Web サーバーによるリアルタイム画像配信システムの運用
OPERATION OF REAL TIME IMAGE DISTRIBUTION SYSTEM BY WEB SERVER

中村卓也 *A)、吉井兼治 A)、中村達郎 B)、帯名崇 B)
Takuya Nakamura* A), Kenji Yoshii A), Tatsuro Nakamura B), Takashi Obina B)
A)Mitsubishi Electric System and Service Co.,Ltd.,
B)High Energy Accelerator Research Organization (KEK)

Abstract

The KEKB control group has been built the system which delivered the accelerator operation panel images and video signal images with a Web server. After that, we created an image distribution system using Node.js to add functions and operate it stably. In addition, in order to make it easier to check the operation status of accelerator from a remote location, we have created a new web page that distributes a screenshot image of the console screen. In this paper, we report the operation of real time image distribution system by web server.

1． はじめに

KEKB 制御グループでは、2007 年から Web サーバーを用いたリアルタイムな画像を配信するシステムを運用している [1]。この画像配信システムは、加速器運転パネルのスクリーンショット画像やビデオ映像を取り込んだ画像を、Web サーバーから各クライアントの Web ブラウザへ定期的に送信することで実現している。配信する画像ファイルの作成方法は、加速器運転パネルのスクリーンショット画像を保存するスクリプトや、ビデオ映像を画像ファイルとして取り込む機器を使用している。なお、この画像配信システムでは、同時に表示する画像の数に制限があり、複数の画像を同時に閲覧することができなかった。また、利用する Web ブラウザの種類によっては、長期間閲覧している途中で画像の更新が停止してしまう問題もあった。そのような問題に対応するため、Node.js を用いた新たな画像配信システムを整備した。Node.js による画像配信システムでは、任意の数の画像を表示する機能や、画像の表示サイズの変更機能などを整備し、画像の表示方法をユーザー側で自由に選択することができる。また、長期間の閲覧時にも画像の更新が停止することなく、安定した動作を実現することができた。また新たな画像配信のコンテンツとして、制御室にある主要な運転用端末の画面全体のスクリーンショット画像の配信を開始した。最近推奨されているリモートワークでは、制御室の画面を見ることができず加速器の運転状況を把握しにくい状況であった。制御室の運転用端末の画面の画像を配信することで、リモートから加速器の運転状況を把握しやすくなるよう対応した。

本件では、KEKB 制御グループで運用している、Web サーバーによるリアルタイム画像配信システムの運用について報告する。

2． 画像配信システムの構成

2.1 画像配信システムの概要

KEKB 制御グループでは、加速器運転パネルの画像やビデオ映像から取り込んだ画像を、Web サーバーを通じて配信するリアルタイムな画像配信システムを運用している。この画像配信システムは HTTP server push と呼ばれる技術を使用し、Web サーバーからユーザーの Web ブラウザへと定期的に画像データを送信することで実現している。

加速器運転パネルの画像は、VNC 上で実行している加速器運転パネルについて、定期的にスクリーンショット画像を保存して用意している。ImageMagick [2] の import コマンドを定期的に実行するシェルスクリプトを用意し、加速器運転パネルのスクリーンショット画像を保存している。ビデオ映像から取り込む画像は、AXIS 社 [3] のビデオサーバーと呼ばれる機器を利用して用意している。このビデオサーバーは、ビデオ信号の処理機能と LAN への接続機能を備えており、入力されたビデオ信号を画像ファイルとして FTP サーバーへ送信することができる。また、これらの画像ファイルは頻繁に読み書きが行われるため、HDD の負荷軽減と高速なファイルアクセスを行うよう、サーバー計算機のメモリ上の RAM ディスクに保存している。

なお、HTTP server push による画像配信システムでは、同時に表示する画像の数の制限や、長時間の閲覧時に画像の更新が停止するなどの問題があった。そのような問題に対応するため、Node.js [4] を用いた新たな画像配信システムを整備した。

HTTP server push、及び Node.js による画像配信システムについて、運用する計算機の仕様を Table 1 に示し、画像配信システムの構成図を Figure 1 に示す。

Table 1: Specification of Server for Image Distribution System

Model	HP ProLiant BL460c G1
CPU	Dual-Core Intel Xeon 5160, 3.00 GHz (x2)
Memory	4 GB
OS	CentOS 6.10

* nakataku@post.kek.jp

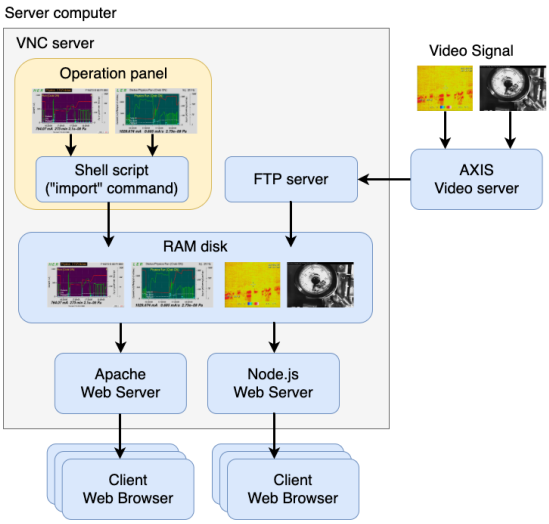

Figure 1: Structure of image distribution system.

