SCALABLE WEB BROADCASTING FOR HISTORICAL INDUSTRIAL CONTROLS DATA

B. Copy, O. Andreassen, P. Gayet, M. Labrenz, H. Milcent, F. Piccinelli
CERN, Geneva, Switzerland

Abstract

With the widespread adoption of asynchronous web communication, thanks to WebSockets and WebRTC [1], it has now become possible to distribute industrial controls data (such as coming from devices or SCADA software) in a scalable and event-based manner to a large number of web clients [2] in the form of rich, interactive visualizations. There is yet, however, no simple, secure and performant way to query large amounts of aggregated historical data.

This paper presents an implementation of such an architecture, which is able to make massive quantities of pre-indexed historical data stored in ElasticSearch available to a large number of web-based consumers through asynchronous web protocols. It also presents a simple, Opensocial-based [3] dashboard architecture that allows users to configure and organize rich data visualizations (based on Highcharts Javascript libraries) and create navigation flows in a responsive, mobile-friendly user interface.

Such techniques are used at CERN to display interactive reports about the status of the LHC infrastructure (*e.g.*, Vacuum and Cryogenics installations) and give access to fine-grained historical data (such as stored in the LHC Logging database [4]) in a matter of seconds.

THE CERN ENGINEERING DASHBOARD

The CERN Engineering Dashboard is primarily a data streaming facility, giving secure and scalable web-based access to live, critical data coming directly from the CERN Accelerator infrastructure.

The CERN Engineering Dashboard also promotes the usage of modern, standards-based data visualizations, leveraging the best open web standards (*e.g.*, CSS 3, HTML 5, SVG), thereby ensuring that the solution can cope with technological debt and clearly separate the data broadcasting mechanism from the way data is rendered.

Figure 1 below gives an overview of the Broadcasting Dashboard architecture, allowing the scalable distribution of both live and historical data and the rendering of said data through modular and reusable visualizations.

STREAMING HISTORICAL DATA

While our previous publication on the topic of web broadcasting (MOPPC145 [1]) focused on the ability to stream live data, the CERN Engineering Dashboard project has aimed for the past two years to provide the same functionality for historical data, introducing in the process a whole new set of challenges:

- Instantly retrieving and serving large amounts of data, instead of atomic values, while still being able to serve in the vicinity of one thousand web clients concurrently.
- Aggregating data on the fly so that mobile clients are not submerged with unnecessary information that they will, in any case, be unable to display on a small viewport.
- Not making any assumptions on the data at hand, yet still providing efficient indexing and filtering capabilities.

Over the course of this project, it soon became obvious that traditional data storage techniques, such as relational databases, were insufficient:

- Connections to relational databases must usually be pooled (about a dozen concurrent connections are tolerated on the central CERN Oracle database); serving the needs of thousands of clients concurrently was not possible.
- Relational data access times impose dedicated connection drivers and too much latency, and transmitting this data over the web imposes yet more delay and processing cost in data format translation.

A new generation of data storage solutions, identified under the umbrella term “NoSQL,” has emerged in the past few years. NoSQL solutions (thus coined for their breaking free from traditional SQL relational data storage constraints) rely on asynchronous, massively parallel processing and cloud clustering techniques in order to address the aforementioned requirements.

NOSQL DATA STORAGE SOLUTIONS

There are several NoSQL data storage solutions on the market at present. Our goal was to identify a solution that:

- was web-friendly, could natively communicate via HTTP, and support a query language as expressive and feature-capable as SQL;
- could operate in cluster mode, so as to increase availability and support data partitioning;
- could integrate with online processing frameworks such as Apache Storm;

