Abstract— Service Computing is a interdisciplinary approach that covers the science and technology of bridging the gap between business services and IT services. The underneath breaking technology suite includes Web services and Service-oriented architecture (SOA), Cloud computing, Fog computing, Grid computing, Cyber physical systems, Utility computing, Distributed Computing, Cluster computing, business process modeling and business consulting methodologies. The biggest threat to Service Computing is Design Decay which is generally sensed by Code Smell. Code Smell enable prediction and discovery of antipatterns which are deprived solutions to the various design issues. They are not technically inappropriate, but they might produce non positive results in the long run. In other words an antipattern is something that looks like a good idea, but which backfires badly when applied and leads to higher fault rates. [1]. There is a need for preliminary classification of such false positives with the aim of facilitating a better understanding of the effects of antipatterns and code smells in practice. Threat Landscape is getting more and more cluttered and attackers are getting more and more skillful thereby development and further refinement of such a classification will promote more pragmatic, context-relevant detection and analysis tools for antipatterns. The initial part of paper will provide a basic introduction to cloud design pattern, next to it relationship between design and antipattern is set up and finally set of antipatterns are described that should be considered before migrating the application on the cloud so as to enhance quality, maintainability and extensibility. The research will combine features of Service Computing, Software Engineering, and Cloud Computing.

Keywords—antipatterns, cloud, Patterns, performance, computing

I. INTRODUCTION

Service oriented computing architecture is mostly preferred to design technology applications. There are variants of computing techniques available which includes cloud computing, fog computing, cluster computing and distributed computing. Cloud computing technology is widely used nowadays. It refers to all the Internet-based services, applications and development. Starting from google docs to the development of virtual organization cloud acts as a boon. The three basic deployment models Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1] enable development of high performance computing applications.

The National Institute of Standards and Technology (NIST) defines cloud computing as “a model that enables on demand, universal, useful, on-demand network access to a shared pool of configurable computing resources (e.g. e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [2]. Development of large-scale reference logical and conceptual models is a multifaceted activity which makes use of best practice design patterns in order to provide safe and secure cloud architecture Quality of a cloud based applications is defined either as its compliance or confirmation to a given design, based on functional specifications or requirements. Design patterns used in development of cloud application suffers from challenges of correctness, availability, maintainability, data management, performance and scalability. Software performance is one of the most critical aspect used to judge the quality of design applications. User satisfaction, operational cost indirectly depends on the performance of a software application.

Cloud computing is one of the latest technology that provides higher performance in comparison to on premise architecture. Applications are migrated to the cloud in order to provide enhanced performance and scalability. However, migration decisions are not easily made and can require significant time, resources, and personnel to assess the feasibility and readiness of an organization to make such a move [3]. This assessment considers decisions such as the selection of suitable cloud services and deployment models. If sensitive and business critical information are involved, a careful migration analysis must be conducted to identify the performance risks and benefits of cloud services. Research shows that there are certain software design and implementation mistakes which lead to performance degradation of a software application and are called performance antipatterns [7, 9, 10] [4]. Threats and challenges to performance and scalability leads to decay of design and generation of antipatterns. An antipattern is something that initially sounds like a good concept but later looks like a good idea, but which backfires badly when applied and leads to higher fault rates.
In this paper performance and scalability requirements are examined before migrating to cloud services. Injection of faults into real application has become an interesting field of research. Automatic detection of antipatterns is an important activity that eases the future maintenance and evolution tasks. Moreover, it helps developers to refactor their applications and thus, to improve their quality. While antipatterns are well-known in object-oriented applications, their study in cloud applications is still in their infancy. Despite the wide use of service computing techniques no automated antipattern detection approach has yet been developed. The research focuses on development of techniques for detection and exaction of performance and scalability antipattern. It will enable design of a system which will enable removal of performance and scalability antipattern before migrating the application on cloud.

II. DESIGN CHALLENGES IN CLOUD

Cloud service providers have very well defined following design challenges in cloud.

2.1 Accessibility
Accessibility or Availability defines the amount of time the system performs its normal functioning even if it is affected by bugs, attacked by malicious users or overload conditions. High-availability is the minimum requirement for any cloud application. It embodies the idea of anywhere and anytime access to services, tools and data and is the enabler of visions of a future with companies with no physical offices or of global companies with completely integrated and unified IT systems. Service level agreement describes the availability statistics in it.

2.2 Performance and Scalability
Organizations migrate their applications on the cloud in order to enhance scalability and performance. Performance aspects are major obstacles in cloud adoption and growth. Performance can be defined as responsiveness of a system to complete its execution within the stipulated period. Cloud applications are designed to handle variable load conditions whose prediction is difficult in multitenant architecture. The paper focuses on performance concerns specially in case of scaling in and out to meet the requirement needs. Scalability includes memory storage, no of instances.

2.3 Data Management:
One of the basic component of cloud applications is data management. Different locations, different servers and different servers creates complex situations whereby it’s difficult to host cloud applications. In order to maintain consistency, data must be replicated across all the availability zones.

2.4 Security
Security is the phenomenon of protecting your digital assets and information from unauthorized user attacks. This defense includes detection, prevention and response to threats through the use of security policies, software tools. Cloud providers need to ensure that customer’s data is safely maintained and no unauthorized attack is made so as to maintain confidentiality and integrity of the data.

2.5 Management and Monitoring
Data is replicated in a minimum of two availability zones in order to maintain backup and recovery. Management of replicated data is itself a task. Data monitoring must be done in order to ensure that changes made in one availability zone are replicated to all other zones as well.

