Optimizing economics of microservices by planning for granularity level

Experience Report

O. Mustafa
University of Oldenburg
Oldenburg 26129
Germany
ola.mustafa@uni-oldenburg.de

J. Marx Gómez
University of Oldenburg
Oldenburg 26129
Germany
Jorge.marx.gomez@uni-oldenburg.de

ABSTRACT
Microservices application architecture emerged as a special application architecture of Service Oriented Architecture (SOA). It is also known as cloud native applications architecture. For component based web applications, the adoption of microservices requires an extensive refactoring process to be evolved to microservices based application. Current practice for the main activity in this refactoring process, is following the white box based techniques which requires enormous time to analyze all business processes in the business domain and not considering non-functional requirements such as performance and resource consumption. This paper expresses the need for such consideration through an empirical experiment followed by proposing a black box based technique, based on extending the utilization of web usage mining techniques.

CCS CONCEPTS
• Software Architecture → Service Oriented Architecture (SOA); Microservices • Web Based Applications → Cloud Native Applications; Migration; Optimization

KEYWORDS
Microservices, Cloud native applications, Microservices based web applications, SOA based web applications

1 INTRODUCTION
Service based applications and microservices are widely adopted by e-commerce applications, motivated by the high flexibility for this special type of application architecture, and its ability to adapt in front of extremely high workloads, that what let this architecture to be considered as a native architecture for cloud based applications [1]. Moreover, cloud business models like pay per use, are making the optimization of web based applications reflecting directly on the final invoices for cloud service providers. Microservices architecture proved through previous research efforts [2], its high ability to optimize utilizing resources over the cloud, compared to other known web application architectures.

In the last few years, very large e-commerce applications announced the migration of their web based applications architecture to microservices, to gain the benefits of higher performance and scalability besides better resource utilization over the cloud [3]. The main idea of this architecture is not only to isolate the business layer from the presentation layer, but also to split the business layer (back end) into a group of small web services (microservices), those of which are triggered from one single Interface called the application gateway. The migration of an existing web application to microservices architecture implies the process of splitting the application code to a group of small web services, so that each service contains a unique part of the business processes, this part should be highly decoupled and isolated from other parts [4].

Figure 1 shows two different web application architectures; the left one follows the component based architecture, where the whole code is released as one large body, no matter how it is arranged inside (factions, classes, libraries, etc.). On the other hand, microservices architecture is shown on the right hand where the application is divided into small isolated services all are communicated through the main interface (application gateway). The question of how big the service should be or in other words, how many business processes each web service should encapsulate, is a main design issue which is known as service granularity [5]. That is because low granularity (which implies very small and high number of services) will increase the complexity of managing those services and high granularity (which implies bigger services and few number of services) will
not achieve the desired economic design over the cloud, where resource consumption is calculated per use.

The economic impact caused by granularity level in microservices based web applications, is what the empirical experience within this paper examines. The main hypothesis to be examined is that different granularity levels have different impacts on performance levels and resource utilization levels, for the same workload.

This paper lies in four main parts; the first is to discuss some related work, those of which mentioned microservices architecture design and the way taken toward it. This part aims to spot the light on the gap filled by this research. The second part discusses the results of an empirical experiment, which reveals the impact of splitting loaded business process into two microservices. The aim of this part is to proof the relation between planning for the sustained granularity and the economic microservices design. The third part proposes a clustering based solution to extract the workload pattern from the web access logs for a web application and using that pattern to decide the sustainable microservices design to be taken in the future. Finally, the last part applies the proposed solution to validate it by applying the same load test again on the new generated design.

Merging all parts of this paper; a new application for web usage mining is proposed, motivated by the need to have structured approach that guarantees an economic microservices design for cloud based applications.

2 Related work

Refactoring applications to microservices architecture, would require a decomposition phase, to break the architecture into loosely coupled services. The objective from discussing the related work is to analyze the strategies taken to decompose the monolithic architecture. The discussed papers are those from the literature, re-factored the monolithic architecture to microservice architecture, driven by the need to evolve into scalable applications on the cloud. This is due to the main feature of microservice architecture; which is to scale effectively in front high workload [2].

