proxy caching can reduce the service demands on content providers since cache hits need not involve the content provider.

However, the proxy caching has no advantages on the dynamic pages, so do other techniques. Because the dynamic pages are made up while the server is requested, proxy servers cannot cache them. Although there are some new techniques that support the way to hold dynamic pages, they just reduce the web server's loading. Take IBM's Data Update Propagation for example [6]. It is to cache dynamic pages the first time they are created, so it will not have any help when the same dynamic pages are seldom created.

2.2 Electronic Data Interchange

Although the business computer enabled companies to store and process data electronically, companies needed an expedient method to communicate the data. This method was realized by the widespread use of computer telecommunications. Using telecommunications, companies could transmit data electronically over telephone lines, and have the data input directly into a trading partner's business application. These electronic interchanges improved response time, reduced paperwork, and eliminated the potential for transcription errors. Computer telecommunications, however, only solved part of the problem. Early electronic interchanges were based on proprietary formats agreed between two trading partners. Due to differing document formats, it was difficult for a company to exchange data electronically with many trading partners. What was needed was a standard format for the data being exchanged. In the 1960's a cooperative effort between industry groups produced a first attempt at these common data formats. The formats, however, were only for purchasing, transportation, and finance data, and were used primarily for intra-industry transactions. It was not until the late 1970's that work began for national Electronic Data Interchange (EDI) standards. Both users and vendors input their requirements to create a set of standard data formats that:

were hardware independent;

were unambiguous, such that they could be used by all trading partners;

reduced the labor-intensive tasks of exchanging data (e.g., data re-entry);

allowed the sender of the data to control the exchange, including knowing if and when the recipient received the transaction.

Although today there are many syntaxes for EDI, only two are widely recognized: X12 and the Electronic Data Interchange For Administration, Commerce, and Transport (EDIFACT). These two families of standards are mandated for use within the Federal Government. We can find out EDI's evolution from figure 7.

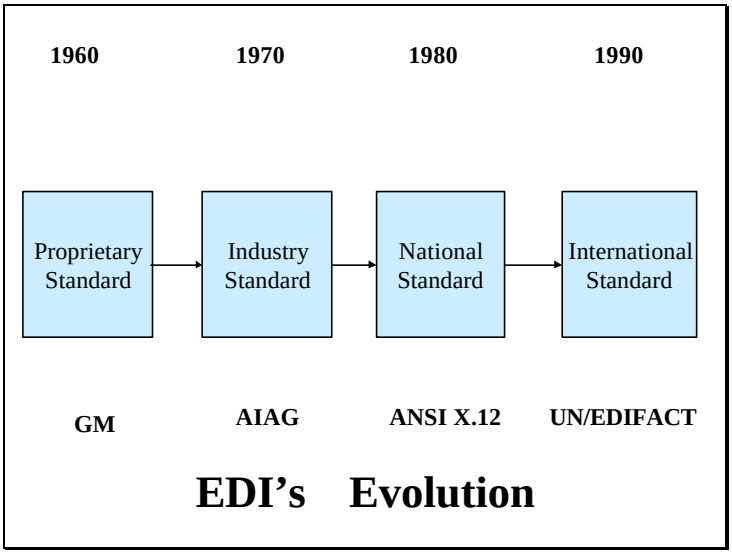

Figure 7

Traditional EDI system contains two major components:

(1) EDI translation software that converts and maps EDI formats to/from internal business applications.

(2) Communication channels that deliver EDI documents to the desired trading partners.

Over the years, different industries have developed their own EDI standards. To translate EDI documents, one must first know what EDI standards the trading partner is using. EDI standards define the document formats that enable trading partners to speak the same language when conducting business activities with each other. Yet, each company usually has its own internal or proprietary data formats, business logic, and business flow, which are typically unique. Therefore, a key requirement of EDI translation software is the ability to integrate the incoming EDI

formats with internal business applications [8]. That is, EDI translation software basically converts to a standard acceptable to the trading partners; conversely, it maps incoming standard formats into the proprietary formats recognized by internal business applications. The functionality of translation software could be obtained in three ways: lease or purchase software from a vendor; have a third party (such as a VAN) perform the translation; or develop software in house. The first two alternatives are usually the most cost- and time-effective as they are easy to install, maintain, and expand.

Trading partners traditionally exchange EDI documents via direct link, private or proprietary networks, and third-party VANs [9].

Direct link networks, including lease lines, are the most straight-forward communication method. They allow a company to dial up and connect directly to partners' computer. They are most cost-effective alternative for transmitting high volumes of data and are thus very appealing to those large companies that must transmit huge amounts of data daily. With direct link, each trading partner provides its own technical support to address issues such as protocol and speed conversion, because different computer system use different communication protocols and transmission speeds. In addition, companies must have phone lines available at the same time, deal with substantial administrative overheads to ensure reliable delivery, provide audit control and recovery procedures in case of communication link failure or unavailability, and so on. These issues are compounded when the number of direct-linked trading partners increases. As a result, direct link network is only applicable to large companies that must transmit high volumes of data daily.

A Private or Propriety Network, usually provided by a hub company, is a closed network only available to trading partners. The hub handles protocol conversion and administrative overheads so that the spokes can dial up to the hub private network without conversion and pay only the cost of a telephone call. This type of network is limited and is only available to those trading partners that have a close relationship. An automobile manufacturer and its part suppliers are a typical example. When this type of network is used, the hub company provides technical supports to both itself and its spokes.

A Value-Added Network plays an intermediary role analogous to a post office or delivery service that provides reliable delivery of documents in a secure environment. VANs provide the following value added services to support EDI: mailboxing, security, administration, implementation assistance, etc.

In the traditional EDI environment, most companies exchange EDI documents via VANs. Despite the popularity, convenience, and flexibility of VANs, their costs are frequently the document expense of EDI. Implementing the translation software is a one-time expense which typically costs from $5,000 to $250,000. VAN services expenses consist of an installation fee, recurring per-transaction fees, and monthly subscription and maintenance fees [8]. In general, typical monthly fees are $50, transaction charges are $0.55-$0.70 per transaction. There can be additional charges for value-added services. In summary, as the VAN charges are mainly based on per-document transaction fees, the accumulated costs can be tremendous.

2.3 E-Commerce

Electronic commerce can be viewed as consisting of interorganization systems (IOS) and electronic markets. Interorganization systems are also known as business-to-business systems. In contrast, electronic markets are primarily for business-to-consumer commerce.

The Interorganization impact of information technology has been a topic of vigorous research dating from the 1970s. An abundance of scholarly work has addressed the rationale underlying this aspect of electronic commerce and its influence on competitive strategy. Similarly, much has been written about the supporting information technologies, especially electronic data interchange. Today there is also intense focus on the Internet for both business and consumer-related commerce in the marketplace and the marketspace.

