An Intelligent Web Browser Plug-In for Automatic Translation to Ajax Approach

Hanakawa, Noriko

Abstract— ***We can receive large information from various web sites on Internet society. The web sites are important means to collect information in researching, learning, and even commercial activities. Importance of web browsers grows increasingly as web sites increase. Web browsers are powerful tools to refer to web sites. However, due to necessary of synchronous communication with web servers, operationality of web pages on web browser is not better than operationality of desktop applications. Therefore, we propose a new intelligent web browser plug-in for Ajax approach. The browser plug-in can improve operationality as same as desktop applications without revising program codes of web applications. A feature of the browser plug-in is asynchronous communication with web servers. As a result, we have confirmed improvement of operationality of "Yahoo auction sites" on the web browser plug-in when load of the Yahoo web server is heavy.***

Index Terms— ***Browser plug-in, Ajax approach, Asynchronous communication, JavaScript***

1. INTRODUCTION

RECENTLY, we have received valuable benefits from large amount of web sites on Internet. Much information is provided through web sites. Net-surfing is more important means to search valuable information. Moreover, web sites are useful in cases not only searching for information but also commercial activities such as Net-shopping and Net-auctions. Even education in universities and schools can not ignore information from web sites. Web sites are indispensable information resources on various scenes in current societies. On the other hands, most usual tools to refer to web sites are web browsers such as Internet Explorer (IE) and Firefox. Web browsers generate a visible web page while browsers are analyzing HTML source codes. This is a typical client-server system called "web application". Everyone can access easy web applications through web browsers.

Manuscript received February 21, 2007. This research was partially supported by KAKENHI, Grant-in-Aid for Scientific Research(C), 18500032, 2007.

N. Hanakawa is a associate professor at the Hannan University, Graduate school of Corporation information , Matsubara-si, Osaka Japan (e-mail: hanakawa@hannan-u.ac.jp).

However, we have problems of web applications on web browsers. Operationality of web applications on web browsers is less than operationality of desktop applications. Slow performance and limited interactivity of operationality on web application. Desktop applications mean that all programs are installed to a user computer. Desktop applications are able to achieve quick responses to user's requests because desktop applications do not need communication with servers on other computers, and waiting for calculations in server computers. Especially, if it takes long time for communicating with servers in web applications, a visible web page on a web browser may become a white-out page that has no texts and no images. The white-out page frequently occurs when a user requests "Reload" under heavy loaded servers. The white-out web page leads to not only no-better operationality but also users' confusions because users lose sight of operation on a web page.

To resolve the weakness of operationality of web applications, Ajax (Asynchronous JavaScript + XML) technology is adopted to web applications [1][2]. If a web application is constructed based on Ajax technology, white-out web pages will be avoided because client programs (e.g. JavaScript) in HTML source asynchronously communicates with web servers. Client programs in HTML source do not need to wait for finish of communication with servers. Because Ajax technology can improve operationality of web applications, many developers adopt Ajax technology to their projects. A most famous web application is Google Map [3]. The web application of Google map avoids status of white-out web pages because of asynchronous communication with web servers.

Although Ajax approach is a better way of improving operationality of web applications, there is a significant problem. Usually, developers have to change program codes of client software and server software of web applications. End users can not receive benefits from Ajax approach in existing web applications unless the developers modify current program codes of web applications. General end users only do nothing but waiting for modifying program

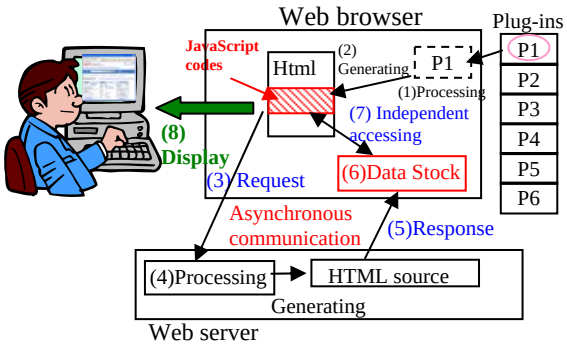

Figure1: A conceptual asynchronous communication model

codes of the existing web applications. However, there are a large number of web sites and web applications on the Internet society. Because opportunities to receive benefits from Ajax depend on progress of modification of the program codes, users need to wait long time for receiving benefits from Ajax approach in current web applications.

Therefore, we propose a new intelligent web browser plug-in based on an asynchronous communication model. The browser plug-in supports asynchronous communication with web server computers like Ajax approach. If users access web applications using the browser, users will be able to receive similar benefits from Ajax even if the web applications do not adopt Ajax approach. The browser has a function of translating automatically JavaScript codes that can communicate asynchronously with web servers. The function can improve operationality of web applications that do not adopt Ajax approach.

