an introduction

wingfoil

the ultra-low latency data streaming framework

November 2025

background

What is streaming data?

Streaming data is data that is continuously generated and delivered in real, or near-real time. Unlike more traditional batch processing, where data is collected, stored and processed later, streaming data allows systems to handle information as it arrives, which in turn makes it possible to detect patterns, generate insights, and trigger actions without waiting for a complete dataset to be collected first. Streaming data has emerged over the last 10-15 years as a very significant component of the digital ecosystem, driven in part by the exponential growth of data and in part by the push to develop ever faster and more responsive systems. As a result, streaming data pipelines have become a common part of many organisations' digital infrastructure, with new streaming technologies like Kafka and ReactiveX making it possible to scale digital applications and data infrastructure in ways that were previously impossible.

A large and growing market

Streaming data is a large and rapidly growing market, driven by both the general growth in data, and an increased demand for faster and more responsive data driven applications across a range of sectors from e-commerce to finance.

$128Bn
2030
large market

30%
CAGR - 2030
rapid growth

diverse market

The streaming data market was worth $27Bn in 2024.* Confluent, the developers of Kafka, one of the leading streaming data products, generated $963 million in total revenue for fiscal year 2024,** a 24% YOY growth. The streaming data market as a whole is also growing rapidly and is predicted to reach $110 Bn*** in total value by the end of 2030 with a CAGR of 30%.

* https://www.fortunebusinessinsights.com/streaming-analytics-market-108619
** https://investors.confluent.io/news-releases/news-release-details/confluent-announces-fourth-quarter-and-fiscal-year-2024
*** https://www.grandviewresearch.com/industry-analysis/streaming-analytics-market

Ultra low latency – the next frontier

In digital communication, latency refers to the delay between sending and receiving data: from video streaming to High Frequency Trading, latency determines responsiveness, efficiency and often, competitive advantage. And while latency is a universal property of streaming networks, its significance varies widely across markets and applications. In real-time video for example, a latency of sub-200 milliseconds is good enough to ensure natural conversation. In consumer-facing web services, sub-100 millisecond performance ensures users don't feel like the system is lagging. However, when it comes to building electronic marketplaces, for example, much lower latencies are required with small differences in microseconds and even nanoseconds determining billions of dollars in outcomes.

There is no universally agreed benchmark for measuring streaming data latency, but the following table represents a broad overview of acceptable latencies in various environments.

	range	***applications***
ultra -low latency	< 500 μs	electronic market making, high frequency trading
low latency	500 μs – 100ms	multiplayer games
medium latency	100 ms – 500 ms	web data, video streaming
high latency	> 500 ms	batch jobs, offline analytics, ETL pipelines

Medium latency - 100ms - 500ms
Studies show that delays above 200-300 ms can reduce user engagement or sales conversions. Applications like Zoom or Microsoft Teams demand 50-150 ms one-way latency to support fluid, conversational exchanges. Once delays exceed 200 ms, participants notice lags that disrupt turn-taking.

Low latency - 500 μs - 100 ms
For online multiplayer games, fairness hinges on latency - or "ping" - between players. Competitive gaming requires <50 ms latency, with professional standards often at <20 ms. While sub-millisecond improvements don't matter to human perception, small differences between players can decide outcomes in fast-paced competitions.

Ultra-low latency < 500 μs
At the extreme end of the scale are ultra-low latency applications where latencies are measured in microseconds and nanoseconds. Financial markets move at machine speed. Here, milliseconds translate into significant financial advantage. Exchanges invest in colocation (housing servers physically near the exchange engine) and low latency network infrastructure to keep order execution fast. Some firms measure order execution and quote updates in single-digit microseconds, and innovations such as microwave transmission or FPGA hardware are deployed to shave fractions of microseconds off latency for high frequency trading. The reason is simple: if one firm can act 50 μs faster than another, it captures profit opportunities first. In this context, latency is not about user experience but about competitive advantage.

It is common for investment banks to use stream-oriented electronic trading systems that operate in the 50-200μs range. Software-based High Frequency Trading systems, can operate in the 2-10μs range. FPGA (hardware) based HFT systems operate in the 100 ns - 1μs range.

