www.ijecs.in
International Journal Of Engineering And Computer Science
Volume 10 Issue 12 December 2021, Page No. 25501-25515
ISSN: 2319-7242 DOI: 10.18535/ijecs/v10i12.4654

Cloud Native Architecture for Scalable Fintech Applications with Real Time Payments

Kishore Challa
Staff Software Engineer, Bayer/Legacy Monsanto, Chesterfield, MO,
ORCID ID: 0009-0000-6672-8852

Abstract
This tutorial explains how to build a cloud-native architecture for fintech applications using scalable components and frameworks that provide enterprise-grade features. The tutorial focuses on applications that process payments and that expose a service-oriented public API, even if the concepts and components used can be extended to all sorts of fintech applications. provide illustrative step-by-step examples to show how each component can be implemented using open-source technologies. Testing and employment on the cloud are also considered. The reference architecture consists of: a distributed event-driven design pattern to decouple the business process from the storage mechanisms. The Distributed Data Store (DDS) component allows to store and query sensitive data in a performant way, providing out-of-the-box, high-availability; scalability; and enterprise-grade features, such as ACID transactions and a row-level security mechanism; a remarkable, open-source distributed engine that enables to massively parallelize ingestion, visualization, and mesh queries on large datasets, leveraging such a DDS; a highly scalable Cloud-Native API Gateway that exposes GraphQL APIs. It can also serve static content such as documentation; server-side rendering; and login pages, among others, for the UI scheme; a Native-Cloud Serverless Function executor that processes resilient workloads and enables a pay-for-use architecture. It automatically scales workers up and down in a pay-per-execution pricing model; a cloud-native load balancer to distribute workloads between multiple instances of the resilient microservice; a Managed Workflow Engine that orchestrates the execution of long-running business processes in a distributed way, internally relying on a durable DDS. The architecture is implemented at the level of a complete payment service. The focus is on how to build a scalable service adapter capable of performing resilient messaging and processing of payments between an exogenous payment manager and an internal event store.

Keywords: Fintech, distributed systems, cloud architecture, cloud native, payment system, high speed payments, real-time payments, blockchain, high frequency acquirer, bank payments, e-wallet.

1. Introduction

Digital currencies and instant payment systems have drawn growing attention. Stimulated by potential advantages in terms of reduced operational costs, innovation, and efficiency, Central Banks have actively explored the creation of Central Bank Digital Currencies. Other experiments of this nature have matured enough to be deployed in many countries across Europe, China, England, and the USA, spurring regulatory interventions to ensure adequate responses to new events and risks. Innovations in the fintech sector, such as lightning

networks and decentralized finance, have reinforced the concerns of regulators and central banks alike, propelling further studies and modeling approaches. New actors with similar dimensions and features have been scrutinized for their systemic risks and other characteristics corresponding to universal banks, challenging established institutional arrangements and attracting wide attention.

Most current studies focus on specific facets, such as enhanced regulatory models, emergent dimensions of moral hazard, and controllability, and tend to exploit approaches that abstract away technological details. This might be appropriate in some contexts, but there is neglect or lack of consideration on how transaction finality is practically achieved by new actors, despite the fact that such methods are a key point of emergence of systemic connection and risk between decentralized elements on one hand, and established actors on the other. Blockchain-based systems can ensure this finality using decentralized mechanisms, however these require widespread distribution of participants and/or considerable economic incentives. In contrast, existing exchanges with proprietary solutions, and most current CBDC designs, rely on trusted authorities.

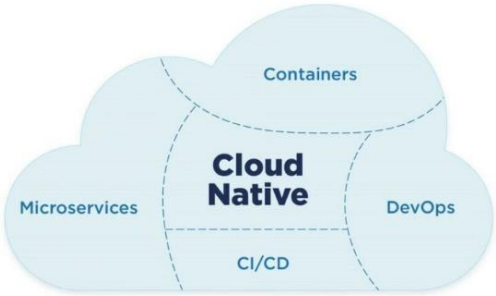

Fig 1: Cloud Native Development

1.1. Background and Significance

The advent of smartphone apps is dramatically changing how we perform P2P payments. This metamorphosis was first seen in China, where a payment platform has been successfully adopted by many users, and then replicated in different forms in the USA. However, unlike China, the USA does not benefit from an established environment in which many people have access to bank accounts. These smartphone payment platforms may be considered as channels for online bill payments, and this approach optimizes the reliability of all previous channel types—but has drawbacks. A fixed transaction limit of $99999 over 5 days is imposed. This channel is assumed to be limited to bank-account holders. The design of the banks-acquiring-network system makes it difficult to introduce domestic money directly into channels used by non-regulated KYC-free apps. Even internationally dominant KYC-free and P2P-only stablecoins cannot fulfill the task yet without the similar approach of pooled reserves of fiat currencies.

This paper introduces an architecture of regulated P2P payment channels for banks and its specific design of API and client types based on existing knowledge and experience common to the banking industry, while observing KYC-free principles and obligations. The key component of this architecture is the establishment of the complete bank-authority technique, including libraries, test cases, and PPP describing the authority and information collection in principle, in which RPC protocols mathematically guaranteed by the independence of DBD must be strictly followed by all participating banks. The design of such a technique creates stringent realities in the current operation metaverse according both to legality and algorithm. The channel itself mimics a text-chat-like application, providing ways to send on-chain payments and off-chain Bitcoin, lending and borrowing assets, messaging and news feed in a manner similar to traditional social media. All aspects mentioned above are analyzed in detail.

