Hadean

Hadean is a deep tech startup with a reimagined distributed computing platform. The Hadean platform and libraries are written in Rust to ensure high performance, reliability and memory safety. Users can interact with the platform through frameworks, including muxer, which optimises the communication flow of geographically distributed clients and compute clusters.

Rust

Rust is an open-source programming language focussed on concurrency, speed, and memory safety. Although syntactically similar to C++, it addresses security vulnerabilities related to memory errors and concurrency. You can still introduce security vulnerabilities with Rust. It's most used for applications that require performance and safety, including simulation, operating systems, file systems and browser components. Rust is a programming language that can be used to develop things for the Internet of Things and the Edge Cloud.

Muxer

Muxer provides real-time streaming and event handling across a distributed cloud and edge network. It enables previously unreachable network scale and data throughput speed. Rust ensures its memory safety and reliability, and asynchronous I/O handles thousands of connections simultaneously without needing a single thread per client.

- Bidirectional data flow
- Long running processes
- Low latency data streaming
- Fully customisable
- Dynamic scaling
- Asynchronous architecture

MUXER

Hadean and Applications of the Rust Programming Language

Although customers can use C++, the majority of Hadean libraries are written in Rust, supporting high-performance programming and ensuring applications scale across a distributed cloud and edge network.

Consistent runtime behaviour makes Rust an excellent choice for Hadean's purposes, and thanks to its static type system, carefully selected features, and having no garbage collector, it compiles into performant and predictable code. It is statically typed, so the type system helps the developer to deter certain classes of bugs during compilation. For complex systems like the Hadean platform, this guarantee is crucial.

Rust provides the Hadean Platform with several distinct advantages:

Performance: Rust is fast and memory efficient. Not having runtime or a garbage collector, it empowers performance-critical services and easily integrates with other languages. Rust's provides automatic memory management through its unique ownership system, which makes garbage collection unnecessary.

Reliability: Rust has a rich type system and ownership model that guarantees memory and thread-safety. It enables the developers to eliminate several bugs at compile time. Thanks to the borrow checker, Rust avoids data races at compile-time. Data races lead to unpredictable behaviour and take place when two threads access the same memory at the same time.

Productivity: Rust is easy to use, has very good documentation, a friendly compiler, and a smart editor support tool that provides auto-completion and type inspections. The autocompletion tools and code analyzers were developed by the Rust community to support developers when programming in Rust. Moreover, zero-cost abstractions make sure that there is virtually no runtime overhead for the abstractions that you use, so you only pay for what you use. It also has several conveniences of a 'modern' language, like a standard package manager and assorted language conveniences.

Hadean, Rust and the Edge

The muxer library runs on edge datacentres and optimises the communication flow of geographically distributed clients and compute clusters. It supports data aggregation, buffering, prioritisation and conditional routing of traffic through customisable out-of-the-box algorithms.

Being sensitive to latency, Muxer finds that tail latencies are nearly null while using Rust as there is no runtime garbage collection. Rust is a great fit for cloud services like Muxer, with its focus on speed and elimination of entire classes of bugs common in C or C++. Rust is an excellent choice when performance matters, such as when processing large amounts of data, which is typically the case across distributed networks with millions of connected devices. Furthermore, Rust gives you high control over how threads behave and how resources are shared between threads, which in the context of distributed computing, ensures efficient parallelisation of tasks across multiple edge data centers.

In cloud and edge computing, low-level infrastructure components such as the hypervisor, the network or the storage, require high performance to reduce critical overhead. Rust maintains speed and performance, (without negating security and safety), thanks to its compiler that produces very efficient machine code.

