NetApp Instaclustr

White Paper

An architect's guide

FOR SELECTING SCALABLE, DATA-LAYER TECHNOLOGIES

Table of contents

OVERVIEW

As a solution architect, one of our most critical early-stage tasks is selecting the right foundational data-layer components to build applications. Key factors you need to consider for the long-term viability of your applications include Scalability, Availability, Security, and Performance.

Architects and CTOs must be especially considerate in their evaluation of how a specific technology solution delivers value, and careful of any added complexity it may bring. Matching the right tool for the job from the outset is a huge advantage that could help avoid headaches (and costs) down the line.

In this white paper, we will explore 4 open source technologies that all reside at the data layer:

- Apache Cassandra®
- Apache Kafka®
- Apache Spark™
- OpenSearch®

We will detail the challenges faced when making these technology choices and how to select the best technology for your application architecture, including:

- When to choose these technologies
- When to avoid them
- How and why you would use these technologies together

6 Key challenges WHEN MAKING TECHNOLOGY CHOICES

1. Hiring

When implementing a new technology, decision-makers generally have two choices: hire engineers with specific experience (which can be hard to come by for the newest solutions) or to build upon the skillset of their established team (which can take considerable time to gain the required knowledge); organizations often lean towards the latter option.

However, choosing technologies with large communities and broad exposure among engineers can drastically reduce these hiring challenges because the knowledge and expertise are readily available.

2. Quick response to business requirements and customer needs

Technology choice, along with how a solution is built and how a team services it, has an outsized influence on how quickly an organization can respond to the requirements and demands. The right design and implementation of technology will allow teams to act with agility.

3. Managing technical debt

Technical debt impacts both employee morale and an organization's ability to respond to product requirements, thus becoming an important component to address when selecting scalable technologies. However, it's possible to consolidate technical debt as solutions mature, like moving from a homegrown container scheduler to a solution such as Kubernetes.

4. Security and compliance

Technology choice has a profound impact on security and compliance, as do business and customer requirements. For example, there are certain regulations that must be taken into consideration such as GDPR, SOC 2, ISO and PCI-compliance depending on the technology in question and the requirements of the application. Additionally, the capability to execute CVE evaluations, 24x7 alerting and incident response are security considerations that need to be taken into account when choosing technology. Other security and compliance capabilities may include comprehensive network security features, including encryption, access controls and VPC peering.

5. Vendor management

Choosing open source implementations can prevent vendor lock-in and provide flexibility in future technology decision making. Both proprietary and open source have their own unique advantages and challenges.

6. Legacy challenges

Organizations older than 5 years will need to address legacy concerns. For example, architects at such organizations may be responsible for digital transformation and moving their existing applications from a data center to a cloud environment.

Not only do you need to choose technologies that work in the cloud (and are easy to work with, from a user perspective) but you also need to ensure the movement of the legacy applications is smooth and makes digital transformation projects considerably simpler.

How to evaluate TECHNOLOGY SOLUTIONS

Architects and technology leaders either making decision themselves or influencing the teams that ultimately will, it's important to focus on business requirements and the real value the potential solution will provide.

Equally important is to ensure that the technology will work as intended and understood, and to realistically recognize any trade-offs involved in the decision.

- What is the impact on the broader organization?
- Does adopting the solution make it more difficult for other teams to interoperate with this particular technology?
- Does it increase complexity?

Ultimately, technology choice is not only about discovering new solutions, but knowing when to say no and staying with something that already works.

Focusing attention within the technology stack

Today, many organizations use task-driven automation to scale up or down cloud resources based on prescheduled and/or repetitive events. For example, to support a daily 8:00 a.m. spike in usage, a task is set to automatically scale up a predefined set of resources to support the workload's needs and scale them down at 9:00 a.m. This is not new technology. However, it is a pre-determined, fixed process that is not suitable for the complexity of today's cloud infrastructure and the dynamic needs of application workloads.

What's required is an agile solution, embedded within cloud operational processes, that can proactively — and automatically — identify and deploy the most efficient resources for specific workloads (both containerized and non-containerized) consistently at precisely the right time and at the right cost.

Consider technology choice trade-offs

Even where a technology change provides more value than it costs, there are trade-offs— for example, there may still be higher value initiatives that those engineers could be working on. Each technology choice, new services, and new line of code is a tradeoff in itself.

