Deep Science Publishing, 2026
https://doi.org/10.70593/978-93-7185-592-1_3

DeepScience
Open Access Books

Chapter 3: Cloud-Native Architectures for Scalable Data Systems

3.1. Introduction

Various cloud-computing platforms provide a wide range of on-demand services geared to enhance the storage and processing of data. Such platforms allow the agility needed to dynamically allocate computation resources according to the volume of ingested data. Nevertheless, supporting the ever-increasing volume, velocity, and variety of data, as well as the demand for seamless access to both real-time and historical information, remains a major challenge.

In addition to these requirements, cloud service providers offer an abstracted, pay-as-you-go approach that gives data managers the opportunity to select the most suitable solution for each use case across the whole platform ecosystem according to factors such as cost, reliability, latency, and transactional needs. With its elasticity, abstraction, and service-oriented approach, the cloud turns out to be a scalable ecosystem that requires a different design and implementation effort compared to traditional on-premise architectures. The principles of cloud-native design aim to ease the growth of cloud infrastructures, supported by complementary architectural patterns that respond to specific requirements, such as distributed externalized microservices, stream processing and event-driven architectures, data lakehouse and unified analytics, and cloud-native continuous-data-processing pipelines built by orchestration and scheduling.

3.1.1. Background and Significance

The cloud computing paradigm provides both university and industry projects with easy and seemingly unlimited access to a powerful computing and storage infrastructure. However, to enable data-intensive systems and applications to take full advantage of cloud capabilities, their architecture also needs to support scalability. Yet scalable design is rarely the primary focus of major cloud-native data-processing frameworks like

Hadoop, Spark, or Flink, mainly designed for batch rather than real-time data processing and analytics. Emerging cloud-native architectural patterns such as microservices, event streaming, and the data lakehouse enable the design of scalable cloud-native data systems. These trends combine cloud-native principles and services with established distributed computing and storage paradigms. Based on an analysis of the principles and architectural patterns of cloud-native data systems, the principal building blocks for scalable data system design and orchestration in cloud environments are addressed. Cloud-native workflow orchestration systems enable the automated execution of workflows composed of parametrized microservices or distributed data-processing jobs from a deployment repository.

Cloud-native patterns for orchestrating and scheduling at-scale data-processing workflows address recent challenges in orchestration-as-a-service. Specialized data services offered by major cloud providers encompass key data-processing and analytics functions across the cloud-native data-processing workflow, including batch or stream ingestion, storage, processing, and querying, as well as analytics-specific functions such as data exploration, data quality, and enriched search.

Fig 3.1: Cloud-Native Architectures for Scalable Data Systems

3.1.2. Research design

Existing data-storage systems are deployed in the cloud. They were not designed to conform to cloud-native principles, however, which target elastic and scalable

applications. These principles are well established and have been applied to construct scalable, user-oriented services. They can also be extended to Structureless Data Services and Cloud-Data at Scale principles to support cloud-native data services. Data lakes and data warehouses belong in a relpecial pattern, the data-validation pattern, agree with Structureless Data Services principles but pose a storage-disposal challenge.

Progressively relaxing the data-service blueprint indicates a set of data systems that are scalable Qos. Streaming and real-time processing pile together in an Event-Driven Architecture augment878 bulk-batch serve as processing patterns consistent with Cloud-Data at Scale principles. Cloud-native Data Processing addresses how Real-Time Processing and Event-Driven controllers made work at scale.

3.2. Foundations of Cloud-Native Data Systems

Cloud-Native Architectures for Scalable Data Systems: Foundations of Cloud-Native Data Systems. These systems depend on, and must exploit, the principles of cloud-native design—on-demand provisioning, horizontal scaling, and fault tolerance through replication. The cloud represents a fundamental shift in computing and the overall architecture should reflect this change. As businesses and consumers move more of their workload to the cloud, cloud-native systems help ensure that strong consistency, data integrity, and transactional support are maintained while achieving scalability and elasticity.

The way data is modeled, stored, and accessed is critical to designing cloud-native data systems. These environments are designed for, and must take advantage of, the key capabilities provided by a cloud provider. These include on-demand provisioning of resources, elasticity through horizontal scaling, reduced hardware costs by utilizing commodity, and fault tolerance through replication and a large number of servers that can mask component failure. Existing consistency models are greatly weakened, with agreement protocols for strong consistency not designed for such systems. The three properties of distributed systems—consistency, availability, and partition tolerance—remain in force, but the classification of the underlying architecture forces a re-examination of the significance of each.

