

# SVILUPPO DI APPLICAZIONI BASATE SU WEB ASSEMBLY SYSTEM INTERFACE (WASI)



Una System Interface per un sistema operativo concettuale

Relatore:

**Prof. Paolo Bellavista**

Presentata da:

**Luca Corsetti**

# Obiettivi della tesi

- 👉 Studiare WebAssembly in contesti non-browser
- 👉 Comprenderne le potenzialità in ambiente cloud
- 👉 Sviluppare un prototipo per valutare la realizzabilità e lo stato attuale della tecnologia

# WebAssembly (Wasm)

formato bytecode standardizzato (W3C)  
originariamente pensato per il Web

efficiente e veloce

leggero

sicuro

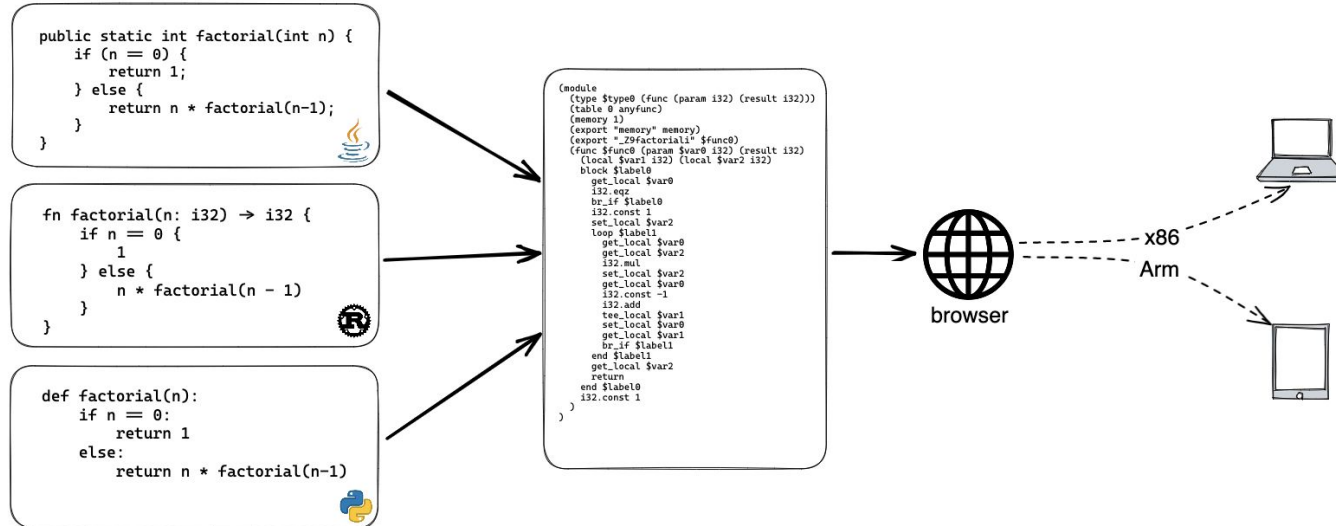
portabile

interoperabile

Ma perché?

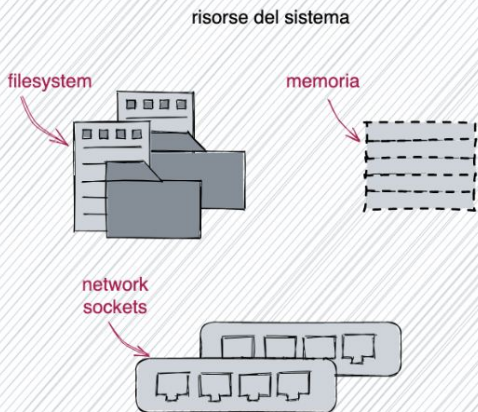
migliorare le performance di applicazioni Web scritte in Javascript