Section 2 shows related works, section 3 presents a basic idea of the browser plug-in, and the detail of the asynchronous communication model of the plug-in. Section 4 shows experiment results of comparing current normal web browsers such as Internet Explore. In section 5, we discuss usefulness and possibility of the browser. Section 6 summaries our research.

2. Related Work

Many useful Internet technologies have been proposed for improving performance of web applications. Sieminski focused on local system cache of Internet Temporary files [4]. Analysis of local cache is useful for not only estimating upper limit of caching efficiency but also measuring changeability of web objects. The changeability is derived from values of the cacheability factor (CF) of caches. Using the changeability, CF is a simple measure of susceptibility to caching and could be used to predict latency. Moreover, Li et al. also studied local system cache of Internet object for improving browser performance [5]. They proposed a peer-to-peer web document sharing technique, called a "browser-aware proxy server". To improve performance of loading web pages, a browser searches not only own local caches but also other computers' caches through network. Therefore, even if a user has never seen a web page, the web page will be able to be displayed on a browser using caches of other computers. Performance of displaying previous web pages is dealt with on both researches. Performance of viewing web pages that have once displayed on web browsers is improved. However, the browser can improve performance in updating partially a current web page that has never displayed anywhere.

On the other hand, intelligent web browsers have been proposed. Many useful functions are embedded to web browsers for improving user operationally. Bergasa-Suso et al. have proposed a new browser embedded a filtering function, a function of inferring learning style, and a function of recommending relevant web pages [6]. Shigesada et al. constructed a new type of browser called BKB (BTRON Kiosk Browser) that has a function of presenting WWW content through a user interface suited to kiosk terminal [7]. Matsuda et al. developed a graphical history browser. The purpose of the browser is to help a user to utilize Undo selectively and easily [8]. These browsers have valuable functions in a case of a specific situation. The browser is also a kind of these browsers including a special function. Each browser is suitable to each situation. When a user views a web page which is updated partially and frequently, the browser will be useful as same as these browsers.

In addition, Ajax approach can be adaptable to other useful techniques. Lei et al. have proposed a integrated system Ajax and Web service for Web-based cooperative image editing[9]. The system can search web components interoperable machine-to-machine interaction over a network. Castro et al. have proposed an integration Ajax approach with client state management services [10]. The client state management services can provide a single page that can possible modify its own presentation based on data exchanged with a server. The browser is also a result of integration of Ajax approach and browser techniques. Ajax approach can be integrated to various techniques.

3. A New Web Browser Plug-In

3.1. An Asynchronous Communication Model

Ajax technology has various possibilities of operationality on web applications. For example, "Google suggestion" [11] can generate a new candidate list as users input characters to a text box. In Google Map site [12], when users click on a map image of Google Map, only a part of map image is updated without reloading the whole web page. Users can continuously operate the map site without waiting for processing in Google Map web servers. Therefore, white-out web pages of the site never occur because client programs on a web browser do not need to wait

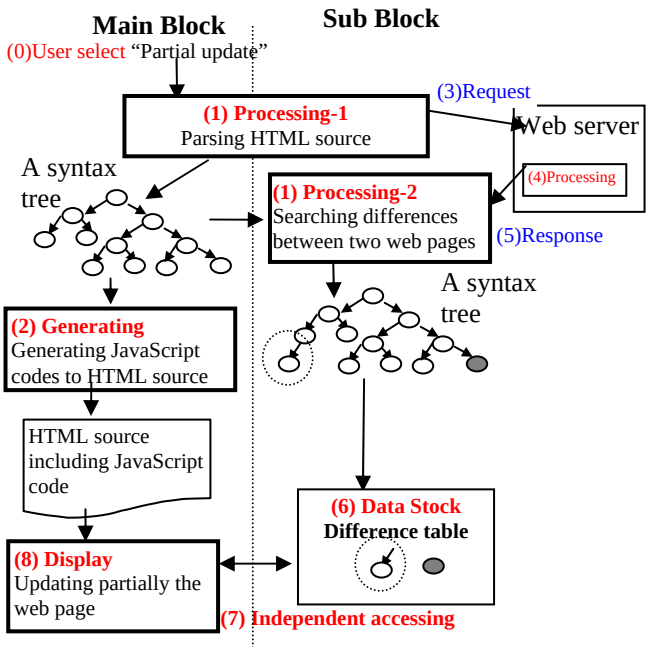

Figure2: An example of the model for partial updating

for communication with web servers.

The browser plug-in is based on an asynchronous communication model. The model can behave like various operations implemented by Ajax approach. A most important feature of the model is no change of client and sever program codes of web applications. Because the model is embedded to a web browser, developers of web applications do not need to revise the current web applications. In addition, the model can asynchronously communicate with web servers in order to achieve smooth operationality of web application on web browsers. The details of the concept of the model are shown in Figure 1. The typical steps are as follows;

(0) Selecting Plug-in: Users select a favorite Plug-in from supported plug-ins of a web browser.

