Webalyt: Implemetation of Architecture for Capturing Web User Behaviours With Feedback Propagation

Petr Filip, Lukáš Čegan
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
Email: petr.filip1@student.upce.cz, lukas.cegan@upce.cz

***Abstract*—In the world of the Internet where people are consuming web-content, more and more emphasis is placed on user friendliness of web-pages. It means increasing web-page usability and better user-experience. Increasing quality of user-experience (UX) is the task of UX developers. They should always base their work on the best practices and research. Each web-page has its own specificity and it leads to new challenges for the UX developers. One of the biggest issues is the problematic view of a web-page on specific devices with specific web-browser versions. Tools for capturing user behaviour are available, but there are issues with data ownership and with the development of new functionality. Actually, there are no free easily scalable and extendable products for user data gathering on the market. In this paper, implementation of architecture (based on Spring Boot microservices) for capturing web user behaviours with feedback propagation is introduced. Architecture implementation is easily scalable and extendable. All of the components are described in detail. At the end of this paper, summary and limitation of created architecture is discussed. Webalyt is helpful for understanding user behaviour and improving user-experience.**

***Index Terms*—User-Experience, Web User Behaviour, Analytical Tool, Microservices, Spring Boot**

I. Introduction

Based on capturing of web user behaviour, it is possible to make improvements of user-experience. This is primarily about making a web-page more user-friendly. It also means bug fixing in a design layout of web-pages for specific devices such as mobile phones with a specific web-browser version. According to user behaviour, it is possible to make segmentation of user and target product offerings that fit exactly to the user. This discipline is called personalization and it is provided by recommendation systems. It is also possible to track information about used users' devices, type of connection, and data from various sensors which are built in the device.

A combination of these sources of information can be used for dynamic web-page adaption such as optimising of image size, or dynamic changing of font-size [1]. In the case of companies which are focused on e-commerce, optimisation can lead to increased revenues from selling goods or services over the Internet.

Gathering of these data brings new challenges for software companies focused on developing (near) real-time processing systems of data and big data. One of the most critical decisions in software development is about used technologies and proposed architecture which should be chosen based on specific system requirements.

This work is aimed on building modular real-time data gathering architecture with feedback propagation which focuses on web analytics and user-experience. Implementation is based on open web analytics platform called Webalyt [2]. Designed software is touching topics such as big data, software engineering, stream processing, data modeling, business analysis and recommendation systems.

The next sections describe related works which are divided to two segments, analytical tools and real-time data processing tools. The following section is architecture implementation with detailed component description and component interactions. This section also presents selected examples of data and components. At the end of the paper, there is a discussion which is focused on evaluation of presented proof-of-concept and its limitations.

II. Related Works

A. Analytical Tools

Matomo [3] (named Piwik [4] in the past) and Open Web Analytics [5] belong among the best known open-source analytical tools. These tools are written as monolithic apps, which means that the scaling is more complicated. Obviously, scaling is based on event queuing and the event can be processed later. The main used technology is PHP, which is not the best option for long-running data analysis. But one of the biggest advantages is that it is easy to deploy and run on cheap hardware – for small traffic.

Another well known analytical tool is Google Analytics. One of the biggest issues is data ownership, because they are stored on Google servers. Free version of this service has limitations, too [6], [7]. Also many other similar tools with various functionalities are available [8].

Next category of analytical tools which is worth mentioning is tools which do not offer business analysis, but provide

analysis for UX developers, specifically user session recording and replaying. It means, every user action is recorded and later can be replayed. These tools use WEB APIs of web-browsers [9]. These tools are SmartLook, SessionStack, MouseFlow, MouseStats, CrazyEgg, MonkeyTracker, Inspectlet, FullStory, KISSmetrics and LogRocket. The last mentioned, unlike the other tools above, provides request recording with duration of every request and log recording. It is very helpful for JavaScript debugging.

The aforementioned software have unknown implementation, that means functionality is limited by the software supplier. And it is very similar with extendability of the solution, which is very low. Ordinarily, these tools contain only session replay with click heat maps. Even with those that have really interesting and useful data, a full analysis is not provided. Moreover, the previously mentioned software are paid and provided as a service.

Current technologies for real-time data processing are not easy to deploy and scale. On the current market, there is no free scalable tool for user tracking behaviour with supported horizontal scaling. But the main feature which is missing is real-time feedback propagation to the user interface. In addition, cross-domain user-session identification is not considered.

B. Tools for Real-Time Data Processing

The technical part of this paper is focused on using available technologies for real-time data processing. Firstly, two main architecture concepts need to be introduced.

