國泰人壽
Cathay Life Insurance

GPU 應用與管理經驗分享

雲端技術發展部　柯彥宇
2025.05.12

Content

- 那些年我們買過的卡片 -- 場景與用途
- 從實體走向虛擬 -- GPU 分割及虛擬化技術
- 管理議題與未來展望

用過的 GPU 卡片

2018

P100

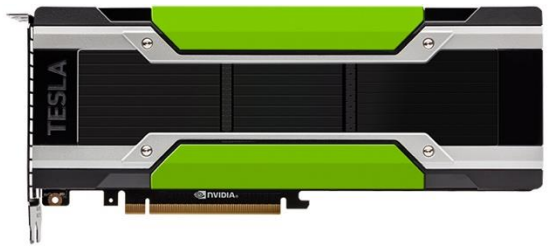
V100

A100

L40S

2025

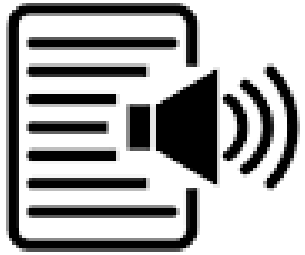
型號	架構	記憶體	SM	CUDA Core	Tensor Core
P100	Pascal	16GB	56	3,584	無
V100	Volta	16GB / 32GB	80	5,120	640
A100	Ampere	40GB / 80GB	108	6,912	432
L40S	Ada Lovelace	48GB	142	18,176	568



