Entrega 02

Começando os teste com JMeter, utilizando o site modelo. A seguir estarei colocando a forma de configuração do **Web Server** que foi rodado os teste, seguido do **Thread Properties**.

WebServer

- Protocol [http]: https
- Server Name or IP: accor-fab.herokuapp.com
- HTTP Request: GET
- Path: /dev/reviews?merchantId=13&offset=12&limit=3

Nosso testes são baseados em requisições feita do tipo HTTP GET, que está descrito na variável Path descrito anteriormente. Sendo feito diferentes conjuntos de testes.

Conjunto 01:

Onde possui diferentes cenários e as diferenças são os incrementos de número de requisições feitas pelo usuário. Começando em 10000 e terminando em 50000, cada cenário rodando 5 Loop's. As configurações feitas no JMeter estão a seguir.

Thread Properties

Cenário 01:

- Number of Threads (Users): 10000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Cenário 02:

- Number of Threads (Users): 20000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Cenário 03:

- Number of Threads (Users): 30000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Cenário 04:

- Number of Threads (Users): 40000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Cenário 05:

- Number of Threads (Users): 50000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Conjunto 02:

Nesse conjunto de teste fizemos as configurações a aumentar gradativamente o número de Loop’s que as requisições iriam ser feitas. Começando por 1 e indo até 20 Loop’s. Segue a configuração.

Thread Properties

Cenário 01:

- Number of Threads (Users): 10000
- Ramp-up period (Seconds): 1
- Loop Count: 1

Cenário 02:

- Number of Threads (Users): 20000
- Ramp-up period (Seconds): 1
- Loop Count: 5

Cenário 03:

- Number of Threads (Users): 30000
- Ramp-up period (Seconds): 1
- Loop Count: 10

Cenário 04:

- Number of Threads (Users): 40000
- Ramp-up period (Seconds): 1
- Loop Count: 15

Cenário 05:

- Number of Threads (Users): 50000
- Ramp-up period (Seconds): 1
- Loop Count: 20

Conjunto 03:

Nesse teste tinha como objetivo deixar as request’s sendo feita até que chegássemos a uma determinada Error %, com a seguinte configuração.

Thread Properties

Cenário 01:

- Number of Threads (Users): 10000
- Ramp-up period (Seconds): 1
- Loop Count: Infinito

Resultados.

Utilizamos os relatórios três relatórios para basear nossos resultados, sendo o Aggregate Report e o View Results in Table, Viwe Results Tree:

- Aggregate Report - Conjunto 01

Labal	# Samples	Average	Median	90% Line	95% Line	99% Line	Min
GET	50000	234	148	576	588	605	133
GET	150000	234	147	575	587	605	133
GET	300000	233	147	575	586	603	133
GET	500000	232	146	574	586	602	133
GET	750000	232	146	574	586	601	132

- Aggregate Report - Conjunto 02

Labal	# Samples	Average	Median	90% Line	95% Line	99% Line	Min
GET	1000	594	591	602	607	705	560
GET	5000	601	599	621	629	642	560
GET	10000	638	626	703	723	768	564
GET	15000	692	660	794	899	1478	554
GET	20000	703	684	824	898	1038	561
GET	25000	727	685	916	1004	1216	560

- Aggregate Report - Conjunto 03

Labal	# Samples	Average	Median	90% Line	95% Line	99% Line	Min
GET	69069	3132	1570	9235	11275	15479	0

Finalmente chegamos aos finais dos testes, onde podemos observar que nos testes agregados dos dois primeiro conjuntos, não tivemos erros ao fazer as requisições de GET.

Já para o Conjunto 03, obtivemos erros de conexão e conexão excedida, que podemos verificar e visualizar nos relatórios View Results Tree e View Results in Tables.

Conclusão

De acordo com os resultados obtidos, o servidor HTTP testado teve um bom desempenho, com um total de 0.0% de Erros, para os dois primeiros conjuntos, conseguindo fazer todas as requisições com as diferentes cargas estipuladas no teste. No entanto, o conjunto 03 foi onde tivemos o maior tempo e requisições feitas, obtendo um valor de Erro de 16.10% e maior carga de dados recebido.

Sendo assim, para cenário de utilização de grande volume de usuários utilizando o serviço, pode ser necessário a supervisão para que o mesmo continue atendendo durante toda a demanda.