Gem

GemOS Platform Whitepaper

Contributors: Siva Kannan, Matt Smith

Gem

Introduction

Purpose

The purpose of the document is to provide a high level technical overview of the GemOS platform. It assumes that the reader has a basic understanding of blockchain technology and distributed applications. A list of documents and books is provided under the reference section for someone who wants to have a foundational understanding of blockchain technology.

Scope

The scope of the document is to provide a technical overview of the GemOS platform. It presents an overview of the core components of the GemOS platform which are required for building and deploying distributed applications. This document does not dive into the architectural details of individual components of GemOS.

Reference

Bitcoin whitepaper - https://bitcoin.org/bitcoin.pdf
Ethereum whitepaper - https://github.com/ethereum/wiki/wiki/White-Paper
Mastering Bitcoin - https://github.com/bitcoinbook/bitcoinbook
Understanding the blockchain - https://www.oreilly.com/ideas/understanding-the-blockchain
Blockchain Revolution - http://blockchain-revolution.com/

Gem

Overview

Blockchain technology offers a completely new way to store, manage, and distribute information. Blockchain networks have demonstrated the capacity to eliminate single points of failure and circumvent an entire class of security issues associated with traditional databases and centralized architecture, but they present their own unique challenges for effective deployment, scalability, and maintenance.

The potential advantages in terms of transparency, immutability and security that are characteristic of blockchain solutions have motivated prominent organizations in finance, healthcare, logistics, IoT, and other fields to explore all kinds of new applications with a view to deliver services more profitably and efficiently, but many have discovered that secure and reliable development and deployment of blockchain-based applications is not yet easy. It entails extensive blockchain-specific expertise, infrastructural overhead, complex application development, security and ongoing maintenance. Blockchain protocol implementations like Ethereum or Hyperledger Fabric provide the network protocol, shared persistence, and shared execution environment which make up the blockchain layer, but not the necessary services on top of that layer required to build production-scale distributed applications.

GemOS is a platform for building and deploying distributed applications. It connects your existing systems to blockchain networks, enabling the automation of arbitrary business processes using the data and identities of those existing systems. The platform provides the necessary foundation elements to not only utilize blockchains, but to also power any complex business rules processing required to produce truly valuable automation.

Gem

Architecture

GemOS seeks to present a **scalable**, **flexible**, and **extensible** platform for building, deploying, and managing distributed applications. With these goals in mind, GemOS was designed based on few core principles:

- Use well-defined and well-managed microservices to encapsulate functionality, isolate sensitive data, enable responsive and efficient scaling, and provide a framework for future improvements.
- Speed matters, but so does ease. Favor event-driven architecture to construct fast, responsive systems, but provide support for alternate modes of operation.
- Expose high-level abstractions of the foundational elements of distributed applications in a powerful, well-understood programming environment to simplify development of extensions, adapters, and applications.

In adherence to these principles, GemOS has been constructed as an event-driven microservice-based platform with a clear separation of concerns across different microservices. Currently, the GemOS core consists of 4 critical components simply called Data, Identity, Network and Logic. Individual microservices communicate via events (messages). The GemOS platform provides explicit mechanisms for both real-time and historical event processing. This dual processing capability facilitates complex business rules execution for certain use cases.