- Lambda architecture – is composed from batch layer, speed layer and serving layer. Common processing on this architecture is following – the speed layer creates transient results which are lately replaced by results from batch layer. As indicated, the biggest disadvantage is double implementation of transformation logic (speed and batch layer), which increases maintenance.
- Kappa architecture – is very similar and simplified lambda architecture, but the batch layer is removed. The main idea is that the data are processed as stream. The main advantages are easy development, debug and test.

In this proof-of-concept, Kappa architecture is implemented because of faster and easier development [10], [11].

There is a large number of technologies and solutions for real-time data processing [12]–[14]. For example Apache Samza, Apache Storm, Apache Flink and Apache Spark. The above-mentioned technologies have some disadvantages which are not suitable with following requirements.

Functional and non-functional requirements for software development are specified in the following list:

- easy to deploy – primarily independent on HDFS or YARN,
- easy to extends – this is ensured via service registry pattern and interface communication,
- scalable – micro-services architecture naturally support scaling,
- real-time processing with feedback – computed results are sent to the web-browser which reacts on the feedback. This feedback can affect the user interface such as content zooming, layout adaption, or update of advertisement,
- data ownership – stored data are not managed by 3rd party companies (optional),
- open-source – for transparent code which should lead to higher development speed and better code quality.

The best-practices of software development were emphasized during the development phase [15]–[19].

III. Implemented Architecture

Component distributed microservice Webalyt architecture for data collecting is based on Apache Kafka [20], Spring Boot framework, and MySQL database (will be replaced by Apache Cassandra which provides super easy horizontal scaling). Due to used technologies, designed architecture is easily scalable and extendable. One of the biggest advantages is that architecture is "cluster-free", which means no cluster resource manager such as YARN or MESOS is needed (like in the case of Hadoop). Usually it is hard to setup whole environment.

Presented solution uses Spring Boot module which contains an embedded servlet container (Tomcat), and compilation of the project source code generates a JAR file, which is possible to run without any special system dependencies (only Java is needed) and complicated configurations. That also allows us to use a popular container solution such as Docker for deploying of architecture. Proposed architecture is also independent of an infrastructure solution and can be run on top of Amazon Web Services, Google Cloud Platform, or on a single machine self-hosted solution. That also means, modules are easy to debug for developers. All of the used components are developed as open-source that brings many advantages in the development stage.

Implementation of the Webalyt architecture is composed of a few types of components which are divisible to two groups, namely system components for configuration and data processing components.

Figure 1 shows complete architecture with interaction between components. The left-hand side of the figure shows a system configuration layer which contains system components. The right-hand side is a streaming layer which processes data. All of the components are described in the following part.

A. Configuration Server

The architecture contains a lot of components and it is hard to maintain because configuration of every component is time consuming. This problem is solved by using a Spring Cloud Config module which "provides server and client-side support for externalized configuration in a distributed system" [21]. All configuration files are kept in a Git repository, that naturally allows changes to auditing. A necessary condition for correct behaviour is visibility of the Webalyt Configuration server across the network. When a new instance of the Webalyt module has been started then settings for the module are get

