White Paper

An Architect's Guide for Selecting Scalable, Data-Layer Technologies

Overview

As a solution architect one of our most critical early-stage tasks is selecting the right foundational data-layer components on which you will build your application. The data-layer provides the platform on which you must rely to build in resilience through key factors including scale, availability, security, and performance.

The technology choices that provide the platform or foundation for critical business applications will ultimately have a far wider and deeper reaching impact across the organization than just a specific application deployment. These choices will also affect everything from hiring decisions, to how quickly customer demands and business requirements can be met. Choosing a niche technology can draw dedicated and excited individuals, whereas, this can also lead to not being able to readily attract qualified candidates.

Architects and CTOs must be especially considerate in their evaluation of how a specific technology solution delivers value, and careful of any added complexity or other drawbacks a solution may bring with it. Matching the right tool for the job—from the beginning—is a huge advantage that could help avoid headaches (and costs) down the line. It is important to note that this is also a continual process that needs to be considered and managed as strategic imperatives and requirements change.

In this white paper, we detail the challenges faced while making technology choices and the basics of selecting the right technology for your application architecture. We will cover four open source technologies that all reside at the data-layer: Apache Cassandra®, Apache Kafka®, Apache Spark™, and Elasticsearch™ and we will consider both when to, and when not, to use them. We also explore how and when these technologies can be used together.

instaclustr

Key Challenges Faced When Making Technology Choices

Hiring

Technology choices both influence and are influenced by personnel hires. When implementing a new technology, decision-makers must choose whether to hire engineers with specific experience (*which can be hard to come by for the newest solutions*), or to hire smart people and build up their skill sets. At the same time, organizations with established teams will naturally lean toward the technology choices that fit those engineers' skills and preferences. Choosing technologies with large communities and broad exposure among engineers can drastically reduce these hiring challenges.

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 can accelerate teams—for example, implementing microservices can allow large teams to iterate concurrently.

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—for example, by moving from a homegrown container scheduler to a solution such as Kubernetes.

Security and Compliance

Technology choice has a profound impact on security and compliance, as do business and customer requirements.

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.

Legacy Challenges

In addition to the challenges listed above, organizations older than five years need to also address legacy concerns. For example, architects at such organizations may be responsible for digital transformation and may be responsible for 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

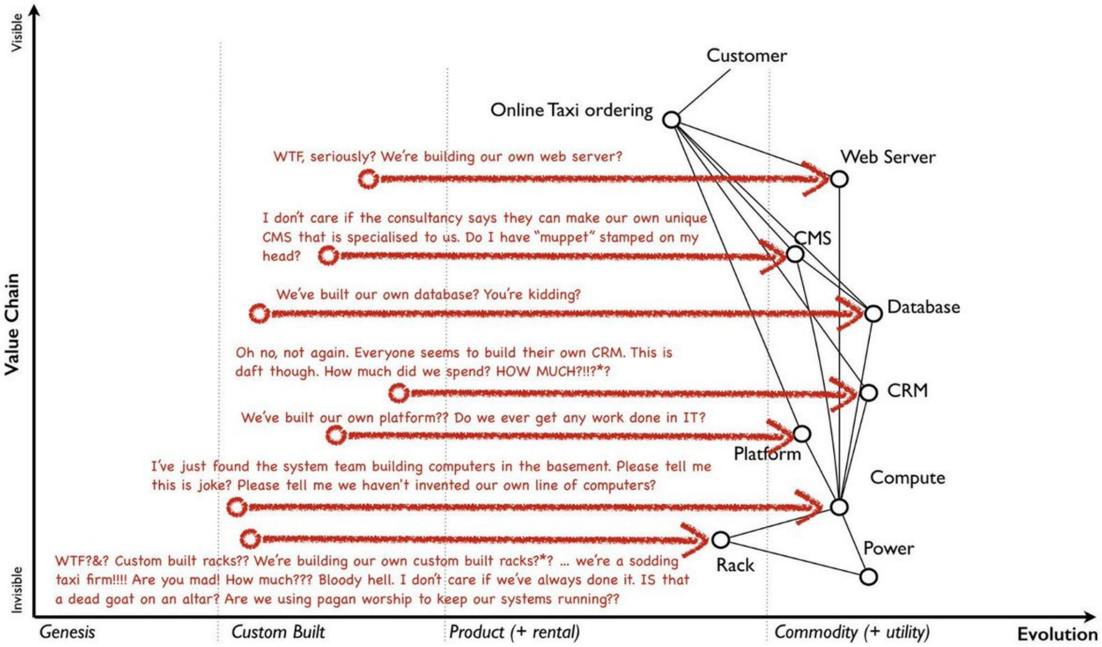
As architects and technology leaders making (*or influencing teams to make*) technology decisions, it's imperative to focus on business requirements and the real value the potential solution will provide—while watching out for buzzwords and the pull of the latest trends that might not be the best fit for your use case.