Another body of knowledge concerning the structure of IOS is emerging. Applegate and Gogan identified a continuum of interorganization relationships consisting of transaction, contracts, and partnerships [11]. Others have described the partnership from used in practice. Yet little has been published to explain how interorganization systems evolve within companies and industries, or to describe the manner in which related business strategies evolve. Extending the theories developed for the earliest stages of interorganization systems so they encompass the advanced forms is an important next step. While there is widespread awareness that technologies and changing rules of business competition are rapidly advancing the forms of electronic commerce in use, there is less awareness of how these advances occur. This investigation identified an evolutionary pattern and key factors of IOS success.

Four stages of business-to-business systems were found:

- Receive and Respond System: It exchanges the documents of standard formatted business and recipient responds to requirements of dominant partner, such as EDI.
- Sense and Respond System: It exchanges the information of performance and status and recipient interprets information to determine appropriate actions, such as universal product code.
- Embedded Business Response System: Elements of one partner's business systems are embedded within those of the other partner. It will be triggered a series of related activities by one action.
- Community Response System: It is a specific business objective that makes members share systems and information through communicating each other. Members may compete and cooperate at the same time.

Chapter 3 Analysis and Methodology

3.1 Analysis of Dynamic Pages

A web application server has become future tide. Nowadays web application servers deal with searching what you want, handling trades, exchanging information, and etc. These functions will handle amounts of dynamic pages that can seriously reduce web server performance. So one web application server must do its best to increase whole efficiency. Due to these reasons, this paper studies the problems of Internet and WWW, and expects to find out solutions applied in the web application server.

Therefore we decide to observe web sites that implement with web application server. We expect to get and understand the relation of dynamic pages and context, dynamic pages and requested Server, dynamic pages and wait-time of client. We choose searching web sites, because searching web sites are the model and primary web sites that implement with dynamic pages. We input the same keyword on different searching web sites to search datum, and collect these dynamic pages' information. We orderly input the keywords of "Web", "Internet", and "E-commerce" to search on the web Sites of Yahoo, Openfind, and Kimo. Then we request ten dynamic pages of every keyword and every web site. (Table 1)

Web Site / Keyword	Yahoo	Openfind	Kimo
Web	10/2164	10/7640	10/143
Internet	10/1421	10/16948	10/45
E-commerce	10/83	10/550	10/20

Keyword & Observed Web Site

Table 1

We can find out the relation of dynamic pages and requested web server is when we request one dynamic page, the web server must handle and produce one dynamic page. (Table 2)

Web Site / Keyword	Yahoo	Openfind	Kimo
Web	10/10	10/10	10/10
Internet	10/10	10/10	10/10
E-commerce	10/10	10/10	10/10

Relation of Page and Web Site

Table 2

We can find out the relation of dynamic pages and wait-time of client is as we request one dynamic page, we must wait for a while, including the time of

transmitting in tangible network line and handling on passing hosts. Because dynamic pages cannot be cached, we have to wait this time.

About the relation of dynamic pages and context, we collect information as following rules:

- We do not take embedded objects that are requested by dynamic page into account, because embedded objects are static objects as static web pages.
- We anatomize the context of dynamic page to count its size without the datum of screen's display in the browser and not the datum that are used on setting display's attributes. (Example 1) Why do we anatomize the dynamic page's context in such way? Because the dynamic pages are from oneness of function on the single URL, the dynamic pages' outside styles are similar.

```
<html> <head><title> Openfind: Global Search </title></head>
<body text="#000000" bgcolor="#FFFFFF" vlink="#641EAA">
 <table width="80%" cellpadding="10"> <tr align="left"><td>
   <ul> <b>1.</b>
    <a href="http://www.perl.com"><b>WWW.PERL.COM</b>
    </a> <br> <ul>
     <font color="#800040">http://www.perl.com</font><i>
     <font size=-1 color=gray>-count:</font>
     <font color="black">1645</font>
     <font color=white>.</font> </i>
    </ul> </ul></td> </tr> </table></body></html>
```

Example 1

We call dynamic page's context as "Source" and the segment that is anatomized by us as "Tag." Tag has a certain size in source from collected information. (Figure 8~ Figure 16) In addition, Tag's total size increases as loading pages from pictograph. (Figure 17~ Figure 25)

