Circuit Breaker in Microservices: State of the Art and Future Prospects

Falakah*, Kridanto Surendro, Wikan Danar Sunindyo
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganesha 10 Bandung, Indonesia

E-mail: *falakah@telkomuniversity.ac.id

Abstract. Resilience is one of the important issues in the implementation of microservices architecture, and to handle it, various approaches and solutions are available. A circuit breaker is a solution to handle failure in microservices and is now widely implemented both as a library and as a pattern embedded in the service and client modules. Despite the widespread use of circuit breaker, the research on circuit breaker is relatively less than research subject on other microservices area, or in microservices resiliency. This article provides an overview of recent research in circuit breaker, map the research subject, and find the opportunities for future research. The research is conducted using a systematic map study but the results focused on extract the knowledge and find important keywords to build a conceptual overview of the research field circuit breaker. The result shows that there are five categories of research subjects which are concept, strategy, management, implementation, and products. After exploring each category, we can propose the opportunity in enhancing the use of circuit breaker in microservices architecture, by explore the potential implementation of proxy circuit breaker.

1. Introduction
The popularity of microservices architecture kept arise in recent year. Lots of research has been conducted in the field of microservices and it still have some issues need to address on implementation. Some important issues such as managing microservices, monitoring, DevOps culture, integration testing, and fault tolerance [1] are emerging along with the rise of complexity in microservices implementation. The basic of MSA principles that the services can be provided by a different organization and running on different platforms, make resilience become an important issue. The resilience can be viewed from two perspectives, from the client that accessing the services and from the service provider. In the microservices-based system, the client does not interact directly with the service provider, so the responsibility of resiliency that should be a consideration for sides, the client and service provider, usually not as easy as in a monolithic-based system.

The usage of the circuit breaker as a pattern for resilience has been populated since it first mentioned in Martin Fowler’s blog [2]. Lots of technology available for supporting the implementation of the circuit breaker in microservices. But there is very little research interest on circuit breaker, as a specific topic in microservices resilient strategy. Most of the research articles mentioned the circuit breaker as a pattern that usually exists in MSA, but not much of them discussed the detailed process on the circuit breaker.
This research aims to find the opportunity for extending existing research on the circuit breaker, by reviewing existing state-of-the-art, classifying, and categorizing existing research on the circuit breaker, exploring the issues and challenges, and finding the opportunities for future research. The rest of the paper will be organized as follows: section 2 will discuss the method we use, section 3 will explain the basic concept of the circuit breaker, section 4 will discuss the research subjects we extracted from recent literature, detailed breakdown and connection between them, and proposed the map of the research subject on the circuit breaker, section 5 will discuss the future research opportunities, and section 6 is the conclusion of our research. To make it easy for writing, we will use some abbreviations according to our topic of discussion as follows: Microservices Architecture (MSA), Circuit Breaker (CB), and Research Questions (RQ).

2. Research Method
The objective of this research is to explore the role of CB in MSA and find the opportunities for enhancing the utilization of circuit breaker on MSA. The motivation of our research is drive by the fact that there is little interest in research that has focused on CB, and most of the research reports mention CB as a component needed to maintain MSA resilience. We conduct the research using a systematic mapping study approach, but we choose to present the result by discussing and summarizing the knowledge extracted from various literature. The steps for doing the research are as follow:

1. Define research questions as the motivation for this study.
2. Define searching strategy that includes source for searching, search criteria or keywords, and inclusion and exclusion criteria.
3. Analysis and extract the data from selected resources.
4. Result, discussion, and summarize the finding from the study.

2.1. Research Questions
We define 3 research questions and backgrounds for motivation. The goal of this research is to find the opportunity of enhancing the use of CB in MSA, based on existing implementation, both concept and the technologies available. From the papers and articles that discuss the MSA, most of them mentioned the circuit breaker as a part of resilience, but only little provides a detailed description of how it works [3]. To achieve our goal, we define some research question as shown on table 1.

| Research Question | Motivation |
|-------------------|------------|
| RQ1: What is the role of Circuit Breaker in MSA? | Get a basic understanding of how the CB works in MSA and find the internal structure of CB. |
| RQ2: What is the research subject of the circuit breaker? | Understanding the problem space of CB based on recent research and its result. |
| RQ3: Are there any opportunities for enhancing the use of circuit breaker in MSA? | Finding the opportunity for enhancing the use of CB in MSA |

2.2. Search Strategy
We searched for articles from several databases such as ACM, CiteSeer, IEEE Xplore, Wiley, and Springer Link. We also consider the articles that come from blog and whitepaper, to cover recent issues in CB implementation in practices and industries.