Equ 1: Scalability Model

$$N = \left\lceil \frac{T}{t_i} \right\rceil$$

Let:

- T = Total transaction volume per second (TPS)
- t_i = TPS handled per instance
- N = Number of microservice instances required

2. Understanding Cloud Native Architecture

Cloud native applications (CNA) bring forth new patterns such as microservice architectures to decompose existing monolithic applications into smaller units with clear responsibilities. Each microservice is an independently deployable application composed of generally different technologies. CNAs are stateless applications managing volatile data and therefore, rely on other service entities such as databases for persistence. The advent of public cloud enables companies to rapidly provision infrastructure resources to run CNAs. Cloud providers offer many different services for compute and messaging. Therefore, applications are constructed using a mix of services in cloud facilities resulting in complex architectures with many components that communicate over a network.

This evolution of cloud computing led to the emergence of a novel breed of applications termed cloud-native applications (CNA). These applications, on the one hand, represent business opportunities for modern organizations in various industries. CNAs unlock modern business paradigms such as using third party providers to offer moving target services to clients. A burgeoning financial infrastructure relying on payment services from multiple third party providers to accelerate decoupling and modernization of banking and finance. On the other hand, CNAs pose many challenges in observability and monitoring, especially if the CNA is bound by compliance requirements. Cloud-native applications, in its generality, refer to applications built and run on a cloud infrastructure. More formally, cloud-native applications refer to applications built and tested using a collection of cloud services and technologies.

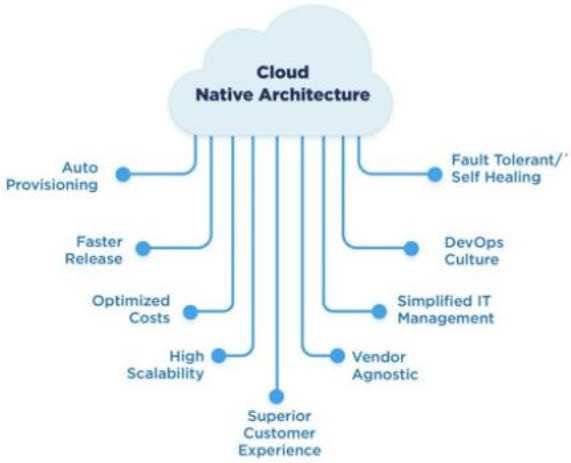

Fig 2: Cloud Native Architecture

2.1. Definition and Principles

Cloud-native develops applications designed and deployed using cloud computing architecture. The cloud-native architecture refers to cloud-based design and infrastructure in which the system architecture is designed to run in environments packaged as microservices. Impossible to deploy on one server, a single service is decomposed into a cluster of multiple microservices to improve scalability and fault tolerance. Each service communicates independently and is equipped with a small amount of memory and CPU resources. It processes requests from its upstream service, independently collects its own operating metrics and logs, and stores the data into its own database or data storage service. Based on the requirements for the scalability of different services in the same application, various private cloud deployments may be selected for data storage and processing. The communication layer consists of components such as Express, Nginx, Redis and the RPC framework. The orchestration layer is built based on technologies to perform the orchestration and tracking tasks on structured streaming jobs through a multi-hop DAG model. An internal scheduling framework provides an interactive service management platform for deploying and running services, data analysis and saving and restoring service states. The cloud-native architecture needs an orchestration framework to manage dynamic changes of the distributed cloud-

native system due to stateful microservices, scaling clouds, intermittent errors, and failure recovery. A DCC-based orchestration framework is presented to manage complex workflows in a cloud-native architecture. The redundancy, statefulness, and checkpointing of services could be modeled as DCC-based services and tasks. A cloud-native service orchestration system provides an abstraction of cloud-native workflows. The system provides a DCC planner that generates a consolidated workflow for a given DCC from a group of DCC-based services. An online workflow manager is used to manage the execution of the composite DCC. A cloud-native orchestration framework decouples components of the DCC planner and workflow manager into microservices, thus being deployable and scalable. With the online workflow manager, a DCC-based workflow could run and reserve resources in cloud-native environments, in which case a newly created service instance will execute the corresponding service task.

2.2. Benefits for Fintech Applications

The financial sector is one of the sectors undergoing a rapid digital transformation, offering people a variety of financial services through online channels instead of traditional banks. Although digitalization of financial services has become a requirement to provide a pleasant customer experience, customers are still skeptical toward FinTech services because of their online nature. Accordingly, preventing fraud in online transactions is one of the challenges that FinTech companies face during their development. A payment fraud occurs when a fraudster makes a payment using the customer's account. For instance, the customer's transaction is rejected by the bank's core banking system. In this case, Suspicious Payment Alerts are created by the bank's analytics engine for fraud detection and prevention. One of the typical rules is to check whether the suspicious payment is over than a previously set threshold. If the payment is flagged, it is required to be first checked by the user. This check could be conducted either by calling the user or sending a check message to the user's mobile application. If the user confirms the payment, it is committed to the bank's core banking system; otherwise, it is rejected. In this case, Reject Payment Alerts are created by the bank's analytics engine which could be visualized in the web-based alert management view. FinTech applications are run on service-oriented architecture on multiple distributed cloud servers. Each service follows the microservice architecture pattern. Service-oriented architecture enables different FinTech services to operate independently. On top of this service-oriented architecture, a cloud-native application is designed and developed. This cloud-native application consists of Kubernetes topics composed of software containers. Cloud-native architecture enables services to scale automatically, recover from failures, and survive the error of a data center. For example, account checks, transaction saving, and alert management are three separate services composed of Docker containers. Services are automatically generated and moved across physical servers based on demands. However, capturing all activities related to payment flows requires orchestrating multiple services. Latencies in such FinTech applications depend on the time that it takes to process a request after it is received to a service. Such latencies are based on multiple conditions such as network latencies, service availability, and distributed deployment.

