Development of Portal Web Pages for the LHD Experiment*)

Masahiko EMOTO, Hisamichi FUNABA, Hideya NAKANISHI, Chie IWATA, Masanori YOSHIDA and Yoshio NAGAYAMA

National Institute for Fusion Science, 322-6 Oroshi-cho, Toki 509-5292, Japan

(Received 8 December 2010 / Accepted 25 February 2011)

Because the LHD project has been operating with the cooperation of many institutes in Japan, the remote participation facilities play an important role. Therefore, NIFS has been introducing these facilities to its remote participants. Because the authors regard Web services as essential tools for the current Internet communication, Web services for remote participation have been developed. However, because these services are dispersed among several servers in NIFS, users cannot find the required services easily. Therefore, the authors developed a portal Web server to list the existing and new Web services for the LHD experiment. The server provides services such as summary graph, plasma movie of the last plasma discharge, daily experiment logs, and daily experimental schedules. One of the most important information from these services is the summary graph. Usually, the plasma discharges of the LHD experiment are executed every three minutes. Between the discharges, the summary graph of the last plasma discharge is displayed on the front screen in the control room soon after the discharge is complete. The graph is useful in evaluating the last discharge, which is important information for determining the subsequent experiment schedule. Therefore, it is required to display the summary graph, which plots more than 10 data diagnostics, as soon as possible. On the other hand, the data-appearance time varies from one diagnostic to another. To display the graph faster, the new system retrieves the data asynchronously; several data retrieval processes work simultaneously, and the system plots the data all at once.

Keywords: remote participation, portal, web, AJAX, Ruby on Rails

DOI: 10.1585/pfr.6.2406051

1. Introduction

Because of the growth in the scale of nuclear-fusion experiments, collaboration between different organizations has become necessary, and remote participation has been adopted for various projects [1, 2]. The LHD project has been operating with the cooperation of many institutes in Japan. NIFS has also been introducing remote participation facilities to its remote participants [3]. Because the authors regard Web services as essential tools for the current Internet communication, Web services for remote participation have been developed. For example, a user can use his Web browser to modify diagnostic parameters, visualize the acquired data, and search for other information such as the schedule or the configuration of the experiment. However, as the number of Web servers has increased, it has become difficult, especially for visitors, to find the necessary information. To improve this situation, a portal Web server that guides the user in retrieving the necessary information is required. There are several portal Web pages for nuclear-fusion experiments; for example, for the DIII-D project, General Atomic provides a portal page [4]. This page uses Web 2.0 technology to enable the user to create customizable interfaces.

The basic requirement of the portal page for remote participants is a collection of the links to useful services. However, it is also important to provide basic information without requiring visits to other Web pages, such as the summary of the experiments currently underway and emergency messages. Therefore, the portal page must be interactive and must automatically update its contents, depending on the status of the experiment. The Asynchronous JavaScript + XML (AJAX) technique is popular for developing such Web pages. However, it is rather difficult to build an AJAX application, because the programmer has to develop applications whose GUI dynamically changes. Furthermore, the programmer has to use different computer languages simultaneously: JavaScript for the client side, and other languages such as Perl or PHP, for the server side. Ruby on Rails (RoR) [5] is widely used to efficiently develop AJAX applications. RoR provides the Model View Controller architecture. Therefore, it is simpler for the programmer to develop dynamic GUIs, because of the separated logic and GUI components. Furthermore, because RoR encapsulates JavaScript and SQL, the programmer can develop an AJAX application by using Ruby alone. Because the authors have realized the utility of RoR through the experience of developing several Web applications such as data viewer and experimental scheduler, the author chose RoR to develop the portal Web page.

author's e-mail: emoto.masahiko@nifs.ac.jp

*) This article is based on the presentation at the 20th International Toki Conference (ITC20).