Can the organization support this? Is something that already exists "near enough and good enough", meaning does it help someway to solve the problem?

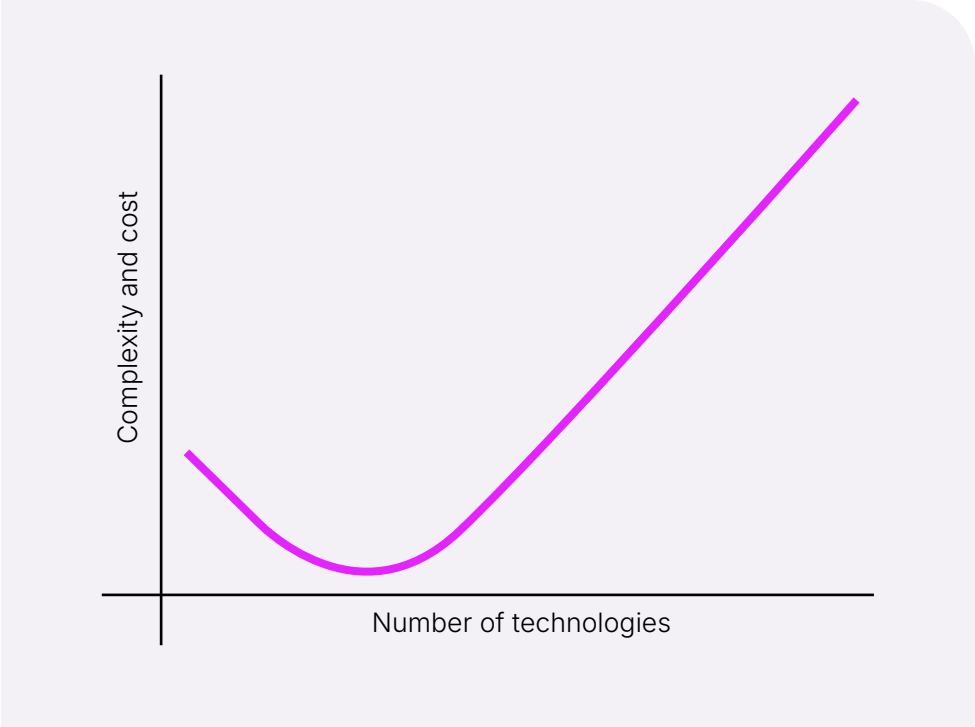
Technology decision-makers must have the broader business awareness to make near enough/good enough solution choices knowing that business requirements will change in the future.

A high-potential service may seem to need scalable databases, but if the team is mostly experienced in PostgreSQL, that near enough/good enough choice is the right decision for the time being, as you get started and then build later as you scale. It's critical to recognize the opportunity cost, and the reality of who is responsible for running a solution in production.

Be mindful of complexity

As an organization grows and adopts more technologies, complexity follows a specific curve. Initially, adopting technologies to solve particular needs reduces inefficiencies. This happens as organizations benefit from a greater range of tools that are better suited to their tasks. However, adopting more technologies increases the mental overhead on the teams who run (and develop) these solutions.

Complexity can be tricky to manage, especially within microservices environments where independent teams generally have 100% ownership of both the development and operational capabilities of those services. One strategy that works well is a "blessed tool kit" approach, where a central IT organization promotes certain technologies and offers broader support and incentives for teams that adopt those tools. This can help reduce the rate of technology adoption, while ensuring that independent teams can still choose the right tool for the right job.



The more technologies you have, the more complex–and costly–they will eventually become to manage

Leveraging the Instaclustr MANAGED PLATFORM

Our integrated managed platform includes 100% open source technologies like Apache Cassandra, Apache Kafka, Apache Spark, and OpenSearch. These data-centric, highly available and highly scalable technologies work well together and solve a common set of problems across a range of use cases and industry specific requirements—all with no vendor lock-in.

These technologies are well-suited to helping organizations meet customer SLAs and similar obligations in any environment: single- or multi-cloud, hybrid and on-prem.

UNDERSTANDING OPEN SOURCE *Data layer technologies*

Apache Cassandra

Apache Cassandra is a highly available, highly scalable, NoSQL data store that leverages a Dynamo architecture with a bigtable-style data model. Dynamo architecture allows you to scale linearly, with each added node owning a proportionally smaller portion of the total address space.

