The Compute Continuum for Cyber-Physical Systems

The Compute Continuum is a seamless infrastructure crossing the borders of Cloud and Edge, deploying a single runtime in every host in the system, whether a resourceful Cloud node or a constrained Edge device, regardless of its architecture and operating system.
In our vision, the Compute Continuum should feature three pivotal aspects:

(1) Application components should interact through Web-level protocols, adopting a Web-like resource-sharing approach.
Protocols such as HTTP/3 offer language-agnostic interfaces between application components, which eases access to remote resources.

(2) Components should migrate in a lively manner, hence stateful and connection-preserving, as a result of orchestration-level decisions.
It represents a more efficient alternative to *offloading*, enabling the execution of components where it is more *convenient* to run without redeploying and restarting them.

(3) Resources, both in Cloud and Edge, should be dynamically (and temporarily) federated to tackle application-specific objectives.
The scarcity of resources intrinsic to individual Edge nodes is conveniently mitigated by pooling-and-federation, enabling roaming where computation can follow physical changes and other needs instead of “staying put” statically.

Real-time and embedded systems, as those applied in cyber-physical applications, are among the domains that could benefit from this computing model thanks to:

(1) A homogeneous runtime enabling an application component, written in a memory-safe programming language, to be executed in virtually any computing node in the system.

(2) Enhanced performance and efficiency, due to the capability of the system to migrate the computation where it is more convenient to execute.
For instance, to grant data locality or reduce latency.

(3) Live-update, allowing the system to start an upgraded component and then migrate it to its proper location while it remains operational.
The facilities introduced to support live migration, namely checkpoint and restore, would support the creation of live-update mechanisms.

Two existing technologies, WebAssembly (Wasm) and Rust, can play a crucial role in the construction of this infrastructure.
The former offers a portable bytecode format that can be run in every host exposing a Wasm runtime, regardless of the underlying architecture. Moreover, Wasm components execute in a sandboxed environment with restrained access to external resources.
Thanks to ownership-based memory management, the latter provides statically-assessable memory safety, enforced at compile-time.
In terms of gap analysis: a Rust-based Wams runtime, Wasmtime, currently exists that provides sound isolation guarantees between a wasm component and the external environment, while granting memory safety within and outside of itself.
At present, Wasmtime can run on top of Linux-based OSs (among others) and is still under active development.
In this presentation, we are going to illustrate the envisioned model for the Compute Continuum, the main challenges in its construction, and the scenarios emerging from it.