用GHZ壓測，gRPC希望觀察到的壓測數據

註:第一點不用了，直接看第2點

1. ~~吞吐量(RPS)極限值~~

* ~~目的：~~

~~關注在穩定提供服務的前提下，針對該功能、使用主機，每秒最多能夠處理多少呼叫與回應，也觀察平均耗時、主機CPU、Memory的狀況。~~

* ~~模擬線上尖峰時間總呼叫次數(total)，用ghz加倍總呼叫數壓測，觀察全部呼叫都能正常消化的情況下，~~

~~每秒吞吐量極限值到哪(RPS)、Server效能數據又是如何。圖1.1~~

* ~~此觀察焦距在吞吐量極限值，此時增加併發數(concurrency)，對於吞吐量數值並沒有太大的變化(如圖1.2)，~~

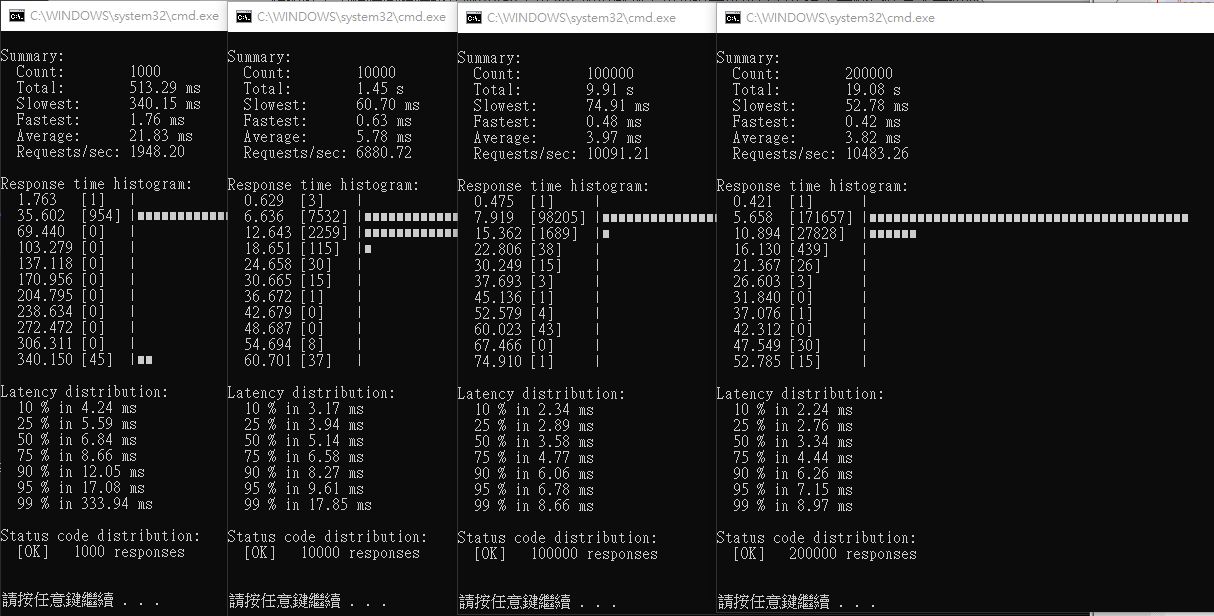
~~故固定併發數ghz defult值50~~

* ~~此測試模式為固定一個總數，讓ghz消化完這個總數的所有值分析出來的數據~~
* ~~補充觀察cpu吃滿的情況下(total設100000)，增加併發數時，吞吐量的表現(如圖1.4,與1.2雷同)~~
* ~~連線數是否影響吞吐量?因ghz連線數不能高於併發數，故實驗將併發數設為1000並遞增連線數觀察~~

~~應該是會被連線建立時間影響數據，故連線數都使用ghz官網默認值1進行，不考慮連線數了(如圖1.3)~~

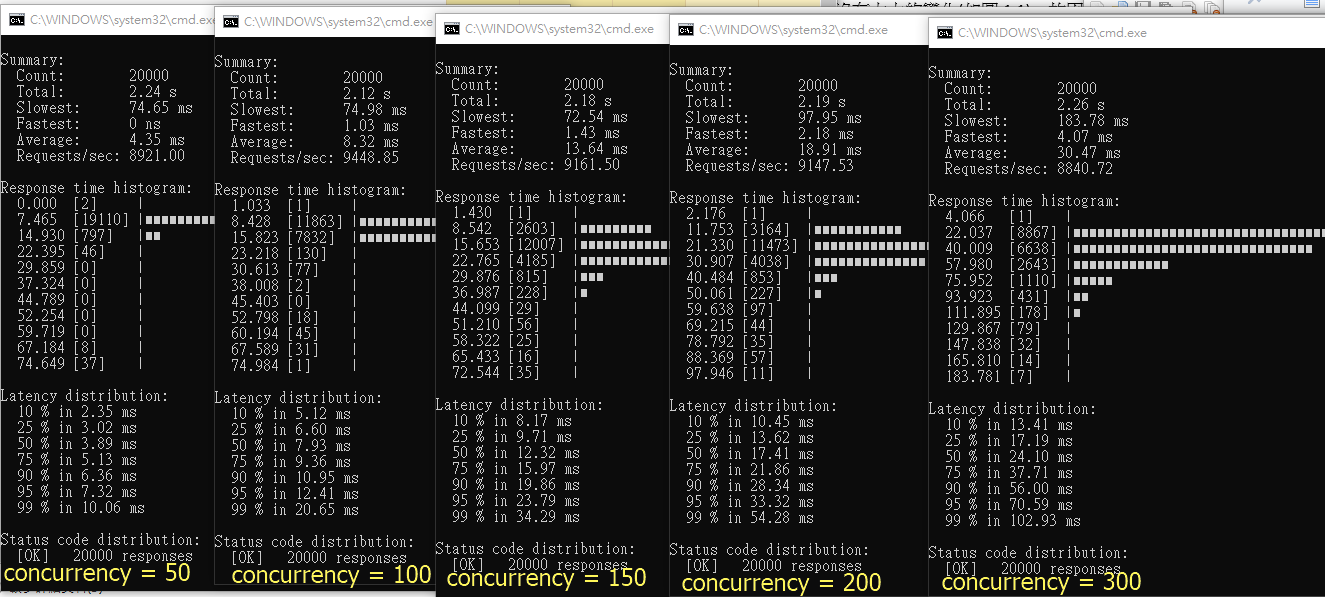
* ~~圖1.1(呼叫數1000、10000、100000(cpu吃滿)、200000，~~

