Garbage Collector Impact

David Ferreira Quaresma (david.quaresma@ccc.ufcg.edu.br)

janeiro, 2019

Descrição

Os resultados analisados neste documento foram obtidos através da execução de múltiplas chamadas de uma mesma função de modo sequencial, isto é, não concorrente. Para tal, utilizamos o script **curl-workload** para gerar carga e observar o impacto do coletor de lixo no **thumbnailator-server**.

Experimento

Função e Ambiente

- Ambiente de execução: **servidor HTTP extraído do OpenFaaS**.
- Lógica de negócio da função: **Redimencionamento de uma imagem**.
- Escala de redimencionamento: 0.1.
- Tamanho da imagem: 131kb.

Setup

- workload: 10.000 requisições enviadas sequencialmente.
- jvm: openjdk version "11"
- gc: Garbage First Garbage Collector (G1 GC)
- heap: 128mb
- taskset: 2 CPUs

Dados observados

- Quantidade de coletas de lixo durante execução da função.
- Duração das coletas de lixo durante execução da função.
- Tempo de execução da função.

Observações:

- Scavenge: coleta na Young Gen.
- MarkSweep: coleta na Old Gen.
- Warmup: removemos as 100 primeiras requisições

Resultados

```
results = read.csv("./results/thumbnailator-01s-131kb-128h-j11.csv", header=T, dec=".")
results = tail(results, -100)
nocollect <- filter(results, scavenge_count + marksweep_count == 0)
withcollect <- filter(results, scavenge_count + marksweep_count > 0)
```

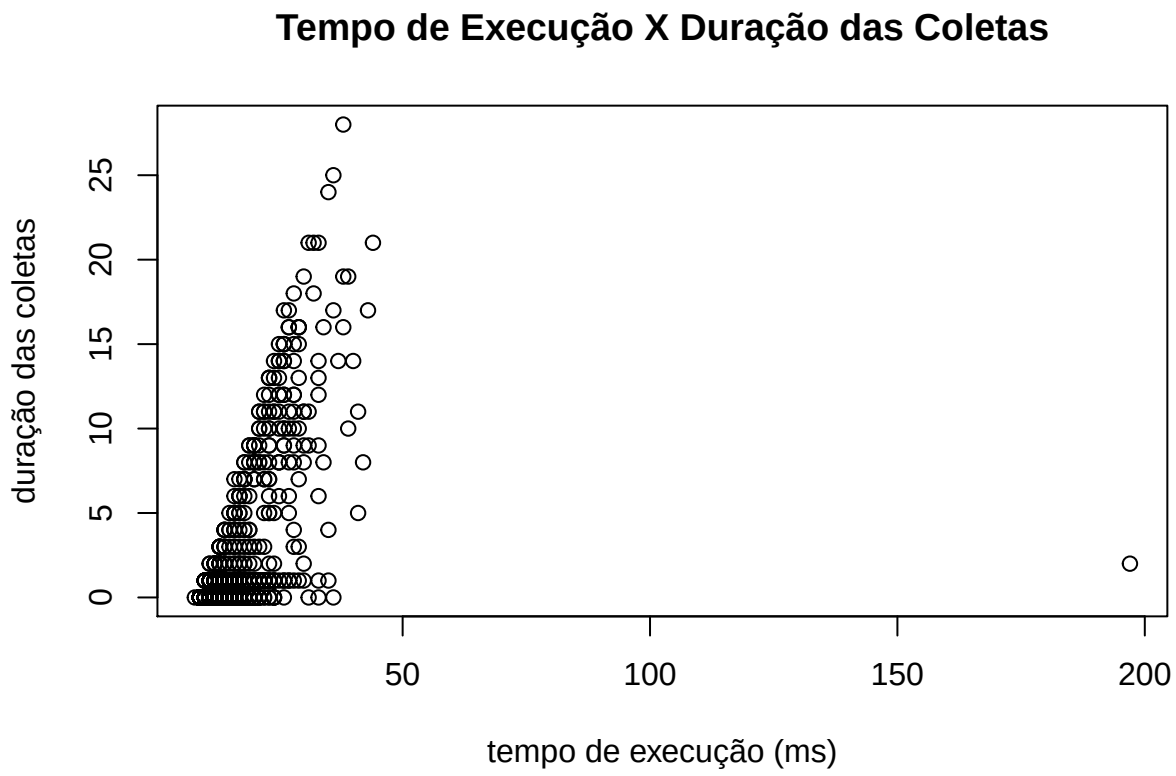
Número de coletas

```
coletas_dataframe(results)
```

```
##                   tipo coletas
## 1 scavenge (young gen)     756
## 2  marksweep (old gen)       0
```

