INFORME - LAB 08

ARSW

1. ESCENARIO DE CALIDAD DE LA ESCALABILIDAD

Cuando un conjunto de usuarios consulta un enésimo número (superior a 1000000) de la secuencia de Fibonacci de forma concurrente y el sistema se encuentra bajo condiciones normales de operación, todas las peticiones deben ser respondidas y el consumo de CPU del sistema no puede superar el 70%.

2. COMPARATIVO ESCALABILIDAD VERTICAL VS ESCALABILIDAD HORIZONTAL

A. NEWMAN (PARTE 1)

i. Escalabilidad vertical con size B1ls

Size: B1ls

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 6m 13.4s		
total data received: 1.99MB (approx)		
average response time: 37.2s [min: 21.8s, max: 44.6s, s.d.: 9.9s]		

	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 5m 41.4s		
total data received: 1.4MB (approx)		
average response time: 40s [min: 24.1s, max: 49s, s.d.: 11.6s]		

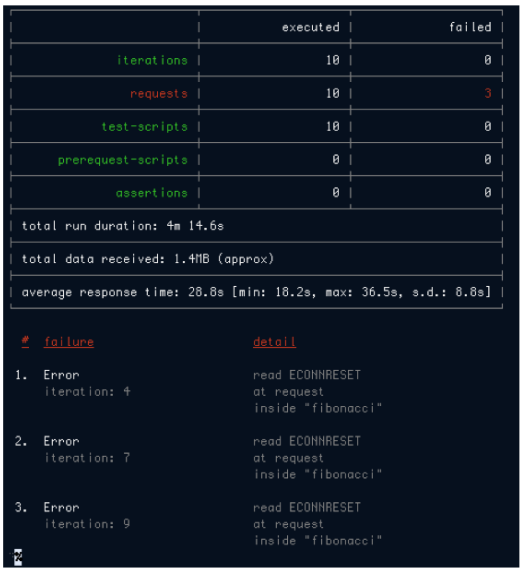
#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 8	read ECONNRESET at request inside "fibonacci"

En las ejecuciones paralelas realizadas se obtuvo en una 10/10 y en la otra 7/10 peticiones exitosas con un tiempo de respuesta promedio de 37.2s y 40s

ii. Escalabilidad vertical con size B2ms

i. Con 2 peticiones en paralelo

Size: B2ms



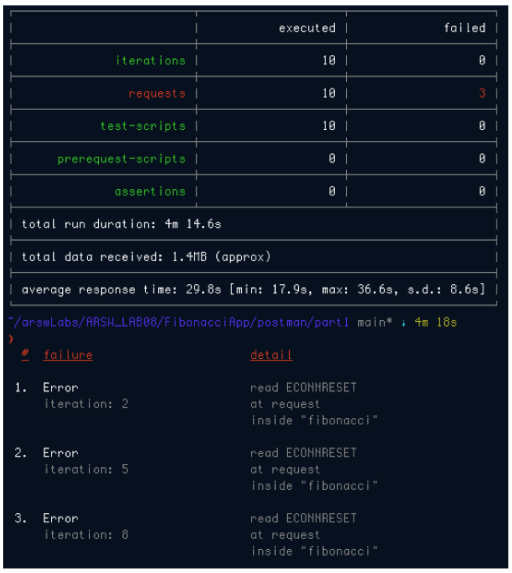
	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 4m 14.6s

total data received: 1.4MB (approx)

average response time: 28.8s [min: 18.2s, max: 36.5s, s.d.: 8.8s]

#	failure	detail
1.	Error iteration: 4	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 7	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 9	read ECONNRESET at request inside "fibonacci"



	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 4m 14.6s

total data received: 1.4MB (approx)

average response time: 29.8s [min: 17.9s, max: 36.6s, s.d.: 8.6s]