3.2.1. Principles of Cloud-Native Design

Firstly, cloud-native applications have to scale horizontally. Cloud resources can be provisioned or released on demand, which allows applications to support varying load profiles. The scaling of microservices in response to their actual load requirements is supported by cloud service providers through auto-scaling features. Requirements for

scaling and resource allocation are determined by monitoring components, which drive scaling policies. Scaling by replication increases the number of component instances, while the individual instance size is adjusted for load balancing.

Secondly, cloud-native applications must use multiple data replicas to achieve fault tolerance, availability, and disaster recovery goals. Cloud providers deliver virtual machines in clusters of operating-system images (machine images) that are available in multiple regions and availability zones. Cloud-native applications automatically provision infrastructure resources in multiple availability zones or regions to achieve fault tolerance. Disaster recovery, however, has to be implemented by parallel deployment of cloud-native applications in disaster-recovery sites, rather than by active-active operations that span multiple sites.

Thirdly, cloud-native applications should be designed for failure. Individual components are stateless and maintain no internal state. As a result, the failure of a component can be tolerated by allowing the service to temporarily deliver inconsistent results or throwing an error, under the assumption that clients retry requests. Service-consumers do not cache partly-completed results and ignore returned error codes without user-interaction expectations, because service-providers are expected to be ephemeral.

3.2.2. Data Model and Storage Titting

The data model used in the cloud-native architecture influences how data is physically stored in the underlying storage system. The cloud platform provides a range of storage services with different characteristics, supporting different consistency guarantees, access patterns, performance profiles, and cost models. Each service can be specialized for a particular type of data, optimized for its exploitation, and charged according to the selected data access characteristics. Hence, systems need to adapt their placement and servicing of data according to these characteristics in order to minimize latency, reduce storage cost, and optimize performance. Conventionally, the storage scheme of a system follows the corresponding data model. In cloud-native systems, the opposite holds true: the data model used should express the optimization along the most important axis for the current architectural design. These characteristics should in turn guide the actual placement of data within the cloud storage services.

An extended event log or data-warehouse model is optimal for many complex data-processing systems. The event-log model is the shared-access pattern traditionally used by complex infrastructures such as data pipelines or analytical data warehouses: as data is ingested, it is stored as logs in an append-only manner, and the logical access patterns usually translate into full-read batches that correspondingly read large partitions while incurring little cost.5 This log form usually serves as the raw form of the data: it can be

converted into a tabular format maintained in a traditional data warehouse, into formats optimized for interactive query processing, or it can serve as the starting point for more specialized models optimized for specific-access patterns by data lakes or lakehouses. The tabular model is optimal when two-dimensional analytics (with a well-known schema and low cardinality) are predominant. Libraries equipped for OLAP workloads can use indexes to keep cost low, while the actual update logs can remain with raw and functional levels of churn. Specialized higher-dimensional optimizations can be likewise built for more specialized operational workloads or for manual explorations such as in OLAP use cases.

3.2.3. Consistency, Availability, and Partition Tolerance in the Cloud

Cloud computing services exhibit high availability, so availability must usually be guaranteed. This comes from pushing the idea of resource replication further than in traditional architectures. In distributed systems residing in unreliable environments, like the cloud, it is clear that failures will happen with some probability, and therefore possessing replicas in disjoint infrastructure entities makes sense. Eventual consistency is a form of consistency that guarantees that if no new updates are made to a given resource, eventually all accesses to that resource will return the last updated value. Availability guarantees that a request will always be made to a non-failing node, and hence a response is always returned. Eventual consistency can be tolerated by many applications, or can be fully handled at the application level in such a way that it can be presented to the user as linearizable.

In cloud-native environments, where massively scaled services are distributed across many data centers, addressing the CAP theorem without any given tradeoffs becomes even more desirable. Three main strategies have been adopted. The first is to use a replicated storage system with a simple form of quorum replication and a strong (but tunable) consistency model. The second is to accept eventual consistency for the storage but introduce mechanisms that help manage the inconsistency at a more abstract service layer. The third is to propose a replication protocol targeting a multi-data-center environment without a consistency guarantee, but that also minimizes latency when serving reads. The first option is provided by a few cloud service platforms that expose storage as an online service. The second is exemplified by the architecture of large-scale services at Facebook, among others. The third corresponds to the Spanner system developed at Google.The automation of data pipelines built on top of these services is increasingly handled by dedicated workflow orchestration systems that support the declarative definition, scheduling, execution, and monitoring of workflows composed of parametrized data-processing functions or jobs. In addition, cloud-database services leverage Cloud Provider’s global network to enable a new set of applications requiring

geographical data distribution while significantly reducing the cost of single-datacenter-to-multi-datacenter replication.