The experience report [6], mentioned the requirements for scalability, which considered to be fulfilled by applying the stateless design of services and REST. The decomposition phase was not mentioned. The empirical evaluation experiment [2] evaluated the microservice architecture under peak loads, compared the generated cost with the monolithic architecture under the same load. The better performance and the cost efficient of resource utilization concluded for the microservice architecture. Both architectures tested under the vertical scaling method only, and the decomposition phase was not mentioned.

In [7], the author mentioned the microservices architecture as a novel architecture and considered it as the cloud native architecture. The scalability needs on the application level addressed, fulfilled by refactoring the application into service based architecture, and the decomposition is done based on domain entities. The report did not mention any tests to evaluate application scalability after migration. Development and deployment complexities were the main challenges expressed. In [8], the author proposed a recursive approach for decomposing the business logic into web services and the metric of loose coupling was the only one mentioned. The need for an adaptive architecture to scale addressed, to serve the increasing demand on peak loads. In [9], a method for model based automated transitioning the monolithic architecture into objects and services was proposed, the refactoring motivated by the need to improve the performance efficiency, by processing large volumes of data in parallel. Scaling infrastructure resources was not mentioned. For identifying services, a recommendation engine built based on applying clustering techniques with loose coupling metrics, services decomposed on class level.

The previous mentioned papers and reports, studied microservice architecture but they did not mention the decomposition phase, how it was done or based on what. Which is normally refer to the experience of the software architect to decide the decomposition method.

The researchers in [10], referred to this issue; where most references about microservices explained the decomposition phase as an art and totally depends on the experience of the software architect. Based on that, the authors proposed a tool for decomposing microservices (Service Cutter) with a basic concept of applying different cluster based algorithms on the business process model. The tool is said to be the first structured approach for decomposing microservices.

Cloudmig model based approach [11], mentioned the need for economic design, for scalable cloud based application. The approach proposed by the authors is basically an optimization approach combining the business process model, the utilization models and the cloud pricing in order to reach the most economical design.

In all related work mentioned previously, the need to evolve the application’s architecture to a scalable one using microservices is mentioned in all papers. Most of them did not mention the decomposition approach in details. While the mentioned approaches for decomposition is heavily relying on starting from the business process model. Which is basically a white box based approach requires a lot of analysis and could not be used as a decision support tool nor as an optimization tool for designing microservices based applications. This paper will propose a practical and economical approach for decomposing microservices based on that we claim the sustainability of this design approach.

3 Micro-BookShop

Micro-BookShop is an online book shop, providing the main e-shop functional requirements (browse, search, order) as well as the main shop-cart features. It is part of an ongoing research, aims to investigate the impact of service granularity on the performance of microservices applications over the cloud. Moreover, the impact of considering the workload pattern onto deciding the level of granularity which leads to sustainable and
Optimizing economics of microservices by planning for economic performance on the cloud, based on the upcoming results.

The presentation layer for Micro-BookShop web application is implemented using Angularjs, while the backend of service layer is implemented with VS.net 2015 (web api/c#) where REST is the only protocol used for communication between web services. Micro-BookShop is deployed on cloud azure, the database used for this experiment is MS SQL Server 2016. The data are real and scraped from amazon.com with 5000 records. Finally, all load testing for this experiment implemented using visual studio team services on azure cloud and the workload generated automatically using the technique of record and play.

The main action of this empirical experiment is to observe the response time and the total resource consumption for the browse scenario, under special generated workload, over two different levels of granularity. The first level, is of which the whole browse functions are encapsulated in one single web service. While the second level is of which the browse functions are separated into two web services. The next section will explain the details about the browsing scenario and the specifications of the load tests.

4 Browsing scenario under load testing

Referring to Figure 2, let us consider two different versions for designing the browsing scenario; the first version contains one web service (the left design), responsible for retrieving all book details once the book id received as a parameter; it is encapsulating functions for getting book details, getting book reviews and getting other recommended books related to the book id. The second version (on the right) splits the functions for retrieving book reviews in another new and isolated web service, and the first one remains for retrieving all book details, and the books recommendations. The reason behind isolating the functions of retrieving reviews is because the reviews for this book shop presents a rich text which makes the action of retrieving details for the books with high number of reviews very heavy action to perform.