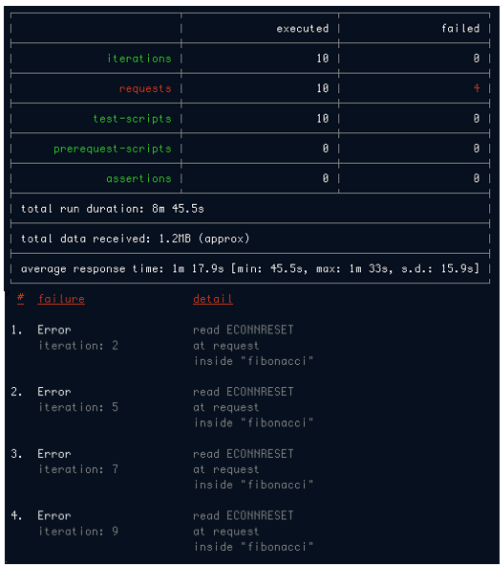
~/arswLabs/ARSW_LAB08/FibonacciApp/postman/part1 main* ↓ 4m 18s

#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 8	read ECONNRESET at request inside "fibonacci"

En las ejecuciones paralelas realizadas se obtuvo 7/10 peticiones exitosas en cada una con un tiempo de respuesta promedio de 28.8s y 29.8s

ii. Con 4 peticiones en paralelo

Con 4 ejecuciones paralelas del comando de postman



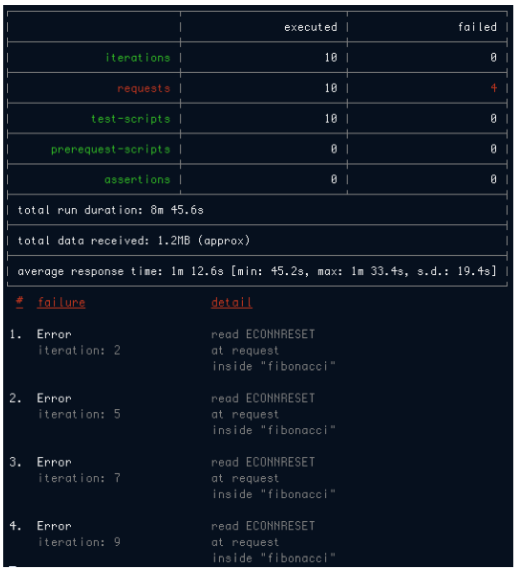
	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 8m 45.5s

total data received: 1.2MB (approx)

average response time: 1m 17.9s [min: 45.5s, max: 1m 33s, s.d.: 15.9s]

#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 7	read ECONNRESET at request inside "fibonacci"
4.	Error iteration: 9	read ECONNRESET at request inside "fibonacci"



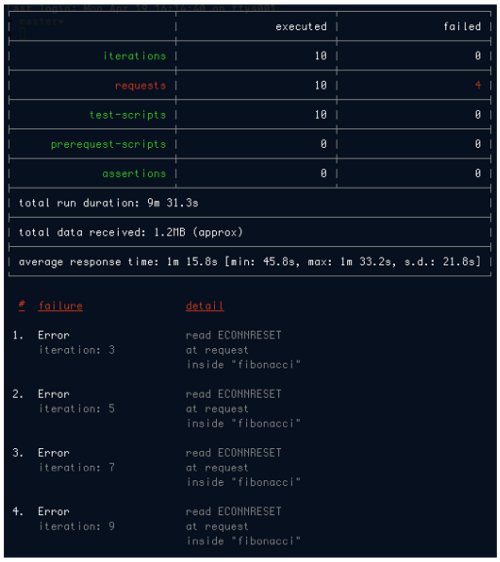
	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 8m 45.6s

total data received: 1.2MB (approx)

average response time: 1m 12.6s [min: 45.2s, max: 1m 33.4s, s.d.: 19.4s]

#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 7	read ECONNRESET at request inside "fibonacci"
4.	Error iteration: 9	read ECONNRESET at request inside "fibonacci"



