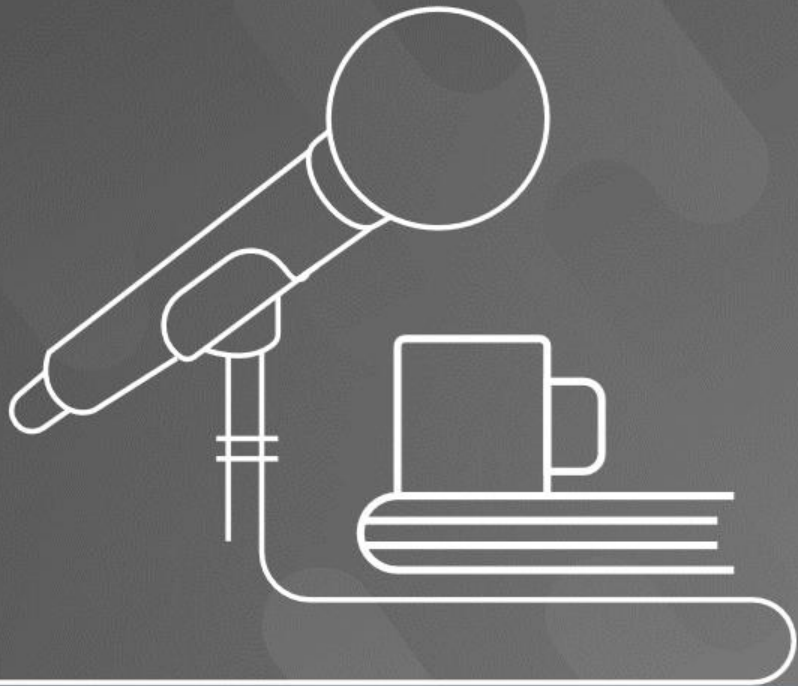




STUDY4.TW

為 學 習 而 生

STUDY4 DEV 8 月活動



活動日期 2020/08/23 下午 2 點

活動地點  Build School 台北教室

後端工程師的修練之路

雲端時代，後端工程師要學會的技能琳琅滿目，光是 Azure / AWS / GCP 的服務加起來就超過 1000 個，熱門的程式語言也有幾十種，開發框架也有上百種...

實在都學不完啊！在後端開發技術百家爭鳴的時代，該掌握那些技能才能維持自身的競爭力？這場議程，安德魯將分享過去 20 年在後端打滾的心得，給每位有志從事後端工程師，以及想要晉級到架構師的朋友們參考。