ISBN 978-3-95450-148-9

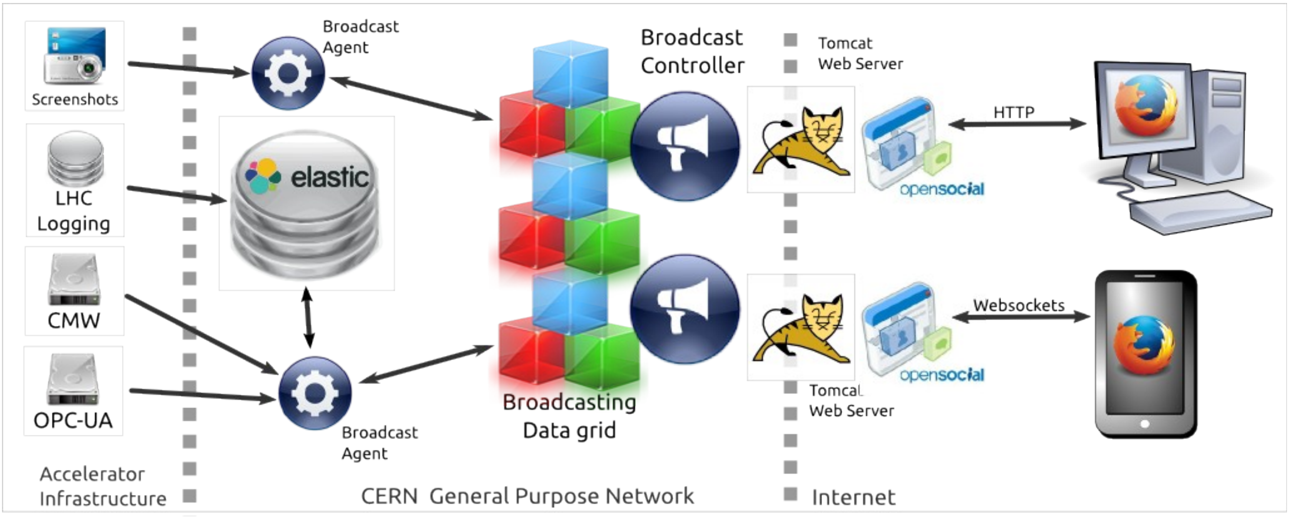


Figure 1: Broadcasting Dashboard architecture

- and would be able to natively manipulate the JSON data format, which is currently used by our web broadcasting framework.

Among the multitude of options available, the open-source solution ElasticSearch [5] fulfils all of the above criteria and benefits from a very strong momentum from large Internet companies and its developer community. Able to operate in isolation or as part of a cluster, it is also extremely easy to embed in any unit tests or integration tests, making it much more comfortable than a monolithic, relational database with regards to development.

Using ElasticSearch, we can store a moderate amount of LHC activity data (typically forty days of history), that we can aggregate on the fly, so as to serve end-users with only the appropriate amount of information suitable for their browsing platform, while retaining the initial expressiveness of the source data.

INTUITIVE NAVIGATION AND REUSABLE VISUALIZATIONS

While ElasticSearch features a data visualization technology called "Kibana," an online web tool that allows one to create dashboards interactively, it suffers from several important shortcomings:

- Visualizations inside Kibana cannot be integrated into another website – they must run inside a Kibana portal interface;
- Kibana is very much focused on ElasticSearch monitoring – it offers a limited number of visualizations that are mostly suitable for line charts, data tables, and tag clouds.
- Extending Kibana with custom visualizations is difficult, and would, in any case, not comply with any web standards, thereby forcing one to develop Kibana-only data visualization modules.

There are, however, existing web standards for contents reuse, such as Opensocial, which defines the concept of **Gadget**, a page fragment that can be parameterized and dropped into any web page (provided certain security constraints are respected).

We therefore developed all our data visualizations as Opensocial gadgets. Even when combining multiple visualizations into a single page layout, each gadget can be easily extracted and reused in another, unrelated website while preserving its interactivity and appeal.

Data visualizations offered by the Engineering Dashboard include Scalable Vector Graphics (to bind data to scalable, vectorized diagrams that offer a pixel-perfect rendering at any display size), Image streaming (to stream bitmap images such as camera captures or screenshots), Highcharts (to display data in a large variety of interactive, zoomable charts), and Impact tree (to render hierarchical, tree-like data and support simple root cause analysis).

