O'REILLY®
Report

Streaming Data for Real-Time AI Applications

Performant, Efficient, and Reliable Apps for Real-Time Intelligence

Blaize Stewart & Bipin Singh

Redpanda

Simple. Efficient. Interoperable. Safe.

Build the foundation for enterprise AI with an event-driven platform designed for performance, tuned for cost, and compatible with your ecosystem.

StoneX

Optimized its trading and ML model training systems, processing 4TB of logs daily using 5-6x fewer resources than Apache Kafka®

ACRES

Powered real-time analytics at the edge using live data, unlocking instant insights that increased customer engagement and revenue

LACEWORK

Scaled its data streaming 10x and streamlined operations, simplifying its expansion to new clouds while saving 30% on costs

Build powerful real-time and AI applications—the easy way.

TRY REDPANDA FOR FREE

> **Redpanda is an entire streaming engine ready to go. The results we got with almost no configuration were incredible."**

CTO at Alpaca, **Raja Bhatia**

Streaming Data for Real-Time AI Applications

Performant, Efficient, and Reliable Apps for Real-Time Intelligence

Blaize Stewart and Bipin Singh

O'REILLY®

Streaming Data for Real-Time AI Applications
by Blaize Stewart and Bipin Singh

Published by O'Reilly Media, Inc., 141 Stony Circle, Suite 195, Santa Rosa, CA 95401.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*https://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Aaron Black
Development Editor: Gary O'Brien
Production Editor: Gregory Hyman
Copyeditor: Audrey Doyle
Cover Designer: Susan Brown
Cover Illustrator: Susan Brown
Interior Designer: David Futato
Interior Illustrator: Kate Dullea

August 2025: First Edition

Revision History for the First Edition
2025-08-12: First Release

979-8-341-62388-0

[LSI]

Table of Contents

CHAPTER 1

Why Real-Time AI Needs Streaming Data

At its core, real-time AI is about immediacy; that is, delivering insights, making decisions, and adapting to new information as it emerges. This demand for immediacy stems from the fundamental nature of intelligence itself, which is the ability to respond to the world as it changes. Human cognition does not process events in fixed intervals; rather, people react to them as they happen. Similarly, real-time AI reacts instantly, drawing on extensive pre-training while adapting continuously to live conditions. In these systems, freshly retrieved information brings immediate relevance into the context window, either through mechanisms like retrieval-augmented generation (RAG) or by leveraging recent context. This fresh data enables timely, informed responses grounded in the most relevant available knowledge.

Unlike batch processing, which relies on delayed inputs, streaming data provides a continuous flow of up-to-the-moment information. This empowers organizations to act in real time, rather than waiting for insights that may already be outdated. Consider the following use case that involves a customer calling into a call center:

1. The customer begins a conversation with a support agent through live chat, or a phone call is transcribed into text in real time.

2. As the conversation unfolds, the system captures the most recent portion of the dialogue to maintain an up-to-date context window.
3. This context is transformed into a structured representation that allows for semantic comparison with existing knowledge.
4. The system uses this representation to search a repository of internal content, such as documentation, frequently asked questions, and previous support tickets.
5. The search returns the most relevant documents that match the current conversation context.
6. The retrieved documents are combined with the live conversation to form an input prompt for a language model.
7. The language model generates a response that incorporates both the retrieved content and the customer's current inquiry.
8. The response is either sent directly to the customer or reviewed by a support agent before being delivered, depending on the workflow.
9. As the conversation continues, this process repeats, continuously updating the context and generating new responses.
10. Feedback from agents or customers, including edits or ratings, is recorded and may be used to improve the quality of future retrieval and generation.

This workflow delivers immediate, context-aware assistance that enhances the experience for both customers and support agents. Customers benefit from accurate, relevant answers without long wait times or the need to repeat themselves, while agents are supported by intelligent suggestions that reduce cognitive load, allowing them to focus on complex or sensitive issues. The system draws on institutional knowledge in real time, making even new or junior agents more effective by surfacing resolutions drawn from prior cases and documentation. Managers gain insight through feedback loops, enabling continuous improvement of both the knowledge base and AI performance. As a result, the interaction becomes more efficient, scalable, and aligned with the organization's service standards.

This customer service example uses AI as part of its processing in real time. It, like most real-time systems, is reducible to three entities: producers, brokers, and consumers (see Figure 1-1), although the terminology may differ slightly between implementations:

Producers
: These entities create and broadcast data or events, triggered by human input or automated systems capturing live interactions. In the customer support scenario, the producer is the ongoing chat or voice transcription, which generates conversational data as the user engages with the support interface.

Brokers and pipelines
: These entities transport and refine data as it moves through the system. Pipelines may clean the input, extract context, or enrich it with additional metadata such as user history or ticket categorization. AI assists in these steps by maintaining context windows, identifying intents, and performing semantic encoding for retrieval. In this case, the broker is the orchestration layer that receives the live input and prepares it for retrieval and generation.

Consumers
: Consumers listen for specific, relevant input, retrieve associated knowledge, and generate responses. These responses are then presented to the support agent, who may review, edit, or approve them before sharing them with the customer. In this scenario, the consumer is the agent-facing AI assistant that synthesizes responses in real time based on retrieved knowledge and evolving conversation context.

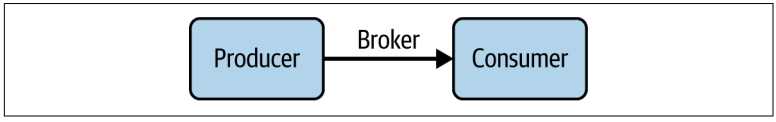

Figure 1-1. The basic message pattern

This is a basic template for more particular messaging patterns commonly used in real-time data systems.

Common Messaging Patterns

Most modern systems follow different architectural patterns, depending on the use case. Three widely used messaging patterns are point-to-point (queue), publish–subscribe (pub/sub), and event streaming. All of these build on the concept of a producer, broker, and consumer. They all involve message passing, but they serve distinct purposes in terms of message distribution, scalability, and processing guarantees. Table 1-1 gives a summary of the patterns.

Table 1-1. Message pattern summary

Pattern	Common use cases	Supporting technologies	How AI assists
Point-to-point (queue)	Task/job distribution; background operations; decoupling app layers	RabbitMQ (queue), Azure Queue Storage, Amazon SQS, ActiveMQ	Filters messages before enqueue (noise reduction, context awareness); classifies and enriches data before queuing; prioritizes queue handling (value, latency); enables dynamic, context-driven consumer logic
Publish–subscribe (pub/sub)	Real-time notifications; event-driven microservices; Internet of Things (IoT) data processing	Google Pub/Sub, Azure Service Bus (topics), Amazon SNS, RabbitMQ (fanout)	Enriches, filters, and routes messages at broker; predicts subscriber needs; manages topics dynamically; enables adaptive consumer logic (dynamic escalation/suppression); manages stateful consumers (buffering, retry)
Event streaming	High-volume data ingestion; real-time analytics; replayable event history	Apache Kafka, Redpanda, Apache Pulsar, Amazon Kinesis, Azure Event Hubs	Classifies and detects anomalies on ingestion; performs trend detection and aggregation over time windows; enables stateful, sequential, and multievent reasoning for consumers

Point-to-Point

The point-to-point (queue) pattern ensures that each message is consumed by only one recipient. A producer sends messages to a queue, where they are held until a single consumer retrieves and processes them, as seen in Figure 1-2. Once processed, the message is removed from the queue, preventing duplicate processing. This model is ideal for task distribution and job processing, such as handling background operations in a web application.

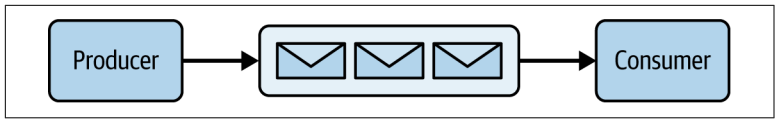

Figure 1-2. A consumer queue

AI enhances this traditional pattern at all points:

Producer
: AI can decide whether to queue data based on context, thresholds, or patterns, reducing noise and ensuring relevance. The customer service example uses an algorithmic approach with heuristics to trigger events.

Pipeline
: Before enqueueing, AI can preprocess, classify, or enrich data with metadata or transformations for better downstream use. As mentioned, the customer service agent pipeline filters and enriches context with AI.

Queue management
: AI can guide prioritization based on value or latency needs, though queues remain FIFO (first in, first out) unless they are priority based. The customer service agent pipeline can use AI or other enrichments to filter or throttle messages on a queue.

Consumer
: AI can analyze messages and decide on actions like escalation, reprocessing, or follow-ups, replacing static handling with dynamic logic. The app does do AI, but it could further enrich the data or do analysis against historical data.

Technologies supporting this pattern include RabbitMQ (queue-based exchange), Azure Queue Storage, Amazon SQS, and ActiveMQ.