HADEAN

Key Features of Rust in Relation to Hadean

Zero Cost Abstractions and Monomorphisation

Rust is based on the concept of zero-cost abstraction, meaning that "What you don't use, you don't pay for [Stroustrup, 1994]". In most cases, the abstractions are resolved at compile-time, leading to 'zero-cost abstractions' without any runtime overhead. This allows the Hadean platform code to be declared at compile-time, making it already safe and with no runtime overhead. Rust lets us write highly composable and re-usable code that gets optimised out at runtime. A good example of this is its trait system: by making all the channels conform to a particular trait, Hadean developers can write code that is general over any kind of channel implementation they might want to use (or come up with in the future). But when code is compiled, Rust performs monomorphisation, generating high-performance code specialised to each type of channel, so the Hadean developers don't pay any performance penalty for that flexibility. This is particularly important for channels, which are a core part of Hadean and used everywhere in both platform and user code — any performance cost in the channel implementation can compound to produce a significant slowdown across the system as a whole!

Asynchronous I/O

Asynchronous programming is very well-integrated and stable in Rust compared to C++, allowing Hadean's developer to run a unit of work separately from the primary application thread and providing better performance and enhanced responsiveness to the final operating system.

Low Latency, Real-time Streaming

Muxer's long running processes remove the need for repeated inefficient connection requests, reducing the overall time it takes for data to be sent back and forth. At the same time, by elevating interest management to a first class concern, Muxer optimises bandwidth usage and prioritises sending key information to the client in real-time.

Borrow Checker

By keeping track of where data is used throughout the program and by following a set of rules, the borrow checker can determine where data needs to be initialized and where it needs to be freed (or dropped, in Rust terms). It auto-inserts memory allocations and frees for you, giving you the convenience of a garbage collector with the speed and efficiency of manual management.

No Garbage Collector

Thanks to its memory management ownership mechanism, Rust ensures memory safety without a garbage collector. The absence of a garbage collector makes performance in critical applications more powerful and avoids additional checks at run-time.

Practical Applications

Rust is the most popular programming language in the world and there is a significant increase in adoption by leading tech companies, including AWS, Microsoft and Dropbox.

Rust applications are naturally suited to scenarios that require powerful, cross-platform command-line tools, distributed online services, embedded devices or systems programming. Existing applications are as diverse as city simulation, bioinformatics and industrial automation.

Hadean's tech has been validated in distributed agent-based simulation and can be used to model real-world complex simulations. Using Rust to build and connect IoT devices, enables the creation of much more secure and robust low-cost products and plays a crucial role in unlocking smart cities and digital twins projects where security is the main concern. IoT development requires a very high application performance and developer productivity making Rust the ideal choice.

Rust applications are already making a big impact on IoT and edge computing. Microsoft cloud developers defined Rust as "the Industry's 'Best Chance' at Safe Systems Programming" explaining why Microsoft is gradually switching to Rust to build its infrastructure software, away from C/C++.

Microsoft Azure IoT Edge Security Daemon has proven it by exploiting in its implementation the efficient safety characteristics of Rust. The Security Daemon is a component of the Azure IoT Edge system that enables secure SAS token and certificate provisioning for IoT Edge modules. It acts as a communication broker between the Azure IoT Edge runtime and many host services such as the container runtime and hardware-based cryptography devices Hardware Security Modules (HSM) and Trusted Platform Modules (TPM).

Hadean's tech applies these similar principles to its communication tool, Muxer. A client authentication layer provides security mechanisms that protect against throttling, security risks and client misbehaviour such as failing to accept updates or sending too much event data back to the application. Ultimately, the use of Rust with the Hadean's platform favors its distributed and parallel computing nature and dynamically allows high-performance activities regardless of computational intensity.

> "Imagine writing a 50 line program to analyse megabytes of data on your laptop. Now imagine applying the same program to terabytes of data, using thousands of processors in the cloud. This is the system Hadean is building. It will enable everyone to access and explore the world's data using unlimited computing power – leading to new understanding, ideas and innovations.
>
> — David May, Inventor of the first parallel microprocessor

www.hadean.com | info@hadean.com

**8th Floor, 9 Appold Street,
London, EC2A 2AP
0020 3514 1170**