3． 新たな画像配信システム

3.1 Node.js による画像配信システム

Node.js は、サーバー計算機側で動作する JavaScript の実行環境であり、Web サーバーとして運用することができる。Node.js では Socket.io [5] と呼ばれる通信ライブラリの利用が可能で、サーバーとクライアント (ユーザーの Web ブラウザ) の間で、双方向の通信を行うアプリケーションを構築することができる。この Node.js と Socket.io を用いて、サーバーとクライアント間で通信を行う新たな画像配信システムを構築した (Figure 2)。なお、我々の画像配信システムでは、Node.js v10.15.1 と、Socket.io 2.2.0 を使用している。

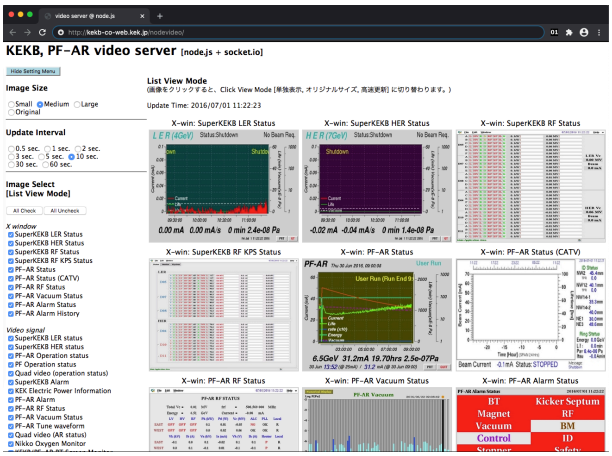

Figure 2: Overview of Node.js image distribution web page.

Node.js による画像配信システムでは、クライアントからサーバーへ画像配信の要求を送信し、要求を受けたサーバーが要求元のクライアントへ画像のデータを送信する動作を行う。このクライアントからの画像配信要求と、サーバーからの画像データ送信を定期的に繰り返すことで、リアルタイムな画像配信を実現している。このとき、サーバーからクライアントへ送信する画像のデータは、base64 の文字列のデータに変換した情報を送信している。

3.2 画像データの取り扱い

Node.js による画像配信システムでは、画像ファイルのバイナリデータを base64 の文字列のデータに変換して扱っている。画像ファイルをそのまま表示する方法では、Web ブラウザのキャッシュデータにより最新の画像が表示されない問題や、Web ブラウザのキャッシュデータが肥大化する問題があったため、base64 の文字列に変換した画像データを扱う方法を選択した。base64 の文字列データを用いて画像を表示する方法により、最新の画像が毎回正しく表示され、また Web ブラウザのキャッシュデータが肥大化することなく、安定して画像データを配信することができた。

3.3 サーバー側の動作

Web サーバーとして動作する Node.js では、以下の処理を継続して行っている。

- 画像データを base64 に変換するスクリプトの実行
- base64 に変換された画像データの読み込み
- クライアントからの要求に対する画像データの送信

サーバー側の Node.js のアプリケーションは、定期的に保存される画像ファイルについて、シェルスクリプトによる base64 の文字列データへの変換処理と、変換した文字列データを読み込む処理を継続的に行っている。サーバーはクライアントからの画像データの配信要求を受信すると、読み込んでいた画像の文字列データをクライアントへと送信する。

3.4 クライアント側の動作

クライアント側である Web ブラウザでも JavaScript を実行しており、サーバーとの画像配信の通信処理や、表示画面の制御の処理を行っている。

サーバーとの画像配信の通信処理では、サーバーに対して画像の配信要求を定期的に送信する処理を継続して行っている。画像データの配信要求を送信した後、サーバーから返信された base64 の文字列に変換された画像データを受信すると、Web ブラウザで表示している画像データの更新を行う。サーバーとクライアントの間の画像配信の構成について、Figure 3 に示す。

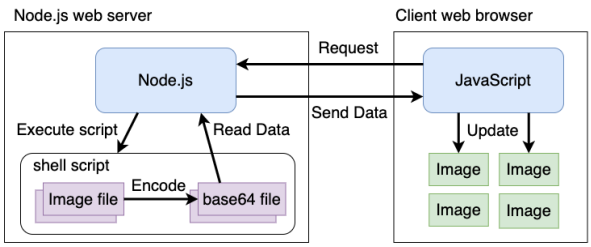

Figure 3: Structure of Node.js web server and client web browser.