運用 GPU 的服務

診斷書 OCR
收據 OCR

醫療影像判讀

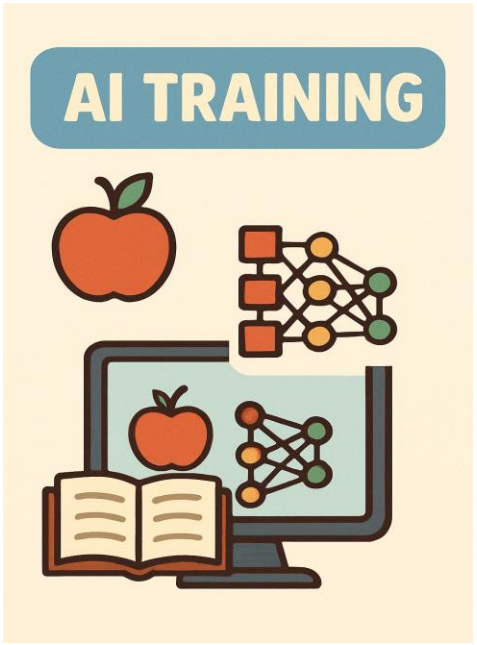


語音轉文字

人臉辨識

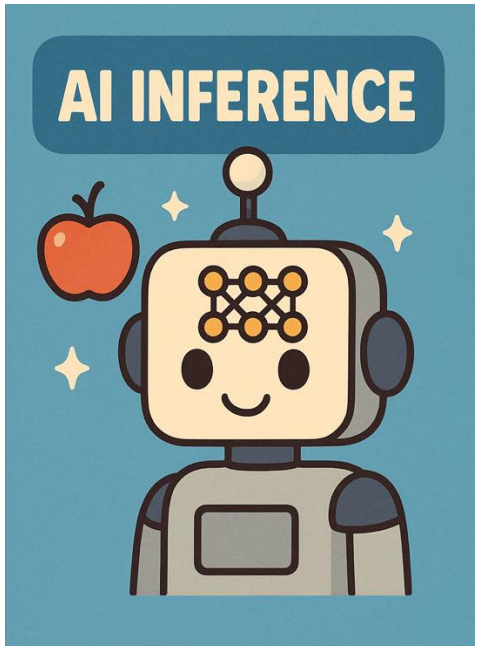
LLM大語言模型
相關運用

用途



讓開發者快速取得所需環境

- 事先打包好的開發環境
- Jupyter Notebook 透過瀏覽器開發



模型打包服務部署符合規範

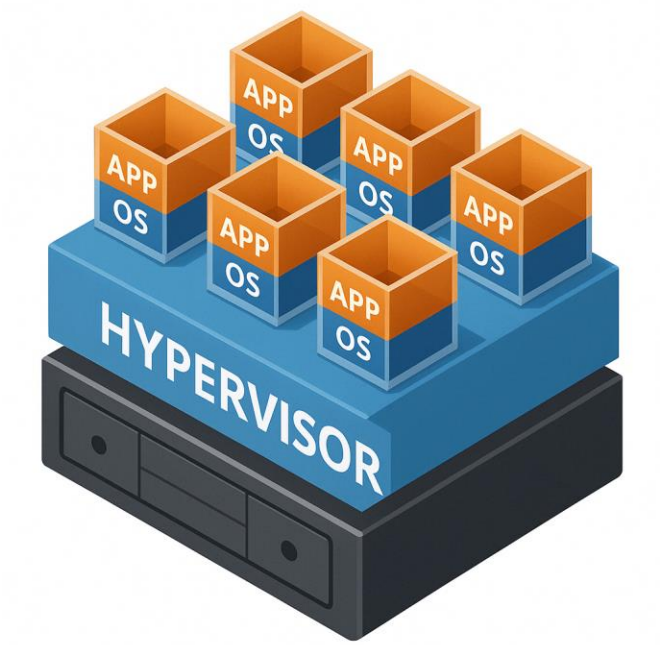
- CI/CD 流水線自動化
- Ansible + docker-compose

問題 1.

- 資源的配置，應該由應用開發人員或系統管理人員來設定？

從實體走向虛擬

- 一開始 GPU 伺服器都是以實體機的方式來使用
- 去年將新購的推論用 GPU 伺服器改為虛擬機



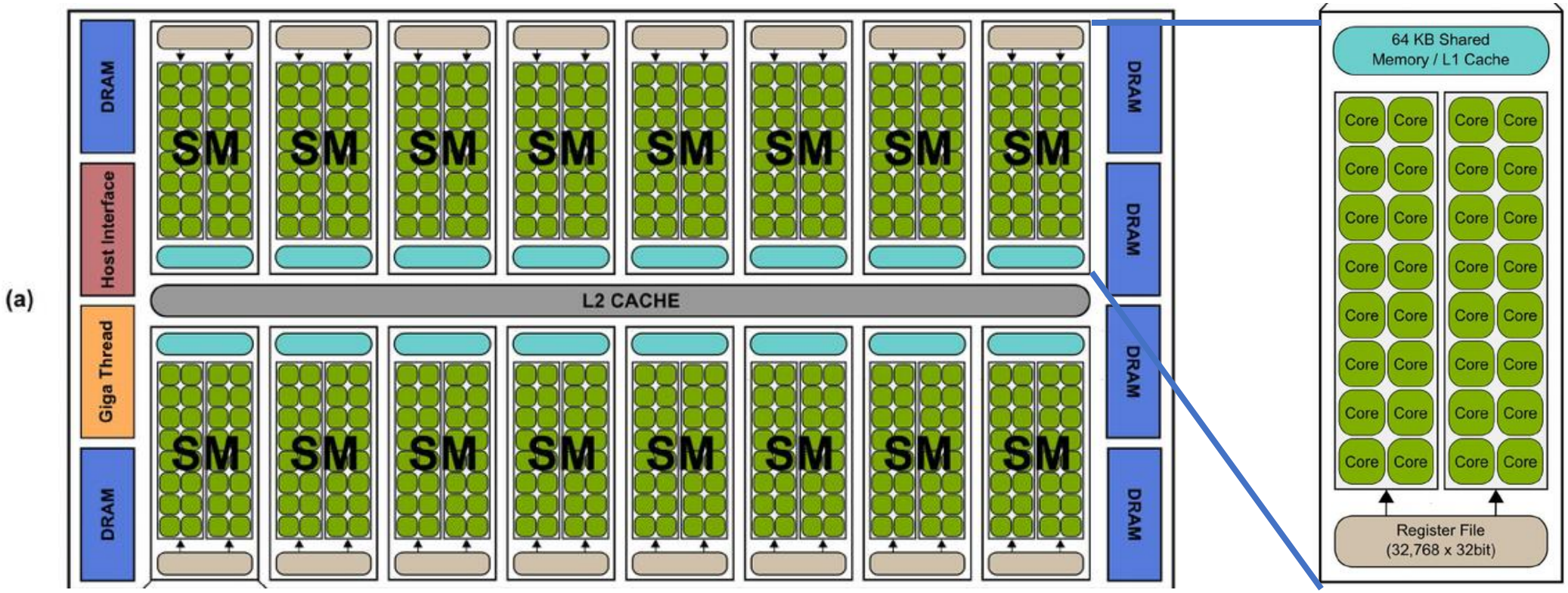
- 部署迅速、資源隔離、效用最佳化
- 備份還原、簡化流程、遷移不中斷

GPU Virtualization

一般 Hypervisor 可以分配或模擬 CPU、Memory 等硬體給虛擬機，讓虛擬機像是擁有自己的硬體。

- 虛擬機中可以**看到並使用底層的 GPU** 嗎？
- 一台實體機會開很多台虛擬機，若底層只有一張 GPU，**每一台虛擬機都可以使用這同一張 GPU 而互不干擾**嗎？