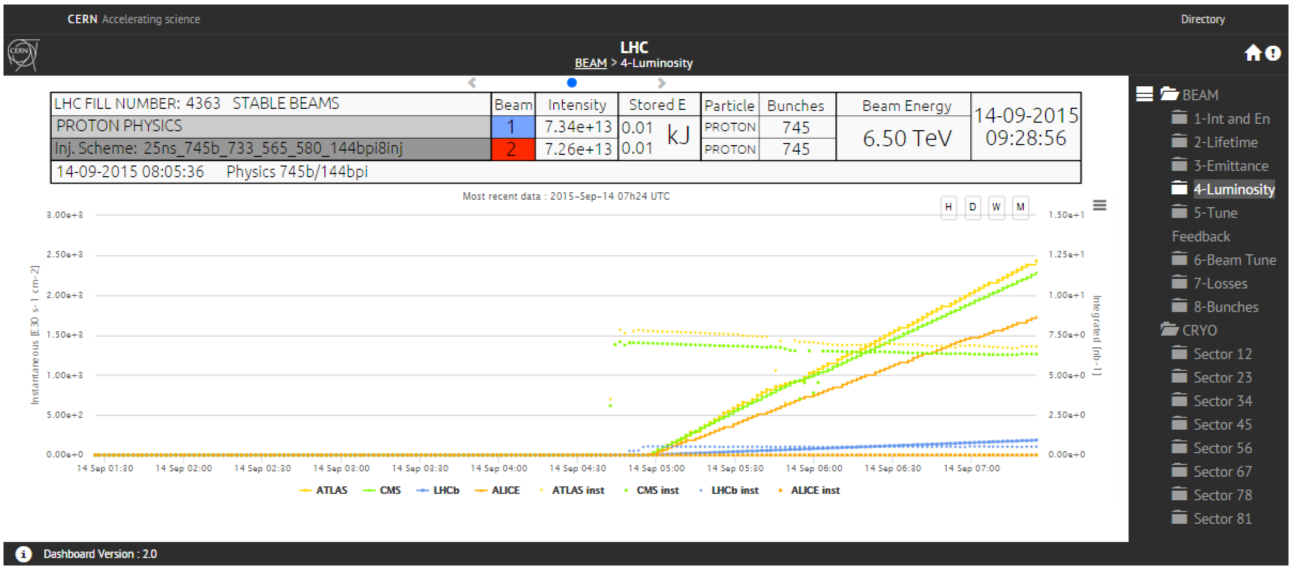
DASHBOARD USER INTERFACES

Unfortunately, offering single, isolated data visualizations is not sufficient to provide a comfortable browsing and analysis user experience. Visualizations typically need to be combined, offering different perspectives over the same data (for example, in the LHC cryogenic systems to compare the global temperature versus time of a given sector in a line chart against a temperature profile of individual equipment).

The Engineering Dashboard uses the concept of page, employing a layout to arrange gadgets dynamically on the screen. Since the Dashboard project targets mobile devices, layouts must be responsive [6], that is, automatically adaptable to a range of screen geometries.

ISBN 978-3-95450-148-9

The Dashboard's responsive layout implementation relies on the popular Bootstrap [7] framework, which offers a grid abstraction, making it possible to express the most complex layouts in terms of extensible columns and cells, coping gracefully with any display size ratio. Figure 2 below provides an example of visualization, (available at http://dashboard.web.cern.ch/LHC).



LHC FILL NUMBER: 4363 STABLE BEAMS	Beam	Intensity	Stored E		Particle	Bunches	Beam Energy	14-09-2015 09:28:56
PROTON PHYSICS	1	7.34e+13	0.01	kJ	PROTON	745	6.50 TeV	
Inj. Scheme: 25ns_745b_733_565_580_144bpi8inj	2	7.26e+13	0.01		PROTON	745		
14-09-2015 08:05:36 Physics 745b/144bpi								

Most recent data : 2015-Sep-14 07h24 UTC

Figure 2: Dashboard view of LHC experiments luminosity history

CLUSTERING AND FAILOVER SUPPORT

Another phase of improvements to the CERN Engineering Dashboard has been the search for a clustering and high-availability solution. To this day, the central CERN web hosting infrastructure does not support high-availability beyond the usage of simple DNS round-robin practices (for example, if three web servers have been defined as part of a cluster then every third request will be sent to the same server; this practice provides basic load-balancing, but no redundancy).

