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

Distributed Systems

通常工程居多

- Motivation
  - Resource and information sharing
    - Ex: Infrastructure/ platform/ service/ storage/ data provision
  - CSCW
    - Ex: **團隊協作、電子商務**

關鍵特色：網路、合作

- 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

Consequences of Distributed Systems

難點

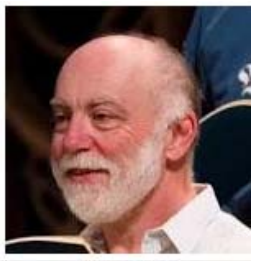
- Concurrency 平行可能產生 **race condition**
  - Concurrent processes is the norm
    - Computers carry out tasks simultaneously and independently
    - Coordination of the concurrent processes is a recurring topic
- No global clock 無法做到精確 分散式沒有辦法用 clock 使大家同時做 Single host app sync by CPU cycle
  - No single global notion of the correct time
  - Coordinate by exchanging messages
- Independent failures 死了還是很慢
  - **分散式系統上無法區分「crash」還是「running slow」**
  - Dealing with distributed failure is the responsibility of developers

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

理想的假設



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。

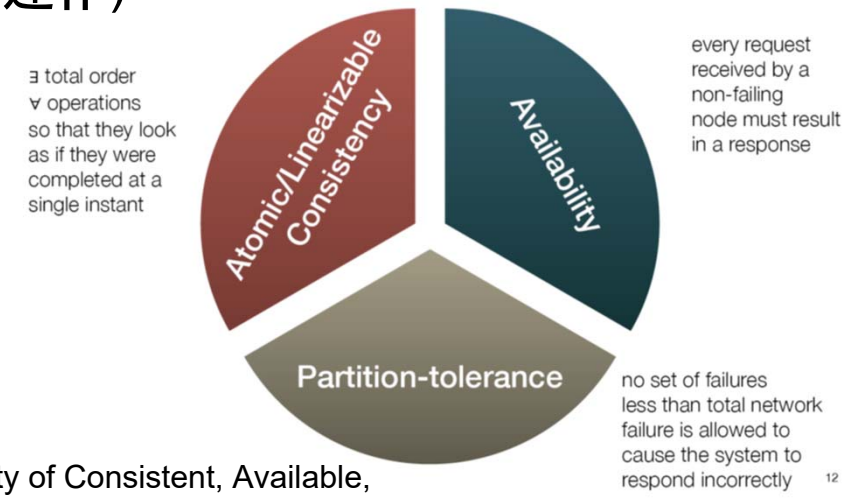
Brewer's Conjecture：CAP

同一時刻為同一瞬間

- 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　　只能三選二
  - Consistency (狀態、資料一致)
  - Availability (不斷提供服務)
  - Partition Tolerable (斷網不影響運作)

可以用延遲方式依次滿足三個

第三點是分散式系統必選的



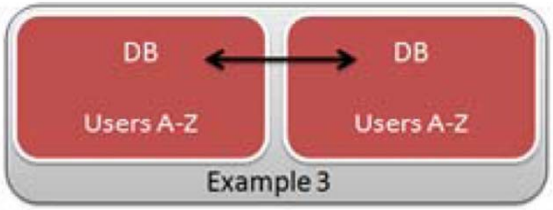
Seth Gilbert and Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. ACM SIGACT News, vol. 33, no. 2, 2002, p. 51-59.

分散式系統一定要此項
所以該項不能 DROP

Drop Partition Tolerant (不能斷網)

分區容忍，如果其中一個壞掉，其它還能確保正常運作
(此處就是指網路封包的正確性)

- 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

短時間內網域會掛掉

Drop Availability

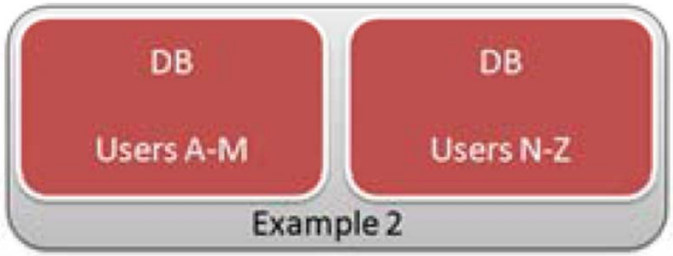
確保東西正確，但穩定度不佳

- 做法:
  - 假設網路中沒有node會死掉，或是死了也沒關係
  - 不需redundancy
- 結果

每個東西只有一份

  - 確保C: 一份資料只保留在一處→達成Consistency 可確保→ 無SPOF

SPOF: Single Point of Failure



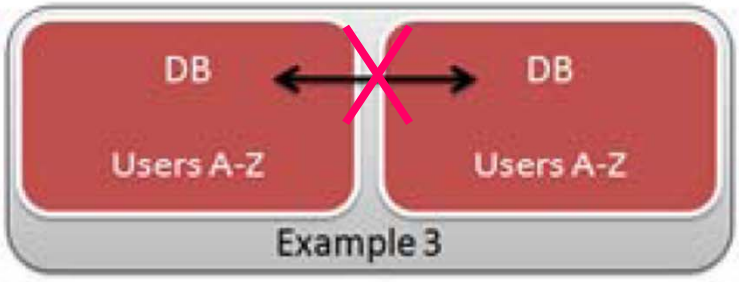
Consistent

  - 確保P: Network 可以partition→沒有sync必要，二者不相連也沒關係

不同人、不同時間對相同的東西
會查到不同的資料

Drop Consistency

- 做法:
  - Nodes中的states可不一致 可能不一致 但能順跑
- 結果
  - 確保P: 不需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

一定要網路(P)

- "The reality is that you will always have network partitions"
  - Leaves only two choices
    - Optimize either for consistency or high availability
  - Choose Consistency

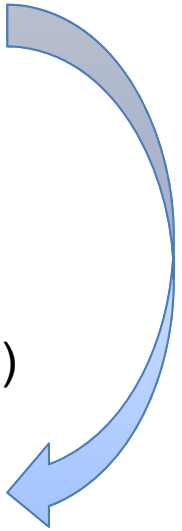
~2015

    - Drop availability → Focus on MTTR
    - Centralized design (SPOF)
    - Easier to understand and design
  - Choose Availability

系統非常複雜

2016~

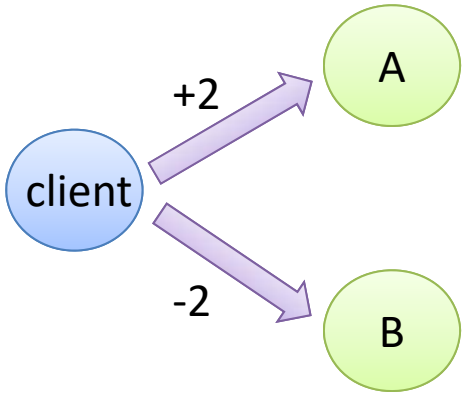
    - Drop consistency (enable eventually consistency)
    - De-centralized design
    - Hard to design, ex: blockchain



Cloud Native Transition

Other Issues of Distributed Systems

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



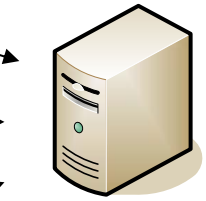
Theme of this Course

- 呈現與論述分散式系統設計 足以評論別人系統
  - 分散式架構(Architecture)與範式(Patterns)
  - 使用UML (Unified Modeling Language)
  - 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 網路
  - Direct and indirect communication 直接/間接通訊
  - HTTP and RESTful WS http
  - Resource management
  - Cloud native + DevOps
    - Observability/ IaC/ CQRS and Event Sourcing
    - SOA and Microservices
  - (optional) Blockchain

Reasoning for the Design of Systems

1. Know the context 知道背景
2. Make a claim (i.e. a comment or an idea) 什麼假設下去評論
3. Justify your claim 量化或質性的講解有什麼問題

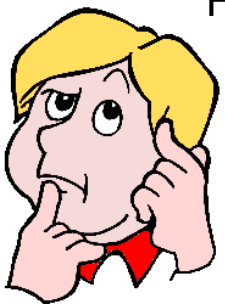
Please present your claim “logically”.
No emotional terms.



