

WASI

WebAssembly System Interface

Una System Interface per un sistema operativo concettuale



WebAssembly (Wasm)

formato bytecode standardizzato (W3C)
originariamente pensato per il web

efficiente e veloce

leggero

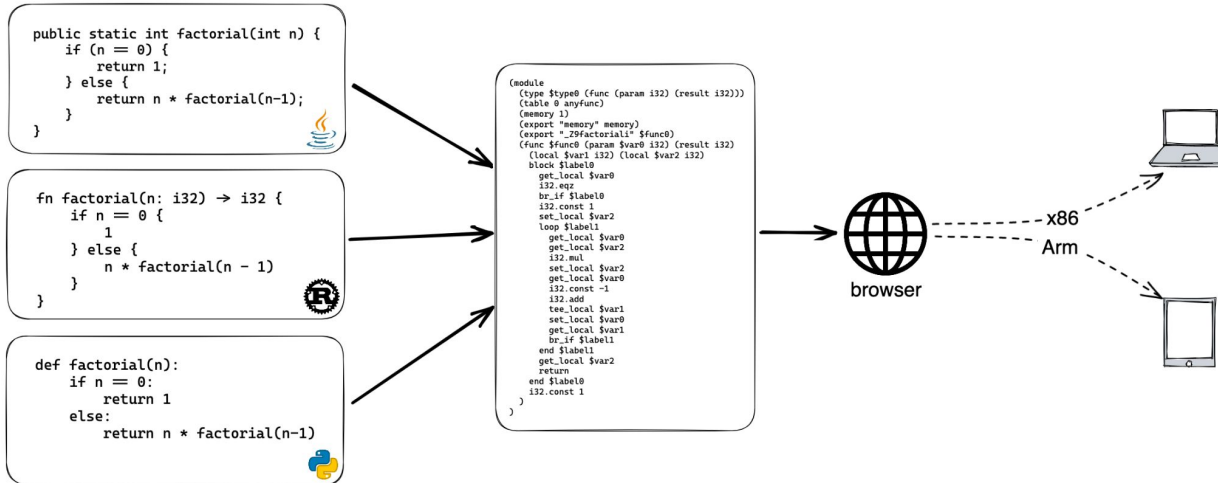
sicuro

portabile

interoperabile

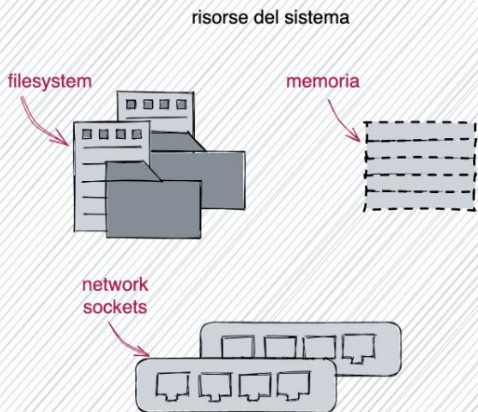
Ma perché?