Andrew Wu

後端工程師的修練之路

Andrew Wu, 2020/08/23

投影片下載: <https://github.com/andrew0928/Meetup/>

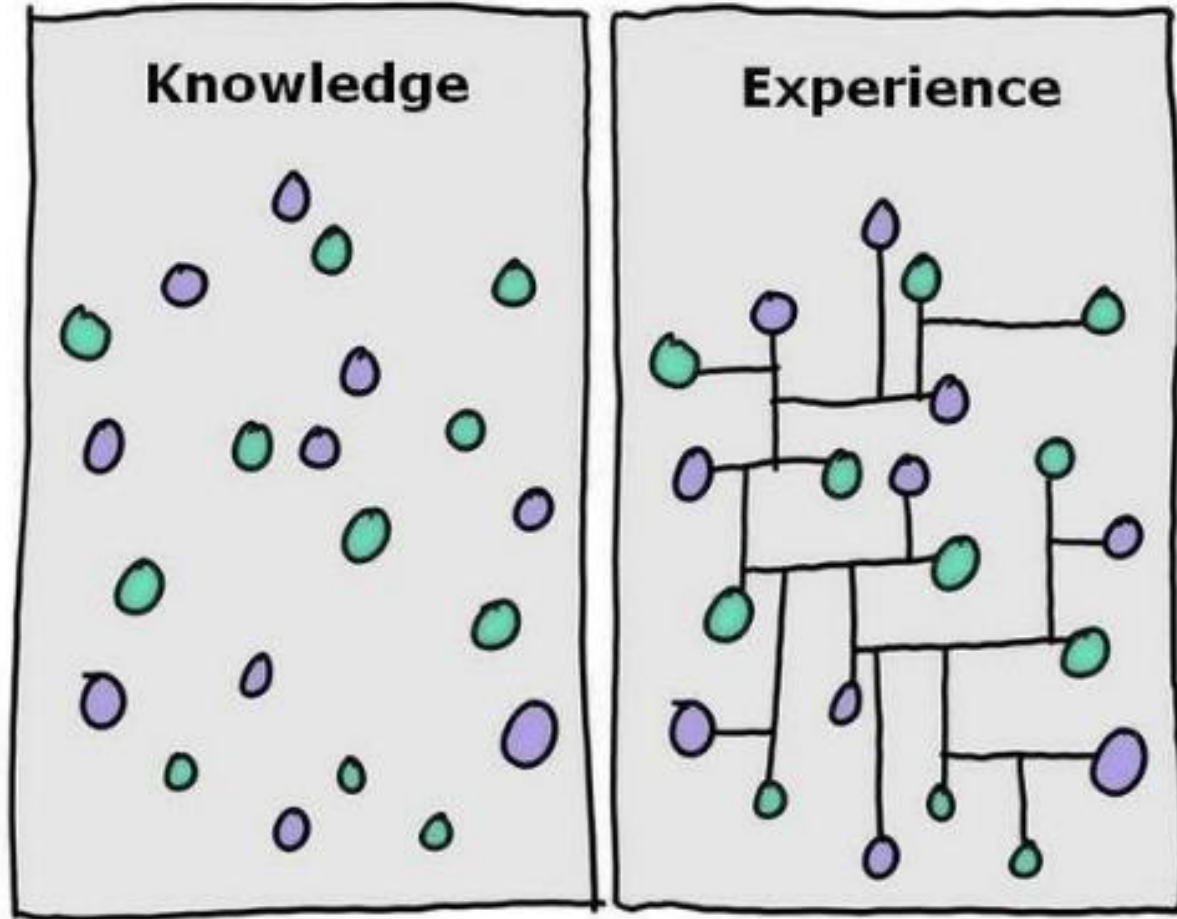
STUDY4.TW
為 學 習 而 生



Build School



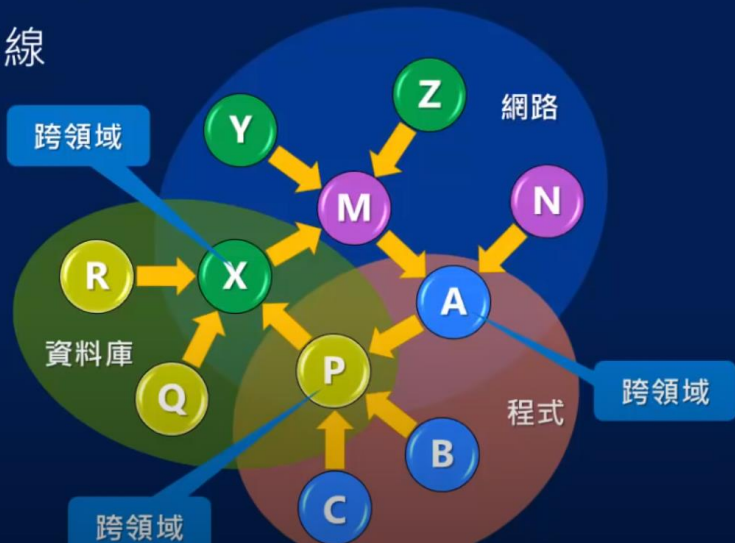
在進入正題之前...



學習的點線面

⊙ 點與線

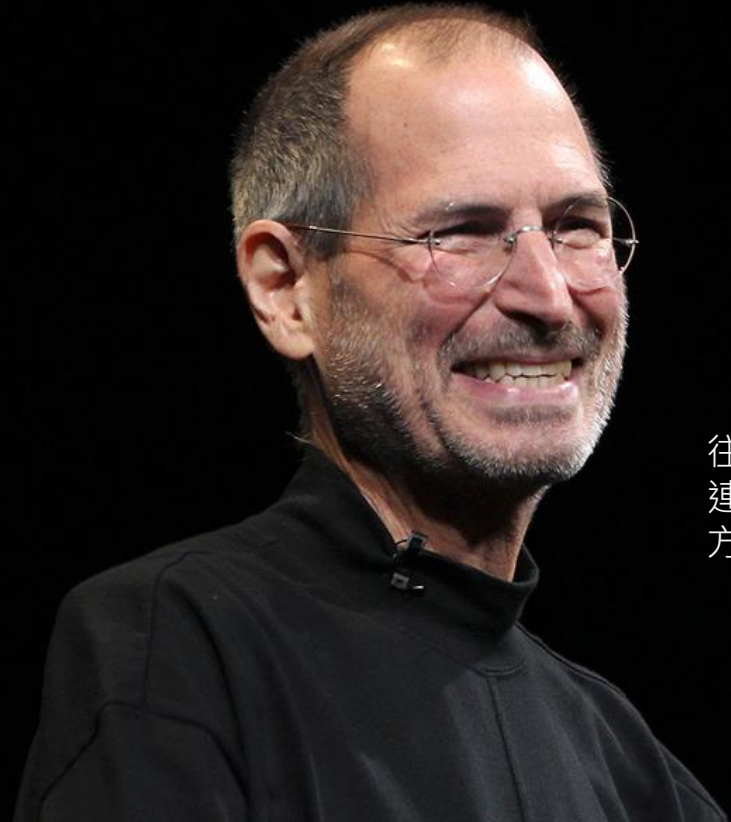
⊙ 面



影響學習速度的因素

- ⊙ 學習速度是關鍵、態度是關鍵
- ⊙ 你是不是明明記得某件事，卻常常在關鍵的時刻想不起來？為什麼？
- ⊙ 因為缺少腦神經元之間的連結
- ⊙ 所以
- ⊙ 重要的不是學會某項技術、也不是之間的連結，而是建立這些技術之間連結的能力
- ⊙ 學習速度慢的原因，正是因為概有知識不夠，或者無法和要學的東西產生連結

https://youtu.be/LIsKa4_yI48?t=1056

A portrait of Steve Jobs, smiling and wearing his signature round glasses and a black turtleneck. He is positioned on the left side of the frame against a dark background.

往前看時你無法把點連起來。只有往後看時你才能
連接它們，所以你必需相信點將在你的未來以某種
方式連接。

Steve Jobs' Stanford Commencement Address 2005

<https://mropengate.blogspot.com/2015/05/steve-jobs-stanford-commencement-speech.html>

怎麼做才能把你的知識串聯起來？

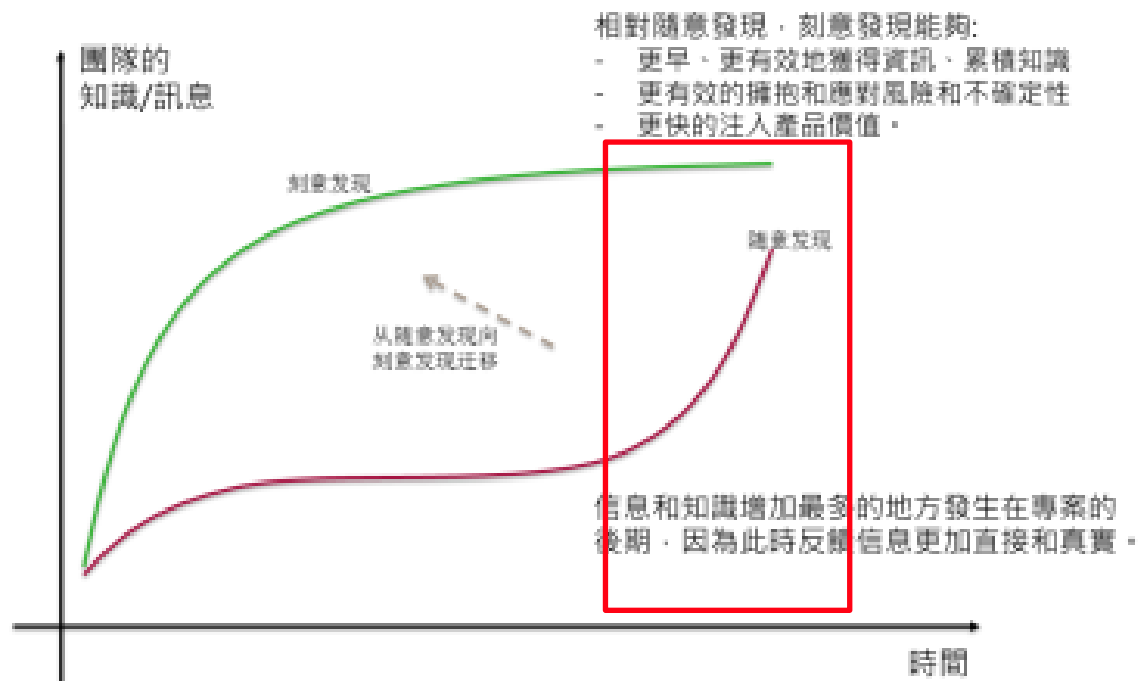
1. 打好基礎
2. 至少熟悉一套語言或是開發工具
3. 別不求甚解，刻意發現；刻意練習 (刷題)