(1) Processing of the Plug-in: The program of the selected Plug-in is processing in the browser.

(2) Generating JavaScript codes: By processing of the Plug-in, a new JavaScript codes are generated. The JavaScript codes can achieve functions of the Plug-in on specific web pages. The JavaScript codes are automatically embedded to original HTML source codes of specific web pages.

(3) Request: the embedded JavaScript codes request any function to web server. The requests occur by user operations, or automatic requests in the plug-in.

(4) Processing in web server: Corresponding to the request from the JavaScript codes, the web server is normally processing.

(5) Response: results of the processing in the web server are returned to the web browser.

(6) Data Stock: the returned results from the web server are stocked in Data Stock area. That is, the response from the server does not immediately reflect to the web page. At once, the response is accumulated to the stock area.

(7) Independent accessing: The embedded JavaScript codes access to the data stock area. Because the accessing is independent from response timing of the web server, the web page can behave like Ajax approach. That is, the web page does not need to wait for response from the web server.

(8) Display: By the execution of the embedded JavaScript codes, users can smoothly operate the web page like web application supported Ajax technique.

Various Plug-ins for Ajax approach can be prepared for web browsers. Each Plug-in has an original processing that generate appropriate JavaScript codes. The following sub-section shows a function of updating partially a web page without reloading wholly. Even if users click "Reload" button, the web page avoids updating wholly a web page.

3.2. The Model for Partial Updating

Figure 2 shows the asynchronous communication model for partial updating a web page without wholly reloading. The model of Figure 2 is established in accordance with a conceptual asynchronous communication model of Figure1. "(1) Processing" of Figure 1 consists of two processing in Figure2, "Parsing HTMLsource" and "Searching differences between two web page". "(2) Generating" of Figure1 is "Generating JavaScript codes" of Figure2. "(6) Data Stock" of Figure 1 matches with "Difference table" of Figure 2. "(8) Display" of Figure 1 matches with "Updating partially the web page" of Figure 2. The following sub sections describe each part of the model.

3.2.1. Processing-1 (Parsing HTML Source)

HTML source is analyzed by a parser based on HTML tags. Syntax trees are generated by the parser. Nodes of syntax tree mean HTML tags, leaves of syntax tree mean text contents such as texts, numerical, and JavaScript codes. Because the analysis is based on HTML tags, other elements such as Script codes are dealt with as general texts (leaves of syntax tree). In addition, to update partially a web page, a new tag's property named "ID" is inserted to each node of syntax trees. When a web page is updated partially, the browser distinguishes update parts from non-update parts of web page using "ID" property.

3.2.2.Generating (Generating JavaScript Codes)

An original HTML source does not have a function of partial updating. Therefore, new JavaScript codes are added to the original HTML source. JavaScript codes refer to a difference table of "Data Stock". JavaScript codes include "innerHTML" method. Figure 3 shows an example of JavaScript codes. The JavaScript codes consist of 2 functions, LoadText() and

```
1  var change;
2  var loader = new Jamritas.Loader;
3  var changeTxtArray = new Array();
4  function loadText(){
5
6          loader.loadtext('difference.txt', function(text) {
7          change = text;
8
9          });
10         if(change == null) {
11
12                 return; }
13         var txtArray = change.split("¥n");
14         for(i = 0;i < txtArray.length; i++) {
15                 var txtTemp = txtArray[i].toString();
16                 var txtArray2 = txtTemp.split("@:@----------@:@
");
17                 changeTxtArray[i] = new Array();
18                 changeTxtArray[i][0] = txtArray2[0];
19                 changeTxtArray[i][1] = txtArray2[1];
20         }
21 }
22 function changeHTML(){
23         loadText();
24         for(k = 0; k < changeTxtArray.length;k++) {
25         if(document.all("fontID" + changeTxtArray[k][0]) != null)
           {document.all("fontID" + changeTxtArray[k][0]).innerHTML
26             =changeTxtArray[k][1];}
27         }
28         setTimeout("changeHTML()",1000);
29 }
```

