International Journal of Multidisciplinary Research in Arts, Science & Commerce (IJMRASC) ISSN Online: 2583-018X

Vol. 4(1), March 2024, pp. 118-123

COMPARATIVE STUDY OF STATIC RESILIENCE PATTERNS OVER DYNAMICALLY MODIFIED RESILIENCE PATTERNS

E. Punithavathy[1], Dr. N. Priya[2]

[1]Department of Computer Applications,

Madras Christian College, Chennai, India

[2]Research Department of Computer Science,

Shrimathi Devkunvar Nanalal Bhatt Vaishnav College for Women, Chennai, India

ABSTRACT

Resilience is the most important feature of any cloud application that is hosted in distributed systems. This paper presents an analysis of static circuit breakers over the dynamically modified retry and auto retry circuit breaker using a cloud application, developed using microservice architecture. The execution results prove that the dynamically modified circuit breaker performs in a better way when compared to static circuit breaker and dynamic retry pattern at transient failure cases that leads to improved availability of the microservice application hosted in distributed systems.

INTRODUCTION

Cloud computing has widened its scope due to its usage and benefits obtained from hosting the applications economically and resourcefully. Such cloud applications have increasing demands to satisfy the cloud native applications[1]. To meet these demands, an application architecture has to be focussed upon, how to increase their availability, performance, scalability and other aspects. Nowadays, Microservice, an application architecture has become popular in meeting these demands. Its independent characteristics such as deployment, mixed coding style , loose coupling with other services are the reasons towards the success of this architecture. Such microservice architecture is in a growing phase, with respect to the resilience patterns[2].

The Resilience patterns can be described as the ability of application modules or services to cope with failures. It decides the way a service or module must behave at transient failure cases. These patterns can be applied only to transient failure conditions, since it is the temporary failure which happens due to the specific service down, a network problem etc[3].All these failure cases are recoverable in nature. At

International Journal of Multidisciplinary Research in Arts, Science & Commerce (IJMRASC) ISSN Online: 2583-018X

Vol. 4(1), March 2024, pp. 118-123

the due course of failure, in order to protect the remaining services from getting affected, these patterns come into the rescue.

Whenever a client's request has some problem in reaching the specific service, it might take more than the longer duration as expected This will further lead to the request overload. It may bring the entire application down, if it has not been treated in a timely manner. This condition results in cascading of failures(failure caused by other failures). To overcome all these features, resilience patterns are used.

The patterns that are available under the resilience are circuit breaker, retry, timeout, bulkhead, ratelimiter etc. Out of which the circuit breaker and retry pattern was the widely used one. They are added to the application service properties, through their manual parameters. Based on these values the services are made to react at times of failures. The retry pattern is one of the oldest and widely used patterns in the network[4].

a) Static Circuit Breaker

One of the widely used patterns for safeguarding the application services from transient failures. They perform in three states and are similar to the electronic circuit used in electrical devices[5]. When they encounter any failure, it redirects the clients request to fallback instead of specified service. By redirecting the requests, they make some time for the services to recover, or the network issues to get rectified. It works like a cycle, during closed state- they receive successful requests, open state- waiting time or recovering time , half-open- check for recovered services[5].

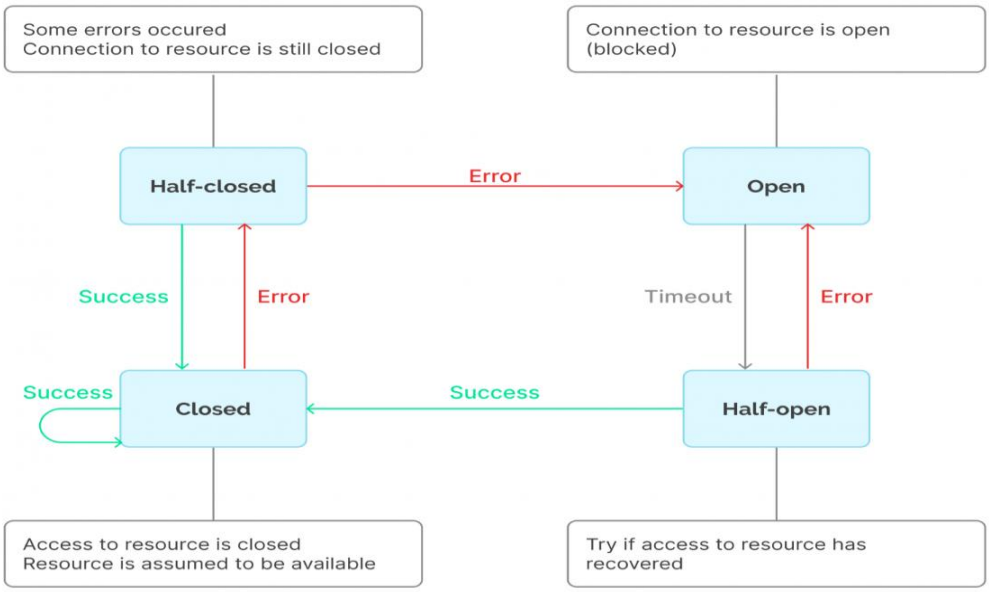

Fig1: Static Circuit Breaker Diagram Process Flow[6]