Dan North

刻意的發現

Deliberate Discovery

強調了無論是需求還是設計，都要避免過多的預設計，而要通過在日常功能開發過程中不斷地、有紀律地、刻意地去發現，去抽象，去演進，才會做出好的系統。反覆運算貫穿整個產品開發的方方面面，是刻意的。



Dan North提出了刻意發現 **Deliberate Discovery**

AT BUILD SCHOOL, YOU

LEARN HOW TO LEARN

這才是你最該學的

BUILD SCHOOL X 中華大學 2017 春季班

91APP

進入正題: 後端工程師該累積什麼經驗與能力?

安德魯是誰？

架構師 Microservices, Cloud Native,
Application Infrastructure (91APP PaaS / Framework design)

技術長 Cloud, Multiple Tenancy Application Design, SOA
專案管理；公司經營；管理能力；人資領域 know how

技術總監 .NET XML-ORM

工程師 C#, XML, Tamino (XMLDB)

研究所 資工所；OOP, DBMS, OODB, C++, Java, SQL

大學 電機系；C, ASM, Micro processor, (作業系統, 系統工程師, 資料結構)

市場上的後端工程師需要具備什麼能力？

高級後端工程師 **月薪60,000~80,000** 08/12更新

 本公司其他工作

其他條件 * 熟悉 Node.JS & Express Web Framework

* 熟悉 MongoDB

* 熟悉 Broker System (ex: Rabbit MQ)

* 熟悉 Unit Test & Node.JS test framework, (ex: Mocha, Jest...)

* 熟悉 CI/CD

* 熟悉 版本開發工具 GIT

有以下經驗加分 (尤佳)

* 熟悉 Container 技術 (ex: docker)

* 讀過 Clean Code, Design Pattern & MongoDB Design Principle

Think: 我該開始學習每一項技術嗎?

- 。 我需要現在就會每一項技術嗎? 我有可能每一項都學嗎?
- 。 如果不會的話... 我多快能學起來?
- 。 比起學會這些技術, 更重要的是什麼? (我應該怎麼運用這些技術)
- 。 怎樣才能算是 “熟悉” 這項技術? 怎樣才能算是 “精通” ?

HR觀點 (專業職能評量):

LV0. 不具備該項能力

LV1. 我知道該領域的知識

LV2. 我知道該領域的知識並可以獨立操作

LV3. 我可以擔任講師並教授他人

LV4. 我可以獨立解決和改善問題

Think: 怎樣才算是 “可以解決 & 改善問題” ？

舉例: Rabbit MQ

LV2:

我會 Rabbit MQ, 代表我知道如何安裝、設定、管理 Rabbit MQ 服務；我也知道如何透過 C# 將 Message 送入 MQ, 也知道如何將 Message 取出處理。

LV3:

我了解什麼情況下需要用 Message Queue 來解決非同步任務執行的問題；我知道 Rabbit MQ 的特色: Exchange / Queue 的組合與應用，何時該用 Fanout, 何時該用 Topic。

LV4:

Rabbit MQ 適合用來處理 “生產者與消費者” 問題，非同步任務產生後可以交給 MQ 處理，不需等待回應可以繼續後面的動作，而另一端可由 Worker 負責消化任務。MQ 後端可以善用平行處理的技巧，提高整體的處理能力，也能同時用多個 Worker 分攤處理的任務。每個 Worker 都能夠以最佳效能運作，處理完畢後才領取下一個任務，輸出穩定，不會受到瞬間大量所影響，這種情況下能達到最佳的處理效能。

聽起來好難，我該如何培養這樣的能力？

高可靠度的微服務通訊 - Message Queue

📅 2019/01/01 📁 系列文章: .NET + Windows Container, 微服務架構設計 📁 系列文章: 架構師觀點

🔗 microservice 🔗 系列文章 🔗 架構師 🔗 DevOps 🔗 POC 🔗 MessageQueue 🔗 RPC 🔗 ASYNC



分享

你和其他 236 人都說這個讚。

💬 2 Comments



Search



Edit Post ([Pull Request](#))