Database

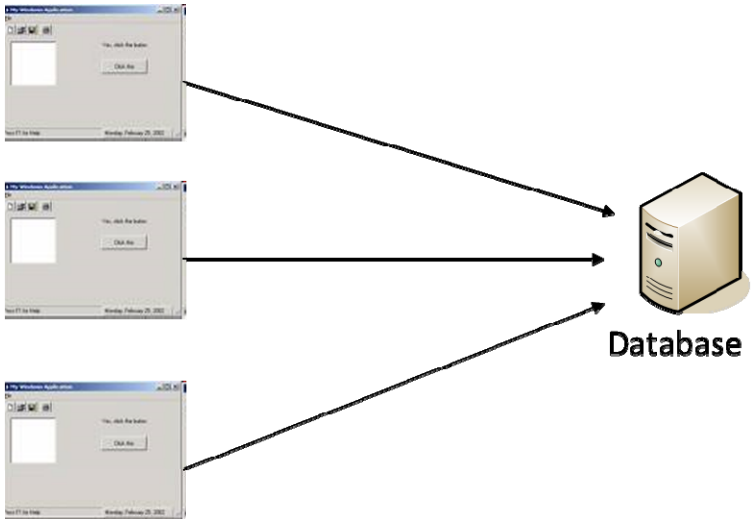
Please present your idea “logically”.
No emotional terms.

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



背景： client 數很多時
問題：可能負擔不住資料庫

An Example



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

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

評論 (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年規劃全面汰換爲JAVA程式語言，2009年起將28項系統簡化爲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。

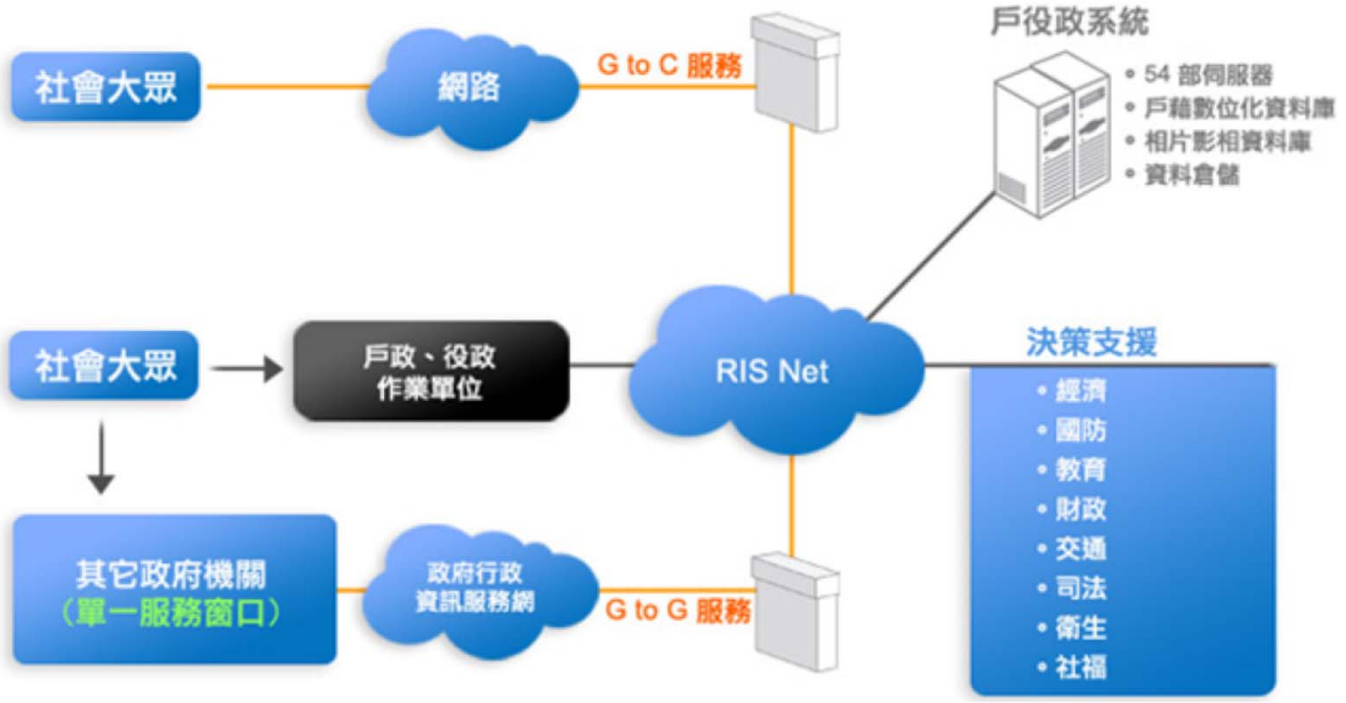
謝愛齡表示，內政部目前已知受影響民眾，是透過1996專線反映及抱怨，共有一百二十七件，其他未透過1996的申請民眾可能也不少，主要都是申請遷出、遷入或結離婚的案件，而且出問題的都在跨縣市，她對民眾所帶來的不便，深感抱歉。

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



資料來源：資拓宏宇網站

- **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
  系統導入之初由資拓宏宇之獨立軟體測試小組，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**

對的軟體也可能有不好的品質

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

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？」

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！」

資拓宏宇國際股份有限公司

內部公開信

2014/02/11

發信人：資拓宏宇國際股份有限公司(台灣)執行長沈安石

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。

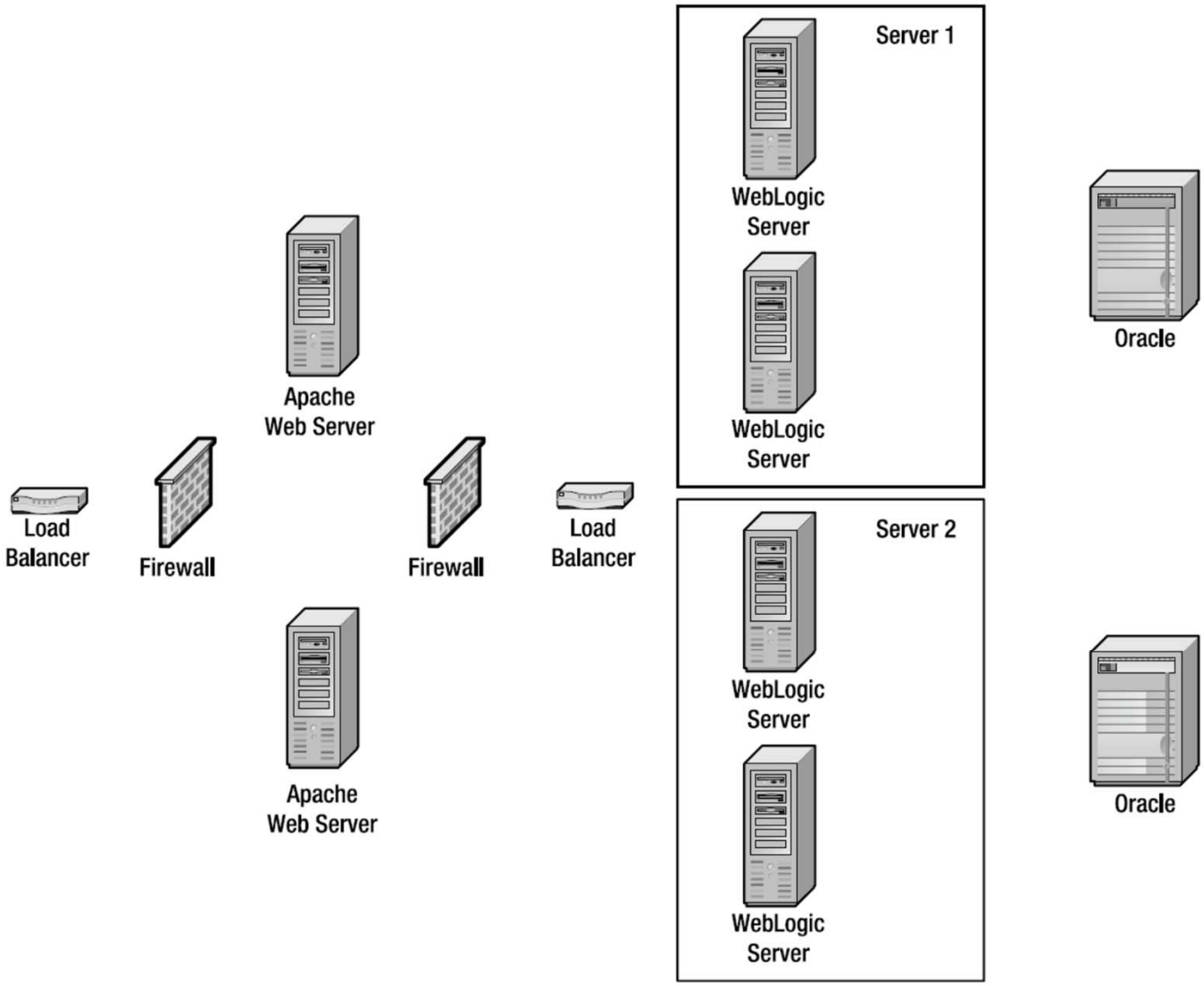
三、 從上週迄今，本公司戶役政團隊做了大量 AP Bypass 及 AP 優化的工作，我們期望戶役政系統在版本不相容問題未解決前，系統能暫時穩定，並提供民眾正常品質的服務。