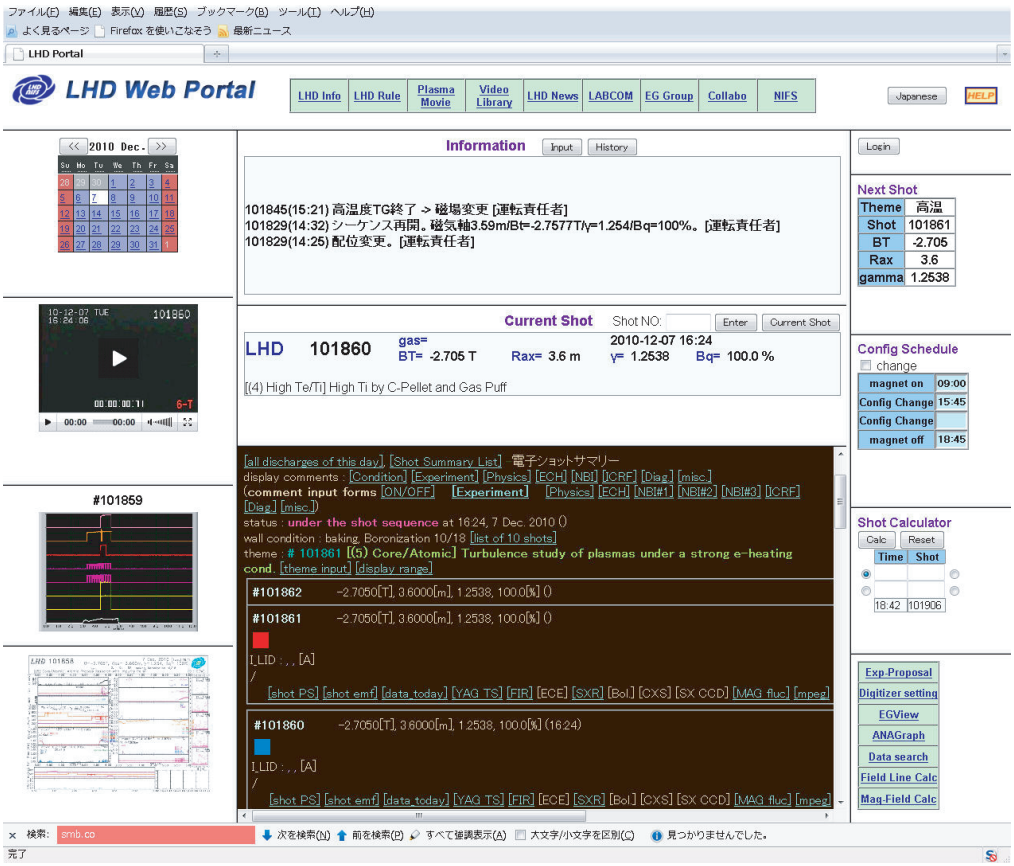

Fig. 1 Portal Web page of the LHD project.

2. System Overview

Figure 1 shows the LHD portal Web page. The page consists of 13 frames, each of which is an independent RoR application. Therefore, the modification of one frame does not affect the other frames, which makes it easy to develop a single component. Furthermore, this structure makes it easy to customize windows by choosing the necessary functions.

The calendar in the top left corner is linked to the experimental scheduler. The user can view the schedule and submit experiment proposals from here. To edit a schedule, the user has to log into the system, else, the applications will operate in guest mode and only read-only access will be available to the user.

The following frames display the summary information of the last plasma discharge, including a plasma video, an NBI current graph, and a summary graph. The plasma video image is recorded by a camera attached to the LHD vacuum vessel [6]. It is converted to MPEG format by personal computers. The daemon program running on the file server periodically checks if new MPEG files exist in a shared folder. If it finds new MPEG files, it converts them into Adobe Flash format files because the size of MPEG files is too large to transfer to many users via a 100 Mbps network. The size of the converted file becomes about one-fifth of the original size. The NBI current graph is used to determine whether NBI is injected. This image is a screen dump of the PC that monitors the NBI signals and is saved into the shared folder of the file server. The summary graph plots the important values of the last plasma discharge. The summary graph is a PV-Wave application and a part of the Information Display System for Plasma Operation. The graph is displayed on the front screen of the control room in real time. After completing the graph, it sends PostScript data to the file server. When the server program receives the PostScript file, the program converts it into a PDF file and a PNG file. The flow of data used by the Web portal is shown in Fig. 2.

In the middle column (Fig. 1), there is a daily log of the experiments. In the top area, it displays user comments. Every user can enter comments to notify the participants of important information. At the bottom, there is the summary information of each plasma discharge.

The right column displays information on the following plasma discharges; in the middle area, there is a shot calculator, which is an application that estimates the time corresponding to the shot number, or vice versa; it is beneficial to know the number of discharge experiments that can be performed within the assigned time.

The Web contents are updated as the experimental sequence progresses. To notify the computers of the experimental sequence, IP multicast packets are broadcast in the

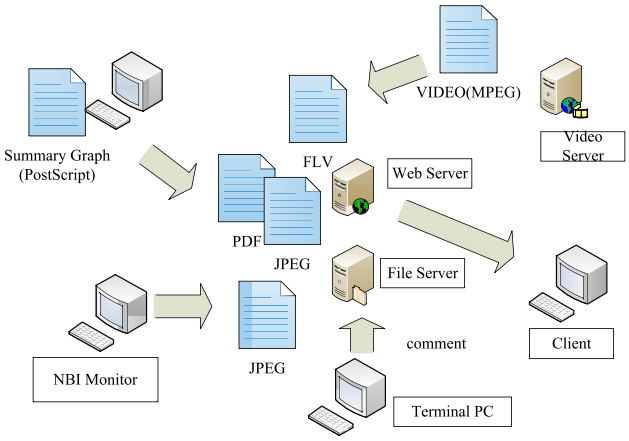

Fig. 2 Data flow of the portal page.

local network. The packets are sent when the sequence status or experimental number changes. When the server program receives the IP multicast packet, it updates the contents.

3. Shot Summary Graph

One of the most important pieces of data displayed in the portal page is the summary graph, which is useful in obtaining the results of the previous experiment while conducting an experiment. The LHD plasma discharge experiments are typically executed every three minutes. The scientists configure the diagnostic parameters and experimental conditions between the discharges.