Figure3:JavaScript codes in HTML source

ChangeHTML(). The function of ChangeHTML() is called when a user selects “update partially” plug-in. The function of ChangeHTML() calls a function of LoadText() (See 23rd line of Fugure3). The function of LoadText() searches a difference table of “Data Stock”. The file name of the difference table is “difference.txt”(See the 6th line). A reason of usage of a file "difference.txt" not on memory is that JavaScript program on HTML can not directly access memory areas of the browser. The information of the difference table saves to a two -dimensional variable named “changeText Array[][]” (See from 17th line to 19th line). The value of “changeTextArray [k][0]” means “ID” property given to each node of syntax tree, the value of “changeTextArray [k][1]” means a new text having “ID” property. The new text is a leaf of syntax tree. In 25th line, using document. all().innerHTML” method, partial updating actually execute. The tag having “ID” (change TextArray [k][0]) of HTML source is modified by the new property (changeTextArray[k][1]). At a moment of execution of “document.all(). innerHTML” method, the web page has been updated only tags having specific values of “ID” propoerty. In addition, the function of Change HTML() includes “setTimeout()” method at the last line of the JavaScript codes. The function of ChangeHTML() is repeated at constant intervals. In short, the function of ChangeHTML() refer to the different table at constant intervals, after that, if there are some difference between a current web page and a following web page, the two-dimensional variable is created. By the two-dimensional variables, “document.all(change TextArray[k][0]). innerHTML” method runs.

Because “ID” properties have been assigned to nodes of syntax trees, common JavaScript codes are available to various HTML sources. A “JS” file including the JavaScript codes in Figure 3 is prepared beforehand. New line that should be inserted to an original HTML source is only “<SCRIPT onload=xxx.js></SCRIPT>”. After that, the browser wholly updates once again the web page including the JavaScript codes. The “changeHTML()” runs at constant intervals. The value of constant interval can change in an option menu of the browser plug-in. The various intervals can be set according to various web pages. For example, at near the closing time of Yahoo auction, interval time is set to 1 second, because the value of “current bid” on the web page may be frequently updated.

3.2.3. Processing-2

With comparing two syntax trees, difference between two web pages is detected. There are many researches about difference calculation between two tree-structures. In the model, xmdiff algorithm [13] is adopted to detect difference. In the xmdiff algorithm, delta scripts (insert, delete, and update) and costs are used. The delta scripts mean construction process of a tree. The costs mean weights of the construction process using the delta scripts. Edit Graph (dynamic programming) is achieved using the delta scripts and costs. Nodes of one tree in depth first searching are plotted to on x-axis of the Edit Graph. Nodes of another tree in depth first searching are plotted to on y-axis of Edit Graph. Horizontal members of Edit Graph mean “delete” of a node. Vertical members of Edit Graph mean “insert” of a node. The hypotenuse members of Edit Graph indicate “update” of a node or leaves. Each edit script (delete, insert, and update) is set to each weight named “cost”. Path with minimum cost among plots on Edit Graph means a most adaptable delta scripts. According to adaptable delta script, different nodes and leaves are determined. If you want to see details of xmdiff algorithm, please see [13].

In our model, cost of each edit script is defined as; “Insert” script is costI, “Delete” script is costD, “Update” script is costU. In addition, we have to determine whether a following web page wholly updates. If difference between two web pages is large, partial updating of a current web page is meaningless. Therefore, in searching algorithm, we set a threshold value for judgment of whole updating. If value of sum of costs on minimum path in Edit Graph is greater than a threshold value, the searching algorithm selects “wholly updating”. The judgment is the following;

$$\frac{\sum(\cos tI + \cos tD + \cos tU)}{node} > A \quad then \quad whole \quad updating, \quad (1)$$

cost: cost of edit script such as “insert”, “delete”, “update”

node: the total number of the node of two syntax trees.

A : threshold value

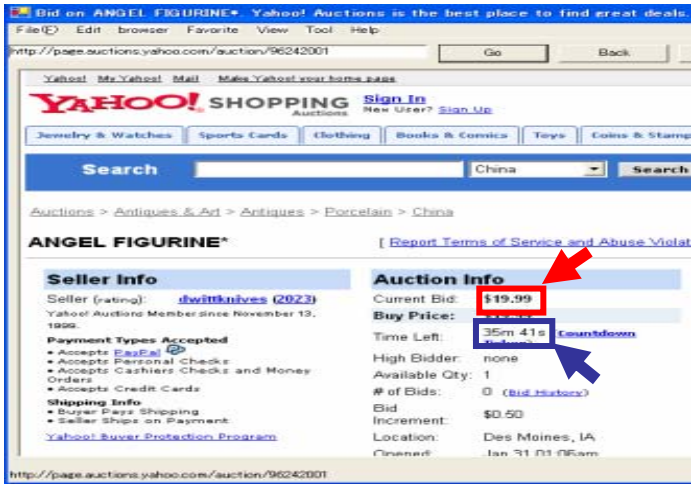

Figure4. The browser displaying USA Yahoo auction web page

After that, if partial update of the web page is required, a difference table is generated. The difference table includes difference node information such as tag name, ID, tag's properties, text.