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 stay with something that already works.

Focusing Attention Within the Technology Stack


Visible
Value Chain
Invisible
Customer
Online Taxi ordering
WTF, seriously? We're building our own web server?
Web Server
I don't care if the consultancy says they can make our own unique CMS that is specialised to us. Do I have "muppet" stamped on my head?
CMS
We've built our own database? You're kidding?
Database
Oh no, not again. Everyone seems to build their own CRM. This is daft though. How much did we spend? HOW MUCH?!!?*?
CRM
We've built our own platform?? Do we ever get any work done in IT?
Platform
I've just found the system team building computers in the basement. Please tell me this is joke? Please tell me we haven't invented our own line of computers?
Compute
WTF?&? Custom built racks?? We're building our own custom built racks?*? ... we're a sodding taxi firm!!!! Are you mad! How much??? Bloody hell. I don't care if we've always done it. IS that a dead goat on an altar? Are we using pagan worship to keep our systems running??
Rack
Power
Genesis
Custom Built
Product (+ rental)
Commodity (+ utility)
Evolution

Wardley Maps

Image source: https://medium.com/@ianwaring/it-trends-for-2017-or-the-many-delusions-of-ian-waring-34e70f2657c8

The graph shows a high level, generic technology stack with a web server, a CMS, a database, a CRM that interacts with the database, a compute platform, all the way down to the racks and servers and power infrastructure. What we see is that most of the time, the value an organization provides is focused almost entirely on the top layer. There is not enough consideration and design effort focused on the foundation infrastructure and data-layer components that are required to deliver the application with fundamental requirements or security, scale, availability, and performance.

Part of the technologist's dilemma is choosing solutions that enable an organization and the application to succeed, while the organization is focused on the features of the application or the very top of the value chain. As an organization grows the inefficiencies in adopting off-the-shelf components start to magnify. Building a foundational infrastructure and data-layer that delivers the key characteristics on which applications can be built is the way large enterprises can focus technology choices on key areas where the most value may be derived for the business.

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.