Pub/Sub

The publish–subscribe (pub/sub) pattern is designed for broadcasting messages to multiple independent consumers (see Figure 1-3). In this pattern, a publisher sends messages to a central message broker, which then delivers them to all subscribers who have expressed interest in a given topic. This allows multiple systems or services to consume the same event simultaneously without

direct awareness of one another. This model is commonly used in event-driven microservices, real-time notifications, and IoT data processing.

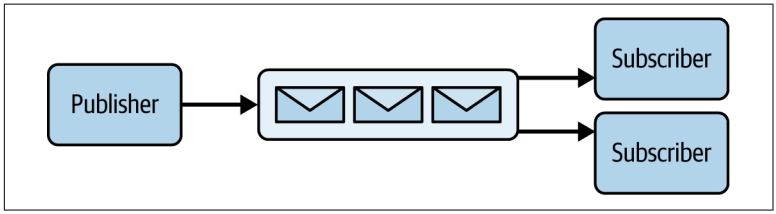

Figure 1-3. A publisher with subscribers

Messages are delivered to subscribers in real time as soon as they are published. The system pushes these messages to consumers, who process them and typically discard them once their task is complete.

AI in this context works much like point-to-point messaging, but with some nuances specific to the way queues work with AI:

Broker
: AI can enrich, filter, or reclassify messages, predict usage, and route them to the most relevant subscribers.

Subscription
: AI can manage when and what topics a subscriber listens to, optimizing for relevance and workload.

Consumer logic
: AI enables adaptive responses, adjusting actions like thresholds, escalations, or suppression based on context.

Stateful consumers
: AI guides buffering, retries, and message retention when offline, based on value or urgency.

Examples of pub/sub technologies include Google Pub/Sub, Azure Service Bus (topics), Amazon SNS, and RabbitMQ (fanout exchange). While this pattern enables flexibility and scalability, it does not guarantee strict message ordering, and messages are often transient unless explicitly configured for persistence. These are often supplemented by processing technologies, like serverless functions or stream processing frameworks.

Event Streaming

Unlike pub/sub or point-to-point messaging, event streaming is designed for high-volume, continuous data processing. In this pattern, events are persistently logged into a stream, allowing multiple consumers to read and process events in order. Whereas pub/sub messages may be transient, event streaming systems generally allow consumers to replay past events from a given point in time.[1]

Because of replayability, event streaming—in contrast to pub/sub—introduces a more stateful approach to consuming events. Rather than simply pushing messages to subscribers in real time, events are persistently logged in an ordered stream. Consumers read from this log at their own pace, allowing them to process independently (see Figure 1-4). The trade-off is that event streaming systems, such as Apache Kafka, require more infrastructure and state management. Consumers must track their position in the stream to ensure that they process events in the correct order.

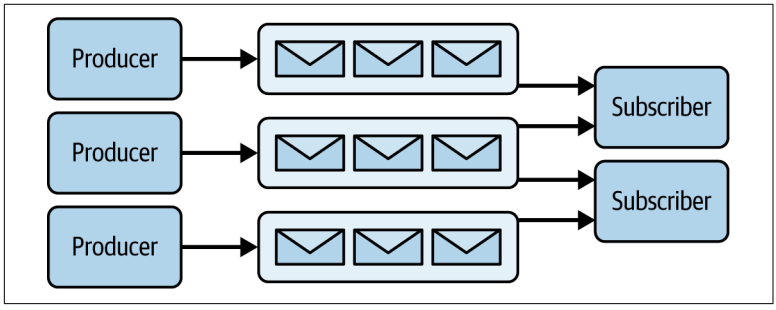

Figure 1-4. Event streams with subscribers

AI in this context has some unique uses that enrich how event streaming works:

Producers
: AI can classify, transform, or detect anomalies in real time before data enters the log.

1 Sooter Saalu, "Enterprise Messaging and Event Streaming Comparison," Redpanda blog, October 3, 2023, *https://www.redpanda.com/blog/enterprise-messaging-vs-event-streaming*.

Pipelines
: AI processes event sequences, performing aggregations, trend detection, and adaptive reasoning over windows of time.

Consumers
: AI analyzes current and historical data with stateful context, enabling sequential decisions and multievent reasoning.

The customer service example is designed to give real-time feedback of conversations as they occur. It could take advantage of event streaming if the agent wants to replay a context window, such as the conversation for the last five minutes, or replay the entire conversation for training purposes.

Technologies that enable event streaming include Redpanda, Apache Kafka, Apache Pulsar, Amazon Kinesis, and Azure Event Hubs. These systems support many-to-many processing, ensuring that multiple consumers can process the same data at different times, either in real time or retrospectively.

Key Trade-offs in Latency and Throughput with Streaming AI

Real-time streaming and batch processing differ mainly in latency. Streaming delivers insights in milliseconds to seconds for immediate actions, while batch processing processes data in intervals for deep analysis and historical reporting. Systems often combine both balanced speed and analytical depth across a spectrum from real time to batch.

Building a scalable, real-time AI system is not a single architectural choice; rather, it comprises layered optimizations that compound. Each one of these categories is a lever, and there are trade-offs between latency, persistence, computational complexity, and scalability that are deeply interconnected. A system that prioritizes low latency must minimize computational overhead, which in turn influences the choice of AI models, favoring simpler, more efficient approaches when possible. No single optimization exists in isolation. *Latency*—the time between data generation and consumption—is shaped by factors such as pipeline complexity, data volume, computational load, and reliance on persisted data for context (see Figure 1-5).

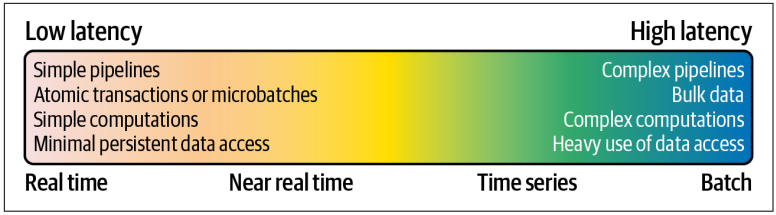

Figure 1-5. Trade-offs in latency

Simpler Pipelines Versus Complex Pipelines

Data pipelines range from simple, fast, and low latency with minimal steps, to complex pipelines with multiple transformations, branching, and parallel processing. Unlike simple pipelines, complex ones integrate with various data sources, APIs, microservices, and cloud services.[2] There is no strict boundary between what defines a simple or complex pipeline; complexity increases as more processing steps, integrations, or conditional logic is added.

Location

Latency is impacted by where AI runs. All things being equal, the closer the AI processing happens to the data source, the lower the latency, because remote processing can introduce delays due to network communication and data transfer times. Also, the more intermediators something has to pass through, the more latency it experiences.[3] AI, however, can live at any potential location. AI typically will run in one of three different categories or locations. Not every application has three or even two, but for illustrative purposes, consider Figure 1-6, which illustrates an idealized hierarchy comprising devices, edge, and centralized processing for AI distribution.

2 Kayly Lange and Laiba Siddiqui, "Data Pipelines and Optimizing Pipeline Efficiency," Splunk blog, September 19, 2024, *https://www.splunk.com/en_us/blog/learn/data-pipelines.html*.

3 "Real-Time AI Inference Latency Analysis," Restack.io, accessed March 2, 2025, *https://www.restack.io/p/real-time-ai-inference-answer-latency-analysis-cat-ai*.

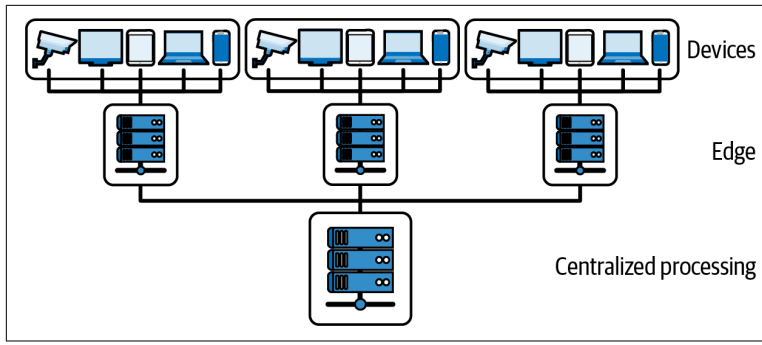

Figure 1-6. Device, edge, and centralized processing

The hierarchy in Figure 1-6 is described as follows:

On-device AI
: This eliminates network delays, achieving near-zero latency by processing data directly on hardware such as smartphones, IoT sensors, or embedded systems.[4] This allows for instant decision making in real-time applications like voice recognition, AI-powered camera filters, and drone navigation. However, the trade-off is limited processing power and energy efficiency, thus making it unsuitable for complex deep learning models or computationally heavy tasks.[5] Many consumer-oriented surveillance applications do on-device AI with object detection.