As mentioned, there are no absolutes when it comes to latency, context and perception are key, but these broad categories are useful when it comes to understanding uses cases and competencies.

The ultra-low latency opportunity

There are a range of streaming data products and frameworks that cover a range of use cases from simple event based streaming protocols like Tokio and Reactive X to more complex streaming frameworks like Kafka, but none of these are fast enough to build ultra-low latency applications. This is, in part, because many existing streaming products were designed for use cases where throughput was more important than ultra-low latency, e.g. processing large amounts of click stream data on highly-scaled web applications - Kafka was developed by a team at Linked In for just this purpose. And while latency was obviously a consideration when these systems were designed, they were not explicitly designed to work at the ultra-low latencies required to build the core infrastructure of electronic market places and certain realtime AI applications.

Kafka Streams
Kafka Streams, the streaming component of the wider Kafka framework, is built in Java (a productive but relatively slow language). Kafka Streams is also tightly bound to the wider streaming framework and message bus, which adds further overhead.

Reactive X
Reactive X was developed by Microsoft for streaming in .NET and then became a wider method for handling data in Java and other languages. Unlike Wingfoil, Reactive X doesn't have historical mode and operates with a depth first rather than breadth first graph.

Currently, the only way to develop the ultra-low latency streaming performance required for applications like electronic market places is by 'hand rolling' your own bespoke solution using methods like Asynchronous Streams in Rust or Coroutines in C++, but many developers find these methods complex to work with and hard to scale. For example, in Rust, the learning curve for Asynchronous Streams can be steep, even for experienced developers. In addition, stream support in the standard library remains limited, often requiring external crates and macros, and it can be hard to debug - asynchronous code, especially streams composed with combinators, which can produce cryptic error messages and stack traces. There are also some non obvious issues with the performance of Asynchronous Streams - e.g. waking up threads adds overhead.

All of which means maintenance of these complex systems is difficult and expensive with skilled staff difficult to find and retain. It also means that every bank, exchange and hedge fund develops and maintains it's own ultra-low latency streaming solution - a vast duplication of effort that, in many cases, could be avoided.

And this is why we've developed Wingfoil - the ultra-low latency streaming framework - because there is a huge opportunity to develop a single, user-friendly framework that can be used for ultra-low latency streaming. A framework that is easy to deploy and maintain and ultimately saves time and money for those who adopt it.

No other existing, widely-available streaming framework can operate at the ultra-low latencies require to run applications like electronic market places

wingfoil

Wingfoil – overview

Wingfoil is an ultra-low latency, highly-scalable stream processing framework used to receive, process and distribute streaming data. Built in Rust, it's open source, user friendly and seamlessly supports both real-time and historical workloads.
Wingfoil uses a Directed Acyclical Graph (DAG) to efficiently coordinate the execution of its nodes, so application developers implement logic to wire up their graph of calculations which Wingfoil's graph engine then executes. This helps Wingfoil deal with high scalability of throughput and remain ultra-low latency. Wingfoil comes with near zero-cost abstraction (see below), meaning that performance approaches that of Asynchronous Streams - the native Rust method for streaming data. However, Wingfoil is much simpler to use than developing directly with Asynchronous streams, making it much more productive and scalable, again, for negligible cost overhead.

<500μs

Wingfoil is ultra-low latency
Wingfoil prioritises end-to-end latency under 500μs for small messages, even under high throughput, ensuring systems respond immediately to changing conditions. It uses a zero-copy architecture to reduce serialisation and deserialisation overhead, and it handles back pressure efficiently without stalling the pipeline, maintaining system stability under load.

In a benchmark we measure graph overhead by wiring up a trivial graph, 10 nodes deep and 10 nodes wide, with all nodes ticking on each engine cycle. Running on 3.80 GHz CPU, we observe a latency around 2μs per engine cycle, equivalent to 20ns per node cycle. For a smaller graph of 10 nodes, this would give a graph overhead around 200ns per engine cycle.well within the performance envelope of software-based High Frequency Trading systems. (Note that the total number of cycled nodes is what counts — large, sparsely cycled graphs are handled very efficiently.)