Post Directory

文章目录

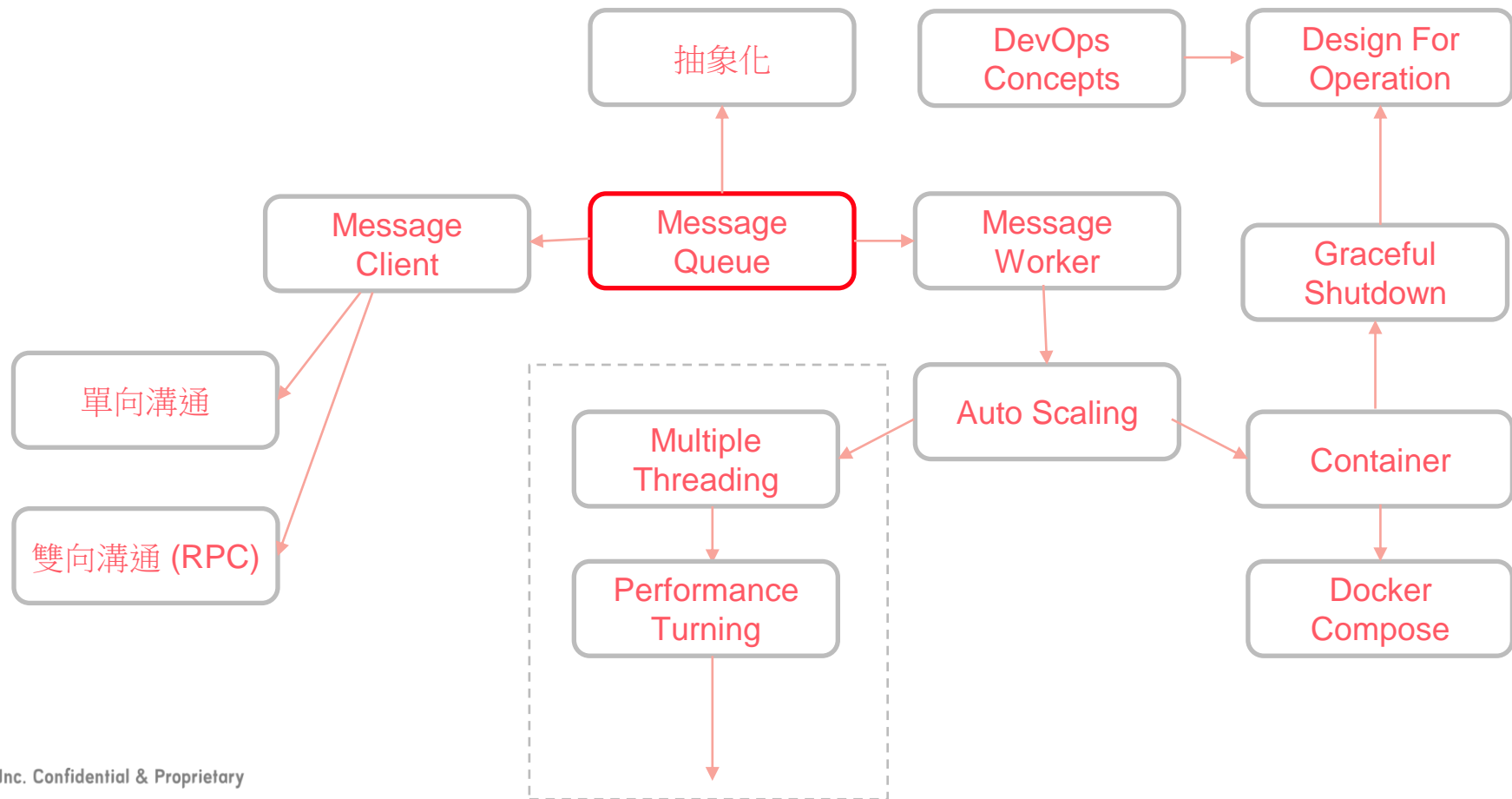
[前言: 微服務架構 系列文章導讀](#)

[團隊專屬 SDK: 通訊機制封裝](#)

[抽象化: Message Client / Worker](#)

[單向傳遞: MessageClient 介面定義](#)

[單向傳遞: MessageWorker 介面定義](#)



刻意練習: 創造適合練習的環境 (不要在正式環境練習)

1. 用 code 定義你的問題
2. 定義明確的評估指標
3. 將解決方案抽象化，先寫好主程式，跑完直接看指標

```
{  
    開始寫第一個 solution ...  
  
    不滿意? 繼續寫第二個 solution ...  
}
```

1. 試著思考理想值 (天花板) 還有多遠?
2. 找志同道合的人一起練習
3. 整理、歸納、收斂、分享

後端工程師必備: 平行任務處理的思考練習 (0916補完)

📅 2019/07/06 🔖 系列文章 🔖 架構師 🔖 Practices

✓ 讚 分享 你和其他 823 人都說這個讚。

💬 0 Comments

前面兩篇聊了不少 CLI / PIPELINE 開發的技巧跟基本功夫，這篇換個方式，來聊聊後端工程師該如何自己練習基本功夫吧。這次談的是“精準”控制的練習。



Search



Edit Post ([Pull Request](#))

Post Directory

文章目录

解題規則說明

- 1, 建立並執行你的 TaskRunner
- 2, 檢查 console output
- 3, 進階監控資訊 (csv) 說明

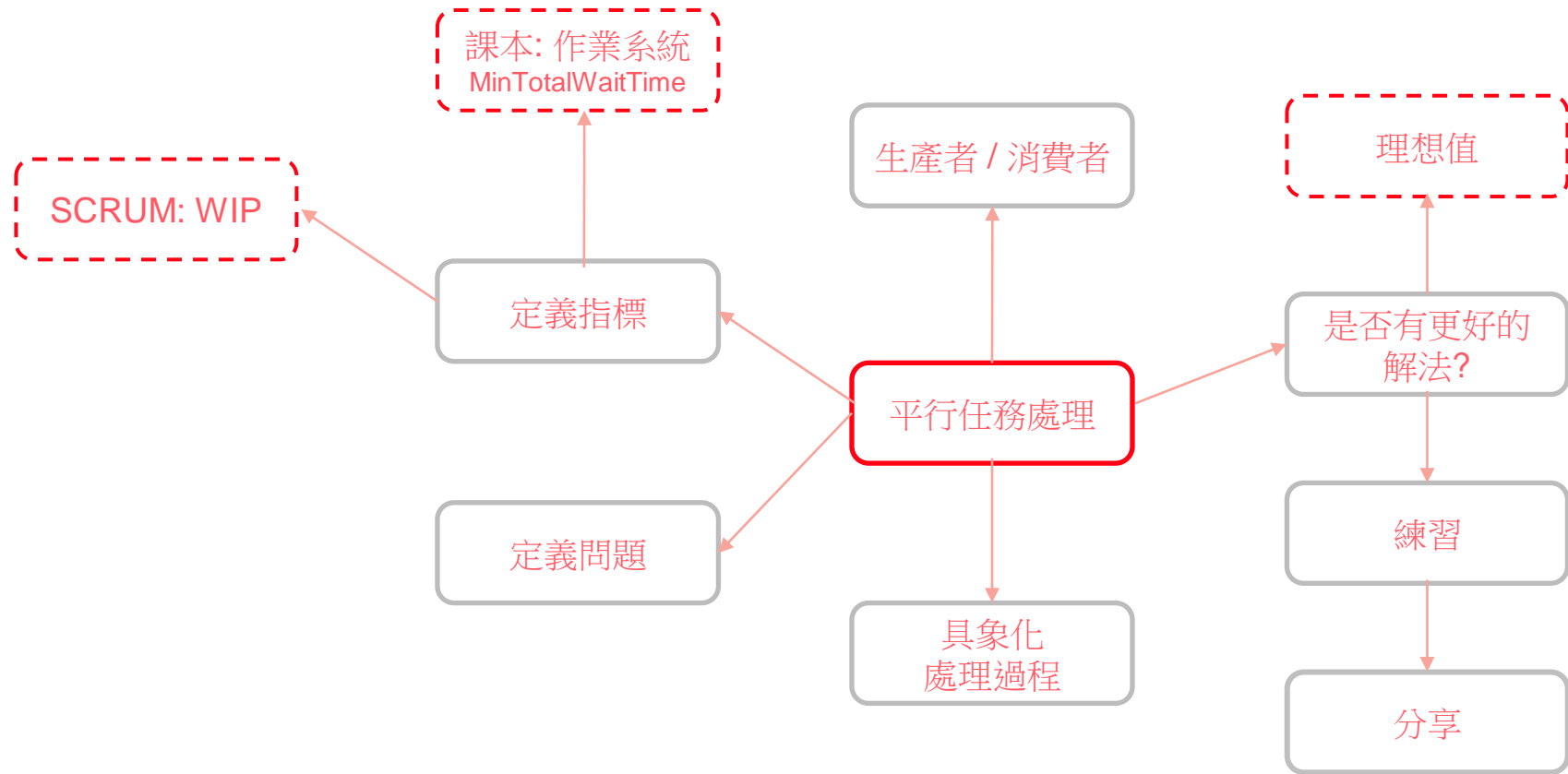
品質指標的挑選

指標: 最大處理中任務數 (Max WIP)

