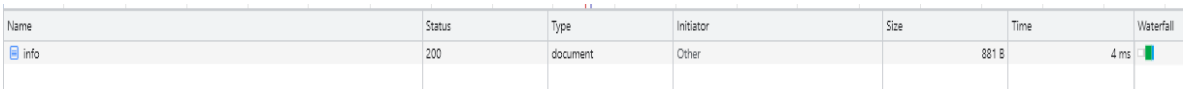
Logs, debug y profilling

1) **Verificando diferencia con y sin Compression();**

http://localhost:8080/api/info

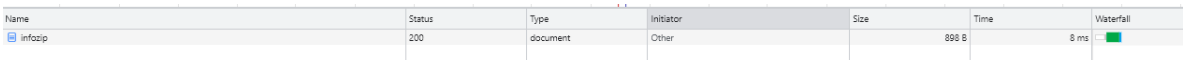
Tamaño: 881 Bytes



Name	Status	Type	Initiator	Size	Time	Waterfall
info	200	document	Other	881 B	4 ms	

http://localhost:8080/api/infozip

Tamaño: 898 Bytes



Name	Status	Type	Initiator	Size	Time	Waterfall
infozip	200	document	Other	898 B	8 ms	

Si bien no se aprecia la compresión hice las pruebas con repeat sobre un string y en esos casos si se observa una gran diferencia en tamaño.

2) Utilizando –prof

- **Artillery:**

Without Child:

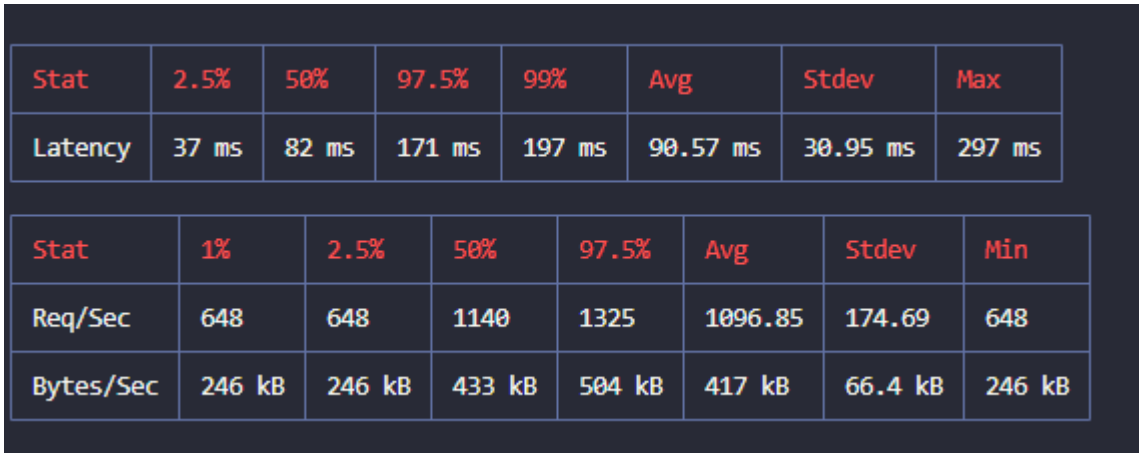
```
http.codes.400: ..................................................................... 963
http.request_rate: .................................................................. 212/sec
http.requests: ...................................................................... 984
http.response_time:
  min: .............................................................................. 29
  max: .............................................................................. 426
  median: ........................................................................... 206.5
  p95: .............................................................................. 278.7
  p99: .............................................................................. 295.9
http.responses: ..................................................................... 963
```

With Child:

```
http.codes.200: ..................................................................... 8
http.request_rate: .................................................................. 19/sec
http.requests: ...................................................................... 58
http.response_time:
  min: .............................................................................. 8562
  max: .............................................................................. 8689
  median: ........................................................................... 8692.8
  p95: .............................................................................. 8692.8
  p99: .............................................................................. 8692.8
http.responses: ..................................................................... 8
```

