Web-Based Plankton Data Visualisation

T.T.Ho and P.S.Dowland

Centre for Information Security and Network Research,
University of Plymouth, Plymouth, United Kingdom
e-mail: info@cscan.org

Abstract

This paper briefly presents the findings from a project to build a web-based system to draw abundance maps, visualising plankton data from the Continuous Plankton Recorder (CPR) survey undertaken by the Sir Alister Hardy Foundation for Ocean Science (SAHFOS). The system implements a rich client application which draws maps from specific structured data transmitted from server side web service. An AJAX based model has been built for the whole system to provide better user interaction and features. A client cache is also implemented as a component in that model.

Keywords

Web CPR, SAHFOS, plankton, data visualisation, client cache, script cache, client drawing, browser drawing, AJAX, XML, JavaScript, web service, ASP .NET

1 Introduction

Since 1931, millions of plankton samples of biogeography and ecology of plankton from the North Atlantic and the North Sea has been collected by the Sir Alister Hardy Foundation for Ocean Science (SAHFOS) under Continuous Plankton Recorder (CPR) survey on a monthly basis (SAHFOS, 2006). Several windows and web applications, named WinCPR and WebCPR respectively, have been developed for a few years to assist scientists around the world in analyzing this growing data. The programs make seeing and analyzing the data easily by many kinds of visual charts.

The project Web-Based Plankton Data Visualisation contributes to the development of WebCPR site. Performance and interaction are the main targets of the project. The current underdeveloped WebCPR site bases totally on server-side processes, heavy-work load is put into a single centre system and both requests and responses between client and server consume a lot of network traffic. Moreover, because of limits of text-format-based HTML standards, user interactions are inconvenient. Current version of WinCPR also has limited interactions features.

The project concerns about North Sea databases and involves various experiments on client drawing, data structure and transfer, application architecture to provide better services for analyzing abundance maps.

2 Web technologies overview

World Wide Web uses HyperText Transfer Protocol (HTTP), a stateless protocol, and presents information in HyperText Markup Language (HTML), which uses text based standards. Therefore, human interaction and presentation is significantly limited, especially in comparison with standalone programmes. For example, every time user wants to get detailed information in a small piece of a page, he has to wait until the entire page is reloaded from web server. Since HTML file is simply a static text language and all interacts are based on hyper links and forms, the way to interact lacks of flexibility. In addition, the simplicity of HTTP protocol and HTML also leads to the difficulty of implementing complicated data presentation. Current WebCPR application is using this approach. HTML contents and images are generated from server side and transferred to client side to replace previous page, as a result of a user request.

Web contents are usually created by the use of traditional three tier application architecture, which is also applied to current development of WebCPR. In this architecture, a system is divided into three tiers with different functions. The first tier is client which presents information for user and also receives input. The second tier is web application tier which generates the content of the requested page and interacts directly with client. Finally, the last tier is data tier which stores data and responds to any data request (for retrieving or modifying) from application server. The architecture has several crucial advantages (Ramirez, 2000) so that it has been being applied for web development for a long time.

Because of the limited ability and low resource requirement of web browser, it is sometimes called a “thin client”. It maintains strong flexibility for end-user devices and provides compatibility and portability for web, which is a crucial factor of the Internet. It also puts all application (or business) process on one central point so implementing a system is easier. On the other hand, it limits the use of growing processing power in today clients. To compensate, it places heavy workload on web server, which sometimes challenges developers.

In order to compensate to these drawbacks, several refinements have been built, although the basic concept of three-tier model is still kept. One of them is Plug-in Object Approach, in which one or several objects are added to traditional HTML content in order to give content sent back to client processing abilities. The embedded objects are independent entities so they do have no relation or interaction with HTML content and theoretically they have limitless abilities of processing, presenting and interaction. However, browser is intended to HTML standards so it natively supports no embedded objects. Hence, a “plug-in” component needs installed to extend browser’s functions. The most popular embedded object solutions seem to be Adobe Flash (Millward Brown, 2007) and Java Applet.

