高併發概述

Overview of High Concurrency

雲端科技科
林欣柔

目錄
Table Of Contents

請稍後，並避免進行任何操作...

01 . 什麼是高併發？

高 併 發 的 定 義 、 特 徵 、 衡 量 指 標 與 解 決 方 向

01 . 什麼是高併發？

定義

併發數：系統在某一時間點上同時處理的請求數量。

高併發：系統在某一時間點上同時處理大量併發請求。

特徵

大量請求

系統需要同時處理可能來自不同的使用者或客戶的大量請求。

同時存取

這些請求幾乎同時到達系統。

資源競爭

CPU、記憶體、網路頻寬等系統資源可能面臨競爭與爭用。

In network applications, high concurrency refers to a large number of simultaneous requests accessing the server or proxy server at the same time.

These requests may come from different users, applications, or systems, and processing them simultaneously may have an impact on the performance and stability of the server or proxy server.

Therefore, high concurrency is often regarded as an indicator that tests the capability of a server or proxy server.

資料來源：https://blog.twjoin.com/

01 . 什麼是高併發？

1.主要使用什麼衡量指標衡量高併發？

2.衡量指標到達什麼程度才算是高併發？

衡量指標

表一、高併發各項衡量指標

類別 \ 指標與說明	指標	說明
Throughput	QPS (Queries Per Second)	系統每秒可以處理的**查詢**請求數
	TPS (Transactions Per Second)	系統每秒可以處理的**交易**請求數
	RPS (Requests Per Second)	系統每秒可以處理的**API**請求數
Resource Utilization	CPU	衡量處理器在執行工作負載時的利用程度
	RAM	反映系統記憶體的占用情況
	Network	描述網絡頻寬的利用情況
Latency	Latency	**伺服器收到請求到做出回應**所花費的時間
	RT (Response Time)	**用戶發出請求到收到回應**所花費的時間

01 . 什麼是高併發？

解決高併發問題的三大方向

1. 高效能

體現系統的平行處理能力，反應使用者體驗。

衡量指標： RT

2. 高可用

表示系統可以正常服務使用者的時間。

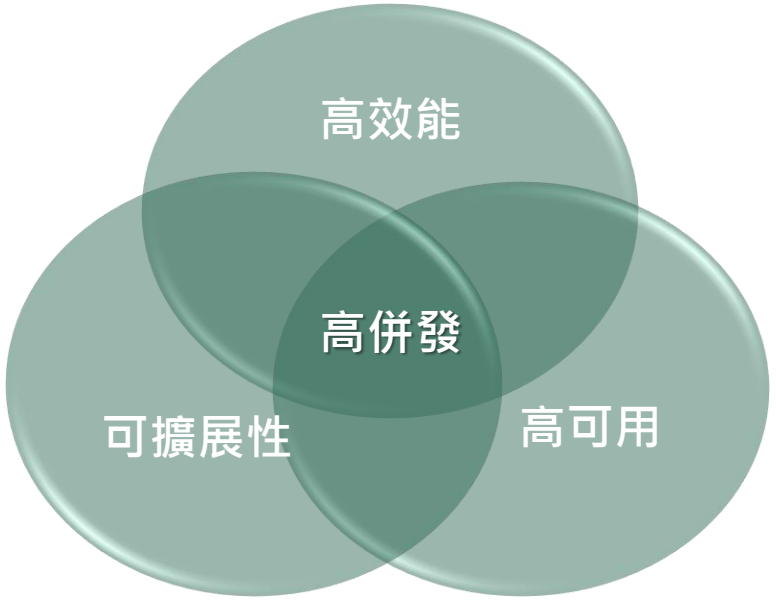
衡量指標： Availability = MTBF / (MTBF + MTTR)

3. 可擴展性

系統的擴展能力越高，效能與可用性都會提升。

可以分成垂直、水平擴展兩種方式。

註：1. MTBF（Mean Time Between Failure）：平均故障間隔
2. MTTR（Mean Time To Repair）：平均故障時間



圖一、高併發、高效能、高擴展、高可用的關係

表二、系統可用性指標對應故障時間

系統可用性	年故障時間	日故障時間
90%	36.5天	2.4小時
99%	3.65天	14.4分鐘
99.9%	8小時	1.44分鐘
99.99%	52分鐘	8.6秒
99.999%	5分鐘	0.86秒

例如： DynamoDB global table保證高達 99.999% 的可用性

02. 高併發案例

- 售票系統

關鍵雲端服務介紹、系統架構圖

02. 高併發案例

解決方向 以本案例AWS服務所使用的solution為例

表三、高併發解決方向涵蓋之類別與實作

解決實作(AWS服務) \ 解決方法類別	擴展(分散式架構)	快取	資料庫 高可用	非同步 高效能
Route53 + CloudFront	✓	✓		
AutoScalingGroup + Load Balancing	✓			
ElastiCache	✓	✓		
DynamoDB	✓		✓	✓