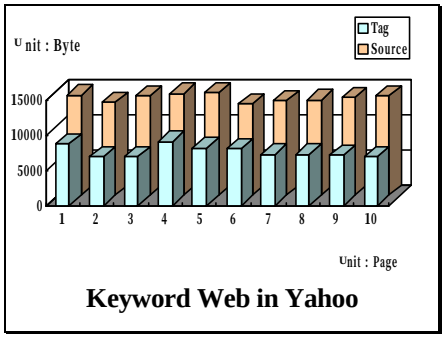

Figure 8

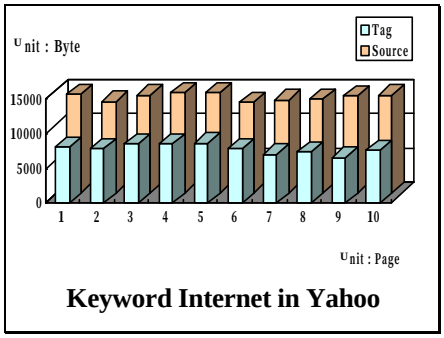

Figure 9

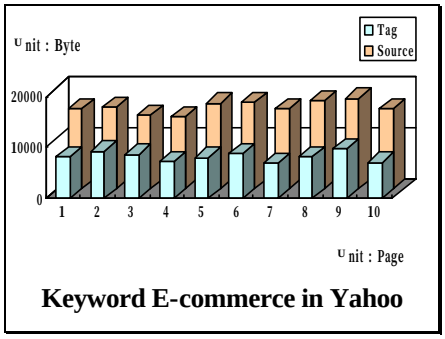

Figure 10

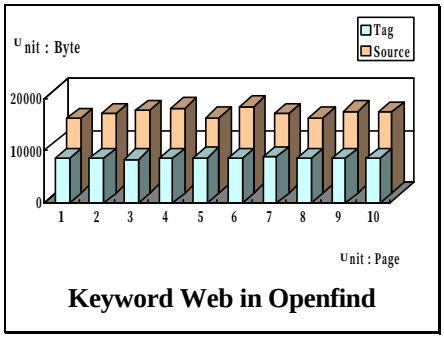

Figure 11

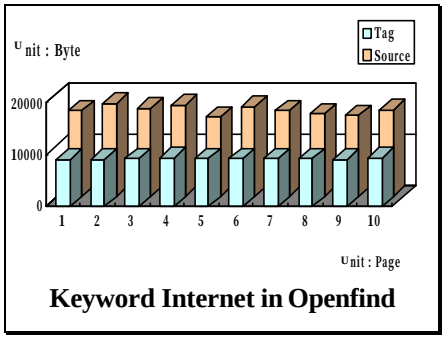

Figure 12

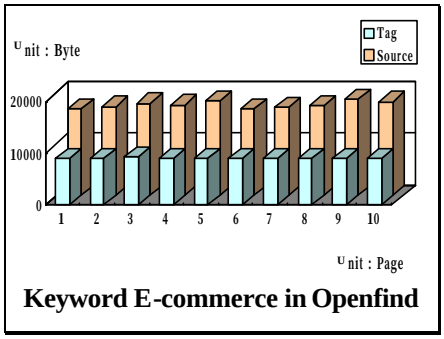

Figure 13

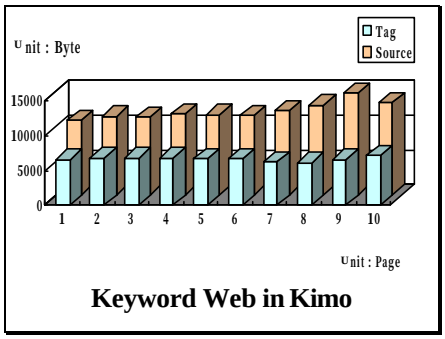

Figure 14

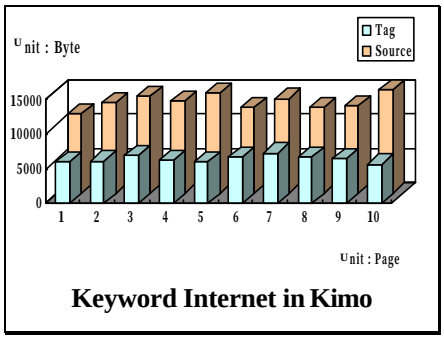

Figure 15

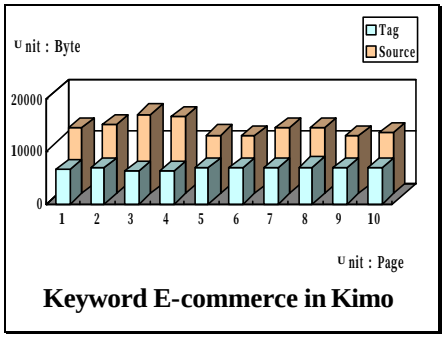

Figure 16

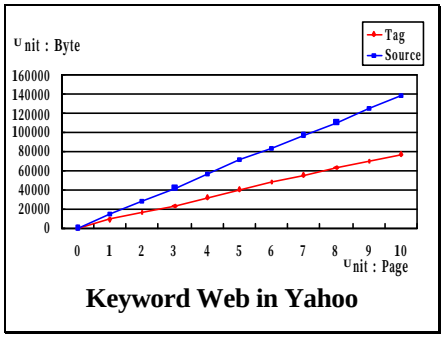

Figure 17

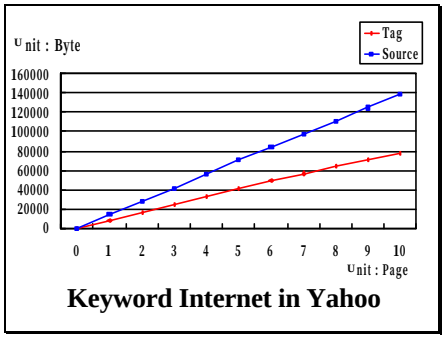

Figure 18

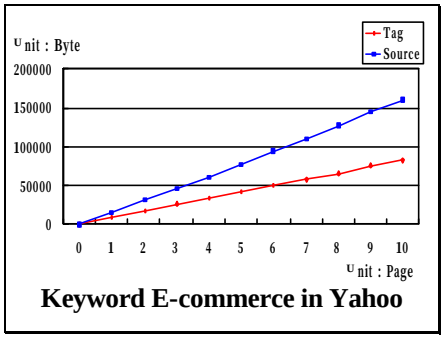

Figure 19

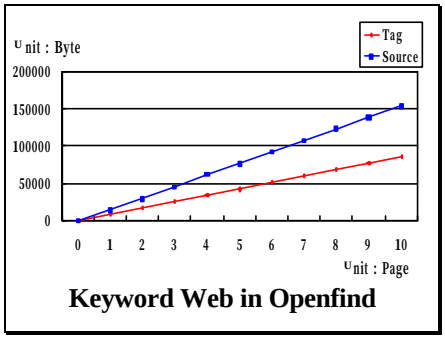

Figure 20

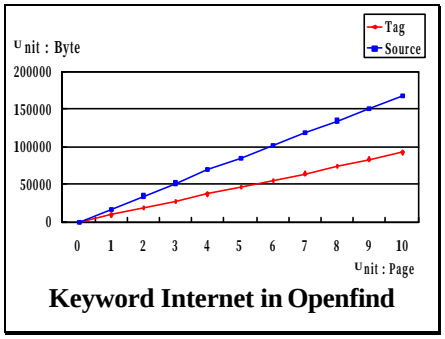

Figure 21

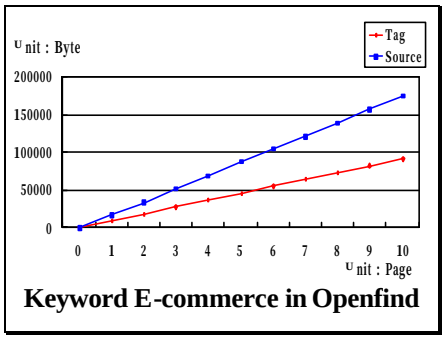

Figure 22

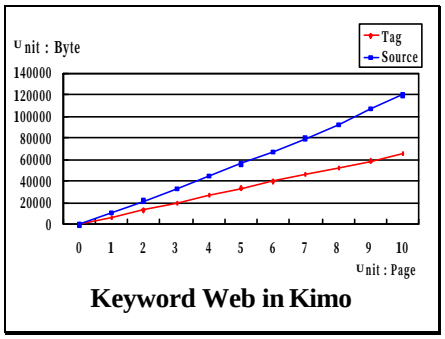

Figure 23

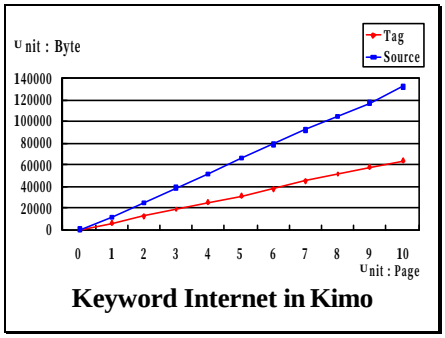

Figure 24

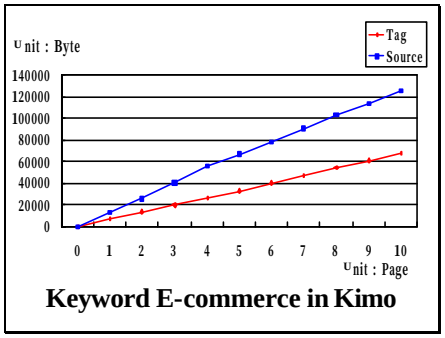

Figure 25

According to the result of observation and measure, we get the ideas as following:

- Because tag has a certain size in source and dynamic page is made up as the same rules, if we can divide tag from source and only transmit one time, it will save much more bandwidth in network.
- Because the act of combining dynamic page is transferred to client-side's browser, it can reduce the loading of combining dynamic page on web server.
- Because the same page size can include more information, it can reduce the loading of requiring dynamic page on web server.
- Because dynamic page is combined in client-side, if the information that you want exists, it can reduce the wait-time of waiting for dynamic page.