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(一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。

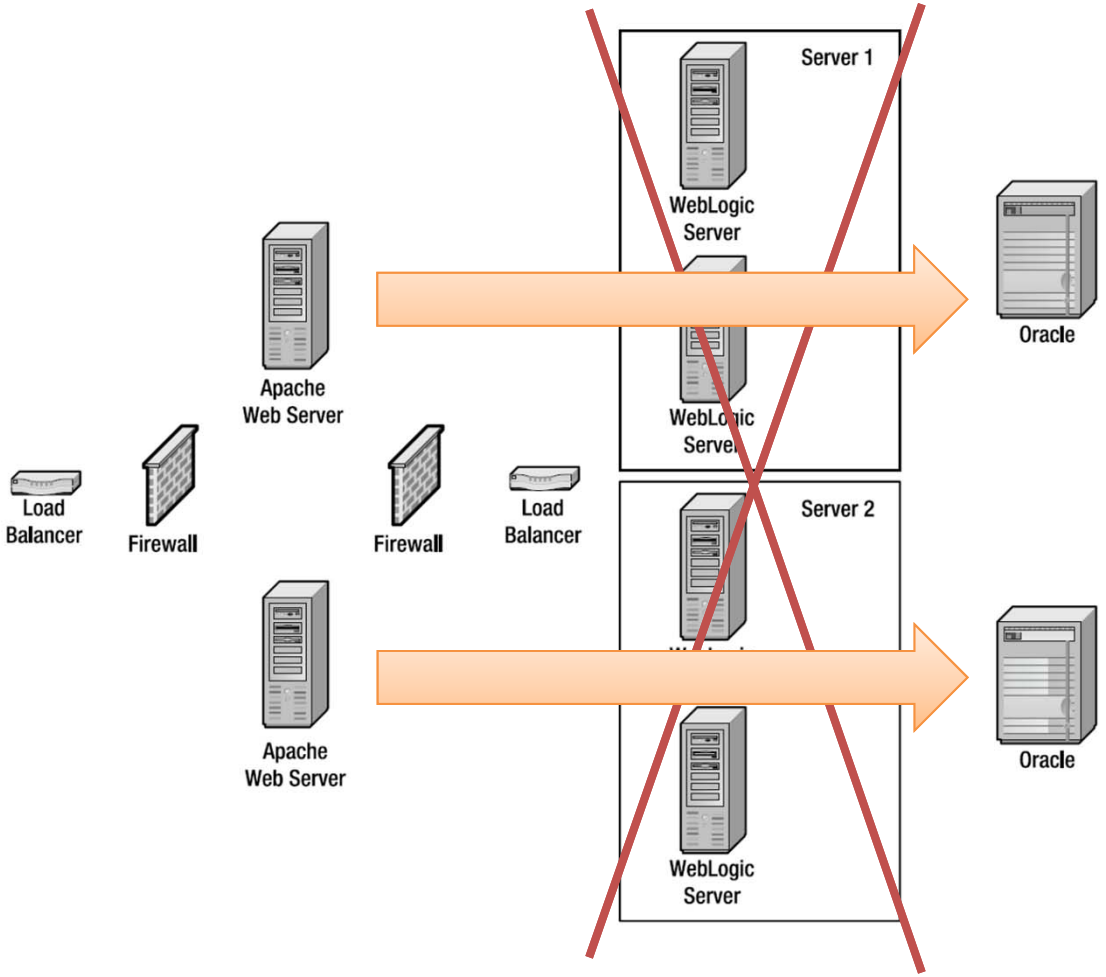
謝愛齡表示，初步調查雖是中介軟體介面出問題，但與這項系統有合約的廠商，還有其他公司，內政部會查清楚到底是那家公司有問題，再依合約處以記點或罰款。

程委員會

典型3-layer架構

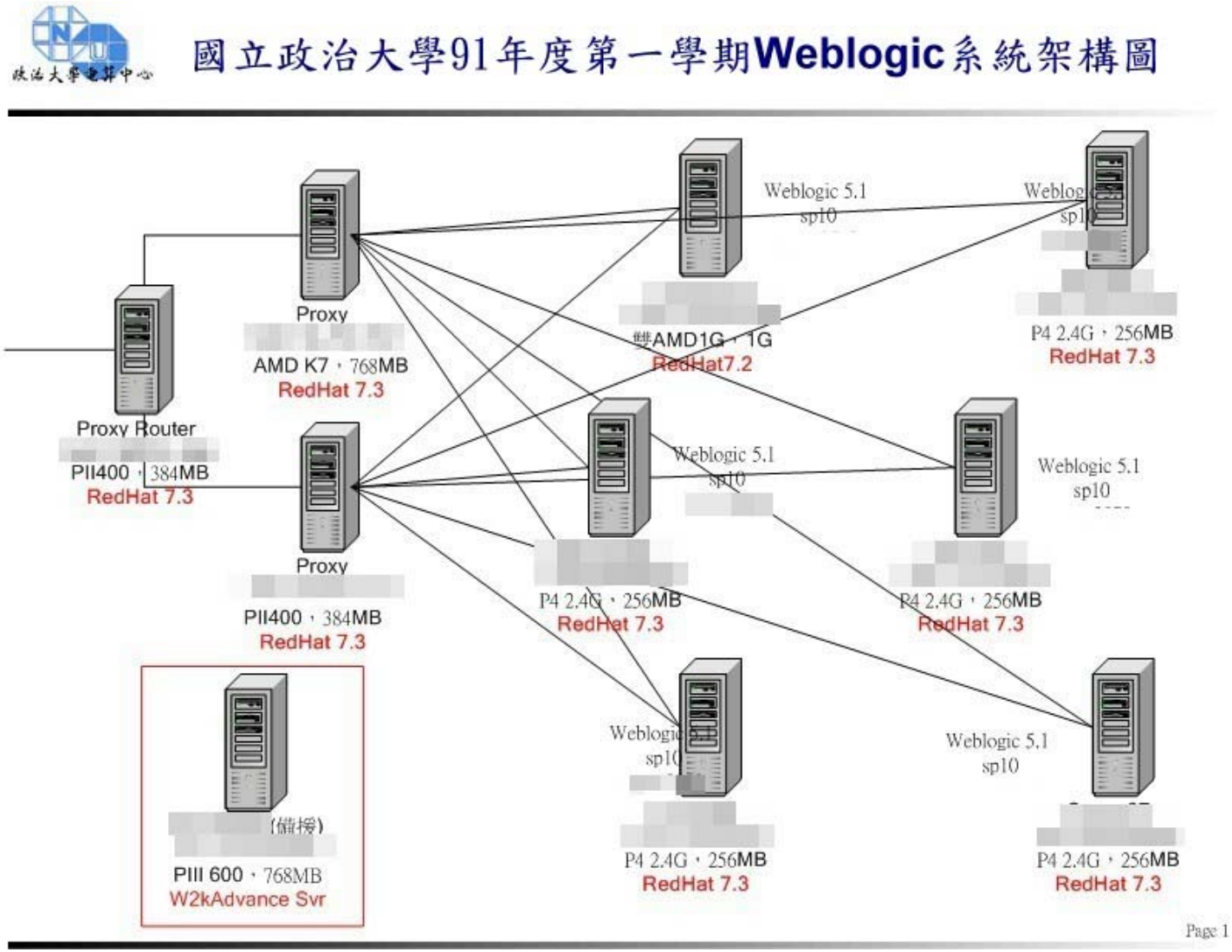

Server 1
WebLogic Server
WebLogic Server
Oracle
Apache Web Server
Load Balancer
Firewall
Firewall
Load Balancer
Server 2
WebLogic Server
WebLogic Server
Apache Web Server
Oracle

什麼是AP ByPass ?