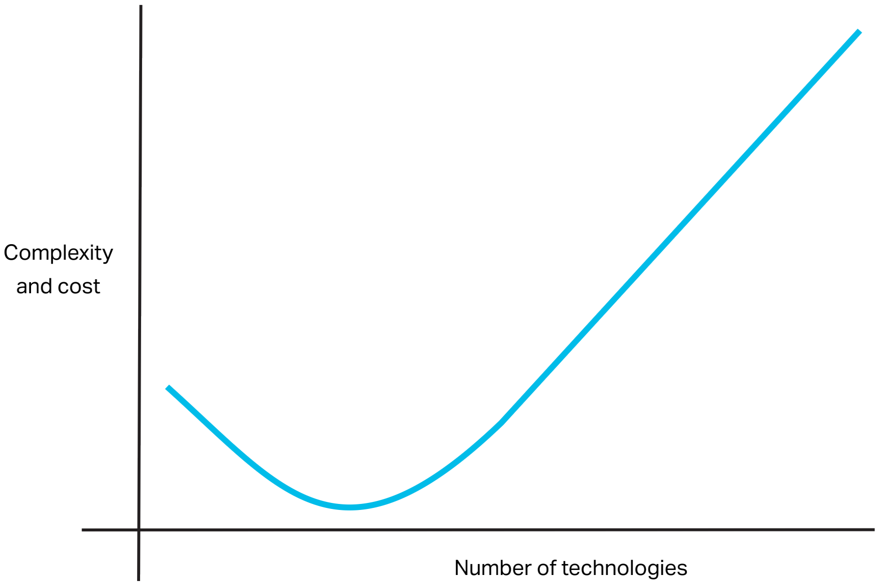
Can the organization support this? Is something that already exists is "near enough and good enough" meaning, does it help someway to solve the problem? Technology decision-makers must have the broader business awareness to make near enough/good enough solution choices when it's known that business requirements will change in the future. For example, a high-potential service may seem to need scalable databases, but if the team is mostly experienced in MySQL, 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 for) 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.



Leveraging Instaclustr Managed Platform

Our integrated managed platform includes open source technologies such as Apache Cassandra, Apache Kafka, Apache Spark, and Elasticsearch. 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.

Instaclustr provides these technologies in a way that offers flexibility in any environment – whether it's cloud, multi-cloud, hybrid or on-prem. All this as a pure-play open source option with no vendor lock-in. These data technologies are also well-suited to helping organizations meet customer SLAs and similar obligations.

Understanding Open Source Date Layer Technologies

Apache Cassandra®

Apache Cassandra is a highly available, highly scalable, NoSQL data store, which originally came out of Facebook (in 2007) and 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. It uses a leaderless architecture, such that any node can serve any request. Cassandra 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?

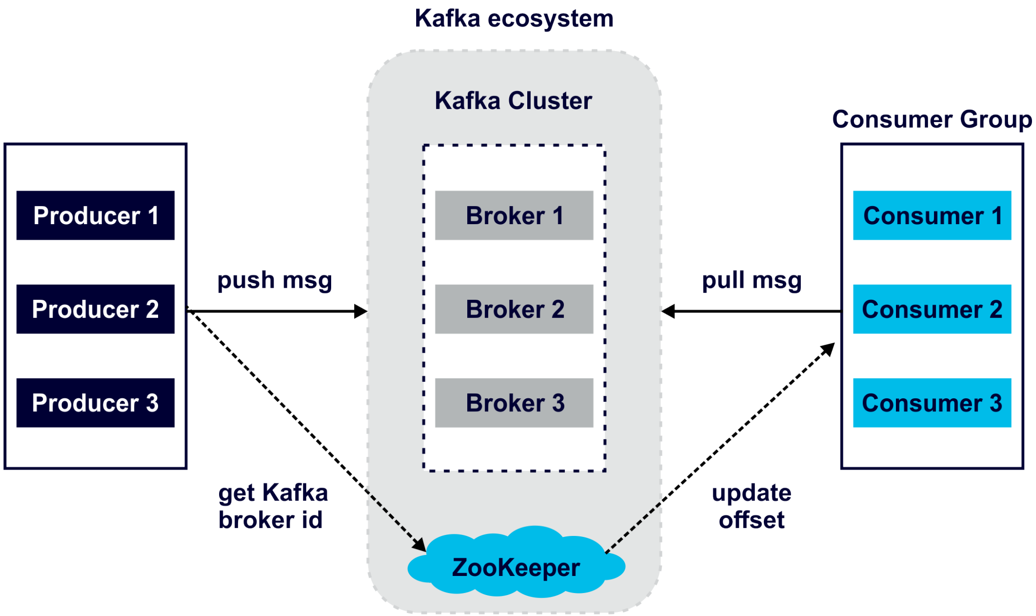
As a general-purpose transactional database, Cassandra's strengths are in providing scalability and availability. Cassandra is a strong option for organizations requiring availability of 99.9% or more. It's also a dependable choice from a data redundancy and availability perspective, capable of replicating data across different data centers and environments (while offering relatively seamless data portability). Additionally, Cassandra is an apt fit for meeting the scalability needs of enterprises with environments facing increasing workloads, or those looking to grow services flexibly as load increases as you can scale up and down from 3 to 100 nodes in minutes.