指標: 第一個任務完成時間 (Time To First Task, TTFT)

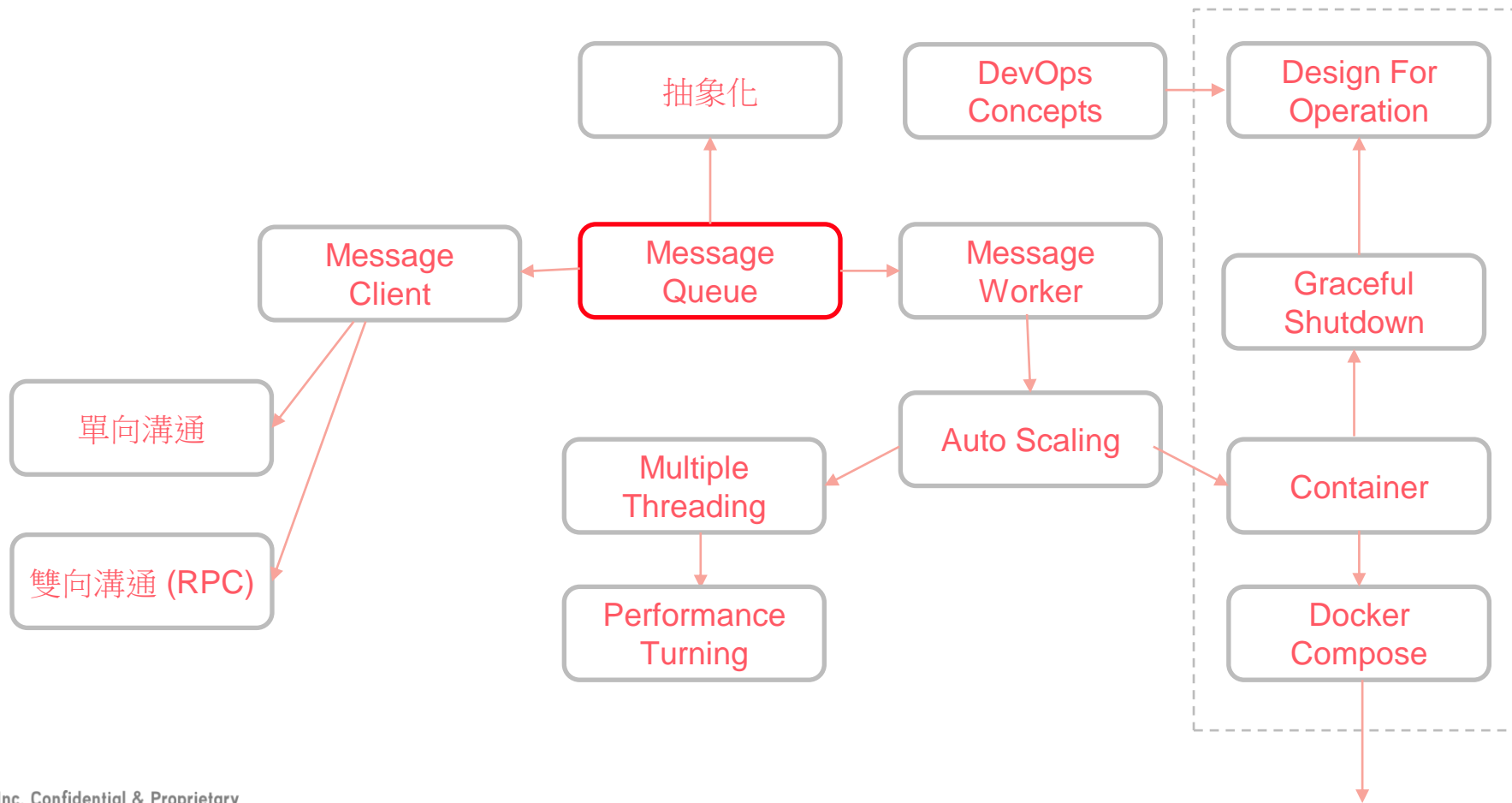
指標: 所有任務完成時間 (Time To Last

需要的基礎知識...



接續前面的 connection...