等同於完全放棄 WebLogic !

Example：選課系統



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。

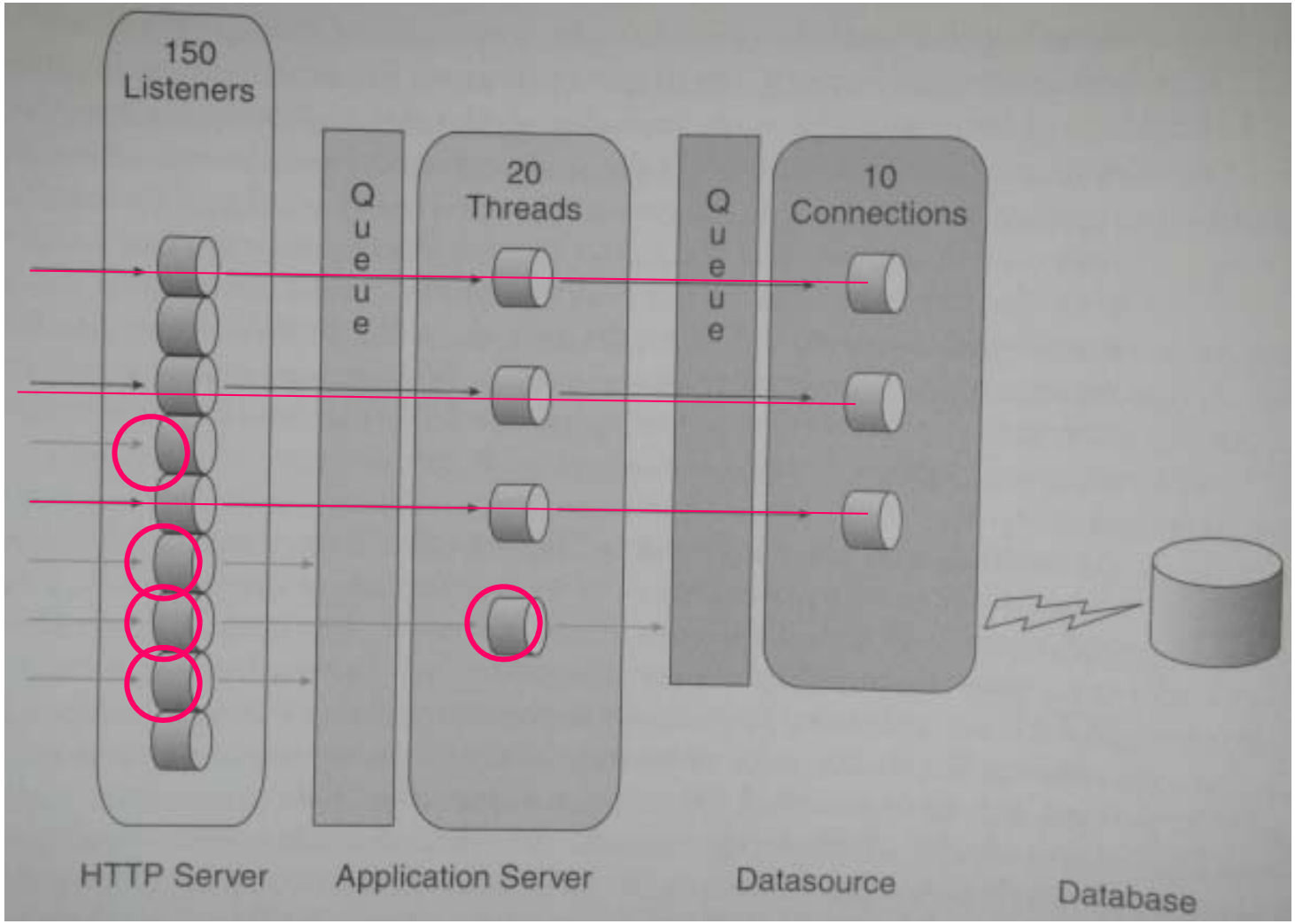
(5) 連結介面在250個使用者同時於線上作業的情況下，出現因無法消化查詢請求，而持續累積使用者端發出之查詢請求，導致無法再處理使用者查詢作業，必須重置連結介面，建議應修正停止服務的問題。