Scatterplot

```
time_collecting = results$scavenge_time + results$marksweep_time
plot(results$execution_time, time_collecting, xlab="tempo de execução (ms)",
     ylab="duração das coletas", main="Tempo de Execução X Duração das Coletas")
```



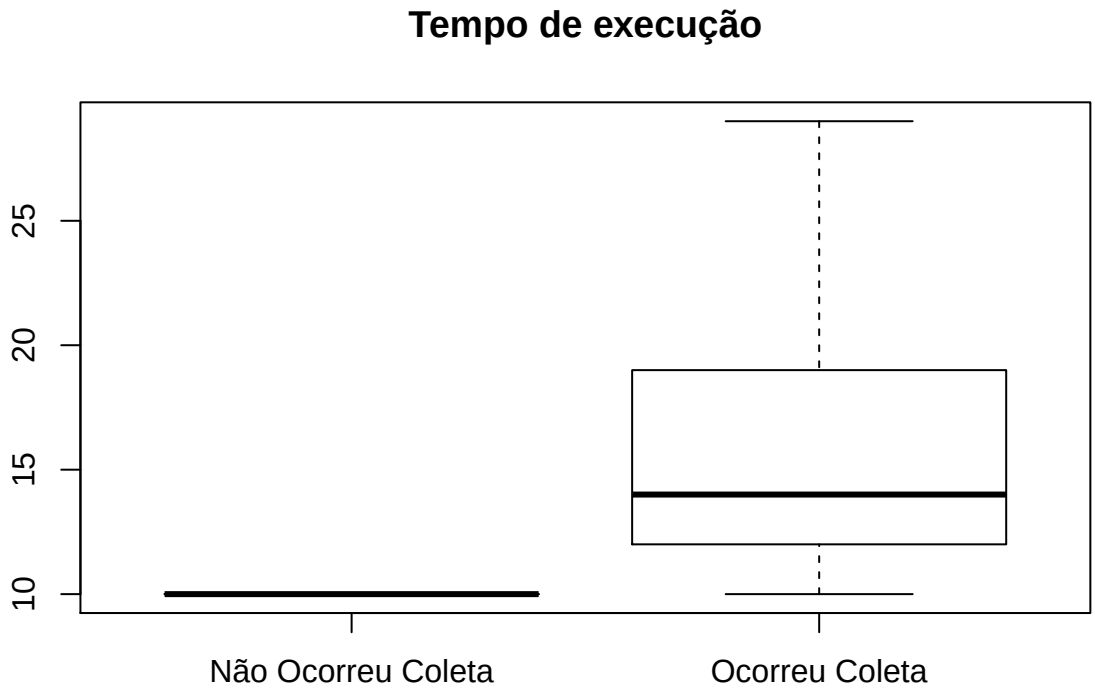
Correlação Linear

```
cor_dataframe(results)
```

```
##                                correlacao     valor
## 1 Tempo de serviÃ§o X nÃºmero de coletas 0.5107718
## 2    Tempo de serviÃ§o X tempo coletando 0.6116190
```

Boxplot

```
boxplot(nocollect$execution_time, withcollect$execution_time, outline=FALSE,
        names=c("Não Ocorreu Coleta", "Ocorreu Coleta"), main="Tempo de execução")
```