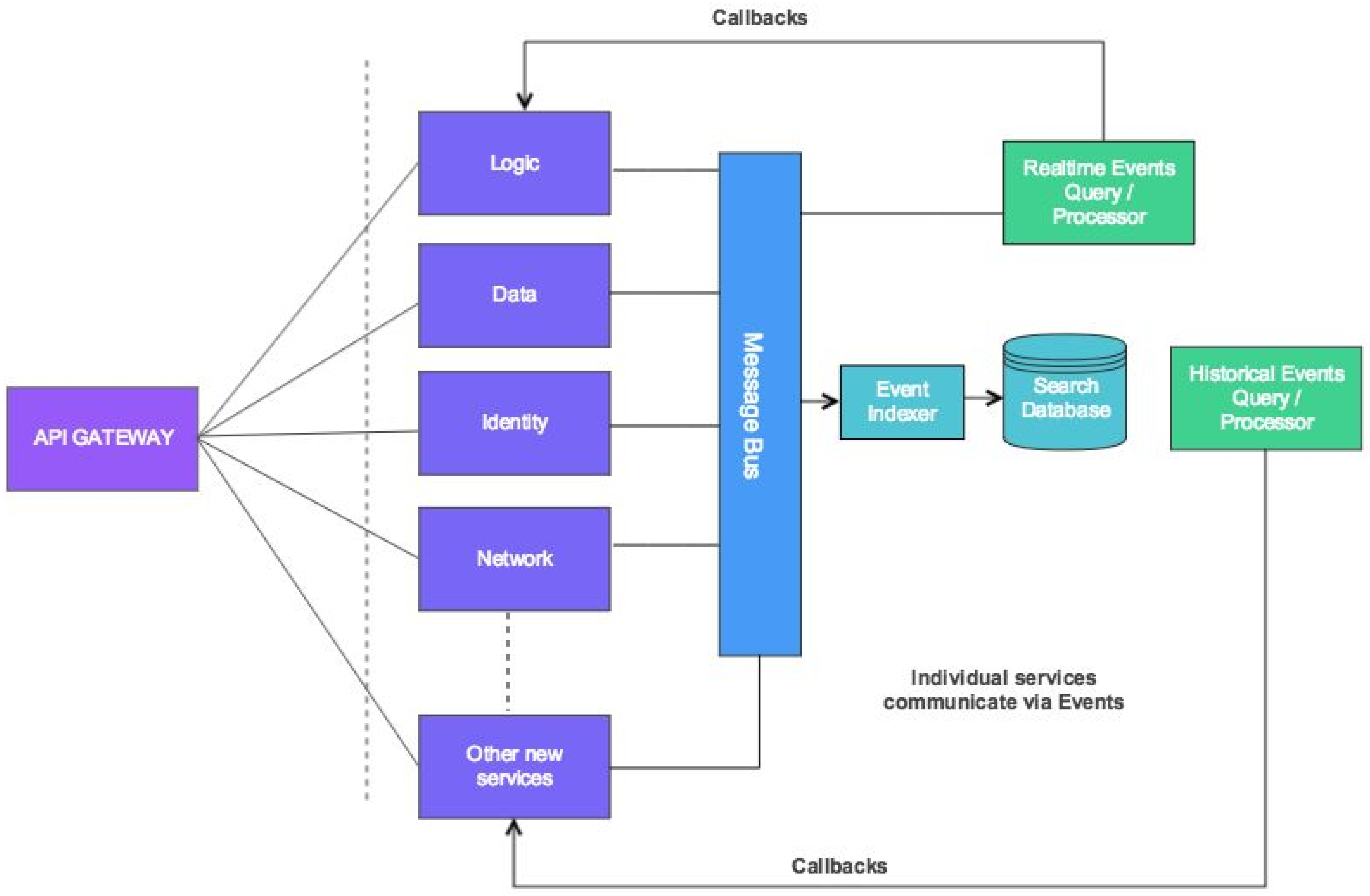
The current microservices in brief:

- **Data** is a data persistence microservice that provides a secure arbitrary data storage service for your application. The blockchain is not an ideal storage medium for most types of data. Data integrates with your existing private datastores to allow database-level integration with existing systems. It supports multiple backend datastore drivers, schema validation, canonical serialization, encryption, and document hashing.
- **Identity** provides secure key management and signing capabilities. It supports multiple formats and protocols and can optionally integrate with your existing identity management solution. This allows for secure management of organizational identities.
- **Network** is a blockchain-agnostic service which provides a generic interface for creating and interacting with arbitrary resources on one or more blockchains. It manages blockchain nodes, processes events from the network, and translates high-level instructions into native on-chain code.
- Finally, **Logic**: the rules engine. Logic provides a familiar programming environment with full access to other GemOS microservices. Logic programs can execute **on-chain** via smart contracts, but can also respond to and control **off-chain** events as

well as execute or respond to any other business functions needed to automate processes. External services can also be accessed via the Logic.

Simplified view of the microservice based architecture is listed below.