Funnel Effect

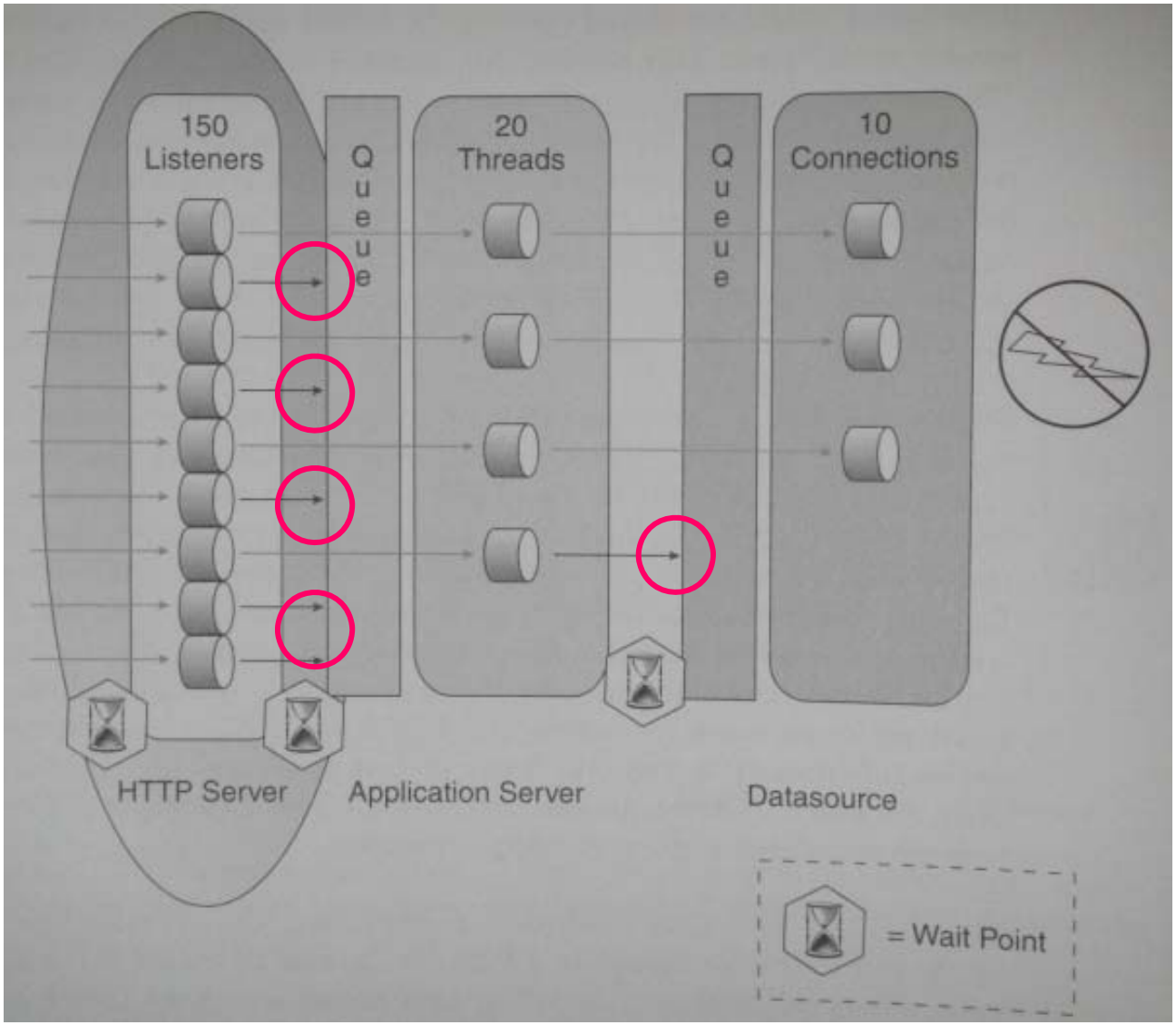
縮短session timeout
節省記憶體，降低GC機會

3-layer間負載量的差異

分散負荷，中間要放 queue 擋住
否則若負荷過重可能造成效率低落



Funnel Effect

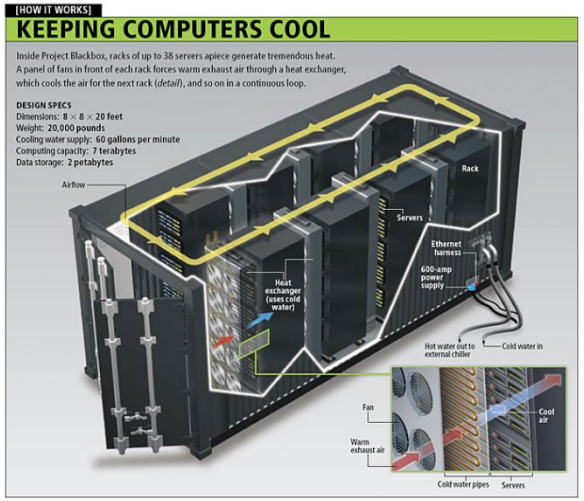


Case: Google Data Center

- 定義
  - A facility to house a large collection of computers, networking, and communications equipment
- Four (computer system) design issues
  - Structure
  - Scalability
  - Flexibility
  - Power efficiency

Data Center

- 伺服器置放於 19-inch rack cabinets (標準規格)
  - 1櫃可放42 1U伺服器
  - 成排單行放置，中間走道相隔
  - 在超大型Data Center中，可以貨櫃模組化



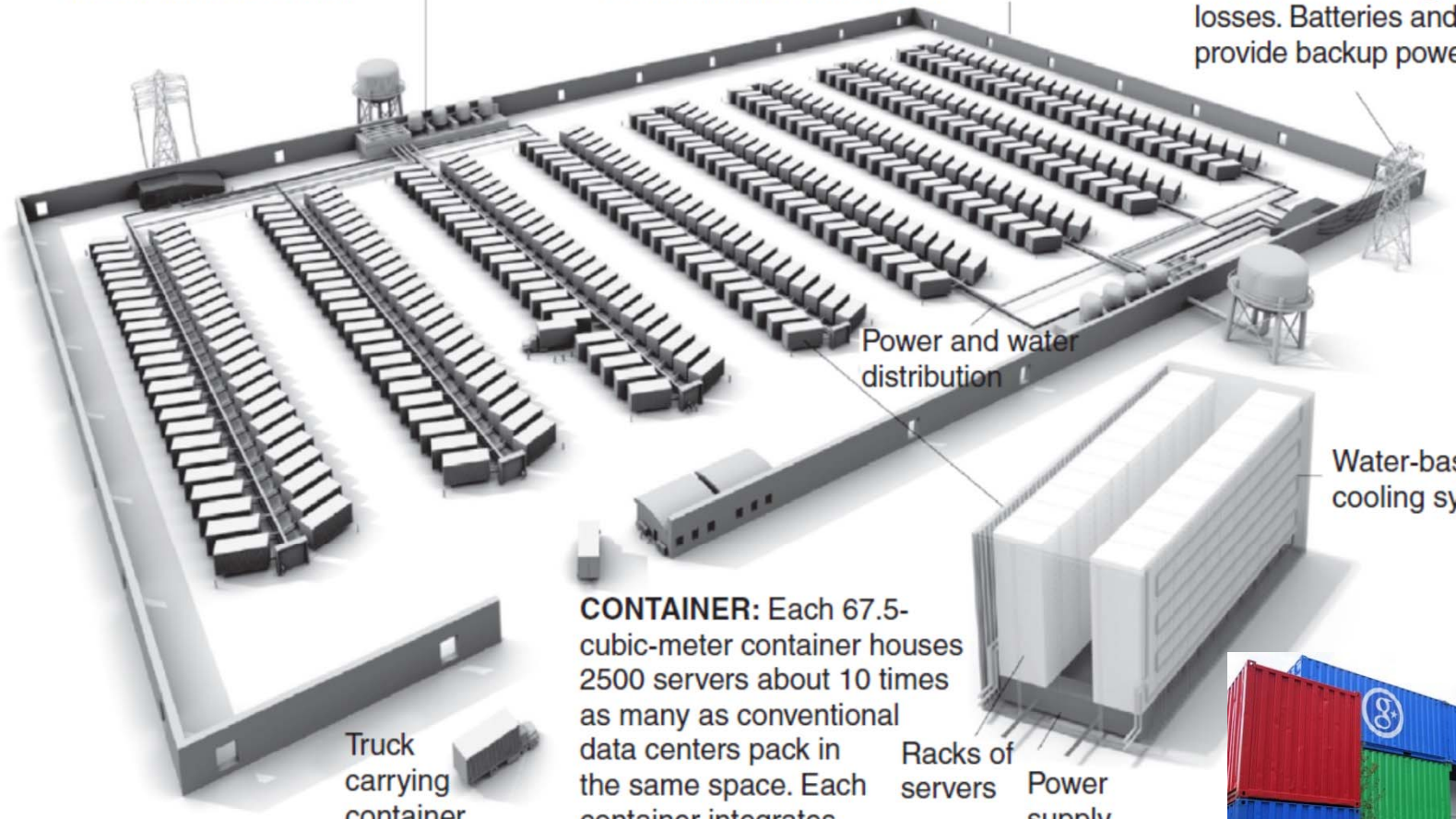
Data Center Requirements

- Unwavering power
  - Should be prepared to handle brownouts (電力短少) and even power outages (斷電)
  - Backup batteries and diesel (柴油) generators must be available to keep power flowing
- Broadband network connectivity
- Security
  - Firewalls, VPN gateways, intrusion-detection software

COOLING: High-efficiency water-based cooling systems–less energy-intensive than traditional chillers–circulate cold water through the containers to remove heat, eliminating the need for air-conditioned rooms.

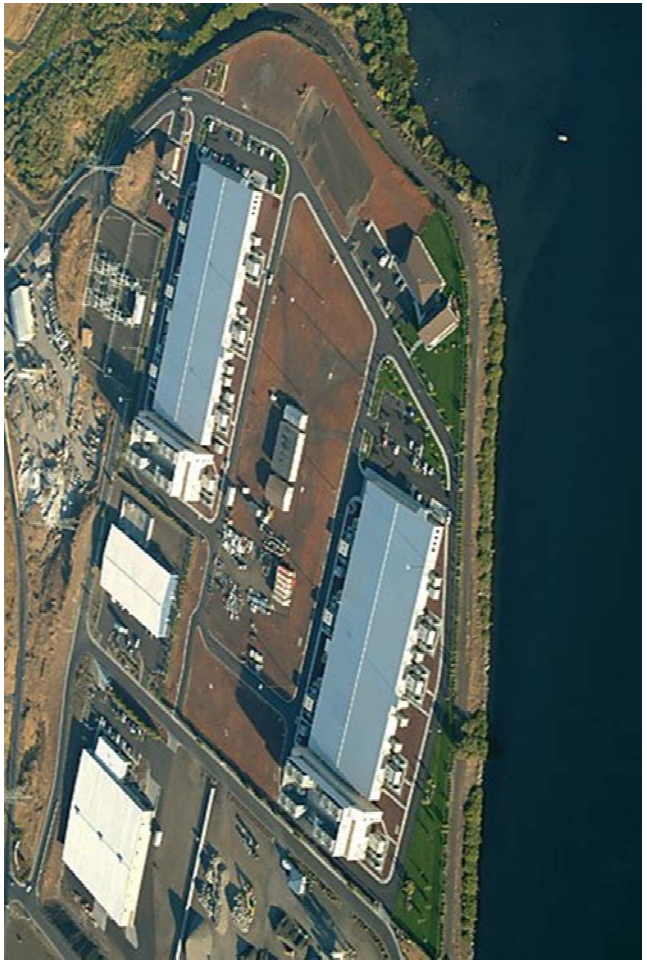
STRUCTURE: A 24 000-square-meter facility houses 400 containers. Delivered by trucks, the containers attach to a spine infrastructure that feeds network connectivity, power, and water. The data center has no conventional raised floors.