Edge AI
: This reduces latency by processing data on nearby computing infrastructure, such as edge appliances, routers, or gateways. While it balances speed and scalability better than on-device AI, it requires investment in localized computing resources and careful management of distributed workloads to avoid bottlenecks. Edge AI hardware—such as NVIDIA Jetson, Google Coral, and Intel Movidius—provides efficient inference for applications that require low latency.[6] In the customer service example, edge AI could leverage localized AI for call centers if

4 "Understanding On-Device AI: Benefits and Applications," 8allocate blog, September 13, 2023, *https://8allocate.com/blog/understanding-on-device-ai-benefits-and-applications*.

5 "The Key Benefits of On-Device AI," Deloitte, accessed March 2 2025, *https://www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/on-device-ai-key-benefits.html*.

they are geographically distributed. In this case, the call centers have agents working on a local network, with AI applications running on edge systems to perform the enrichments as part of the real-time streams.

Centralized AI

This provides the highest computational power because it offers large-scale data processing for AI, such as the cloud. Advanced strategies such as tensor parallelism and pipeline parallelism, commonly used in large language models (LLMs), take advantage of multi-GPU execution that improves latency in centralized AI.

However, it introduces increased latency due to the time required for data to travel across networks to centralized data centers and back. Additionally, network congestion and bandwidth limitations can further impact latency, making cloud-dependent AI unreliable in environments with unstable connectivity.[7] Modern workforces nowadays, though, leverage remote workers for customer service agents. In these cases, centralized cloud AI makes more sense because of the wide geographic distribution of such a workforce.

Hybrid AI

This combines the advantages of local and cloud processing, optimizing latency by ensuring that immediate tasks run on-device or at the edge while offloading computationally intensive processes to the cloud.[8] While hybrid AI reduces overall latency, it introduces complexity in determining which tasks should be processed locally versus remotely.

Note that shifting AI workloads between different points can improve latency by taking advantage of more powerful compute hardware. The net savings reduces the total amount of time. Also,

---

6 "Edge AI Hardware Comparison," Restack.io, accessed March 2, 2025, *https://www.restack.io/p/edge-ai-answer-hardware-comparison-cat-ai*.

7 Salvatore Salamone, "How Real-Time Decisions at the Edge Avoid Critical Latency Problems," *RTInsights*, September 25, 2024, *https://www.rtinsights.com/how-real-time-decisions-at-the-edge-avoid-critical-latency-problems*.

8 Deval Shah, "Balancing the Cloud and the Edge: A Close Look at Enabling Hybrid AI," *Wevolver*, September 21, 2023, *https://www.wevolver.com/article/balancing-the-cloud-and-the-edge-a-close-look-at-enabling-hybrid-ai*.

distributed AI—where AI runs at multiple points—can have a net reduction in latency by spreading the AI workload across a fleet of devices.

Multistep processing

Distributed systems often use multistep pipelines where data undergoes several transformations before delivery. While this enables richer analytics, it increases latency, as each step depends on the timely completion of the previous one. Complex operations can cause bottlenecks, especially with large datasets. Reducing the number of steps is key to minimizing latency in real-time applications.

The service agent system uses several different approaches with AI. Some of them are more algorithmic, while others leverage embedding models and generative AI models for the RAG components.

Atomic Transactions and Microbatches Versus Bulk Processing

Data processing strategies vary in granularity, thus affecting how AI models analyze and act on incoming data. Atomic transactions and microbatches are fundamental to real-time AI, ensuring that each event is processed independently or in small, sequential groups. This enables AI to react continuously to individual transactions, such as detecting fraud in a single credit card purchase or adjusting a recommendation system based on a user's latest interaction.[9] By contrast, bulk processing is computationally efficient for long-term analysis, but it lacks the ability to respond dynamically to live changes.[10]

For stream processing, focus on data simplicity, as not all data points require deep analysis, and not every event needs to be processed at the same level of detail:

9 "Real-Time Data Processing Versus Micro-Batch Processing," CloverDX blog, April 26, 2021, *https://www.cloverdx.com/blog/real-time-data-processing-versus-micro-batch-processing*.

10 "Stream Processing vs. Batch Processing: A Comparative Analysis of Their Key Benefits and Limitations," Edge Delta blog, June 5, 2024, *https://edgedelta.com/company/blog/stream-processing-vs-batch-processing*.

- Before data enters the system, unnecessary or redundant inputs can be discarded, allowing only the most relevant events to be processed.[11]
- Instead of processing full data payloads, selecting only the essential attributes for real-time AI models minimizes processing time.[12]
- Some decisions can be made at the edge or closer to the data source, reducing the need for centralized processing.

In addition to simplifying the data, a system should be capable of handling a constant influx of data without bottlenecks. This requires distribution such that workloads are spread across multiple computing resources. However, distribution comes with its own trade-offs: the more spread out the system becomes, the more effort is required to manage synchronization, consistency, and fault tolerance. The goal is to distribute workloads intelligently so that no single node is overwhelmed, while also avoiding unnecessary duplication of processing:

- Efficient routing mechanisms ensure that only the necessary components process specific events, preventing wasted computation. (AI can control routing through a concept known as *agentic AI*. More on that in Chapter 4.)
- Even distribution prevents some nodes from being overburdened while others remain idle.
- Where possible, processing should avoid retaining excessive state, reducing dependencies between distributed components.

Simpler Computations Versus Complex Computations

Real-time AI prioritizes speed, using lightweight, pretrained models for instant decisions with minimal computation. Batch processing handles heavy tasks like deep learning on large datasets, producing deeper insights but requiring more time and resources. For example, spam filters classify emails in real time, but training those models

11 Nikolaj Buhl "Mastering Data Cleaning and Data Preprocessing," Encord, August 9, 2023, *https://encord.com/blog/data-cleaning-data-preprocessing*.

12 "Data Preprocessing: Artificial Intelligence Explained," Netguru S.A., accessed March 3, 2025, *https://www.netguru.com/glossary/data-preprocessing*.

requires batch processing. AI's complexity drives its computational demands, affecting latency and scalability.[13] That said, the following is a tiered view of computational complexity; it is not a strict ranking of speed, but a relative view of computational complexity based on the time required to complete tasks:

Rule-based and statistical methods
: This involves using predefined logic or basic statistical relationships to make decisions. These methods are fast and lightweight and do not involve learning. They are used in email filtering rules, fraud detection thresholds, thermostat controls, and simple quality checks in manufacturing.

Basic machine learning
: This entails learning patterns from structured data using simple algorithms. It is suitable for small-scale tasks and requires minimal processing power. It is used in credit scoring, customer churn prediction, basic product recommendations, and sentiment analysis on small datasets.

Advanced machine learning
: This uses more complex structures and ensemble methods. It requires more data and processing but remains practical for most modern systems. It is used in loan approval systems, medical diagnostics (e.g., predicting disease risk), stock price prediction, and targeted marketing optimization.

Deep learning models
: This involves using multilayered neural networks to capture complex patterns in data. They are computationally intensive and typically require GPUs or specialized hardware. They are used in image and speech recognition, real-time

13 "Computational Complexity Theory," Autoblocks, accessed March 3, 2025, *https://www.autoblocks.ai/glossary/computational-complexity-theory*; Vijay S. Agneeswaran, "Computational Complexity of Deep Learning: Solution Approaches," *Medium*, May 27, 2021, *https://medium.com/walmartglobaltech/computational-complexity-of-deep-learning-a-birds-eye-view-2250b7c098a1*; Paritosh Kumar, "Computational Complexity of ML Models," *Medium*, December 14, 2019, *https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770*; and Rayan Yassminh, "Time Complexity in ML: Ensuring Model Performance with Growing Data Sizes," *Medium*, November 1, 2024, *https://medium.com/@ryassminh/time-complexity-in-ml-ensuring-model-performance-with-growing-data-sizes-42007e5b7305*.

language translation, autonomous driving perception systems, and advanced chatbots.

Frontier AI
: This is the most advanced and resource-intensive category. These models involve massive datasets, distributed computing, and high-performance hardware. They drive breakthroughs in AI capabilities and are used in robotics control systems, generative AI for images/text/code (e.g., DALL-E, ChatGPT), cross-modal search, adaptive gameplay in video games, and autonomous scientific discovery.

  As AI advances, balancing complexity, scalability, and real-time needs is key. Simple models offer lower latency and better scalability, while complex models yield deeper insights but demand more resources. Developers should choose task-optimized methods, favoring rules or lightweight models like Naive Bayes or *k*-nearest neighbors (KNN) over resource-heavy LLMs when possible.

Minimal Use of Persisted Data Versus Heavy Data Persistence

Data storage architecture directly influences the performance and scalability of AI systems. Real-time AI uses in-memory processing with minimal data persistence, handling events quickly and discarding them to maintain low latency. Batch AI relies on persistent storage, processing large, historical datasets for deeper analysis and model training.