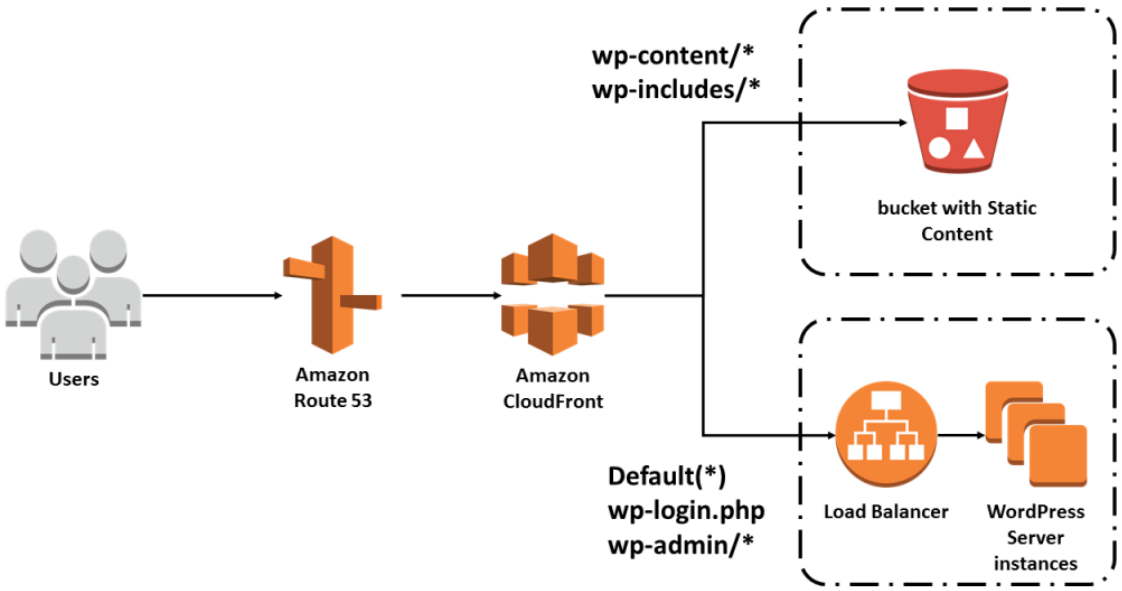
02. 高併發案例

CloudFront / Route53

Route 53：**DNS服務**，負責註冊域名、域名解析、流量路由與檢查資源的運作狀況。

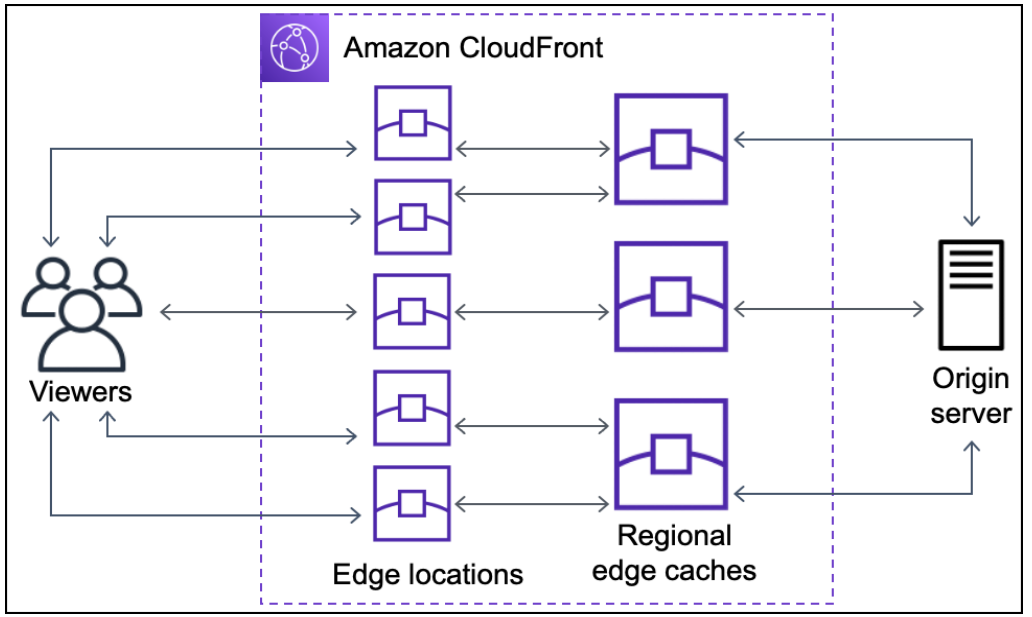
CloudFront： **CDN服務**，負責靜態與動態內容的全球分發，降低延遲。**➔例如：公司簡介頁面**

使用CloudFront的優點：快取靜態內容，提升效能 / 快速交付動態內容。



圖四、 快取靜態資源的流程

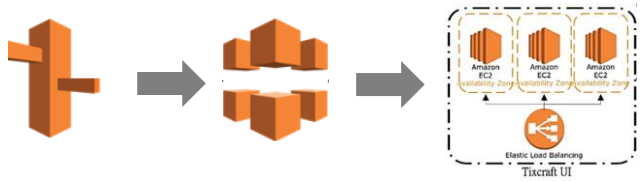
(資料來源： https://isotropic.co/what-is-cloudfront-net/)



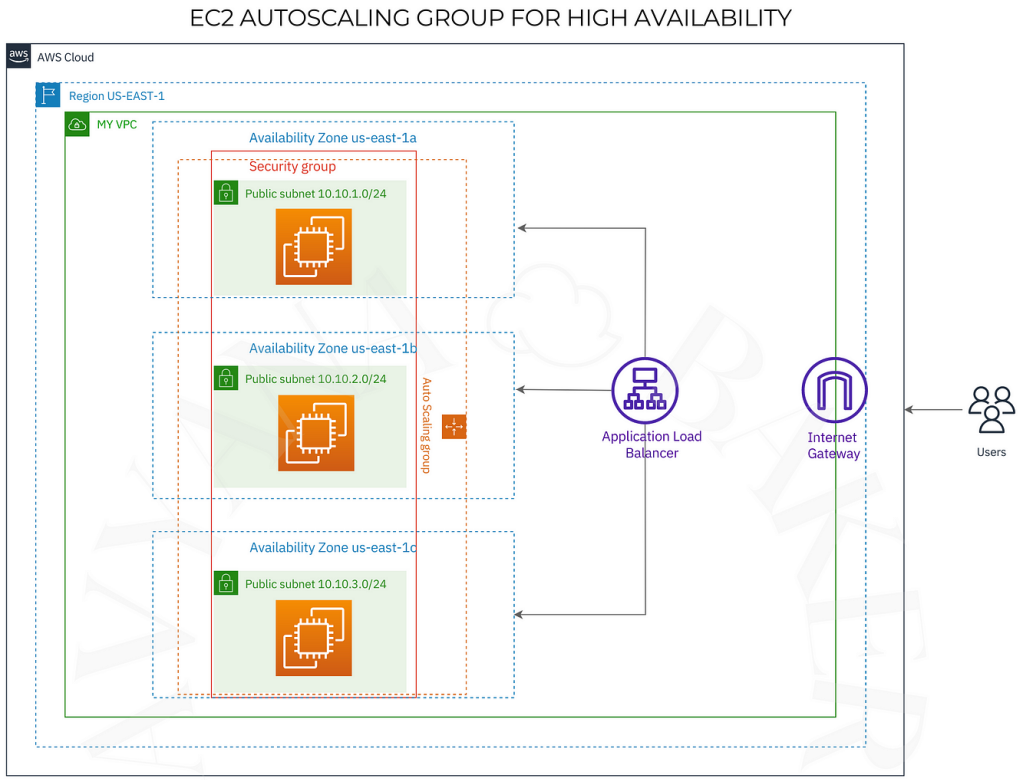
圖五、 CloudFront Edge location 與 Reginal edge cache

(資料來源：AWS)

02. 高併發案例



Auto Scaling groups + Elastic Load Balancing



圖二、ELB 與 Auto Scaling groups 常見架構

(資料來源：https://aws.plainenglish.io)