3.2 Methodology

Based on these ideas, we call tag and extra JavaScript as **Shape** page and the segment of dynamic content as **Data** page. (Figure 26) We present web page's shape and data dividedly with JavaScript. And the method that divides shape and data is expected as ideas to save the network bandwidth, reduce the web server loading and cut end user wait-time.

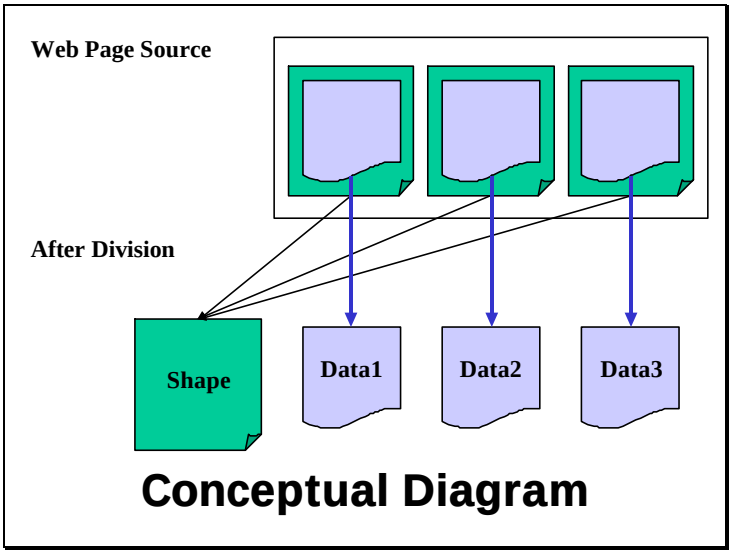

Figure 26

The process of the method is, at first, browser will download the shape page then the shape page will automatic request the CGI program to get dynamic page of data. It implements with "FRAME" that is a tag of HTML. Secondly, the data page will call the shape page's JavaScript to read data when the download of the data page completes. Finally, the shape page's JavaScript makes up the primary page and replaces the data page when the action of reading the data page's data completes. (Figure 27)

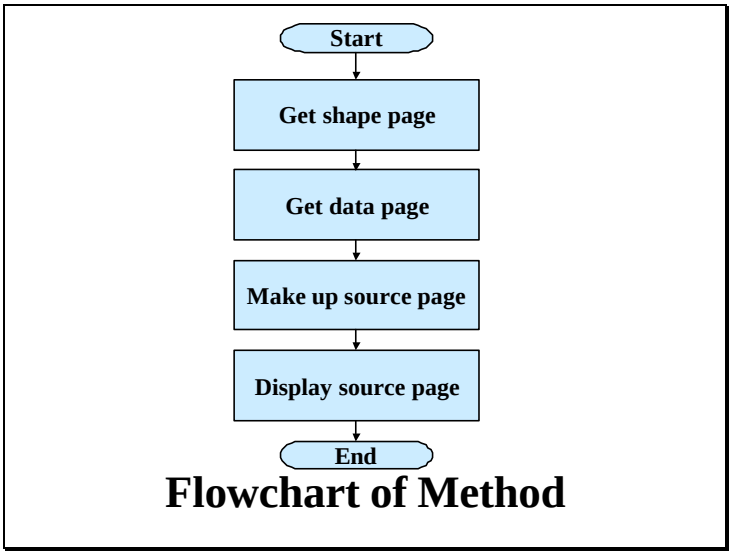

Figure 27

We also find out one thing that a source page's size can carry more dynamic content in data page without affecting other factors. So if we do in this way, it will reduce the times that are requested to response a dynamic page in the web server. But it maybe has two statuses, one is the data that have gotten and the other is the data that have not gotten. The former can reduce the times that are requested to response a dynamic page in the web server, and the latter will not have any extra loading compared with traditional CGI program in every respect. (Figure28, 29)

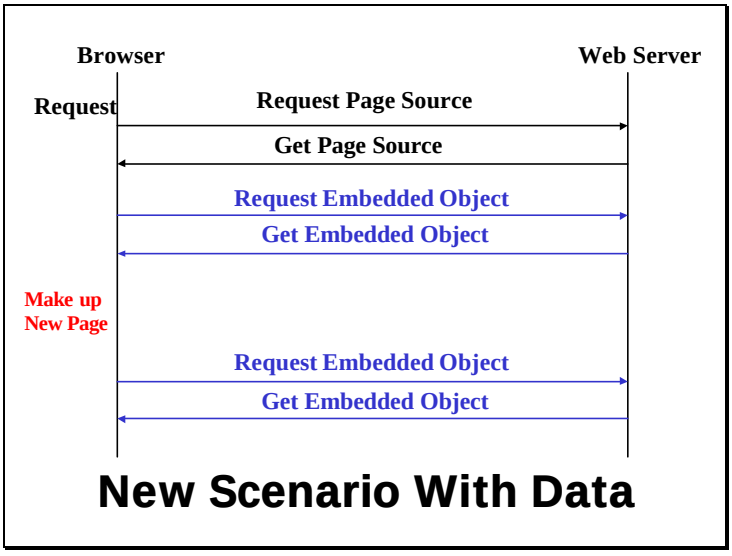

Figure 28

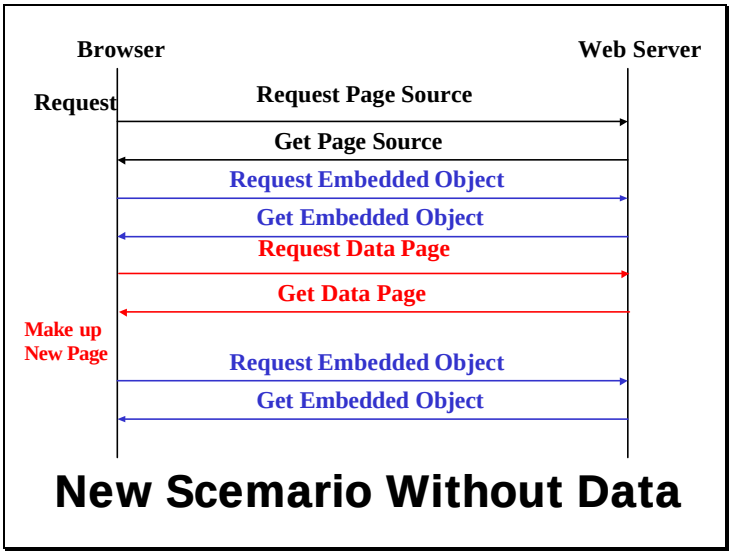

Figure 29

However, we must notice the effect on the web server while using the way to transmit between more data in less request times and original data in original request times. That is to compare the CGI program called one time to finish more works with the CGI program called more times to finish less works, but the works are the same size.