3. Key Components of Cloud Native Architecture

To support the P2G platform, a cloud native architecture must be designed to satisfy the requirements of scalability, security, reliability, observability, and maintainability. This architecture should include the features and components of a cloud native architecture with proper information flow between microservices, security, observability, compliance, and collaborative services. Cloud Native Architecture of the P2G platform is composed of multiple components categorized into Frontend Application, Cloud Native Deployment, Azure Infrastructure Services, Managed Services, and Third-Party Services. The Frontend Application Microservices are hosted in Azure's App Service

where it provides the Load Balancer and HLS feature. Azure's API Management acts as an API Gateway to control the cloud native application access and traffic policies by formation of underlining Cloud Native Deployment components. Cloud Native Deployment which builds on Kubernetes Cluster and Azure Container Registry hosts the application's core P2G and Blockchain components with observability tools as all managed by Azure Kubernetes Services. Azure's Infrastructure Services consist of Pub-Sub and DB components including Azure Event Hubs, Azure Cosmos DB, Microsoft's Event Store, Redis, and Microsoft SQL where each component contains its clustered instance fulfilling the observability and compliance requirements. P2G Compliance Services consist of services for Business Audit Data Store Management, Latency Measurement, and Zero Balance Reporting. Where the Business Audit Data Store service when configured with compliance requirements, issues notification events upon data selection for easy data extraction. Latency Measurement service collects, indexes, and storage latency metrics for every transaction type while Zero Balance Reporting Service ensures the compliance conditions are satisfied for every day-end closure.

3.1. Microservices

Microservices, a cloud-native architecture style, have experienced a meteoric rise. Major companies have adopted this technology, and the new stack behind their services is now open-scoped, where hundreds of services can be deployed or updated from 100 to 1000 times per day. Microservice architecture has shifted traditional monolithic applications into loosely-coupled, light-weight, and self-contained components. Each microservice unit can be deployed and operated independently for different business objectives and functionalities. A microservice takes the responsibility for a specific domain (bounded context) and implements related capabilities. Each service is implemented as a complete, self-contained application and developed with the most suitable technology stack with regard to the developer's high productivity. Light-weight communication through API endpoints is established for interaction among microservices. An application composed of a set of microservices can be viewed as a microservice system or microservice architecture.

Microservice architecture has significantly improved the efficiency of application development and deployment. The loose-coupling and self-contained design of microservices allows autonomous development by different teams and effortless deployment over cloud. For applications composed of hundreds of microservices, microservices in different business domains can be migrated, rolled-back, and grey-released in an independent way. Due to the popularity of microservices, large-scale microservice systems and platforms have emerged. A considerable share of cloud revenue derives from hosting services for microservice applications. However, the growing scale of microservice platforms and dynamic inter-dependencies between microservices raise new challenges in resource provisioning of cloud infrastructure.

Besides, the inter-dependencies between the microservice units on request processing make it challenging to investigate the relationship between the provisioned resources and quality of service (QoS). Consequently, services under plan violation would be degraded in quality. Microservice-based applications can be much more latency-sensitive compared with a monolithic application. A slight increase in the response time on the system's average can lead to dramatic losses. Co-locating different microservices can reduce costs while also incurring a dramatic change in resource usage due to workload variance and cross-machine communication overhead. The management of large-scale microservice clusters composed of hundreds of microservices is non-trivial. The cloud-native architecture and microservice have been trending for years, and globally leading companies have migrated its services to cloud-native clusters and comprehensively applied microservice-based technologies. For applications of such a scale of a cluster in a production environment, the ordinary

resource provisioning approaches validated in a research environment cannot play as efficiently as there is a gulf of ordinality.

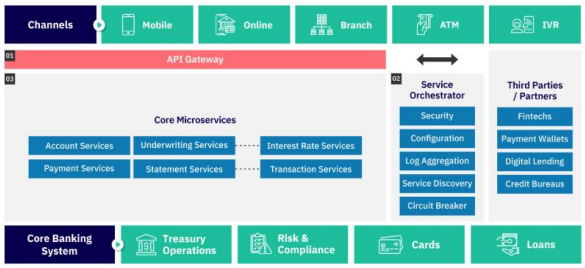

Fig 3: Microservices-based architecture

3.2. Containers and Orchestration

A container is a smaller virtualization unit that provides a way to run multiple isolated applications on the same host. Containers provide a similar level of isolation as VMs. They package a single application with dependencies, enabling it to run on any machine that has the same version of the container engine installed . Containers are easily deployable and much lighter than VMs, and can contain any application or system. They provide an environment that allows running an application on any machine or infrastructure without compatibility issues. They support Docker images that package everything an application requires to run. Images are built based on a directive file called a Dockerfile. The images are composed of layers that map to file system changes associated with commands in the Dockerfile. The container engine runs the application when the image is executed. A container orchestration (CO) platform is a software that provides a layer of abstraction on top of a cluster of hosts running a container runtime. It facilitates managing the lifecycle of containerized applications on clusters. Tasks include provisioning hosts, managing containerized applications, and monitoring status of containers and hosts. Kubernetes is currently the most popular open-source CO platform, with a market share of around 64%. It provides an API to interface with a master that controls all operations in the cluster. A Service is a proxy that allows automatic load balancing of requests to the applications by exposing a port accessible only within the cluster. Secrets are used to provide credential information to containers without requiring hard-coded values in application images. A DaemonSet manages an application running on every node in the cluster, while a Job ensures that one or multiple execution of a task completes. The operator pattern allows users to make Kubernetes aware of the unique features of their applications, providing orchestrator-like functionality to a specific kind of application.