3.3. Architectural Patterns for Scalability

Distributed systems can be obtained that layer a set of principles defined through the microservice architectural style and an elastic scaling property over services residing in a cloud environment. Such systems allow an application to grow in a cloud-native manner; that is, not by simply scaling the services of a monolith but by scaling the microservices in a microservice architecture. The demand for scalability becomes important when the throughput demand placed on the application rises beyond the capabilities of a single service instance. Scalability can also become more than just important: a changing business environment can change the ecosystem of an application, in some cases, to the extent that the only viable way forward for the application is to react by scaling the service instances. Some services can exhibit a naturally high change rate of incoming requests, while other services can see a growth of incoming requests either punctually or permanently over time. Depending on the change rate of the incoming requests, one or more instances of services must also be created to reduce the processing delay, which if too long can lead to a bad user experience, and perhaps worse, lost clients.

When increasing the number of instances of a service or a service cluster can be done independently and with low overhead compared to multiplexing requests over a few instances, that service or service cluster is said to exhibit the elastic scaling property. Services can be defined that make use of this property at their particular point of interaction. Microservices scale independently of one another, and are therefore natural candidates for scaling; and incidentally the familiarity and expressiveness of service interaction through the RESTful paradigm affords also an elegant solution, since the storages of a cloud can be exploited to intercept requests at entry points to performance-scalable clusters of replicated microservices. A distinction has arisen between data services and business services: where data services interface with storage diagrams and apply rules to or retrieve data from storages, business services encapsulate business logic and interact with one another to fulfil user requests.

3.3.1. Microservices and Data Services

Data Systems built using cloud-native principles provide scalable and manageable solutions by relying on dedicated service implementations that react to their environment. Microservices and Data Services are regularly combined with event-driven designs using Café and Snab. They can use a message broker like Kafka to

asynchronously publish events in a topic partitioned by object ID, and subscribe to relevant partitions to react to events of interest.

Microservices are a popular architectural style for building applications on the Cloud. Applications are decomposed into a suite of small services which can be developed and deployed independently. They communicate with each other over a network using language-agnostic APIs or "endpoints". Each service encapsulates its own data and implements a single business function. A microservice often contains an HTTP server, triggers or jobs that run background tasks, and a data access layer with persistence in a database. Some APIs expose data to external clients, while others provide internal coordination. Clients might use a set of endpoints synchronously before proceeding, or make calls asynchronously and react when data becomes available.Heartbeat and health-check APIs are commonly included to monitor availability.

Data services are a specialisation of microservices designed to carry out data-specific functions, such as transforming, loading, or replicating large volumes of data. A data service might read a large set of objects from a message queue. It could then transform them according to business logic, update or insert them in a data warehouse, or write them in a storage location. Such services often access external APIs or data contained in other services through internal APIs, and can therefore act as a data flow integration layer coordinating data workflows internally and externally.

Fig 3.2: Microservices and Data Services

3.3.2. Event-Driven Architectures and Streaming

Event-driven design has been adopted in many cloud-native applications in response to increasingly volatile workloads, enabling them to respond to changes as they occur rather than being restricted to processing periodic batches of data. The scaling requirements for cloud-native systems can also be addressed by decoupling the services and allowing the data to be processed by multiple consumers. In this context, streaming frameworks constitute an essential infrastructure for cloud-native applications, as they provide not only a reliable messaging channel between the services, but also data streaming and processing capabilities.

Decoupling the services allows workloads to be elastically scaled by provisioning additional consumer instances for increasingly demanding workloads. Additionally, dedicated services processing data generated by the system—often referred to as data pipelines—allow storing aggregated or pre-transformed data. Such datasets can accelerate the processing of subsequent data analysis jobs, as they contain pre-computed values that would otherwise incur in significant processing costs. The dwell time of the processed datasets can be scheduled so that they match the future demand for the data: they can be deleted when they are no longer necessary, but also retained for a longer period so that future query requests can benefit from incremental costs.

3.3.3. Data Lakehouse and Unified Analytics

A growing number of cloud-native systems are adopting a hybrid approach that brings the best of two worlds: the warehouse and the data lake. Peters et al. coined the term data lakehouse to denote a solution that provides the management capabilities and performance characteristics of a data warehouse while supporting the cost-effective storage of large data volumes in open formats as in a data lake (Peters et al. 2021). By enabling both batch and real-time analytics workloads at scale on the same data and infrastructure, the hybrid approach can eliminate information silos, reduce operational burdens, and improve the quality of business insights.

A lakehouse architecture combines the low-cost storage and scale-out features of a data lake with management and optimization features traditionally associated with data warehouses, such as management of metadata and data reliability (Acid acidity), schema enforcement and evolution, and indexing to accelerate queries and support low-latency analytics workloads. These solutions allow users to create Delta tables, which include additional metadata that informs the system about data changes and is used for versioning.