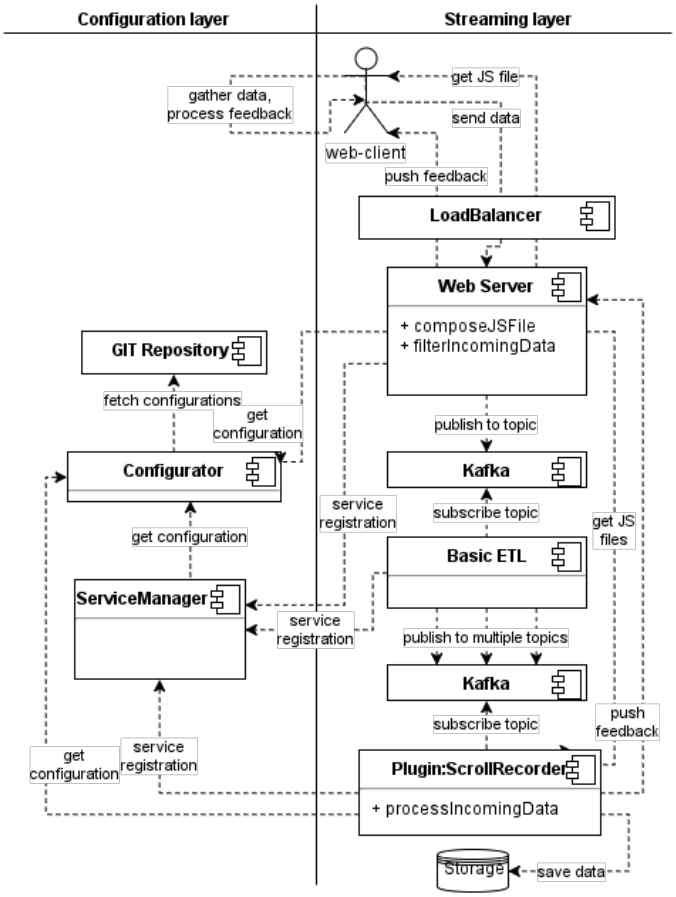

Fig. 1: Architecture of Implementation with Interaction among Components

from the configuration server and used accordingly. For this purpose, the default Spring Cloud Config implementation is used.

B. Service Manager

The Service Manager (SM) gathers and provides information about all registered modules, such as type of service, module name, IP address and port. This information about registered services are used by the Webalyt web-server for serving of JavaScript code to web-browsers. The service discovery is provided via the Spring Cloud Eureka module [22]. From the system point of view, this component keeps watching for states of other modules.

C. Web-Server

The Webalyt web-server is used as an input gateway for data and it is responsible for the following three tasks. The first responsibility is receiving of incoming data which are created by plugins in the clients web-browsers. The second responsibility is checking data format, and the third is pushing to the Apache Kafka's topic.

From this topic, data are moved to the basic ETL processor, which is used for data transformation of the incoming data to a suitable format for plugins.

The Webalyt web-server also collects JavaScript codes of other plugins and composes all of those codes into one file which is served to the client web-browser. This approach reduces the number of HTTP requests between the web-browser and server to one. The final JavaScript file can be minimised and cached. When a new plugin is registered, then the JavaScript code is updated.

The target composed JavaScript file contains a function for sending the created JSON object to the server at regular intervals. This approach is more efficient than sending data immediately to the server every time.

In listing 1, data-structure of JSON generated by plugins which are running in the client web-browser is shown. Generated data which are created by plugins are collected into one main JSON object. As shown, the "mp" object (mouse move recorder plugin) has a collection of captured information about mouse-moves (attributes "x" and "y") in time (attribute "timestamp"). Data are collected when a specific event is fired.

Listing 1: Example of Incoming Data in HTTP Request

```
{
    "pageView": {
     "pageViewId":"a898385d",
     "sessionId":"5edc06c9",
     "pathUrl":"/index.html",
     "websiteId":"101",
     "timestamp": "2018-03-08T16:36:20.232Z"
    },
    "mp":[
        {
          "timestamp":"2018-03-08T16:36:21.577Z"
          "x":904,
          "y":44
        },
        {
          "timestamp":"2018-03-08T16:36:22.611Z"
          "x":920,
          "y":44
        }
    ]
}
```

One of the most important objects is the "pageView". This object contains meta information about a current user. Namely,

- pageViewId – represents currently rendered page. The identification must avoid being the same across all of the system, because all of the plugin data are marked with this identifier,
- sessionId – is an identifier assigned to a user in a time. Identifier is stored in local storage,
- pathUrl – represents initial path of url address,
- websiteId – is a website identifier which is introduced in Webalyt system (multi web-site support),
- timestamp – is assigned when the object is created.

D. Base ETL Processor

The base ETL processor contains an algorithm for generic data processing. The data are got from the Kafka and then processed in the following steps:

1) split JSON object according to direct child nodes (ex.: "mp" and "pageView" in listing 1) whose names are corresponding to the Webalyt Plugins (natural selection of Kafka topic),
2) enrich every JSON object with the identifier "pageViewId",
3) compute Kafka partition,
4) push to Kafka.

E. Apache Kafka

Apache Kafka "is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production, in thousands of companies". Webalyt architecture use Kafka as a publish-subscribe message system for data which have to be processed. The components (basic ETL processor and plugins) are getting data from the exactly specified topic.

F. Plugin

Plugins are composed from the following parts:

- JavaScript code – which will be collected by the web-server and served to the web-client. The JavaScript code produces data which are processed by the ETL process. The code can react to feedback from the evaluation plugins.
- ETL process – prepares data for saving to data storage and other logic can be implemented. Data processing can be stateless or stateful, depending on the implementation of processing.
- Feedback function – sends hint to web-server which propagates data to web-client.

How simple the writing of a plugin can be is shown in listing 2. Insertion of the array "mousePositions" to the object "mp" is automatically provided by the Webalyt Platform based on plugin configuration. As shown in the listing, event listener fires the function every time when mouse moved and it generates a huge amount of data. In this case, data approximation of mouse movement should be considered [23].

