

Distributed Systems

Chun-Feng Liao

廖峻鋒

Department of Computer Science

National Chengchi University

Distributed Systems

Course Introduction

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science
National Chengchi University

有關加選

- 修課總人數上限108人

- 優先序

- 資科三、四
- 有限期畢業之資訊師培生
- 雙修資科
- 資科碩補修
- 輔資科
- 資訊院屬學程(如數位內容)
- 其它 (外院、外校)

資科所碩博: 請將系級、姓名、學號 email 給課務助教(周佳慧)加選

資科系、雙、輔: 自行以加退選方式加選
加退選後仍選不上者，印出加簽單，簽名後親交給資科系課務助教，以優先手動加簽處理



仍有餘額時，2/26上課依排序手動加簽

- 同區人數過多時

- 抽籤方式來決定加簽序
- 中籤同學記得到資科系辦索取「資科系加簽單」給老師簽名後交系辦

課務助教email: ch.chou@cs.nccu.edu.tw

老師email: cfliao@nccu.edu.tw

加選意願表

- 資訊院屬學士學程(如數位內容、人智應)、外院、外校才需要填
- <https://forms.gle/Akxv7bhgZNeQr8rg9>
- 9/20 23:59 關閉表單，不再接受登記



ChatGPT is at capacity right now

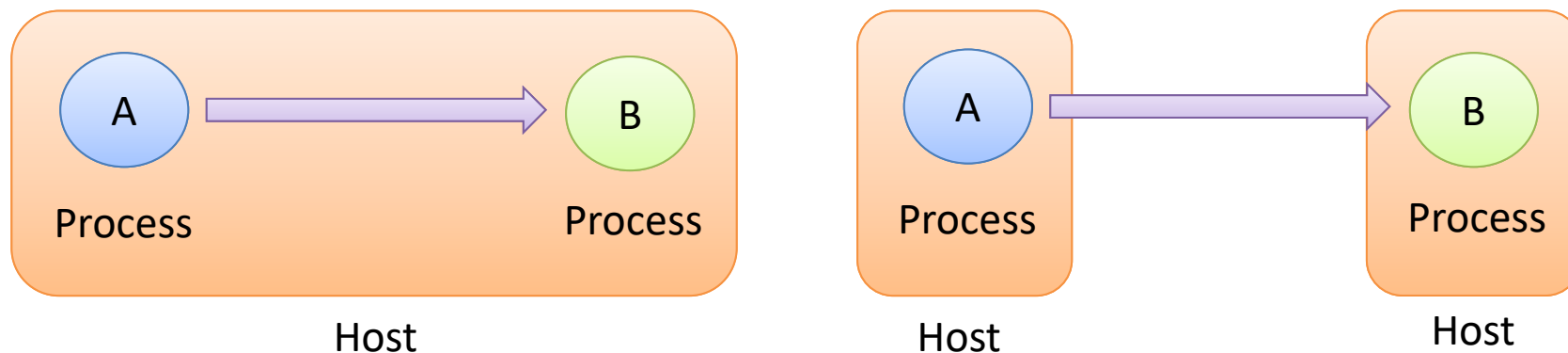
[Get notified when we're back](#)

Write a guided meditation about the status of ChatGPT.

As you read this message, please take a moment to pause and breathe. Notice the sensations in your body and the rhythm of your breath. Remind yourself that you are not alone in wanting to try out ChatGPT. Many others are interested in it as well. Be patient and know that the website is doing its best to accommodate everyone. Trust that the right time for you to try ChatGPT will come. In the meantime, continue to focus on your breath and stay present in the moment.

Distributed Systems

- 什麼是分散式系統
 - 定義
 - 透過網路連接的電腦節點上的Processes互相溝通所形成的應用
 - Processes彼此透過網路定址
 - 可能位於同一台實體電腦; 但彼此不使用記憶體參考來定址



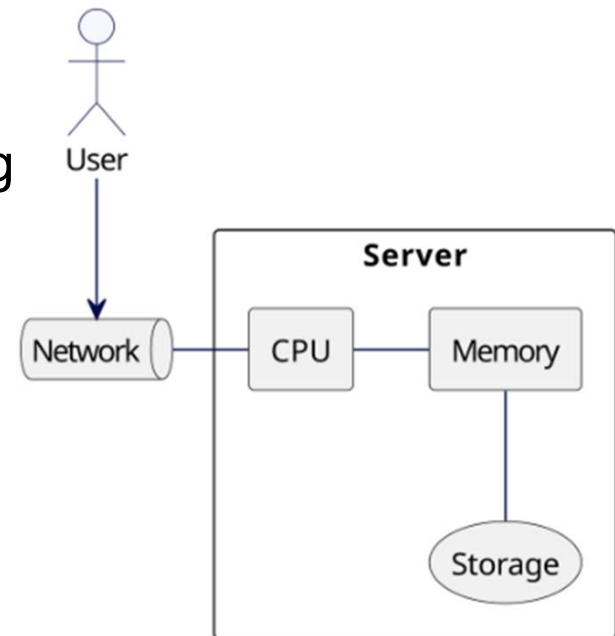
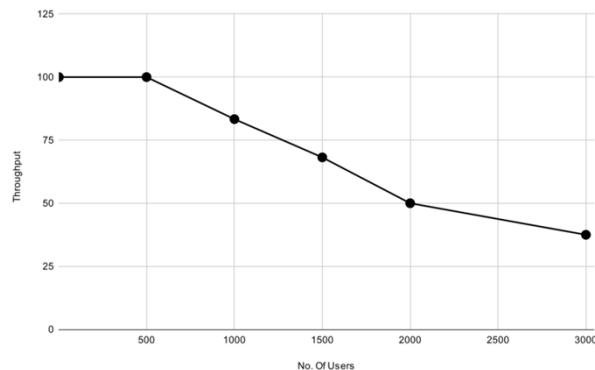
Distributed Systems

- Motivations
 - Resource and information sharing 資訊共享
 - Ex: Infrastructure/ platform/ service/ storage/ data provision
 - CSCW (Computer-Supported Cooperative Work) 協同合作
 - Ex: 團隊協作、電子商務
- Coulouris et al.
 - Networked: components located at networked computers
 - Collaborate: coordinate their actions by message passing
- Tanenbaum et al.
 - Networked: a collection of autonomous computing elements
 - Collaborate: appears to its users as a single coherent system

