申論題:

真實環境會是從 DB 讀取課程清單後顯示於網頁上並在選課期間有三萬人同時使用,在學生選完課的當下存入 DB。請問您會如何設計系統架構呢?

Ans:

* 情境一：

server端http request queue超出負荷的上限，假設request queue可等待500個request，server處理500選課的request為1分鐘，但是1分鐘有800個選課request進入，剩下的300個request因為無法等候，所以server會回覆http status code 503 (伺服器過載)。

或者在proxy (Nginx or Apache)假設有預設request在600 ms內必須response，但是server端因為太多資料需要寫入DB，無法將每個request在600 ms內response，這時候proxy會自動回覆504 (Gateway Time-out。

首先在proxy的部分需要將config內可等待的時間加長，例如延長到30秒，server端的timeout也一樣為30秒。在server端的部分，選課的部分需要做的業務邏輯判斷應該是滿少的，所以瓶頸的部分會在每次DB資料寫入的部分，推測使用情境，必須等選課時間結束後，根據所有學生的選課願望清單，排定最後選課分配，因此最後的選課結果並不會馬上產出，但是學生可能會隨時改動選課願望清單，或是有查看排序的需求。所以可以把學生的選課願望清單儲存於像Redis、Rabbit MQ、Kalfka等等Message Queue內，相較於資料庫此類cache server能做到快速讀寫，等到選課時間結束後，在根據這些message產生真正的選課結果後寫入DB，這邊我會想要選用Kalfka，相較於其他兩者，kalfka在message的保存和回朔有更多的支援。

* 情境二：

不使用上述的cache server的話，可以將DB先做垂直切分，就是專門用於選課的DB，比如原本學校的DB會有學生、課程、老師等等table，如果將選課也儲存於這個DB，在選課期間可能會造成其他服務也受到影響，除了獨立選課DB之外，還可做水平切分，就是各學院都會有一個選課DB，但是會增加資料管理上的複雜度。或是可以做DB讀寫分離，例如Master DB Server負責寫入資料，Slave DB會同步Master DB的內容，由Slave DB來負責提供查詢。

* 情境三：

在選課期間加大虛擬機上server和DB的硬體規格，或是增加server的數量，利用load balance減輕大流量，如果時間充裕的話，可以做到auto scaling，這種機器的垂直或水平擴增，就不用靠人力來處理。