| Major terms | Synonym terms |
|-------------|---------------|
| Microservices | Architecture, pattern |
| Resilience | Performance OR Failure OR failover |
| Circuit breaker | Communication OR closed OR half-open |

Microservices is an emerging architecture style, lots of interesting discussions, and examples on the blog, and some of the sites consist of huge information such as microservices.io and martinfolwer.com.
We also collected some information from GitHub as media for discussion among programmers. In the searching process, we did the four steps as follows: define the major keys, identify alternative words, synonyms, or related terms to major keywords, and verify the contents of articles are matching with the major keyword. We focused on CB in MSA, because the term circuit breaker is commonly used in some field such as electrical circuits, computer networks, and electrical power. Table 2 shows the keywords using in the searching process.

The papers that were found then evaluated using some criteria to find the most important and relevant papers that can help us answer the research questions. We use the criteria as follow:

Inclusion criteria:
1. The article discusses the concept of circuit breaker and its implementation in a microservices architecture.
2. The article shows the example of circuit breaker implementation.
3. The article reports the best practice on circuit breaker technologies.

Exclusion criteria:
1. The article discusses the implementation of circuit breaker outside of microservices, such as in the electricity field or networking system.
2. The article does not describe detailed information about the circuit breaker.
3. The paper was older than five years, to find the most recent improvement in circuit breaker techniques and its implementations.

After implementing the search criteria above, we select 12 papers published between 2015-2020. From the papers we choose, we can identify some important keywords and can extract some important information from the papers.

Figure 1. Mechanism in Circuit Breaker [5]

3. Concept and Structure of Circuit Breaker
This section will answer the first research questions: what is the role of CB in MSA? Since it was mentioned in Martin Fowler’s blog in 2014 [1] as a technique for preventing the error cascade on the system, some articles and blogs discuss the concept of CB. The idea first mentioned in Nygard book [2] that proposed some pattern to keep the stability of the software, which are: circuit breaker, timeouts, decoupling middleware, handshaking, and test harness.

The benefit of using CB can be mentioned as follow [4]: prevent to access failure component, handling error quickly and gracefully, prevent the caller stuck on failure services, can custom the fallback action, prevent the services for overwhelmed managing request.

Most of the articles have the same perception of implementing the CB concept: a CB is a bridge between the clients to a target service to prevent accessing the service when it failed. As a bridge, CB can have three states, open, close, and half-open. If it closed means that all the requests are forwarding to the service, if it opened means that all the requests are blocked. Figure 1 shows the basic concepts of the circuit breaker.

The three states above can be described as below:
1. Closed: all the request is forwarded to the service targeted, it kept happens until the system detect the failure in certain threshold (such as timeout counters or sum of failure request), the
The circuit breaker will become open. The threshold can be defined by the system or pre-set by the user.

2. Open: all the request is blocked by the circuit breaker, and circuit breaker or any others component, send a failure message to the requestor or client, or we can call any failure-handle procedure.

3. Half-open: on this condition -that happens during the services failed and circuit breaker in open state- the circuit breaker can be set half-open, which means that the circuit breaker will try to contact the service to check when it becomes responsive. It can be done periodically by pinging the service or after a certain time. When the circuit breaker received the response from a service, the state can be set closed again and all the requests can be forwarded as normal.

Most previous works use the same concepts of how a circuit breaker works, except there was a proposed idea for modifying the states into two states: open and closed, that implement Markov Chain to detect the fault without a timeout, so the timeout can be set dynamically [6].

As a pattern for microservices, the structure of a circuit breaker can be summarized as below, from several sources as [2], [7], and [8]:

```plaintext
[Initialization]
Parameters (timeout, failure_threshold)
Initial_state = closed

[the states]
Case state
State=closed Begin
Forward_message (the services)
If_error error_handling
Set_state= open End

State=open begin
error_handling
Set time_out
Set state=half-open
End

State=half-open
Begin
Call_services(period)
If response=true then
Set state = closed
End
```

From the structure code above, we can identify the fundamental part of the circuit breaker structure as follow:

1. Parameter, for controlling the circuit breaker status and process.
2. Minimum three-block code for execute the action for each state.
3. Error handling or alternative actions can be written inside the circuit breaker or can be put outside as separated components of the circuit breaker.

The taxonomy of the circuit breaker was proposed by [4] but the author refers to the implementation in IoT. The taxonomy can give us an overview of the general pattern of the circuit breaker. It divides into two segments, structural and behavioral.