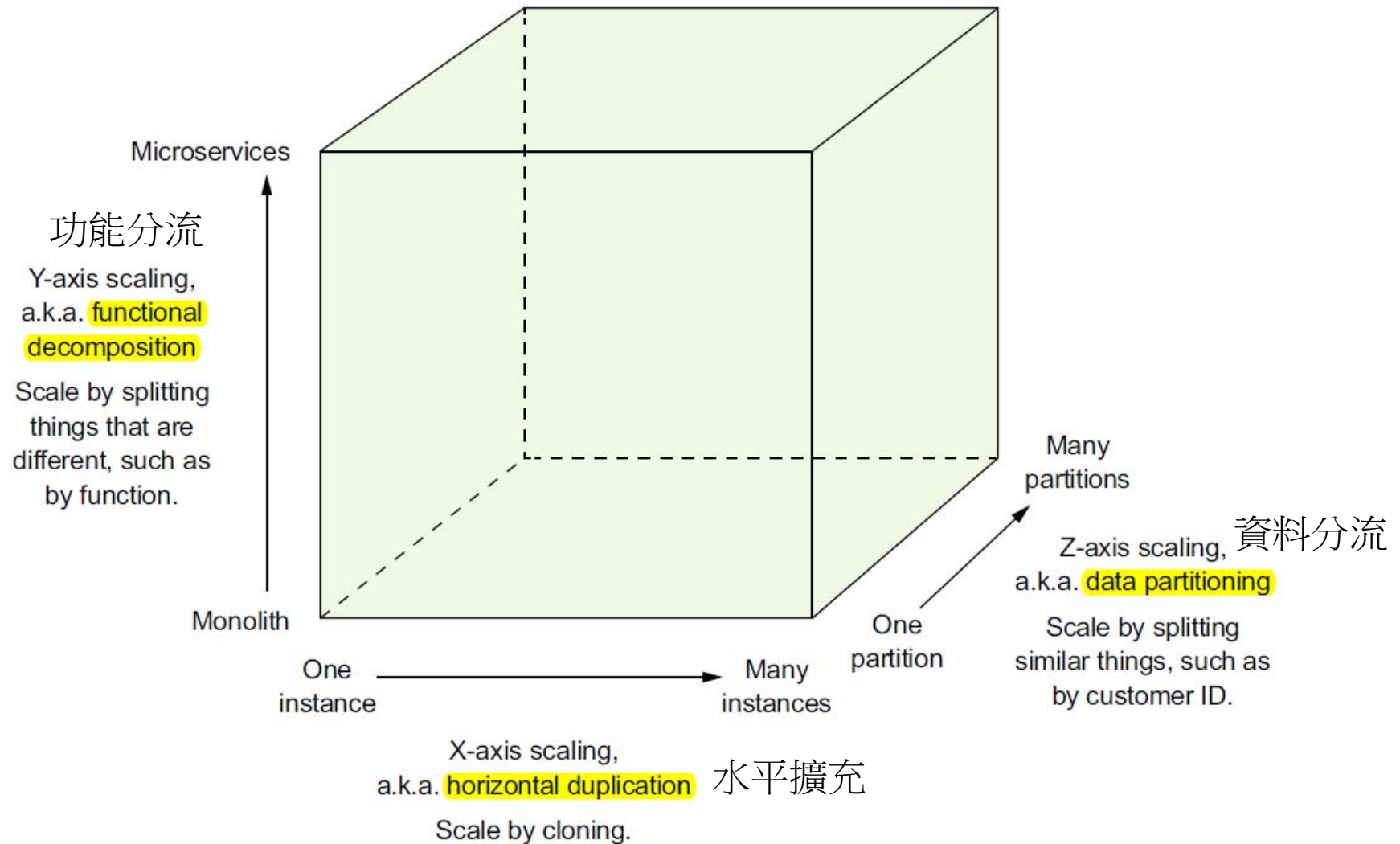
Scale: 當代系統「分散」的主要原因

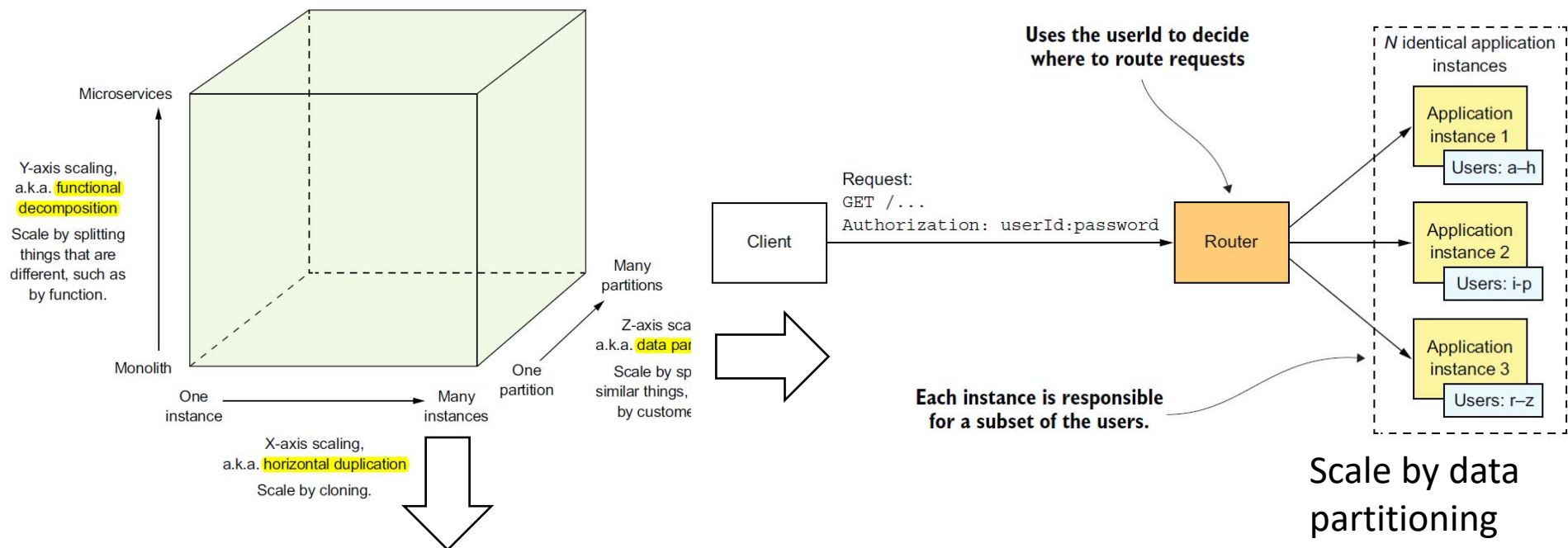
- Network
 - 假設網路設施為1Gbps (125M bytes/sec)
 - Record size 1K, 125,000 requests/s (upper bound)
 - Record size 5K, 25,000 requests/s (upper bound)
 - 傳送內容大小決定了每秒可支援user要求上限