Streaming systems must balance speed with minimal retention, sometimes using summaries or short-lived states. Batch systems require scalable, high-performance storage to manage massive data volumes. Regardless of the approach, AI systems must retain only essential data while supporting learning and improvement without overwhelming storage resources. Here are some tips for making the most of persistent memory performance:

- For ephemeral memory, one approach is short-term state retention, which provides fast access to recent data without long-term storage overhead. In-memory caching solutions, such as Redis or Memcached, allow embeddings or AI processing results to be stored temporarily for rapid retrieval. For AI

applications that rely on fast similarity searches, approximate nearest neighbor (ANN) indexing structures embeddings in memory.

- Beyond short-term retention, summarization techniques help AI systems retain useful insights while reducing storage demands. Simpler methods like rolling statistical summaries capture trends over time while discarding granular records.
- For tasks that require the persistent storage of embeddings or AI-generated insights, consider using vector databases—such as Milvus, FAISS, or Weaviate—for storing and indexing embeddings.[14] When dealing with continuously generated AI data, time-series databases such as InfluxDB or TimescaleDB provide a structure that efficiently handles high-speed read and write operations and have integrated mechanisms for inputs and outputs that allow them to integrate as part of pipelines.[15]

Many streaming systems may not even need persisted data for analysis, but some might. The agent example leverages both the historical context of the conversation and a broader library of historical resources to provide enriched, contextually relevant feedback as the conversation unfolds. Table 1-2 summarizes the trade-offs between streaming and batch. (Note that data processing is not always one or the other.)

Table 1-2. Summary of real-time versus batch processing

	Streaming data	Batch data
Latency	Milliseconds to seconds; enables immediate insights and action	Minutes to hours or longer; insights are delayed
Processing mode	Continuous, event by event, or microbatch	Periodic, large batches at set intervals
Pipeline complexity	Simpler, optimized for speed; minimal steps	Can be more complex, with multiple transformations and integrations
Computation	Lightweight, pretrained models for speed; simple algorithms	Heavy, deep learning or complex analytics for depth

14 Mike Vincent, "Which Vector DB Should You Choose?," *Medium*, October 28, 2024, *https://mike-vincent.medium.com/which-vector-db-should-you-choose-cee2a89e0939*.

15 "System Properties Comparison InfluxDB vs. Quasardb vs. TimescaleDB," DB-Engines, accessed March 6, 2025, *https://db-engines.com/en/system/InfluxDB%3BQuasardb%3BTimescaleDB*.

	Streaming data	Batch data
Location of processing	On-device, edge, centralized, or hybrid; closer to data source for lower latency	Typically centralized in data centers or the cloud
Granularity	Atomic transactions or microbatches; per event	Bulk processing of large datasets
Data storage	In-memory, minimal persistence; short-lived state	Persistent storage; retains historical records
Actionability	Enables real-time decision making and automation	Supports deep analysis, reporting, and long-term trends
Scalability	Scales for high data velocity and immediate response	Scales for high volume and complex aggregation
Trade-offs	Lower latency but may sacrifice analytical depth	Higher latency but richer insights
Example use cases	Fraud detection, personalized recommendations, real-time monitoring	Periodic reporting, model training, historical trend analysis

CHAPTER 2

Real-Time Data as Part of Application Architecture

Modern applications operate in dynamic data ecosystems, requiring real-time processing and action. Traditional synchronous models struggle with growing data and latency demands, leading to the rise of real-time data pipelines. These pipelines enable asynchronous communication, high-throughput event handling, and consistency across distributed systems. Key architectures include microservices, modular monoliths, and serverless. Each uses real-time data effectively, but with distinct trade-offs.

Microservices Architecture

Microservices architecture is a design paradigm that breaks an application into a collection of small, independent services, each responsible for a specific function.

Services in a microservices suite have traditionally communicated over a network using lightweight protocols like HTTP or gRPC. This kind of communication has relied on synchronous request-response mechanisms with direct service-to-service calls, but it can create bottlenecks, thereby increasing complexity and reducing resilience. To address these challenges, many organizations turn to message brokers to communicate asynchronously through messaging and event streams rather than direct API calls. With

event-driven messaging, services operate autonomously, processing events as they arrive. The backbone of streaming pipelines is built on event streaming platforms like Apache Kafka, Redpanda, Apache Pulsar, Google Pub/Sub, and solutions from cloud service providers, such as Azure Event Hubs and Amazon Kinesis.[1]

Two common application patterns are used with microservices to manage the decoupling through brokers and are often used together: *event sourcing* and *command query responsibility segregation* (*CQRS*).

CQRS separates read and write operations into distinct models so that each is optimized for its specific workload. Instead of forcing a single database to handle everything, CQRS introduces two specialized components. The first is the *command model*, which is responsible for processing writes and updates. This model often works in tandem with event sourcing, where changes are stored as immutable events rather than direct database updates. The second is the *query model*, which is optimized for fast and efficient reads.[2] Figure 2-1 depicts the CQRS application pattern.

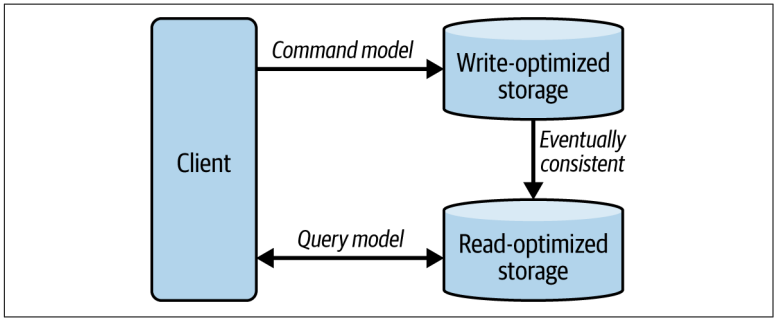

Figure 2-1. CQRS

1 "gRPC vs Message Broker for Microservices," *GeeksforGeeks*, last modified July 23, 2025, *https://www.geeksforgeeks.org/system-design/grpc-vs-message-broker-for-microservices/*.

2 "CQRS Pattern," *Microsoft Learn*, February 21, 2025, *https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs*.

Event sourcing works with CQRS. With event sourcing, every change to an application's state is recorded as an immutable event, instead of updating and overwriting records in a database. These events form an append-only log (much like a ledger) of historical record of everything that has happened in the system.[3]

AI integrates and enhances CQRS and event sourcing in a few different ways:

- AI-powered analytics can process the historical event logs stored in an event-sourced system to detect patterns, anomalies, and trends that inform business decisions.
- In CQRS, AI models can optimize the query model by dynamically generating predictive insights and personalizing data retrieval based on user behavior.
- AI-driven automation can enhance data pipelines by intelligently prioritizing events, reducing noise, and ensuring that critical updates propagate efficiently through the system.
- Machine learning models can also improve resilience by predicting potential failures or bottlenecks in the microservices workflow.

Adopting microservices brings challenges. This is especially the case with CQRS and event sourcing, which introduce eventual consistency, meaning that parts of the system may not always align. Ensuring message ordering and deduplication is critical, as out-of-order or duplicate events can cause errors in fast-paced systems. Observability is also complex, requiring tools like OpenTelemetry or Jaeger to trace events across services. Operationally, microservices demand robust service discovery, API gateways, distributed logging, and seamless deployments to handle frequent updates. Despite this, microservices excel in scalable, high-availability environments like

3 Mehmet Ozkaya, "Event Sourcing Pattern in Microservices Architectures," *Medium*, September 8, 2021, *https://medium.com/design-microservices-architecture-with-patterns/event-sourcing-pattern-in-microservices-architectures-e72bf0fc9274*; and Chris Richardson, "Pattern: Event Sourcing," *Microservices.io*, accessed March 4, 2025, *https://microservices.io/patterns/data/event-sourcing.html*.

web apps, cloud native systems, and software-as-a-service (SaaS) platforms.[4]

Serverless Architecture

In a serverless architecture, applications are composed of small, stateless functions that execute on demand, typically in response to external events such as HTTP requests, database changes, or message queue events.

This approach often relies on tightly integrated cloud services provided by vendors such as AWS Lambda, Azure Functions, and Google Cloud Functions. These platforms automatically manage computing resources by scaling them up or down as required.[5]

As serverless apps grow more complex, especially with real-time data and AI needs, serverless functions alone fall short. Their stateless, ephemeral nature limits data persistence and coordination. Streaming pipelines extend serverless by providing a persistent, scalable way to manage event flows, enabling reliable, sequential, and asynchronous data processing.[6] Integrating streaming pipelines with serverless architectures enables seamless handling of unpredictable workloads. Streaming buffers traffic spikes, while serverless functions auto-scale to process data in parallel, allowing for real-time analysis and responsive actions—such as triggering events on anomalies—without overloading traditional systems.[7]

---

4 Matt Tanner, "Ten Common Microservices Anti-Patterns and How to Avoid Them," vFunction blog, February 4, 2025, *https://vfunction.com/blog/how-to-avoid-microservices-anti-patterns*; and Erick Zanetti, "Microservices Architecture: Principles, Patterns, and Challenges for Scalable Systems," *Medium*, March 4, 2025, *https://medium.com/@erickzanetti/microservices-architecture-principles-patterns-and-challenges-for-scalable-systems-9eac65b97b21*.