Microservices based architecture providing both realtime and historical event processing



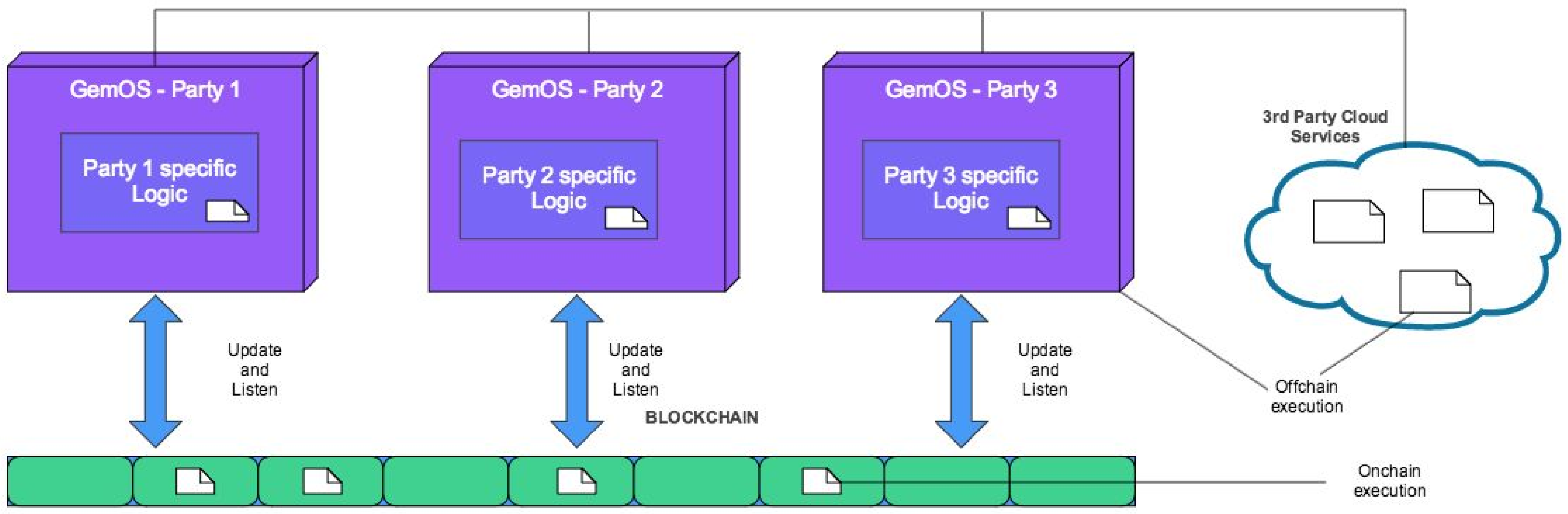
Callbacks
Logic
Data
Identity
Network
Other new services
API GATEWAY
Message Bus
Realtime Events Query / Processor
Event Indexer
Search Database
Historical Events Query / Processor
Individual services communicate via Events
Callbacks

The API gateway acts as an interface between the GemOS and the external, off-chain world. It exposes the public interfaces of the individual services as well as some administrative functions. The microservices communicate with each other over a message bus. Individual messages are indexed and can searched through a search database. Users and microservices register event queries which are either applied to new events in real-time by the real-time event processor, execute on-demand against the historical event database, or are applied on a scheduled basis against batches of events by the historical event processor.. Specific functionality from the microservices can be invoked by callbacks associated with registered queries through the event processors. This is where the event-driven architecture lives up to its promises of extensibility and flexibility: GemOS allows new microservices to be added in the future that can process the same set of events, but provide a new set of functionality.

GemOS Logic and Elements

The Logic service provides the processing capability needed to automate business processes. It has access to events from other microservices both in real-time through a callback mechanism and on demand via the event query service. Logic introduces **elements** which are basically code units offering any type of functionality. They are functions that anyone can compose. Elements execute off-chain in the private execution environment of Logic by default, but can create and interact with on-chain resources through the Network service. They can interact with the external internet and have their own private storage. They can be triggered by an external call to the GemOS API, blockchain events, each other, messages from GemOS services, even messages from e.g. a third party cloud service. These elements form the foundation of automating various kinds of business processes.

