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?

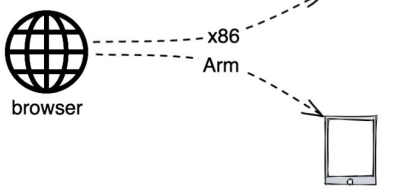
migliorare le performance di applicazioni web scritte in Javascript JS

```java
public static int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n-1);
    }
}
```

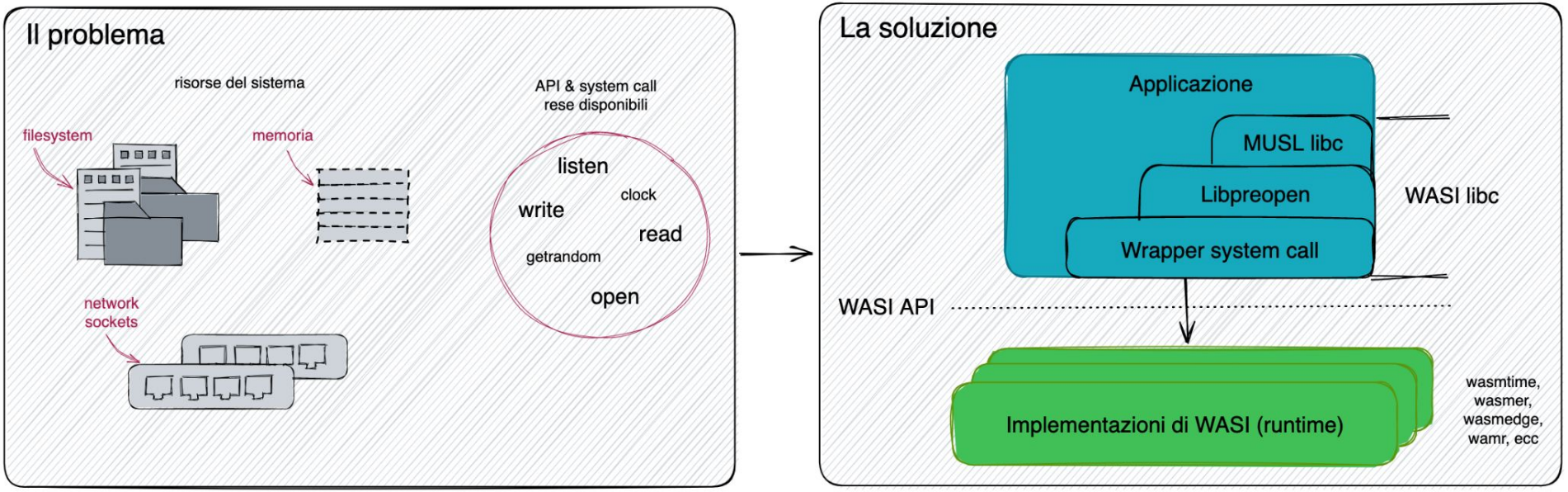
```rust
fn factorial(n: i32) -> i32 {
    if n == 0 {
        1
    } else {
        n * factorial(n - 1)
    }
}
```