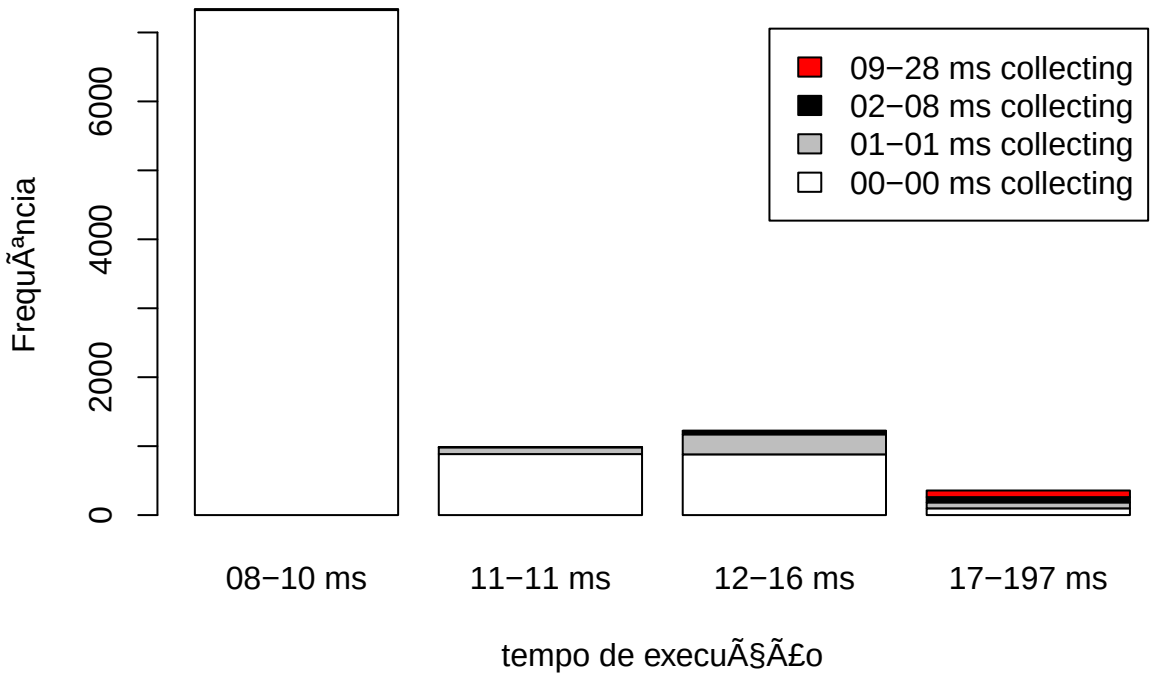
Média, Variância e Desvio Padrão

```
meanVarSd(nocollect, withcollect)
```

```
##   statistic noCollect withCollect comparison
## 1      mean 10.305446   16.802910   1.630488
## 2       var  2.204953   83.914746  38.057379
## 3        sd  1.484909    9.160499   6.169066
```