91APP



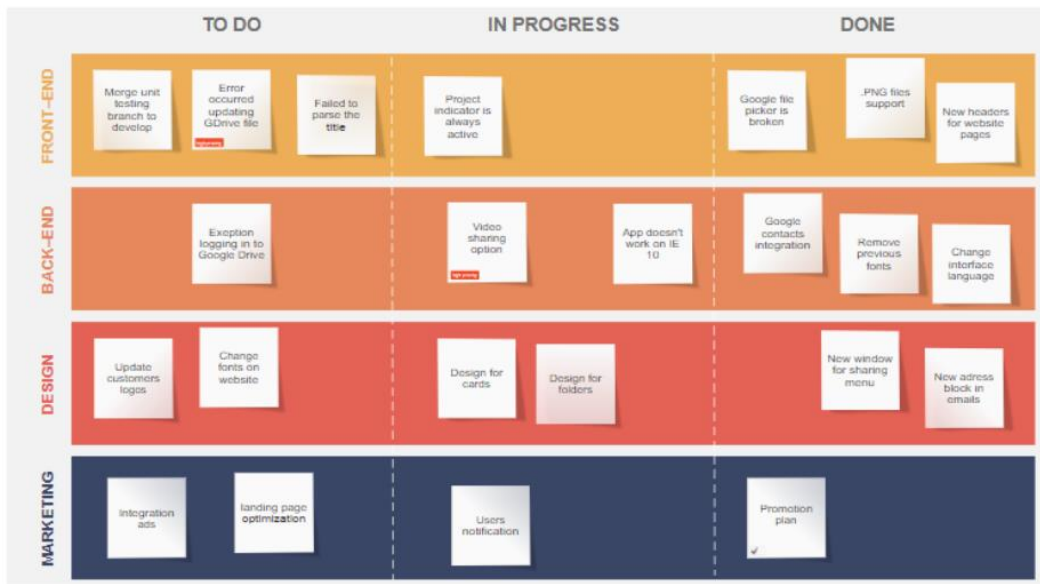
微服務基礎建設: Process Pool 的設計與應用

📅 2020/02/09 📁 系列文章: .NET + Windows Container, 微服務架構設計 📁 系列文章: 架構師觀點

📁 系列文章: 架構面試題 🔍 microservice 🔍 系列文章 🔍 架構師 🔍 POC 🔍 ASYNC

✓ 讚 分享 你和其他 696 人都說這個讚。

💬 1 Comment



Search

Search 🔍



Edit Post (Pull Request)

Post Directory

文章目录

前言: 微服務架構 系列文章導讀

挑戰: Isolation 的機制研究

Benchmark: 隔離環境的啟動速率

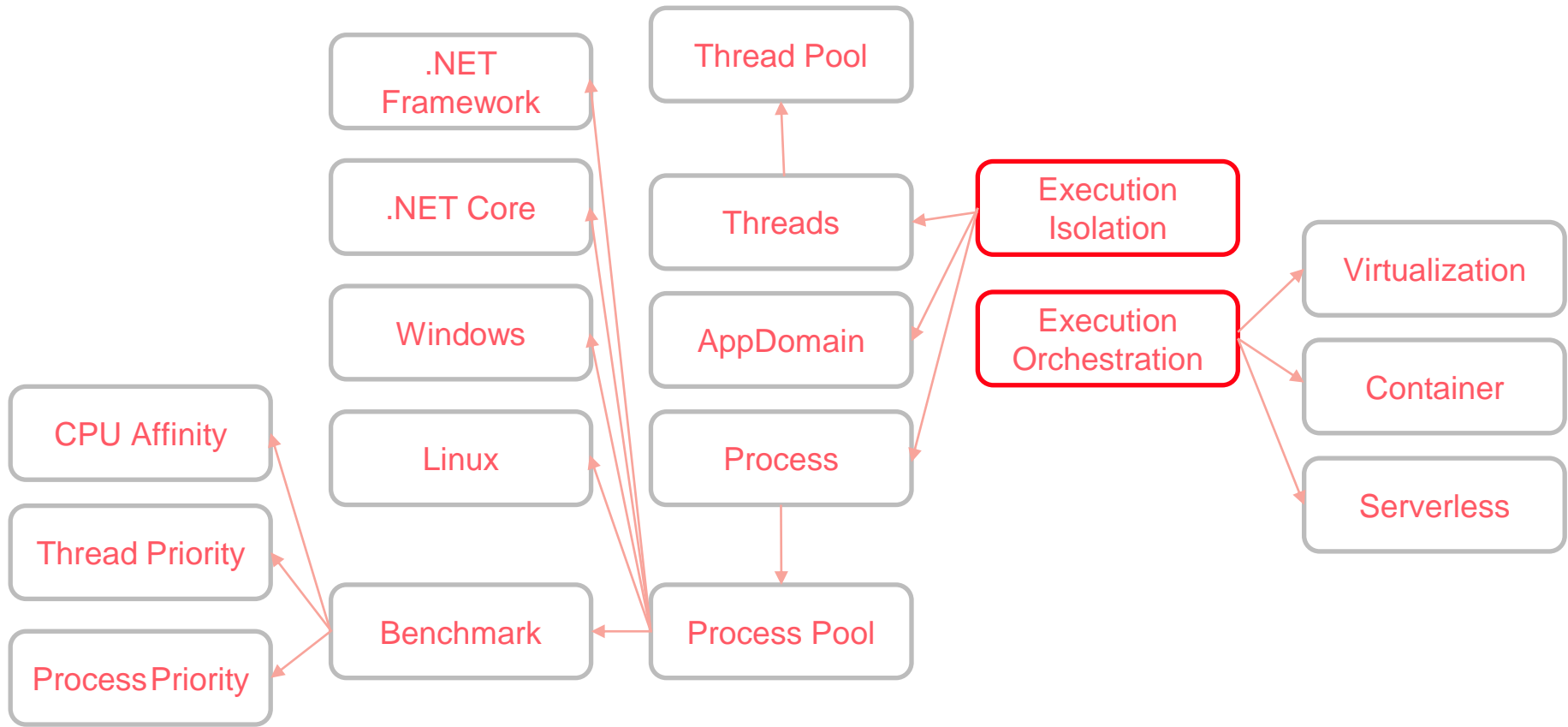
InProcess Mode

Thread Mode

AppDomain Mode

Process

小結



學完之後, 怎麼應用在工作上?

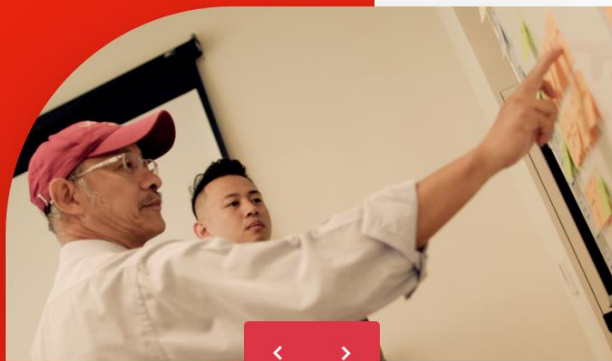
2020 線上技術分享

6/4、6/11、6/18，連續三週的線上 Meetup，

與您分享 91APP 研發處第一手的技術實踐經驗！

一窺 91APP 支撐雙11千萬流量的電商秘密！

立刻報名



議程二

講師 | 蔡奕陵 (Steven Tsai) 資深軟體工程師

解密 91APP Queue System

Queue 是現今後端服務經常會使用到的功能之一，透過 Queue 我們能更良好的因應非同步處理、應付瞬間巨量且有助於系統解耦。在 91APP 工程師必須面對超過 100+ 的 Queue 及 Worker，面對這龐大的數量，究竟該如何開發、維運、監控與部署呢？本場演講將為大家解密 91APP 的 Queue System，分享針對 Queue 的運用我們克服了哪些挑戰！

Bonus Track: 我該學習 / 使用 NoSQL 嗎?