We observe these existent web application server' s behaviors and divide them into following two classes:

- The first behavior is that users only gain wanted data simply.
- The second behavior is that users will handle got data, send to server and get a response page.

Now we discuss the above two classes individually.

3.2.1 Only Gain Wanted Data Simply

At first, we give the precondition that is under the same environment of server, network and client. And all actions are worked after the client has connected with web application server.

We call L_d as the time that is occupied from the client requests a dynamic page to the client displays the dynamic page. We call T_r as the time that is occupied from the client requests a dynamic page to the server receipts the request. We call T_s as the time that is occupied to handle and respond the dynamic page by server. We call T_n as the time that is occupied to transmit the dynamic page on the virtual networks. We call T_c as the time that is occupied from the client receipts the dynamic page to the client displays the dynamic page. When we request a dynamic page in traditional way, we can get the following expression: $\mathbf{L_d = T_r + T_s + T_n + T_c}$

Now, we give that the data amount in the new way is N times of the original one in the same page size. Because server must handle N times data, the time becomes T'_s. And the dynamic page must is made up in the client, so the time becomes T'_c. When we request a dynamic page in the new way, we can get the following expression: $\mathbf{L'_d = T_r + T'_s + T_n + T'_c}$ When N is very small, we can say that T'_s is approximate to T_s. T'_c and T_c are much smaller than L_d, so we can ignore T'_c and T_c. Thus we can say that L_d is approximate to L'_d. We call R is that the client actually requests web application server's times. And we guess the pages' size are the same.

So we can get the average wait-time' s comparison is:

$[L'_d \times (R \div N)] \div R : [L_d \times R] \div R = 1 : N$

We can save the wait-time' s percentage is:

$[L_d \times R] - [L'_d \times (R \div N)] / [L_d \times R] \times 100\% = 100 \times N/(N-1)\%$

We call T_{CGI} as the time that is occupied from the CGI program starts executing to the CGI program finishes executing. Because CGI program must handle N times data, the time becomes T'_{CGI}. But when N is not very big, we can say that T_{CGI} is approximate to T'_{CGI}.

So we can get the average CGI executive time' s comparison is:

$[T'_{CGI} \times (R \div N)] \div R : [T_{CGI} \times R] \div R = 1 : N$

We can save the CGI executive time' s percentage is:

$[T_{CGI} \times R] - [T'_{CGI} \times (R \div N)] / [T_{CGI} \times R] \times 100\% = 100 \times N/(N-1)\%$

We call S_o as original page' s size, S_s as shape and JavaScript program' s size in the new way, and S_d as data page' s size in the new way. When JavaScript program is not very complicated, we can know that S_s is smaller than S_o. We give that S_o and S_d are the same at first.

So we can get the average size' s comparison is:

$$1 : (N+1) < [S_s + S_d \times (R \div N)] \div R : [S_o \times R] \div R < 1 : N$$

We can save the source page size' s percentage is:

$$100 \times N/(N-1)\% > [S_o \times R] - [S_s + S_d \times (R \div N)]/[S_o \times R] \times 100\% > (100 \times R \times N)/[(R \times N) - N - R]\%$$

We can find out that the efficient of new way has great relation with N under this behavior. When N grows bigger, the saving of wait-time, CGI executive time and the total page size will become more. But the N cannot be too big because of our assumption that N is not very big at first. And the methodology lies on above preconditions. So the actual measure number may differ from the methodology number. However, the new way still can save certain amount of wait-time, server loading and networks bandwidth.

3.2.2 Handle, Send Data and Get a Response Page

At first, we also give the precondition that is under the same environment of server, network and client. And all actions are worked after the client has connected with web application server.

We call L_s as the time that is occupied from the client requests a static page to the client displays the static page. We call L_d as the time that is occupied from the client requests a dynamic page to the client displays the dynamic page. We call T_r as the time that is occupied from the client requests a dynamic page to the server receipts the request. We call T_s as the time that is occupied to handle and respond the dynamic page by server. We call T_n as the time that is occupied to transmit the dynamic page on the virtual networks. We call T_c as the time that is occupied from the client receipts the dynamic page to the client displays the dynamic page. When we request a dynamic page in traditional way, we can get the following expression:

$\mathbf{L_d = T_r + T_s + T_n + T_c}$ Because the response page is made up in the client and it does not request the web application server through the networks, the time becomes T'_c. When we request a dynamic page in the new way, we can get the following expression: $\mathbf{L''_d = T''_c}$ Because we also need to send the data that we have handled to server, and wait the response from the server, we can get the following expression:

$\mathbf{L'_d = T_r + T'_s + T_n + T'_c}$ T''_c, T'_c and T_c are much smaller than L_d, so we can ignore T''_c, T'_c and T_c. When we transmit the same size of page, we can know that the L_s is smaller than L_d. And we call R is that the client actually requests web application server's times. We know that $\mathbf{0 < L_s < L'_d}$

So we can get the average wait-time' s comparison is:

$$1 : R < [L_s + L'_d] \div R : [L_d \times R] \div R < 2 : R$$

We can save the wait-time' s percentage is:

$$100 \times (R-1)/R\% > [L_d \times R]-[L_s+L'_d]/[L_d \times R] \times 100\% > 100 \times (R-2)/R\%$$

We call T_{CGI} as the time that is occupied from the CGI program starts executing to the CGI program finishes executing. Because CGI program must handle more data in the new way, the time becomes T'_{CGI}. But when data is not very big, we can say that T_{CGI} is approximate to T'_{CGI}.

So we can get the average CGI executive time' s comparison is:

$$[T'_{CGI}] \div R : [T_{CGI} \times R] \div R = 1 : R$$

We can save the CGI executive time' s percentage is:

$$[(T_{CGI} \times R) - T'_{CGI}]/(T_{CGI} \times R) \times 100\ \% = 100\ \times (R-1)/R\ \%$$

We call S_o as original page' s size, S_s as shape and JavaScript program' s size in the new way, and S_d as data page' s size in the new way. When JavaScript program is not very complicated, we can know that S_s is smaller than S_o. Because the data amount is the same, S_s is smaller than S_o.

We can get the average size' s comparison is:

$$\mathbf{1 : R < [S_s + S_d] \div R : [S_o \times R] \div R < 2 : R}$$

We can save the source page size' s percentage is:

$$\mathbf{100\times(R\text{-}1)/R\% > [S_o \times R]\text{-}[S_s+S_d]/[S_o \times R] \times 100\% > 100\times(R\text{-}2)/R\%}$$

We can find out that the efficient of new way has great relation with R under this behavior. When R grows bigger, the saving of wait-time, CGI executive time and the total page size will become more. And the methodology lies on above preconditions. So the actual measure number may differ from the methodology number. However, the new way still can save certain amount of wait-time, server loading and networks bandwidth.