International Journal of Multidisciplinary Research in Arts, Science & Commerce
(IJMRASC) ISSN Online: 2583-018X

Vol. 4(1), March 2024, pp. 118-123

b) Retry

One of the oldest and simplest used patterns. The main parameters of this pattern are retry_attempt, base_waiting *time, time*out of a request. By default, only static values of developers, given by developers. In dynamic retry, based on the request time value the retry attempts and the base waiting time was determined. After the specified number of attempts, the values are made to reset.[9]

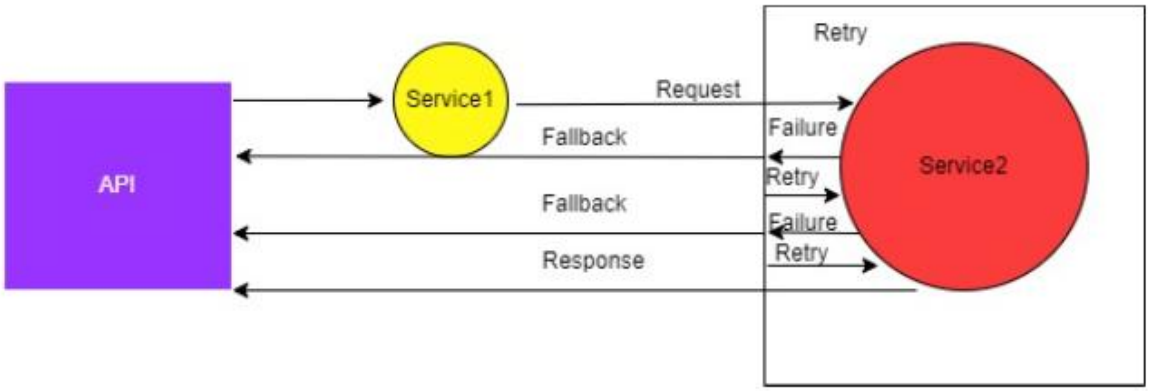

Fig2: Retry pattern Process flow[8]

c) Auto Retry Circuit Breaker

This pattern is the modified version of a static circuit breaker, where the input is determined based on the response time of each request. The main difference between ARCB and static circuit breaker is, the three states such as closed open and half-open are reduced into two states such as open and closed. Similar to dynamic retry the waiting time in open state is determined using the response time obtained from the application programming interface[8].

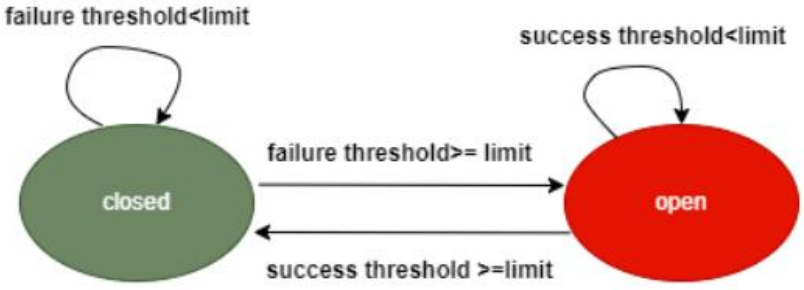

Fig 3. Auto Retry Circuit Breaker.[8]

International Journal of Multidisciplinary Research in Arts, Science & Commerce (IJMRASC) ISSN Online: 2583-018X

Vol. 4(1), March 2024, pp. 118-123

METHODOLOGY

This study focuses on the behavioral patterns of these applications, at transient failure cases or recoverable failure cases. Out of these three patterns, a static circuit breaker is the pattern with manual configurations. The value must be set by the developers, without the knowledge of failures. The other two patterns dynamic retry and auto circuit breaker works well, with response time as the only input.

All these resilience patterns were incorporated in microservice based applications using the IDE intellij idea and Visual studio. The programming languages used for developing the applications are Java, ASP .Net. The Resilience framework and Polly libraries were added to include the basic configuration properties. For generating API requests, the POSTMAN tool is used, apart from passing the get method, it also receives the response time along with the response.

RESULTS

Here the comparative study is done based on the response time obtained from Application Programming Interface named Postman. Responses were recorded based on these static circuit breakers and dynamically modified retry and autoretry circuit breakers.

Table 1: Comparison of request execution at failures

Request	Dynamic Retry[9]	Static Circuit Breaker[7]	Auto Retry Circuit Breaker[8]
10	24	112	55
20	30	109	50
30	25	17	13
40	60	68	10
50	29	39	18
60	39	65	39
70	45	7	35
80	32	112	11
90	65	14	40
100	20	125	11

International Journal of Multidisciplinary Research in Arts, Science & Commerce (IJMRASC) ISSN Online: 2583-018X

Vol. 4(1), March 2024, pp. 118-123

The average time taken by dynamic retry=36.9ms

The average time taken by circuit breaker=66.8ms

The average time taken by auto retry circuit breaker=28.2ms

The Table 1 data proves that when compared with static and dynamic patterns, the dynamically calculated resilience pattern performs well at unexpected cases of failures. Without affecting any services, they make the application perform well by using these dynamic patterns of retry and circuit breaker. The only drawback with the dynamic retry pattern is, its value is made to reset, when they reach the request limit. By comparing all the patterns, it is proved that the auto retry circuit breaker performs well with minimum execution time.