SQL (關聯式) 與 NoSQL (非關聯式) 資料庫的比較

關聯式資料庫

NoSQL 資料庫

最佳工作負載

關聯式資料庫專門用於交易性以及高度一致性的線上交易處理 (OLTP) 應用程式，並且非常適合於線上分析處理 (OLAP) 使用。

NoSQL 資料庫專門用於包含低延遲應用程式的多樣資料存取模式。NoSQL 搜尋資料庫專門用於進行半結構資料的分析。

資料模型

關聯式模型將資料標準化，成為由列和欄組成的表格。結構描述嚴格定義表格、列、欄、索引、表格之間的關係，以及其他資料庫元素。此類資料庫強化資料庫表格間的參考完整性。

NoSQL 資料庫提供鍵值、文件和圖形等多種資料模型，具有最佳化的效能與規模。

關聯式資料庫則提供單元性、一致性、隔離性和耐用性 (ACID) 的屬性：

- 單元性要求交易完整執行或完全不執行。
- 一致性要求進行交易時資料就必須符合資料庫結構描述。
- 隔離性要求並行的交易必須分開執行。
- 耐用性要求從意外的系統故障或停電狀況還原成上個已知狀態的能力。

NoSQL 資料庫通常透過鬆綁部分關聯式資料庫的 ACID 屬性來取舍，以達到能夠橫向擴展的更彈性化資料模型。這使得 NoSQL 資料庫成為橫向擴展超過單執行個體上限制的高吞吐量、低延遲使用案例的最佳選擇。

ACID 屬性

效能

一般而言，效能取決於磁碟子系統。若要達到頂級效能，通常必須針對查詢、索引及表格結構進行優化。

效能通常會受到基礎硬體叢集大小、網路延遲，以及呼叫應用程式的影響。

擴展

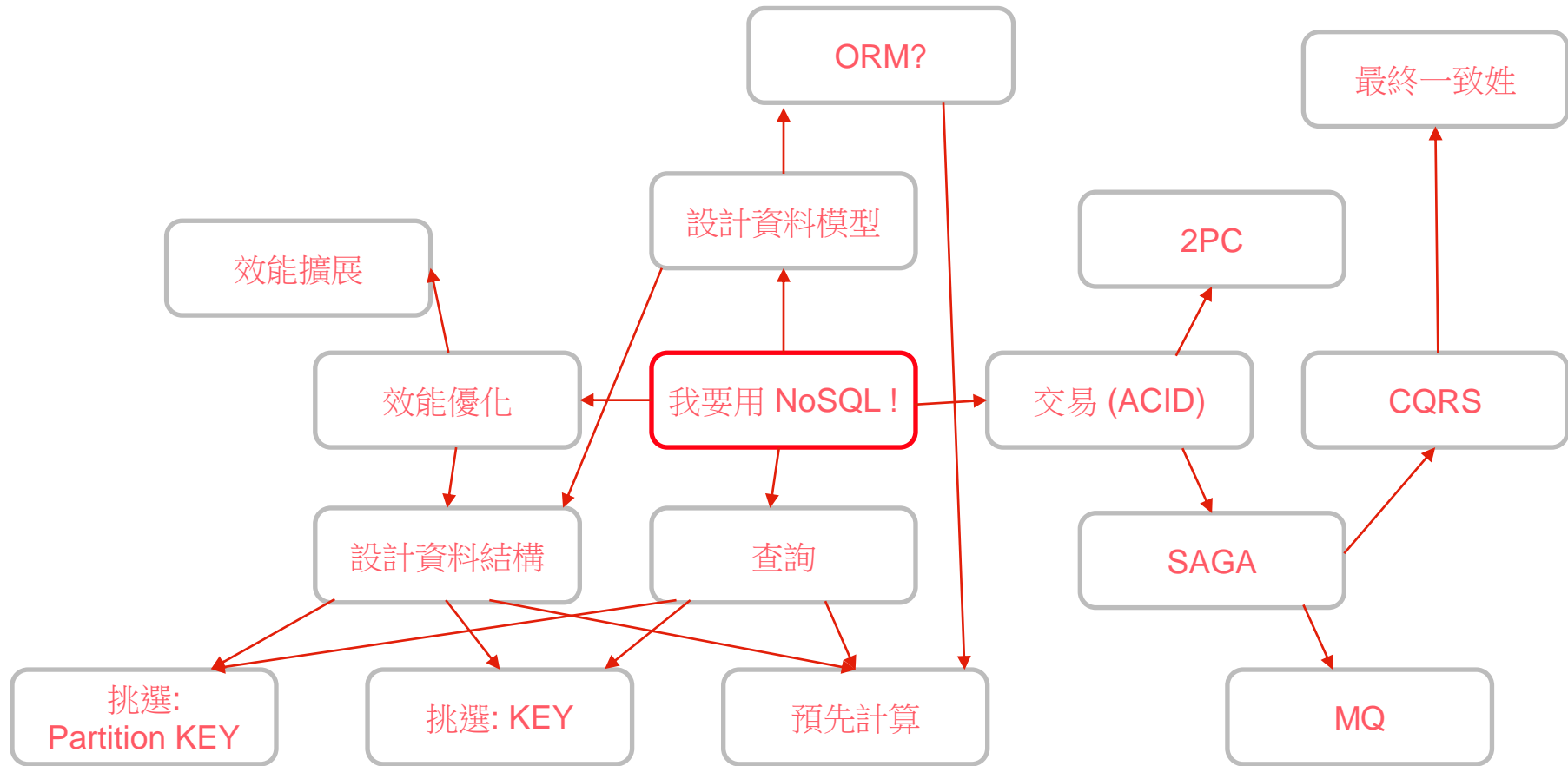
關聯式資料庫通常透過增加硬體運算能力向上擴展，或以新增唯讀工作負載複本的方式向外擴展。