POWER: Two power substations feed a total of 300 megawatts to the data center, with 200 MW used for computing equipment and 100 MW for cooling and electrical losses. Batteries and generators provide backup power.



CONTAINER: Each 67.5-cubic-meter container houses 2500 servers about 10 times as many as conventional data centers pack in the same space. Each container integrates computing, networking, power, and cooling systems.

Google Dallas Data Center



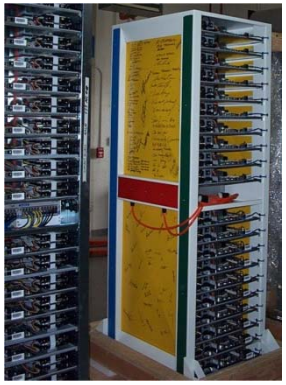
Google Data Centers

- 36 data centers / 500 Ips (2013-彰化; 2019-台南; 2021-雲林)
- Continuous evolution: 7 significant revisions in last 10 years
- An ordinary search query involves 700 to 1,000 servers

Inside a Google Data Center (2003)

- Google servers are custom made
- A rack (機櫃) consists of 40 to 80 x86-based servers
  - 1 rack = 20 2u or = 40 1u
- CPU
  - Intel-Celeron 533 MHz
  - Intel PIII Dual 1.4G
- HDD: IDE 80G
  - Index server has less disk space (cpu-intensive)
  - Doc server has more disk space
- Network
  - Core: Gigabyte Ethernet
  - Rack: 100 Mbps Ethernet

Inside a Google Data Center (2009)



- A small data center consists of a minimum of 2,400 servers
  - racks of 80 servers tied together with 10Gb Ethernet or other high-speed network fabrics
  - 30 or more of these racks are deployed into a single cluster
- Each of these servers has 16GBs of RAM with fast 2TB (Terabyte) hard drives
  - A patent on a power supply that integrates a battery, allowing it to function as an uninterruptible power supply (UPS)
  - Google-optimized Ubuntu Linux

YouTube - Google container data center tour (2009.4.7)

Google Search Engine

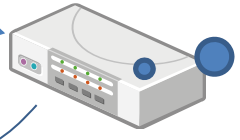
- Google查詢需耗費大量計算與I/O資源
  - 讀取 > 數百MB資料
  - 計算需耗 > 數十億CPU cycles
- Google 搜尋引擎如何建置
  - 硬體 > 15,000 一般PC
  - 軟體：需要特殊設計
    - Energy efficiency
    - Price-performance ratio

Google Search Engine

- 針對巨量資料特別設計的軟體架構
  - 在不穩定的硬體上，透過軟體架構設計建立穩定的系統
  - 二個關鍵理念
    - Provide reliability in software rather than server-class hardware
      - 以軟體取代硬體的部份
        - » Replication
        - » Failure detection
    - Tailor design for best throughput

http://www.google.com/search?q=nccu

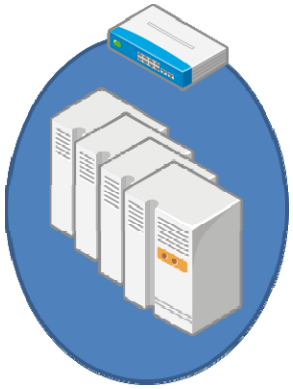
1) nslookup www.google.com



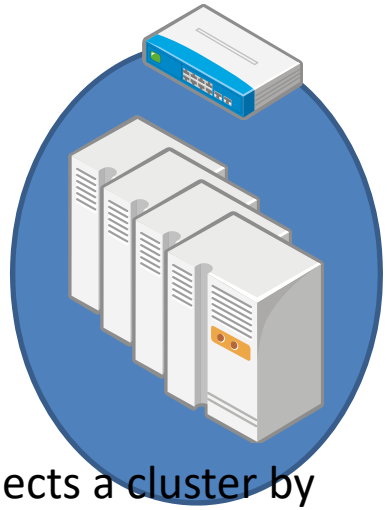
2) 173.194.127.50

3) http://173.194.127.50/search?q=nccu

173.194.127.50



173.194.127.51

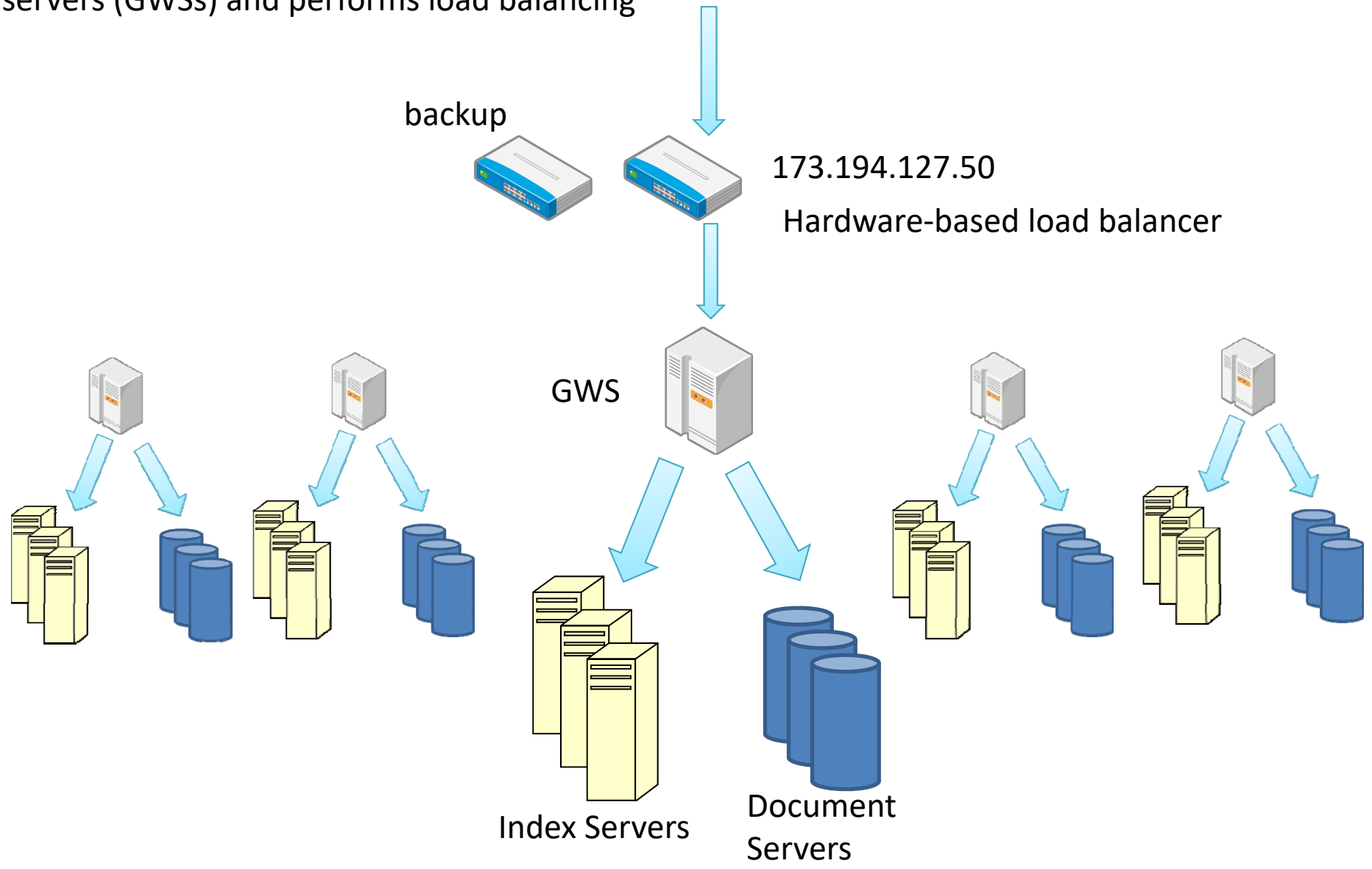


173.194.127.52

173.194.127.48

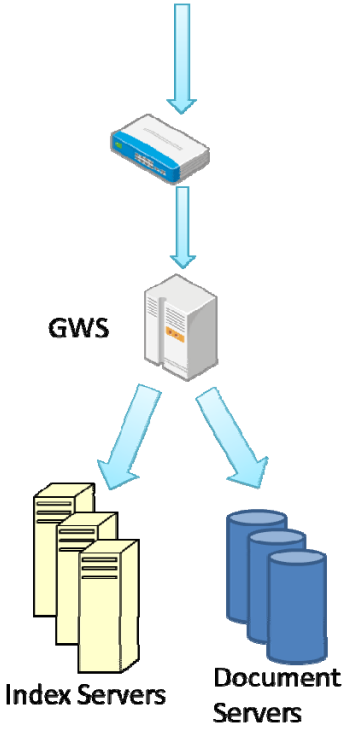
A DNS-based load-balancing system selects a cluster by accounting for the user's geographic proximity to each physical cluster