- CPU/Memory
 - 資源飽和後，導致throughput下降:thrashing
 - 資源飽和前夕為可能最佳解

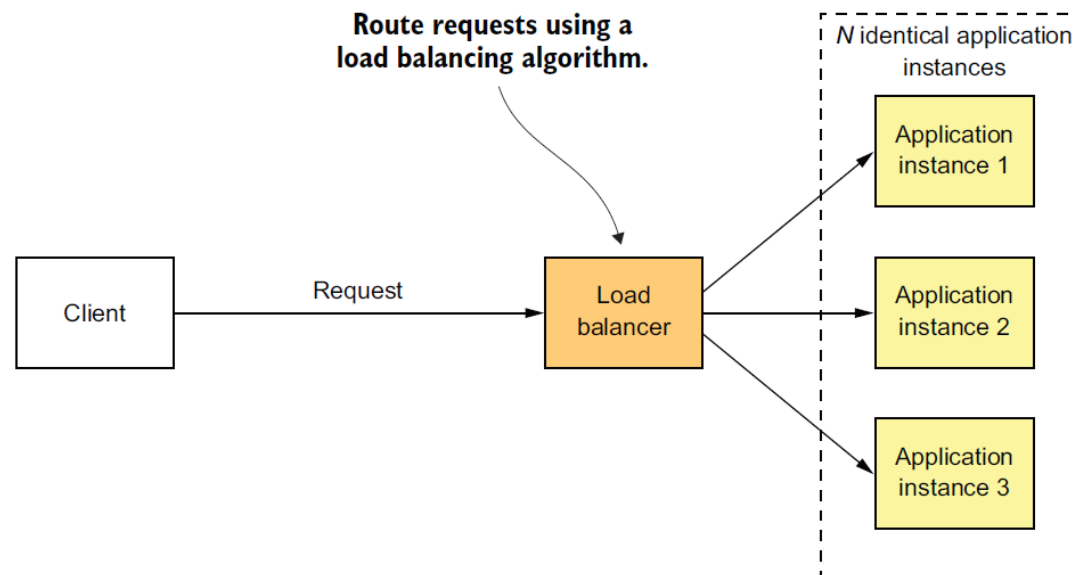


The Scale Cube

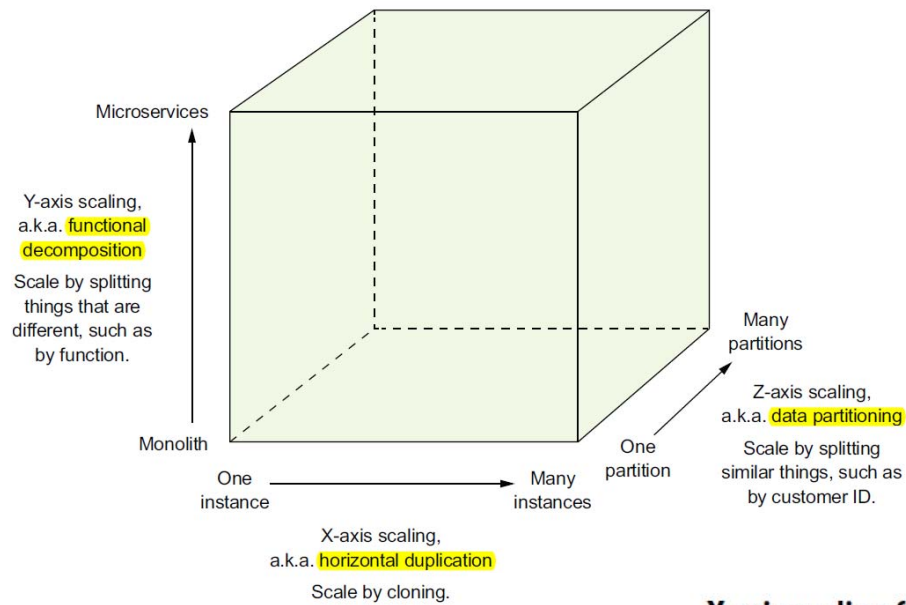




缺點: SPOF

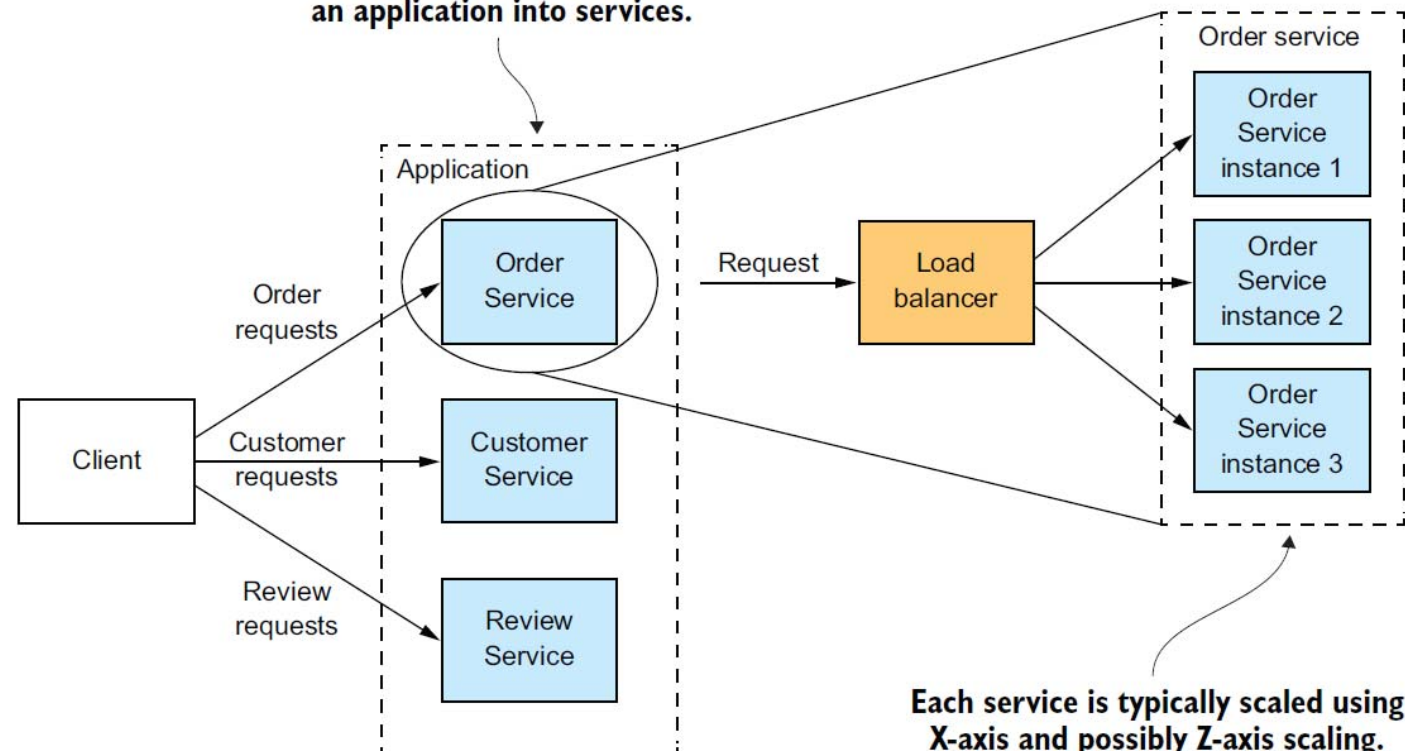


Scale by cloning 缺點: 副本間的資料同步



Microservice scaling
 Layer 1: Functional Decomposition (Y)
 Layer 2: Horizontal (X) + data (Z)

Y-axis scaling functionality decomposes an application into services.



Consequences of Distributed Systems

- Concurrency
 - Concurrent processes is the norm
 - Computers carry out tasks simultaneously and independently
 - Coordination of the concurrent processes is a recurring topic
- No global clock
 - No single global notion of the correct time
 - Coordinate by exchanging messages
- Independent failures
 - 分散式系統上無法區分「crash」還是「running slow」
 - Dealing with distributed failure is the responsibility of developers
- Networked
 - (見下頁)

Single host app sync by CPU cycle

Fallacies of Distributed Systems

錯誤認知

- A set of assertions made by Peter Deutsch
 - False assumptions that programmers new to distributed applications invariably make
- The network is
 - Reliable
 - No latency
 - Secure
 - Cost is zero
 - Homogeneous
 - (Topologically) fixed
- The network has
 - One administrator
 - Infinite bandwidth

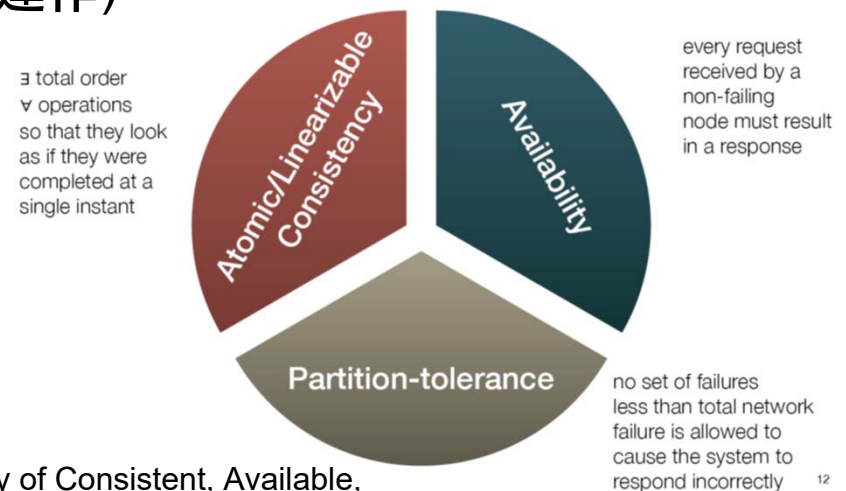
Peter Deutsch

彼得·多伊奇自幼被視為神童。1963年，在他17歲時，完成並發表了PDP-1 Lisp 1.5版。曾參與Smalltalk的創作，啟發了Java程式語言的即時編譯概念。他曾在全錄帕羅奧多研究中心（Xerox PARC）及昇陽電腦工作。1973年，在加州柏克萊大學取得計算機科學博士學位。他創立了阿拉丁企業後，於1988年推出Ghostscript。



Brewer's Conjecture : CAP

- 一個系統無法在**同一時刻**兼具CAP屬性
 - 已被證明，理論上是正確的
- CAP
 - Consistency (狀態、資料一致)
 - Availability (不斷提供服務)
 - Partition Tolerable (斷網不影響運作)



生成式AI帶來的衝擊

- Matt Welsh
 - PhD, EECS, UC Berkeley
 - Staged Event-Driven Architecture (SEDA)
 - Advisors
 - David Culler (Mote, SMAP, Tiny OS)
 - Eric Brewer (The CAP Theorem)
 - Faculty of CS, Harvard University (Tenure)
 - Mark Zuckerberg took his OS course
 - Worked in Google and Apple after leaving Harvard

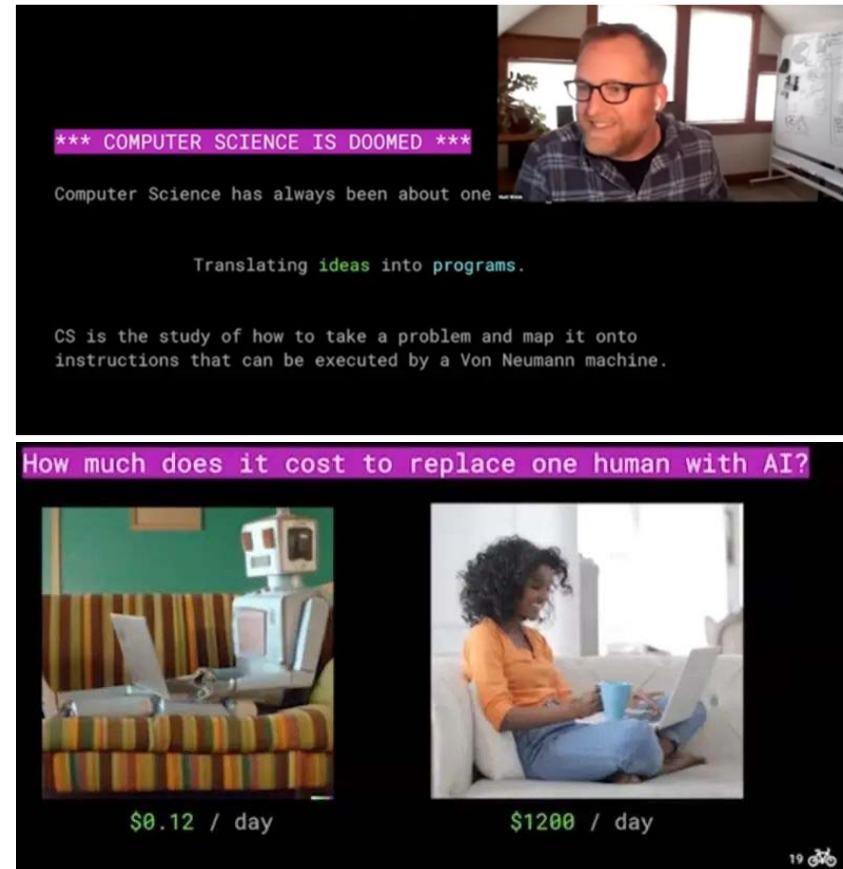
Welsh, M. (2023) The End of Programming, Communications of the ACM, Volume 66, Issue 1, January 2023



生成式AI對資訊工程領域的影響

照片來源: Matt Welsh於Chicago ACM meetup的演講:
<https://youtu.be/qmJ4xLC1ObU>

- 在適當引導下，AI有能力進行專業程式設計
 - 程式設計(Programming)
 - 將「想法」轉譯為「程式碼」(Translating ideas into code)
 - 「轉譯」是生成式AI的專長
- Welsh的論點
 - 未來的programming:
 - Let AI know the context
 - Let AI know the problem
 - Review and revise
 - 大部份實質程式碼會由AI產出
 - 還是要有人具備底層、專業的知識
 - 但不需要像現在那麼多



*** COMPUTER SCIENCE IS DOOMED ***

Computer Science has always been about one ---

Translating ideas into programs.