3.4. Storage Systems in Cloud-Native Environments

Cloud computing enables a new set of distributed storage technologies. The cloud provides virtually unlimited object storage that is specialized for an enormous number of small data files that are accessed and organized by global services. Cloud storage, driven by the needs of web-scale systems, adopts a more relaxed notion of consistency. High-availability data-intensive applications often replicate and partition data across multiple locations. These properties significantly change the operational profile of databases and file systems. New patterns in data management emerge as database-as-a-service offerings leverage cloud storage and minimize operational overhead.

Object stores, such as Amazon S3, are designed primarily for expenditure-minimizing storage of immutable data (for example, files with their change history), but they also provide inexpensive global synchronization services. The low cost of storage encourages the idea of storing everything in an as-is form for long-term archiving. Various systems automatically direct the setting up of an object-store hierarchy for new applications (bellwether applications) and automatically manage versioning (both software and data) based on actual usage patterns.

3.4.1. Object Stores and Immutable Data

Cloud-native storage systems rely on data islands created from a scalable, network-accessible pool of storage. A prominent option for object-based storage is the object store designed for an economic model based on the data's lifetime and access frequency. This approach exploits the Cost of Storage Theory, which argues that the total cost of an online information system is minimized when the stored information is segmented into data islands. It is important that data is read only and seldom updated during its lifetime. These systems are also becoming competitive for more semantically richer data with heavier access.

Object stores are typically specified for semistructured data and provide a distributed virtual filesystem built atop a set of reliable but relatively simple key–value storage engines. They are designed with a very lightweight interface based on HTTP/S and/or REST protocols. Data written to the store is organized as a hierarchy of buckets (namespaces) and objects. Buckets may be associated with different access policies, with a varying granularity, provided both at the bucket and object levels. Several vendors provide cloud storage services accessible via these rest-style APIs. An open-source version called OpenStack Swift has been proposed.

Though semistructured data is the primary target for cloud-based object storage services, it is equally valid to exploit these services for the less-incremental storage of all kinds of data—structured, semistructured, or unstructured. The Cost of Storage Theory strongly

suggests that the capability to leverage the economic model of cloud storage is a strategic weapon for small and medium-enterprise systems providers._accessor

3.4.2. Distributed Filesystems and Block Storage

Two other types of storage systems frequently found in cloud-native environments are distributed filesystems and block storage. Both play an important role in cloud-based application architectures, although they are seldom required by characteristic cloud-native data services. Distributed filesystems provide a shared data source for multiple compute nodes, whereas block storage enables a cloud-native storage strategy for single-instance applications.

Distributed Filesystems. Cloud service providers often offer distributed filesystems as a managed service to cater to workloads requiring a shared data repository of mutable files or objects. These systems are designed to provide durability and availability. Resilience against process or process crashes is guaranteed by the underlying cloud infrastructure or application architecture, ensuring that the loss of a single instance need not affect long-term data durability. Durable write-in-place filesystems are rarely found within a cloud context, since block storage can usually satisfy the same use case more efficiently and with superior performance characteristics.

Block Storage. Block storage is an additional software stack, operating within a higher layer of the cloud infrastructure, that allows data to be stored into the same durable storage subsystem used to store backup files, in a manner more consistent with localized storage of a virtual machine. While any cloud-native application may rely on this service—for example, the runtime block storage service of AWS may be specified as the transaction layer for cloud-native Oracle, SQL Server, or MySQL installations—it is essentially a convenience feature. Latency-sensitive systems operating on a single instance typically benefit from local storage.

3.4.3. Managed Databases and Data Replication

Besides open-source alternatives, most traditional enterprise-grade databases are now made available as managed services by cloud service providers. These solutions extend the serverless paradigm to relational databases, allowing seamless scaling without the burden of infrastructure management. Microsoft Azure SQL Database and Amazon RDS for Oracle provide Microsoft SQL Server and Oracle Database as fully managed services, OAuth compliant with the latest security features; Amazon RDS and Azure Database for PostgreSQL offer managed PostgreSQL database-as-a-service implementations; Amazon Aurora combines advanced database management features

with full serverless capabilities for MySQL and PostgreSQL compatible environments; and Azure Cosmos DB provides a globally distributed multi-model database service with a very flexible schema.

These services simplify the provisioning, administration, and scaling of traditional databases, but organizations with a more liberal attitude towards distributed, NoSQL-style architectures may find cloud-native solutions more appealing. These native implementations remove any concerns deriving from data replication, providing highly available and elastic architectures where frontend services interact with geographically replicated copies for low-latency reads while asynchronously writing to a consensus cluster receiving updates from the entire set of replicas. Data updates can then be processed by dedicated data analytics microservices in real time or in a batch mode with an ELT approach.