III. CLOUD DESIGN PATTERNS

Cloud Design Patterns are a pool of design ideas and solutions which can enable one to find the solution of common systems design problems. A pattern, in short, can be described as depiction or layout for solving a problem. It permits reuse of information or work, attached to the pattern, increasing agility and thereby lessening the errors. A cloud design pattern comprise of name or brief summary of the pattern, explanation of the pattern, issue it aims at solving, implementation, advantage and disadvantages of using it. Microsoft Azure has defined the following design patterns with respect to each and every challenge in the cloud.

![Figure 1: Types of Design Patterns in Cloud](image)
Performance and Scalability patterns
- For faster access cache memory is required that can load data on demand.
- Reading and writing data should be segregated
- All the update actions should be recorded in the data store. Frequently used stores should be indexed.
- The concept of priority queue should be used to segregate higher priority operations.
- Resource management should be done to manage the baseline performance of each component.
- Static and dynamic content should be segregated.
- Separate life cycle policies can be attached to infrequently used data

3.2 Accessibility Design Patterns
High Availability or accessibility refers to a component or system that continuously remains operational for a desirable amount of time.
- Use the concept of buffering to ensure smooth flow of data
- The instances should be used in a controlled manner. Neither over nor underutilization of resources should be done.
- Use functional health point checks at various endpoints to ensure smooth functioning of the system.

3.3 Security Design Patterns
- Use a central key management service that can store the keys in encrypted manner and control key granting access as well.
- Use appropriate algorithms for granting services to the user, majorly for the transactions that store confidential data.
- Use authentication and authorization at each and every level.

3.4 Management and Monitoring Design Patterns
Data is stored in remote data Centre. This makes management and monitoring difficult in comparison to on premise data monitoring.
Following are the management and monitoring design patterns:
- Robust cloud-based applications require features such as circuit breaking, routing, metering and monitoring. It may be difficult or impossible to update legacy applications or existing code libraries to add these features, because the code is no longer maintained or can’t be easily modified by the development team. We need to create helper services that send network requests on behalf of a consumer service or application.
- On addition of a new feature to an existing application add an intermediate layer for isolated monitoring of both the layers.
- Make use of a centralized management and monitoring system

IV. CLOUD ANTIPATTERNS
An antipattern is something that looks like a good idea, but which backfires badly when applied and leads to higher fault rates. Antipattern exist at all level of application design be it developmental, managerial or architectural. The negative consequence of the results generated by employing antipatterns is much higher as compared to the positive consequences of the design patterns.

4.1 Structure of Anti pattern
An antipattern comprise of following components:
- Basic description
- Symptoms on how to recognize the general form.
- Causes that led to the general form.
- Consequences of the general form
- Refactored solution on how to change the antipatterns into healthier situation
4.2 Classification of Antipatterns

Following are the antipatterns for the cloud design patterns:

a) Performance Antipattern

The heterogeneity of the services makes the process of identifying a deployment solution that minimizes costs and guarantees Quality-of-Service (QoS) very complex.

Unbalanced Processing in concurrent processing systems demands extensive processing.

It is not necessary that each and every application when shifted from on premise to cloud will guarantee elasticity and high performance at the maximum level.

b) Accessibility Antipattern

Use of a single database server for multiple application servers can lead to performance degradation and availability failures. Failure of a single database server can lead to collapse of entire network which in turn can degrade the performance of the network and hence availability is dropped to zero.

V. RELEVANCE OF PERFORMANCE ANTIPATTERNS IN CLOUD

Performance antipatterns play an important role before migrating the application on the cloud. Shifting the application on cloud doesn’t necessarily lead to higher performance. It depends on cloud platform, configurability and internals of the application. Antipatterns demands increase in resource utilization, memory and bandwidth. They make application sluggish and resource deprived. This prevents application from utilizing cloud elasticity feature. Empty semi-truck and Circuitous Treasure Hunt antipattern leads to slow network and increased cost whereas Blob antipattern causes processing overhead [1].

VI. PROPOSED METHODOLOGY

6.1 Identification of design patterns that should be considered before migrating the application on cloud

A pattern analyzer is used for design pattern identification which takes as input a source code or a model artifact. Physical and conceptual descriptors of design patterns are defined by performing code inspection. Structural analysis is done for highlighting the basic functions and properties that determine design patterns. Abstract syntax tree is drawn by reading the source code. Intelligence rules using prolog are defined for structural identification of the pattern. Queries are posed based on the threshold defined in the rules. The output of the queries determine the resulting pattern candidates.

6.2 Design a monitoring module for antipattern identification

After identification of the design patterns, a monitoring module is developed which collects run time performance data for each component in the distributed application. Based on that an ontology is built to provide semantic definition to antipatterns.

6.3 Specify the antipattern detection logic

The antipattern detection logic is specified which takes as input the service antipatterns detected in Step 2. The logic is specified using performance metrics.

6.4 Identification of positives and false negatives for computation of precision rates

At times there are few candidate patterns which are mistakenly judged as antipattern. This step involves discrimination between the two.

6.5 Propose a refactored solution

A refactored solution needs to be proposed for the detected antipattern for performance enhancement. It’s the choice of the user to use refactored solution or not in the developed application because the requirements vary from application to application.
Fig 3. Proposed Methodology

References


[12] Reengineering (SANER)” ,Suits, 2016, pp. 609-613.doi: 10.1109/SANER.2016.84