CS is the study of how to take a problem and map it onto instructions that can be executed by a Von Neumann machine.

How much does it cost to replace one human with AI?

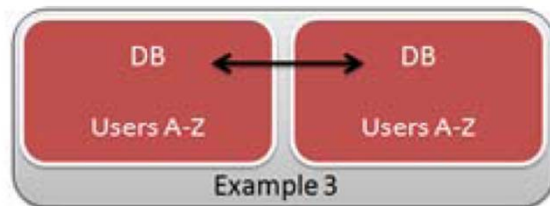
Task	Cost
Robot sitting at a desk with a laptop	\$0.12 / day
Woman sitting on a couch with a laptop	\$1200 / day

19

還需專業工作者介入的部份: 主要是NFR (Non-functional requirements), 例如「例外處理」、可維護性、安全性、強健性...; 或是較少人處理過的新領域(量子運算?)

Drop Partition Tolerant (不能斷網)

- 做法：假設網路永遠可靠，或是不依賴網路
 - 所有Process在單機上跑且不共享資料
 - 不需要網路通訊
 - 所有Process在不掉封包的小型網路上跑
 - 例如: 以bus連結的CPU cores
- 結果: 可用redundancy 解決其它二個問題
 - 確保Consistent: 單機或網路不掉包→狀態一定可以被sync
 - 確保Available: 除非**所有**instance死掉，否則會一直正常提供服務



Highly available

Drop Availability

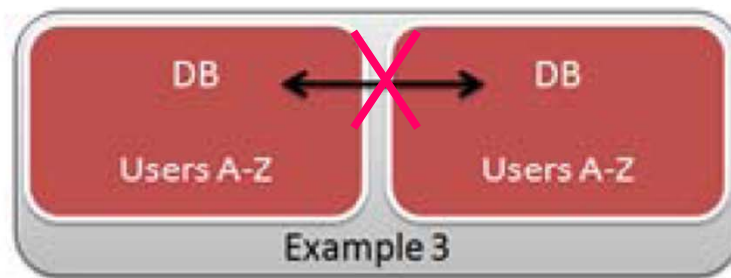
- 做法:
 - 假設網路中沒有node會死掉，或是死了也沒關係
 - 不需redundancy → 確保C
- 結果: 不需要availability，所以不需要redundancy
 - 確保C: 一份資料只保留在一處 → Consistency 可確保



- 確保P: Network 可以partition → 沒有sync必要

Drop Consistency

- 做法:
 - Nodes中的states可不一致
- 結果:可用redundancy 解決其它二個問題
 - 確保P: 不sync states → 斷網不影響運作



- 確保A: 除非在network中所有instance死掉，否則會一直正常提供服務

CAP in a Distributed System

- “The reality is that you will always have network partitions”
 - Leaves only two choices
 - Optimize either for consistency or high availability
 - Choose Consistency
 - Drop availability → Focus on MTTR
 - Centralized design (SPOF)
 - Easier to understand and design
 - Choose Availability
 - Drop consistency (enable eventually consistency)
 - De-centralized design
 - Hard to design, ex: CQRS/ES 、 blockchain



Cloud Native
Transition

SPOF=Single Point of Failure

Synchronous and Asynchronous Networks

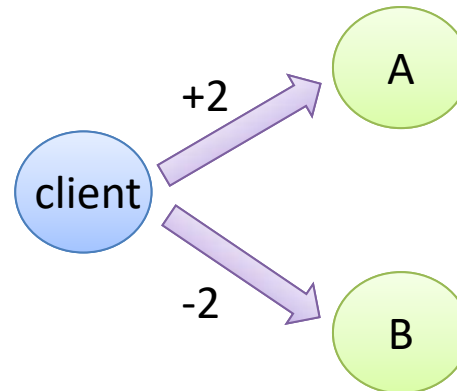
- Asynchronous network
 - Nodes can send data at any time
 - Half-duplex
 - 收和送是二個獨立的動作，不確定何時完成
 - Receiving node can be unavailable
 - Buffering
 - Data can be lost
 - Resend可能造成問題(ex: double-spend)→ensure idempotent
 - No global clock
 - 各節點時間可能不一致，timestamp變得沒有意義
- Synchronous network
 - 和上述相反的(理想狀況)
 - 許多分散式演算法是基於synchronous network假設

分散式系統常見議題

- Leader Election
- Search (Discovery)
- Routing (Shortest Path)
- Termination Detection (錯誤偵測)

分散式系統常見議題 (2)

- Heterogeneity
 - Huge amount of standards!
- Concurrency (Share Memory)
 - Distributed transaction (All or nothing semantic)
 - Distributed consensus (de-centralized, no single commander)
- Quality of Services (QoS)
 - Scalability
 - Stability
 - Security
 - Transparency (ease of use)



Theme of this Course

- 呈現與論述分散式系統設計
 - 分散式架構(Architecture)與範式(Patterns)
 - 使用UML (Unified Modeling Language)或其它自訂圖示
 - 評估各種性質(QoS)
- 了解並善用當代分散式系統議題與應用
 - Network protocols
 - Direct and indirect communication
 - HTTP and RESTful WS
 - Dynamic resource management
 - Cloud native + DevOps
 - Container/ Observability/ CQRS and Event Sourcing
 - SOA and Microservices
 - (optional) Blockchain

Reasoning for the Design of Systems

1. Know the context
2. Make a claim (i.e. a comment or an idea)
3. Justify your claim

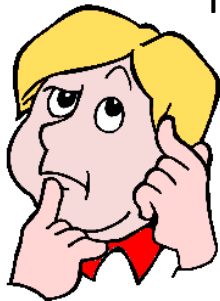
Please present your claim “logically”.
No emotional terms.





Database

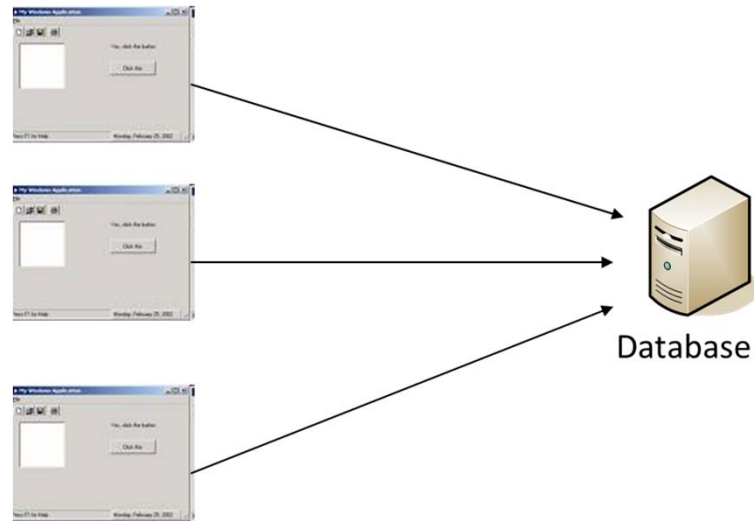
Please present your idea “logically.”
No emotional terms.



Hmm..I **feel** that it is not a good design....



An Example



(Context) If there is a large number of clients, and no automatic client updating mechanism is available

(Claim) then updating client applications will be a nightmare.

(Justification) Because N clients require N times of redundant deployment efforts.