5 "Serverless Architecture," *GeeksforGeeks*, last updated July 23, 2025, *https://www.geeksforgeeks.org/serverless-architectures*.

6 "What Is a Serverless Data Pipeline: A Comprehensive Guide," Airbyte, November 19, 2024, *https://airbyte.com/data-engineering-resources/serverless-data-pipeline*.

7 "Real-Time Streaming Data Architectures That Scale," Tinybird blog, April 24, 2025, *https://www.tinybird.co/blog-posts/real-time-streaming-data-architectures-that-scale*; and Lucas Rettenmeier and Kirill Bogdanov, "Build a Near Real-Time Data Aggregation Pipeline Using a Serverless, Event-Driven Architecture," *AWS Database Blog*, November 1, 2021, *https://aws.amazon.com/blogs/database/build-a-near-real-time-data-aggregation-pipeline-using-a-serverless-event-driven-architecture*.

Despite its advantages, serverless computing, and its natural integration with pipelines, comes with several trade-offs:

Cold start latency
: Since serverless functions do not run continuously, they may experience a delay when spinning up after inactivity. This can impact real-time processing.

Execution time limits
: Most serverless platforms enforce strict time limits on function execution, making them unsuitable for long-running processes without careful orchestration.

Observability and debugging
: The ephemeral nature of serverless functions makes logging, tracing, and debugging more complex compared to traditional applications. Developers often rely on managed observability tools to track event flows across distributed pipelines.

Despite these challenges, serverless architecture combined with streaming pipelines provides a powerful and scalable approach to building event-driven applications.[8]

One of the most impactful AI integrations is in intelligent scaling and routing. Machine learning models analyze historical workloads and predict traffic surges. The output from the models will prewarm instances and mitigate cold start latencies. AI also enhances observability in serverless architecture by looking at intelligent log analysis and anomaly detection, automatically identifying performance bottlenecks or failures across distributed event-driven pipelines. Additionally, AI-driven orchestration optimizes workflow execution by dynamically adjusting function triggers, event priorities, and message routing based on changing data patterns.

In a serverless architecture, the agent example is implemented as an independent function triggered by events. All components are connected via an event-driven system, such as message queues and event streams powered by products like Redpanda, to ensure scalability and cost-efficiency while minimizing infrastructure management.

8 Softvery Solutions, "Serverless Functions and Cloud Functions for Event-Driven Architecture," *Medium*, July 17, 2024, *https://medium.com/@softverysolutions/serverless-functions-and-cloud-functions-for-event-driven-architecture-62dc82dac70b*.

CHAPTER 3
Real-Time AI in Industries

AI is already impacting businesses in dramatic ways. The use cases that follow all share a general pattern: they identify data and collect data for real-time AI, which involves capturing relevant inputs. Once the data is collected, the AI models and processing techniques are selected, ranging from machine learning algorithms and deep neural networks to rule-based systems and reinforcement learning, depending on the complexity of the analysis required. The AI system then generates outputs, which may take the form of predictive analytics, automated decisions, or adaptive responses that modify system behavior in real time. All of these use cases have real-world business impacts that show how AI transforms data into actionable intelligence, optimizing processes and driving innovation.

The use cases follow the same basic pattern outlined in the opening section of the report, with a producer, broker, and consumer. Chapter 1 explained how the customer service agent system follows this basic pattern, but in general it goes as follows:

1. The *producer* identifies and streams relevant data; AI may filter noise to focus on significant events.
2. The *broker* performs real-time analysis, transforming, enriching, and applying AI techniques (e.g., natural language processing [NLP], computer vision, predictive analytics) for deeper insights.

3. The *consumer* receives the results, which may be an AI system (e.g., chatbot, fraud detection) or a human operator; results can also be stored for future use.

Whether driving immediate action or informing future analysis, real-time AI enables organizations to respond more quickly, make smarter decisions, and gain deeper insights from their data streams.

Finance (Algorithmic Trading)

Real-time AI processes live market data streams and makes trading decisions at speeds far beyond human capabilities. In financial markets, timely insights can mean the difference between profit and loss, making high-frequency trading (HFT) and quantitative strategies heavily reliant on AI-powered data streams. The core use case is to automate trading decisions with precision and minimal latency, allowing traders to capitalize on fleeting opportunities while managing risk dynamically.[1]

The primary data sources for real-time AI in trading come from multiple financial and economic channels, such as price feeds from exchanges that supply up-to-the-millisecond updates on stock, foreign exchange (forex), cryptocurrency, and commodity prices. Order-book data captures bid-ask spreads, liquidity changes, and market depth. Macroeconomic indicators, central bank announcements, and corporate earnings reports influence market sentiment, while news streams and social media sentiment analysis help gauge investor reactions to events.

With these data sources, AI-driven trading relies on a combination of deep learning, reinforcement learning, and NLP models to extract insights.[2] Recurrent neural networks (RNNs) and long short-term

1 Leo Mercanti, "AI for High-Frequency Trading: The Hidden Engines Behind Lightning-Fast Market Decisions," *Medium*, September 20, 2024, *https://leomercanti.medium.com/ai-for-high-frequency-trading-the-hidden-engines-behind-lightning-fast-market-decisions-e0a571cc6a03*.

2 Jesse Anglen, "Deep Reinforcement Learning: Definition, Algorithms, and Uses," Rapid Innovation, accessed March 6, 2025, *https://www.rapidinnovation.io/post/deep-reinforcement-learning-definition-algorithms-uses*; and Guanghe Cao, Yitian Zhang, Qi Lou, and Gaike Wang, "Optimization of High-Frequency Trading Strategies Using Deep Reinforcement Learning," *Journal of Artificial Intelligence General Science* 6, no. 1 (November 5, 2024): 231–245, *https://doi.org/10.60087/jaigs.v6i1.247*.

memory (LSTM) are commonly used for time-series analysis, capturing historical patterns to predict future trends. Transformers, initially for NLP, are now also used in financial forecasting for their strength in handling long-range dependencies. Reinforcement learning techniques allow AI agents to optimize trading strategies dynamically based on trial and error, and NLP models analyze sentiment from news headlines, earnings reports, and financial statements to assess the impact of market events on asset prices. The fusion of these AI techniques enables more accurate predictions and adaptive trading strategies.

After processing, the AI generates actionable trading signals, risk assessments, and portfolio adjustments. These outputs include buy/sell signals for automated execution, alerts on market anomalies, volatility estimates, and liquidity forecasts. The AI continuously refines these outputs by optimizing trade execution through adjustments in order timing, sizing, and pricing to minimize market impact. It also provides real-time portfolio analytics by assessing risk exposure and recommending hedging strategies to mitigate potential losses.[3]

Implementing real-time AI in trading can significantly enhance decision making, profitability, and risk management by reducing latency in trade execution.

Cybersecurity (Real-Time Threat Detection)

Unlike traditional rule-based security solutions that rely on predefined signatures of known threats, AI-powered systems use machine learning to establish baselines of normal network behavior and identify deviations that might signal an attack. Real-time AI-powered cybersecurity monitoring systems continuously analyze network traffic and user activity to detect and respond to threats as they emerge. The core use case of this stream is proactive threat detection and mitigation. This stream enables organizations to monitor millions of network events per second; identify threats like zero-day exploits, unauthorized access attempts, or insider threats; and trigger immediate response actions.

3 "Harnessing AI in the Stock Market: An Overview," Newo.ai, accessed March 6, 2025, *https://newo.ai/insights/harnessing-ai-in-the-stock-market-an-overview*.

The AI-driven cybersecurity system ingests multiple data sources in real time. These sources include network packets, firewall logs, intrusion detection system (IDS) alerts, user authentication logs, cloud activity logs, endpoint security events, and DNS queries. Additionally, behavioral telemetry from devices and users—such as login times, file access patterns, and data transfer volumes—feeds into AI models to distinguish between legitimate activity and potential threats. AI also integrates threat intelligence feeds, which provide information on emerging attack patterns, known malicious IPs, and vulnerabilities.