Another refinement, which recently develops rapidly, is Scripting Approach. In this approach, when user interacts with HTML page, underlying script is activated and changes HTML content to present new suitable page. The browser immediately updates the outlook of the page. Two main missions on client side can be obtained by script are underlying process behind user interface and modifiable HTML content.

Unlike embedded object, script is natively supported by browser and furthermore, it is similar from one browser to another. Therefore, it is more flexible to use and easier to be accepted by client. Moreover, basing on HTML, script keeps web page conception, including the way a page is designed and the way user interacts with the page. In development aspect, scripting approach requires source code of script to be sent back to client so it reveals all underlying mechanisms. More importantly, script is difficult to implement and maintain. Furthermore, because it bases on HTML elements in a page, it has a tight coupling association with those elements. Coupling association makes maintenance of code more difficult.

A lot of technologies have been being developed to overcome weaknesses of this approach. The most well-known one tends to be Asynchronous JavaScript and XML (AJAX). According to his famous article, Garrett (2005) defined AJAX as a combination of a number of technologies such as HTML, CSS, DOM, XML, XMLHttpRequest and JavaScript. In this model, AJAX page roles a client application interacting continuously with user, removing all interrupted times between requests and responds. All exchanges with server are hidden behind AJAX engine, written in JavaScript, by asynchronous XMLHttpRequests. Furthermore, as mentioned above, script can generate HTML and CSS content; only necessary raw data is needed to be transferred from server so both web traffic (then latency) and server workload is reduced. Some AJAX frameworks have both server and client parts coupling tightly together, some have only client part containing all client controls (Mesbah and van Deursen, 2007a). AJAX user interface is usually developed in a familiar way to graphics use interface (GUI) systems so it becomes much more convenient for GUI application developers (Mesbah and van Deursen, 2007b).

However, there are several clear disadvantages when scripting approach is used for client side image creation. Because it is implemented to manipulate HTML content but no other file types or other special elements, it has no ability to generate true image or drawing objects. Besides, reliance on web standard formats limits scripting codes and web services to efficiently structure data transferred in the network. HTML and XML are used to describe data and unfortunately these text-based languages are for easiness, comprehensibility and compatibility instead of efficiency.

On server side, web framework and platform are usually used to develop applications. They provide essential components for complicated services and simplify web development (Shan and Hua, 2006). One of the most popular are ASP .NET and Java technologies. ASP.NET is considered easier to implement Model View Controller pattern than Java technologies (Masoud *et al.*, 2006). Model View Controller is an architectural pattern used for separating user interface from business logics.

3 A new model

ASP .NET and ASP .NET AJAX are chosen to develop the project and therefore AJAX technologies and development concepts are taken into the main points of development. Assisted by technologies, a model for the whole system, which is shown in the figure below, is built and several web architectures as well as popular application architectures are reflected into it.

The model looks slightly different from three-tier model since databases are put into server side services part. It is because transactions between server side web services and database system are not currently covered by the research so database is kept into the web project to simplify the model. However, the model in fact follows three-tier model as there is no direct connection between client and databases. Hence, databases can be easily moved out to a dedicated tier.

The system is a composition of two joining applications: one is a server side web application, containing several services as is seen in traditional web applications, and the other is a client side application. Unlike traditional applications, except for initiating user interface, the server side application plays passive roles in the whole system and put active actions into the client side application. This design is significantly similar to desktop based or plug-in object design for network applications, not only in the outlook but also in deep details and functions. It is because the existence of a client side application. The main difference is the platform where the client side application runs. For a desktop based application and plug-in object, the platform is the operating system and a plug-in library (pre-installed into web browser) respectively, whereas in this design, it is the browser itself and its sandbox. The two key technical points to maintain this similarity are abilities of client technologies, with the help of AJAX framework, and one unified client page.