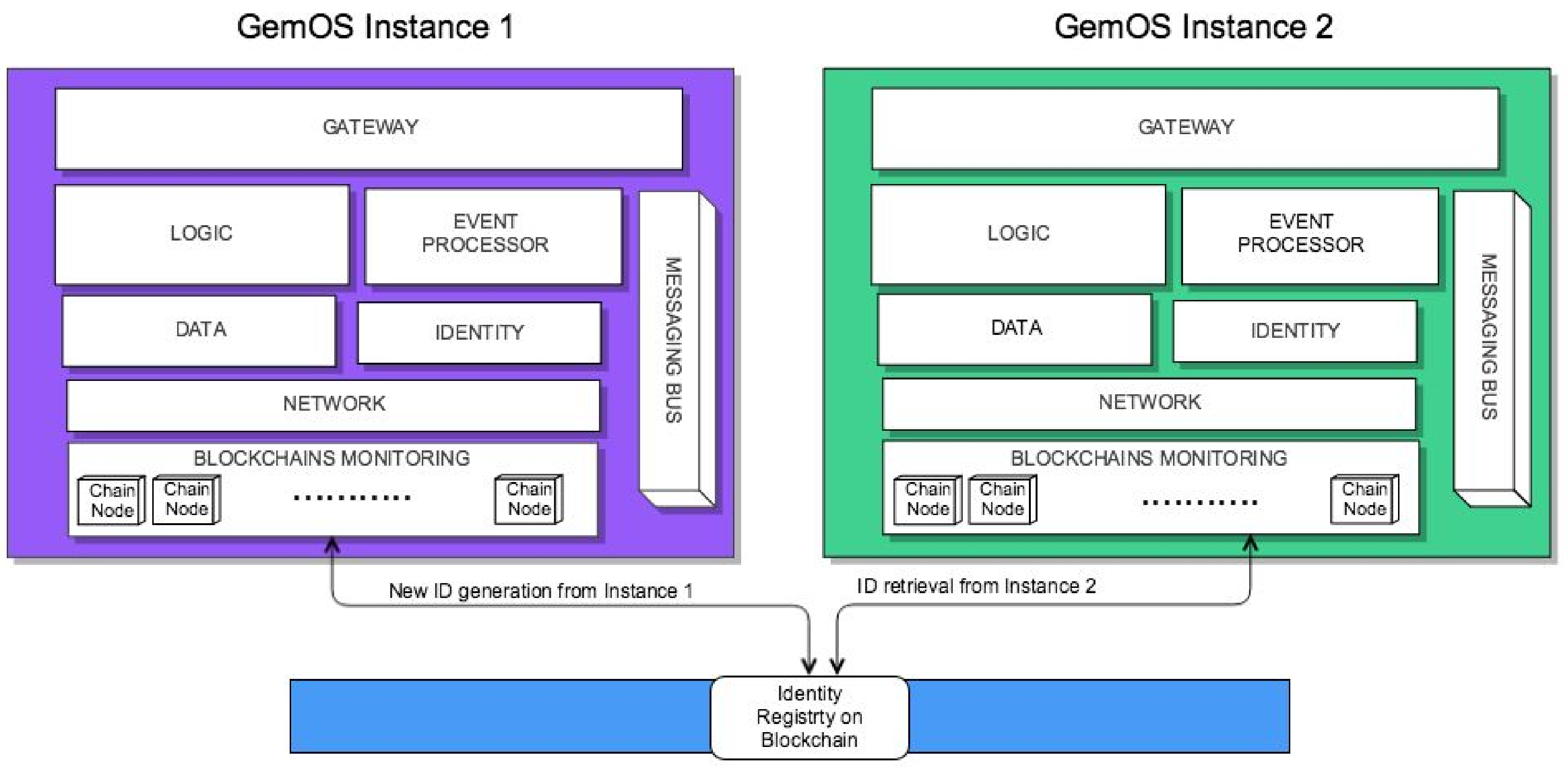
Off-chain and On-chain Execution



With the basic understanding of how an individual GemOS instance functions, let's look at a simple example depicting multiple GemOS instances working together.

Blockchains for Inter-Platform Communication

This is a simple example to show identity sharing across multiple GemOS instances. **Instance 1** creates an identity which it manages via the Identity microservice and stores a record of it in a distributed registry on the blockchain via the Network microservice. **Instance 2** can access the same identity by watching updates to the blockchain. **Instance 1** can also store private data related to the identity via the Data microservice. GemOS also provides the capability to lay down access control to the data associated with the identity. **Instance 2** can request access to the private data stored by **Instance 1**, but it's up to **Instance 1** whether he delivers it.