Various AI techniques and models are employed to analyze real-time security data effectively. Machine learning models, including supervised and unsupervised learning, are used to classify threats and detect anomalies. Deep learning models enhance malware detection by learning to recognize malicious patterns within network payloads. NLP models can analyze phishing emails and suspicious messages for social-engineering attacks. Reinforcement learning models continuously improve security responses by adapting to new threats over time. Additionally, graph-based AI models map relationships between entities in a network, identifying the lateral movement patterns of attackers.[4]

The output of the AI-powered security stream includes real-time alerts, risk scores, and automated response actions. AI assigns risk scores to detected anomalies so that analysts can prioritize their responses. In automated systems, the AI stream can trigger security measures such as isolating compromised devices, blocking suspicious IP addresses, or enforcing multifactor authentication when an account exhibits unusual behavior.[5]

4 Youssef Singer, "AI and Machine Learning in Cybersecurity," Bachelor's thesis (German University in Cairo, 2024), *https://www.researchgate.net/publication/380889139_AI_Machine_Learning_in_Cybersecurity*; and Xianghui Meng, "Advanced AI and ML Techniques in Cybersecurity: Supervised and Unsupervised Learning, and Neural Networks in Threat Detection and Response," *Applied and Computational Engineering* 82 no. 1 (July 26, 2024): 24–28, *https://doi.org/10.54254/2755-2721/82/2024GLG0054*.

5 Lizzy Ofusori, Tebogo Bokaba, and Siyabonga Mhlongo, "Artificial Intelligence in Cybersecurity: A Comprehensive Review and Future Direction," *Applied Artificial Intelligence*, 38(1), *https://doi.org/10.1080/08839514.2024.2439609*.

AI-driven cybersecurity delivers faster threat detection and quicker response, improving efficiency by filtering false positives and prioritizing real threats. This allows security teams to focus on critical incidents. Automated mitigation reduces downtime and financial risks from ransomware, fraud, and compliance issues, strengthening overall security.

AdTech (Personalization and Campaign Performance)

Digital advertising relies on real-time AI to deliver highly personalized ads and optimize campaign performance dynamically. The core use case of streaming data in this context is to enable instant decision making for ad selection, bidding, and placement. Real-time AI ensures that ads remain relevant, timely, and engaging while maximizing return on investment (ROI) for advertisers.[6] Streaming data enables automated campaign adjustments, such as reallocating budgets and shifting bidding strategies, based on evolving user interactions and market conditions.

The AI-driven ad selection process relies on a continuous stream of user interaction data collected from multiple sources. This includes behavioral data such as website visits, search queries, purchase history, and engagement with previous ads, clicks, page views, and past interactions, to determine the most relevant ads. Contextual data, such as keywords on a web page or the type of content being viewed, is also incorporated to enhance ad relevance. Data is sourced from ad exchanges, demand-side platforms, and supply-side platforms, creating a real-time flow of information across the advertising space. Privacy-compliant identifiers, such as hashed user profiles or contextual signals, help AI models make informed decisions while respecting user privacy regulations like the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA).

6 Megan Graham, "Taco Bell and KFC's Owner Says AI-Driven Marketing Is Boosting Purchases," *Wall Street Journal*, November 15, 2024, *https://www.wsj.com/articles/taco-bell-and-kfcs-owner-says-ai-driven-marketing-is-boosting-purchases-ab3a5f36*.

A variety of AI techniques and models power real-time advertising decisions on the data sources:[7]

- Recommendation systems, often based on collaborative filtering and deep learning models, predict the most relevant ads for individual users by analyzing past interactions.
- Reinforcement learning models optimize bidding strategies in real time, learning from each auction outcome to improve future decisions.
- NLP models analyze web page content to ensure contextual ad placement.
- Computer vision models analyze video frames or images to match ads with relevant visual content.
- Predictive analytics models, often using logistic regression or gradient boosting, assess the likelihood of user engagement (clicks, conversions) and adjust ad targeting accordingly.

The AI-powered stream outputs real-time decisions that determine ad selection, placement, and bidding values. In a user session, the system instantly returns an optimized ad choice tailored to that user's behavior and context. In the case of programmatic advertising, it outputs bid recommendations for each available ad impression, ensuring that advertisers compete effectively for the most valuable placements. Additionally, real-time analytics and performance insights flow from the stream, providing advertisers with up-to-the-second data on impressions, clicks, conversions, and audience engagement trends.[8]

Integrating real-time AI into AdTech results in smarter, data-driven advertising strategies that drive revenue growth and competitive advantage by improving ad relevance, increasing user engagement, and optimizing ad spending. The automated decision-making

7 "AI in Digital Marketing - The Ultimate Guide," Digital Marketing Institute blog, April 14, 2025, *https://digitalmarketinginstitute.com/blog/ai-in-digital-marketing-the-ultimate-guide*; and "7 machine learning algorithms for recommendation engines," lumenalta, January 28, 2025, *https://lumenalta.com/insights/7-machine-learning-algorithms-for-recommendation-engines*.

8 Mar Fernández Parra, "The Future of Personalization with Gen AI and Real-Time Data," *Medium*, February 18, 2024, *https://medium.com/@MarBlue-Bucket.AI/trully-real-time-personalization-in-genai-times-c4bd0bae3416*.

process ensures that advertisers maximize ROI by bidding only on impressions likely to convert, reducing wasted ad spending. AI-driven personalization enhances user experience by displaying relevant ads, leading to higher engagement and improved brand perception. The ability to adjust campaigns in real time allows marketers to react swiftly to changing consumer behaviors and market trends, ensuring sustained effectiveness. Predictive insights help advertisers allocate budgets more efficiently by prioritizing high-performing audience segments and content channels. The net gains from these different streams create highly optimized advertisements for users.

Manufacturing (Predictive Maintenance)

Real-time AI in industrial operations is used for predictive maintenance, ensuring that equipment runs smoothly by detecting failures before they cause costly downtime. IoT sensors continuously stream data from factory machines, capturing metrics like temperature, vibration, pressure, and voltage. Instead of relying on scheduled maintenance or reacting to failures after they occur, AI analyzes this data in real time to identify subtle changes that indicate wear and tear or malfunctions.

The effectiveness of real-time AI depends on a continuous influx of data from industrial IoT (IIoT) sensors embedded in machinery. These sensors monitor key operational parameters such as temperature fluctuations, vibration patterns, pressure levels, motor efficiency, and electrical performance. This high-frequency, time-series data is transmitted to AI-driven analytics platforms, where machine learning models process it instantaneously. Some systems also integrate historical maintenance records, environmental conditions, and operational logs to improve predictions. By aggregating data from multiple sources, AI can establish normal operating baselines and detect deviations that signal potential failures, allowing maintenance teams to intervene before serious issues arise.[9]

9 Sai Surya Mounika Dandyala, Vinod Kumar Karne, and Parameshwar Reddy Kothamali, "Predictive Maintenance in Industrial IoT: Harnessing the Power of AI," *International Journal of Advanced Engineering Technologies and Innovations* 1, no. 3 (2020), 1–21, *https://www.researchgate.net/publication/384295658_Predictive_Maintenance_in_Industrial_IoT_Harnessing_the_Power_of_AI*.

A combination of AI techniques processes the real-time data to provide accurate predictions and actionable insights:[10]

Time-series forecasting
: AI models (often using RNNs or other forecasting algorithms) project sensor readings into the future to predict when they might cross a failure threshold. For example, a model might forecast that a motor's vibration level will exceed safe limits in two days, indicating a probable failure if not serviced.

Anomaly detection
: The AI establishes normal operating ranges for equipment performance and flags anomalies as soon as they emerge. If a normally stable temperature starts fluctuating unpredictably or a pump's output pressure drops suddenly, the system recognizes these as out-of-profile behaviors.

Sensor fusion
: Data from multiple sensors on a machine is combined to improve diagnostic accuracy. For instance, an increase in vibration taken together with a rise in motor temperature and a drop in output efficiency forms a clearer picture of an impending fault than any one sensor alone could. AI models merge these signals to reduce false alarms and pinpoint the true cause (e.g., a misaligned shaft or impending bearing failure).

The real-time AI system generates multiple types of outputs that drive decision making. When a potential issue is detected, the system can produce alerts or notifications for maintenance personnel, detailing the nature of the anomaly and its predicted impact. It can also generate prioritized maintenance schedules, identifying which machines require immediate attention. In advanced implementations, AI can trigger automated responses, such as adjusting machine parameters, shutting down malfunctioning equipment to prevent damage, or dynamically updating maintenance workflows.[11]

---

10 Angel Jaramillo-Alcazar, Jaime Govea, and William Villegas-Ch, "Anomaly Detection in a Smart Industrial Machinery Plant Using IoT and Machine Learning," *Sensors* 23, no. 19 (2023), *https://www.mdpi.com/1424-8220/23/19/8286*.

11 Jesse Anglen, "AI in Anomaly Detection for Businesses," *Rapid Innovation*, accessed March 6, 2025, *https://www.rapidinnovation.io/post/ai-in-anomaly-detection-for-businesses*.