- **Autocannon:**

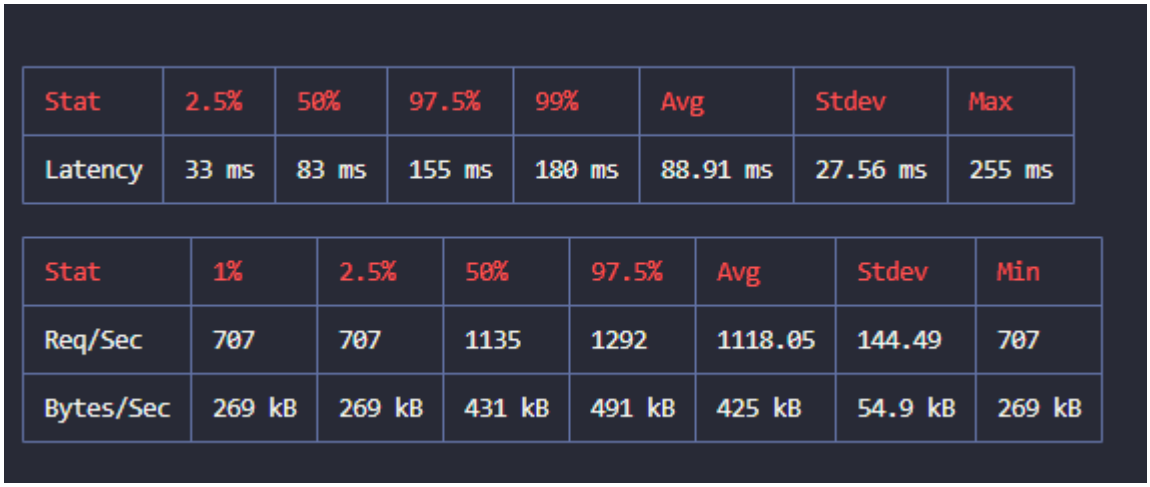
Without Child:



Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	37 ms	82 ms	171 ms	197 ms	90.57 ms	30.95 ms	297 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	648	648	1140	1325	1096.85	174.69	648
Bytes/Sec	246 kB	246 kB	433 kB	504 kB	417 kB	66.4 kB	246 kB

With Child:

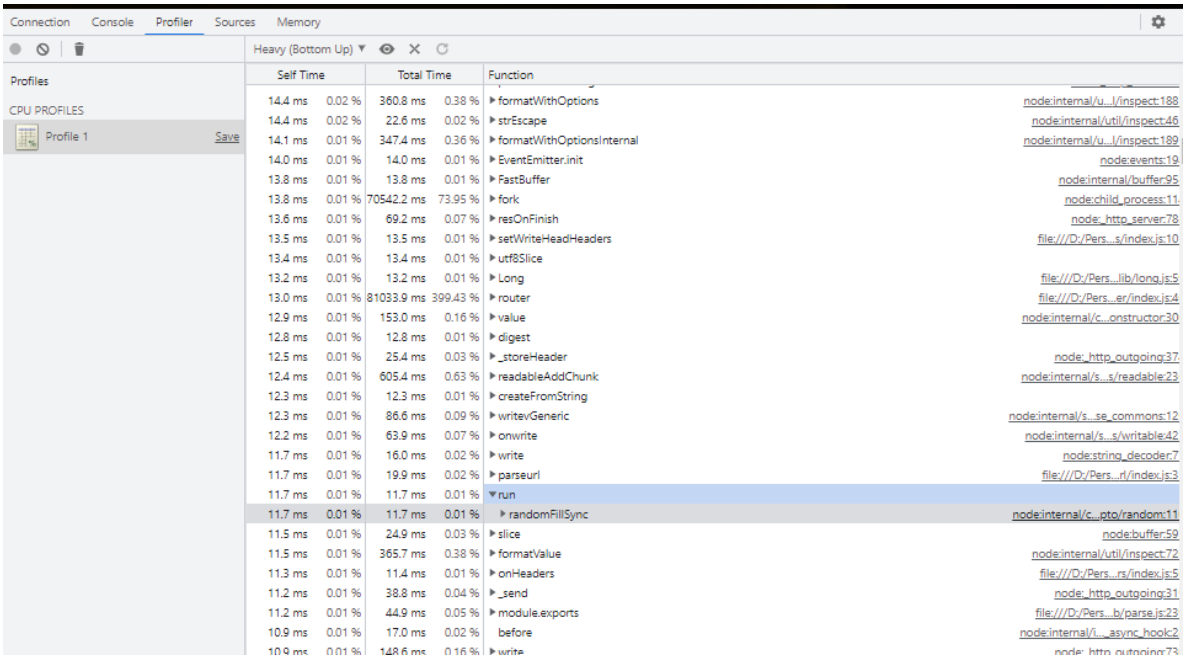


Stat	2.5%	50%	97.5%	99%	Avg	Stdev	Max
Latency	33 ms	83 ms	155 ms	180 ms	88.91 ms	27.56 ms	255 ms

Stat	1%	2.5%	50%	97.5%	Avg	Stdev	Min
Req/Sec	707	707	1135	1292	1118.05	144.49	707
Bytes/Sec	269 kB	269 kB	431 kB	491 kB	425 kB	54.9 kB	269 kB

Se observa en ambos que se produce un rendimiento superior utilizando child-procces.

3)Utilizando devTools google:



Acá intente entrar tal cual indicaba en el documento, pero nunca abrió el archivo.

4)Diagrama de flama:

Conclusión:

En todos los casos se observa una gran mejora a la hora de utilizar child-procces. Esto agiliza el acceso a los diferentes recursos.

Nota: Se aclara que utilice la ruta /random bloqueando y desbloquenado la condición de child procces para poder obtener diferencias a la hora de utilizar las distintas herramientas.