Case

原戶役政系統採用COBOL程式語言，戶政人員需事先背指令輸入再操作十分不便，加上COBOL程式語言維護不易，內政部2007年規劃全面汰換為JAVA程式語言，2009年起將28項系統簡化為14項，同時設計新戶口名簿，新增「記事」、「非現住人口」項目，可取代戶籍謄本使用。經4個月新舊系統平行試辦及10次系統壓力測試，各軟體已經24次修改，今年2月5日正式上線，但因設計未完善仍造成大混亂；整個系統共花費5億多元。

謝愛齡表示，內政部目前已知受影響民眾，是透過1996專線反映及抱怨，共有一百二十七件，其他未透過1996的申請民眾可能也不少，主要都是申請遷出、遷入或結離婚的案件，而且出問題的都在跨縣市，她對民眾所帶來的不便，深感抱歉。

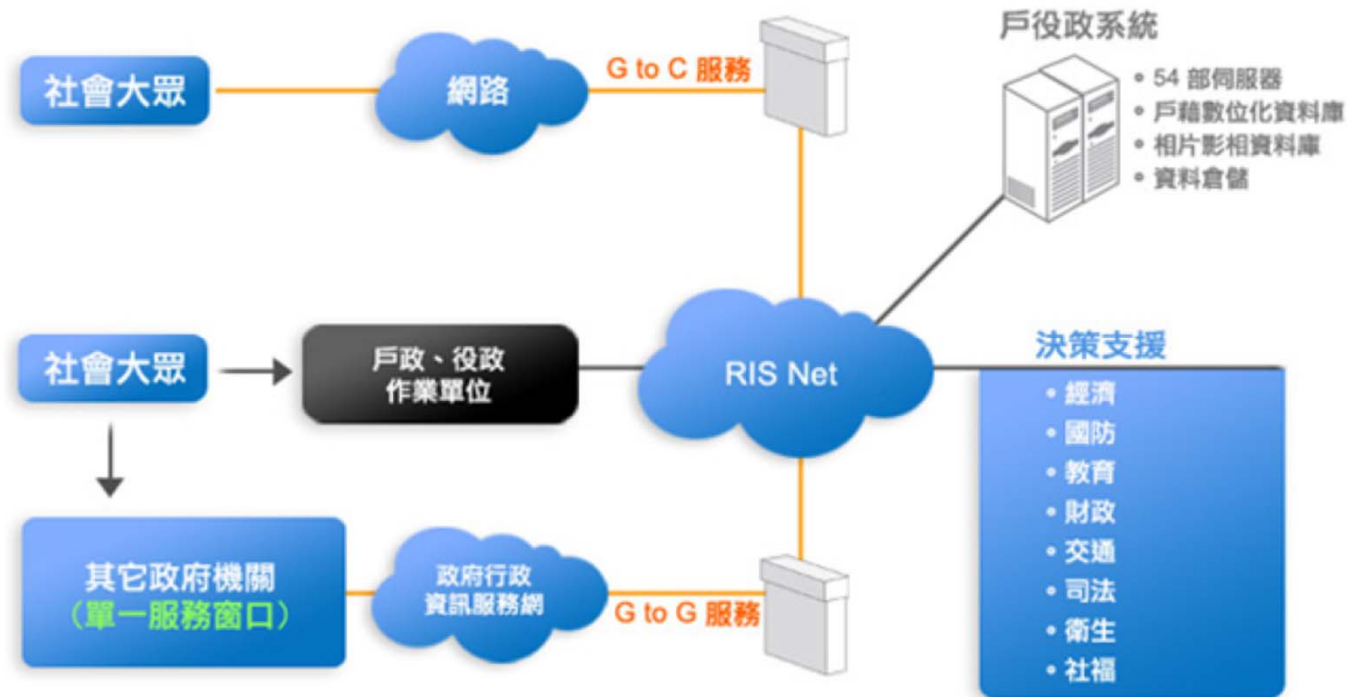
新系統擾民//快開學辦無戶籍 家長跳腳 (2014-2-7)

〔記者林惠琴、劉彥甫、謝佳君、郭安家、謝武雄、張聰秋、蔡清華／綜合報導〕新一代戶役政系統上線，昨天系統時好時壞、作業依舊「龜速」，戶所接獲內政部要求跨縣市戶籍登記業務一律先收件請民眾返家，待辦妥再郵寄，但有人擔心個資外洩乾脆不辦。五日至七日受影響民眾可能達四萬至五萬人。

內政部：要求下週一前恢復正常運作



戶役政資訊系統建置計畫



• 原來宣稱的專案效益

- 創新規劃設計開放式軟體架構（**Open software architecture**）

建立多廠牌電腦系統共融運作環境，應用軟體程式具可攜性（**Portability**），使在不同的廠牌設備環境下，提供一致性的使用者介面及操作方式，簡化人員培訓工作。

- 強化軟體系統品質

系統導入之初由資[REDACTED]獨立軟體測試小組，進行為期**12個月**的獨立軟體測試，並引進先進的軟體測試工具，進行壓力測試、績效測試及負載測試，確保系統上線後，持續維持高品質的運作及服務。

• 結果

段宜康比對標案資料後，昨天拋出三大質疑。第一，**戶政新系統先買硬體、再設計軟體，時間本末倒置，等到新系統上路，硬體早過保固期，導致軟、硬體不相融。**他舉買衣服為例質疑「哪有人先把衣服買好，再找人來穿衣服？」

戶政系統又出包，有民眾不耐久候，質疑戶所是不是把他的號碼偷偷跳掉，讓戶所人員頻頻解釋。

記者周志豪／攝影

 分享

包，最後卻由戶政司發包，明顯是外行領導內行。此外，內政部對外宣稱進行**十二次壓力測試**，但其實根本不是獨立壓力測試，而是內政部自行進行的「**系統測試**」，「內政部公開說謊！」

國際股份有限公司

內部公開信

2014/02/11

發信人：國際股份有限公司(台灣)執行長

一、 近日戶役政資訊系統事件，已有媒體大量報導，同仁均表達關心之意，特此對各位同仁說明。

二、 本公司於此事件所扮演之角色實為搶救雷恩大兵。此次系統不正常運作，原廠 IBM 證實，主因為軟體作業系統(AIX)與中介軟體(WebLogic)版本不相容所致。

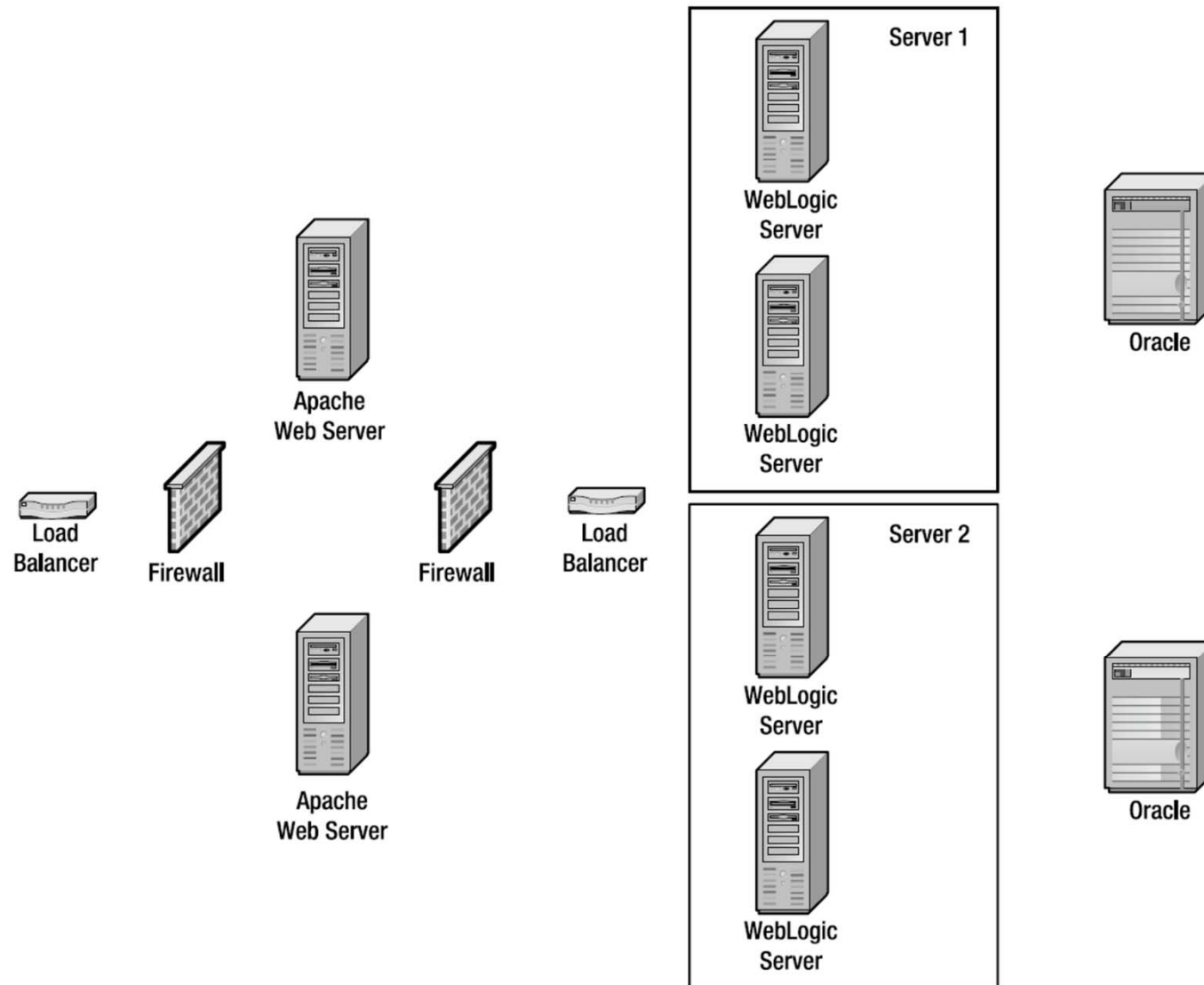
三、 從上週迄今，本公司戶役政團隊做了大量 AP Bypass 及 AP 優化的工作，我們期望戶役政系統在版本不相容問題未解決前，系統能暫時穩定，並提供民眾正常品質的服務。