For manufacturing, real-time AI transforms industrial maintenance from a reactive expense into a strategic advantage, optimizing productivity and reducing waste. Implementing real-time AI for predictive maintenance enhances operational efficiency and cost savings by preventing unplanned downtime—factories can avoid production halts that lead to revenue loss. Additionally, AI-driven prioritization ensures that maintenance resources are allocated more efficiently, and proactive maintenance extends the lifespan of machinery.[12]

Gaming (Personalization and Player Analytics)

Real-time AI in video games uses continuous data streams to dynamically adapt gameplay to create an immersive and personalized player experience. The primary use case is to analyze player behavior in real time and modify game elements accordingly. This includes dynamic difficulty adjustment (DDA), where AI monitors performance metrics (such as reaction time, accuracy, and success rate) and modifies in-game challenges to keep players engaged. Additionally, AI-powered personalization tailors content to individual playing styles, such as prioritizing exploration-based missions for those who favor open-world discovery or adjusting enemy behavior based on combat tendencies.

Such AI streams rely on multiple sources of real-time data generated by player interactions within the game. Every action—such as movement patterns, time spent on objectives, choices made, win/loss records, and response times—is captured and fed into the AI system. Data from input devices (keyboard, controller, mouse movements) helps assess reaction times and precision, while telemetry from the game engine tracks in-game progress, player choices, and combat efficiency. Multiplayer environments further enhance data streams by incorporating player-versus-player interactions,

12 Magda Dąbrowska, "Power of Predictive Maintenance with IoT: Reducing Downtime and Costs," *IoT Now*, accessed March 6, 2025, *https://www.iot-now.com/2024/10/23/147629-power-of-predictive-maintenance-with-iot-reducing-downtime-and-costs*.

teamwork dynamics, and communication patterns. AI models process this vast dataset in real time, looking for trends that indicate player skill levels, frustration points, or engagement drops.[13]

Several AI techniques are employed to analyze and respond to the incoming data stream:

- Machine learning models, including deep neural networks, predict player behaviors and make real-time recommendations.
- Reinforcement learning is often used to power adaptive nonplayer characters (NPCs) that learn from player strategies and modify their behavior dynamically.
- NLP may also be applied in narrative-driven games to generate responsive dialogues that reflect player choices.
- Rule-based AI, often seen in game directors (the part of the program that adjusts pacing, difficulty, tension, or events in real time based on the player's actions, performance, or situation), operates alongside machine learning models to enforce predefined difficulty-scaling rules and ensure smooth game progression.
- Clustering algorithms segment players based on behavioral patterns so that AI can tailor content or offer incentives for engagement.

The AI-driven stream generates multiple forms of output that directly impact gameplay. In the simplest form, it modifies game variables such as enemy difficulty, puzzle complexity, and item availability based on real-time player data.

AI analysis aims to boost player engagement, retention, and satisfaction, directly affecting business outcomes. By enabling adaptive difficulty and personalized experiences, it prevents frustration and boredom. AI also optimizes monetization by recommending

13 Milijana Komad, "Product Design and Psychology: The Use of Dynamic Difficulty Adjustment in Video Game Design," *Medium*, August 12, 2023, *https://medium.com/@milijanakomad/product-design-and-psychology-the-use-of-dynamic-difficulty-adjustment-in-video-game-design-7a1e2d919b96*.

in-game purchases tailored to playing style. Real-time analytics help developers quickly refine level design and balance, while adaptive matchmaking improves fairness in multiplayer games, reducing player drop-off.[14]

14 Svitlana Varaksina and Ivan Dyshuk, "AI in Game Development: Analyzing Player Behavior," Mind Studios blog, January 23, 2024, *https://themindstudios.com/post/ai-in-analyzing-player-behaivor*; and "AI in Gaming: How AI Is Creating Personalized Gaming Experiences," Openfabric blog, July 26, 2024, *https://openfabric.ai/blog/ai-in-gaming-how-ai-is-creating-personalized-gaming-experiences*.

CHAPTER 4

Getting Started with Real-Time AI

Implementing real-time AI requires more than just technical capability; it demands a clear understanding of the business value it delivers. Real-time AI is not merely about speed but about making timely, data-driven decisions that enhance efficiency, enable automation, and improve responsiveness. Organizations must first define their objectives by asking:

- What real-time insights will create a competitive advantage?
- What specific problems can immediate data processing solve?
- How will AI improve decision making beyond traditional data pipelines?

Answering questions like these aligns real-time AI initiatives with business goals, because they focus on meaningful outcomes rather than simply adopting technology for its own sake. The following step-by-step guide provides a road map for building an effective real-time AI strategy.

Identifying Data for Real-Time AI for a Producer

In real-time AI, not all data is equally important. Only relevant data should be classified:

Event-triggering data
: These sources generate immediate-action events, such as sensor readings, user activity logs, social media reactions, financial transactions, or security alerts. They drive AI decisions.

Enrichment/contextual data
: These refine decisions by adding background—historical trends, customer profiles, knowledge graphs, or third-party APIs. Some need frequent updates (e.g., weather, stock prices), while others remain static.

AI filters noise, reducing unnecessary processing, and can automate data classification, identifying relevant patterns in structured and unstructured data to support real-time decisions.

Implementing a Real-Time Data Processing Pipeline as a Broker

Once event triggers and contextual data sources are defined, the next step is to establish a real-time data processing pipeline that rapidly transforms data into actionable insights. This pipeline must efficiently ingest, process, and analyze data to support timely decision making.

The first priority is efficient data ingestion, which requires an event-driven architecture capable of handling high-velocity data streams. Technologies such as Redpanda, Kafka, RabbitMQ, or real-time databases enable seamless message queuing, ensuring that incoming events are processed in an orderly and scalable manner. Next, AI-driven real-time transformations and enrichments refine the data, adding relevant context to enhance decision making. AI models analyze events using various techniques, such as:

- Anomaly detection to identify irregularities (e.g., fraudulent financial transactions)
- NLP to extract meaning from customer complaints, social media posts, or support tickets
- Predictive analytics to forecast demand in logistics, supply chains, or market trends
- Image and video analysis to detect defects in manufacturing, monitor security footage, or process medical imaging in real time

To ensure low-latency processing, AI models must be optimized for real-time inference. Technologies like edge AI enable on-device processing, reducing dependence on cloud latency, while lightweight models—such as quantized or distilled versions—maintain speed without sacrificing accuracy.

Delivering Processed Data to Consumers

Once data has been processed and enriched, it must be directed to the appropriate endpoint for action or storage. For many applications, automated AI-driven consumers act on processed data instantly.

However, some scenarios require human-in-the-loop decision making, where AI provides insights but ultimate decisions rest with human operators. Processed data can be visualized through dashboards, real-time alerts, or reports, allowing experts to assess the situation and take appropriate action. This approach is often used in fields like cybersecurity, health care diagnostics, and financial risk management, where human judgment is still crucial.

Additionally, data storage and feedback loops play an essential role in refining AI models over time. Not all processed data requires immediate action.

Emerging Applications for Real-Time AI

Artificial intelligence is rapidly evolving, and new approaches are emerging that push the boundaries of what machines can do. Agentic AI, real-time RAG, and the movement toward self-hosted models all seek to enable systems that act with intention, adapt

on the fly, and place control back in the hands of organizations. Their core ambition is to bridge the gap between static automation and true machine agency with AI systems that reason, reflect, and interact with their environments in fundamentally new ways.

Agentic AI

Agentic AI refers to autonomous, goal-oriented systems that make decisions and act without human input, using machine learning and LLMs like GPT. These systems interact with data sources, APIs, and other systems to make real-time, informed choices, adapting to changing conditions or deciding when to act. Unlike *nonagentic AI*, which performs fixed tasks based only on input and context, agentic AI can learn and adapt dynamically.[1]

Agentic AI is defined by its autonomy and reflective capabilities. It operates independently, adapting to changing conditions to achieve goals, and improves over time by learning from experience. Drawing from human psychology, agentic AI involves planning, action, memory, and reflection, allowing it to set objectives, create strategies, monitor outcomes, and refine future behavior.

These modules work together to give agentic AI the ability to act intentionally, adapt dynamically, and reflect on past outcomes to improve future performance.

Agentic AI can work in numerous applications. It operates independently in complex environments by continuously processing real-time data from sensors like cameras, LiDAR, radar, and ultrasonics. It identifies objects such as pedestrians, vehicles, traffic signs, and obstacles, constructing a dynamic three-dimensional map to maintain situational awareness and enable safe, autonomous decision making.[2]

Agentic AI in autonomous systems uses real-time sensor data to build a dynamic map of the environment, evaluating possible trajectories based on traffic rules, road conditions, and the behavior of other road users. It predicts movements to avoid collisions,

1 Vanna Winland, Jess Bozorg, and Cole Stryker, "What Is Agentic Architecture?," IBM, accessed March 7, 2025, *https://www.ibm.com/think/topics/agentic-architecture*.