Auto Scaling Groups：用於管理、**自動擴展實例**的數量。

Elastic Load Balancing (ELB)：**分配流量**到多個運行正常的實例。

➔ **例如：搶票API、UI**

Instance：部分使用EC2 Spot Blocks，成本更低，保證持續1-6小時。

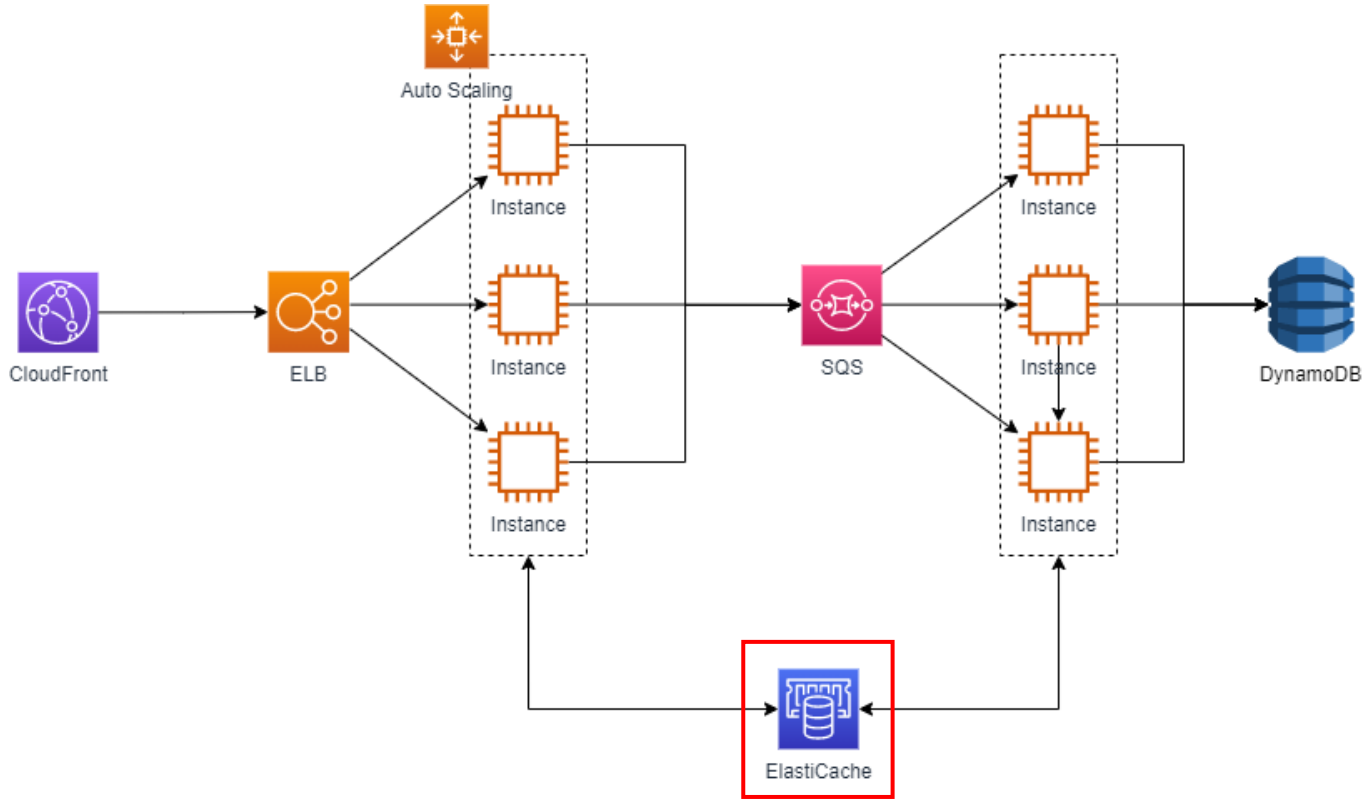
使用Auto Scaling groups + ELB 的優點

- 尖峰負載自動擴展，需求降低自動縮減，避免資源浪費。
- 保證流量分配穩定，將流量路由到運作狀況良好的目標。
- 可設定多可用區，避免單一區域故障。

02. 高併發案例

ElastiCache

ElastiCache：全託管的分散式快取服務。



圖三、Elasticache 工作流程

(資料來源： https://ithelp.ithome.com.tw/m/articles/10297127)

為何需要快取

- 高併發場景下讀取需求高。

➔ 例如：每個搶票的請求都要先讀取目前剩餘的門票張數。

使用Elasticache分散式快取的優點

- 自動跨多個可用區備份提升可用性。
- 分片提升讀寫效率與單節點容量。

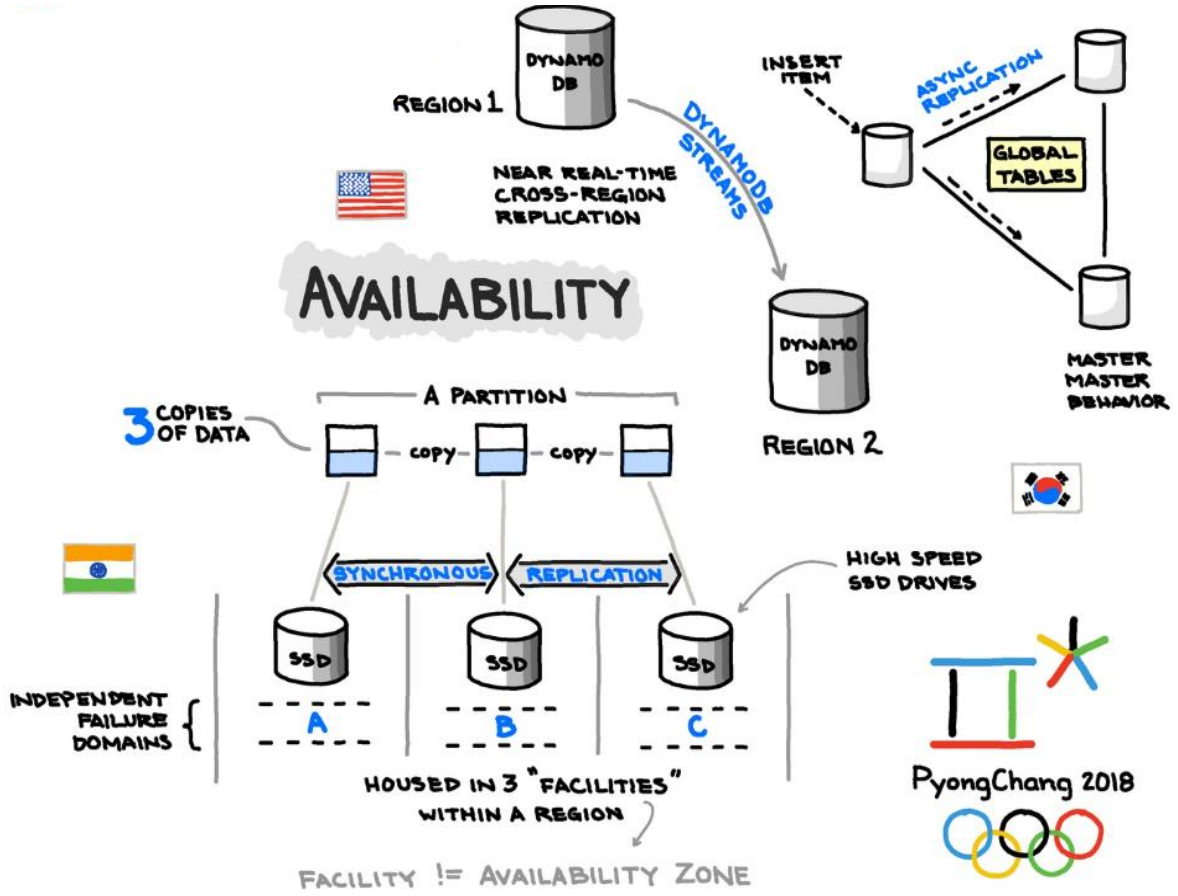
使用快取須注意

- 快取命中率。
- 快取穿透問題。
- 資料不一致。

02. 高併發案例

DynamoDB

DynamoDB：全託管的NoSQL資料庫。



圖六、 DynamoDB global tables

(資料來源：https://www.awsgeek.com/Amazon-DynamoDB/)