The parameters that commonly implemented in a circuit breaker are [4] [9]:

1. Timeout
2. Max Concurrent Request
3. Request Volume threshold
4. Sleep Window
5. Error Percent Threshold

The role and position of circuit breaker compare with others component on microservices architecture implementation:
1. CB as a pattern: Most articles, books, and blogs mentioned CB as a pattern. [9] categorized the CB as a reliability pattern. It was put as a running pattern along with service discovery, security, and observability pattern. [10] put it as a generic pattern and then become a decorator. As a pattern, CB structure can be inserted as a block of code when a client calling a service. It can be used for supporting microservices resiliency and prevent failure cascading to other services. The main task of CB is to block the access and send the request to check the availability, but it has no responsibility to repair the failed service. We need to implement a different scenario for handling this.

2. CB as a tactic: CB is also categorized as a tactic that support three case: prevent single dependency, set timeouts, and providing fallbacks [11].

3. CB as a security aspect: the existence of CB can be considered as a security aspect for designing microservices that running on service-mesh architecture, but it can be achieved using the combination of some strategies such as throttling (rate-limiting), load balancing, blue/green deployments, and canary releases [12].

![Circuit Breaker Taxonomy]

**Figure 2.** Circuit Breaker Taxonomy [4]

4. Research Subjects in Circuit Breaker

In this section, we will analyze and find the relationship between the keywords and breakdown the main keywords into a specific issue to get the map of the research subject of CB. This section aims to answer the RQ2: what is the research subject of the circuit breaker?

After collecting and reading 23 articles on CB, we use five keywords for identify the research subjects on CB at MSA. The approach of finding the keyword can be defined through answering the sequences of questions as follows:

1. What is the role of CB in MSA? To answer this question, we need to understand what is CB and how it actually works, it leads us to identify the basic concept of CB, what is it? Are there any specific components? How we use it? Is there any parameter or behaviour we need to know? On this section, we discuss the CB only on two elements: the client that request a service, and the service itself.

2. How to implement it? After knowing the basic concept on how the CB works in two elements above, we then expand the exploration that in the MSA, there is many microservices and clients, and usually client and a service is built separately, where and how we insert the CB? In client or in a service or both? We resume this question into the keywords: strategy.

3. In complex MSA, there is possibilities that we have to manage more that CB. Each CB will have specific features on the service under its monitored or supervised. It leads us into the question how to manage the CB if we need to access different services? Do we need to set one CB for each instance of services, and if we have to set one CB for each service, how to control it? We resume this question into a keyword: management.

4. The implementation of CB on MSA will follow the scope of MSA implementation. Since we know that today, the MSA can be exist on web application, mobile application or IoT, then the CB will exist on the same platform of MSA. We use the keyword: implementation.
5. To answer our last question about the opportunities of enhancing existing implementation, we search about the existing product than contain CB and how the product is use. We define the keyword for this issue: product.

After defining 5 main keywords above, we can describe the scope of research subject in CB as follows: Can we explore the possibilities of enhancing the implementation of CB among existing products, technology or concept, so we can find the new approach for managing CB and implement it using certain strategy. Through five keywords above, we extract the relevant keyword from the papers or articles and mapping them into conceptual model as shown in figure 3.

**Figure 3.** Research Area of Circuit Breaker in Microservices Architecture

### 4.1. Strategy

In this section, we will analyze and find the relationship between the keywords and breakdown the main keywords into a specific issue to get Most of the articles do not describe the specific strategy for implementation. The common strategy for implementing CB is using existing libraries such as Hystrix Netflix, Armeria, Pybreaker, and Polly. By implementing CB as a pattern, the client has the option for calling the service using the CB, by calling the CB library explicitly on the script. The idea for placing the CB as an external component, not just a pattern, called a proxy circuit breaker is explained in [7], [10], and [12]. But since then, there is no real example of how to implement this strategy. The pros and cons about circuit breaker implementation options, according to [7] and [13] can be explained as bellow:

1. **Client-side CB**, when CB is inserted into the client component, so it provides easy and flexible implementation. But the weakness of this approach is the information about services availability only exist for a local client. If the services are accessed by some clients, each client should implement its CB separately.