Further discussions of several important components are given in the below section.

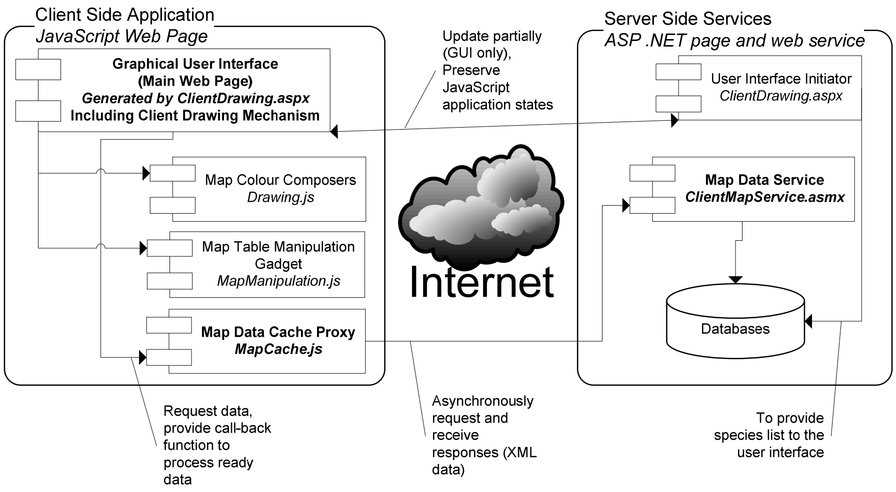
Client Side Application
JavaScript Web Page
Graphical User Interface (Main Web Page) Generated by ClientDrawing.aspx Including Client Drawing Mechanism
Map Colour Composers Drawing.js
Map Table Manipulation Gadget MapManipulation.js
Map Data Cache Proxy MapCache.js
Update partially (GUI only), Preserve JavaScript application states
Internet
Server Side Services
ASP .NET page and web service
User Interface Initiator ClientDrawing.aspx
Map Data Service ClientMapService.asmx
Databases
Request data, provide call-back function to process ready data
Asynchronously request and receive responses (XML data)
To provide species list to the user interface

Figure 1 - Overview of System model

4 Important components

Because transferring image requires the image exists on server side, it consumes server resources to generate the image. In addition, in order to keep network bandwidth low and also as a requirement of web standards, an image compression algorithm should be used to reduce the size of the image. Compression algorithm is always complicated and consumes a lot of processing power. Besides, because of compression techniques, image quality usually decreases. Image transfer also reduces flexibility of client-side application. In plankton map application, client application can give two types of flexibility, relating to data structure. The first is the interaction of user with the map, such as showing pixel values when user points mouse on. Secondly, user may want to change the way map is presented. Changing colour mapping method is a useful example. Therefore, the new system moves image drawing to client side and as a result, raw data is transferred from server to client, instead of image presenting it.

Theoretically, sending necessary data to client instead of its presentation would be more efficient in network traffic manner. Because data can be organized in a condensed form, it saves network traffic. However, XML is specifications web service should follow to exchange data between client and server. The language is text based but not binary based. Text based data is apparently not as condensed as binary data since it is the language for human readable, which is in a long form. Besides, XML is a mark-up and self-descriptive language, which uses tags and other sophisticated conventions to describe data fields and also their meaning.

The second problem of data organization for scripting approach is that it depends completely on web standards and web support from third parties, such as web browser to run client script. Therefore, one solution is to organize data structure in the way that code interpreting data can be written to be compatible with any browser.

Because of those reasons, text based transmission is chosen for the project. Additionally, if an array of pixel values is transferred directly to client side with XML format, the size of the response packet will be considerably large because of XML descriptions overhead, as explained above. To sort out this problem, all pixel values are put into a string (of characters) and then the whole string is formatted into an XML document. Furthermore, bandwidth can be saved more by using proper format of each value, instead of human readable number as in XML standards. XML uses decimal base (radix) to export numbers but larger base is more efficient. On the other hand, in order to make sure that the chosen base is supported by JavaScript interpreter of all browsers, it should be a technical standard base. Therefore, hexadecimal is used to describe each pixel.