migliorare le performance di applicazioni web scritte in Javascript **JS**



System Interface

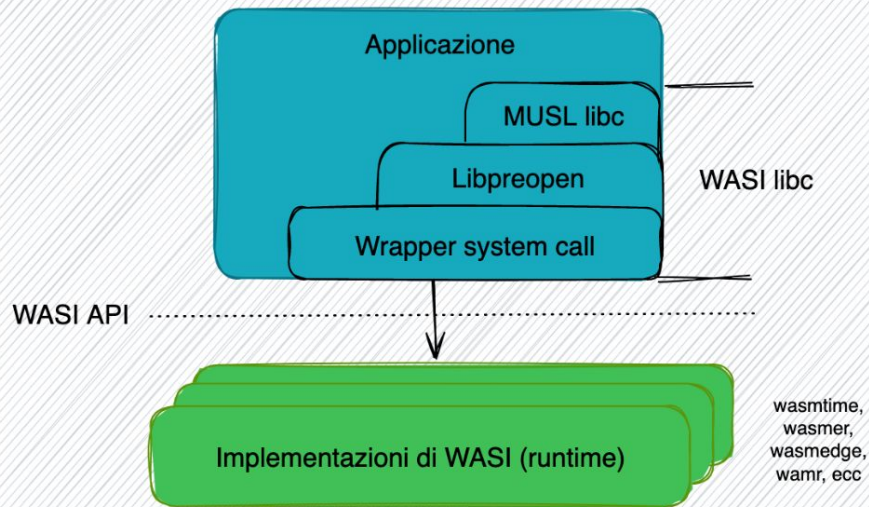
Il problema



API & system call
rese disponibili

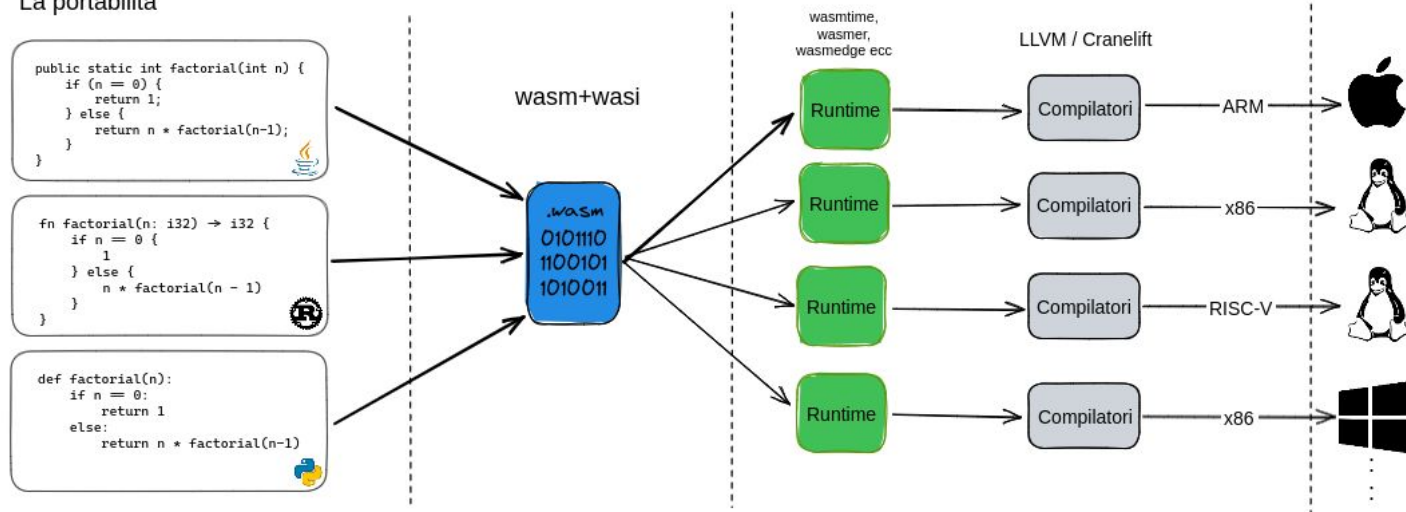
listen
write
clock
read
getrandom
open

La soluzione



La portabilità e sicurezza con WASI

La portabilità

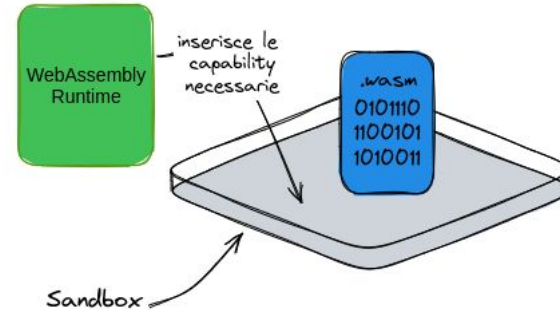


La sicurezza - capability based & sandbox

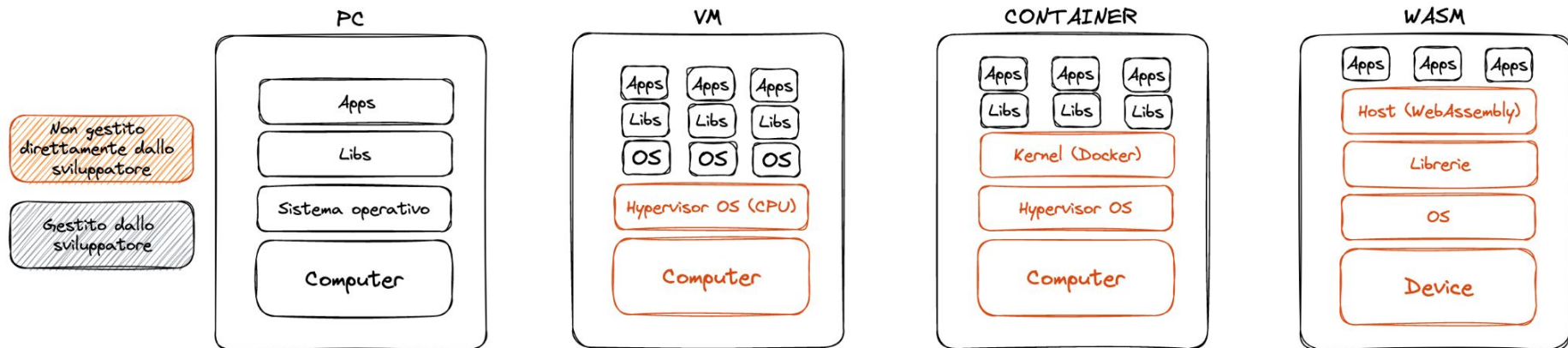
	/etc/passwd	/myfolder/file.txt	...
Alice	{}	{read}	
Bob	{read, write}	{read, write}	
Carol	{read}	{read}	