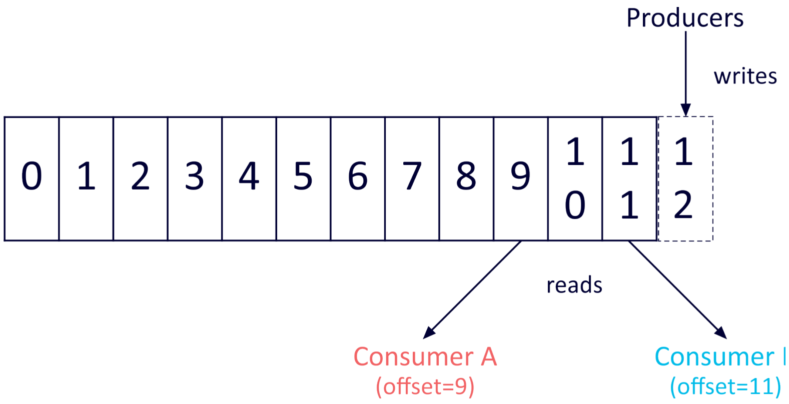
When should I NOT use Apache Cassandra?

Cassandra probably isn't the best option for organizations rapidly iterating on greenfield projects—especially so for those with no prior experience with that particular problem domain. Cassandra is also not as suited for pure analytics storage or data warehousing use cases. While Cassandra does have Spark connectors and Tableau and Hadoop plugins (and allows operations such as full table scans), it's not going to be as fast as other options. Cassandra also doesn't lend itself to translytical or real-time analytics (e.g. end user ad-hoc or custom queries), as the necessity to implement query code application-side that can add a lot of complexity and is more tricky. Cassandra also doesn't provide most ACID requirements, but you will find that most use cases don't have a strict requirement around ACID!

Apache Kafka®

Apache Kafka is a highly scalable, highly available streaming platform, and message bus that originally emerged from LinkedIn's technical team. Kafka is effectively a distributed log: as messages come in, they're appended to the head of the queue, where a number of 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 six 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. It's excellent for use with service-oriented architecture/microservices. Kafka can be used as a powerful work queue, coordinating separate work paths that compute power can listen on and wait for. The platform is great for passing metrics through, and using its stream processing capabilities for roll-ups, aggregations, and anomaly detection. Kafka is also effective for event sourcing, reconciling data across different microservices, and as an external commit log for other distributed systems. Kafka's general-purpose streaming platform offers additional interesting use cases as well, 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 to be 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 just easier to use a true database instead.

In general, Kafka also shouldn't be used for in-order processing across an entire topic as it's not supported out of the box and requires significant additional effort to support this use case. Nor should it be put to work with lossy data streams (such as real-time audio or video), where the goal is to get packets of data to the end source as quickly as possible. Instead, use a solution more purpose-built for those data streams.

