Questão 3

Após rodar os dois códigos para avaliar quão leve goroutines em Go em relação à threads em Java, obtemos esses valores de utilização dos recursos da máquina através do comando top que é usado para mostrar os processos da máquina. Ele fornece uma visão dinâmica em tempo real do sistema em execução:

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP
6292	MailCacheDel	0.2	00:00.07	6	5	71-	2188K+	0B	0B	6292
6291	CacheDeleteE	0.0	00:00.07	6	4	54	2432K	0B	0B	6291
6290	iTunesCacheE	0.0	00:00.06	6	5	54	2104K	0B	0B	6290
6289	iBooksCacheD	0.0	00:00.07	6	5	56	1568K	0B	0B	6289
6288	IDECacheDele	0.0	00:01.78	6	5	81	40M	0B	0B	6288
6287	ReportMemory	0.0	00:00.03	3	3	53	1124K	0B	0B	6287
6286	screencaptur	0.3	00:00.18	4	2	150	3912K	0B	0B	6286
6285	screencaptur	3.0	00:00.78	2	1	54	3000K+	620K	0B	279
6284	top	5.0	00:01.40	1/1	0	33	3700K	0B	0B	6284
6283	questao3	173.6	00:26.15	8/4	0	17	5388K+	0B	0B	6273
6273	go	0.0	00:00.28	13	0	39	8432K	0B	0B	6273

Screenshot do comando top na execução do código em Go.

PID	COMMAND	%CPU	TIME	#TH	#WQ	#PORT	MEM	PURG	CMPRS	PGRP	PPID
6322	mdworker	27.1	00:00.55	5	4	60+	6892K+	0B	0B	6322	1
6321	mdworker	27.3	00:00.53	5	4	68+	7192K+	0B	0B	6321	1
6320	screencaptur	0.0	00:00.12	4	2	150	3968K	0B	0B	6320	1
6319	screencaptur	0.3	00:00.34	2	1	56	3064K	620K	0B	279	279
6318	top	4.6	00:00.74	1/1	0	29	3680K	0B	0B	6318	6266
6317	java	99.2	00:13.70	19/1	1	77	9572K	0B	0B	6293	6293

Screenshot do comando top na execução do código em Java.

Após analisar os screenshots acima, podemos afirmar que a utilização da CPU (%CPU) e da memória (MEM) são maiores quando executamos o código em Java. Sendo assim, pode-se concluir que o uso da linguagem Go para sistemas que utilizam concorrência é mais útil do que Java, isso se dá devido ao fato de que a criação de uma goroutine exige pouca memória (apenas 2kB de espaço no stack). Elas crescem alocando e liberando o armazenamento no heap conforme necessário, Threads, por outro lado, começam em 1Mb (500 vezes mais). Além disso, outro fator importante é que o código implementado em Go é simples do que em Java, devido ao fato de que Go Routines exige menos código do que Channels, isso acarreta em problemas para aplicar concorrência em larga escala, pois a complexidade e a manutenção de bases de código fica significativamente grande em Java. Sendo assim, pode haver os temidos *deadlocks* e as *race conditions*, e identificar isso no código pode ser quase impossível. Por fim, Go proporciona criar milhões de Go Routines, enquanto Java proporciona criar dezenas de milhares de Threads apenas.