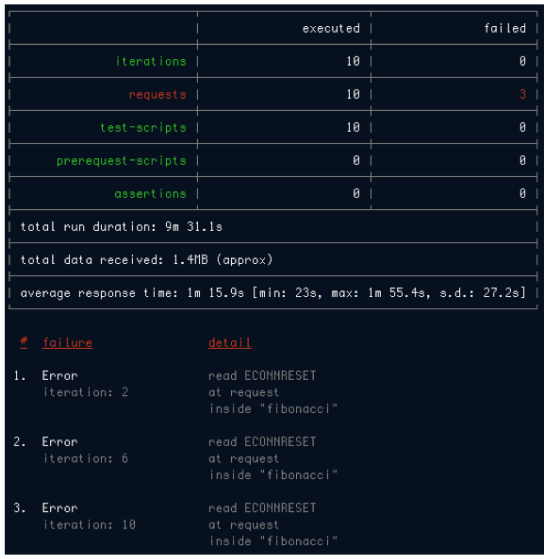
	executed	failed
iterations	10	0
requests	10	4
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 9m 31.3s

total data received: 1.2MB (approx)

average response time: 1m 15.8s [min: 45.8s, max: 1m 33.2s, s.d.: 21.8s]

#	failure	detail
1.	Error iteration: 3	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 5	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 7	read ECONNRESET at request inside "fibonacci"
4.	Error iteration: 9	read ECONNRESET at request inside "fibonacci"



	executed	failed
iterations	10	0
requests	10	3
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0

total run duration: 9m 31.1s

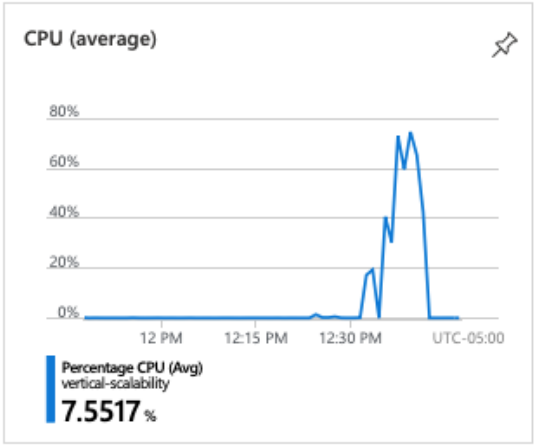
total data received: 1.4MB (approx)

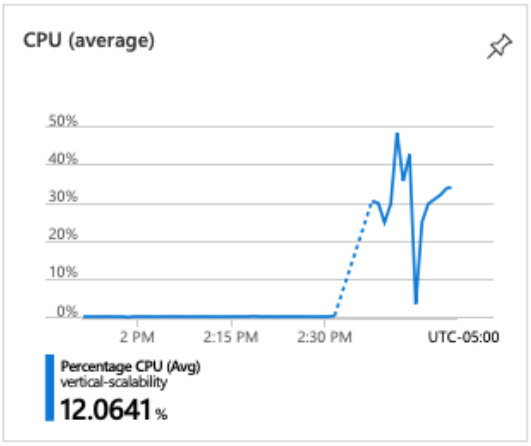
average response time: 1m 15.9s [min: 23s, max: 1m 55.4s, s.d.: 27.2s]

#	failure	detail
1.	Error iteration: 2	read ECONNRESET at request inside "fibonacci"
2.	Error iteration: 6	read ECONNRESET at request inside "fibonacci"
3.	Error iteration: 10	read ECONNRESET at request inside "fibonacci"

En las ejecuciones paralelas realizadas se obtuvo en 3 de ellas 7/10 peticiones exitosas en cada una con un tiempo de respuesta promedio de 12.6s, 15.9s y 17.9s y en la otra 6/10 peticiones exitosas en cada una con un tiempo de respuesta promedio de 15,8 s

Comportamiento de la CPU con tamaño: B1ls vs B2ms





B. NEWMAN (PARTE2)

i. Distribución de carga con 3 máquinas virtuales.