JS



# System Interface

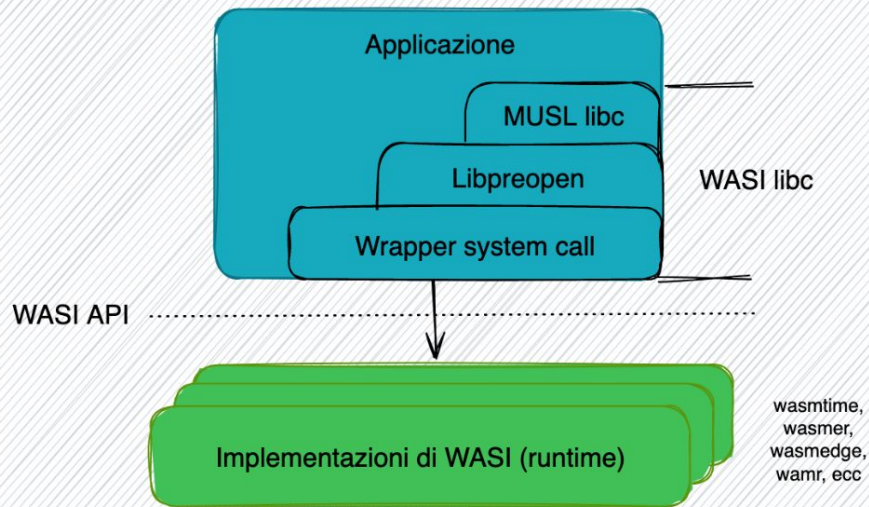
## Il problema



API & system call  
rese disponibili

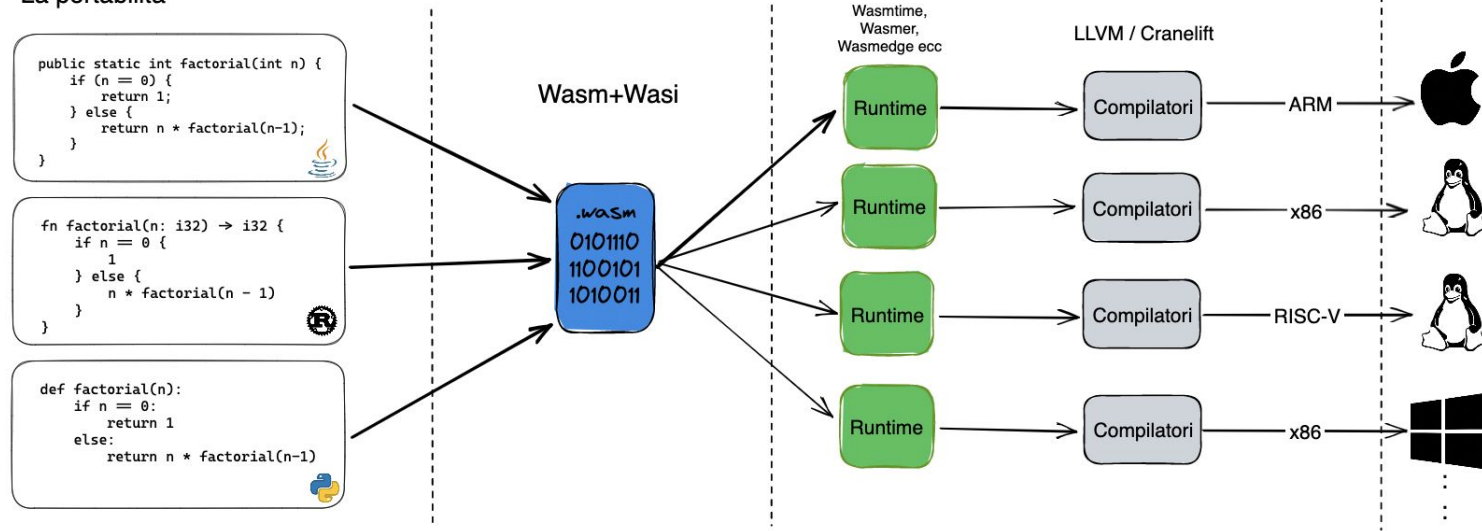
listen  
write  
clock  
read  
getrandom  
open