Most of the raw signal data of the LHD experiment is managed by the LABCOM system [7], and the physical data is managed by the Kaiseki server system [8]. The LABCOM system is not only a data acquisition system, but also a data archive system that handles data acquired by various systems including the LABCOM system. To retrieve the data from the LABCOM system, the system provides a Retrieve command and an API library. Similar to the LABCOM system, the Kaiseki server provides the unified interface and data format of various physical data. Therefore, by using these commands or the API for each system, the user can use only a few interfaces to read the data acquired by various systems.

However, the LABCOM system did not support the necessary diagnostic data to be monitored when the LHD project started, and the Kaiseki server was not available at the time. On the other hand, the summary graph application was urgently needed, and it was developed to retrieve data from heterogeneous data acquisition systems. To receive the data from various systems, the program uses the folder sharing function of Microsoft Windows. Each acquisition system writes the data into the shared folder, and the application reads it if it is available. The application checks all the data one by one. The higher priority data such as plasma stored energy or electron density are read to be plotted soon after they are available. However, the

Fig. 3 Lastshot.py: the summary graph application written in Python.

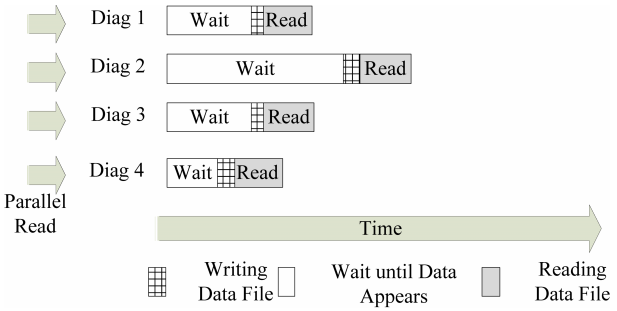

Fig. 4 Parallel/Asynchronous read: The application runs multiple processes to read four diagnostics at once. Each process sleeps until the data is available.

lower priority data is not read until its turn comes, even if it is available. Therefore, there is a delay in plotting the lower priority data. The interval between plasma discharges is about three minutes, but it is not enough time to evaluate the data, because it takes time to transfer the acquired data from the digitizers to the server. Therefore, this delay must be as short as possible.

To minimize the delay, the new application Lastshot.py has been developed (Fig. 3). Lastshot.py is written in Python, which is an object-oriented script language that has characteristics similar to Ruby. The main reason for choosing Python is that it has various scientific modules, such as matplotlib and pylab, to handle the experiment data easily. The new application uses two techniques to minimize the delay (Fig. 4). The first one is simultaneous read; Lastshot.py retrieves all of the experimental data simultaneously instead of sequentially; when the new experimental sequence begins, the application runs multiple child processes to retrieve the data; the parent process receives the data on a first-come, first-serve basis, and plots the data without delay.

The second technique is asynchronous read. Lastshot.py obtains the data from the LABCOM system and the

Kaiseki server system. To read the LABCOM data asynchronously, it uses the wait function of the Retrieve API. The retrieval process is locked until the data are available. When the data is available, the process is unlocked and reads the data soon. On the other hand, to read the data of the Kaiseki server system, it uses IP multicast notification. When the new data is registered in the Kaiseki server, the server sends an IP multicast to notify the clients of the registration. The child process sleeps until the notification arrives and reads the Kaiseki data soon after it receives the packet.

Using these techniques, Lastshot.py can plot the data without significant delay.

4. Summary and Future Plan

Using RoR technology, a real-time portal Web page for the LHD project was developed. It has been helping users to participate in the entire last season of four months of the LHD experiment, and the number of accesses to the portal page has doubled in this season. When visiting this page, the user does not bother to use various tools to obtain the necessary information. The authors believe that this page is useful and that it will promote remote participation.

During the experiment, in response to the user's requests, the application has been modified and some new features have been added. However, because of RoR, it was not necessary to stop the service to maintain the software. This could prove the utility of RoR.

The portal page has been improved to provide a better environment for the remote participants. For example, the improvement of the shot summary-graph application will reduce the delay time, which will enable the participants to use their time more efficiently.

Acknowledgment

This work was supported by a budget NIFS10ULHH014 of the National Institute for Fusion Science. The authors are grateful to all the staffs of the LHD project.

[1] G. Abla *et al.*, Fusion Eng. Des. **83**, 480 (2008).
[2] D.P. Schissel, Fusion Eng. Des. **83**, 539 (2008).
[3] M. Emoto *et al.*, Fusion Sci. Technol. **58**, 458 (2010).
[4] G. Abla, E.N. Kim, D.P. Schissel and S.M. Flangan, Fusion Eng. Des. **85**, 603 (2010).
[5] http://www.rubyonrails.org/
[6] M. Shoji *et al.*, Plasma Fusion Res. **2**, S1046 (2007).
[7] H. Nakanishi *et al.*, Fusion Eng. Des. **82**, 1203 (2007).
[8] M. Emoto *et al.*, Fusion Eng. Des. **81**, 2019 (2006).