為何要使用非關聯式資料庫？

- 沒有schema限制，易於水平擴展。
- 資料間不含關聯性，查詢速度快。

使用DynamoDB 的優點

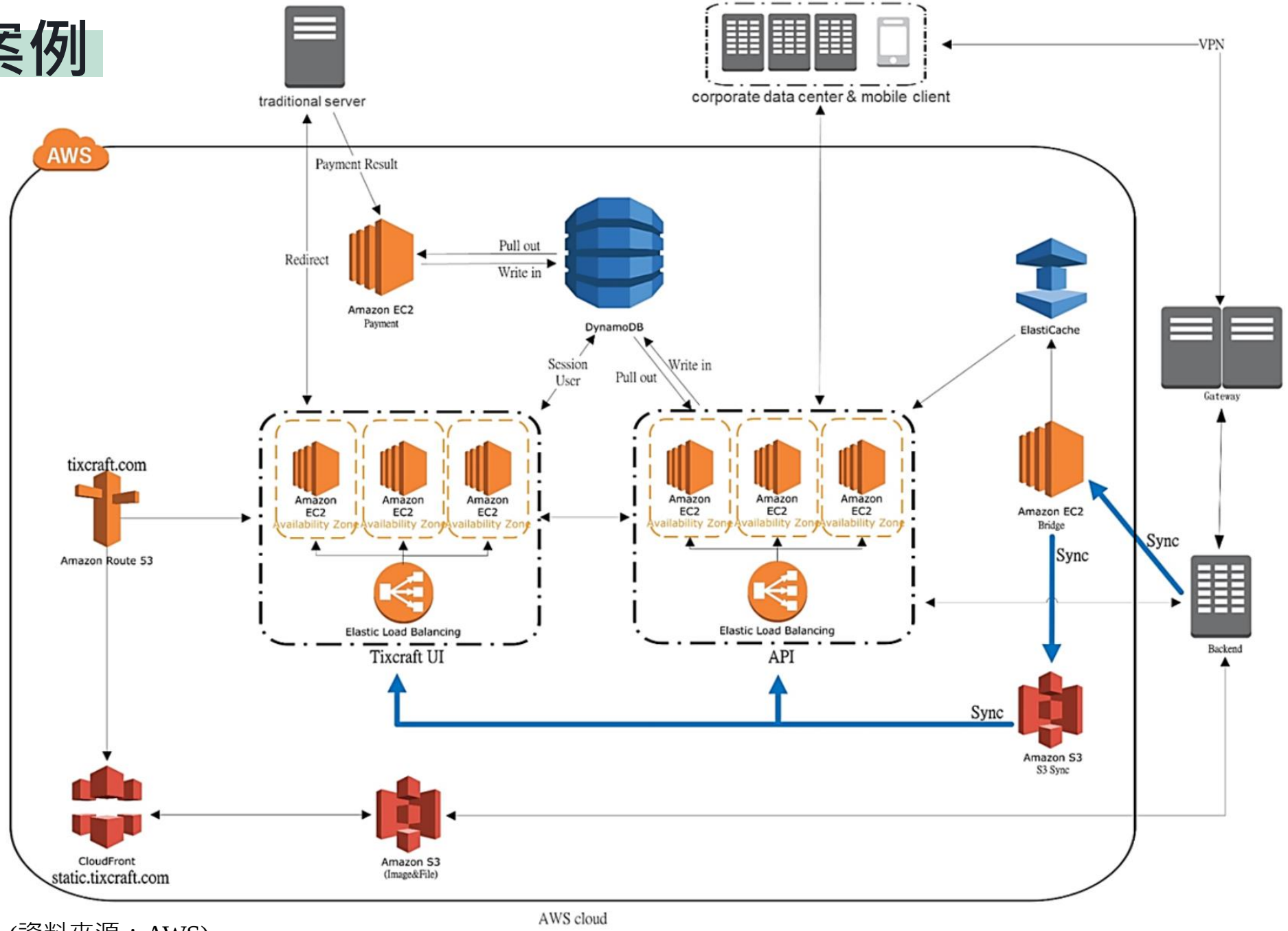
- 區域強一致性(ACID 事務)。
- 自動分片確保數據分佈均勻。
- 無限制擴展，自動分配 RCU/WCU 。
- 提供一致的個位數毫秒效能。

➔ 例如：I/O：20 => 135,000 ；100,000 users

- global tables：五個九的可用性。

02. 高併發案例

架構圖



圖七、 拓元售票系統架構圖 (資料來源：AWS)