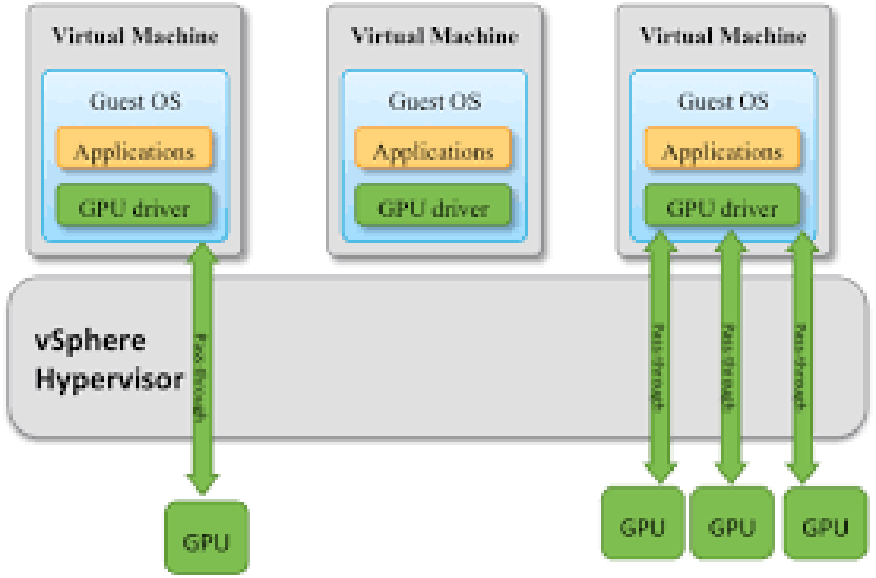
NVIDIA GPU Architecture



項目	CUDA Core	SM（Streaming Multiprocessor）
定義	最基礎的運算單元，執行單一執行緒	包含多個CUDA核心的運算模組，負責調度與管理
功能	執行基本數學和邏輯運算	調度、管理、協調多個CUDA核心和資源
組成關係	多個CUDA核心組成一個SM	一個SM包含多個CUDA核心、指存器、共享記憶體
例子	單一加法、乘法運算	同時執行多個線程塊、管理資源分配

技術 1. Pass-through

- 讓虛擬機繞過虛擬化層，直接訪問底層主機的實體 PCIe 裝置
- 一張 GPU 只能分配給一台虛擬機



技術 2. MIG（Multi-Instance GPU）

- 硬體級分割技術
- 將一張卡切分成多個彼此獨立的 GPU 實例（GI, GPU Instance）
- 每個實例都擁有專屬的運算核心、記憶體
- 這些實例可**分配給不同的虛擬機、容器或工作負載**，
  達到資源隔離與高效利用

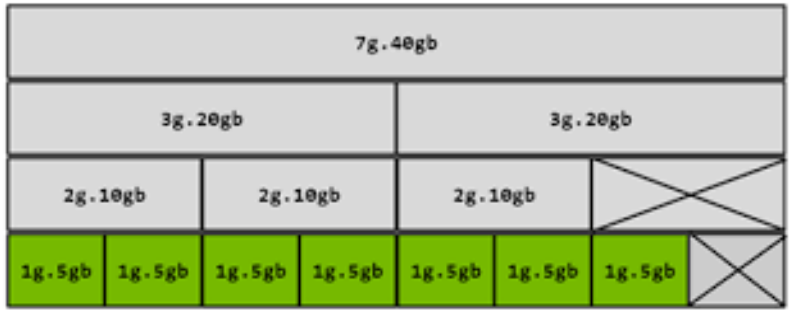
怎麼切？以 A100 40G 為例

分別從**運算單元**和**記憶體**來看

- 運算單元
  - 切分成 7 份，每份有 14 個 SM
  - 每個 SM 有 64 個 CUDA Core、4 個 Tensor Core
- 記憶體
  - 可切成 5G、10G、20G、40G

合在一起看

Profile	運算單元	記憶體	可用數量
1g.5gb	1/7	1/8 (5G)	7
2g.10gb	2/7	2/8 (10G)	3
3g.20gb	3/7	4/8 (20G)	2
4g.20gb	4/7	4/8 (20G)	1
7g.40gb	全部	全部	1