Cassandra uses a leaderless architecture, such that any node can serve any request. It also offers tuneable consistency for strong, weak, and guaranteed consistency either globally, locally, or across a subset of nodes.

These properties make Apache Cassandra an excellent building block for any architecture that has significant availability requirements. Cassandra also supports higher-order concepts like lightweight transactions, batch operations, counters, indexes, materialized views, and offers support for JSON and collections.

When should I use Apache Cassandra?

As a general-purpose transactional database, Cassandra's strengths are in providing scalability and availability. It is a strong option when you:

- Require availability of 99.9% or more
- Need to replicate data across different data centers and environments
- Must scale up or down from 3 to 100 nodes in minutes

When should I NOT use Apache Cassandra?

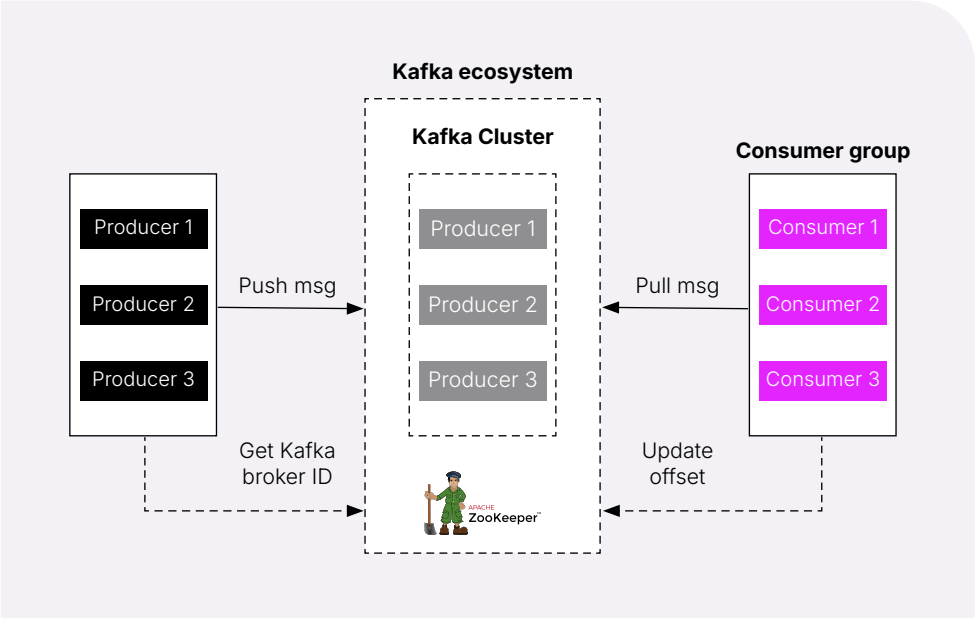
Cassandra probably isn't the best option for organizations rapidly iterating on greenfield projects—especially so for those with no prior experience with that problem domain.

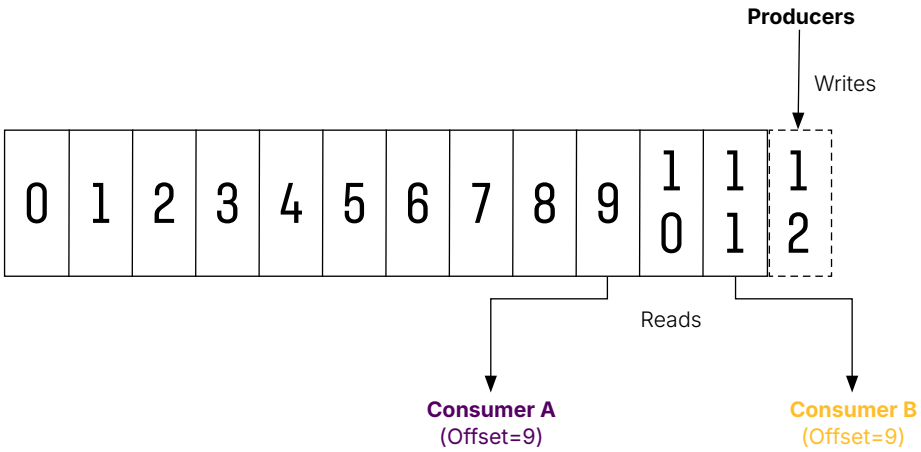
Why?