## La soluzione



# La portabilità e sicurezza con WASI

## La portabilità

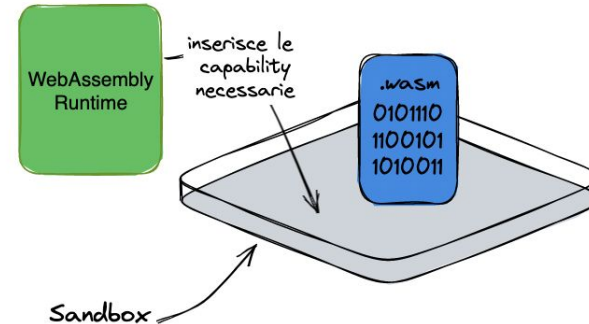


## La sicurezza - capability based & sandbox

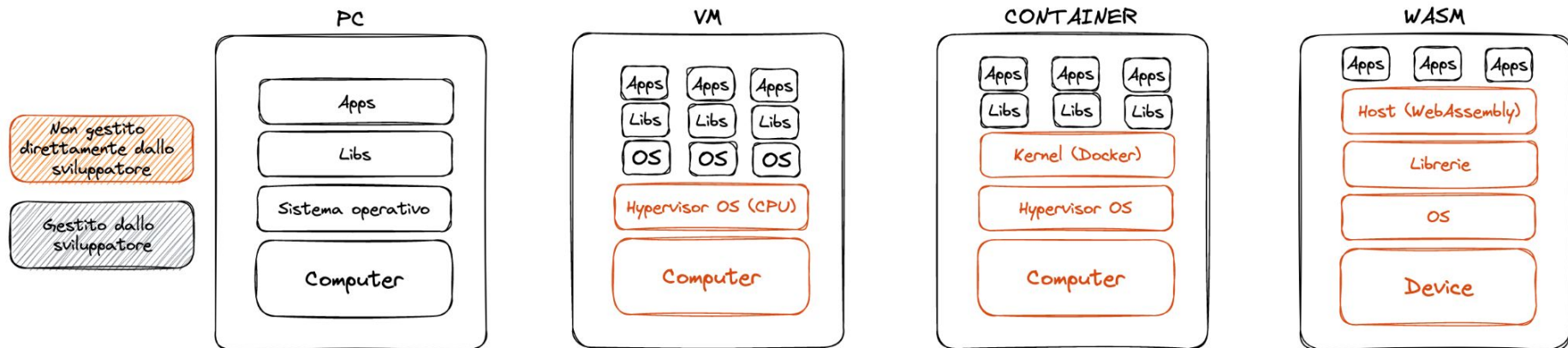
	/etc/passwd	/myfolder/file.txt	...
Alice	{}	{read}	
Bob	{read, write}	{read, write}	
Carol	{read}	{read}	