Chapter 4 Design and Implement

We present web page's shape and data dividedly with JavaScript. Why do we use JavaScript, not use Java Applet? We offer below explanations:

- JavaScript's size will be smaller than Java Applet's size when they do the same work. Because JavaScript is just only script language but Java Applet is a complete object, it will need more time while downloading one Java Applet, especially on slow speed network.
- Using JavaScript communicates with web server more convenient than Using Java Applet does. Because JavaScript is a part of web page, we can use HTTP to communicate with web server. But Java Applet must communicate with web server through Java servlets that uses another port, especially on LAN that has the device like firewall.
- Using JavaScript designs easier and costless than Using Java Applet does. Because Java is scrupulous language, it takes more costs and labors by using Java.

What is there any change in the web server?

- There is not any change on the hardware, software and server's models.
- The primary CGI program that accesses data and makes up web page which will response to end user is divided into a static web page and a new CGI program. The static web page that we call shape page includes style and rules of displaying layout and extra JavaScript program that will combine shape page and data page. The new CGI program will access and output the data.

What is there any change in the client-side’ s browser?

- There is not any change on the hardware, software.
- It must not download a plus-in program in browser.

Now let’ s see how to design and implement the web application server in the new way.

4.1 Only Gain Wanted Data Simply

We can get whole action flow from the Figure 30.

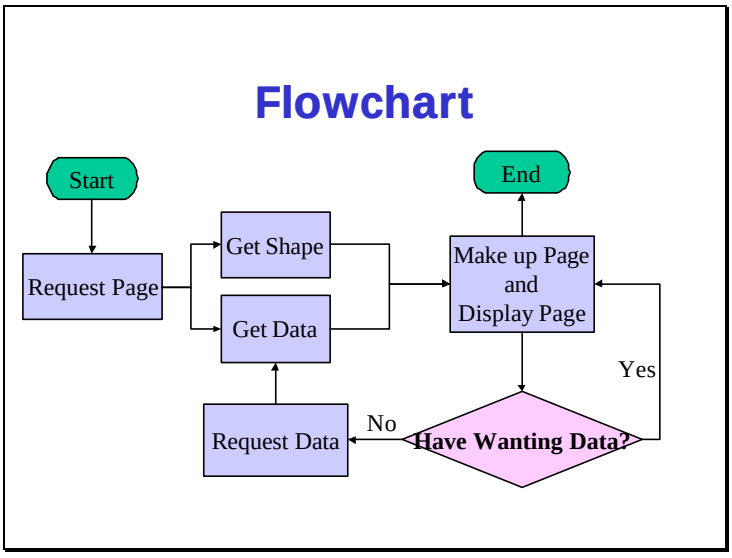

Figure 30

Now the problem is that how we keep the shape page on the client's browser because we cannot control browser to save files in client's disk and when we change another page to see, we will lost the original page. Here we solve this problem by the tag of 'Frame.' The dynamic page that we request will only have shape and JavaScript program and the data page will be requested by the page that we request. And the shape page will keep in the browser until we leave the web application server. Now we can keep the shape page and the next we must do is to make up the page and control the time when we should request data page from web application server. We can see the detail from following Example 2.

```
<SCRIPT>
//shape data
   …….
//make up JavaScript program
   …….
</SCRIPT>
<HTML><HEAD>
<Frameset rows="60,*" BORDER=5 >
<Frame src="/yahoo_test/up.html" NAME="up" SCROLLING=no NORESIZE>
<frame src="" name="fdata" >
</Frameset>
</HEAD></HTML>
```

Example 2

Figure 31 is the picture that is implemented in traditional way and Figure 32 is the picture that is implemented in new way. We can find out that the two pictures look like the same.

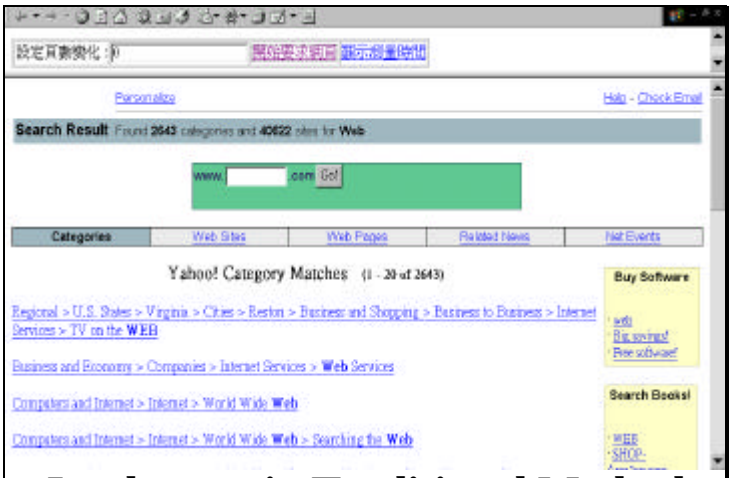

Figure 31

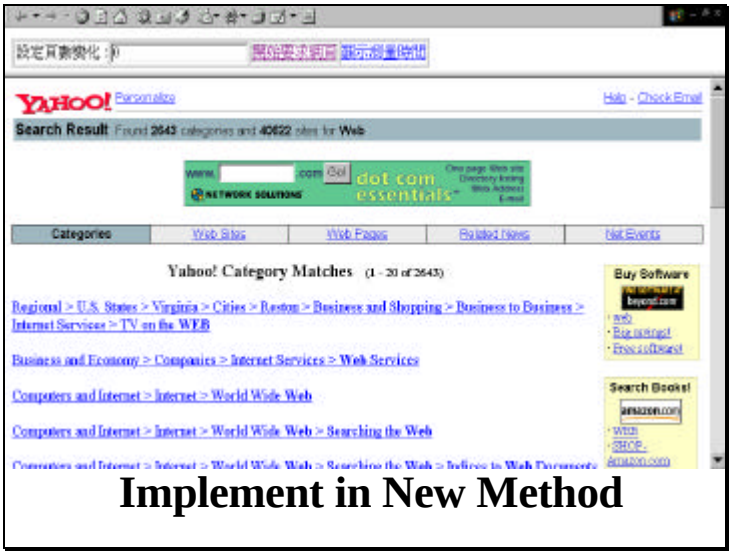
YAHOO!
Search Result
Categories
Web Sites
Web Pages
Related News
Net Events
Yahoo! Category Matches
Buy Software
Search Books!
Implement in New Method

Figure 32

4.2 Handle, Send Data and Get a Response Page

We can get whole action flow from the Figure 33.