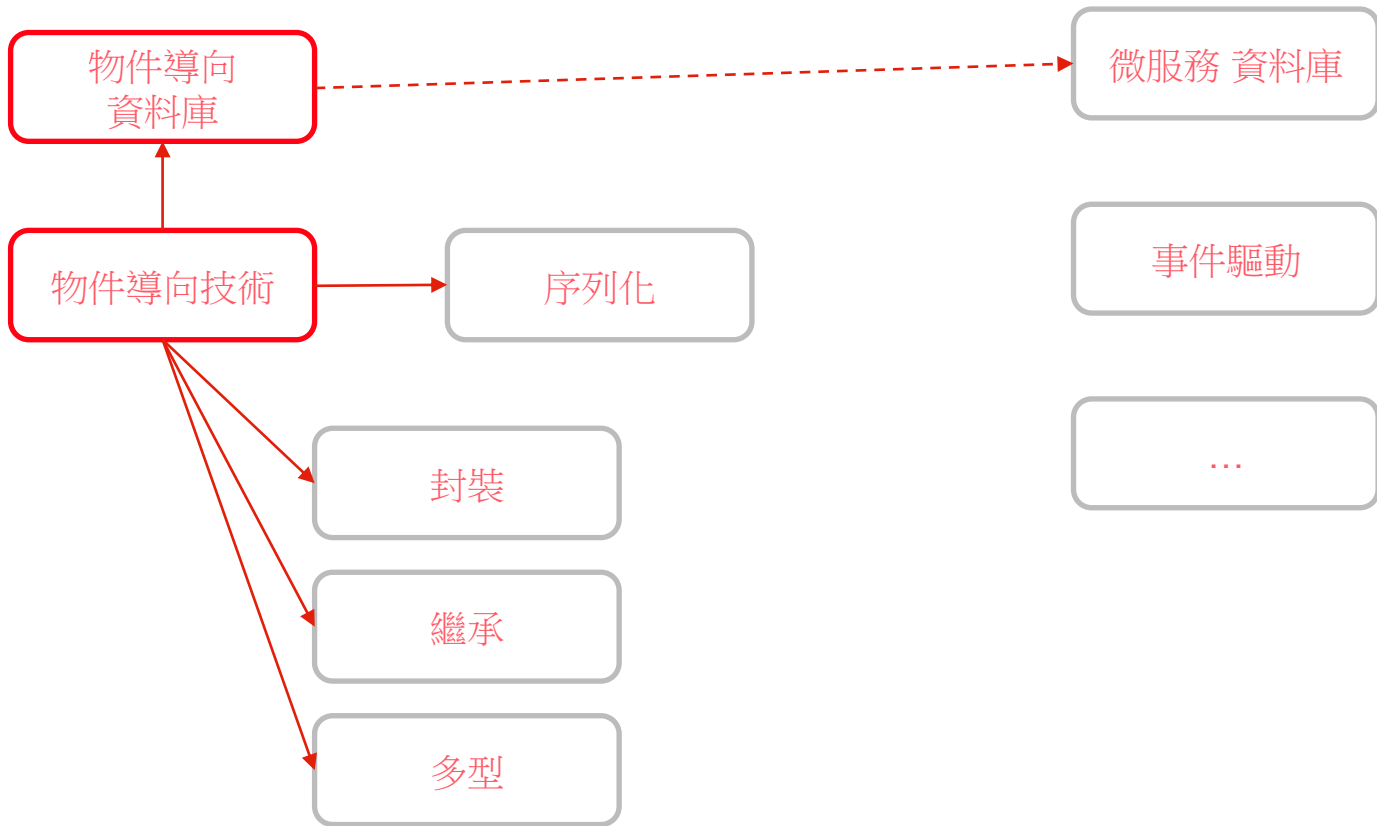
NoSQL 資料庫通常可分割，因為存取模式可透過使用分散式架構來向外擴展，以近乎無限規模的方式提供一致效能來增加資料吞吐量。

API

存放和擷取資料的請求是透過符合結構式查詢語言 (SQL) 的查詢進行通訊。這些查詢是由關聯式資料庫剖析和執行。

以物件為基礎的 API 讓應用程式開發人員可輕鬆存放和擷取資料結構。應用程式可透過分區索引鍵查詢鍵值組、欄集，或包含序列化應用程式物件與屬性的半結構化文件。





當這些你都能關聯起來時...

STUDY4.TW .NET Conf 2018 - Message Queue Based RPC, 2018/09/29

STUDY4.TW 與大師對談: 轉移到微服務架構必經之路 ~ 系統與資料庫重構, 2018/01/06

Part #4 可靠的微服務通訊 - Message Queue Based RPC; 2019/01/01

架構面試題 #1 線上交易的正確性; 2018/03/25

架構面試題 #2 連續資料的統計方式; 2018/04/01

該如何學好 “寫程式” #2. 為什麼 programmer 該學資料結構 ??, 2008/10/01

該如何學好 “寫程式” #3. 進階應用 - 資料結構 + 問題分析, 2008/10/08

EF#1. 要學好 Entity Framework? 請先學好 OOP 跟 C#, 2009/01/22

EF#2. Entity & Encapsulation, 2009/01/23

EF#3. Entity & Inheritance, 2009/03/03

給各位的建議

1. 打好基礎 (投資在能陪你 20 年的知識)
2. 至少熟悉一套語言或是開發工具 (投資在現在能讓你發揮價值的輔助技能或工具)
3. 別不求甚解，刻意發現；刻意練習 (持續穩定地前進)
4. 如果基礎(內功)夠強，你可以先搞懂 (LV1) 後確認真的需要，再去學習 (LV2)。
打好基礎是困難的 (非線性)，學習工具是需要練習的 (線性)。
5. 不論你是哪一派 (C# or Java ...), 你至少要熟悉一套才能有效的 “刻意練習”。
6. 持續做下去，你會拉開跟別人的距離... 不用貪多，但是花了時間學習的東西就要學到位 (LV3/LV4)

LV1 越多越好，有助於你知道你該怎麼挑選 (決策)

LV2 每個領域熟練一個，有助於你深入練習 (進階)

LV3 挑一個你專精的領域，做到大師的程度 (主軸)



何時開始？

“種樹的最佳時間是20年前，僅次於它最佳時間是現在”

挑選你 想深入 / 有興趣 的領域，開始培養基礎能力吧！

常見後端的領域: 資訊安全；高流量與高可靠度；線上交易；
開發設計；基礎建設與維運；

你可以參考我準備的練習題...

- 架構面試題 #1, 線上交易的正確性
- 架構面試題 #2, 連續資料的統計方式
- 架構面試題 #3, RDBMS 處理樹狀結構的技巧
- 架構面試題 #4 - 抽象化設計；折扣規則的設計機制 (06/25 補完)
- 後端工程師必備: CLI + PIPELINE 開發技巧
- 後端工程師必備: CLI 傳遞物件的處理技巧
- 後端工程師必備: 平行任務處理的思考練習 (0916補完)
- 後端工程師必備: 排程任務的處理機制練習 (12/01 補完)



Question ?