Listing 2: JavaScript Code of Webalyt Plugin for Tracking of Mouse Position

```
var mpPlgn = {
    shortName: "mp",
    fullName: "Mouse position plugin",
    mousePositions: [],
    methodBody: function () {
        document.body.addEventListener('
            ↪ mousemove', function (e) {
            var mp = {};
            mp.timestamp = new Date();
            mp.x = e.clientX;
            mp.y = e.clientY;
            mpPlgn.mousePositions.push(mp);
        });
    },
    onInit: function () {
        this.mousePositions = [];
    },
    onAfterSend: function () {
        this.onInit();
    },
    getDataForSending: function () {
        return this.mousePositions;
    }
};
webalyt.addPlugin(mpPlgn);
```

Writing of specialised plugins is very easy, because the platform contains a main module for message processing. Code example 3 shows how easy plugin implementation is. It is sufficient to extend class and override method "processMessage" which processes single item at a time.

Listing 3: Java Code of Webalyt Plugin for Processing of Mouse Position

```
@Service
public class MousePluginEtlService extends
    ↪ AbstractPluginEtl<MousePosition> {
    @Override
    protected void processMessage(
        ↪ MousePosition object) {
            //implement the process logic
    }
}
```

Some plugins have more difficult data transformation or evaluation logic. This leads to lower data throughput and slower feedbacks. Due to modular architecture, it is possible to horizontally scale every plugin. For example, a plugin for processing one instance of mouse scroll module can be enough. But this is not the case for mouse move processing. Just adding a new instance of a plugin will increase the performance.

G. Base Data Structure for Data Storing

Figure 2 shows proposed hierarchical data structure. This research currently uses only "pageview", "session" and "plugins data". For the use of "user" object, implementation of session ID sharing is needed [24]. The data structure reflects gathered data with respect on manner of usage.

Gathered data of each user could contain specific pattern, such as mouse speed, which can be used for data segmentation. For example, it could be possible to recognise how user is experienced in web usage according to this information. Then, it is possible to prepare A/B test with updated layout for every user group.

IV. Discussion and Conclusion

In this paper, implementation of component architecture (based on Spring Boot framework and Apache Kafka) was introduced. While Apache Kafka is a commonly presented and used software in big companies (which deal with real-time data processing), Spring Boot is not so often highlighted. Connection of these two technologies achieved a flexible, extendable, scalable and clear solution for data gathering and processing. Architecture also allows the implementation of plugins in different technologies such as PHP, Scala and etc. Presented implementation, which reflects Kappa architecture,

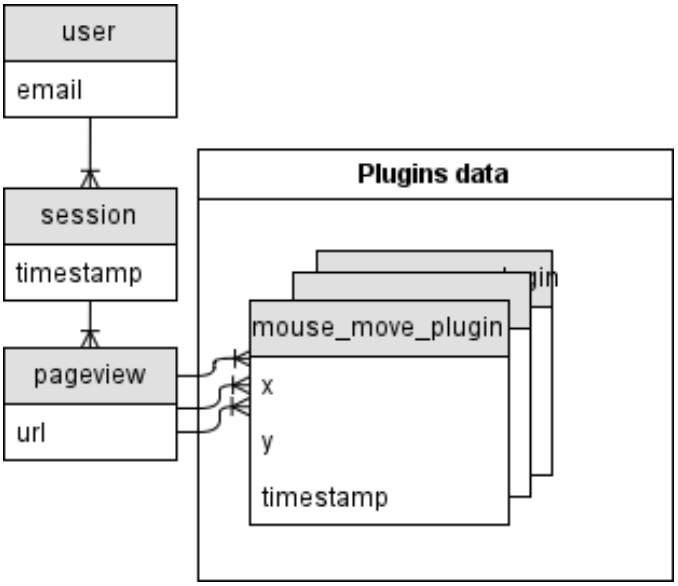

Fig. 2: Proposed Base Data Structure of Data Storage

is actually sufficient, but for the future development, moving to Lambda architecture needs to be considered, because business analysis based on data aggregation (which needs access to a complete set of records) could be the next requirements. Influencing of device resources by data compression and encoding is another issue.

Comparison with previously mentioned software is not possible due to proprietary implementation. Ordinary, data from the client are encoded and sent to the server at regular interval. The data structures of data sets are different and depend on software supplier. In comparison with software for real-time processing, easy development and clear solution were achieved due to Spring Boot framework.