Equ 2: Latency Model for Real-Time Payments

$$L_{\text{total}} = L_{\text{net}} + L_{\text{proc}} + L_{\text{db}} + L_{\text{ext}}$$

- L_{total} = total latency
- L_{net} = network latency
- L_{proc} = processing latency (e.g., fraud check, AML)
- L_{db} = database access latency
- L_{ext} = external API call latency (e.g., to banks or PSPs)

4. Scalability in Fintech Applications

Scaling fintech applications is a core requirement. This chapter discusses scalable approaches and technologies for data ingestion, streaming processing, high-frequency trading and storage, compute microservices, orchestration, CI/CD and infrastructure, result presentation, and storage high-speed data back into storage. Real-time payments are an example use case for these technologies and approaches.

A growing market demand for scalable Fast payment solutions, groundbreaking financial technology and protocols changes along with explosive growth from disruptive fintech alternative schemes, challenging underserved markets with flexible and high-quality Fast payments, to Fintech businesses Fast payment is new competitiveness and new philosophy for traditional financial institutions. Companies require a new infrastructure and different technologies to achieve Fintech competitiveness.

Fintech applications, payment systems, have new challenges. The challenges, like payment transaction throughput and scale up, flexible new schemes adoption, microservice systems architectures, flexible cloud native infrastructures, and data privacy for sensitive information, must be responded to well with fine-tuned approaches or technologies for bundled growth. Requests for flexible new schemes adoption or huge growth come along with application components on workloads surge or storm, and the applications must scale up simultaneously. At the same time, the applications growth, or large market adoption and coverage, need to respond at the other side with compliance and data privacy requirements on sensitive user credentials or sensitive transaction information for many countries involved, banks, currency or other compliance regulations and differences.

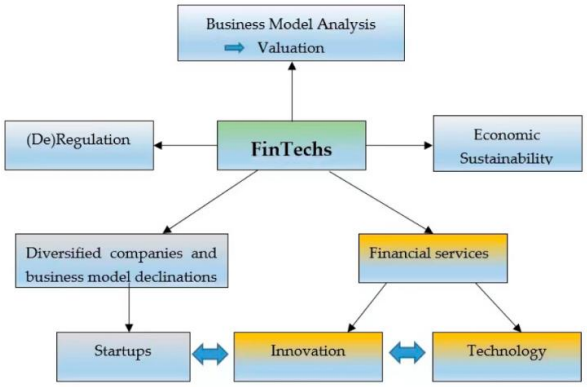

Fig 4: Scalability in Fintech Applications

4.1. Horizontal vs Vertical Scaling

Cloud-native architectures have gained increasing interest among financial market participants seeking to achieve real-time processing capabilities. A widely accepted approach entails developing a data model that allows the definition of atomic transactions over the entire dataset. Writes to a cloud-native database can then be applied using two-phase commit together with a publish-subscribe messaging system to achieve data consistency between microservices. However, this approach prevents the application of horizontal scaling and sharding methods that are essential to cope with increasing processing load. The processing of transactions has to be serialised per shard to adhere to the ACID properties of the database. This can lead to high latencies and is prone to the starvation of slow transactions. All cloud-native systems show non-constant latencies which depend on the internal structure of the transaction batch or contention situations on parsing resources or the database. Wide variations in transaction throughput can be expected even if all backends exhibit constant processing times.

Approaching the problem from the horizontal scaling of transaction-creating machines leads to new representations of a transaction stream that are consumed by a cloud-native architecture. This approach avoids the stalling of transactions, creates guarantee mechanisms for throughput increase, and paves the way for further ideas such as allocating client profiles to equalised load on backends. The aim is to present a competitive solution to all presented stress tests that can be improved as research progresses. While focused on transaction creation, a new representation of the snapshot of the transaction-creating machine can also be created through services on the stream or a facade, so the system also approaches vertical scaling or mini-perks.

Vital identity protection mechanisms against attackers, hacks, and cover-ups of system integrity need to be developed. Follow-up research should also account for dynamic data distribution among backends, its consistency, failure recovery, and post-mortem on observational data. Currently, consumer safety is guaranteed by waiting for the results of the design model so prosecution evidence can be collected in the event of corruption of a stakeholder. Long-term retention of those records is also necessary.

4.2. Load Balancing Techniques

Load balancing is a fundamental part of cloud computing. It can be referred to as the distribution of work between many tools to maximize efficiency across the board. In this case, the tools in question are computers. The work is whatever the computers are meant to do within a cloud network. The efficiency measures can be energy, performance, or

some factor relating to performance or energy, such as parallelism, throughput, or popularity. Load balancing becomes a necessity when the amount a cloud can handle is exceeded either through traffic spikes or continued growth. The premise is simple enough, as it largely amounts to creating additional servers to share the load, but the exigencies of load balancing become complex quickly. There are many decisions to be made when adding capacity, once again, on a high level. Load can be distributed according to the individual performance of the servers, the routing of data, or the service requested. The possible solutions can be sorted into three categories based on the operation of the algorithm: centralized, semi-centralized, and distributed. In such cases, the client must not possess their full original CD image, as this would defeat the object of the service. Instead, the image is distributed across the cloud, with clients having access to only some parts of their image. This allows for services that would normally be impossible with low-spec client devices, such as video-editing software.