- It's not suited for pure analytics storage or data warehousing
- While it does have Spark connectors and Tableau plugins, it won't be fast
- There are no translytical or real-time analytics
- Cassandra doesn't provide most ACID requirements

Apache Kafka

Apache Kafka is a highly scalable, highly available streaming platform that is effectively a distributed log: as messages come in, they're appended to the head of the queue, where many readers (consumers) consume them based on an offset.





Kafka is particularly efficient under the hood and is powerful for building streaming applications and event and message buses. The solution can be configured to be either persistent or in-memory.

Kafka can also deliver high performance from a small cluster; for example 6 nodes can handle millions of messages per second. These properties make Kafka an excellent building block for any architecture with significant availability and throughput requirements. Kafka supports higher-order concepts like stream processing, SQL on top of streams, and connectors for other data services, event sources, or syncs.

Kafka's primary message model is a topic-based system, in which messages can be published to specific topics, and consumers with queues registered for those topics will receive those messages.

When should I use Apache Kafka?

As a powerful general-purpose message bus, Kafka offers a number of strong use cases:

- Service-orientated architecture and microservices
- A powerful work queue
- Passing metrics through and using stream processing capabilities for roll-ups, aggregations, and anomaly detection
- Event sourcing
- Reconciling data across different microservices
- An external commit log for other distributed systems

Kafka's general-purpose streaming platform offers additional use cases, including log aggregation, metrics, fraud and anomaly detection, data masking and filtering, and data enrichment.

When should I NOT use Apache Kafka?

- Kafka is tricky when used as a source-of-record or a database as it requires a fairly large shift in how you work with your data; it is generally easier to use a true database instead.
- In general, Kafka also shouldn't be used for in-order processing across an entire topic as this requires significant additional effort.

Apache Spark

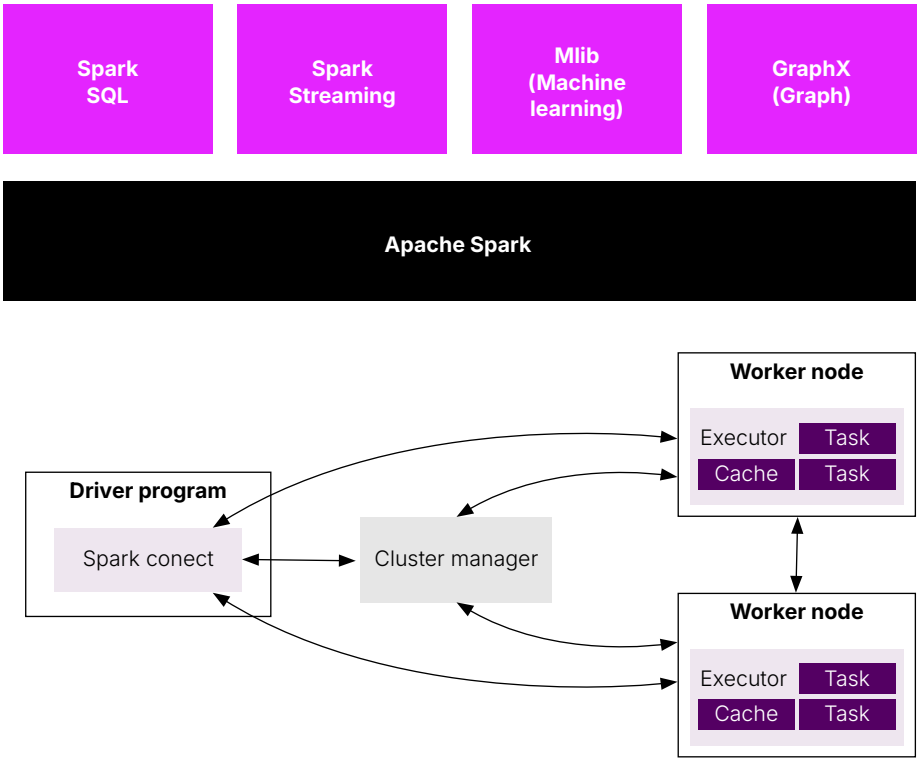
Apache Spark is a general-purpose cluster computing framework that is ideal for working with large data volumes. Spark breaks up data into segments or splits and then runs computation on them—with individual workers doing all they can until they need data from other workers. Spark is highly scalable, and remarkably resilient when it comes to protecting against data loss and delivering availability.