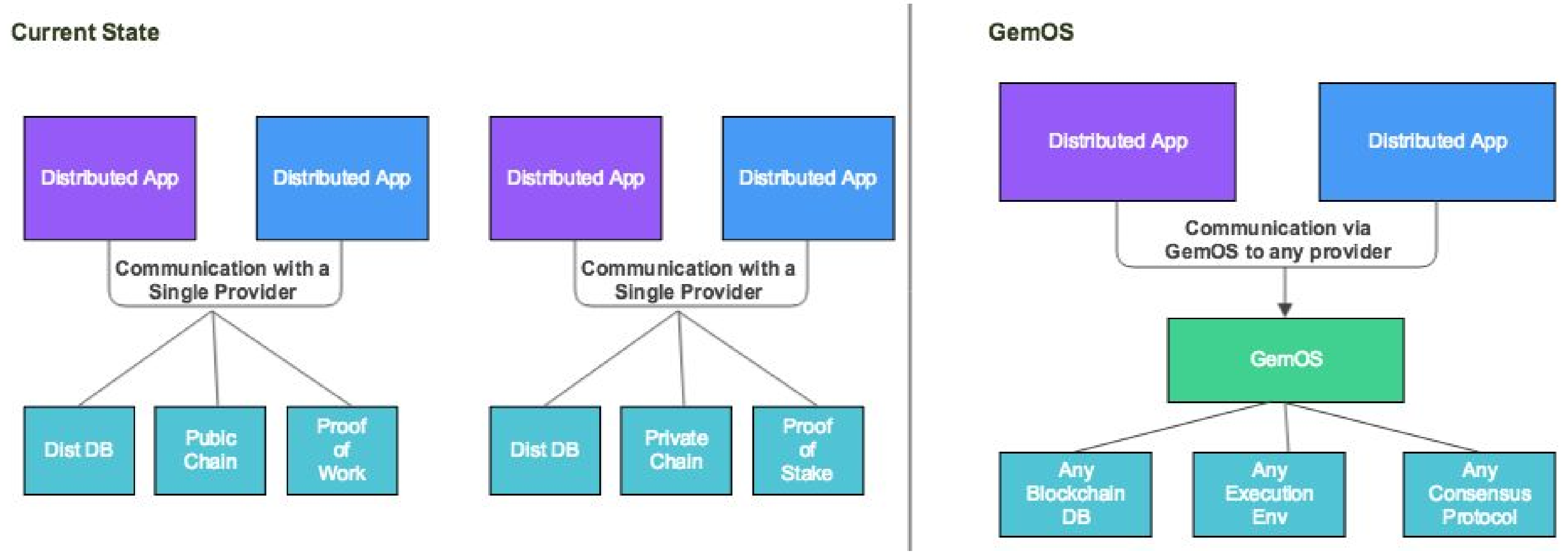


Gem

In addition to GemOS managing the details of participating in a blockchain network, it serves as an abstraction layer on top of the blockchain services.

GemOS as Middleware

Without GemOS, individual clients running on blockchains, would have to rely on a single blockchain provider like Ethereum or Hyperledger Fabric. The modular framework of Gem allows different GemOS clients to pick the best of breed components and build their distributed applications regardless of the details underneath. Portable distributed applications!