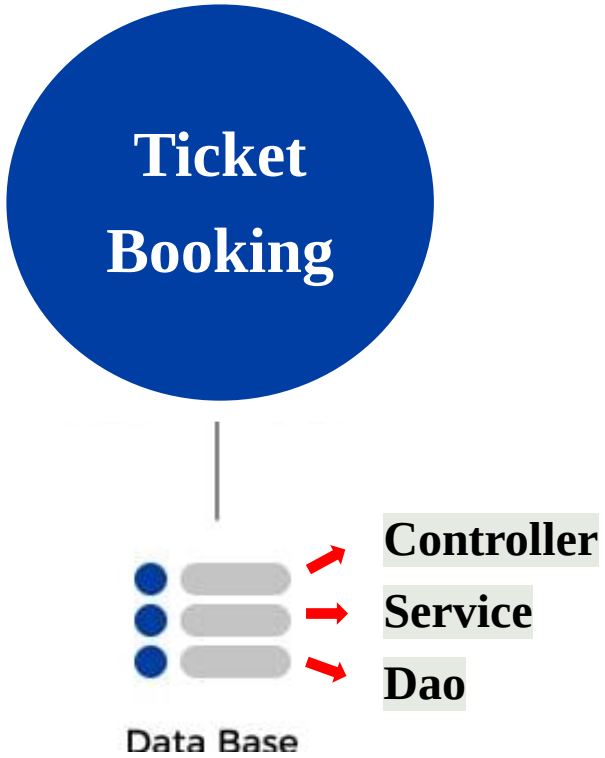
03. 其他常見的高併發解決方案

微服務、訊息佇列、快取與資料庫

03. 其他常見的高併發解決方案

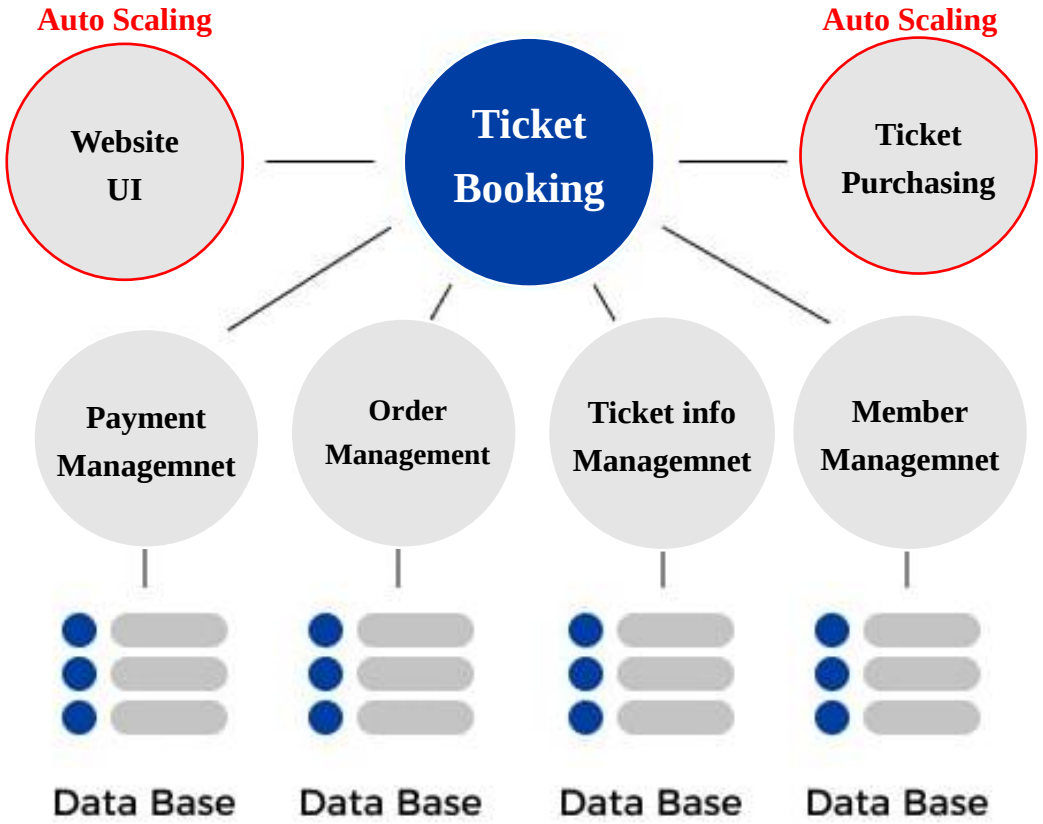
系統分層與微服務

系統分層：針對流程進行拆分，每層專注於特定的功能，透過上下層的依賴和調度組成一個完整的系統。



圖八、系統分層架構

微服務：針對業務進行切分，將應用拆分為多個小型、獨立的服務，每個服務專注於特定的業務功能。



圖九、微服務架構

03. 其他常見的高併發解決方案

訊息佇列

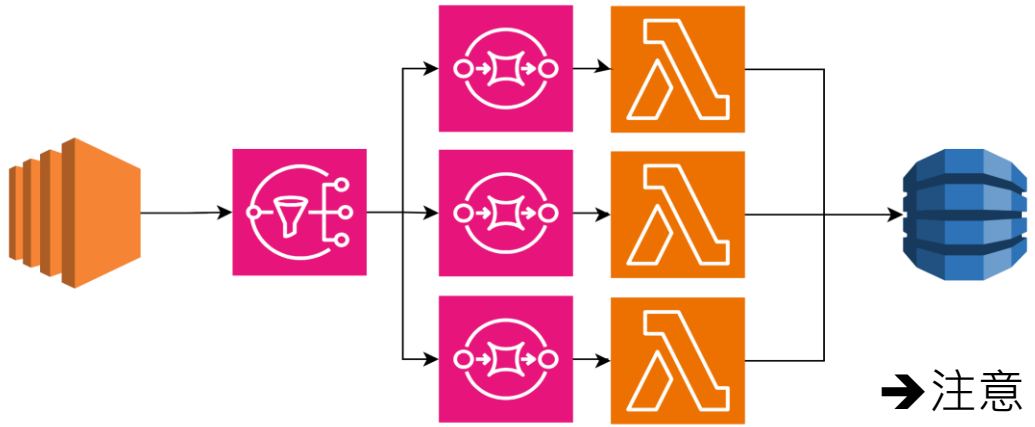
訊息佇列：分散式架構中的通訊機制，Producer-Consumer model的一種實現。

3大功能

削峰填谷：削去峰值流量，業務邏輯的處理更加緩和。

非同步：分離業務流程中的步驟，提升系統效能。

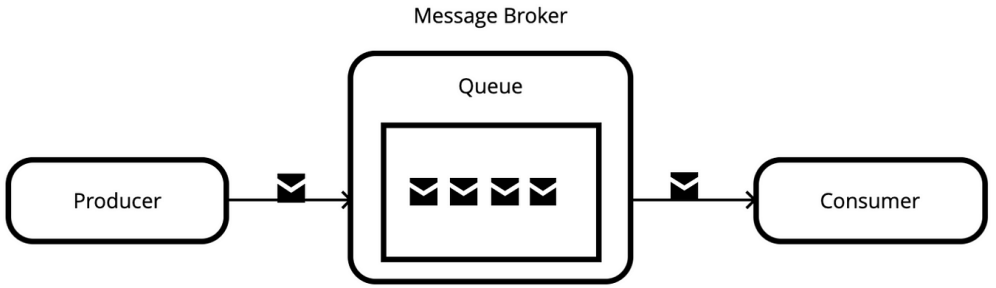
解耦：一個系統的變更或損壞不會影響另一個系統。



圖十一、訊息佇列削峰填谷

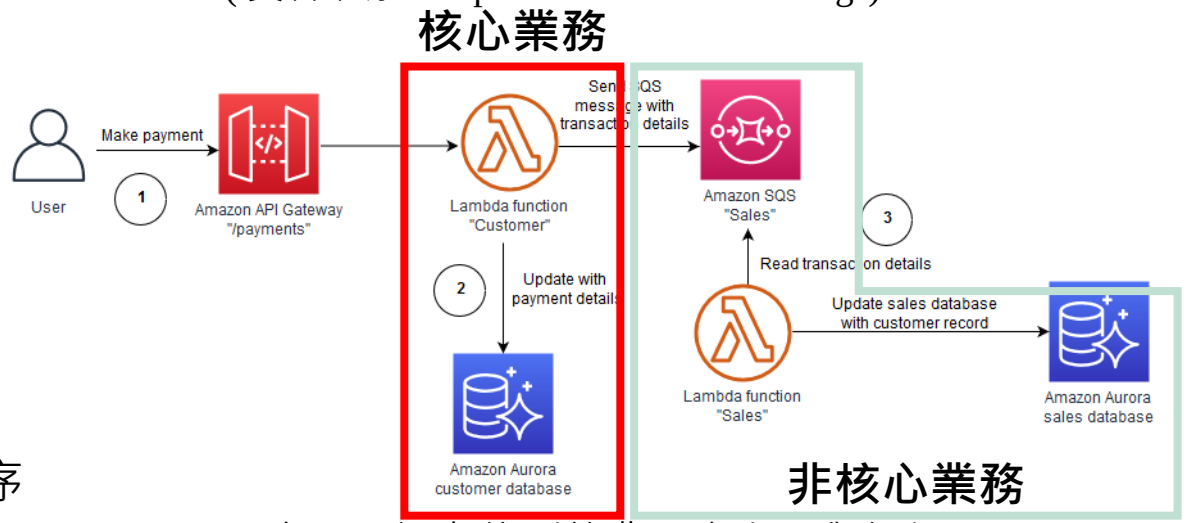
(資料來源：AWS)

➔注意：
全局順序
堆積
延遲...



圖十、Message Queue 的工作原理

(資料來源：https://medium.com/starbugs)



圖十二、訊息佇列的非同步處理與解耦

(資料來源：AWS)

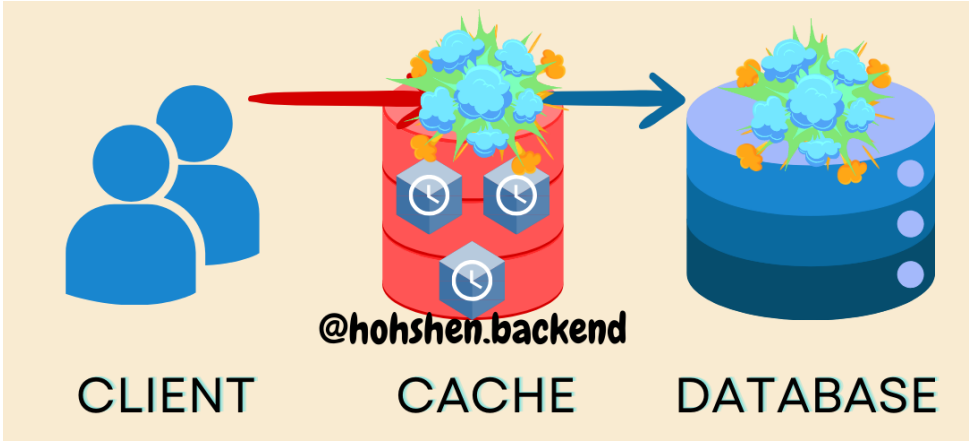
03. 其他常見的高併發解決方案

快取

快取穿透
(Cache Penetration)