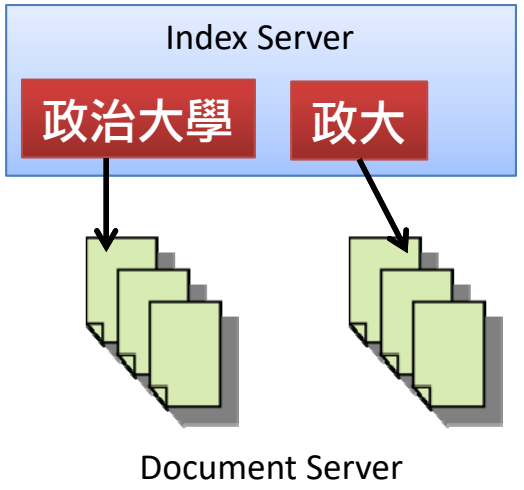
A hardware-based load balancer in each cluster monitors the available set of Google Web servers (GWSs) and performs load balancing



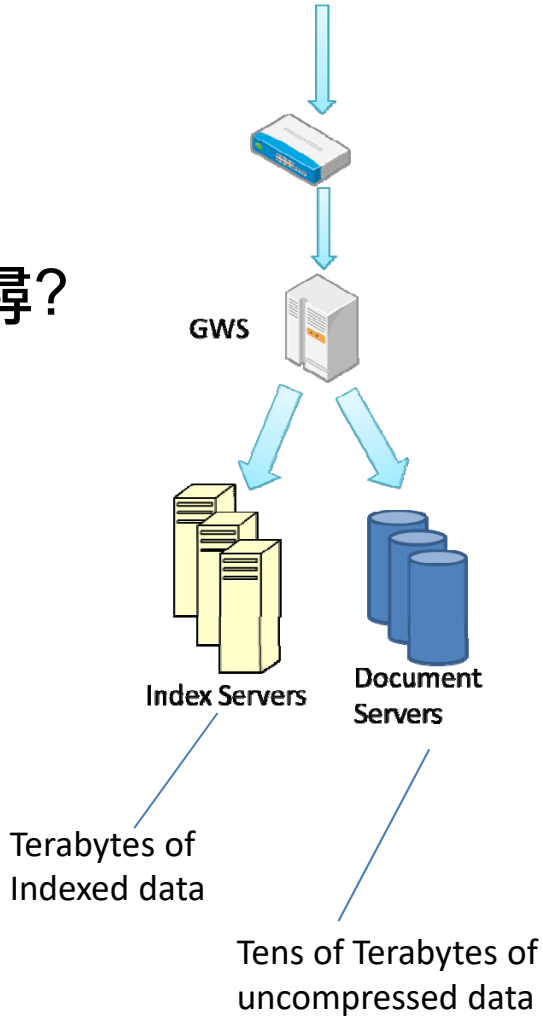
Searching and Ranking

- Index servers consult an inverted index that maps each query word to a matching list of documents (the hit list)
- The index servers then determine a set of relevant documents
  - Intersect the hit lists of the individual query words
  - Compute a relevance score for each document (PageRank)
    - The score determines the order of results on the output page

