Design and Development of Student Activity Record Systems Using the Concept of Reactive Streams in Data Stream Processing

Aila Gema Safitri¹, Denny Chandra² and Mervin Tangguar Hutabarat³

¹Informatika, Universitas Muhammadiyah Bandung, Jl. Soekarno Hatta No.752, Cipadung Kidul, Kec. Panyileukan, Kota Bandung, Jawa Barat 40614
²Pusat Penelitian Teknologi Informasi dan Komunikasi ITB, Gedung PAU Lantai 4 PPTIK ITB, Jl. Ganesa No.10, Jawa Barat 40132
³STEI ITB, Labtek VIII/Gedung Achmad Bakrie, Jl. II, Lebak Siliwangi, Coblong, Bandung City, West Java 40132

*ailagemasafitri@gmail.com

Abstract. Student Activity Record System is an e-learning module that is used to record student activities in the context of behaviour assessment. Student and teacher sent datas in the form of photos, videos, texts and geolocation coordinates. With the number of requests sent by the user, the load from the server side will continue to gain weight. It requires an efficient way to use the resources on the server side, namely Central Processing Unit and memory. The using of Netty as an asynchronous event driven web server and the reactor libraries as a reactive streams implementation on the server side become the right choice to optimize Central Processing Unit and memory usage. This research emphasizes the design of system architecture and library selection that implements the concept of reactive streams on the client-server side. The reactive streams concept is applied to web dashboard and mobile application so that web and mobile application become more responsive because execution is asynchronous.

1. Introduction

Application performance especially on the server side is very important. Many application become slow even become inaccessible when they get requests that are increasing intensively from the user. The impact is application does not run as expected. It happened because the use of resources from the server (Central Processing Unit and memory) is high meanwhile the available resources are limited. Even it can result the requests is not being processed if all of the resources are being used.

The Student Activity Record System (SARS) is an e-learning module that is used to record student activities through a smartphone. This application aims to record student activities in the context of student behaviour assessment. The assessment rubric used is a holistic assessment. A holistic rubric is a rubric that uses a single score in assessing all performance [1]. In SARS, the holistic assessment means that every activity carried out by students has a certain assessment weight. Students do the activities and there is an evidence in the form of photo or video then the teacher can validate the activities.
2. Background

2.1 Reactive Stream

Reactive Stream is an initiative to provide standards for asynchronous stream processing with non-blocking and back pressure. This encompasses efforts aimed at Java Virtual Machines and JavaScript and network protocols. The purpose of reactive stream is to govern the exchange of data streams through asynchronous while ensuring that the recipient side is not forced to do arbitrary buffers. In other words, back pressure is an integral part of this model to allow queues that mediate between threads to be limited [2][8]. The Reactive Stream specification consists of four interfaces, namely: Publisher, Processor, Subscriber and Subscription [3][9].

![Diagram of flow in implementing reactive streams](image.png)

**Figure 1.** Illustration of flow in implementing reactive streams [3][9]

The interface can be explained as follows:

- **Publisher**: a publisher is a data provider that can publish large amounts of data to subscribers. The speed of the published data depends on the subscriber. The interface of the publisher has one method, `subscribe` which is used to register a new subscription for subscribers. This method can be called many times at any time. Publisher can stream data to several subscribers.

- **Subscriber**: a subscriber is the final recipient of the data stream from the publisher. The subscriber interface consists of four methods, namely `onSubscribe`, `onNext`, `onError`, and `onComplete`. The `onSubscribe` method is run by the publisher when the `subscribe` method is called to register a subscription. No data is passed to the subscriber until the request method is run on subscription. When data is streamed and received by a subscriber, the `onNext` method is run to process data. If an error occurs, the `onError` method is run to resolve the error. When an error occurs, no other event is executed until the error is resolved. When the data flows ends and no data is added, the `onComplete` method is run.

- **Subscription**: a subscription represents a one-to-one connection between a publisher and a subscriber. Subscription can only be used by one subscriber and when the subscription is canceled the used resource will be cleared. The subscription interface has two methods, namely `request` and `cancel`. Publisher will not stream data before the request method is called. This method is called by the subscriber to communicate with the data flowed. This method can be called several times. The method argument must be a positive number.

- **Processor**: A processor plays a role among publishers and subscribers. This means that the processor can order to the publisher and stream the data to a subscriber. The processor is only used between a publisher and a subscriber to process the data stream and then stream the results to a subscriber [3][9].

2.2 Reactive Programming

Reactive programming is part of the asynchronous programming paradigm where the availability of new information drives the next logic rather than the flow control that is executed by thread execution. Reactive Programming supports problem solving into several discrete steps where each can be executed
in asynchronous and nonblocking modes and can then be arranged to generate workflows - maybe not limited to input and output [4][2].

2.3 Asynchronous Web Server
Modern web servers are equipped with the ability to process multiple requests simultaneously, to meet the demands of a high workload charged to them. Concurrency can be implemented on a web server using both synchronous and asynchronous capabilities provided by the operating system. Although synchronous mechanisms are widely used, asynchronous mechanisms might be interesting because they provide concurrency benefits while reducing a lot of overhead and multi-threading complexity [5][10]. One of the web server that has asynchronous processing capabilities based on Java Virtual Machine is Netty [6][7].

2.4 Reactor
Reactor is a fourth-generation reactive library based on Java Virtual Machine which is an implementation of the Reactive Streams specification to build non-blocking applications [7] [4]. The reactor is a standard library used by the Spring Framework to support the reactive concept [8][5].