5. Real-Time Payment Systems

A real-time payment (RTP) system is a service that enables financial transactions between accounts at different financial institutions (or payment networks) to be settled in real-time. RTP systems may settle interbank payments in batches, but transactions settled by RTP systems are guaranteed to be settled before the end of the transaction's service window. The RTP transactions settle "real-time" or "instant," and thus may allow further financial transactions to occur based on the transaction result with very little latency. RTP systems are mostly expected to process transactions in less than 5 seconds, while some other RTP systems aim for less than 3 seconds. Payment immediate and often instantaneous payment services increase efficiency and speed of payments. As customers increasingly expect immediate crediting of funds when making payments, interest in faster payment systems is gaining momentum globally. Real-time payments are transactions that are credited and debited to/from a payer and payee's respective bank accounts almost instantaneously (typically within seconds), even at off-business hours or weekends.

Real-time payment (RTP) systems have developed rapidly in both technology and regulation. Some banking regulators require or allow their central banks to establish RTP systems as critical financial infrastructures. Increasingly more commercial banks implement RTP systems that provide instant payment services to retail and corporate customers. However, most existing RTP systems are operated by a single entity or a bank-led consortium of trusted banks, limiting their customer reach. Furthermore, a trusted clearing house and correspondent banks should be in place to operate a decentralized, bank-independent RTP system. The Blockchain-based peer-to-peer (P2P) system of programmable payments on the Internet of Value is considered the third generation of payment systems. However, the use of blockchain for financial payments is still an ill-posed problem. Acquiring financial licenses is an arduous endeavor; crypto banks are very poorly regulated and have little incentive to comply with stringent regulations; and, notwithstanding that blockchain is highly secure, crypto assets are vulnerable to phishing attacks, Ponzi schemes, and financial hacks.The advent of the internet and information technology has also driven rapid changes in global payment systems. In addition to the traditional importance of accountability, security, efficiency, and fairness, it is also critical that payment systems comply with all relevant laws and regulations. Complying with regulations is complex and requires dealing with heavy burdens such as transaction limits, sanctions screening, or multi-party guarantees. Therefore a reset of the entire payment systems architecture in an industry is inevitable and could take ages.

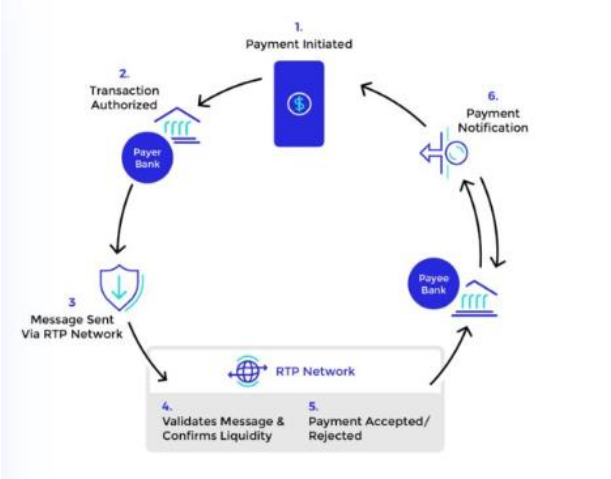

Fig 5: Real-Time Payments

5.1. Overview of Real-Time Payments

Payments systems refer to asset transfer systems, including things such as transfer of ownership of goods between different parties, transfer of rights to money and securities, transfer of account balances in the payment systems. Either end may also act as clients and each party may play a role of both clients and servers. Payment systems have been important components of our economic and financial systems for centuries. The transfer of ownership in any object requires well-defined transfer protocols which are generally referred to as the payment system. A payment system is any system used to settle a financial exchange. It may include physical exchange of cash, payment of checks and debit or credit transactions among banks, and electronic funds transfers. Legal concepts and laws related to payments systems have changed little over the past two centuries.

Payment relates to the transfer of cash, obligations and ownership rights which are usually refined by payment scripts. The term payment systems refers to a combination of both concept and process. Payment systems exist in any financial and economic ecosystem, and broadly refer to a combination of one or more protocols and tools used to facilitate the transfer of financial balances. Financial assets are often represented in the form of digital data, such as computerized account balances. Payment systems may include tools and services for the creation, collection and validation of digital documents representing the right to transfers of ownership of such assets on a common ledger. An intermediate form of object in the Ledger mechanism pays its owners with a promise of a compensation in a certain currency which could be facilitated by local bank accounts. Automated payment systems are widely prevalent in modern societies. Each bank or financial transaction company needs to maintain its own ledger over accounts which are simply the means of identification.

Unlike the widely adopted systems, the proposed off-ledger mechanism does not require the expensive interoperation among banks/companies, and each institution could deploy its own payment systems. Central banks adopt both ledger and off-ledger payment systems together to accommodate large and small organizations as well as both high and low throughput transactions.