3.5. Data Processing and Orchestration at Scale

Processing and orchestrating data at scale is a major capability provided by cloud services. Batch processing is typically realized in the Extract-Load-Transform (ELT) paradigm: raw and clean data is ingested into the storage layer using an inexpensive service, and transformational workloads are executed at a later stage. These transformations are defined in a Dataflow system, which divides a workload into smaller jobs, schedules them, and orchestrates the execution of a data-processing pipeline with the help of a Data Orchestration service. Real-time data processing systems are similarly based on a Lambda Architecture, employing batch processing for periodic updates of the serving layer. New operational workloads are executed as Stream Pipelines, consuming data from one or more message brokers and producing processed or enriched data into an immutably stored layer.

Event-Driven Architecture and Workflow Orchestration further contribute to scaling data processing. A Workflow Orchestrator manages the dependencies between the batch and real-time processes, enabling simple installations, version control of the deployments, and out-of-the-box integration with many external components. A microservices architecture can inherently support a distributed, scalable, testable, and maintainable environment, driven by events and responses: code can be packaged independently in a separate container, and each invocation can independently use the correct version of the functionality on storage for stability.

Cloud-Native Data Systems offer ideal principles for building scalable systems. The distribution of workload is usually managed by Cloud Services: backups and partitions are automatically realized by a Cloud Storage service, and visibility and security are handled by Data Access services, as well as during the definition of sensitive data

environments. Each Cloud Data Service exposes an API service which enables direct access to relevant operations and usually presents either a REST or Thrift interface.

Fig 3.3: Data Processing and Orchestration at Scale

3.5.1. Batch Processing and ELT Paradigms

Batch data processing has been the norm in data management and analytics for several decades and the volume of data processed using this approach continues to increase. A key characteristic of batch processing is that the data is not processed until it has been accumulated, often over long intervals of time. This results in multiple terabytes of data being managed and processed in single jobs. Typical batch-processing jobs run in batch mode and make use of shared data processing clusters. Commercial data processing systems such as Snowflake and Amazon Redshift have matured by leveraging optimizations that are specifically applicable to processing multi-terabyte datasets at scale.

The concepts of batching and scheduling can be extended to cover not only large volumes of inherited structured data but also data coming from other sources such as semi-structured or even unstructured sources. The source data is now kept in a data lake,

the data represented as a data lakehouse, and the whole data processing paradigm is generally referred to as ELT (Extract, Load, Transform). In the ELT paradigm, the data is first extracted from the sources and loaded into the data lakehouse without any transformation. The transformation stage is then implemented by different data processing frameworks that can run at scale over the data in the lakehouse. Commercial data lakehouses such as Databricks are particularly suitable for the ELT paradigm.

Batch data processing continues to be a fundamental capability that is present in all cloud-native data ecosystems. It provides the ability to manage very large volumes of data in shared clusters in a cost-effective manner. However, the increased emphasis on real-time processing has meant that the time-to-insight associated with traditional batch data processing is increasingly important provided new use cases are not enabled.

3.5.2. Real-Time Processing and Streaming Pipelines

More-than-mere consumption of data in motion is also a major pattern in cloud-native data systems: it consists in the continual read-and-process of data at each ingest in order to derive new insights or augment and persist the continually read-and-consumed data stream. As for batch modes, two sub-patterns which can be seen as on opposite extremes of a spectrum can be identified: real-time processing and streaming pipelines. Real-time processing differs from classic data processing by makeing the system more responsive to users' input, but not necessarily performing the processing at a fixed delay or with bounded latency conditions. Streaming pipelines are designed to retain the semantics of a pipeline while providing continual reading and processing, thus often with delay guarantees; streaming pipelines implement the data-processing-adoored pattern in the sense of consuming data in motion as their main goal.

Data streams in real-time processing applications exhibit non-constant data rates because they arise from users' interactions with a service; even motivation is for data from the processing to be reflected back in the service used by users (as opposed to a separate dashboard). On the contrary, streaming pipelines operate on data continually sent from an onion.ishing and fileeng service (for instance). To be useful in real-time applications, it is desirable that the data-processing model offers a fresh-view guarantees, that is, an eventual consistency guarantee in the semantics of providing the most up-to-date processed data.