capabilities

ACLs



# Un caso d'uso: il cloud



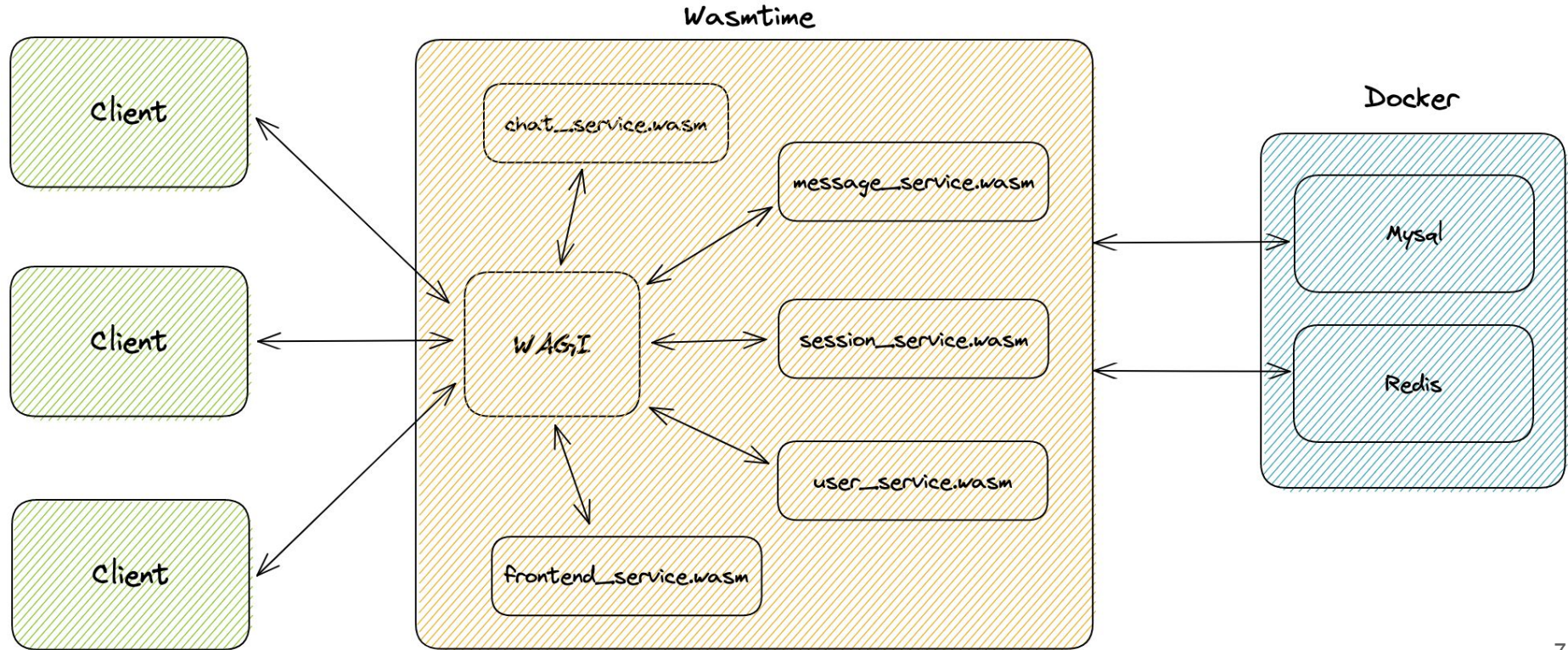
Gestito da Sviluppatore	Tutto	OS, App, Lib	App, Lib	Wasm
Astrazione	-	CPU	Linux Kernel	Sandbox
Impatto su disco	Alto	Medio	Basso	Molto Basso
Portabilità	-	Bassa	Media (CPU, Linux)	Alta
Tipo di Sicurezza	Intro sistema	OS	A livello di processo Unix	Capability-Based



# Verso l'implementazione



<http://46.101.162.175/>



# Un confronto



<http://46.101.162.175/>

## Differenza di size (container OCI)

NodeJS: ~65Mb

Wasm: ~2.5Mb

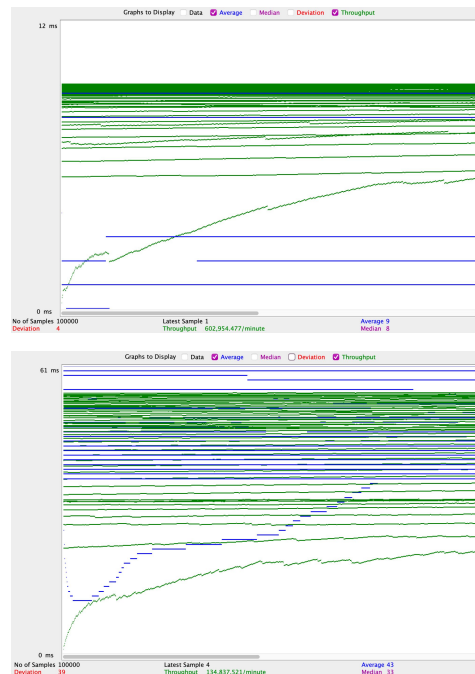
## Benchmark

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%





# Conclusioni



<http://46.101.162.175/>

- 😞 ancora acerbo
- 😞 standard lento
- 😞 ecosistema emergente
- 😊 garanzie di sicurezza
- 😊 interoperabilità del codice sviluppato
- 😊 stessa codebase, architetture diverse

Docker co-founder



Solomon Hykes / @shykes@hachyderm.io  
@solomonstre

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task! [twitter.com/linclark/statu...](https://twitter.com/linclark/status/1107000000000000000)

9:39 PM · Mar 27, 2019 · Twitter Web Client