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

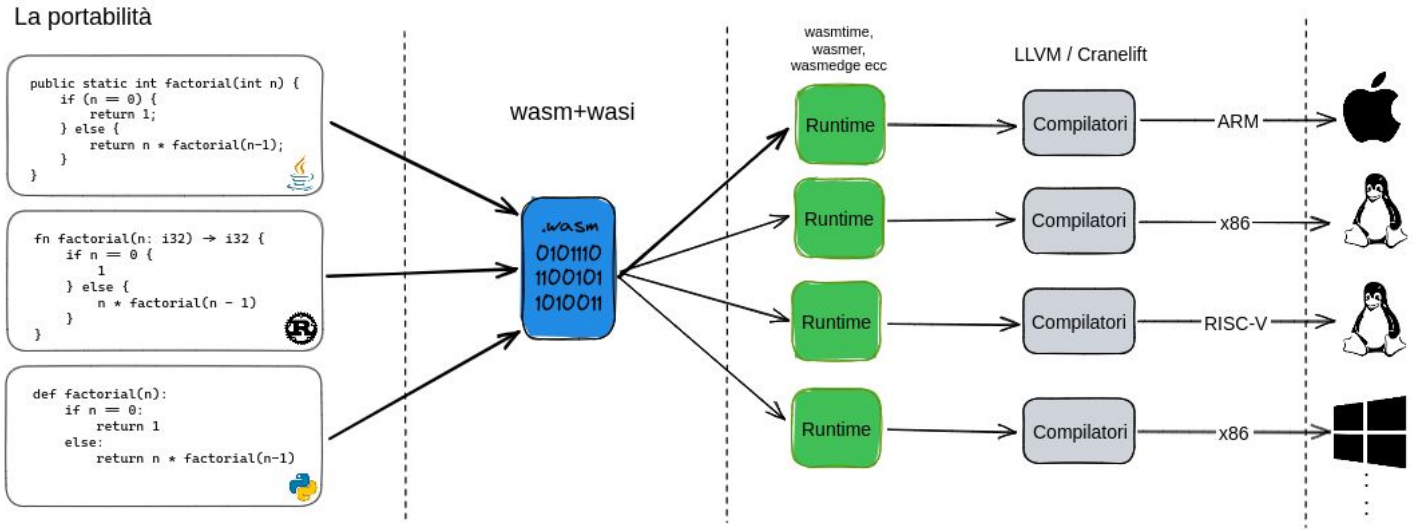
```
(module
  (type $type0 (func (param i32) (result i32)))
  (table 0 anyfunc)
  (memory 1)
  (export "memory" memory)
  (export "_Z9factoriali" $func0)
  (func $func0 (param $var0 i32) (result i32)
    (local $var1 i32) (local $var2 i32)
    block $label0
      get_local $var0
      i32.eqz
      br_if $label0
      i32.const 1
      set_local $var2
      loop $label1
        get_local $var0
        get_local $var2
        i32.mul
        set_local $var2
        get_local $var0
        i32.const -1
        i32.add
        tee_local $var1
        set_local $var0
        get_local $var1
        br_if $label1
      end $label1
      get_local $var2
      return
    end $label0
    i32.const 1
  )
)
```