快取雪崩
(Cache Avalanche)

快取擊穿
(Hotspot Invalid)



圖十三、快取攔截資料庫請求失敗

(資料來源：https://medium.com/@hohshencode)

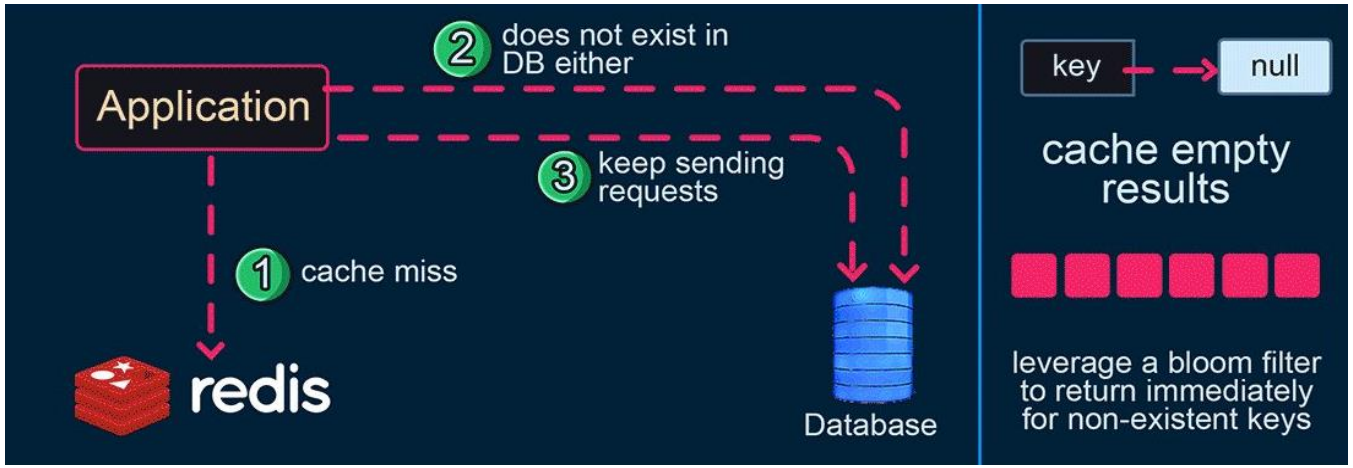
03. 其他常見的高併發解決方案

快取

快取穿透(Cache Penetration)：資料不在快取也不在DB。

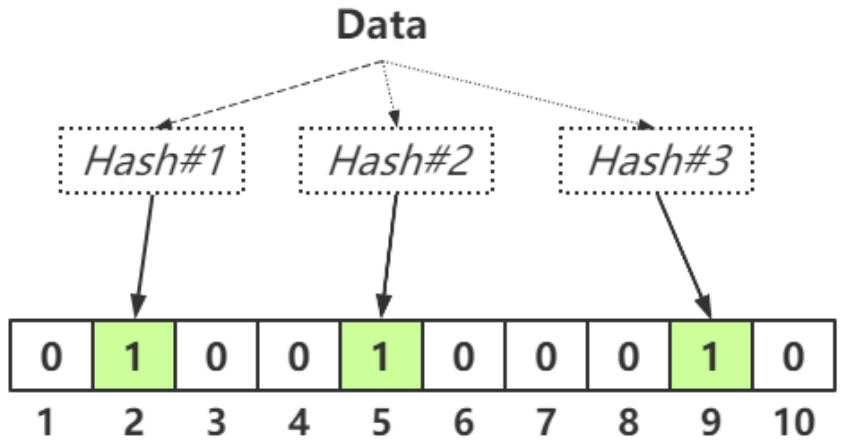
➔例如：未註冊過的會員資料、設定錯誤的票據 ID

- 解法1：回種空值+快速過期
- 解法2：布隆過濾器(Bloom filter)



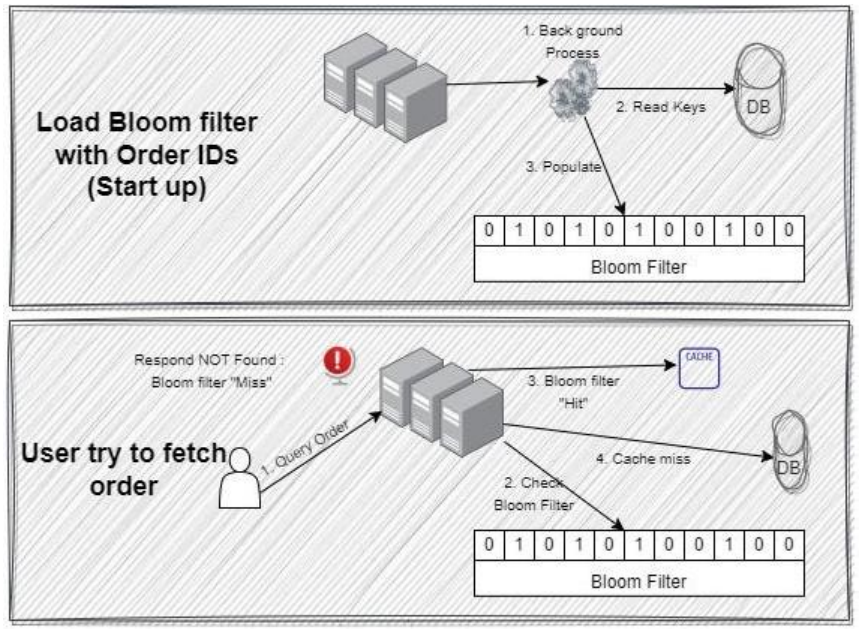
圖十四、快取穿透示意圖

(資料來源：https://x.com/alexxubyte?lang=zh)



圖十五、布隆過濾器計算過程

(資料來源：https://www.cnblogs.com)



圖十六、布隆過濾器工作流程

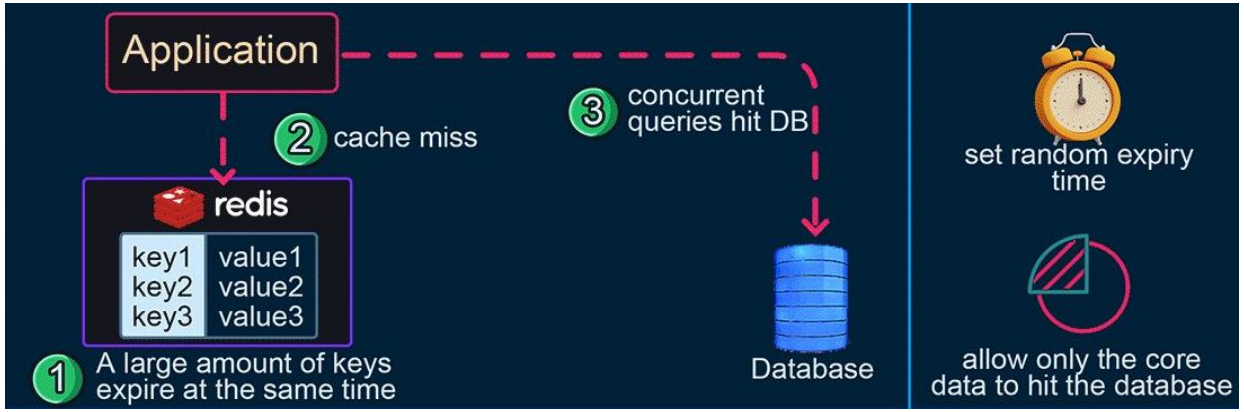
(資料來源：https://harish-bhattbhatt.medium.com/)