On client side, although receiving and interpreting pixel values is pretty simple, drawing them into a map faces a vital issue. Client side script is designed to manipulate web page elements so it is given no general power like server side framework. Therefore, creating new type of data, file is impossible from client side code, especially in high security user devices. Applied to drawing maps, client side code can create no true image. Additionally, there are no HTML elements designed for graphics purposes. To work around, HTML elements with such formats as colours, borders, position, and size are used to mimic pixels. Fortunately, plankton abundance maps mainly consist of a few block pixels, arranged in rows and columns. Therefore, use of a table with well aligned cells is suitable. Each cell is then formatted so that only its background (which is the pixel colour) is shown. The whole table is generated on the fly as an inner HTML content of a DIV panel. Upon the blocks presenting pixels, a North Sea geographical map with additional information is placed. This map gives user a geographical view of abundance pixels. Because it is fixed across maps, it can be prepared by web server as a pre-drawn picture and downloaded only once.

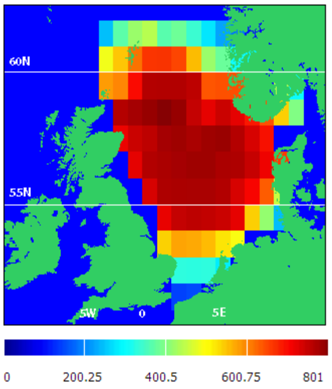

Figure 2 - A map with all relevant areas

Another problem of the client system is that because the new system supports various flexible ways to interact and swap maps, changing among maps should be as responsive as user interface. Therefore, a cache on client-side not only provides more conveniences for users but also supports additional abilities for other components or for the system as a whole. One of the conveniences can be shown in a manner that user waits no time for a repetitive map. Besides, an example for the supporting role of a client cache is that user can send multiple requests to save network latencies, which accumulate if single request is sent one by one. The user switches from map to map to request them asynchronously and requests are made without interruptions behind the scene. Data is cached whenever a respond comes to the user device and he then switches back to each map to work with client stored data. Because there are no interaction delays to wait for responses (a feature not from cache), requests are sent consecutively and there is no latency accumulation.

In technical aspect, due to AJAX technologies and program-generated server contents, browser cache does not work well with map data. Therefore, a client cache is implemented as a component of the client side application. As a consequence, it is a JavaScript component. The cache is in fact a proxy standing between user interface and communication back end. The proxy is responsible for data and hence uses cache records to serve the user interface.

5 Achievements and limits

In comparison to current WebCPR as well as WinCPR versions, the new system has reached several improvements as expected.

Firstly, a completely new model for the system has been designed and proved to work properly. The new model still follows three-tier web architecture but creates an application on client side. The research has proposed a way to maintain a true client application, similar to standalone desktop programme, running inside a web browser.

Secondly, web server application becomes thinner, which needs far less power than a normal web site. The web page is mainly run on client-side, except for updates of the species list so the server-side page does noticeably fewer works than usual, when it processed all input changes. In addition and more importantly, map generation mechanisms, which consumed the majority of server-side power, have been moved down to client side. Map related works on server side now shrink to just formatting relative data. Fewer requests for the same resource, supported by client cache, also helps reduce server demands. Several test cases were conducted to measure server side resource consumption. Synchronous sequences of map data requests were generated in a desktop computer to mimic volumes of simultaneous or consecutive remote user requests. Test cases, with a lower system configuration and not optimized settings, were in a far weaker server condition than in real life situations. However, the results proved the efficiency of the approach. 60 pairs of requests/responses took around 3.8 seconds and 14% CPU use to be processed in cases requiring least database manipulations. (It is important to note that web server operations are separated from database system ones but test cases included both of them).