將不同 Profile 組合在一起

Config	GPC Slice #0	GPC Slice #1	GPC Slice #2	GPC Slice #3	GPC Slice #4	GPC Slice #5	GPC Slice #6
1	7						
2	4				3		
3	4				2		1
4	4				1	1	1
5	3			3			
6	3			2		1	
7	3			1	1	1	
8	2		2		3		
9	2		1	1	3		
10	1	1	2		3		
11	1	1	1	1	3		
12	2		2		2		1
13	2		1	1	2		1
14	1	1	2		2		1
15	2		1	1	1	1	1
16	1	1	2		1	1	1
17	1	1	1	1	2		1
18	1	1	1	1	1	2	
19	1	1	1	1	1	1	1

```
version: v1
mig-configs:
  all-3g.20gb:
    - devices: all
      mig-enabled: true
      mig-devices:
        "3g.20gb": 2

  all-balanced:
    - devices: all
      mig-enabled: true
      mig-devices:
        "1g.5gb": 2
        "2g.10gb": 1
        "3g.20gb": 1
```

技術 3. vGPU

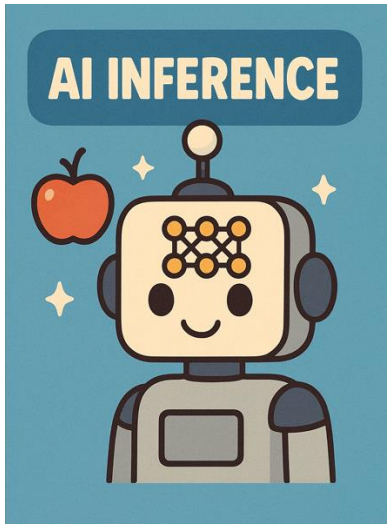
- 讓多台虛擬機能夠同時共用一塊實體 GPU 資源的技術
- 在伺服器虛擬化層安裝 Nvidia vGPU 軟體
  並與虛擬機管理程式協作，
  將一塊實體 GPU 分割成多個虛擬 GPU（vGPU）
- 需要**購買 license** 才能使用完整的功能
- 有 **MIG** 和 **Time-slicing**（時間切片）兩種模式

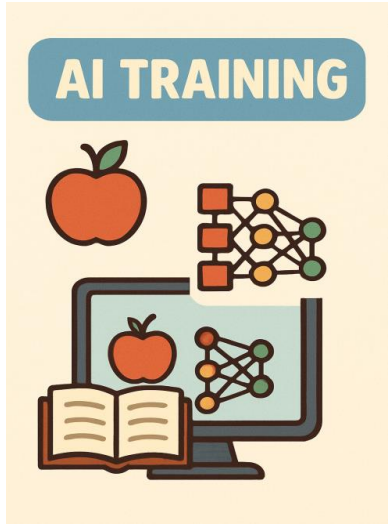
Time-Slicing

- 將 GPU 的處理時間切分成許多小的時間切片
- 依序分配給不同的應用程式或使用者
- 每個應用在其分配到的時間片內可**獨佔 GPU 運算資源**，時間片結束後再輪到下一個應用

	Time-Slicing	MIG
運作原理	時間輪替共享 GPU	硬體層分割 GPU
支援 GPU	多數 Nvidia GPU	僅限支援 MIG 的 GPU (如 A100、H100)
資源隔離	無實體隔離，僅時間切換	完整硬體隔離
效能穩定性	低 - 會排隊	高 - 獨立 slice
可用 CUDA 核心	全 GPU，但排班共享	指定的 SM 數量

國泰人壽目前的使用方式





推論用 A100 + 虛擬機

Production	MIG + vGPU
Staging	Pass-through + MIG

訓練用 L40S、V100 + 實體機

問題 2.

- 我們有一張 A100 **80G** 的卡片，為什麼它的記憶體只能用到 **70G**？

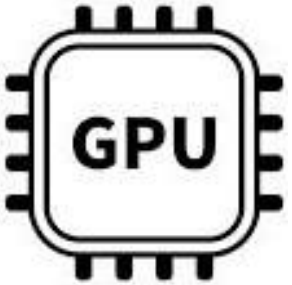
管理議題篇

功高耗

- 耗電
- 散熱

軟硬體配合

- 開發套件限定 CUDA 版本
- CUDA 版本限定 GPU Driver
- 作業系統版本也可能踩跎



資源利用

- 誰可以用？
- 要用多少？要用多久？
- 如何讓效能最大化？

未來展望篇

AI 整合平台

- 快速取得開發環境
- 模型快速發佈測試
- Gen AI (LLM、RAG)

Kubernetes

- Nvidia GPU Operator
- Nvidia Device Plugin for Kubernetes

公有雲

- 快速取得所需資源
- 合規議題
- 成本控管

問題 3.

- 使用 Kubernetes 來作容器服務調度的優勢為何？
  為什麼我們現階段會用 docker compose 而不是 Kubernetes？

Take Away

- GPU 虛擬化技術
  - Pass-through, MIG, vGPU
  - MIG 和 Time-Slicing 的差異

- 對於企業在應用部署及資源管理的做法，有一些想法和啟發

國泰人壽
Cathay Life Insurance

Q & A