5.2. Challenges in Implementation

While there are benefits in migrating to cloud-native solutions, there are several challenges and concerns that companies and organizations have when adopting cloud services, especially for mission-critical applications. In this section, the concerns and challenges of migrating scalable Fintech applications to cloud, hosted and core services (HaaS, IaaS, FaaS and MaaS) are discussed. * Security and Compliance: Financial transactions not only need to have a business logic, but also should comply with accounting standards. Thus, the first concern when implementing cloud services for Fintech applications is security. This includes the IT security controls such as prevention from unauthorized access and abuse of the system, vulnerability detection and prevention efforts, and securing sensitive data against unauthorized users, including data at rest and in transit as well as preventing data loss. The second concern is compliance. There are various regulations and directives that organizations comply with. Depending on where the cloud service is hosted or the companies and organizations operating the services reside, the governing body, regulations, and

requirements changes. Compliance takes time and should be observably demonstrated to third-parties. In compliant applications, audit trails would have to be stored for extended time spans and access would need to be restricted. Furthermore, external auditors would have to be granted access to more extensive parts of the system. * Control and Vendor Lock-in: Decoupling from a cloud vendor is far harder than prescribed in marketing. All Fintech applications have native data models with interrelated data definitions. The potential of losing control over the cloud service is a legitimate concern. Depending on the cloud vendor, application vendors would have to acquire extensive knowledge of the IaaS providers to migrate between cloud vendors. Harder examples are the so-called "vendor lock-in". Organizations using widely used cloud vendors are granted access to multiple services on wide geographic distribution. However, the data processing APIs offered by these vendors differ from each other in ways that are hard to spot at first glance, yet vital in built systems that migrate from one vendor to another.

Equ 3: Cloud Cost vs. Performance Optimization

$$C = c_i \cdot N \cdot h$$

- C = total monthly cost
- c_i = cost per instance per hour
- N = number of instances
- h = hours per month (~720)

6. Integrating Real-Time Payments in Cloud Native Architecture

In recent years, developments in cloud computing have provided new opportunities for creating fintech solutions. However, this cloud transformation and providing always-on services require new architectures and technologies that can support the real-time nature of the services. In this context, the powerful microservices architecture of cloud-native applications can help design fintech services that deal with online and near real-time queries but with a much steeper learning curve compared to traditional approaches. Achieving consistency and availability across distributed microservices also presents its own challenges.

First, cloud-native applications offer the ability to provide a wealth of new services to customers and agile product development lifecycles. The granularity of decomposition gives designers the flexibility to create independently deployable and scalable services for different areas of the applications. This enables rapid deployment of new features and scalability based on customer needs independently in different sections of the applications.

Second, the cloud-native deployments support the rapid development of a comprehensive set of services on top of a relatively simple state-searching service. A small part of comprehensive stateful services is responsible for all functionalities concerning a rich data model. The processing of data is continuum from real-time streams to batch processing on analytic databases. This enables models trained on historical data to be used on incoming streaming data for generating real-time predictions and new trending topics on data querying services. Stateful streaming engines dramatically ease the programming of complex stateful real-time ingestion and enrichment rules and their execution in a distributed setting. All relevant real-time events for a service are gathered by data pipelines and placed in queues. Upon receiving a new event, the related microservices can be invoked in a distributed manner without affecting the performance of the service. This supports parallel processing of batch queries in distributed computation clusters and the use of better-optimized logic and data structures for low-dimensional static queries.

Third, fintech applications pose various challenges to designing relevant applications. Most importantly, the services have to meet strict SLAs and other regulatory requirements. Catering for such high standards requires architectural and topological choices on a huge scale and often also further

allocations of architectural and data engineering efforts. Also, the differences in payment schemes imply that different protocols, internal data, processing logic, and applications need to be engineered. Introducing new features or changes necessitate thorough evaluations of the systems and possible rewrites of service components.

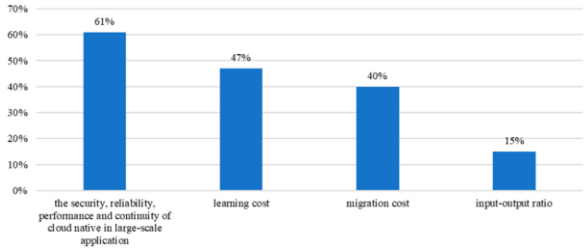

Fig : Cloud-Native under Digital Transformation

6.1. Design Patterns for Payment Processing

Payment processing is a subdomain in fintech, and numerous payment standards have been developed around the world. Payment architectures typically need to be open, extensible, and compatible to facilitate their interaction with various service providers like banks, digital wallets, and payment scheme providers in a way similar to web services. A few design patterns in the field of payment processing for this domain of the cloud native architecture are introduced in this section.

The first design pattern is the strategy pattern Payment, which is applicable for systems where the client needs to interact with backend systems in an interchangeable way. For example, consider a payment system where customers can pay their bills in various ways through internet banking. These payment systems rely on payment service providers to validate the payment and settle the money transfer. The major businesses of such a service provider are, first, to interact with customers to gather necessary payment information, and second, to carry out connectivity with banks or other external parties to initiate the payment.

Given that the payment service provider should avoid vendor lock-in, a standard API and many different implementations in a variety of programming languages for a variety of payment providers have been developed. Patterns of this kind help to improve the design and reusability, and have been used extensively in different areas already. The strategy pattern is a behavioral design pattern that defines a family of algorithms, encapsulates each one of them, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

The motivation for this pattern is how the customer can easily choose the algorithm for money payment without needing to change the client. As they can choose the credit card bill payment or even the BSNL bill payment, one of the payment algorithms would be executed depending on the situation the customer is using. To make this possible, different types of payment methods are implemented in the system, such as payment through a credit card, bill payment through a bank, or through an electronic wallet system. These methods are all implementing the payment interface as shown in the UML diagram. Based on customer needs, they can choose any type of payment to pay bills or money through the internet.