capabilities

ACLs

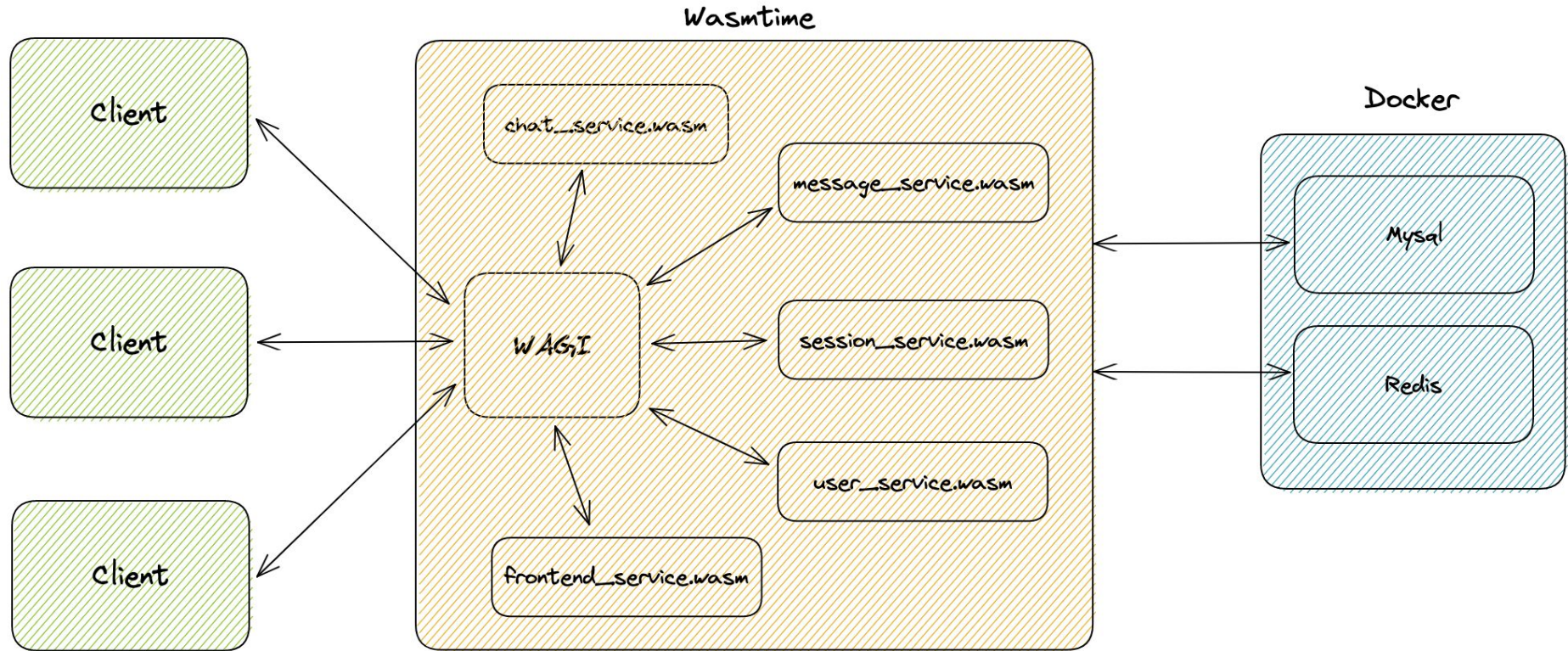


Un caso d'uso: il cloud



Gestito da Sviluppatore	Tutto	OS, App, Lib	App, Lib	Wasm
Astrazione	-	CPU	Linux Kernel	Sandbox
Impatto su disco	Alto	Medio	Basso	Molto Basso
Portabilità	-	Bassa	Media (CPU, Linux)	Alta
Tipo di Sicurezza	Intro sistema	OS	A livello di processo Unix	Capability-Based

Verso l'implementazione



Un confronto e Conclusioni



Benchmark - CPU Apple M1 8 core e 16 GB di RAM

100 utenti connessi in simultanea, inserimento messaggi nell'applicazione

NodeJS:

- tempi di risposta < 12ms
- throughput 603 kReq/min
- utilizzo della cpu < 5%

Wasm:

- tempi di risposta < 61ms
- throughput 135 kReq/min
- utilizzo della cpu tra il 28% e il 38%

Differenza di size (container OCI)

NodeJS: ~65Mb

Wasm: ~2.5Mb