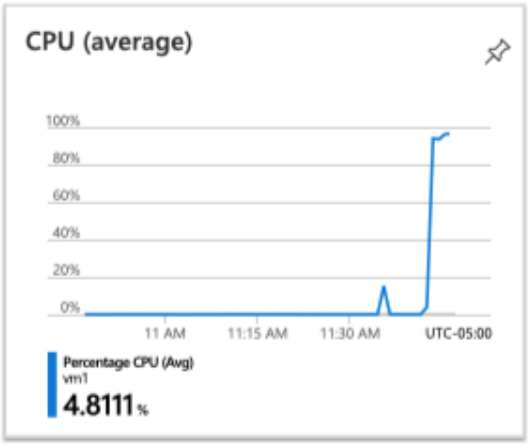
Con 2 peticiones en paralelo

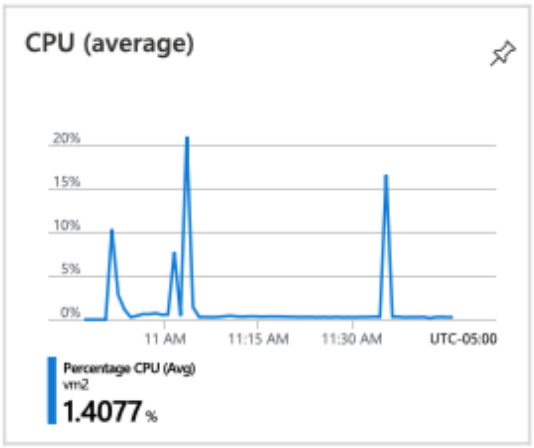
	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 3m 53.7s		
total data received: 1.99MB (approx)		
average response time: 23.2s [min: 22.9s, max: 24.4s, s.d.: 452ms]		

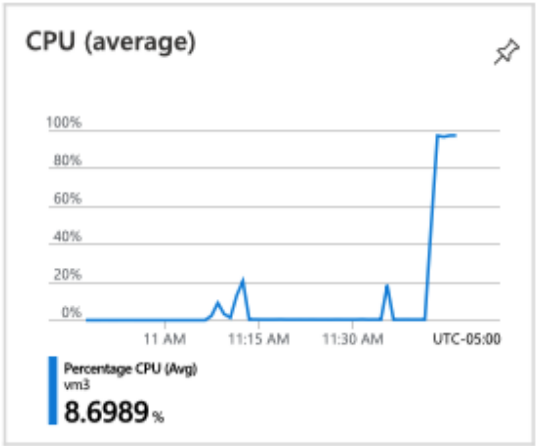
	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 3m 57.1s		
total data received: 1.99MB (approx)		
average response time: 23.6s [min: 23.4s, max: 24.7s, s.d.: 360ms]		

En las ejecuciones paralelas realizadas se obtuvo 10/10 peticiones exitosas en cada una con un tiempo de respuesta promedio de 23.2s y 23.6s

Comportamiento de la CPU de las 3 VM







ii. Distribución de carga con 4 máquinas virtuales

Prueba de newman con 4 peticiones en paralelo

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 3m 49.4s		
total data received: 1.99MB (approx)		
average response time: 22.8s [min: 22.8s, max: 23.1s, s.d.: 86ms]		

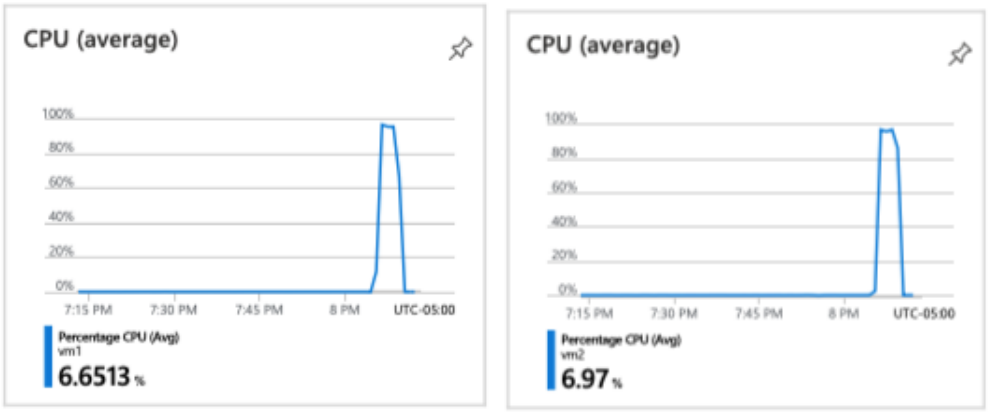
	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 4m 21.4s		
total data received: 1.99MB (approx)		
average response time: 26s [min: 25.8s, max: 26.5s, s.d.: 211ms]		