3.5.3. Workflow Orchestration and Scheduling

Data pipelines are composed of many interconnected tasks and stages, coordinated by a dedicated workflow orchestration layer. This layer takes care of triggering tasks, moving

data between tasks, monitoring execution, and recovering from task failures. Orchestration can be performed at multiple levels, from micro-orchestration at the task level to macro-orchestration at the batch-job or pipeline level, and even up to steering the whole enterprise data pipeline.

Persistent running agent frameworks such as Airflow and Azkaban allow parametrization and scheduling for batch jobs composed of multiple tasks. Such frameworks assign cells for execution and move the data for each task. Since individual tasks are designed as self-contained units, task-level failures are automatically handled by these persistent agents. Triggered execution is more suited for data streams, where a persistent agent monitors the incoming stream and triggers new processing tasks as fresh data arrives. The batch-job-level dependency graph can be built in a similar manner by sharing a directory of incoming data.

Cloud providers offer additional scheduling services for their local resources. With such services, containers are dynamically created and solicited to execute the tasks without prior manual cell assignments. Continuous data processing can also be related to auto-scaling, as more data triggers the creation of more cells to speed up execution. Changing schedules and macro-orchestration at the enterprise data pipeline level is also of interest. Managing the repeated flow of data between the several producing and consuming systems can exploit proven scheduling experiences, and these solutions can be shared among pipeline developers.

3.6. Conclusion

The cloud-native paradigm in software engineering has been adopted across hundreds of enterprise and startup systems, enabling a new generation of applications. The scalability requirements of modern applications often motivate splitting backend services into hundreds of interactive, stateless microservices. Data processing pipelines constitute a new breed of application that scale horizontally and can consume messages at scale. Data-lake-houses enable new use cases for both analysts and data scientists. Multiple patterns and principled choices drive the architecture of these scalable applications. Servicizing, running applications as a set of independent functional units, is a well-known pattern being unconsciously employed for other layers with the emergence of data services. The event-driven architecture pattern is characterized by the decoupling of components through asynchronous message passing. When leveraging the event-driven architecture pattern with a stream-processing framework, applications become capable of processing events in real time with subsecond latency. Another important pattern of cloud-native data systems is the lakehouse, which aims at a unified analytics experience for data engineers, analysts, and data scientists.

Emerging trends driven largely by the cloud-capable vendors include simplicity, interoperability, managed services, and pay-as-you-go. Products are becoming simpler and easier to use, with a focus on their main functionality and removing distracted features. Cloud-capable vendors provide wide-ranging services that go beyond their core products but also tie users to their ecosystem. Supervised machine-learning products, such as computer vision and speech to text, are appearing as managed services, allowing users to leverage the power of cutting-edge technology without requiring ML experts across the organization. Lastly, a pay-as-you-go consumption model lowers the barrier of entry and reduces the need for upfront investment.

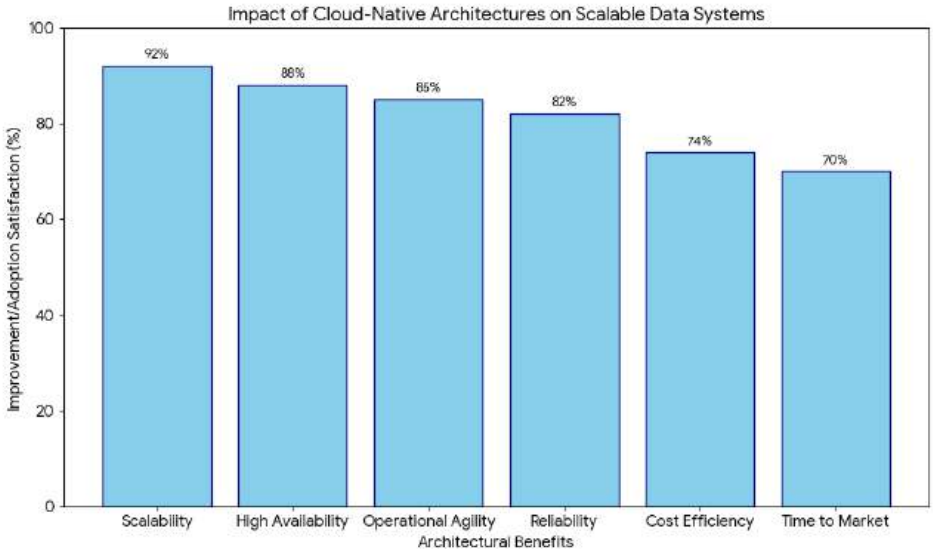

Fig 3.4: Cloud-Native Architectures for Scalable Data Systems

3.6.1. Emerging Trends

New data technologies and services have emerged to increase the effectiveness of cloud-native architecture and address previously raised concerns through deeper specialization and optimization. Specialized data systems have further advantages due to lower costs. Data services are now natively supported by cloud providers as managed cloud services, enabling seamless access by other components within the same cloud provider. Cost-based algorithms analyze query workloads beforehand and select the data service that can answer a specific query more efficiently. For this purpose, Multi-Cloud Runtimes orchestrate heterogeneous cloud-native data services across different providers in a seamless and transparent manner. Similarly, Multi-Cloud Data Management solutions offer to simplify the management of different cloud-native data systems across different providers.