A simple workaround, and a great way to provide high availability and scalability, is simply to bypass altogether the restrictions of traditional web development (relying on Apache web server proxy definitions and HTTP sessions) by adopting a data grid technology [8]. Data grids extend the concept of random-access memory (RAM) by distributing it over a high-speed network. Data grids are transparent to the application programmer, who employs familiar data structures – such as maps, arrays, queues, locks, and even ring buffers – as if they were in local memory. In a data grid, all these data structures are distributed and synchronized across multiple hosts via low-latency data exchanges.

In-memory data grids constitute an ideal way of making our data broadcast service scalable and resilient to failures by maintaining, in a distributed manner, the entire state of data distribution, that is:

- the state of client subscriptions (*i.e.* "who needs what?");
- the availability, health statistics, and heartbeat of broadcasting agents;
- the publication-subscription channels that are currently active;
- last known value cache (no need to re-query the data source if nothing has changed);
- and even supporting the archiving of live data into queues that can later be indexed in a NoSQL data storage.

After a period of evaluation and research, the Engineering Dashboard project adopted Hazelcast [8] as its in-memory data grid implementation. Hazelcast [9] is a high-performance, open-source data grid library that proves extremely simple to use: adding it to any Java application immediately enables a data grid cluster.

Thanks to clustering support from Hazelcast, our broadcasting infrastructure can go from accommodating five hundred concurrent clients from a single server to transparently accommodating five hundred clients over N nodes, without a complex reconfiguration, a code rewrite, or even requiring the restart of any part of the broadcasting cluster.

CONCLUSION AND PERSPECTIVES

Advances made in the past few years in the matters of Java clustered data storage and in-memory data grid technologies have made affordable the deployment of

ISBN 978-3-95450-148-9

large and scalable data distribution media, such as the one used by the CERN Engineering Dashboard.

The reuse of web content, however, is still a field in mutation:

- Security issues continue to plague the World Wide Web and content reuse is only supported by software vendors insofar as it is done within their own social platform. For example, Microsoft Social and Facebook still do not allow the data visualization mechanisms they provide to be used in the context of other web hosting products. Importing their data into another web site implies relying on their proprietary data formats and protocol specifications, even if open alternatives are readily available.

- Over the past year, Opensocial, as a social data exchange and visualization platform, has been abandoned by Google and converted into a W3C working group (most likely opening an era of lengthy specifications and recommendations that will be ignored by major web companies).

The emergence of a new web standard entitled "Web Components" [10] is, however, poised to revitalize web content reuse and offer a widely accepted method of implementing content sharing. Already supported on all major web browsers, Web Components offers the right balance between the ease of usage and the implementation of strict content separation and code security practices.

The CERN Engineering Dashboard is certainly bound to explore the capacities of this social technology, as a means of overcoming the artificial boundaries imposed by software vendors in the free exchange of data and interactive data visualization gadgets.

REFERENCES

[1] H. Hickson, "Real-time communication between browsers", Feb 2015, W3C working group, http://www.w3.org/TR/webrtc/
[2] B. Copy et al., "Mass-Accessible Controls Data for Web Consumers", MOPPC145, Oct 2013, ICALEPCS'13, San Francisco, USA
[3] T. Çelik, "Opensocial and the Social Wg", July 2014, W3C working group, http://www.w3.org/Social/WG
[4] C. Roderick et al., "The LHC Logging Service", WEP005, Oct 2009, ICALEPCS'09, Kobe, Japan
[5] S. Banon, "ElasticSearch: you know, for search", 12 June 2012, http://thedudeabides.com/articles/you-know-for-search-inc
[6] E. Marcotte, "Responsive Web Design", May 2010, http://alistapart.com/article/responsive-web-design
[7] M. Otto, "Bootstrap from Twitter", 17 January 2012, http://www.markdotto.com/2012/01/17/bootstrap-in-a-list-apart-342
[8] J. Belzer et al, "Very large data base systems to zero-memory and Markov information source", Encyclopedia of computer science and technology, Vol. 14, 1980, Dekker, New York, USA
[9] T. Ozturk, "Clustering your application with Hazelcast", 16 Dec 2013, JAXLondon Conference, London, UK
[10] D. Glazkov and H. Ito, "Introduction to Webcomponents", 24 July 2014, http://www.w3.org/TR/components-intro/

ISBN 978-3-95450-148-9