Complete proposed Webalyt architecture is very complex and contains another few components such as modules for data evaluation and visualisation. Future work will be focused on implementation of replaying users' sessions in relation to web-page performance evaluation for discovering bottlenecks in the use of web-pages. It will include implementation of different kinds of plugins which capture all user actions, such as scrolling, mouse movement, and information about system actions such as DOM mutation observation and loading performance. Combination of the results can give a new perspective on how web-page is really used.

ACKNOWLEDGEMENT

The work has been supported by the Funds of University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

REFERENCES

[1] M. Nebeling, M. Speicher, and M. Norrie, "W3touch: Metrics-based web page adaptation for touch," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. New York, NY, USA: ACM, 2013, pp. 2311–2320. [Online]. Available: http://doi.acm.org/10.1145/2470654.2481319

[2] L. Čegan and P. Filip, "Webalyt: Open web analytics platform," in *2017 27th International Conference Radioelektronika (RADIOELEKTRONIKA)*, April 2017, pp. 1–5.

[3] matomo.org, "#1 free web & mobile analytics software," 2018. [Online]. Available: https://matomo.org/

[4] S. A. Miller, *Piwik web analytics essentials*. Packt Pub., 2012.

[5] P. Adams, "Web analytics. open source." 2018. [Online]. Available: http://www.openwebanalytics.com/

[6] Google LLC, "Google analytics collection limits and quotas," 2016. [Online]. Available: https://developers.google.com/analytics/devguides/collection/analyticsjs/limits-quotas

[7] ——, "Compare free and enterprise versions - google analytics solutions," 2018. [Online]. Available: https://www.google.com/analytics/analytics/compare/

[8] I. Bekavac and D. G. Praničević, "Web analytics tools and web metrics tools: An overview and comparative analysis," *Croatian Operational Research Review*, vol. 6, no. 2, pp. 373–386, 2015.

[9] Mozilla.org, "Introduction to web apis," 2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction

[10] J. Lin, "The lambda and the kappa," *IEEE Internet Computing*, vol. 21, no. 5, pp. 60–66, 2017.

[11] N. Marz and J. Warren, *Big Data: Principles and best practices of scalable realtime data systems*. Manning Publications Co., 2015.

[12] V. Gurusamy, S. Kannan, and K. Nandhini, "The real time big data processing framework: Advantages and limitations," vol. 5, pp. 305–312, 12 2017.

[13] W. Inoubli, S. Aridhi, H. Mezni, and A. Jung, "An experimental survey on big data frameworks," *arXiv preprint arXiv:1610.09962*, vol. abs/1610.09962, 10 2016.

[14] X. Liu, N. Iftikhar, and X. Xie, "Survey of real-time processing systems for big data," in *Proceedings of the 18th International Database Engineering & Applications Symposium*, ser. IDEAS '14. New York, NY, USA: ACM, 2014, pp. 356–361. [Online]. Available: http://doi.acm.org/10.1145/2628194.2628251

[15] M. Stine, *Migrating to Cloud-Native Application Architectures*. O'Reilly Media, Inc., 2015. [Online]. Available: http://www.oreilly.com/programming/free/files/migrating-cloud-native-application-architectures.pdf

[16] M. Eisele, *Modern Java EE Design Patterns*. O'Reilly Media, Inc., 2016. [Online]. Available: http://www.oreilly.com/programming/free/files/modern-java-ee-design-patterns.pdf

[17] C. Richardson, "What are microservices?" 2017. [Online]. Available: http://microservices.io/

[18] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st ed. O'Reilly Media, February 2015.

[19] M. Richards, "Microservices, antipatterns and pitfalls," 2016. [Online]. Available: http://www.oreilly.com/programming/free/files/microservices-antipatterns-and-pitfalls.pdf

[20] J. Kreps, N. Narkhede, J. Rao *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, 2011, pp. 1–7.

[21] Pivotal Software, Inc, "Spring cloud config," 2018. [Online]. Available: https://cloud.spring.io/spring-cloud-config/

[22] Pivotal Software Inc. Spring cloud netflix. [Online]. Available: http://cloud.spring.io/spring-cloud-static/spring-cloud-netflix/1.3.1.RELEASE/

[23] L. Čegan and P. Filip, "Advanced web analytics tool for mouse tracking and real-time data processing," 2017.

[24] M. Falahrastegar, H. Haddadi, S. Uhlig, and R. Mortier, "Tracking personal identifiers across the web," in *Passive and Active Measurement*, T. Karagiannis and X. Dimitropoulos, Eds. Cham: Springer International Publishing, 2016, pp. 30–41.