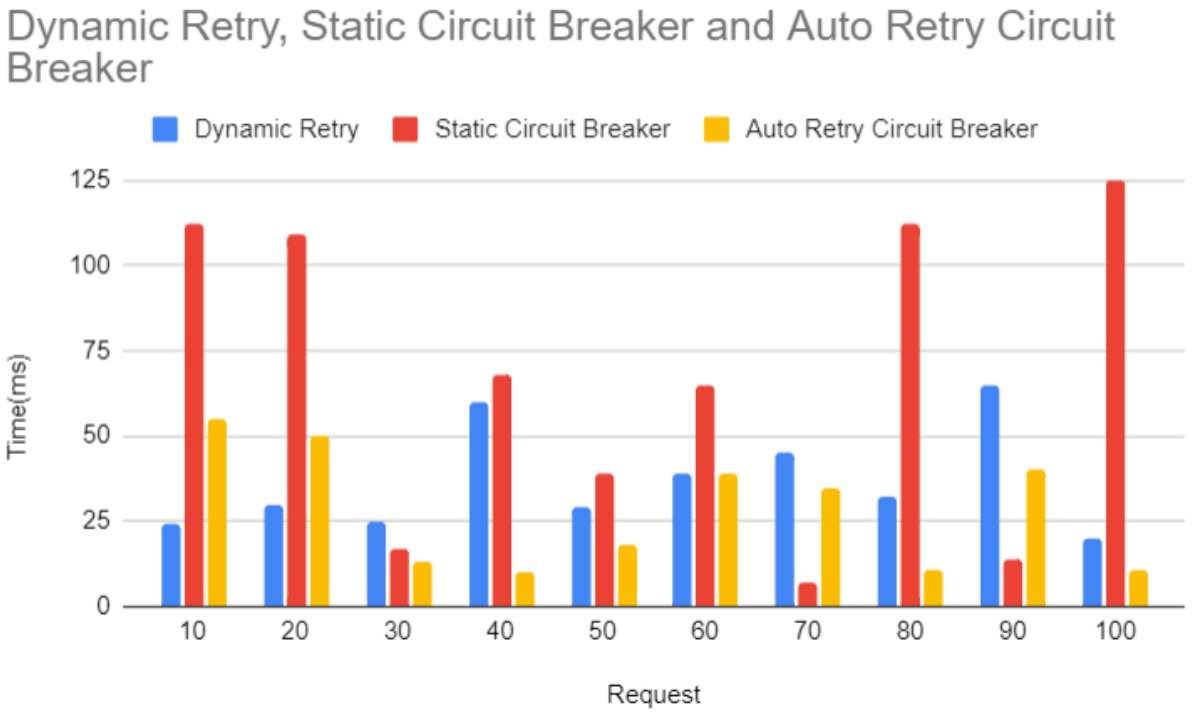

Fig 4: Comparison of execution time of dynamic retry, static circuit breaker and Auto Retry circuit breaker

CONCLUSION

The cloud systems achieve resilience by using these patterns. Standardization has not been done, due to its behavioral changes caused by different types of failures. Moreover, the pattern that is implemented for distributed systems cannot satisfy the needs, if it is static. And hence the dynamically modified patterns , can perform well since based on the failure nature . This factor results in better availability factor of these cloud systems.

REFERENCES

1. Khan, Habib Ullah, Farhad Ali, and Shah Nazir. "Systematic analysis of software development in cloud computing perceptions." *Journal of Software: Evolution and Process* 36.2 (2024): e2485.

2. Cerny, Tomas, et al. "On code analysis opportunities and challenges for enterprise systems and microservices." *IEEE access* 8 (2020): 159449-159470.

3. Silva, Marco António Rodrigues Oliveira. *Improving the resilience of microservices-based applications*. Diss. Universidade do Minho (Portugal), 2021.

4. Rahman, Mohammad Imranur. *Analysis of Microservice Coupling Measures*. MS thesis. 2022.

5. do Nascimento, Helisson Luiz, et al. "Towards a Resilient Spatial Data Infrastructure." (2020).

6. https://commons.wikimedia.org/wiki/File:Circuit_breaker_pattern_state_diagram.svg

7. Punithavathy Ellappan, Priya N," Cloud based Scalable Resiliency Pattern using PRISM", International Journal of Cloud Computing, doi:10.1504/IJCC.2024.10058869(Forth Coming article)

8. E. Punithavathy, N. Priya, " Auto retry Circuit breaker for Performance enhanced microservice applications", International Journal of Electrical and

   Computer Engineering(IJECE),Vol 14, No 2, April 2024, ISSN:2088-8708,

   Doi: 10.11591/ijece.v14i2.pp2274-2281.

9. E.Punithavathy, N.Priya "Performance of dynamic retry over static towards Resilience Nature", Indian Journal of Science and Technology(Forth coming Issue).