Spark supports many different models, enabling functions like map/reduce, SQL, batch processing or streaming, graph processing, and machine learning capabilities. Spark also supports different environments and schedulers like Kubernetes.

When should I use Apache Spark?

Spark is useful for large scale analytics—and especially analytics involving multiple sources of data:

- ETL (Extract, Transform, Load) use cases
- Moving data from one system to another (whether on a constant basis or one-off)
- Building ML pipelines on top of existing data, high latency streaming and exploratory analysis
- Meeting compliance needs, such as masking data, data filtering and compliance audits



When should I NOT use Apache Spark?

Spark is generally not appropriate for real-time or low latency processing; other technologies, like Apache Kafka, offer better end-to-end latency (this applies to real-time stream processing as well). Spark also tends to be overkill for small or single datasets. And, while there are data warehousing and data lake products built around Apache Spark, it's better to use something higher level.

OpenSearch

OpenSearch is a full-text search engine with a great web interface. It's generally used to provide scalable linear search in near real-time and offers strong drop-in search replacement. OpenSearch is distributed, which means that indices can be divided into shards and each shard can have zero or more replicas. Each node hosts one or more shards and acts as a coordinator to delegate operations to the correct shard(s).

When should I use OpenSearch?

OpenSearch is a particularly strong fit for use cases that include full-text search, geographic search, logging and log analysis, scraping and combining public data, event data and metrics (at a small volume), and visualizations.

When should I NOT use OpenSearch?

OpenSearch is generally not appropriate for use as a database or source-of-record, for use with relational data, or for meeting ACID requirements.

How these scalable technologies COMPLEMENT EACH OTHER

Apache Cassandra, Apache Kafka, Apache Spark, and OpenSearch provide a set of common attributes and complementary capabilities. When leveraged together, these technologies can be used to build applications that are highly scalable, resilient, portable, and free from license fees and vendor lock-in. Entirely open source, they are tried and tested, super resilient and suitable for enterprise-grade and mission critical deployments.

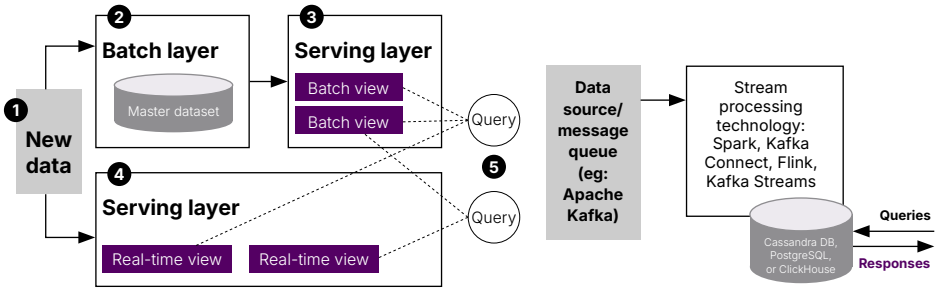
There are numerous ways these technologies can work together, but Lambda, Microservices and Extract, Transform, Load (ETL) architectures are most prominently used.

Lambda architecture

Lambda architecture splits up a task/responsibility on how it works with data. Designed to handle massive quantities of data by taking advantage of both batch and stream processing methods, it attempts to balance latency, throughput and fault-tolerance by using batch processing to provide a comprehensive and accurate view of batch data while using real-time stream processing to provide views of the online data.

The architecture comprises 3 distinct layers: speed layer, batch layer, and serving layer.