3.2.4. Display

By the difference table, a current web page embedded JavaScript codes can partially update. JavaScript codes refer to the different table. Only elements of HTML tags that are set to the different table are refreshed by "InnerHTML()" method of JavaScript codes. Users can continue to operate a web page without user's confusion caused by wholly refreshing a web page.

The semantic inconsistencies may occur rarely. However, usually, the constant interval of accessing web server is more than the constant interval of accessing the different table. The two intervals' values can be set by users using optional menu. If large inconsistencies occur, that is, a web page can not display for HTML error, users will stop "partial-update", then, select "refresh" wholly.

3.3. Implementation of the Browser Plug-In

Based on the asynchronous communication model for partial updating, a new web browser plug-in has been implemented in C# language. Of course, various plug-ins should be prepared. However, in this paper, a plug-n for partial updating is implemented, and evaluated. Figure 4 shows a screenshot of the web browser including partial updating plug-in. The browser has especial buttons, "Partial-update", "Auto-update". At once, the browser displays normally a web page. If a user improves an operationality of the web page, user clicks the "Partial-update" button. The HTML source of the web page is modified in the asynchronous communication model mentioned above. A JavaScript code "<SCRIPT onload= xxx.js></SCRIPT>" is added to the original HTML source. Based on the modified HTML source, the web page is wholly updated once again on the browser. The new JavaScript code is started in the web page using "SetTimeOut" method (See the last line of Figure3). Therefore, in the JavaScript code, the web page refers to the difference table at constant intervals.

After that, when user clicks "Update" button, the browser requests a following web page to a web server. The following web page is analyzed in the asynchronous communication model. A syntax tree of the following web page is generated in the browser. Next, a difference table is generated in the browser. Because the modified web page (the original web page added JavaScript codes) is accessed to the difference table, the web page is partially updated. In Figure4, only a red rectangular area is updated without wholly updating.

In addition, if the web page is frequently modified such as Yahoo auction web page at near the closing time, users can select "Auto-update" mode. The browser requests automatically a next web page to a web server at constant intervals. Therefore, the browser updates partially the web page without users' clicking "Update" button, automatically.

If the browser judges a whole updating for a following web page, the following web page has few relationships with the current web page. For example, when a following page moves to other URL link, difference of two syntax tree becomes large. Analysis at the model judges whether wholly updating is better than partial updating. This case is caused almost by changing URLs. Although change of URLs can indicate "wholly updating" of a web page, sometimes URL has parameters to web servers such as "http//portal.hannan-u.ac.jp/login.do;jsessionid=893E". In this case, the URL is different from the previous URL, although the following web page has changed partially. Therefore, the browser can not judge wholly updating using only change of URLs. Of course, the judgment of wholly updating is available at "On" status of "Partial-update" button on the browser.

4. EXPERIMENTS

4.1. Updating Texts in Yahoo Auction Sites

In Yahoo auction site [14], we have confirmed partial update of a web page on the browser plug-in. Figure 4 shows a web page of Yahoo auction. "Angel figurine" item is exhibited to Yahoo auction. Seller is "dwittknives", current bid is "$19.99". In the web page, the value of time left such as "35m 41s" (See blue rectangle area of Figure 4) changes without "Reload" event because the value of time left is written in JavaScript codes in original HTML sources. However, the value of current bid such as "$19.99" does not update because "$19.99" is a simple text content from the web server. The

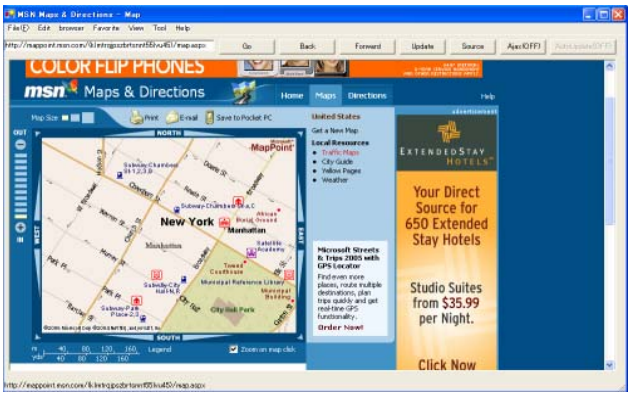

Figure5: USA's MSN map site on the browser

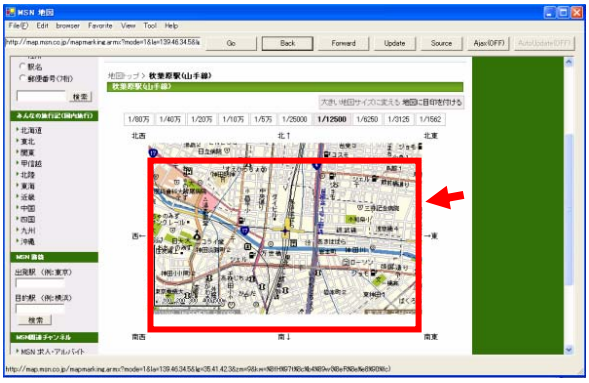

Figure6: Japanese MSN map site on the browser

value of "$19.99" is not updated as long as user doesn't click "Reload" button on a browser. When a user clicks "Reload" button, a whole web page is updated even if a value of current bid did not change. A flickering of the web page occurs when the web page is wholly updated. Therefore, the web page of Yahoo auction is displayed on the browser. When a user clicks the "Partial-update" button on the browser, only value of current bid is updated without wholly updating. Of course, if the value of current bid does not change, nothing occurs on the web page.