四、 目前該系統之硬軟體維護廠商「環安達科技實業有限公司」於今日自由時報對本公司所提不實之指控，本公司至感遺憾。該公司背景經本公司人員查證，其 101 年 11 月公司才正式成立，資本額新台幣二佰萬元，向經濟部登記高達 87

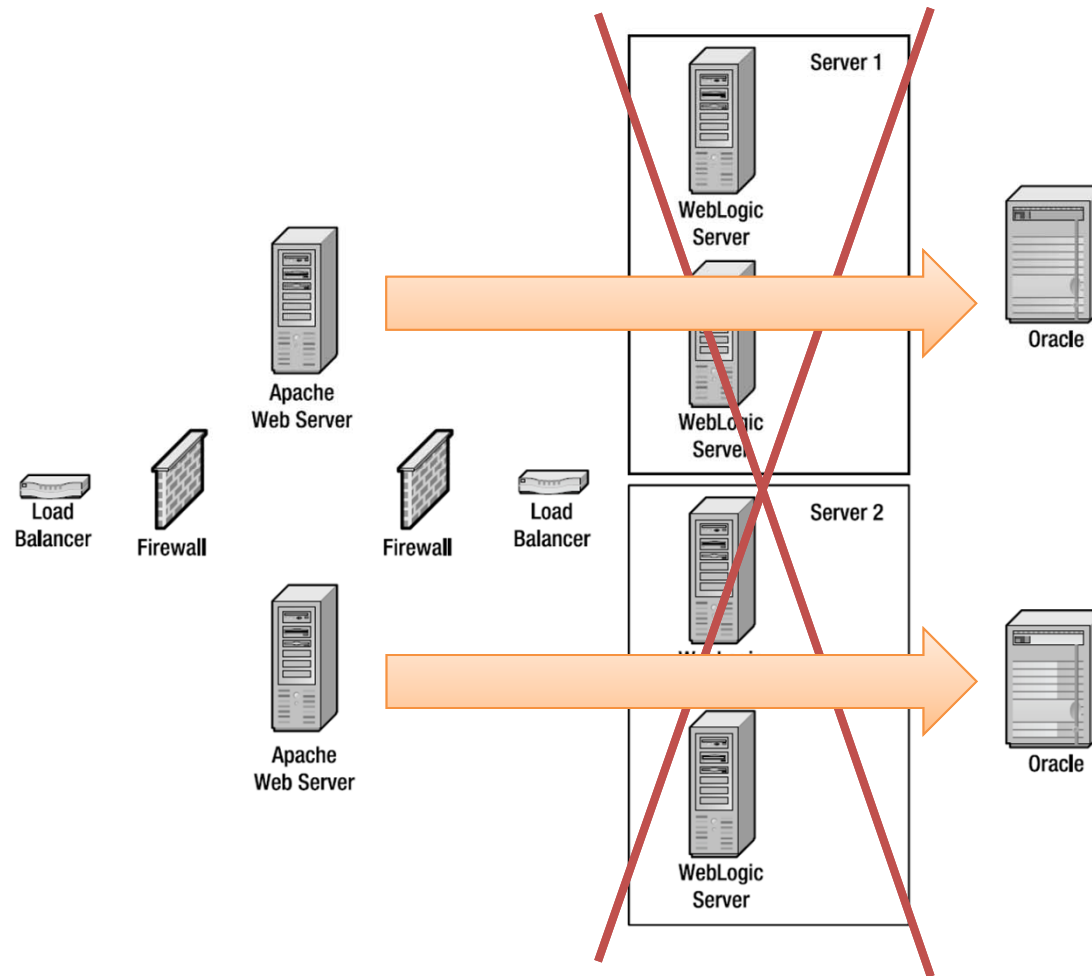
項營業項目。謝愛齡表示，初步調查雖是中介軟體介面出問題，但與這項系統有合約的廠商，還有其他公司，內政部會查清楚到底是那家公司有問題，再依合約處以記點或罰款。

般所謂政府採購黑名單)。此公司發言之公信力，社會當自有公斷。

典型3-layer架構



什麼是AP ByPass ?



等同於完全放棄 WebLogic

WebLogic的優點

- 優點
 - 市場主流。
 - 功能強大。
 - 穩定。
 - 效能高。
 - 擴充性高。
 - 完整「有用」的線上文件。

WebLogic的缺點

- 缺點
 - 貴。
 - Licensing不合理。
 - 管理人員要有經驗，才能發揮強大功能及效能。
 - 上手前要花一段時間學習。

系統優化變因

- Application Server (AP) 本身
- JVM選擇與參數調整
- 資料庫
- Socket與Thread數目
- Connection Pool
- Web Application的程式碼