Service abstractions hide low-level details of hardware resource deployment and management. They provide a cloud-user-centric view of the underlying resources. Data-at-rest, data-in-motion, Data-as-API and Database-as-a-Service (DBaaS) are shadowed by Data-in-storage. Data-lakehouse architectures unify the storage of all enterprise data by directly linking to data in external data lakes and data placed into immutable data stores. ELT shifts processing to cloud-smart Data.Frame formats for cost-effectiveness. Serverless frameworks ease the paradigm shift towards massively-parallel data processing, and hide the complexities of providing a wide variety of data-processing functionalities in a data analytics pipeline.

Data-intensive systems are now built using cloud-native architecture to meet global-scale processing challenges and unpredictable scaling requirements. Cloud-native Data systems are based on the ELT paradigm for better Data-at-rest service integration. Data-at-rest services are integrated through naive Data-in-storage runtime transparency, and natively supported as managed services by cloud providers. Data-at-rest Data-in-motion infrastructure and corresponding tools are maturing and finding adoption.. Cloud-native systems achieved their intended benefits with scaling and supporting eclipsing workloads, while incurring larger operational costs due to user-induced multi-cloud adoption..

References

Deng, S., Zhao, H., Huang, B., Zhang, C., Chen, F., Deng, Y., Yin, J., Dustdar, S., & Zomaya, A. Y. (2023). Cloud-native computing: A survey from the perspective of services. arXiv. https://doi.org/10.48550/arxiv.2306.14402

Reddy Segireddy, A. (2024). Federated Cloud Approaches for Multi-Regional Payment Messaging Systems. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(2), 442–450. https://doi.org/10.61841/turcomat.v15i2.15464

Dong, H., Zhang, C., Li, G., & Zhang, H. (2024). Cloud-native databases: A survey. IEEE Transactions on Knowledge and Data Engineering, 36(12), 7772-7791. https://doi.org/10.1109/tkde.2024.3397508

Dritsas, E., & Trigka, M. (2025). Database systems in the big data era: Architectures, performance, and open challenges. IEEE Access. https://doi.org/10.1109/ACCESS.2025.11008610

Babaiah, Ch., Dobriyal, N., Shamila, M., Aitha, A. R., Patel, S. P., & Upodhyay, D. (2025). Intelligent Fault Detection and Recovery in Wireless Sensor Networks Using AI. In 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1–6). IEEE. 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG). https://doi.org/10.1109/ictbig68706.2025.11323980

Kratzke, N. (2018). A brief history of cloud application architectures. Applied Sciences, 8(8), 1368. https://doi.org/10.3390/app8081368

Vajpayee, A., Khan, S., Gottimukkala, V. R. R., Sharma, D., & Seshasai, S. J. (2025). Digital Financial Literacy 4.0: Consumer Readiness for AI-Driven Fintech and Blockchain Ecosystems. International Insurance Law Review, 33(S5), 963-973.

Xu, Q., Yang, C., & Zhou, A. (2024). Native distributed databases: Problems, challenges and opportunities. Proceedings of the VLDB Endowment, 17(12), 4217-4220. https://doi.org/10.14778/3685800.3685839

Nagabhyru, K. C., Rani, M., Reddy, D. S., & Krishnaraj, V. (2025, August). Machine Learning-Driven Fault Detection in Electric Vehicles via Hybrid Reinforcement Learning Model. In 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) (pp. 1-6). IEEE.

Alharthi, S., et al. (2024). Auto-scaling techniques in cloud computing: Issues and research directions. Sensors, 24(17), 5551. https://doi.org/10.3390/s24175551

Guntupalli, R. (2025). Federated Deep Learning for Predictive Healthcare: A Privacy-Preserving AI Framework on Cloud-Native Infrastructure. Vascular and Endovascular Review, 8(16s), 200-210.