In addition, if a web page is modified frequently such as Yahoo auction web page at near the closing time, users can select "Auto-update" plug-in. Just before the closing time, many bidders bid for a bidding item. Therefore, although almost parts of the auction web page do not change, only value of current bid is frequently updated. If a user clicks "Auto-update" button on the browser, only value of current bid is updated automatically and continuously. In short, although Yahoo auction site's programs have not adopted Ajax approach, Yahoo auction site was able to behave like a web page adopting Ajax approach on the browser.

4.2. Updating Images in MSN Map Sites

Map sites on Internet are useful. However, even if a user wants to move a central of map, the web page is wholly updated. Because the web page including map images flickers by the whole updating, users often lose a sight of a specific point on a map image. Because Google map site adopts Ajax technology, the operationality of the map site is smooth. However, the other map sites such as Japanese MSN map do not adopt Ajax technology.

For example, Figure 5 shows a web page of USA's MSN map site adopted Ajax technology. When a user clicks at a point of the map image, only the map image is updated without updating the whole web page. In contrast, in Japanese MSN map site (See Figure 6), when a user clicks at a point of the map image, the whole web page including not only the map image but also all contents of the web page is updated. The operationality of Japanese MSN map site is less than the operationality of USA's MSN map site. Therefore, we display a web page of Japanese MSN map site on the browser. We expect that only the map image (See the red rectangle area of Figure 6) is updated when a user clicks at a point of the map image. As a result, the area of the map image is updated with no-updating the other contents.

5. DISCUSSION

A most important feature of the browser plug-in is to improve operationality of applications that was not developed under Ajax approach. The proposed model implemented to a web browser can asynchronously communicate with web servers. In this paper, the model for partial updating is implemented as a plug-in of a web browser. The browser avoids flickering a web page by "Reload" and "Update". However, the proposed asynchronous communication model has some problems. In this section, we discuss problems and possibility of the plug-in based the model.

5.1. Performance On a Heavy Load Server

Although the browser plug-in can partially update without revising programs of web applications, we focus on two performance problems, speed of partial updating, and correctness of judging the whole web page. At first, we evaluate updating time of a web page under comparing with the existing web browser Internet Explorer. Of course, because Internet Explore displays web pages in normal, the web page is wholly updated. The conditions of the experiments are as follows;

(1) The web server is Apache ver.1.3.4

(2) The web pages of Yahoo auction site are downloaded to local files of a home directory of the web server.

(3) The updating time means a period from an event occurring time to finishing time to update partially or wholly.

(4) Comparing three types; IE's updating, partial-updating on the browser, and auto-updating on the browser.

(5) The server is given various loads by other tasks. The tasks are the downloading files using Wget software [15].

(6) The variation of the loads is generated by

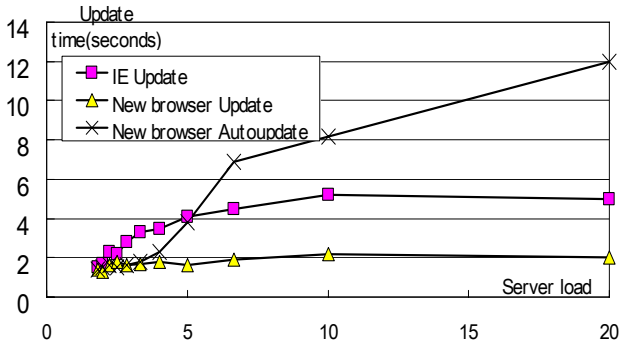

Figure7: Performance of the browser

the number of executions of the tasks from 2 times per 10 seconds to 20 times per 10 seconds.

(7) To compare performances, caches (Internet temporary files) for Internet Explorer are cleared.

(8) When “Auto-update” is ON in the browser, the browser requests the web server at one second intervals.