![Figure 2: Microservices version 1 (left) vs. Microservices version 2 (right).](image)

This imperial experiment aims to test the two versions of browsing scenarios under high generated workloads; three workloads with 5000, 10000 and 15000 users per minute. The main benchmarks to be collected for each workload and each design are the response time and CPU utilization.

The generated workload for the first version, where all browsing functions encapsulated into one web service, was planned to heavily loading book reviews pages in particular, by moving to the full reviews page each time, instead of reading the top reviews or the latest ones in the book details page. After applying this workload again on the second version of granularity, where the functions of retrieving book reviews were separated into a new web service, the response time recorded an improvement for all applied workloads. Fig 2. Shows 3 improvements recorded for each workload those of which .9 ms, 1.9 ms and 4.6 ms improvements recorded for the second design. Fig 3. Shows another improvement for CPU utilization recorded also for the second version; 13.7%, 8.4% and 15% saving (also less) recorded. This means that splitting the web service into two web services has improved the response time under load testing which considered as performance improvement, on the same time the resource consumption decreased under the same load testing.

The improvement in the performance and the resource consumption is the situation of sustainable and economical performance which this research aims to reach. The next section explains about workload characterization and how it could be extended to help us reach the sustainable microservices design, the design of which performing high with the minimal resource consumption.

![Figure 3: Average response time results for two different design versions of browsing scenario.](image)

5 Workload characterization

ProWeb 2017 Programming Technology for the Future Web, April 2017, Brussels, Belgium
Investigating the impact of workloads over microservices based applications stared to activate lately [12],[2]. However, extending the web usage mining techniques to optimize the granularity of microservices is not mentioned yet. Domain Driven Design is the main base for all granularity based decisions, where the microservices are splitted based on the application domain [4]. This approach will be enough to generate the basic level of granularity but not the optimized one. This is due to that business domain analysis may not reveal some high loaded parts of the running system. As in our example, splitting the book reviews into isolated service, improved the response time of the whole application. The load came from such function (getBookReviews), might be predicted by the software architect according to his experience, this is true for simple and known business scenarios. However, this experience would not be enough for more complex scenarios and should be supported with a structured approach as this research aims. This is possible especially with the existence of rich source of users navigation record for the web pages, through web access logs.

Mining web access logs to extract useful information about user behavior has been extensively investigated in research [13]. The mining process goes through main steps to clean the logs and prepare them for sessionizing, this is the main process where the users are distinguished based on the IP address and the time stamp over different web pages of study. Different researches applied different clustering approaches on the huge logs, in order to extract the usage pattern and then making adaptive decisions. Such decisions were focusing on enhancing the user experience such as, designing recommender systems or adapting the design of the interface.

In the same domain of web usage mining, few works extended usage pattern extraction to the level of workload pattern extraction. Even though, such extensions were not focused on adapting the application architecture, but rather to generate a workload models as input for simulated load testing as in [14].

In Figure 5, the main distribution of user sessions and web pages which later will be the base to generate the user pattern. This approach is very useful when the target is to enhance the user experience. However, it could be reformed to extract the workload distribution pattern between the web pages, to see which pages are attracting the high loads within certain periods of time (time slots).

This reformation requires that the main sessionizing phase to be based on time stamps rather than on user sessions (IP addresses), as in Figure 6. The next step is to decide how to cluster time periods into (peaks), to decide which pages attract high loads or user visits. Here where the application of fuzzy based clustering sounds proper and serving the need as will be clarified in the next section.

6 Fuzzy clustering for extracting workload patterns

Among many usage mining approaches [15],[13],[16], unsupervised methods, especially clustering, used to group users with common browsing behavior. However, while k-means clustering used for huge access logs because of its computational efficiency [17], it still suffers from the absence of possibility to obtain overlapping clusters with current approaches, so that a user can belong to more than one group [18]. Fuzzy clustering on the other hand, could deal with the ambiguity and the uncertainty underlying web access data, this is by assigning weights to URLs and user sessions based on a fuzzy membership function.

With session weight assignment approach, time slot based sessions are proposed to weight the sessions over the time stamps rather than the users. After assigning the weights we move to the step of applying a "Fuzzy c-Mean Clustering" algorithm to discover the clusters of peak periods rather than user profiles.