Henning, S., & Hasselbring, W. (2022). A configurable method for benchmarking scalability of cloud-native applications. Empirical Software Engineering, 27(6), 143. Inalahttps://doi.org/10.1007/s10664-022-10162-1

Pareyani, S., Goswami, S., Geetha, Y., Dimri, S. K., Niharika, D. S., & Amistapuram, K. (2025). Smart Resource Allocation in Wireless Sensor Networks Through AI Techniques. In 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1–6). IEEE. 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG). https://doi.org/10.1109/ictbig68706.2025.11323661

Heyman, T., Preuveneers, D., & Joosen, W. (2014). Scalar: Systematic scalability analysis with the universal scalability law. 2014 International Conference on Future Internet of Things and Cloud, 497-504. https://doi.org/10.1109/FiCloud.2014.88

Kolla, S. H. (2024). RETRIEVAL-AUGMENTED GENERATION WITH SMALL LLMS FOR KNOWLEDGE-DRIVEN DECISION AUTOMATION IN ENTERPRISE SERVICE PLATFORMS. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(3), 476–486. https://doi.org/10.61841/turcomat.v15i3.15497

Lungu, S., & Nyirenda, M. (2024). Current trends in the management of distributed transactions in micro-services architectures: A systematic literature review. Open Journal of Applied Sciences, 14(9), 2519-2543. https://doi.org/10.4236/ojapps.2024.149167

Quattrocchi, G. (2020). Cocos: A scalable architecture for containerized heterogeneous systems. 2020 IEEE International Conference on Software Architecture (ICSA), 103-113. https://doi.org/10.1109/ICSA47634.2020.00018

Danghi, P. S., Maniraj, K., Jain, P., Adilakshmi, K., Garapati, R. S., & Jain, S. K. (2025). Artificial Intelligence Based Energy Optimization Framework for Wireless Sensor Networks. In 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG) (pp. 1–6). IEEE. 2025 IEEE 5th International Conference on ICT in Business Industry & Government (ICTBIG). https://doi.org/10.1109/ictbig68706.2025.11323860

Chockalingam, N., Deshpande, A., Butra, L., Bodala, R. S., Saksena, N., Parthasarathy, A., Pothineni, B., & Agarwal, A. K. (2025). Scalable cloud-native architectures for intelligent

PMU data processing. International Journal of Engineering Research & Technology (IJERT), 14(12). https://doi.org/10.48550/arXiv.2512.22231

Rongali, S. K., & Varri, D. B. S. (2025). AI in health care threat detection. World Journal of Advanced Research and Reviews, 25(3), 1784-1789.

Hyrynsalmi, S. M., Koskinen, K. M., Rossi, M., & Smolander, K. (2024). Navigating cloud-based integrations: Challenges and decision factors in cloud-based integration platform selection. IEEE Access, 12, 113826-113841. https://doi.org/10.1109/access.2024.3443750

Vadisetty, R., Polamarasetti, A., Goyal, M. K., Rongali, S. K., kumar Prajapati, S., & Butani, J. B. (2025, May). Generative AI for Creating Immersive Learning Environments: Virtual Reality and Beyond. In 2025 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) (pp. 1-5). IEEE.

Li, D., Yang, Z., Yu, S., Duan, M., & Yang, S. (2024). A micro-segmentation method based on VLAN-VxLAN mapping technology. Future Internet, 16(9), 320. https://doi.org/10.3390/fi16090320

Parasaram, V. K. B. (202Reddy Segireddy, A. (2024). Federated Cloud Approaches for Multi-Regional Payment Messaging Systems. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 15(2), 442–450. https://doi.org/10.61841/turcomat.v15i2.154642). Converging intelligence: A comprehensive review of AI and machine learning integration across cloud-native architectures. International Journal of Research & Technology, 10(2), 29-34.

Puthraya, K., Gupta, R., & Dsouza, B. (2025). The role of cloud-native architectures in accelerating machine learning workflows through data engineering innovations. Sarcouncil Journal of Applied Sciences, 5(2). https://doi.org/10.5281/zenodo.15106432

Uday Surendra Yandamuri. (2023). An Intelligent Analytics Framework Combining Big Data and Machine Learning for Business Forecasting. International Journal Of Finance, 36(6), 682-706. https://doi.org/10.5281/zenodo.18095256

Xiong, J., & Chen, H. (2020). Challenges for building a cloud native scalable and trustable multi-tenant AIoT platform. Proceedings of the 39th International Conference on Computer-Aided Design, 1-8.

Kumar, K. M., Parasar, A., Walia, A., Inala, R., & Thulasimani, T. (2025, August). Enhancing Risk Management Strategies in Financial Institutions Using CNN and Support Vector Regression. In 2025 5th Asian Conference on Innovation in Technology (ASIANCON) (pp. 1-6). IEEE.

Abdul-Azeez, T. I., Nwabekee, U. S., Agu, E. E., & Ijomah, C. N. (2025). A conceptual model for scalable and fault-tolerant cloud-native architectures supporting critical real-time analytics in emergency response systems. World Scientific News.