また、クライアント側の操作として、同時に表示する画像の選択、画像の更新頻度の変更、画像の表示サイズや表示順序の変更など、ユーザーの希望に応じて画像の表示形式を切り替えることができる。これらの機能は、サーバーとの通信は必要とせず、クライアント側の JavaScript だけで制御している。

4. 加速器運転端末のスクリーンショット画像の配信

2020 年の新型コロナウィルスの感染拡大防止のため、リモートワークを活用することにより、制御室に在室する人数を制限して、加速器の運転を行う方針となった。加速器の制御室には、加速器を制御するパネルや、運転状況を示すパネルが多数表示されており、それらを見ながら加速器の運転を行っている (Figure 4)。しかし、リモートワークの環境ではそれらの画面を見ることができず、加速器の運転状況を把握しづらい状況になっていた。そこで、加速器制御室の端末の画面スクリーンショットを定期的に配信するシステムを整備し、リモートからでも加速器の運転状況を把握しやすくなるよう整備した。

Figure 4: Consoles and monitors at SuperKEKB control room.

4.1 Linux 端末

加速器の運転制御を行う端末として、Linux (CentOS 7) の端末を 6 台使用している。各 Linux 端末には 27 インチのディスプレイを 2 台接続しており、多数の制御パネルを表示している。Linux 端末のスクリーンショット画像は、gnome-screenshot コマンドを使用したシェルスクリプトを実行して作成している。gnome-screenshot コマンドは、表示されている画面をそのまま保存することができ、また画像の保存処理中も端末の操作に影響を与えることなくスムーズに処理される。このシェルスクリプトを cron により毎分実行して、スクリーンショット画像の保存、及び画像配信システムのサーバーへの画像ファイルの送信を行っている。

4.2 Macintosh 端末

加速器の運転状況を表示する大型ディスプレイの表示には、Macintosh の計算機である Mac mini を 4 台使用している。Mac mini には、壁に設置された 55 インチのディスプレイを 1 台または 2 台接続して、加速器の運転状況を示すパネルを表示している。Mac mini 端末のスクリーンショット画像は、screencapture コマンドを使用したシェルスクリプトを実行して作成している。Linux 端末と同じく cron による毎分の定期実行を整備し、保存した画像を画像配信システムのサーバーへと送信している。

4.3 端末のスクリーンショット画像配信ページ

加速器運転端末のスクリーンショット画像配信ページでは、縮小したスクリーンショット画像を並べて一覧で表示している (Figure 5)。表示されている画像を選択すると、元の大きさのスクリーンショット画像が表示され、詳しく確認することができる。なお、加速器運転端末のスクリーンショット画像配信ページは、リアルタイムな画像配信ページとは異なる通常の Web サーバーで運用している。

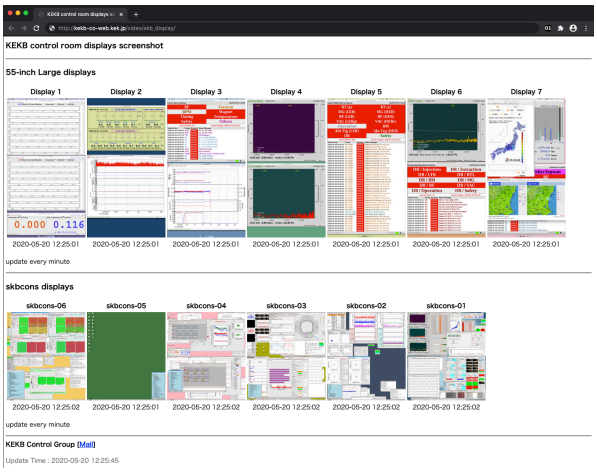

Figure 5: Overview of console screenshot web page.

5. まとめ

Node.js による画像配信システムは 2012 年から運用を開始し、ソフトウェアのバージョンアップや機能の追加を継続して行い、現在も安定した運用を続けている。また、運転端末の画面のスクリーンショット画像の配信の整備により、リモートからでも運転状況を把握しやすくなった。今後もユーザーからの要望に応じて、画像の更新頻度の増加や、Node.js のリアルタイム画像配信システムとの連携など、より便利なサービスの実現を進めていく。

5.1 今後の予定

画像配信システムを運用する環境について、仮想マシンへの移行を検討している。現在は物理マシンで画像配信システムを運用しているが、計算機の不調により画像配信サービスの運用が中断されることがあった。仮想マシンであれば、実行する計算機を移動することで、素早くサービスを復旧させることができると期待される。安定したサービスを継続して運用できるよう、仮想マシンへの運用環境の移行を図りたい。

参考文献

[1] T. Nakamura *et al*., "Delivery of accelerator operation panel active images and video image using web server", Proceedings of the 5th Annual Meeting of Particle Accelerator Society of Japan, Higashihiroshima, Japan, Aug. 8-11, pp. 643-645 (2008).
[2] https://imagemagick.org/
[3] https://www.axis.com/ja-jp
[4] https://nodejs.org/ja/
[5] https://socket.io/