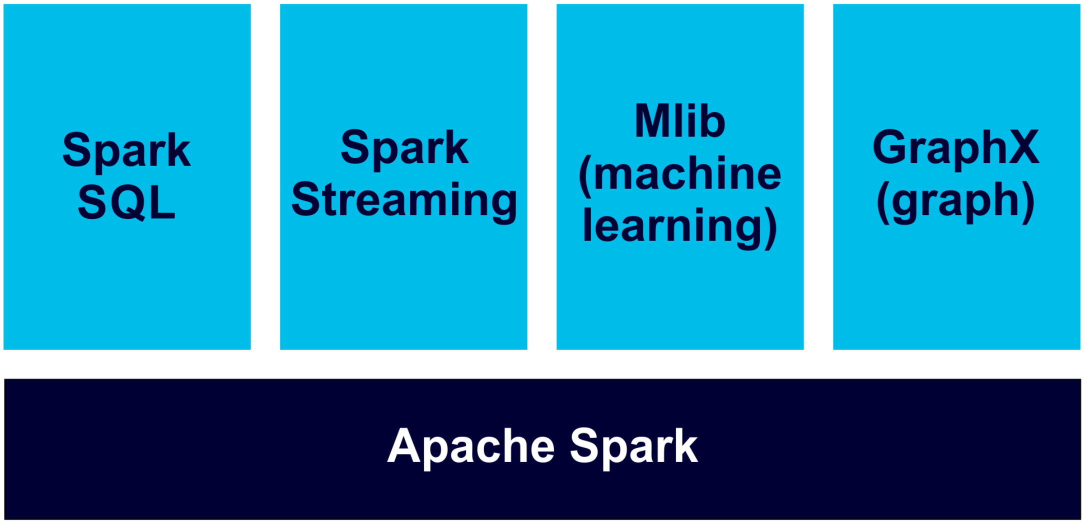
Apache Spark™

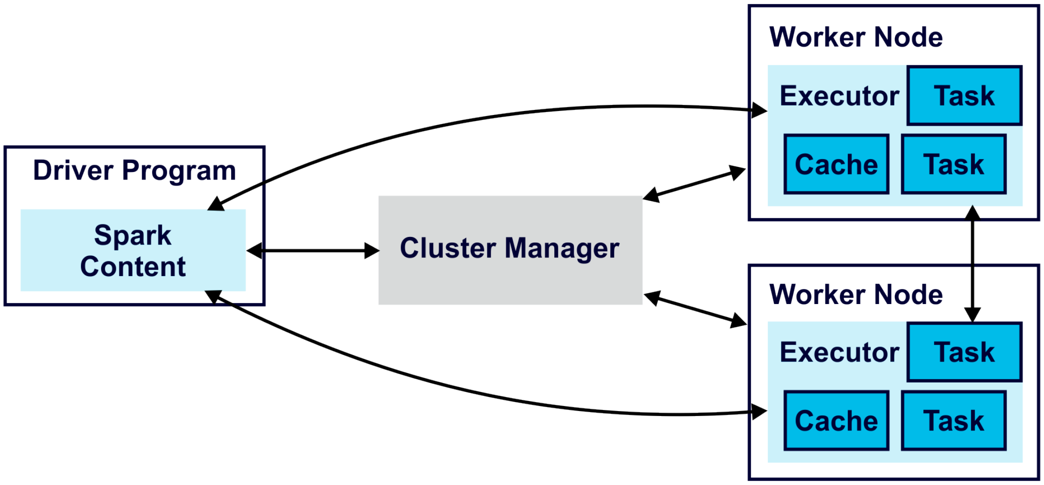
Apache Spark is a general-purpose cluster computing framework that is ideal for working with large data volumes. Spark breaks up data into segments or splits and then runs computation on them—with individual workers doing all they can until they need data from other workers. Spark is highly scalable, 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 Mesosphere and Kubernetes.

When Should I Use Apache Spark?

Spark is useful for large scale analytics—and especially analytics involving multiple sources of data. ETL is another primary use case. Spark is great for moving data from one system to another, whether it's on a constant basis while populating a data warehouse or a data lake from transactional data stores, or a one-off use case such as migrating from one database or one system to another. Spark is also a solid choice for building ML pipelines on top of existing data, high latency streaming, and interactive, ad-hoc, or exploratory analysis. Additionally, Spark can be useful for meeting compliance needs—for example, by providing data masking, data filtering, and compliance audits of large data sets.





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.

Elasticsearch™

Elasticsearch is a full-text search engine with a great HTTP web interface. It's generally used to provide scalable linear search in near real-time, supports multi tenancy, and offers strong drop-in search replacement. Elasticsearch 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 Elasticsearch?