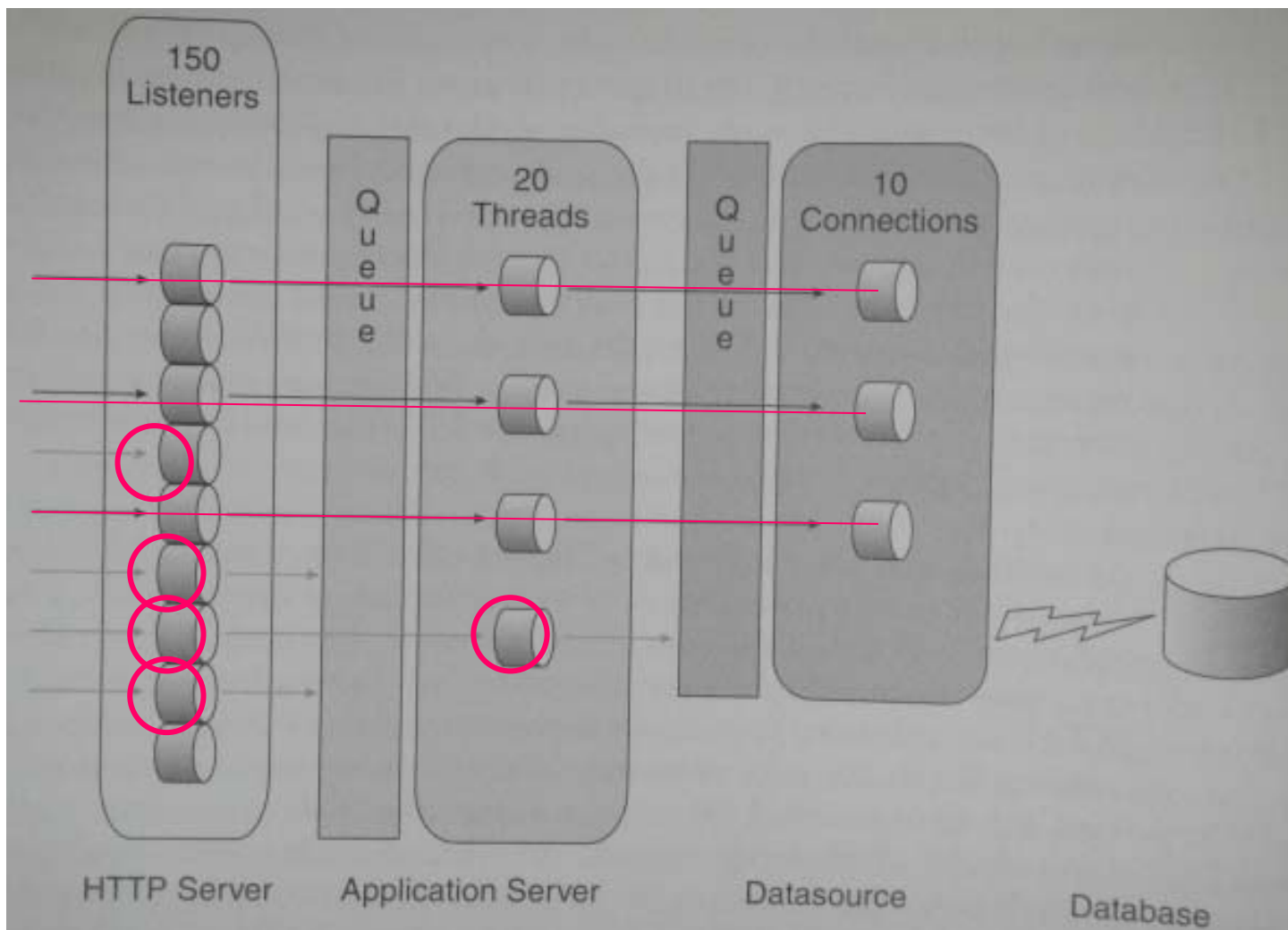
2.4.5. 總結與建議

- (1) 頁面操作有許多的與伺服器互動的設計，導致傳輸量過多，建議適度減少非必要的資料傳輸及非必要的伺服器連結，以降低伺服器的負擔。
- (2) 網頁未完整顯示或頁面元件未完整下載，造成部分連結或元件找不到，可能導致後續功能步驟向伺服器提交錯誤資料或帶有空值(NULL)的請求，建議應有傳輸資料完整性檢核機制及降低伺服器負擔。
- (3) 因系統負荷過高，造成連線逾時，可能導致伺服器存在過多懸置未釋放的連線佔用主機資源，建議請求端與伺服器端應相互配合調整逾時機制，避免懸置未釋放的主機資源持續累增，惡性循環。
- (4) 登入及元件逾時，此現象多為系統回應工作階段逾時，建議檢討負載平衡或伺服器端可能造成使用者仍在操作的工作階段逾時的設定、設計或其他因素。
- (5) 連結介面在250個使用者同時於線上作業的情況下，出現因無法消化查詢請求，而持續累積使用者端發出之查詢請求，導致無法再處理使用者查詢作業，必須重置連結介面，建議應修正停止服務的問題。

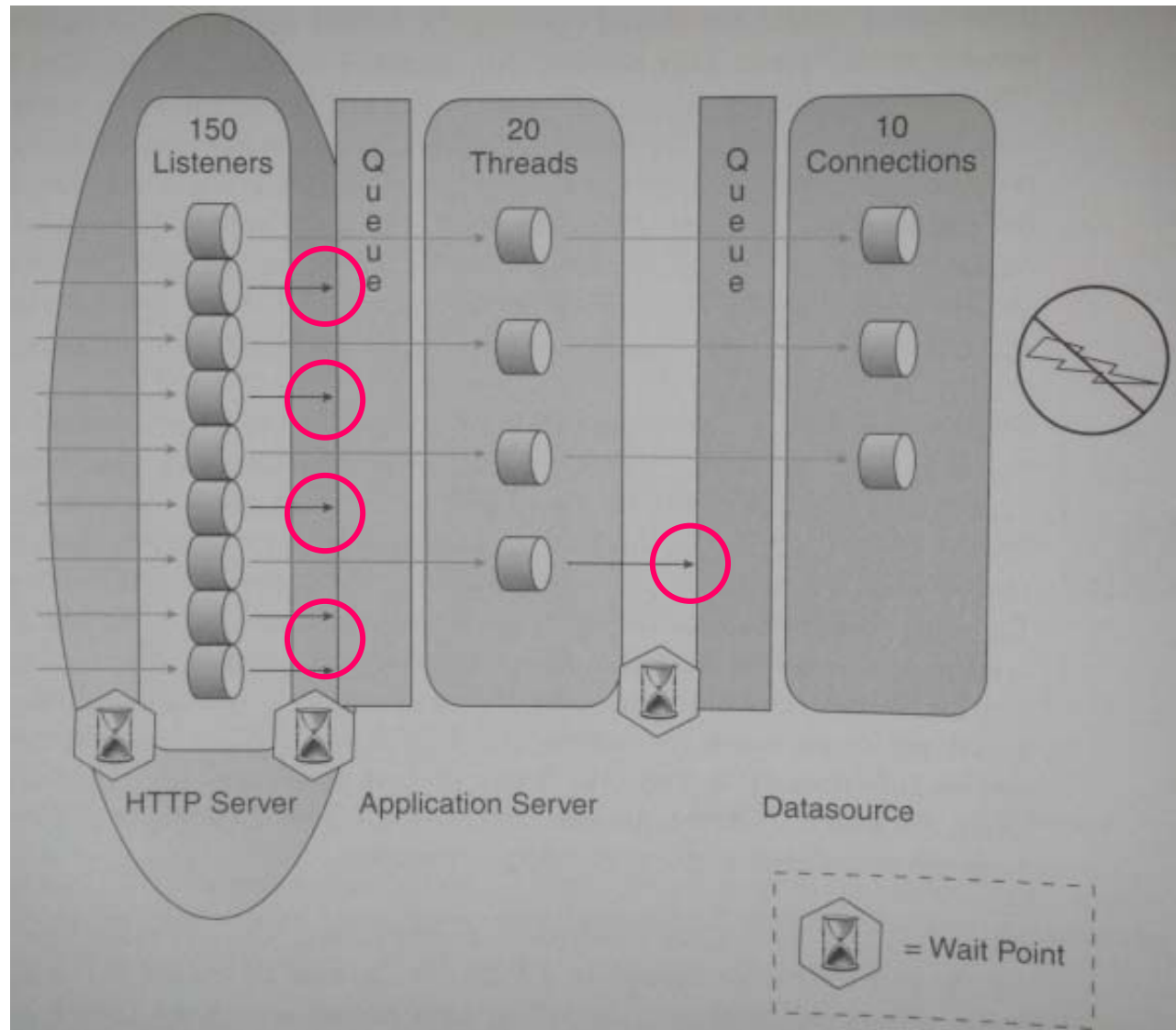
Funnel Effect

縮短session timeout
節省記憶體，降低GC機會

3-layer間負載量的差異



Funnel Effect



Syllabus

- See moodle

Course Evaluation

- 個人工作 (60%)

- 課堂表現 (含點名) 10%
- Take home lab 30%

** 本課程對生成式AI工具之政策：需要時可使用，但需揭露使用時機、用法、用途，對結果的正確性負全責。

- 引導同學透過實作初步了解系統原理，難度均低
- 以moodle時間為準，下週上課前完成; 最高100分; 其它: 0分

- 期中考 20% 選擇題: 考上課講授內容與Lab

- 小組工作 40% (6-8人一組)

- 請於每次報告明列分工

- 期中報告20%: 開源分散式系統解析與試用

- 現場技術解析報告與展示(後面詳述)、現場試用demo (hello world等級即可)
- 要準備投影片，每組10分

- 期末展示20%: 開源分散式系統應用開發

- 現場報告、系統demo

期中報告：開源分散式系統解析與試用

- 目的
 - 善用學期間學得的分散式系統知識，了解最新分散式系統技術進展
- 條件：
 - 建議專案 (如列表)
 - 自行選定: 要符合分散式系統定義的相關技術
 - 一定要是分散式系統且專案須有一定規模(程式碼2000行以上)
 - 經老師同意才可選擇
 - 來源
 - CNCF 專案 (<https://www.cncf.io/>)
 - Apache 專案 (<https://www.apache.org/index.html#projects-list>)
 - 其它在Github中的分散式專案
- 期末展示要包含所選定的專案技術

**** 本課程對生成式AI工具之政策：需要時可使用，但需揭露使用時機、用法、用途，必須對結果的正確性負全責。**

期中報告格式

10

- Introduction
 - Main function of the project, what is the problem it tries to solve?
 - The overall approach taken by the project (core idea)
 - 該分散式系統/技術的應用場景 (Example)，用來說明：
 - 在什麼場合用來解決什麼問題？沒有它又如何？
- 架構解析
 - Explain how this technology works
 - 靜態與動態結構解析 (請使用UML Class/Component/Sequence/Activity diagrams)配合文字說明
 - 此設計如何在某個背景下解決設計問題(如何達成目的)
- 實驗紀錄與心得
 - How to install, configure and use? (安裝並使用該技術實作一個簡單的範例的過程記錄)
 - Demo: A hello world example for this technology，心得與建議
- Evaluation and comments
 - 和相近技術的比較: Why this work is better than others? In which respect?
 - Does the proposed approach solves the problem? How and how well?
 - What can be enhanced?
- 若有使用生成式AI輔助，請明確註明產生之範圍和使用方式

期中報告內容 (A4 4-10頁, pdf)

警告: 未按格式寫作或短少項目會嚴重扣分!