~~觀察到吞吐量(RPS)頂峰值約10000/sec)~~

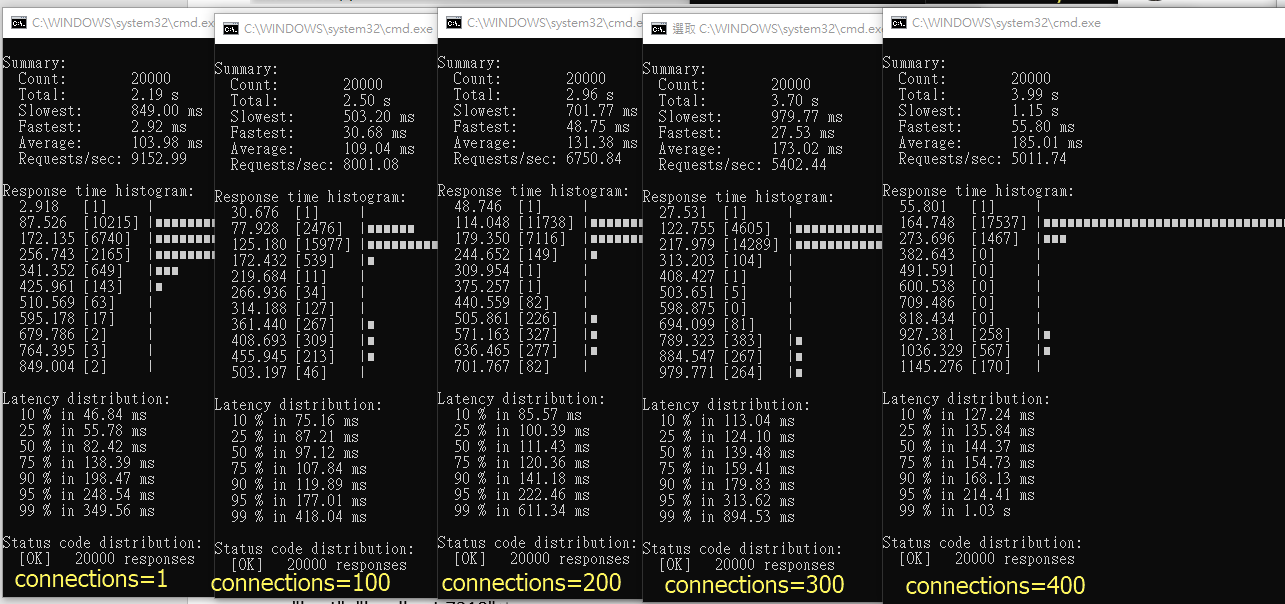


* 圖1.2(以效能未滿載的情況下，實驗增加併發數(concurrency)，觀察吞吐量極限是否有變化，連線數(connections)設預設1，

結論:非關鍵影響值)



* 圖1.3(發現連線數遞增，明顯影響RPS，初判是建立連線時間增加了耗時，故連線數暫不考慮為觀察參數全部都用默認連線1測試)



* 圖1.4



1. 併發數極限值(有點像測瞬間大量能不能吃得下來)
   * 目的：測試瞬間大量呼叫的情況下，使否能持續提供穩定的服務

連續執行一段時間，例如30s，不限制總數，每秒併發量遞增觀察是否都能正常消化(到什麼程度會出現異常)，及耗時平均數、伺服器效能狀況、吞吐量(Request/sec)

~~測吞吐量Ghz Options數值參考~~

~~{~~

~~"proto": "/proto/greet.proto",~~

~~"call": "greet.Greeter.SayHello",~~

~~"total": 20005,~~

~~"concurrency": 1000,~~

~~"connections":1,~~

~~"data": {~~

~~"name": "Joe"~~

~~},~~

~~"skipFirst":5,~~

~~"//max-duration": "30s",~~

~~"enable-compression":true,~~

~~"host": "localhost:7218"~~

~~}~~

測併發量Ghz Options數值參考

{

"proto": "/proto/greet.proto",

"call": "greet.Greeter.SayHello",

"//total": 1005,

"concurrency": 200,

"connections":1,

"data": {

"name": "Joe"

},

"rps":200,

"skipFirst":5,

"//async":true,

"max-duration": "30s",

"duration-stop":"wait",

"enable-compression":true,

"host": "localhost:7218"

}