2 "openpilot," *Wikipedia*, last modified April 1, 2025, *https://en.wikipedia.org/wiki/Openpilot*.

determines the optimal path by considering factors like congestion or obstacles, and sends precise commands to control acceleration, braking, and steering. The AI continuously monitors data to adjust in real time, maintaining lane discipline and speed limits, and executing maneuvers such as lane changes or turns. Through machine learning, it improves decision making by learning from each driving experience, adapting to varied conditions and enhancing performance. Companies like Waymo and NVIDIA have developed agentic AI platforms that enable autonomous vehicles to safely navigate urban environments with real-time processing and advanced decision-making capabilities.

Agentic AI is reshaping education by introducing agentic systems that dynamically adjust educational content to suit individual student needs by analyzing performance data in real time to tailor instruction. This personalization ensures that learners receive material aligned with their pace and comprehension levels, fostering more effective outcomes. Beyond content delivery, agentic AI serves as a virtual tutor and streamlines administrative tasks such as grading and attendance tracking.

Additionally, agentic AI is fundamentally changing the way software is developed by enabling systems that generate code, understand development goals, reason about tasks, and take autonomous actions across the lifecycle of an application. These systems can scaffold projects based on high-level descriptions, iteratively refine code in response to errors or test failures, and coordinate between multiple tools or services without constant human oversight. As the AI interacts with developers, it learns preferred styles, recognizes common patterns, and adapts to team conventions, helping to reduce repetitive work and cognitive load. Over time, this transforms developers into higher-level problem solvers by creating the boilerplate code and handling orchestration in the background. The result is faster iterations with fewer errors.

These examples are already creeping into organizations, but challenges remain:

Data quality
: Agentic AI relies heavily on the quality of the data it processes. Biased or inaccurate data can cause harmful decisions, including discriminatory behavior. Ensuring that data is accurate, representative, and unbiased is critical for reliable outcomes.

Security risks
: These systems often manage sensitive data, making them attractive targets for cyberattacks. Vulnerabilities can lead to breaches or manipulation, as seen in some open source AI models with severe security flaws.

Ethical concerns
: The autonomy of agentic AI raises ethical issues, especially when decisions affect human lives.

The integration of agentic AI into real-time systems marks a significant step toward intelligent, responsive operations, with growing potential for solving complex, time-sensitive problems while requiring vigilance around data, security, and ethics.

Real-Time RAG

In-memory embeddings work like in-memory vector databases. They're conducive to real-time AI applications by facilitating efficient storage, retrieval, and processing of high-dimensional data for real-time RAG.

Embeddings are numerical representations of data—such as words, images, or other entities—transformed into vectors in a high-dimensional space. This transformation allows AI systems to process and understand complex data types by capturing semantic relationships; for instance, words with similar meanings are represented by vectors that are close together in this space.

Consider the words *king*, *queen*, *man*, and *woman*. In a high-dimensional vector space, these words can be represented as vectors with specific coordinates.

A vector database is a specialized system for storing and managing *vector embeddings*, which are numerical representations of raw data generated by machine learning models. These databases enable fast similarity searches for AI applications like recommendation systems. Vector searches use algorithms such as ANN to quickly find similar vectors. In-memory vector databases enhance this process by storing embeddings directly in RAM, thus allowing for rapid retrieval and processing, which is critical for real-time AI applications. Systems like Vemcache, Milvus, and Aerospike support in-memory vector searches and efficiently handle large-scale data to

meet the demands of low-latency environments. Some applications for real-time AI and in-memory vector databases include:

Recommendation systems
: In-memory embeddings enable recommendation systems to swiftly analyze user behavior and preferences, providing personalized suggestions in real time. For instance, platforms can process user interactions and immediately update recommendations for products or content, enhancing user engagement and satisfaction.

Real-time search
: Embeddings stored in memory enable search engines to interpret the contextual meaning behind user queries, delivering more relevant and accurate results instantly.

NLP
: In-memory embeddings are crucial for NLP applications such as sentiment analysis, language translation, and text summarization. They allow AI models to process and understand text data quickly.

Cybersecurity and finance
: In domains like cybersecurity and finance, in-memory embeddings facilitate the rapid detection of anomalies or fraudulent activities by enabling real-time analysis of data patterns.

Virtual assistants
: In-memory embeddings allow chatbots and virtual assistants to maintain context and understand user inputs more effectively, leading to more natural and coherent interactions. This capability is vital for delivering real-time, context-aware responses in conversational AI applications.

Self-Hosted Models Versus Cloud-Hosted Models

Self-hosting AI models involves deploying and managing machine learning models on an organization's own infrastructure, as opposed to utilizing third-party, cloud-based AI services. Advancements in open source LLMs, such as Meta's Llama 3.1, have made self-hosting a practical option for enterprises by offering performance on par with proprietary models. The dramatic improvements in hardware specially designed for AI have made hosting models of all sizes, even complex LLMs, a real possibility for organizations. Companies

like NVIDIA have introduced compact, high-performance systems designed for AI workloads.[3]

One of the biggest motivations for self-hosting is driven by data privacy concerns, especially for industries handling confidential or regulated data, such as health care or finance. Self-hosting ensures that proprietary data remains within the organization's control, reducing exposure to such risks.[4]

Beyond data security, there are a number of reasons to consider self-hosting AI models:

Regulatory compliance
: Industries subject to regulations like GDPR, the Health Insurance Portability and Accountability Act (HIPAA), and other oversight requirements find that self-hosting provides a clear path to compliance.

Reduced external dependency
: Self-hosting mitigates risks associated with vendor lock-in and service disruptions. Organizations gain autonomy over their AI infrastructure.

Performance and customization
: Self-hosted models can be fine-tuned to specific organizational needs, potentially enhancing performance for particular tasks.

Cost management
: While initial hardware investments for self-hosting can be high, ongoing costs are typically low and predictable, encompassing maintenance expenses. In contrast, cloud-hosted proprietary models often operate on a pay-as-you-go model.

3 Umar Shakir, "Nvidia's Cute 'Digits' AI Desktop Is Coming This Summer with a New Name and a Big Brother," *The Verge*, March 18, 2025, *https://www.theverge.com/news/631957/nvidia-dgx-spark-station-grace-blackwell-ai-supercomputers-gtc*; and Stephen J. Bigelow, "GPUs vs. TPUs vs. NPUs: Comparing AI Hardware Options," *TechTarget*, August 27, 2024, *https://www.techtarget.com/whatis/feature/GPUs-vs-TPUs-vs-NPUs-Comparing-AI-hardware-options*.

4 Tim Abbott, "Self-Hosting Keeps Your Private Data out of AI Models," Kandra Labs, May 23, 2024, *https://blog.zulip.com/2024/05/23/self-hosting-keeps-your-private-data-out-of-ai-models*.

Latency and inference speed
: Self-hosted models provide consistent latency and inference speed, limited by available hardware.

Customization
: Self-hosting offers full ability to host fine-tuned models, allowing organizations to tailor models to their specific needs. Cloud-hosted solutions often have limited customization options.

Conclusion

Real-time AI marks an evolution in how organizations perceive and leverage data—not merely as historical artifacts to be analyzed after the fact, but as living signals that demand immediate interpretation and response. Organizations that embed real-time AI into their operations are not just automating tasks; they are enabling systems that perceive, interpret, and act as events unfold. This paradigm shift allows businesses to move beyond reacting to change and toward anticipating and shaping it.

The value unlocked by real-time AI is multifaceted and compounding. In customer service, it augments human agents with contextualized, adaptive support that reduces cognitive load and improves resolution times. In cybersecurity, it transforms passive monitoring into proactive defense, identifying threats the moment they emerge. In manufacturing, it converts maintenance from a reactive cost center to a predictive asset-management function. In finance, it automates trading at a scale and speed that no human could match. In advertising and gaming, it delivers personalization so fluid and dynamic that user engagement becomes self-reinforcing. In each of these cases, it redefines the experience by shortening the time between insight and action, which in turn compresses the time between opportunity and impact. In doing so, it directly influences business outcomes: faster time to resolution, higher customer satisfaction, lower operational costs, greater resiliency, and increased revenue per interaction.

About the Authors

Blaize Stewart is an experienced systems architect with a passion for building scalable, real-time solutions. From writing his first web apps in high school to developing complex applications across embedded systems, handheld devices, desktop apps, and the cloud, Blaize has continuously embraced emerging technologies. His expertise spans cloud-based architectures, IoT solutions, and real-time data streaming. With extensive hands-on experience in building globally scalable systems on Azure and integrating cutting-edge technologies like RAG architectures and vector databases, Blaize is adept at addressing the technical and business challenges of modern AI-driven data pipelines.

Bipin Singh is the Sr. Director of Product Marketing at Redpanda. He is a seasoned product marketing leader with a deep passion for connecting customers to technical products. His expertise spans data infrastructure, AI/ML, observability, automation, and security. Outside of work, Bipin enjoys building things, traveling, and spending time with his wife and two kids. Bipin holds an MBA from Babson College, a PhD from Iowa State University, and a BTech in Chemical Engineering from the Indian Institute of Technology Kanpur.