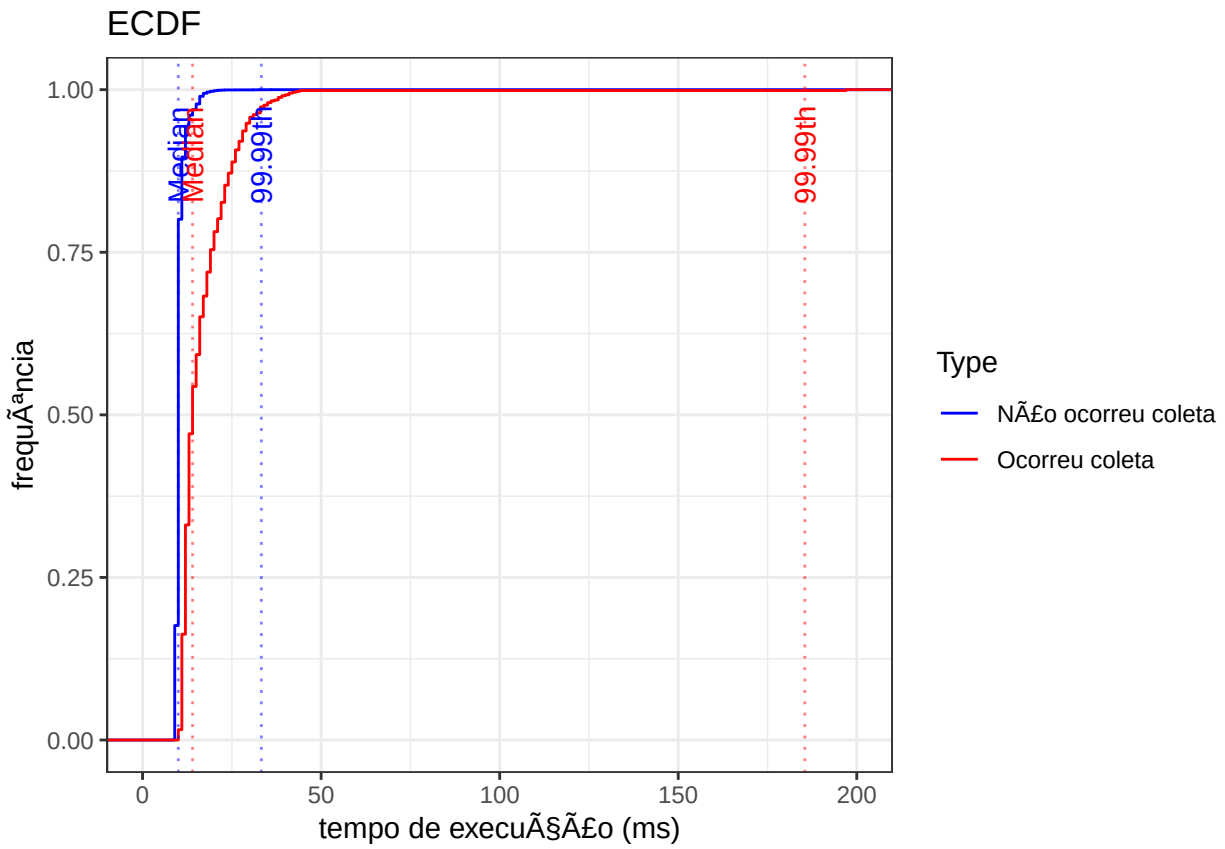
Barplot

```
build_barplot(results, running_names = c("08-10 ms", "11-11 ms", "12-16 ms", "17-197 ms"),
              collecting_names = c("00-00 ms collecting", "01-01 ms collecting",
                "02-08 ms collecting", "09-28 ms collecting"))
```



ECDF

```
graph_tail(nocollect$execution_time, withcollect$execution_time,
           title="ECDF", x_limit_inf=0, x_limit_sup=200, annotate_y=0.90)
```



Quantiles

Tempo coletando

```
quantile_wrapped(time_collecting)
```

```
##       0%      25%      50%      75%      90%      95%      99%    99.9%
##  0.00000  0.00000  0.00000  0.00000  0.00000  1.00000  8.00000 18.10100
##   99.99%  99.999%     100%
## 25.03030 27.70303 28.00000
```

Tempo executando

Com e Sem Coleta

```
quantile_wrapped_for_execution_time(results)
```

```
##       0%      25%      50%      75%      90%      95%      99%    99.9%
##   8.0000  10.0000  10.0000  11.0000  13.0000  16.0000  25.0000  38.0000
##   99.99%  99.999%     100%
##  45.5453 181.8545 197.0000
```

Comparação: Sem coleta X Com coleta

```
quantiles_dataframe_comparison(nocollect, withcollect)
```

```
##         nocollect withcollect comparison
## 0%        8.00000     10.0000   1.250000
## 25%      10.00000     12.0000   1.200000
## 50%      10.00000     14.0000   1.400000
## 75%      10.00000     19.0000   1.900000
## 90%      12.00000     26.0000   2.166667
## 95%      13.00000     30.0000   2.307692
## 99%      17.00000     39.0000   2.294118
## 99.9%    21.00000     81.4850   3.880238
## 99.99%   33.25710    185.4485   5.576208
## 99.999%  35.72571    195.8449   5.481902
## 100%     36.00000    197.0000   5.472222
```

Comparação: Sem coleta X Com coleta

```
print_summary_table("Thumbnailator", nocollect, withcollect)
```

```
## Latency(ms)  Thumbnailator NC  avg: 10 10  | 50: 10 10  | 95: 13 13  | 99: 16 17  | 99.9: 20 24.96
## Latency(ms)  Thumbnailator C   avg: 16 17  | 50: 13 14  | 95: 28 32.99  | 99: 36.2 42.67  | 99.9: 42
```