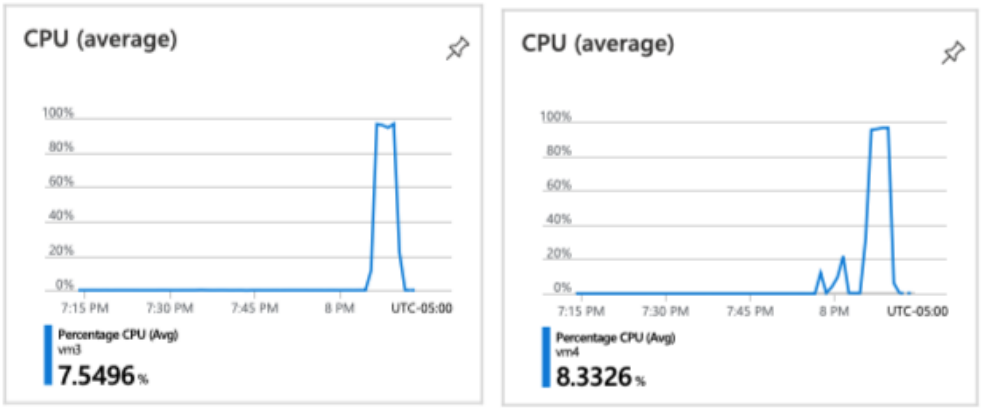
	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 3m 55.9s		
total data received: 1.99MB (approx)		
average response time: 23.5s [min: 23.4s, max: 23.8s, s.d.: 102ms]		

	executed	failed
iterations	10	0
requests	10	0
test-scripts	10	0
prerequest-scripts	0	0
assertions	0	0
total run duration: 4m 22.6s		
total data received: 1.99MB (approx)		
average response time: 26.2s [min: 25.9s, max: 26.6s, s.d.: 199ms]		

En las ejecuciones paralelas realizadas se obtuvo 10/10 peticiones exitosas en cada una con tiempos de respuesta promedio de 22.8s, 23.5s, 26s y 26,2s. Adicionalmente el consumo de CPU del sistema de cada máquina no superó el 70%

Comportamiento de la CPU de las 4 VM





La tasa de éxito de las peticiones aumentó con este estilo de escalabilidad puesto que gracias a su infraestructura y el aumento de recursos (VMs) le permite controlar la carga de la aplicación sin problema. Agregando la cantidad adecuada de recursos (escalado horizontal) podemos controlar un aumento de la carga.

Al repartirse el trabajo entre todos nodos (VMs) cuando el desempeño se ve comprometido con el incremento de peticiones, al añadir nuevos nodos, como en nuestro caso una 4ta máquina al Backend Pool y a medida que es requerido, más y más nodos pueden ser agregados, y así puede responder al número de peticiones concurrentes en cuestión.

3. COSTOS DE LAS 2 INFRAESTRUCTURAS

A. Balanceo de carga horizontal vs una Maquina virtual escalada.

Balanceo de carga horizontal	Maquina virtual escalada (vertical)
○ Un aumento o una reducción del número de instancias de máquina virtual; por lo que este resulta más flexible en un entorno en la nube, ya que puede llegar a ejecutar miles de máquinas virtuales para administrar la carga**.** ○ Soporta la alta disponibilidad y cada zona consta de datos equipados con alimentación, refrigeración y redes independientes. ○ Con el tamaño de B1ls x 4 VMs tiene un costo mensual de aprox $39,94. $16,69 por c/a VM y $23,25 por el load balancer standard.	○ Mantiene el mismo número de máquinas virtuales, pero hace que sean más o menos potentes. ○ La potencia se mide en memoria, velocidad de CPU, espacio en disco, etc; por lo que este tiene más limitaciones, ya que depende de la disponibilidad de hardware de mayor tamaño, que supera el límite rápidamente y puede variar según la región. ○ El escalado vertical también suele requerir que se detenga y reinicie una máquina virtual. ○ Con el tamaño de B2ms se tiene un costo mensual de aprox $66,48.

Fuentes

- https://docs.microsoft.com/es-es/azure/azure-monitor/autoscale/autoscale-overview
- https://azure.microsoft.com/es-es/pricing/calculator/?service=load-balancer