Elasticsearch 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 Elasticsearch?

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

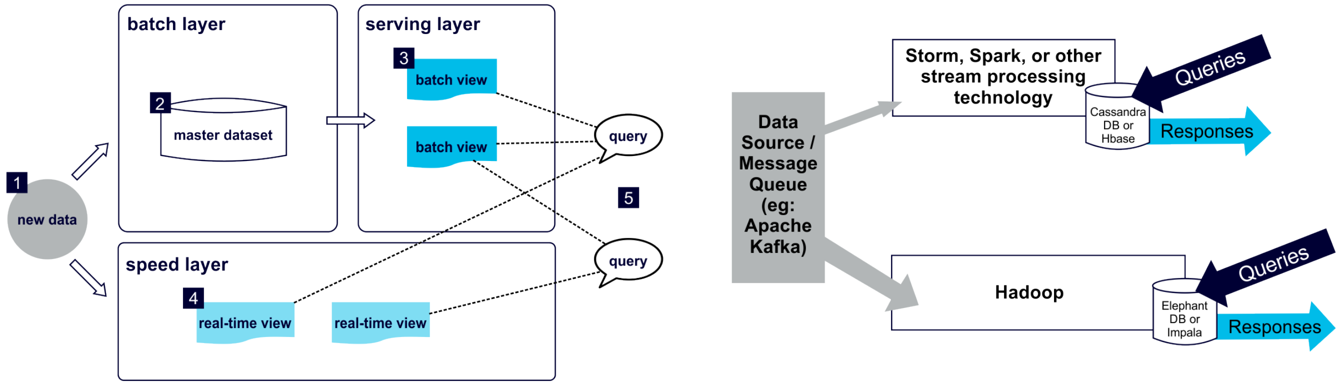
How These Scalable Technologies Complement Each Other

Apache Cassandra, Apache Kafka, Apache Spark, and Elasticsearch 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. These technologies in their open source format are tried and tested, super resilient and suitable for enterprise-grade and mission critical deployments.

Lambda Architecture

There are various ways in which these technologies can be deployed together, for example the Lambda architecture. The 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 three 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.