03. 其他常見的高併發解決方案

快取

快取雪崩(Cache Avalanche)：資料在DB，但因為大量快取同時過期，導致DB亦同時接收大量請求。

- 解法1 ：設定快取資料的隨機過期時間
- 解法2：互斥鎖

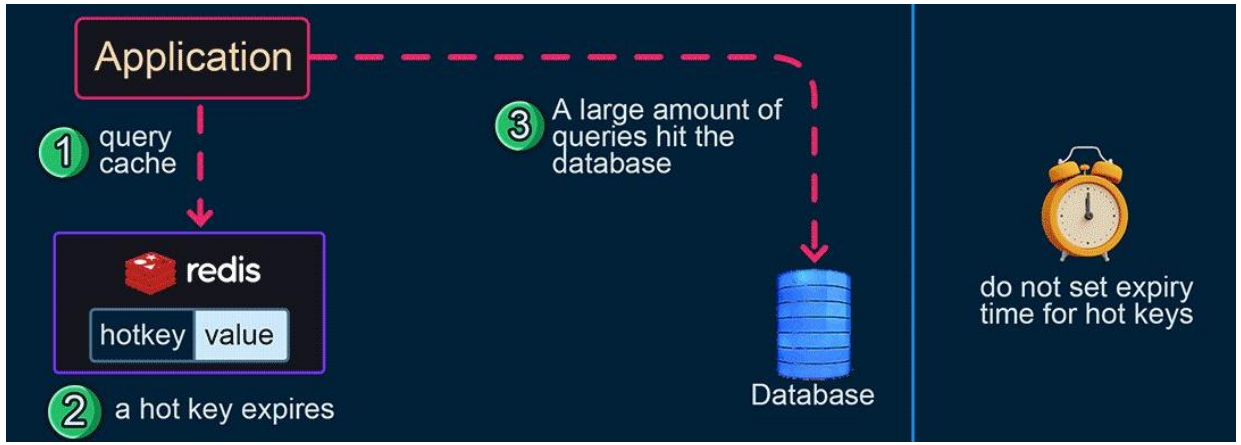


圖十七、布隆過濾器

(資料來源：https://x.com/alexxubyte?lang=zh)

快取擊穿(Hotspot Invalid)：資料在DB，但因為單個熱點快取過期，導致DB同時接收大量請求。

- 解法1 ：不給熱點資料設定過期時間
- 解法2： 互斥鎖



圖十八、布隆過濾器

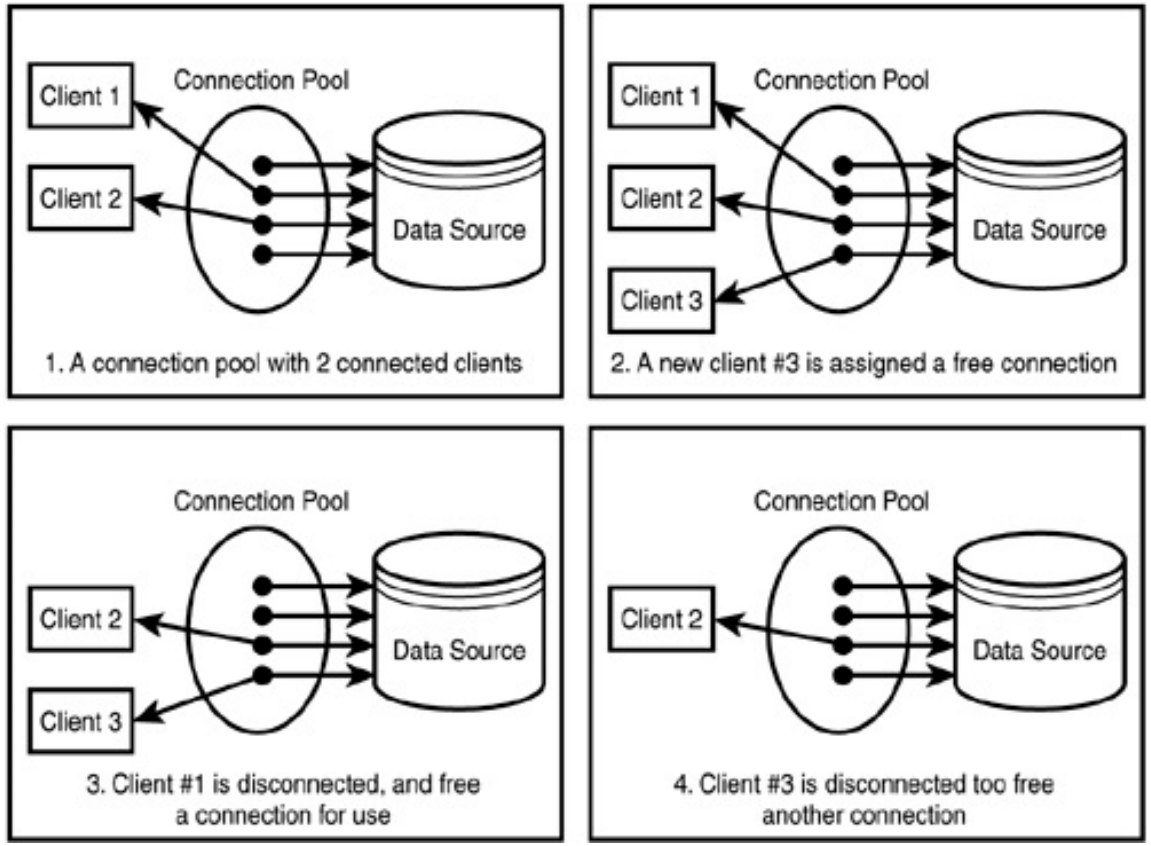
(資料來源：https://x.com/alexxubyte?lang=zh)

03. 其他常見的高併發解決方案

資料庫

Connection Pool：預先創建好資料庫連線，以減少每次新建立連線的耗時。

一種利用空間交換時間的做法。



圖十九、資料庫連接池的工作原理

(資料來源：https://ejbvn.wordpress.com)