2.5 Spring Webflux
Spring webflux is a module in the spring framework that supports reactive HTTP and Websocket reactive as well as for reactive web server applications including REST, HTML browsers, and interaction with WebSocket [8][5].

3. Analysis and System Design
This research is divided into several stages as follows:

- Preparation phase: the preparatory phase starts with formulating problems regarding the research topic, setting goals, determining problem boundaries, to conducting literature studies related to the chosen topic. The literature material for this study was obtained from books, articles, and the internet.
- Design phase: at this stage discussed the system specifications to the system design. System specifications are related to the model that will be implemented while the design system is divided into designs on the developer side and on the user.
- Development phase: at this stage the development process of the Student Activity Record System design is carried out.
- Debuging phase: at this stage the application is debuged for watching the flow or reactive stream processing on the server side.

3.1 Architecture
The Student Activity Record System application consists of mobile, web and server side applications for processing data that will be stored on the database server. The database used is MongoDB. The client-side application consists of a mobile application for students / parents that is used for publication of activities and used by teachers to validate the activities carried out by students. While the dashboard application is used by the school admin to do activity data entry. Following is the architectural picture of Student Activity Record System:
The server-side application consists of a Web Server using Netty, and the MongoDB Server. The web dashboard is created using the library Angular version 6 [11] and PrimeNG [12]. Mobile application development uses native Java programming language. Natively selected because there is already a UI Library called material design that is ready to use to create applications.

4. System Development
4.1 Implementation of Reactive Streams
The web dashboard is created using the version 6 of angular framework that has an asynchronous execution API. Examples of the implementation of the use of asynchronous execution code are the function of storing master data activities:

```
this.http.post(this.globalService.BASE_URL + '/master/aktifitas',
  this.master,
  {params: params})
.subscribe(
  result => console.log(result),
  error => {
    console.log(error);
    this.ngs.push({severity: 'error', summary: 'Error',
      detail: error.error.error.error_messages});
  },
  () => {
    setTimeout(() => {
      this.initTable();
    }, 1000);
  });
```

Figure 3. Examples of asynchronous code on a web dashboard

The above code uses the HttpClient module that is in the @ angular / common / http namespace. The function that is called is a post that returns Observable. An Observable can emit a single packet data or emit streams containing multiple discrete packets. When the function is run, the browser will send data to the server asynchronously so that the user interface is not blocked and when the data is in the sending process, the user can do something else. Server-side applications use the webflux spring module that internally uses a reactor library. Examples of implementing the code are when retrieving master data activities:
Figure 4. Examples of asynchronous code on the server side

From the above code it can be seen that the logic of the program is sequentially in accordance with the data flow which is characteristic of the event driven concept. And the code is a response from requests that are executed asynchronously by Netty's web server.

4.2 Debugging

Debugging is done by running a web dashboard and mobile application. From the side of the dashboard and mobile asynchronous processes can be seen using the postman tools. Whereas from the server side application can be seen using intellij debugger. The initial process before students use the mobile application is data input done by the school admin. Admin inputs the types of activities which consist of activities that must be carried out by students. Each activity has weight to determine the assessment of activities carried out by students. To prove that the communication process to the server runs asynchronously, it can be seen in the postman:
Figure 5. The process of sending to the server asynchronously

After the activity is inputted, students can download the activity and then post the activity. Testing server-side applications using the debugger feature on IDEA intelligence. When end point API /api/rest/initapp//masteraktifitas/{kategorId}:

Figure 6. Server-side debugging
From the picture above, it can be seen that the subscription process is `onNext` and `onComplete` which is the concept of reactive streams.

5. Conclusion

In this research has been designed and developed *The Student Activity Record System* in web dashboard, mobile and server-side applications. Its implement the concept of reactive streams. The next research is focus on proving the efficiency on the server side after implementing the concept of reactive streams on the server side which executes programs asynchronously, non-blocking threads, parallel execution and backpressure.

References

[1] S. C. Wortham, Assessment in Early Childhood Education, New Jersey: Pearson Merril Prentice Hall, 2005.
[2] "Reactive Stream," 17 Juli 2018. [Online]. Available: http://www.reactive-streams.org.
[3] G. Hochbergs, Reactive programming and its effect on performance dan the development process, Departement of Computer Science Faculty of Engineering LTH, 2017.
[4] V. K. Jonas Bonér, "Reactive Programming Versus Reactive Systems," Lightbend Inc, 2017.
[5] S. G. A. G. J. G. U. Praphamontripong, "Performance Analysis of an Asynchronous Web Server," in In Proceedings of the 30th Annual International Computer Software and Applications Conference, 2006.
[6] "Netty Project," 15 Agustus 2018. [Online]. Available: https://netty.io.
[7] "Spring by Pivotal," 15 Agustus 2018. [Online]. Available: https://projects.spring.io/.
[8] "Spring Data MongoDB," 18 Agustus 2018. [Online]. Available: https://projects.spring.io/spring-data-mongodb.
[9] "Angular," 18 Agustus 2018. [Online]. Available: https://angular.io/docs.
[10] "PrimeNG," [Online]. Available: https://www.primefaces.org/primeng/#/.

Acknowledgement

This research has been supported by Ministry of Research and Technology as a funder institution also the research team from PPTIK ITB and Informatika Universitas Muhammadiyah Bandung.