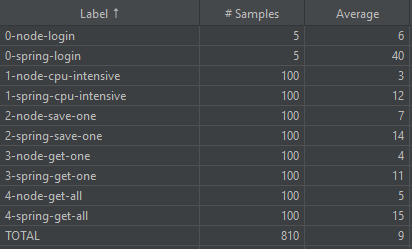
**5 thread, 5 secondi di ramp-up period, 20 loop di tutti I metodi**

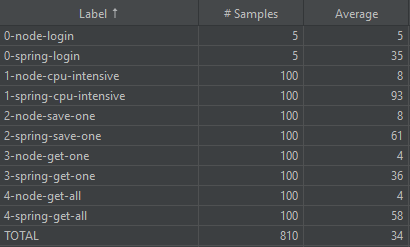
1. Intensive 1 -> BOOT perde contro NODE senza CPU

1cont0.5cpu100Mbram1intensive



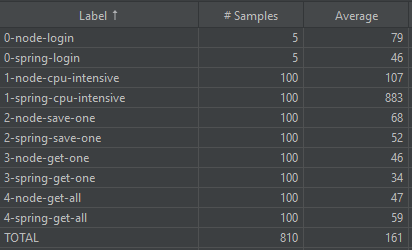
1. Intensive 5 -> boot continua a perdere

1cont0.5cpu100Mbram5intensive



1. Intensive 7 sono pari

1cont0.5cpu100Mbram7intensive

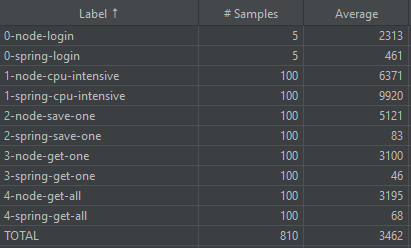


Cpu intensive 10 -> spring crasha per out of memory

Node invece no!

Perchè Java usa i Thread in maniera automatica, e I thread occupano un sacco di memoria…

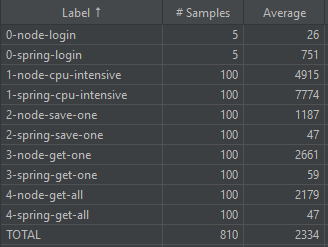
Aumentiamo la ram a 200MB, e rieseguiamo I test CPU intensive 10



Ed ecco qui, I risultati spring sono pressoche inviariati, grazie all’utilizzo automatico dei thread.

I risultati di node invece ne risentono parecchio.

Se proviamo ad aumentare le repliche dei container a 2, ipotizziamo un miglioramento maggiore di nodeJS rispetto a springboot.



**E’ un discorso che si assomiglia ad utilizzare delle strutture di memoria ADDIZIONALI per velocizzare lo svolgersi di alcune operazioni.**

**BUILD AND STARTUP**

[**https://gist.github.com/GaetanoPiazzolla/956470742577f60425fa06784d39eb29**](https://gist.github.com/GaetanoPiazzolla/956470742577f60425fa06784d39eb29)

**Tempo di avvio di NodeJS: 2 secondi**

**Tempo di avvio di spring boot: 20 secondi**

**Build Immagine docker di nodeJS: 6 secondi**

**Build imagine docker di spring boot(gradle): 2 minuti**