2. **Server-side CB**, when CB is inserted into the service component, so the status of service can be published and read by the clients. The problem with this strategy, usually the service provider does not allow their service modified by the other.
3. Proxy-side, when CB is deployed separated, so we do not have to modify client or service component. To adopt this strategy, we need an additional module or mechanism to arrange the request and manage the CB. The proxy side can be considered as an alternative strategy to implement CB since it gives us a more flexible way of implementing CB without modifying the client or service. The possibility of implementing proxy side CB can be mapping into a specific issue on another research area as shown in table 3. In the management area, proxy CB allowed us to manage the CB centralized and transparent. The client and the services do not have to be modified and for distributed CB, we can assign CB for each host, method, or both.

| Table 3. Keywords for Searching Criteria |
|------------------------------------------|
| **Management**                           |
| instance control distribution            |
| single multiple centered independent transpa per per per per host method method |
| client x x x x x x x x x x x x x x |
| server x x x x x x x x x x x x x x |
| proxy x x x x x x x x x x x x x x |

4.2. Management
The important issue of implementing CB is how we manage the CB. The complex system of MSA that consists of hundreds of services and with layered services level need the various approach to manage the CB. We have some options for managing the CB such as:
1. Instance: we can create a single instance for CB to manage some instance for one service, or each instance of services need to monitor by one instance of CB
2. Control: we can implement the three options for controlling the CB, which are centrally managed, independently managed, and transparently managed [14]. On centralized CB, the CBs are managed by the specific component that can run in API gateway. All incoming requests from the client should go to this component for checking the service availability. On independent CB, the client should manage the CB itself, or manage by each service. Transparently managed is the condition when CB is managed by an intelligent runtime environment. It is an ideal condition but preparing such an intelligent environment is not easy.
3. Distribution is the issue that considers how CB managed the services. Do we need to assign one CB for each service, one CB for each host or method [15]. Some problems can arise on this condition such as discussed on [16] using Polly library but it has the issue for quorum decision. Armeria, a framework for microservices, provides the option for grouping the circuit breaker using three methods: per host, per method, and per host and per method [17].

4.3. Implementation
The Circuit breaker can be implemented in any area as MSA can be. Some articles show the implementation on IoT [4], Cloud [18], or mobile applications, shows that CB is an important pattern or library that accesses by the client when calling the services.

4.4. Products
Implementation of CB in industries can be as a library such as in Hystrix, or as a component on a complete framework for building microservices-based application.
1. As a library, the CB pattern can be found on Hystrix (Java), Polly (.Net), Resilient4j (Java), Akka (Scala), Rubyist (Go), pybreaker (Python), JRugged, and Javaslang. Apart from those
mentioned earlier, nowadays we can find many alternative libraries for CB by searching database on GitHub [19].

2. CB also usually exist on a framework for building microservices. We can use some popular framework for creating a microservices-based system such as using Spring, Seneca, Armeria, and Molecular, and adding the circuit breaker feature.

5. Future Research Opportunities
In this section, we will answer RQ3: Are there any opportunities for future research in the circuit breaker? As previously mentioned, there is relatively little research on the CB subject compared to other research topics on MSA. In the area of the internal structure of CB, there is a report that proposed to modify the state of open-closed using the Markov chain approach and eliminate half-open state [6]. In [3], the author tries to analyze the behavior of CB, and in [8], there is a formal description of the internal CB structure. The research conduct by [20] and [21] proposed the integrated model for MSA resilience that consist of CB as a component, but the articles do not focus on a specific issue in CB. The idea of future research in CB is building CB as a separate component of services or clients, instead of just attach as a library, is coming from [7] and [10]. It is called as proxy CB. The need for this CB is relevant to the issue of distributed CB ([17], [16], and [14]), and it falls into the area of managing CB. [22] proposed the idea for packaging the CB as well as packaging the microservices and other components in MSA.

The idea for managing CB as a different component and the possibility to manage the CB as managing the services through service discovery, load balancer, and service routing, could become an interesting topic, but rarely addressed in the research subject. To adopt this idea, we need to consider the basic function of CB and set the clear definition, limitation, and task assignment between CB, load balancer, and service discovery.

Existing condition shows that most microservices-based systems inserted the CB directly as a pattern into a client script may lead to the opportunity to create CB more flexible and intelligent. It opens lots of research opportunities in the future. Some of the opportunities such as:
1. Improving the internal side of how the CB work, such as setting the rule of how CB change the state or identify the basic action need to execute for each state.
2. Extending the ability of environment where the CB is implemented, by adding the capability how to control and manage the CB from the external component so it can be set dynamically as an aware component for changing in environment, such as automatically adjust the parameters based on services load or based on the previous request, manage the incoming request by periodically checking the queue on a service, or resolve the problem that could exist on distributed CB issue.

6. Conclusion
CB is an important pattern on keeping the resiliency on MSA. The use of CB on MSA becomes a necessary component on building the MS-based system, along with other resilient patterns such as retry, failback or bulkhead. Despite the fact that CB almost exists on each MSA-based system, the research that addresses the CB is relatively less compared with other issues in MSA implementation. We conduct the literature research for understanding the concept of CB and identifying the most important keywords on CB research. It helps us to provide a conceptual overview of the research subject on CB, and we present the map to show the problem space in CB. Based on the map, we explore each subject and find the possibilities for future research directions. We identify that there is a lack of research in CB
management, especially the need for managing distributed CB, or manage CB as a separated component from client or services.