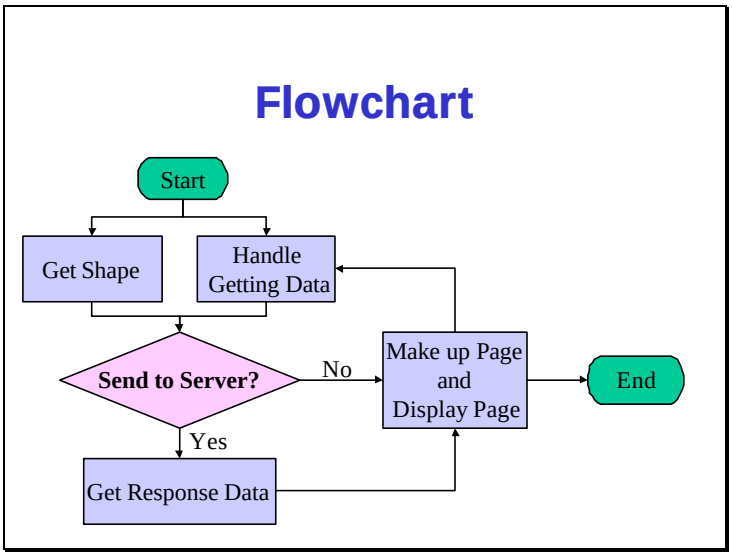

Figure 33

The problem how we keep the shape page on the client's browser has been solved. Now we have the new problem that how we handle the getting data and send to web application server. In traditional way, we can input or handle data then transmit to web application server in the tag 'From,' 'Input' and so on. In the new way, we keep the handled data in a string and send them at proper time. As a result, we will not need to send data while handling data every time like traditional way. We can see the detail from following Example 3.

```
var o_list='';
function add_shopcart(p_content)
{
   o_list=o_list+p_content+"1#";
    show_shopcart();
    return true;
 }
function delete_shopcart(p_content)
{
 str_data=o_list;
  chang_site=str_data.indexOf(p_content,0);
  if(chang_site>0) {pre_str=str_data.substring(0,chang_site); }
    next_str=str_data.substring(chang_site+p_content.length);
 o_list=pre_str+next_str;
 show_shopcart();
}
```

Example 3

Figure 34 is the picture that is implemented in traditional way and Figure 35 is the picture that is implemented in new way. We can find out that the two pictures look like the same.

Figure 34


產品名稱
產品外觀
產品價格
購物車
Love98
400元
加入
Bird
450元
加入
Mtv最愛今晚
410元
加入
883元
加入
水上花園
297元
加入
1800元
加入
330元
加入
Implement in New Method

Figure 35

Chapter 5 Measure

We implement and measure the web application server dividedly in traditional and the new way under the two behaviors that are only gaining wanted data simply and handling, sending data and getting a response page. Now let’ s see the result of measure.

5.1 Only Gain Wanted Data Simply

We call S as the original dynamic page, F&1D as the data page which has the same data amount with original dynamic page in the new way and F&3D as the data page which has three times data amount with original dynamic page in the new way. Here we give that F&3D has the same page' s size with original dynamic page ' S.'

We can see the result of measure on wait-time as following:

Unit:ms

	1	2	3	4	5	6	7	8	9	10
F&3D	27740	50	50	27850	60	60	27900	60	50	27350
F&1D	27460	27460	27400	27290	27460	27460	27510	27350	27300	27350
S	30430	27580	27470	27570	27560	27510	27520	27450	27460	27510

Measure of Wait-time

Table 3

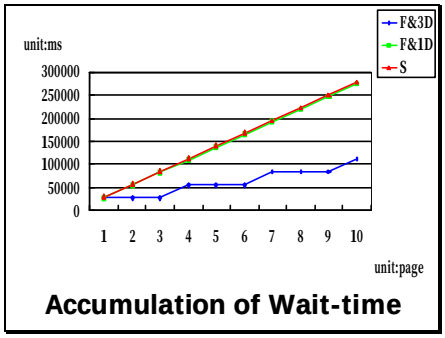

Figure 36

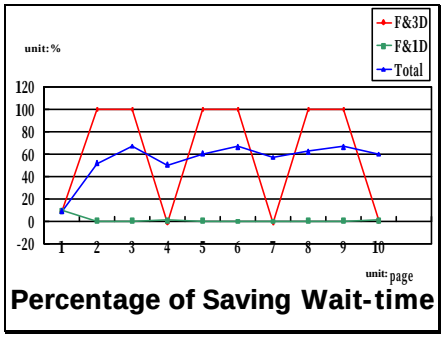

Figure 37

We can see the result of measure on web application server CGI execute time as following:

Unit:ms

	1	2	3	4	5	6	7	8	9	10
F&3D	26116	0	0	26251	0	0	26306	0	0	25802
F&1D	25838	25799	25812	25812	25814	25846	25811	25845	25755	25802
S	26884	25938	25927	25960	25972	25980	25966	25901	25856	25894

Measure of CGI Time

Table 4

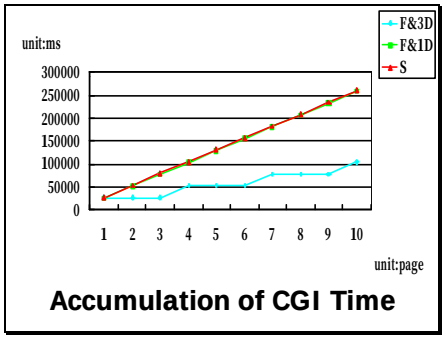

Figure 38

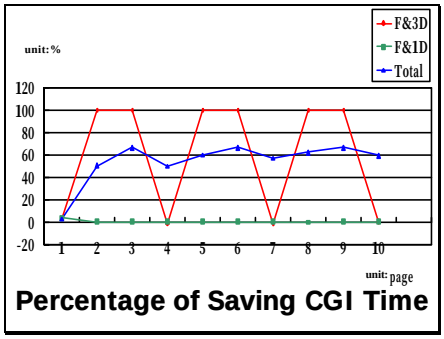

Figure 39

We can see the result of measure on size as following:

Unit:Byte

	1	2	3	4	5	6	7	8	9	10
F&3D	24305	0	0	15729	0	0	12614	0	0	3587
F&1D	16794	3700	3811	5888	4527	5314	3840	3967	4807	3587
S	17231	13036	13161	16515	14649	15244	13298	13450	14829	12973

Measure of Size

Table 5

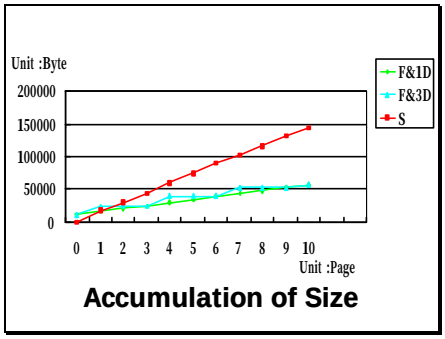

Figure 40

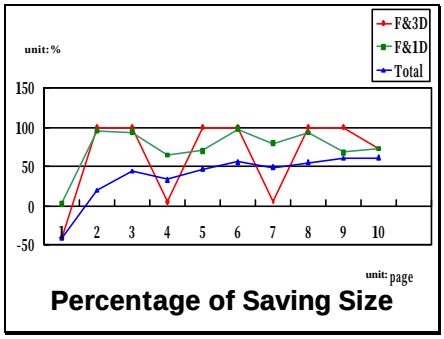

Figure 41

5.2 Handle, Send Data and Get a Response Page

We call ‘Tradition’ as getting response dynamic page in traditional way and ‘New’ as getting response dynamic page in new way.