6.2. Data Consistency and Transaction Management

Achieving data consistency in a cloud-native architecture for fintech applications is essential, especially in scenarios involving high-frequency trades, like real-time payments. Such transactions often take very short execution times, around 70 microseconds, making it difficult to achieve 2-Phase Commit as a data consistency solution across any service instance locations, even within the same region. Moreover, the inappropriate placement of service instances may introduce additional latency. Solutions to globally coordinate distributed multi-system transactions exist, yet a proper balance has not been achieved between complexity, performance, and application resilience. Both transaction architectures essentially add more service instances in different locations where interactions across more than one provider occur. With the same amount of cloud resources, this implies reducing the number of service instances in each cloud provider, which can

result in both performance and resilience issues for fintech applications.

A distributed transaction with rigorous consistency guarantees can be implemented using distributed locking protocols or 2PC based on distributed atomic broadcast protocols. However, such solutions are usually expensive in terms of time costs, especially in low-latency applications, because interactions across different locations are prone to be dropped or delayed, leading to blocking states. A disadvantage of distributed locking protocols is that they may starve for long periods as the number of locations increases. Current solutions such as 2PC simply replay locks along with logs after recoveries, and this process can incur unbounded latencies or deadlocks over multiple time windows. Sessions as Virtual Pricing Models are introduced to add reliability to any services with less than Strong Data Consistency across replicated clouds.

The interactions with real-time payments among multiple service providers, especially when the payment systems must reside in other locations, transcend the application's ability to enforce 2PC. As an alternative, an MVCC-like timestamp-based solution is provided, which supports sequential composable transactions with SNAPSHOT consistency across various shades of CC, aiming at simultaneously optimizing performance and resilience. It serves single-session transactions in very short execution times whereby such transactions can be stringently prioritized ahead of others at the session and protocol levels. The implementations of transactions at service providers' backends span a wide range of complexities, and many are actually built upon some form of concurrency control similar to MVCC.

7. Conclusion

The rise of Fintech, payments, and digital banks and wallets is a broad movement accelerating around the globe, fueled by the rapid rise in mobile phone ubiquity, the explosive growth in the internet, and a socio-demographic shift in emerging and frontier markets toward greater workforce participation by younger generations. Payments on the internet have grown into a gigantic architectural market challenge. Global institutions have labeled this emerging trend the Payments and Settlement System, raising the question of what is needed to address it. A reenvisioning is required at all levels, but more compelling than even that is a net new architectural vision on how to handle real-time payments.

Although there are many difficulties involved, the consumer perception of online payments anywhere must drift to being instant with zero fees. This transformation might be likened to the transition of telephone calls from being booked a day in advance and charged by the minute to the current norm of instant connections at zero cost other than the monthly subscription. At some level this is a viscerally impossible challenge. The sky seems to be the limit for those who benefit from the financial infrastructure, latency and fees, and thus, all efforts in this regard will be seen as net futile. However, the advent of unique and inimitable digital currency is a watershed that brings the dream closer to the possible.

The adoption of "Retail" Central Bank Digital Currency (CBDC) will radically bring current traditional banks to their knees if these systems follow the same protocols as assets today. A system with cryptocurrencies features, already pre-built permissions and design that leverage long provable technologies, coupled with policies agreeing on acceptable IDs for participation, compliance, or registration, data ownership, and privacy, is a tall order to build, but within reach. Designing such technological systems as a subclass of automated market makers is a route that will be considered for use cases in different domains ranging from payments to credit risk management and asset management.

7.1. Emerging Technologies

Fintech applications employ various technologies to offer novel solutions to existing problems. Here emerging technologies adopted by leading fintech companies are examined. These have been grouped

into seven classes based upon their commonalities. Each grouping shows clear trends that typify solutions most common amongst the examined group. Network and Communication Technologies broadly facilitate new business models. Payment fintechs often draw upon new communication protocols, and networks to expose existing payment infrastructures to greater competition and improve the efficiency of payment. Transfersoft facilitates electronic payments in remoteing areas by submitting them to the existing telecommunication infrastructure of mobile networks. Dwolla, on the other hand, resolves existing inefficiencies in the US ACH networks by lowering entry barriers for third-party application developers and merchants who wish to draft ACH transactions on users' bank accounts. Additionally, standard on-boarding procedures and a rich API expose the existing payment infrastructure to greater competition. The use of social networks and business networks to enrich a growing list of contacts is prevalent amongst firms engaged in Payment Initiation and Intermediation innovations as well.

Digital Cryptocurrencies constitute another important technology that payment fintechs often seek to exploit. Blockchain-based technologies are popular as these allow for a secure and trustless environment which removes the need for third parties in the payment process. For example, Bitspark illustrates the use of cryptocurrencies to facilitate faster and more efficient remittance services, allowing for lower fees, updateable cryptographic proofs of transactions and fallback mechanisms to resolve issues. P2P bitcoin lending is illustrated by BTCJam and BitLending services, which allow users to borrow and lend bitcoins for arbitrage opportunities and other purposes. In spite of Bitcoin's alleged anonymity and decentralization, concerns over anti-money laundering legislation exist and firms intent on avoiding legislative backlash seek to protect the identity of users while maintaining KYC compliance. One such example is CoinTrove which uses asymmetric cryptography to obfuscate users' identities.