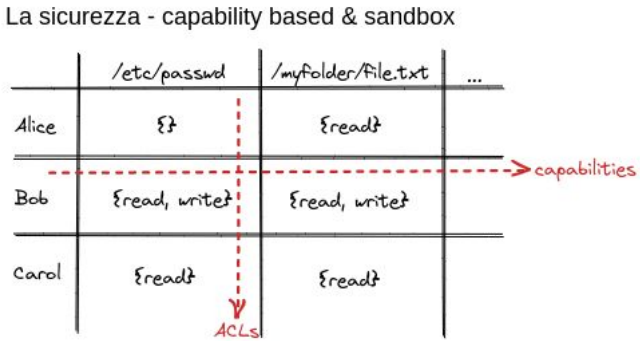


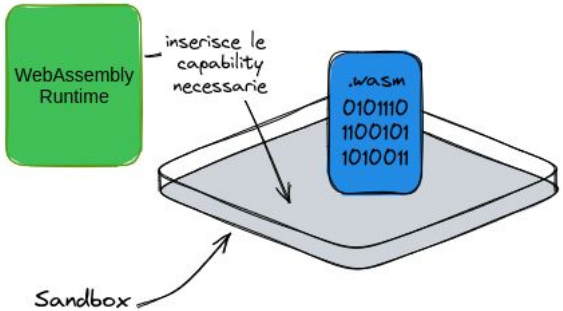
System Interface



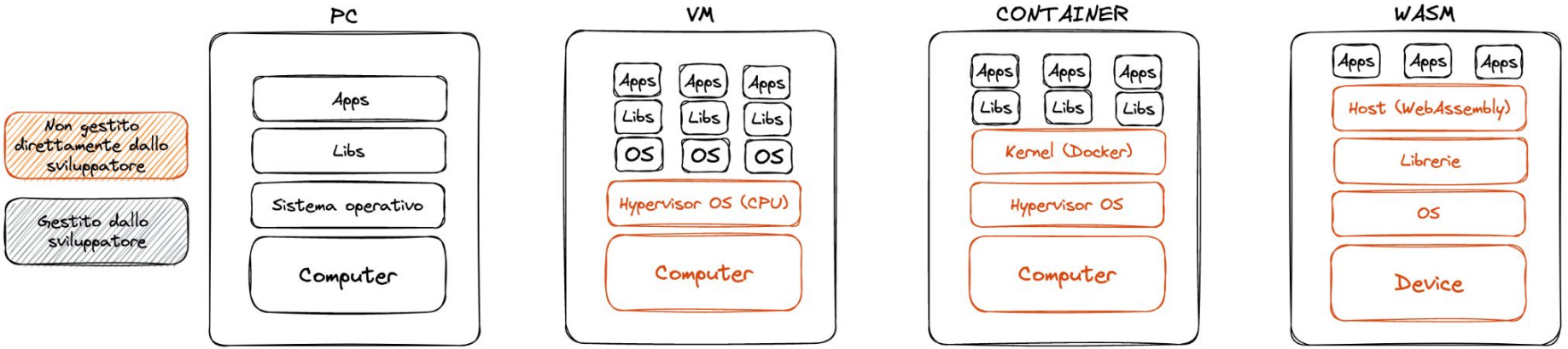
La portabilità e sicurezza con WASI





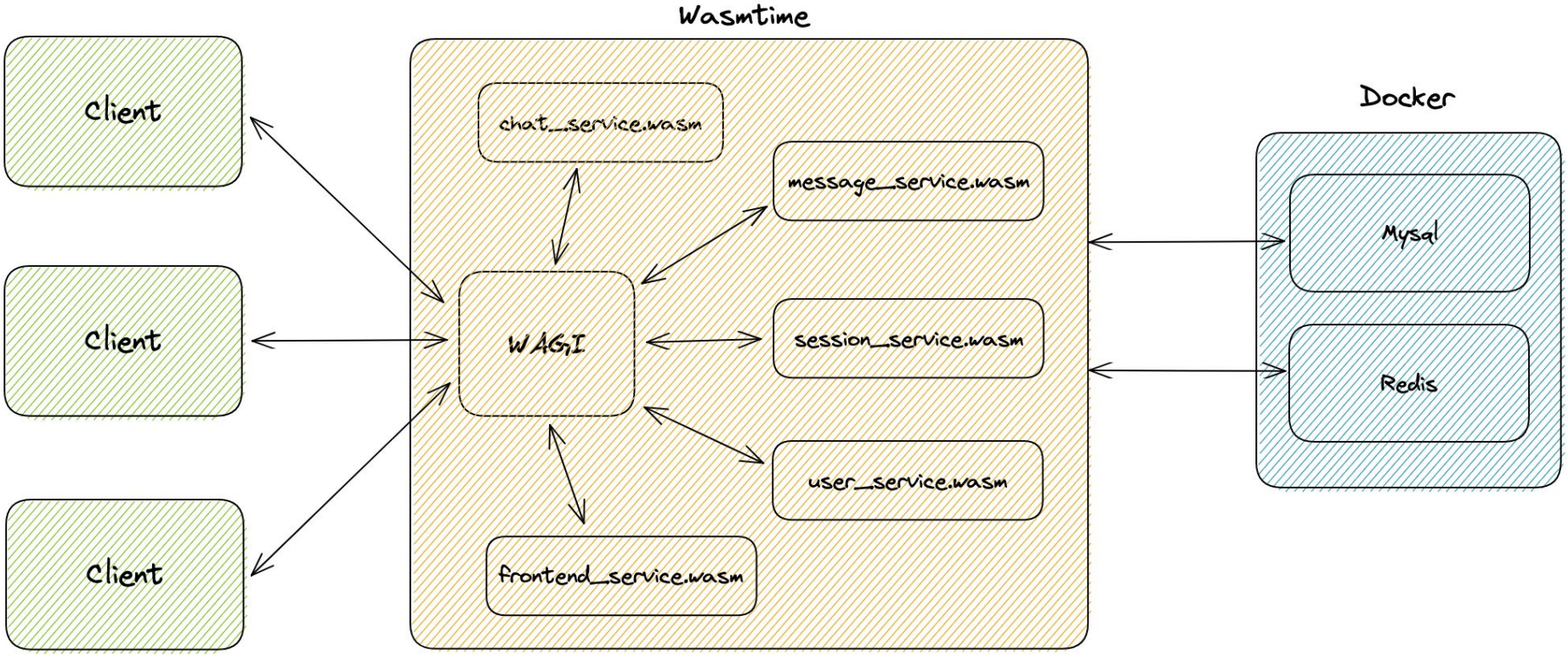


Un caso d'uso: il cloud



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 (NodeJS vs Wasm)

Benchmark
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