- 問題(目的)與背景
 - Briefly explain the core idea of the technology
 - 該分散式系統/技術的應用場景 (Example)，用來說明：
 - 在什麼場合用來解決什麼問題？
 - 沒有它又如何？
- 架構解析
 - Explain how this technology works
 - 靜態與動態結構解析 (請使用UML Class/Component/Sequence/Activity diagrams)配合文字說明
 - 此設計如何在某個背景下解決設計問題(達成目的)
- 技術實作與心得
 - 實驗記錄: 安裝並使用該技術實作一個簡單的範例的過程記錄
 - 使用心得
- 評論
 - Identify a context where this technology is not appropriate and justify your answer
 - Identify a context where this technology is appropriate and justify your answer
- 結語: A brief conclusion and lesson learned

建議專案

- Apache Qpid <https://qpid.apache.org/>
- Apache OFBiz <https://ofbiz.apache.org/>
- Apache IoTDB <https://iotdb.apache.org/>
- YugabyteDB <https://github.com/yugabyte>
- Apache Traffic Control <https://trafficcontrol.apache.org/>
- Apache Kafka <https://kafka.apache.org/>
- Apache Pulsar <https://pulsar.apache.org/>
- Apache Cassandra <https://cassandra.apache.org/>
- Jgroups <http://www.jgroups.org/>
- Apache Camel <https://camel.apache.org/>
- Apache Ignite <https://ignite.apache.org/>
- Apache Mina <https://mina.apache.org/>
- Apache Vysper <https://mina.apache.org/vysper-project/>
- Apache Zookeeper <https://zookeeper.apache.org/>
- Apache Helix <https://helix.apache.org/>

建議專案

- gNMI <https://github.com/openconfig/gnmi>
- Cockroachdb <https://github.com/cockroachdb/cockroach>
- Keycloak <https://www.keycloak.org/>
- Fluentd <https://www.fluentd.org/>
- ContainerD <https://containerd.io/>
- KEDA <https://www.cncf.io/projects/keda/>
- SPIFFE <https://github.com/spiffe/spiffe>
- Operator Framework <https://www.cncf.io/projects/operator-framework/>
- TiDB <https://github.com/pingcap/tidb>
- Event Store <https://www.eventstore.com/eventstoredb>
- Etcd <https://etcd.io/>
- OpenTelemetry <https://opentelemetry.io/>
- Envoy <https://www.envoyproxy.io/>
- Vitess <https://vitess.io/>

期末Demo

- 目的

- 應用期中報告的技術，開發一個實際應用

- 限制

- 要能實際run
- 要應用到期中報告的技術
- 至少3個節點
 - 可以同一台機器，三個Processes、且它們只透過網路溝通

- 要求

- 功能性: 要具有意義的應用 (不能是hello word)
- (加分題) 非功能性: 實現至少一項分散式系統樣式 (容錯、系統偵測、自動回復、隨需伸縮...)
 - 由老師評定，根據規模與難度加本項成績2~4分，成績上限99分
 - 必須是自己額外寫的code不能是直接用tool、可以改寫他人源碼，但要揭露來源和改寫程度 (若要實作，請先與老師討論擬實作的具體想法)
 - 請參考: <https://martinfowler.com/articles/patterns-of-distributed-systems/>

- 如何繳交 (評分依據)

未含影片與原始碼Github網址者不予計分!

- 口頭報告與demo
 - 投影片要上傳到moodle (pdf); demo以影片錄製，上傳到雲端，連結附在投影片上)
- 原始碼請放到小組其中一員的Github公開repository; 在readme.md中詳述安裝與操作步驟
 - Github網址附在投影片中

使用生成式AI於本課程須知

- Know the backgrounds
 - 必須具備該領域一定程度的實力，才有辦法看出AI所提供不正確的回答/資訊
- Identify the outcomes
 - 腦中要有清楚的方向感，才有能力引導、檢查與修正生成的產出
- Supplement, Not Replace
 - GPT是用來輔助學習和研究，不是依賴它來完成所有工作或取代人的創造與思考

使用生成式AI於本課程須知 (2)

生成程式碼

- 依據提供的特定需求，生成不同程式語言的程式碼 (Golgiyaz, 2023)

完成程式碼

- ChatGPT 和 Github Copilot 可建議下一行程式以完整程式碼
- 可透過提問、提供程式用途、期望結果、實踐細節等訊息更有效生成程式

(Golgiyaz, 2023)

轉換程式語言

- 自動在程式語言間轉換，無需使用者知識 (Golgiyaz, 2023)

錯誤檢查與除錯

- 辨識程式錯誤，提出解決方案以解決句法或詞法的錯誤 (Golgiyaz, 2023; Rahman & Watanobe, 2023)

使用生成式AI於本課程須知 (3)

解釋程式概念

- 為程式概念提供易懂的解釋、範例和 **pseudocode (虛擬碼)** (Rahman & Watanobe, 2023)

程式碼最佳化

- 以減少記憶體和時間複雜度的方法優化程式，依程式碼審查提供解釋 (Rahman & Watanobe, 2023)

創建軟體文件與 執行軟體測試

- 自動生成軟體文件，節省時間和人力
- 生成測試案例，自動化軟體測試 (Golgiyaz, 2023)

Rahman and Watanobe (2023) 的調查指出

- **92.9%** 的學生認為 ChatGPT 作為**輔助程式學習**是有用的
- **60%** 的教師將 ChatGPT **用於程式教學**和研究，並對所提供的建議感到**滿意**

應用限制

- 正確性有限
 - 一項「根據演算法與資料結構的問題描述生成程式」的實驗顯示，ChatGPT 生成的程式正確率平均為 **85.42%**，仍需調整以成功編譯 (Rahman & Watanobe, 2023)
 - ChatGPT 回答軟體測試習題，**55.6%** 的情況能提供正確或部分正確答案 (Jalil, Rafi, LaToza, Moran, & Lam, 2023)
 - 一項調查指出，約 **50%** 教師對 ChatGPT 的滿意度是 **3/5**，其在程式設計教育還未獲得足夠信任
 - ▶ 教師實際經驗：「ChatGPT 仍無法完美回答 Java 課程的某些考試」 (Rahman & Watanobe, 2023)
- 難以完成複雜任務
 - 較難完成具創造力、商業邏輯優勢的應用軟體 (Marr, 2023)
 - ▶ e.g. 不能要求 ChatGPT 「建立一個比 Amazon 更有效銷售的電子商務平台」，人類仍需花時間了解 Amazon 的商業優勢，並找出更好的方法
 - 利用 ChatGPT 可短時間解決簡單程式題，但利用 ChatGPT 解複雜題目需花比人類更長時間 (Qureshi, 2023)

建議

- 仍需學習基本理論與技術

任何資源或工具的有效性都受到個人現有知識和背景的限制，技術放大現有的人類力量和意圖，而一個人只能在能夠理解和有效使用資源的範圍內從資源中受益 (Toyama, 2015)

- 仍需教迴圈、方法和變數，初學者過度依賴生成式 AI 可能會阻礙學習進程或產生誤解

(Eland, 2023)

- 學習程式有助了解何時、何地 and 如何使用工具來建立最佳軟體(Eland, 2023)

- 避免聚焦理論、程式或系統等生成式 AI 的專長，而是著眼高層次抽象概念(Goings, 2023; Welsh, 2022)

- 將 ChatGPT 納入教學

- 輔導學生如何有效、正確使用生成式 AI 工具(Hazzan, 2023)

💡 ChatGPT 可輔助學生學習，但學生仍需具備基本知識/技術以指示其做事、評估輸出品質(Eland, 2023)

總結：生成式AI用於這門課

- 解釋程式碼/文本
 - 課堂講義或範例程式中，有不懂的時候
 - 程式實作時，碰到卡住的時候
 - 摘要英文文本
- 生成程式碼/文本
 - 將我們所需要的功能寫成明確的prompt
 - 記得檢核所生成的程式碼/文本是否**正確**
 - 記得檢核所生成的程式碼/文本是否**符合所需**
- 不要做的事
 - 不要將題目直接丟給GPT解
 - 作業可交差但你會學不到東西



Technology Trends 2023: The Competence Challenge

The market for software engineers is changing. During the pandemic lockdown, software experts could select jobs because digital transformation was rapidly growing and so the need for capacity.⁶ Starting in 2022, Layoffs.fyi and other employment-tracking portals, showed an increase in software engineers being laid off. Tech giants, such as Google, Facebook, and Amazon, started the trend. Many companies

e.g., Google: "No Jerks" policy

自學能力、跨域融通、創造力、整合力、
溝通能力、社交能力

tion and test methods, and software process.⁴ Soft competencies show the biggest gaps, such as communication skills and awareness of social and ethical implications of software engineering work.⁵

Without the right competencies, companies face quality problems in the market. They also lack the capability to deliver innovative solutions and thus fall behind.⁶ Innovation is hampered by a lack of

Final Reminder

- Be sure to check moodle website constantly
 - All important course information and materials will be announced through this website
- Bring your laptop if you have one
- The slides will be posted on-line after each lecture
- Important Dates
 - 3/4 繳交分組名單 (一組6-8人)
 - 4/15 Midterm
 - 4/29 期中報告口試與現場demo
 - 6/17 期末demo
 - 6/21 期末demo原始碼/slide/影片 上傳期限