References
[1] Ananthasubramanian A 2018 Challenges in Implementing Microservices DZone Retrieved on 5 September 2020 from https://dzone.com/articles/challenges-in-implementing-microservices
[2] Fowler M 2014 Circuit Breaker Retrieved on 10 September 2020 from https://martinfowler.com/bliki/CircuitBreaker.html
[3] Mendonca N C, Aderaldo C M, Camara J and Garlan D 2020 Model-based analysis of microservice resiliency patterns Proc. of IEEE Int. Conf. on Software Architecture (ICSA)
[4] Aquino G, Queiroz R, Merrett G and Al-Hashimi B 2019 The circuit breaker pattern targeted to future IoT application Proc. of Service-Oriented Computing: 17th Int. Conf. (ICSOC)
[5] Goyal S 2019 Circuit Breaker in Microservices Architecture Retrieved on 2 September 2020 from https://www.codeprimers.com/circuit-breaker-in-microservice-architecture/
[6] Addeen H H 2019 A Dynamic Fault Tolerance Model for Microservices Architecture thesis (South Dakota: South Dakota State University)
[7] Montesi F and Weber J 2016 Circuit Breakers, Discovery, and API Gateways in Microservices Preprint arXiv:1609.05830
[8] Vergara S, Gonzalez L and Ruggia R 2020 Towards formalizing microservices architectural pattern with event-B Proc. of 2020 Int. Conf. on Software Architecture Companion pp 71–4
[9] Richardson C 2018 Microservices Patterns (South Carolina: Manning)
[10] Montesi F and Weber J 2018 From the decorator pattern to circuit breakers in microservices Proc. of 33rd Annual ACM Symp. on Applied Computing pp 1733–5
[11] Marquez G and Astudillo H 2019 Identifying availability tactics to support security architectural design of microservice-based systems Proc. of 13th European Conf. on Software Architecture 2019 (New York: ACM) pp 123–9
[12] Chandramouli R and Butcher Z 2020 Building Secure Microservices-based Application Using Service-Mesh Architecture (Gaithersburg: NIST, US Department of Commerce)
[13] Chandramouli R 2019 Security Strategies for Microservices-based Application System (Gaithersburg: NIST, US Department of Commerce)
[14] Lin T 2018 Which Strategy You Will Choose to Implement Circuit-Breaker for Microservices Architecture Application, Part 1 Retrieved from https://medium.com/@1000lin/which-strategy-you-will-choose-to-implement-circuit-breaker-for-microservices-architecture-7d690f3c90
[15] Scott C 2018 Designing Resilient System - Part 1 Retrieved on 7 September 2020 from https://engineering.grab.com/designing-resilient-systems-part-1
[16] Reisenberger 2017 What is Distributed Circuit Breaker? Retrieved on 1 October 2020 from https://github.com/App-vNext/Polly/issues/287
[17] Yuichi O 2016 Circuit Breakers for Distributed Services Retrieved on 5 September 2020 from https://engineering.linecorp.com/en/blog/circuit-breakers-for-distributed-services/
[18] Balalaie A, Heydarnoori A and Jamshidi P 2015 Migrating to cloud-native architectures using microservices: an experience report Proc. of European Conf. on Service-Oriented and Cloud Computing, Advanced in Service-Oriented and Cloud Computing pp 201–15
[19] Github 2020 Circuitbreaker Retrieved from https://github.com/topics/circuitbreaker
[20] Heorhiadi V, Rajagopalan S, Jamjoom H, Reiter M K and Sekar S 2016 Gremlin: systematic resilience testing of microservices Proc. of 2016 IEEE 36th Int. Conf. on Distributed Computing Systems pp 57–66
[21] Hoorn A v, Aleti A, Dullmann T F and Pitakrat T 2018 ORCAS: Efficiency resilience benchmarking of microservices architectures Proc. of 29th IEEE Int. Symp. on Software Reliability Eng. (ISSRE 2018) pp 146–7
[22] Montesi F and Thrane D 2017 Packaging microservices Proc. of 17th IFIP Int. Conf. on
Distributed Applications and Interoperable Systems (DAIS) pp 131–7

[23] Nygard M T 2007 Release It! Design and Deploy Production-Ready Software (Pragmatic Programmer) 1st Edition (Pragmatic Bookshelf)

[24] Molchanov H and Zhmaiev A 2018 Circuit breaker in systems based on microservices architecture Advanced Information Systems 2 4 pp 74–7