A rich client application is the third achievement. User can interact with a responsive, friendly, informative user interface as well as transformable map. Artefact-free image quality is also an improvement, with the disappearance of image compression.

Moreover, network bandwidth is used more efficiently (or much less). It is because of new type of data transmitted as well as its structure. Another reason is the effort of client application to avoid repetitive transmissions. To measure this improvement, a number of typical XML responses were captured to compare with correlative images made from WinCPR and compressed by Adobe ® Photoshop CS3. Several images in current WebCPR version were also captured, though they were produced from other databases. Transmission by using images requires at least 4 to around a hundred times as much network bandwidth as using XML contents.

However, several issues have been pointed out in the application, but not yet solved. Firstly, because images are composed by HTML elements, but not real digital pictures, image manipulations from web browsers do not work on them. To short it out, user can use screen capture function coming with almost all modern graphical operating systems, though it is an inconvenient way. The second problem is that all errors in asynchronous communication are bypassed. In the future, time-out and request errors can be processed inside the proxy to completely hide network communication from user interface module. Besides, as is an internal component of the client side application, cache lasts only in one session and this issue seems to be a coherent problem of the application.

6 Conclusion and future works

The research aims at proposing ideas to improve current WebCPR system, as well as WinCPR application, which are used to visualise plankton data from the CPR survey of SAHFOS. At the same time, a complete and usable system has been built to show the ideas and can be used as the starting point to develop further abilities.

The research has implemented a new model, in which the client-side application plays an active role instead of server-side application as in traditional approach. By that way, the web application comes closer to a desktop GUI program. Drawing mechanisms are also move from server side to client side, along with new data structure to transmit far fewer bytes on network but to carry information in details. Less required server resource is another result of the move. Additionally, fully comprehensive data received by the client, working with several components such as client caching system, also supports better user interface and interactions, including a transformable and interactive map. Some of new features are improvements over current Win CPR application.

In the future, along with solving some issues in the implementation, a number of directions are open to develop the research as well as the system further. Firstly, other types of charts needs more deep research due to the differences among chart types, which require specific data structures and drawing methods. In addition, visualisation methods can be enriched by several ways such as overlapping multiple maps. Another direction is to modify the system so that it can support other databases from both North Sea and North Atlantic observations. Improving database performance is also a promising area.

7 References

Garrett, J. J. (2005), “Ajax: A New Approach to Web Applications”, Adaptive Path, http://www.adaptivepath.com/ideas/essays/archives/000385.php, (Accessed 26 August 2008)

Masoud, F.A. and Halabi, D.H.. (2006), “ASP.NET and JSP Frameworks in Model View Controller Implemetation”, *Proceedings of International Conference on Information and Communication Technologies 2006 (ICTTA’06) 2nd*, Vol. 2, pp3593 - 3598

Mesbah, A. and van Deursen, A. (2007a), “An Architectural Style for Ajax”, *Proceedings of the Working IEEE/IFIP Conference on Software Architecture, 2007 (WICSA’07),* pp9 - 18

Mesbah, A. and van Deursen, A. (2007b), “Migrating Multi-page Web Applications to Single-page AJAX Interfaces”, *Proceedings of 11th European Conference on Software Maintenance and Reengineering (CSMR'07),* pp181-190.

Brown, M. (2007), “Adobe plugin-in technology study”, Adobe System Incorporated, http://www.adobe.com/products/player_census/, (Accessed 20 January 2008)

Ramirez, A. O. (2000), “Three-Tier Architecture”, *Linux Journal*, Vol. 2000, No. 75es, Article No. 7, ISSN: 1075-3583, Online: http://www.linuxjournal.com/article/3508 (Accessed: 27 August 2008)

Shan, T. C. and Hua, W. W. (2006), “Taxonomy of Java Web Application Frameworks”, *Proceedings of IEEE International Conference on e-Business Engineering July 2006 – ICEBE’06*, pp378 - 385