8. References

1. Ganti, V. K. A. T. (2019). Data Engineering Frameworks for Optimizing Community Health Surveillance Systems. Global Journal of Medical Case Reports, 1, 1255.
2. Maguluri, K. K., & Ganti, V. K. A. T. (2019). Predictive Analytics in Biologics: Improving Production Outcomes Using Big Data.
3. Polineni, T. N. S., & Ganti, V. K. A. T. (2019). Revolutionizing Patient Care and Digital Infrastructure: Integrating Cloud Computing and Advanced Data Engineering for Industry Innovation. World, 1, 1252.
4. Chava, K., Chakilam, C., Suura, S. R., & Recharla, M. (2021). Advancing Healthcare Innovation in 2021: Integrating AI, Digital Health Technologies, and Precision Medicine for Improved Patient Outcomes. Global Journal of Medical Case Reports, 1(1), 29–41. Retrieved from https://www.scipublications.com/journal/index.php/gjmcr/article/view/1294
5. Nuka, S. T., Annapareddy, V. N., Koppolu, H. K. R., & Kannan, S. (2021). Advancements in Smart Medical and Industrial Devices: Enhancing Efficiency and Connectivity with High-Speed Telecom Networks. Open Journal of Medical Sciences, 1(1), 55–72. Retrieved from https://www.scipublications.com/journal/index.php/ojms/article/view/1295
6. Adusupalli, B., Singireddy, S., Sriram, H. K., Kaulwar, P. K., & Malempati, M. (2021). Revolutionizing Risk Assessment and Financial Ecosystems with Smart Automation, Secure Digital Solutions, and Advanced Analytical Frameworks. Universal Journal of Finance and Economics, 1(1), 101–122. Retrieved from https://www.scipublications.com/journal/index.php/ujfe/article/view/1297
7. Gadi, A. L., Kannan, S., Nandan, B. P., Komaragiri, V. B., & Singireddy, S. (2021). Advanced Computational Technologies in Vehicle Production, Digital Connectivity, and

Sustainable Transportation: Innovations in Intelligent Systems, Eco-Friendly Manufacturing, and Financial Optimization. Universal Journal of Finance and Economics, 1(1), 87–100. Retrieved from https://www.scipublications.com/journal/index.php/ujfe/article/view/1296

8. Singireddy, J., Dodda, A., Burugulla, J. K. R., Paleti, S., & Challa, K. (2021). Innovative Financial Technologies: Strengthening Compliance, Secure Transactions, and Intelligent Advisory Systems Through AI-Driven Automation and Scalable Data Architectures. Universal Journal of Finance and Economics, 1(1), 123–143. Retrieved from https://www.scipublications.com/journal/index.php/ujfe/article/view/1298
9. Anil Lokesh Gadi. (2021). The Future of Automotive Mobility: Integrating Cloud-Based Connected Services for Sustainable and Autonomous Transportation. International Journal on Recent and Innovation Trends in Computing and Communication, 9(12), 179–187. Retrieved from https://ijritcc.org/index.php/ijritcc/article/view/11557
10. Balaji Adusupalli. (2021). Multi-Agent Advisory Networks: Redefining Insurance Consulting with Collaborative Agentic AI Systems. Journal of International Crisis and Risk Communication Research , 45–67. Retrieved from https://jicrcr.com/index.php/jicrcr/article/view/2969
11. Pallav Kumar Kaulwar. (2021). From Code to Counsel: Deep Learning and Data Engineering Synergy for Intelligent Tax Strategy Generation. Journal of International Crisis and Risk Communication Research , 1–20. Retrieved from https://jicrcr.com/index.php/jicrcr/article/view/2967
12. Somepalli, S., & Siramgari, D. (2020). Unveiling the Power of Granular Data: Enhancing Holistic Analysis in Utility Management. Zenodo. https://doi.org/10.5281/ZENODO.14436211
13. Ganesan, P. (2021). Leveraging NLP and AI for Advanced Chatbot Automation in Mobile and Web Applications. European Journal of Advances in Engineering and Technology, 8(3), 80-83.
14. Somepalli, S. (2019). Navigating the Cloudscape: Tailoring SaaS, IaaS, and PaaS Solutions to Optimize Water, Electricity, and Gas Utility Operations. Zenodo. https://doi.org/10.5281/ZENODO.14933534
15. Ganesan, P. (2021). Cloud Migration Techniques for Enhancing Critical Public Services: Mobile Cloud-Based Big Healthcare Data Processing in Smart Cities. Journal of Scientific and Engineering Research, 8(8), 236-244.
16. Somepalli, S. (2021). Dynamic Pricing and its Impact on the Utility Industry: Adoption and Benefits. Zenodo. https://doi.org/10.5281/ZENODO.14933981
17. Ganesan, P. (2020). Balancing Ethics in AI: Overcoming Bias, Enhancing Transparency, and Ensuring Accountability. North American Journal of Engineering Research, 1(1).
18. Satyaveda Somepalli. (2020). Modernizing Utility Metering Infrastructure: Exploring Cost-Effective Solutions for Enhanced Efficiency. European Journal of Advances in Engineering and Technology. https://doi.org/10.5281/ZENODO.13837482
19. Ganesan, P. (2020). PUBLIC CLOUD IN MULTI-CLOUD STRATEGIES INTEGRATION AND MANAGEMENT.