

# FPC lista 2 - questão 3

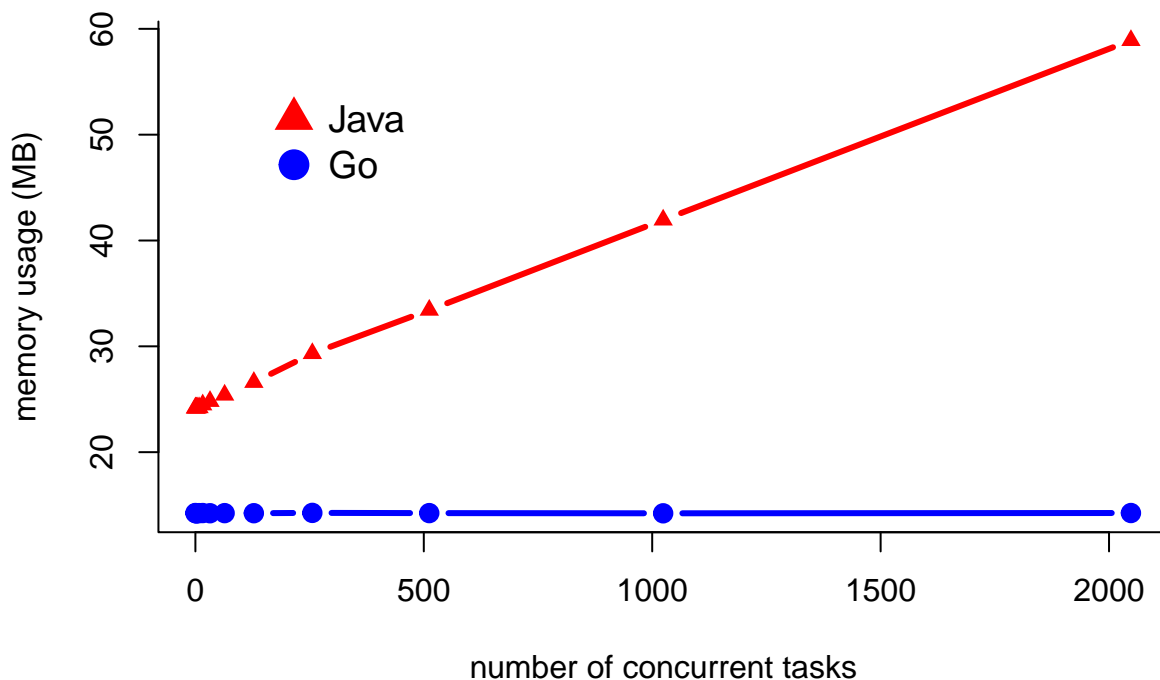
*David Ferreira Quaresma (david.quaresma@ccc.ufcg.edu.br), Renato Dantas Henriques (renato.henriques@ccc.ufcg.edu.br)*

*julho, 2019*

## Descrição

Os experimentos realizados fazem uso de um programa implementado em Go e em Java. O programa escolhido é uma adaptação da solução referente à questão 1A dessa mesma lista e foi escolhido por executar funções concorrentes que apenas dormem, não influenciando no consumo de memória da thread ou goroutine. Executamos 1000 vezes o programa em cada implementação para cada número de execuções concorrentes (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048), resultando em uma amostra com mais de 20 mil dados. Por fim, em cada execução o programa levanta as threads|goroutine e cada dorme por 5 segundos, sendo que após 1 segundo da execução do programa é observado o consumo de memória do programa através do programa PS sob a métrica RSS.

## Análise



No plot acima, os dados coletados dos experimentos associam o número de threads ou goroutines utilizados e o respectivo consumo de memória do programa em uma dada implementação (Java ou Go). Para cada número de threads|goroutine escolhemos a mediana do consumo de memória associado para ser plotado. É

possível notar no plot acima que a medida que aumentamos o número de threads, o consumo de memória de Java aumenta linearmente. Por outro lado, o mesmo não é válido para Go, uma vez que o consumo de memória do mesmo não aparenta ter relação linear com o número de goroutines. Outro dado apontado pelo gráfico indica que a própria runtime do Java consome mais memória que a de Go, isto é observável na posição 0 (zero) do eixo x onde há apenas 1 thread|goroutine em execução.

## Conclusão

Os resultados observados fazem sentido uma vez que as goroutines de Go consomem consideravelmente menos memória que as threads de Java. Isso se dá pois as pilhas de memória das goroutines foram implementadas para melhor ajustar-se ao consumo de memória, enquanto as pilhas das threads de Java possuem tamanho estático. Além disso, o tamanho padrão inicial das pilhas de memória de goroutines é, na versão Go 1.4, de 2KB, enquanto o tamanho padrão das threads de Java é em geral 512KB, podendo variar de acordo com a versão da JVM e SO. Como no design experimental as threads|goroutines executadas pelos programas não consomem memória, é natural esperar que a medida em que o número de concorrência é aumentado o consumo de memória das threads aumente significativamente mais que o das goroutines uma vez que goroutines são absolutamente menos custosas.