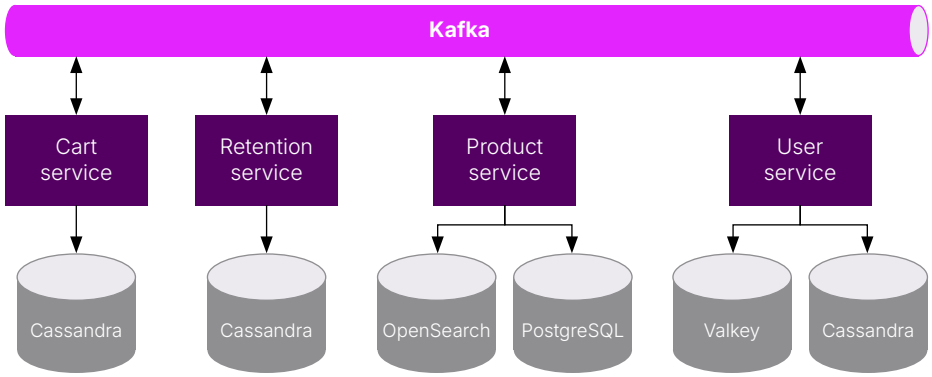
While the batch layer processes the large volume of data from the master dataset, the speed layer provides a real time view of the datasets and the output of both the layers are saved on the serving layer which responds to ad-hoc queries by building a view from the processed data:



(Instaclustr-managed technologies providing Lambda architecture)

Microservices architecture

Within this environment you can have a number of different services. For example, an e-commerce platform may have a cart service, user retention service, product service or user service. Each of these services has a defined set of responsibilities and rules or protocols for interacting with one another.



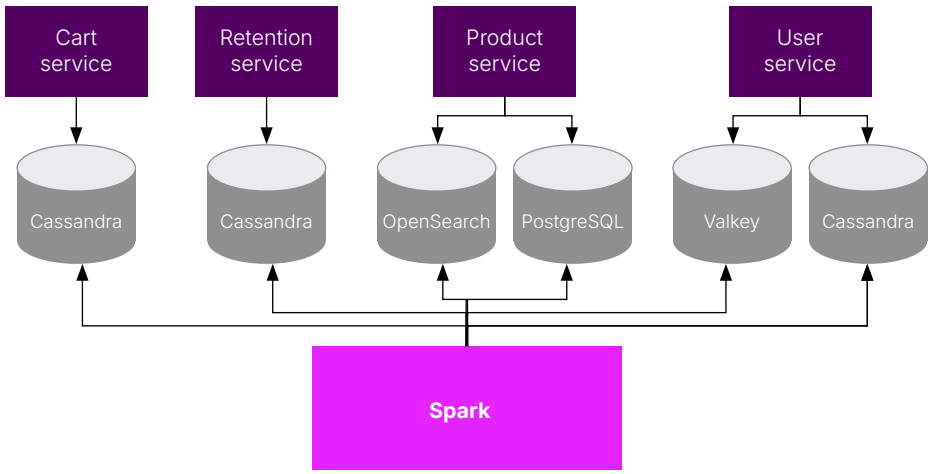
(Instaclustr-managed technologies providing Microservices)

In this type of deployment, shared data is one of the more complex areas that needs to be dealt with. A solution to this complexity is understanding the concepts of an event flow, Apache Kafka provides the fundamentals for this type of architecture.

Microservices are responsible for generating events. These events can be published to a topic that any number of consumers can potentially consume. Different services can subscribe to updates from the message bus as it pushes from the existing service. This is often called event sourcing.

Extract, Transform, Load (ETL) architecture

In an ETL architecture, data is copied from one or more sources into a destination system which represents the data differently from that provided by the source or sources. The deployment of Apache Spark is a great example of the extraction and transformation, pulling data from existing data stores. You can also use Spark as an ETL mechanism for different Microservices.



(Instaclustr-managed technologies providing ETL)

Technology choice is a tradeoff between effort and value. It is about enabling value in what you are building, translating to looking at not just at the technology itself, but how an organization adopts it.

FINAL *Thoughts*

Why open source is the go-to choice for scalable, data- layer technologies

Adopting Instaclustr's **platform of managed services** can accelerate your application's time to market. Instaclustr's commitment to fully open source solutions like Apache Cassandra, Apache Kafka, Apache Spark, OpenSearch and more, further removes risk by ensuring portability and freedom from vendor lock- in.

Instaclustr's distributed technologies and support mean that organizations can realize increased availability, without increasing complexity for internal ops teams.

Reach out to our open source experts and let us guide you through the complexities of modern tech, enabling your organization to focus on innovation and delivering exceptional customer experiences.

NetApp® Instaclustr specializes in open source technologies for enterprises. Our managed platform streamlines data infrastructure management, backed by experts who ensure ongoing performance, scalability, and optimization. This enables companies to focus on building cutting edge applications at lower costs.

3-12feb25