Wingfoil is open source
Wingfoil is an open source project - you can find out more at:
https://github.com/wingfoil-io
https://crates.io/crates/wingfoil

Wingfoil is simple to deploy
Wingfoil offers simple, composable APIs so you can plug it directly into your machine learning models or risk engines and easily integrate with existing tools. Wingfoil has browser bindings in Python and Typescript and integrates with Tokio to simplify the setup of asynchronous I/O adapters, which means workloads can be efficiently distributed across multi-threaded and multi-server environments.

Wingfoil saves time and money
Wingfoil dramatically cuts development operational costs by streamlining data infrastructure. Its ultra-low latency eliminates the need for expensive, complex real-time solutions and reduces debugging time associated with data lag. This efficiency minimizes developer hours, server resources, and ongoing maintenance, leading to substantial savings throughout the development lifecycle and beyond.

Rust - a language that offers performance and productivity

Wingfoil is written in Rust, a language that combines the speed of C++ (Rust consistently performs within 5-10% of C++ in benchmark tests) with the productivity of more 'developer friendly' languages such as Java and Go.
C++ is designed for performance, but has lots of features that slow developer productivity, e.g. memory management pitfalls and cryptic compiler errors. The CLBG benchmark shows Rust outperforming Java by 2-3x in most tests and Go by 3-5x in CPU-intensive tasks. Rust is also widely liked by the developer community - for eight consecutive years, Rust has claimed the top spot as the "most loved programming language" in Stack Overflow's Developer Survey, with 87% of developers expressing love for the language in 2023.

Wingfoil – use cases

Wingfoil can be used across a wide range of general use streaming use cases if you want to improve speed and performance, and it integrates well with other broader frameworks like Kafka, but Wingfoil is specifically designed for use in ultra-low latency use cases like electronic market places and certain realtime AI applications. This is, in part, because many existing streaming products were designed for use cases where throughput was more important than latency, e.g. processing large amounts of click stream data on highly-scaled web applications - Kafka was developed by a team at Linked In for just this purpose. And while latency was obviously a consideration when these systems were designed, they were not explicitly designed to work at the ultra-low latencies required to build the core infrastructure of electronic market places and certain realtime AI applications. Wingfoil is particularly suited to building:

Electronic market places

While some of the current streaming frameworks can manage less critical aspects of electronic market places, only Wingfoil has the ultra low latency capabilities that allow you to build the core infrastructure of these systems, e.g. displaying quotes and prices, matching buyers and sellers. In areas like fraud detection, ultra low latency performance also allows marketplaces to identify and block suspicious transactions before they complete, thereby reducing risk. Additionally, by reducing delays between data generation and system action, ultra-low latency streaming frameworks help deliver a better user experience in competitive, time-sensitive environments.

Realtime AI Applications

Low latency data streaming is vital for real-time AI applications, e.g. self driving cars, UAVs and robots. The quicker you can ingest and process large amounts of sensor data, e.g. LiDAR, video or radar, the more effective an application will be at detecting obstacles, adjusting speed, and making navigation decisions in dynamic environments. By combining streaming data with real-time AI, these physical digital systems can maintain safety, efficiency and reliability in environments where conditions change rapidly and decision delays can have very significant consequences.

Conclusion

Data streaming is a large and fast growing sector: the amount of data in the world is forecast to double from 2024 to 2028, and streaming data is forecast to grow at similar rates. And the demand for better performance at ultra-low latency will only increase, as the physical and digital realms become increasingly intertwined and a need for ever faster real time data drives more systems. Wingfoil is the simple-to-deploy, super lightweight data streaming framework, built for a world where latency is measured in microseconds and even nanoseconds: a framework that saves developers time and ultimately money.

Wingfoil, like other frameworks, saves you time, effort and money and allows you to concentrate to developing value added features.

Learn more about Wingfoil

The Wingfoil framework is open source and is available at:

https://github.com/wingfoil-io
https://crates.io/crates/wingfoil

To get in touch with the Wingfoil team or learn more contact us at:

hello@wingfoil.io

wingfoil

the ultra low latency streaming framework

hello@wingfoil.io

https://crates.io/crates/wingfoil

www.github/wingfoil-io