目的

- 避免每次的請求都須重新建立連線
- 連線與異常管理

優點

- 加快查詢速度
- 減少效能損耗
- 集中管理資料庫連線

缺點

- 空間浪費
- 增加啟動時間

03. 其他常見的高併發解決方案

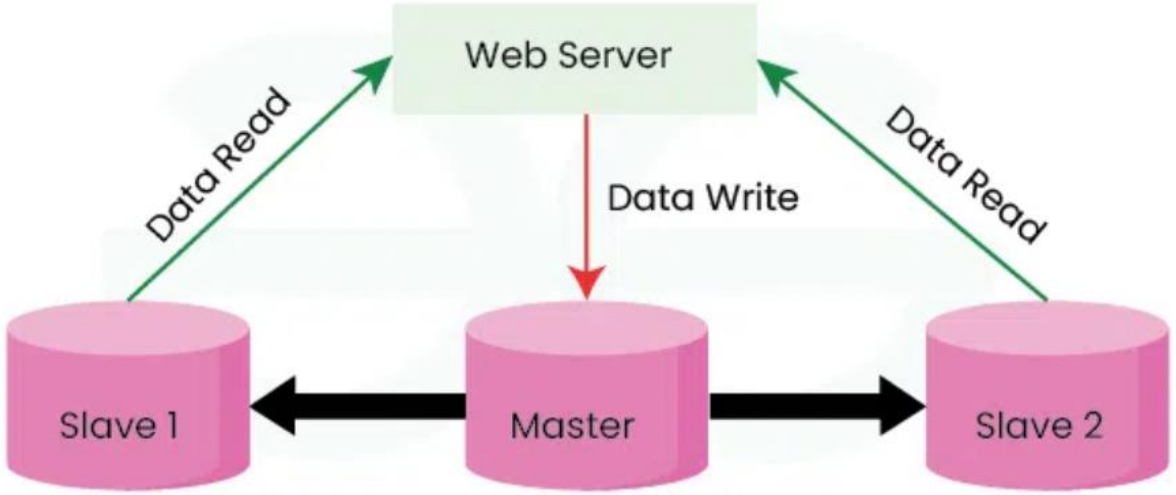
資料庫

主從架構、一主多從 (Master Slave Replication) ：將資料庫分為用來寫入的主庫，以及用來讀取資料的從庫，並將單一從庫水平擴展成多個從庫。

目的：讀寫分離、水平擴展

優點：提高讀取效能、緩解主庫壓力、災難復原(提升可用性)

缺點：主庫單點故障問題、主從庫資料不一致、增加開發與維運複雜度



圖二十、Master Slave Replication架構

(資料來源：geeksforgeeks)

03. 其他常見的高併發解決方案

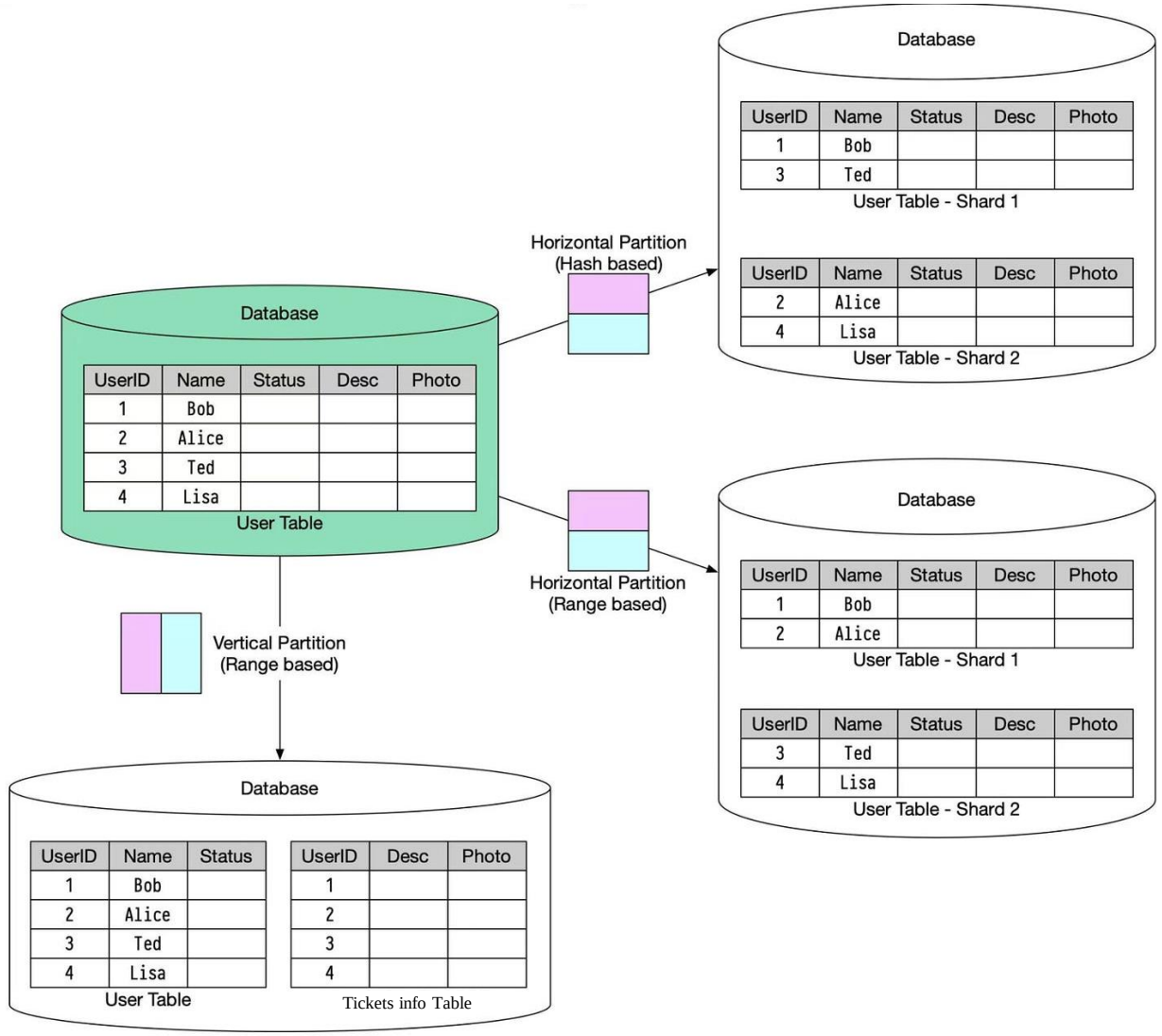
資料庫

分庫 / 分表：

依照某種策略將資料盡量平均的分配到多個資料庫節點或多個表中

拆分方式

- **垂直拆分**：依據**業務相關性**拆分，將耦合度比較高的表拆分到單獨的庫或表中
- **水平拆分**：依據**資料特性**進行拆分，可解決一個業務模組的資料大量膨脹的問題



03. 其他常見的高併發解決方案

資料庫

分片(sharding) / 分區(partition)

- **以部署方式區分**：分片將資料分佈在多台伺服器上，而分區則在一台伺服器內拆分表。
- **以範圍、方向區分**：分區是資料庫劃分資料的通用術語 - 分片是水平分區的一種。
- **以資料結構區分**： - 分片(水平分區)不改變資料結構；垂直分區則改變了資料結構

分庫、分表、分區、分片的優勢

- 解決儲存容量瓶頸 (分表、分區、分片)
- 提升讀取 / 寫入的效能
- 負載平衡：降低單一節點成為瓶頸的可能性 (分片)。

涉及資料遷移成本、難度高，沒有瓶頸就不做；做了就要以長期規劃為主。

04 .總結

-

04. 總結

高併發解決方案的歸納

表四、高併發解決方案歸納

解決實作 \ 解決方法類別	擴展(分散式架構)	快取	資料庫 高可用	非同步 高效能
CDN	✓	✓		
微服務	✓			
自動擴展與負載平衡	✓			
訊息佇列				✓
動態快取	✓	✓		
非關聯式資料庫	✓		✓	
Connection Pool			✓	
主從架構(Master-Slave)	✓		✓	
分庫 / 分表 / 分區 / 分片	✓		✓	

04. 總結

高併發四大原則

圖二十二、常見高併發解決方案歸納

簡報到此結束

2024 / 12 / 13

雲 端 科 技 科
林 欣 柔