Figure 7 shows the clustering result in the case of user based sessionizing compared to our proposed approach in Figure 8 which shows how the weight assignment is to be applied over the time stamps extracted from the logs, in order to cluster the main peaks.

It is also important to mention that, user sessions are distributed over the pages as in Figure 7, while in the peak based approach, peaks are distributed over page categories as in Figure 8.
Optimizing economics of microservices by planning for granularity level

<table>
<thead>
<tr>
<th>User 1</th>
<th>User 2</th>
<th>User 3</th>
<th>User 4</th>
<th>User 5</th>
<th>User 6</th>
<th>User 7</th>
<th>User 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.html</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B.html</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C.html</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>D.html</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 7:** User based sessionization leads to clustering over web pages.

<table>
<thead>
<tr>
<th>Peak 1</th>
<th>Peak 2</th>
<th>Peak 3</th>
<th>Peak 4</th>
<th>Peak 5</th>
<th>Peak 6</th>
<th>Peak 7</th>
<th>Peak 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cat B</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cat C</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cat D</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Figure 8:** Time slot based sessionization leads to clustering over web pages categories.

This approach allows generating peaks oriented sessions rather than user oriented ones. Which also allows the observation of workload distribution over the web pages. Finally, those web pages with (higher) workloads, should have the priority to be splitted to microservices.

Mapping to the example of browsing scenario, if the web logs prepared and processed, followed by a time slot based sessionization, book reviews pages will show high workloads (weights) and as a result, book reviews will have a higher priority to be splitted on two separated web services, that is for the new version of the microservices architecture, this action expected to improve both the performance and the resource consumption under high workload compared with the old version, where getting book reviews functionality is encapsulated with book details service into one single web service.

This approach allows generating peaks oriented sessions rather than user oriented ones. Which also allows the observation of workload distribution over the web pages. Finally, those web pages with (higher) workloads, should have the priority to be splitted to microservices.

### 7 Method Validation

In this section, a new microservices design (a third one) would be generated when applying the proposed approach with its second part (leading paths analysis). That is based on that when we analyzed all lead paths to the book details page, 30% of the leading paths generated from the recommendation functionality. Based on that; the new suggested design based on our proposed approach is based on this result and applied by splitting the recommendation functionalities in a separated web service (see Figure 9).

**Figure 9:** Microservices version 3 where proposed method applies.

Moving further, by applying the same previous workloads on a structured load testing, a further improvement for both response time and CPU utilization noticed, compared to the initial two microservices designs (see Figure 10 and Figure 11).

**Figure 10:** Average response time results for all three different design versions of browsing scenario

**Figure 11:** CPU utilization results for all three different design versions of browsing scenario

### 4 CONCLUSIONS

In this paper, we have discussed the related work in the literature those of which mentioned microservices architecture design and the way had been taken toward it. We tried to spot the light on
the gap aimed to be filled by this research. In the next part, we have discussed the results of an empirical experiment, which reveals the impact of splitting loaded business process into two microservices. This experiment used in this paper to proof the relation between planning for the sustained granularity and the economic microservices design. As a result, a black box based solution proposed in order to extract the workload pattern from the web access logs for a web application. The proposed solution aims for using this pattern to decide the sustainable microservices design to be taken in the future.

This solution could be considered as a new application for web usage mining, and will open wide new doors for black box based techniques specially for optimizing the feasibility of microservices based applications.

The final part of this paper, shows the validity of the proposed solution by applying it and by generating a new microservices design based on it and then, applying the same load testing on it. Results shows that the most optimized microservices design in terms of response time and resource consumption is for the last design.

The importance of this approach lies in two main strength points; the first one is that this approach is not starting from any business domain model (white box technique), which makes it practical and easy to apply, the only assets required is the web logs. The second point is that it will prioritize the decomposition action which will lead to decomposing only loaded services, as a result the level of complexity which will result because of managing the new services will stay controlled.

The future work is to implement the proposed approach and convert it to hard coded then to apply it on large web application with more scenarios and more benchmarks rather than the response time and the resource consumption.

ACKNOWLEDGMENTS

This work was fully funded by the Erasmus Mundus Programme, Phoenix Project - Action 2. Implemented in the University of Oldenburg, Computer Science Department, Very Large Business Applications VLBA, Germany.

REFERENCES