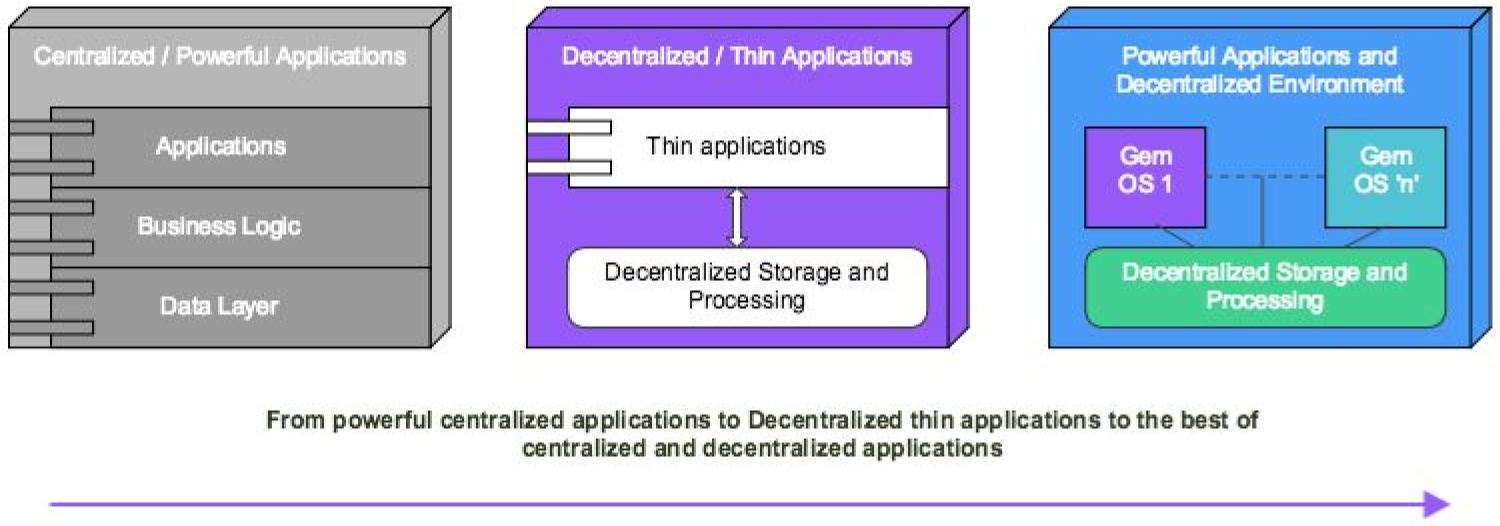


Gem

GemOS offers the secure, immutable, verified operation that blockchains can provide while at the same time delivering the sensitive data storage, identity management, rules processing and intelligence capabilities required to develop powerful distributed applications. To summarize, let us look at the evolution of applications.

Evolution of Applications

In the past, most of the applications have been built around a centralized architecture. Applications managed their own application logic, business logic and data layer – all encompassed in one or few monolithic units. Individual organizations maintained these huge applications at significant cost and risk. For the last few years, there has been a move away from centralized architecture to decentralized architectures where the applications are basically thin clients communicating to decentralized storage and processing services in the cloud. This push from centralized to decentralized architecture shifted the intelligence away from the applications. Application servers have become thin and clients hold greater responsibility for the fate of the system. With GemOS, you maintain the familiar high-level control of application intelligence, characteristic of centralized architecture through highly flexible, composable integration points, while also reaping the benefits of a decentralized storage and processing layer which is accessed via, and managed by, the GemOS microservices.


Centralized / Powerful Applications
Applications
Business Logic
Data Layer
Decentralized / Thin Applications
Thin applications
Decentralized Storage and Processing
Powerful Applications and Decentralized Environment
Gem OS 1
Gem OS 'n'
Decentralized Storage and Processing
From powerful centralized applications to Decentralized thin applications to the best of centralized and decentralized applications

Gem

Conclusion

The evolution of blockchain applications is similar to the evolution of web technologies and cloud infrastructure. As applications moved from enterprise-hosted bare metal servers to the cloud, platforms like AWS, Azure and Bluemix emerged. They were all merely infrastructure providers early in the paradigm shift, whereas now these services have transformed themselves into true platforms as a service. They provide a wide array of services at varying levels of abstraction: virtual and containerized execution environments, plethora storage solutions, identity management, security and business intelligence. Similarly, blockchain networks are infrastructure and you need a platform like GemOS on top of the blockchains to rapidly build useful distributed applications. GemOS manages the blockchain infrastructure and provides these high-level services (identity management, data persistence, cryptography, business intelligence and rules management) on top of it. In addition to these services, it also provides the necessary abstraction layer to generalize applications to utilize any lower-level blockchain services like Ethereum, Hyperledger or Azure Blockchain Service. You can pick and choose Ethereum, Hyperledger, and/or any other blockchain protocol depending on the need, even employing many when appropriate. This modular and extensible blockchain platform enables partners to build distributed applications to solve industry specific use cases.