壓力測試報告

- 壓測目的：探索 E-Invoice System 之效能與負載
- 壓測環境：VM n2-standard-2 (vCPUs: 2, RAM: 8GB) on GCP
- 壓測項目：逐步攀升每秒 Request 數，探索巔峰負載
- 壓測情境：
  - 登入功能
  - 直接開立發票 via WEB
  - 訂單開立發票 via API

登入功能

調整效能前：

RPS(每秒 Request 數)	application CPU 用量	database CPU 用量	Request 總耗時		耗時超過一秒 (%)
			p(95)	max	
1	小於 3%	30~33%	35ms	279ms	小於 1%
5	8~13%	150~180%	194ms	705ms	0%
10	13~16%	175~185%	2.05s	2.56s	75%
25	15~20%	175~190%	4.08s	6.25s	99.9%

調整效能後：

RPS(每秒 Request 數)	application CPU 用量	database CPU 用量	Request 總耗時		耗時超過一秒 (%)
			p(95)	max	
1	小於 3%	小於 3%	43ms	91ms	0%
5	11~13%	9~11%	39ms	123ms	0%
10	18~23%	18~23%	45ms	349ms	0%
25	50~56%	50~56%	49ms	271ms	0%
30	68~75%	68~73%	118ms	1.94s	0~1.3%
35	80~85%	98~104%	965ms	2.72s	11~13%

結論：

調整效能後，

- 在相同 RPS 時 postgres CPU 用量為調整效能前的 1/10。
- 巔峰負載為 RPS=35，tomcat CPU 用量為 80%以上，95%的使用者等待時間少於 1 秒。

直接開立發票

設定：spring.datasource.hikari.maximum-pool-size=50

調整效能前

RPS(每秒 Request 數)	tomcat CPU 用量	postgres CPU 用量	Request 總耗時	
			p(95)	max
1	2~5%	小於 1%	46ms	52ms
10	14~19%	7~9%	30ms	47ms
25	36~40%	18~21%	32ms	138ms
30	41~47%	23~25%	32ms	101ms
35(巔峰負載)	47~52%	26~28%	35ms	277ms
40	大於 190%			

調整效能後

RPS(每秒 Request 數)	剩餘字軌組數大於 1500 組				剩餘字軌組數小於 200 組			
	tomcat CPU 用量	postgres CPU 用量	Request 總耗時		tomcat CPU 用量	postgres CPU 用量	Request 總耗時	
			p(95)	max			p(95)	max
1	小於 3%	小於 1%	36ms	49ms	小於 3%	小於 1%	35ms	38ms
10	9~15%	3~5%	28ms	90ms	6~9%	3~4%	25ms	72ms
25	18~28%	9~12%	22ms	78ms	15~21%	8~9%	18ms	61ms
30	20~28%	11~12%	21ms	76ms	18~21%	9~11%	17ms	53ms
35	22~32%	13~15%	21ms	140ms	21~24%	11~13%	17ms	46ms
40	22~31%	14~16%	18ms	51ms	23~26%	12~14%	17ms	44ms
45	27~32%	16~19%	18ms	267ms	27~30%	14~16%	17ms	43ms
50(巔峰負載)	29~36%	17~20%	18ms	54ms	28~32%	15~18%	18ms	53ms
55	大於 190%				大於 190%			

結論：

調整效能後，

巔峰負載為 RPS=50。

95%的使用者等待時間少於 0.1 秒。

在相同 RPS 時，比調整效能前 tomcat 和 postgres 的 CPU 用量減半。

如果調整 spring.datasource.hikari.maximum-pool-size ，也許可以再提高 RPS。

訂單開立發票

spring.datasource.hikari.maximum-pool-size=30

RPS(每秒 Request 數)	tomcat CPU 用量	postgres CPU 用量	Request 總耗時			耗時超過一秒 (%)
			p(90)	p(95)	max	
1	小於 1%	1~2%	22ms	24ms	48ms	0%
25	10~13%	22~26%	21ms	24ms	150ms	0%
50	17~22%	44~49%	23ms	30ms	213ms	0%
100	33~37%	125~140%	674ms	854ms	1530ms	1~4%
105	34~39%	140~145%	911ms	1020ms	1660ms	5~7%
110	35~40%	142~146%	1120ms	1230ms	1780ms	35~40%

spring.datasource.hikari.maximum-pool-size=80

RPS(每秒 Request 數)	tomcat CPU 用量	postgres CPU 用量	Request 總耗時			耗時超過一秒 (%)
			p(90)	p(95)	max	
1	小於 1%	1~2%	22ms	23ms	46ms	0%
25	10~13%	23~26%	21ms	24ms	328ms	0%
50	17~22%	44~49%	23ms	29ms	465ms	0%
100	33~37%	110~142%	735ms	1032ms	2110ms	4~7%
105	34~39%	120~145%	877ms	1215ms	2280ms	6~8%
110	35~40%	135~150%	1270ms	1458ms	3190ms	21~33%

結論：

資料庫最大連接數=30，

巔峰負載為 RPS=105；

95%的使用者等待時間少於 1.02 秒；5~7%的使用者要等待 1 秒以上。

資料庫最大連接數=80，

巔峰負載為 RPS=100（使 RPS 減少 5）；

95%的使用者等待時間少於 1.03 秒；4~7%的使用者要等待 1 秒以上。

tomcat 的 CPU 用量不變，postgres 的 CPU 用量略微下降

雲端機器現況分析

客戶資料

- 使用功能：主要使用"訂單開立發票"
- 業務高峰：每天的餐食(11:00-13:00、17:00-19:00)
- 歷史資料：110 年，每日最多開立 100 萬張發票。

計算方式

- 若 100 萬張發票，分散於餐食 4 小時使用，則每秒約開立 70 張發票
- 最低規格：2 vCPU

建議雲端規格

- 考量到該客戶資料量較大，因此建議規格為最低規格之 3 倍，以因應瞬間資料湧入的情境。

目前雲端規格&建議

- PROD 在 Load Balancer 後有 3 台 VM。
- UAT 在 Load Balancer 後有 1 台 VM。

VM name	Machine type	環境	建議
api-server-prod-01	n2-highmem-4 (4 vCPU)	PROD	調整為 n2-highmem-2
api-server-prod-02	n2-highmem-4 (4 vCPU)	PROD	調整為 n2-highmem-2
api-server-prod-03	n2-highmem-4 (4 vCPU)	PROD	調整為 n2-highmem-2
api-server-uat-01	n2-highmem-4 (4 vCPU)	UAT	調整為 n2-standard-2 或 n1-standard-2