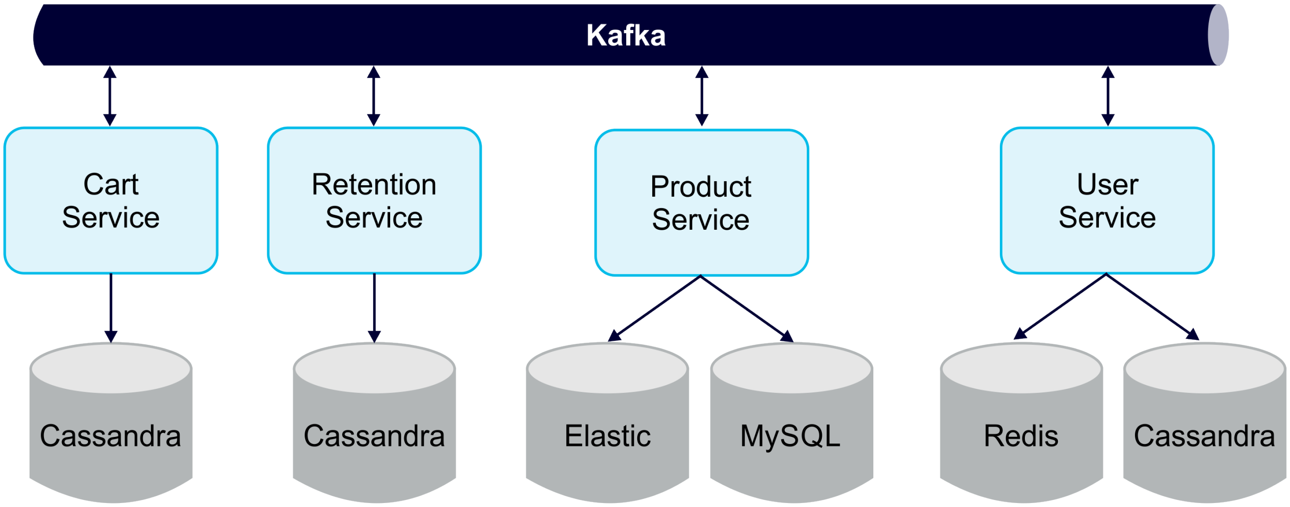


(Instaclustr-managed technologies providing Lambda architecture)

Microservices Architecture

Another great example of the way that these open source scaling technologies are deployed is within a Microservices environment.

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.

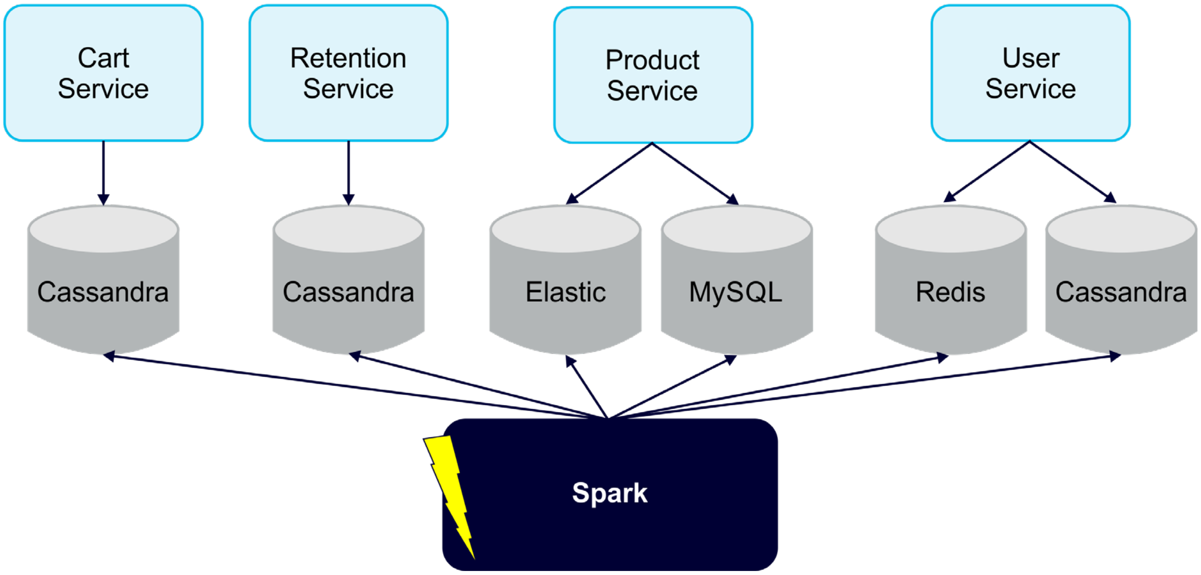


(Instaclustr-managed technologies providing Microservices)

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 from. 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

The third use case would be a traditional extract, transform, load (ETL) architecture. 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 of the data. 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.

Relying on Instaclustr's platform of managed services can accelerate your application's time to market—while reducing the risk of adopting new technologies. Instaclustr's commitment to fully open source solutions 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.

About Instaclustr

Instaclustr helps organizations deliver applications at scale through its managed platform for open source technologies such as **Apache Cassandra®**, **Apache Kafka®**, **Apache Spark™**, **Redis™**, **OpenSearch®**, **PostgreSQL®**, and **Cadence**.

Instaclustr combines a complete data infrastructure environment with hands-on technology expertise to ensure ongoing performance and optimization. By removing the infrastructure complexity, we enable companies to focus internal development and operational resources on building cutting edge customer-facing applications at lower cost. Instaclustr customers include some of the largest and most innovative Fortune 500 companies.