Figure 7 shows results of the experiments of speed performance of updating time. The x-axis of Figure 7 means the server load. The unit of the X-axis is “the number of file-downloading by Wget”(see the above(5)) per 10 seconds. The y-axis of Figure 7 means the updating time as mentioned above (6). When the load is between 2 times and 5 times, the effect of the browser is clear. Updating on the browser is quicker than updating on IE. In addition, updating on IE in 8 times or more loads leads to a white-out page. However, our web browser leads to no white-out page even if the load is large. In auto-updating on the browser, when the server load is bigger than 5, the performance suddenly decreases. It is because that sum total of server load of the other tasks of file-downloading and request from the browser became more than performance limitation of the web server. We confirm that auto-updating of the browser is also not suitable when server load is large.

Next, we discuss correctness of judgment of updating wholly. In the experiments, according formula (1) in section 3.2.3, “Insert” script’s cost is set to 2.0, “Delete” script’s cost is set to 2.0, “Update” script’s cost is set to 0.5, and the threshold value is set to 0.3. When 30 % of HTML source is different between a current web page and a following web page, the current web page should be wholly updated in replacing to the following web page. In Yahoo auction site, and MSN map site, the correctness of judgment is 70%. The correctness of the other web sites such as MSN top page is less than 50%.

The miss-judgment of wholly updating is caused by meaningless contents of a web page. For example, meaningless contents are codes for displaying commercial advertisements, blank code such as “ ”, carriage return code such as “\n”, and comment lines in HTML source. Especially, many advertisements of a web page strongly lead miss-judgment. Therefore, a

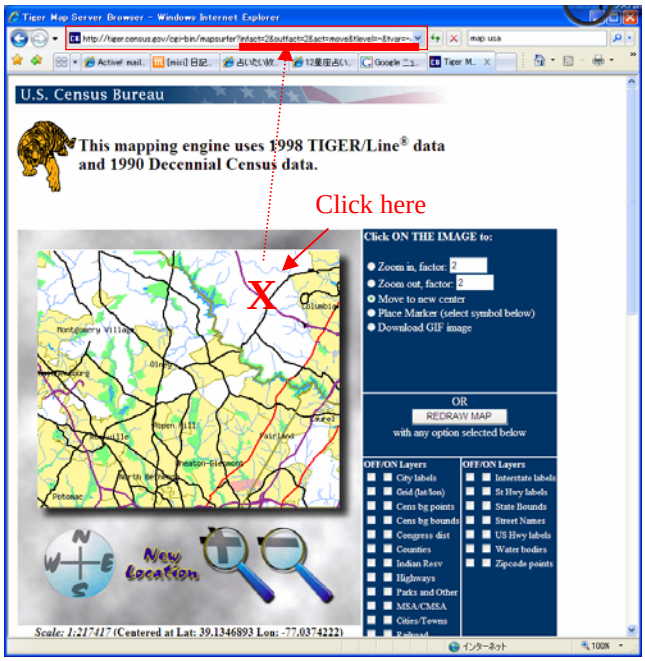

Figure8: Location information in U.S. Census Bureau map site

function of omitting meaningless contents such as advertisements will be implemented to the browser plug-in. Moreover, our judgment way may be more applicable to web pages including specific targets such as auction sites than general information web pages such as top page of MSN. However, because of various types of web pages, we have to improve the values of the costs and the values of the threshold, and the way of judgment of whole updating.

5.2. Performance On No-load Server

We discuss performance of the browser plug-in when server’s load is a little. When “Partial-update” button becomes “On” status in the browser, the additional calculations for making syntax trees, detecting difference based on xmdiff algorithm is required. Of course, the calculations need certain time. Other web browsers such as Internet Explorer do not need such calculation time. Therefore, speed performance of the browser is lower than normal browsers when users click “Partial-update” button on the browser. Especially, when server load is a little, low performance of the browser is clarified. Time to wait for communication with servers in normal browsers is less than time to wait for calculation on the browser. This is demerit of the browser including the plug-in.

However, a most important feature of the browser plug-in covers the demerit of speed performance. A most important feature is to improve operationality of web application. Especially, when a web page has no change between a previous web page and a following web page, nothing occurs on the web page on the browser. In addition, even if the web page changes a little, almost all parts of the web page do not change on the browser. The flickering of meaningless of a web page is avoided. Although

Figure9: Interactive map with Percent of Occupied Housing Units in U.S. Census Bureau map site

the calculation in the browser plug-in is required, users can usually continue to operate the web pages on the browser. Because communication with servers and the calculations (making tree, xmdiff algorithm) in the browser are independently executed from response of the servers (See Figure 2), users will not feel significant inconvenience caused by synchronous communication with servers. Users can operate a web page as same operation as web applications adopted by Ajax technique.