We can see the result of measure on wait-time as following:

Unit:ms

	0	1	2	3	4	5	6	7	8	9	10	send
Tradition	0	880	930	1140	880	990	1100	1150	1120	1200	1150	0
New	660	0	0	0	0	0	0	0	0	0	0	880

Measure of Wait-time

Table 6

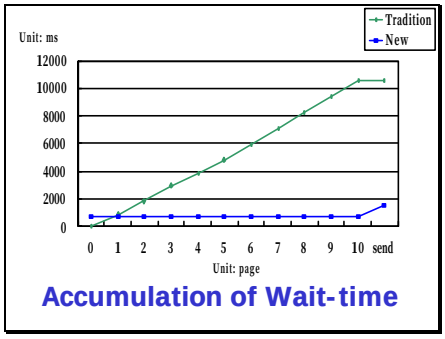

Figure 42

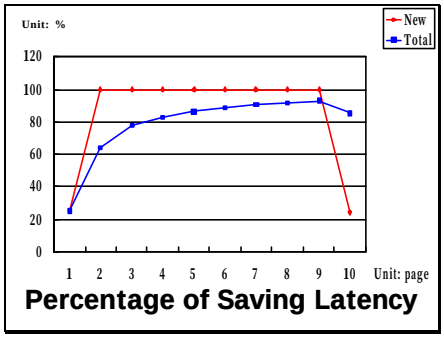

Figure 43

We can see the result of measure on web application server CGI execute time as following:

Unit:ms

	1	2	3	4	5	6	7	8	9	10	send
Tradition	51	59	62	72	50	73	50	53	51	72	0
New	0	0	0	0	0	0	0	0	0	0	75

Measure of CGI Time

Table 7

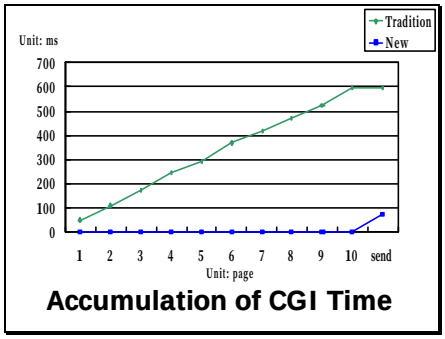

Figure 44

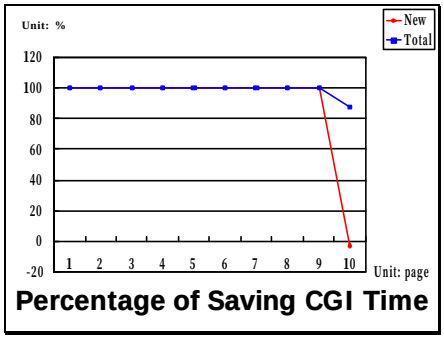

Figure 45

We can see the result of measure on size as following:

Unit:Byte

	0	1	2	3	4	5	6	7	8	9	10	**send**
Tradition	0	633	784	946	784	946	1107	1266	1107	1266	1429	0
New	1949	0	0	0	0	0	0	0	0	0	0	299

Measure of Size

Table 8

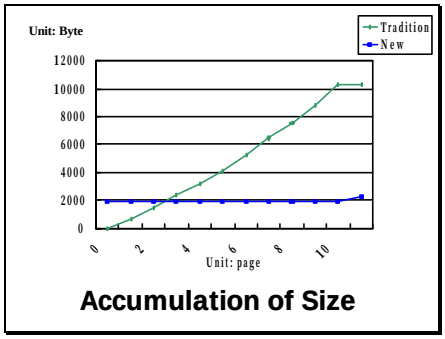

Figure 46

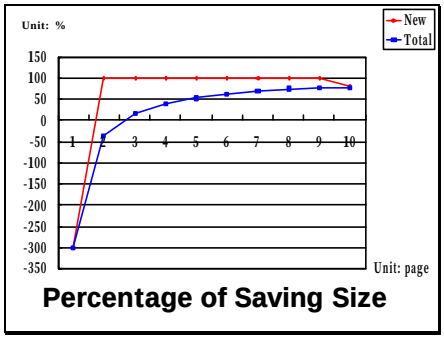

Figure 47

Chapter 6 Conclusion

We present a new method that can improve performance of web application server. The web application server will save network's bandwidth, reduce server loading and cut wait-time of end user more than the traditional web application server will. And we design and implement an efficiency web application server to measure the result of the new method. Finally, we compare the traditional method and the new method to prove that the new method really can save network's bandwidth, reduce web server loading and cut wait-time of end user.

For future work, we will attempt to divide automatically the common information from the source web page. And we will develop the method to be a standard. We hope the method can improve general existed problems in Internet.

References

[1] Anja Feldmann, Ramon Caceres, Fred Douglis, Gideon Glass, Michael Rabinovich, “Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments”, INFOCOM, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1999.

[2] Eric Levy-Abegnoli, Arun Iyengar, Junehwa Song, and Daniel Dias, “Design and Performance of a Web Server Accelerator”, INFOCOM, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1999.

[3] Andy Myers, Peter Dinda and Hui Zhang, “Performance Characteristics of Mirror Servers on the Internet”, INFOCOM, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1999.

[4] Michael Courage and Stephen Manley, “An Evaluation of CGI Traffic on WWW Latency”.

[5] Venkitachalam, G., Tzi-cker Chiueh, “High performance Common Gateway Interface invocation”, Internet Applications, IEEE Workshop, 1999.

[6] Jim Challenger, Arun Iyengar, and Paul Dantzig, “A Scalable System for Consistently Caching Dynamic Web Data”, INFOCOM, Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies, 1999.

[7] Shiwa Fu, Jen-Yao Chung, Walter Dieterich, Vibby Gotemukkala, Mitchell Cohen, and Shyhkwei Chen, “A Practical Approach to Web-Based Internet EDI”, Electronic Commerce and Web-based Applications/Middleware, 1999.

[8] Yankee, “The Internet: The MiniVan of EDI”, Group Enterprise Applications, Vol. 1, No. 11,March 1996.

[9] Klein, S, “A conceptual framework for the assessment of EDI”, System Sciences, 1992, Proceedings of the Twenty-Fifth Hawaii International Conference, 1991.

[10] Garry Froehlich, H. james Hoover, Wendy Liew and Paul G, Sorenson. “Application Framework Issues when Evolving Business Applications for Electronic Commerce”, In Proceedings of the 32nd Hawaii International Conference on System Sciences, 1999.

[11] James A. Senn, “The Evolution of Business-To-Business Commerce Models The influence of new information technology models”, WECWIS, Advance Issues of E-Commerce and Web-Based Information Systems, 1999.

[12] Zhong TIAN, Leo Y LIU, Jing LI, Jen-Yao CHUNG and Vibby GUTTEMUKKALA, “Business-to-Business e-Commerce Open Buying on the Internet”, WECWIS, Advance Issues of E-Commerce and Web-Based Information Systems, 1999.