5.3. Possibility of Map Sites With Interactive Operation

Recently, almost map sites have been supported Ajax approach. Operationality of map sites is smooth using “drag and drop” operation by a mouse. However a few map sites need the browser plug-in. For example, U.S. Census Bureau provides “Tiger Map Server” (http://tiger.census.gov/cgi-bin/mapbrowse-tbl/, See Figure 8). A web page including map image on the Tiger Map Server refreshes wholly when a user clicks a point of a map image. A new location of the following map is “input” tag of HTML source. If a user clicks at a point of a map image, the following request to the web server is based on “input” tag’s information. The “input” tag has “src” property of URL address indicating a current image map. The URL has map location (latitude and longitude) like “39.1346893”. An example of the “input” tag as follows;

<input type=image name=map src=http://tiger.census.gov/cgi-bin/mapper/map.gif?lat=39.1346893&lon=-77.0374222>

Therefore, the difference between a current web page and a following web page is values of the map locations and the URLs.

However, because almost popular map sites such as YahooMap, GoogleMap, and MultiMap have been already supported Ajax approach, the above technique like “Tiger Map Server” is meaningless. Therefore, the other possibility of the above technique is discussed. Figure 9 shows a USA map with percent occupied housing units from U.S. Census Bureau. Map images are displayed with statistical data resources of the percent occupied housing units. Graphical images of the statistical data on a map are more understandable than a numerical data in a textual table. Users can grasp intensively a rough tendency of statistical data. Especially, if the statistical data is related with location of a map, the usefulness of graphical statistical data on a map will increase.

However, the combination map site such as Figure 9 is not yet adapted to Ajax approach. For example, the web page including the map with percent occupied housing units such as Figure 9 is refreshed wholly even if a user would like to move a little the map image. Flickering of the web page occurs. Therefore, the web page is displayed on the browser. Only map image refreshes without whole flickering of the web page. Operationally of the web page is improved by the browser.

In future, such combination web site of statistical data with maps will increase because graphical understanding of web pages is better than textual understanding. Engineers of statistical data web sites will be able to avoid being worried about a new technique of Ajax by the browser.

6. CONCLUSION

We have proposed a new web browser plug-in. The browser plug-in is based on an asynchronous communication model. The model can establish smooth operation like web application adopting Ajax approach. In addition, a plug-in function for partial updating is implemented to a web browser. Because the browser plug-in asynchronously communicates with web servers, partial updating a web page is possible while user continue to operate the web page. As a result of experiments, a web page of Yahoo auction, and a web page of MSN map can be partially updated on the browser plug-in. Moreover, when a server load is heavy, benefits of the browser plug-in have been recognized. In future, speed performance of the browser plug-in will be improved, at same time the correctness of the judging whole updating will be improved.

REFERENCES

[1] Paulson, L.D., “Building rich web applications with Ajax”, Computer, IEEE, vol.38, no.10, Oct. 2005, pp.14-17.
[2] Smith, K., "Simplifying Ajax-style Web development," Computer , vol.39, no.5, pp. 98-101, May 2006
[3] http://maps.google.com

[4] Sieminski, A., “Changeability of Web objects - browser perspective”, Proceeding of the 5th International Conference on Intelligent Systems Design and Applications, Sept. 2005, pp.476- 481.

[6] Bergasa-Suso J., Sanders D.A.. Tewkesbury G.E., “Intelligent browser-based systems to assist Internet users”, Education, IEEE Transactions on , vol.48, no.4, Nov. 2005, pp. 580- 585.

[7] Shigesada Y., Koshizuka N., Sakamura K., “A Kiosk WWW browser for digital museums”, Systems and Computers in Japan, vol.34, no.13, 2003, pp.59-70.

[8] Masuda H., Imamiya A., “Design of a graphical history browser with Undo facility, and visual search analysis”, Systems and Computers in Japan, vol.35, no.12, 2004, pp.32-45.

[9] Lihui Lei; Zhenhua Duan, "Integrating AJAX and Web Services for Cooperative Image Editing," IT Professional , vol.9, no.3, pp.25-29, May-June 2007.

[5] Li X., Xiaodong Z., Zhichen X., “On reliable and scalable peer-to-peer Web document sharing”, Proceeding of Parallel and Distributed Processing International Symposium, 2002, pp.23-30.

[10] Castro] Castro, P.; Giraud, F.; Konuru, R.; Ponzo, J.; White, J., "Before-Commit Client State Management Services for AJAX Applications," Hot Topics in Web Systems and Technologies, 2006. HOTWEB '06. 1st IEEE Workshop on , pp.1-12, 13-14 Nov. 2006

[11] http://www.google.com/webhp?complete=1&hl=en

[12] http://maps.google.com

[13] Chawathe S., “Comparing hierarchical data in external memory”, Proceedings of the 25th International Conference on Very Large Data Bases. Edinburgh, Scotland, U.K., 1999, pp.90-101.

[14] http://auctions.yahoo.com

[15] ftp://gnjilux.cc.fer.hr