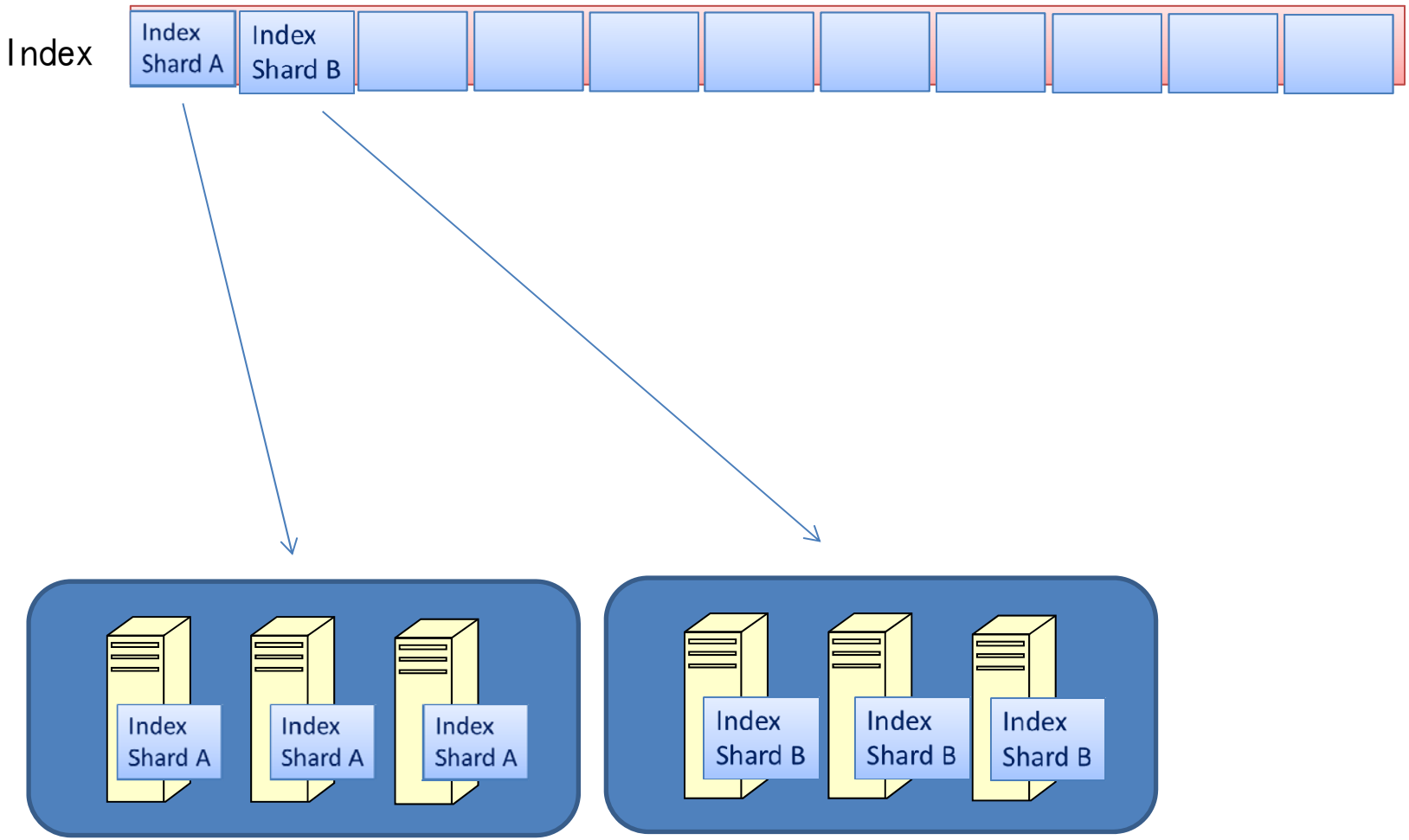


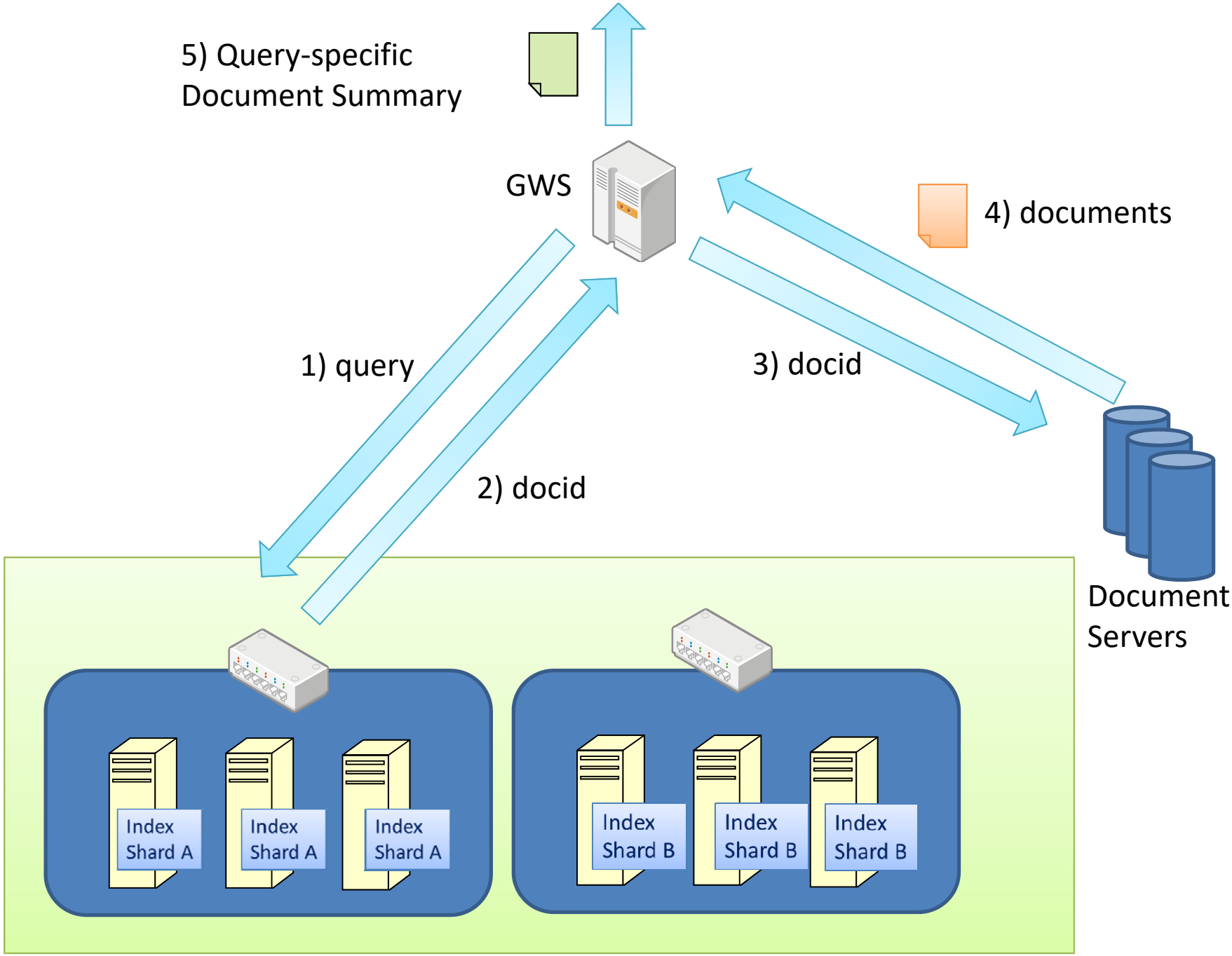


- Index本身也是Tera等級，如何有效率搜尋?
- Index shards
  - a randomly chosen subset of full index
  - A pool of machines serves requests for each shard

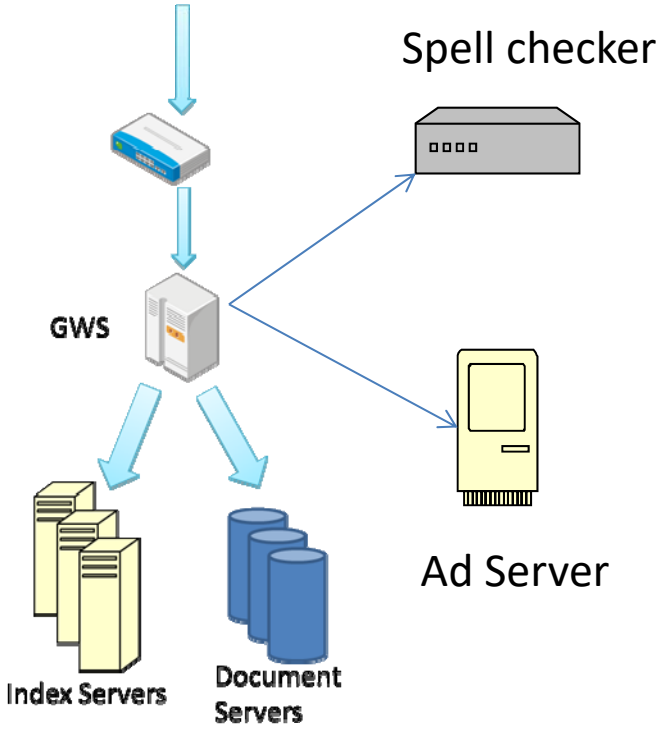


Index Shards




5) Query-specific
Document Summary
GWS
4) documents
1) query
3) docid
2) docid
Document
Servers
Index
Shard A
Index
Shard A
Index
Shard A
Index
Shard B
Index
Shard B
Index
Shard B

- Doc server
  - Online, low-latency copy of the entire Web
  - Google stores dozens of copies of the Web across its clusters
- Ancillary service
  - GWS also initiates several other ancillary tasks
    - Spell-checking
    - Ad-serving system
- Result generation
  - GWS generates the output page and returns it to the user's browser



Google's Key Patent on Cloud Computing

US2008/0262828 "Encoding and Adaptive Scalable Accessing of Distributed Models"

> "Systems, methods, and apparatus for accessing distributed models in automated machine processing, including using large language models in machine translation, speech recognition and other applications."

- Filed in February 2006
- 91 claims

System Example

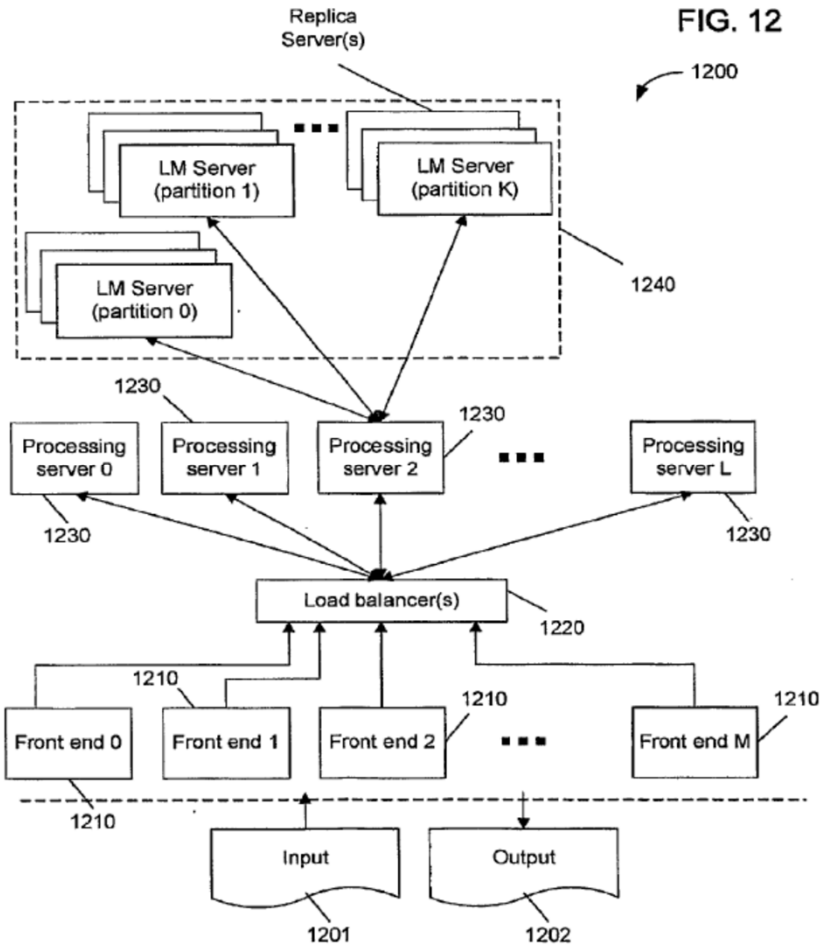


FIG. 12 An example of a distributed processing system that can be configured to provide a language processing function based on a large language model

Machine processing using machines such as computers to perform processing tasks such as machine translation

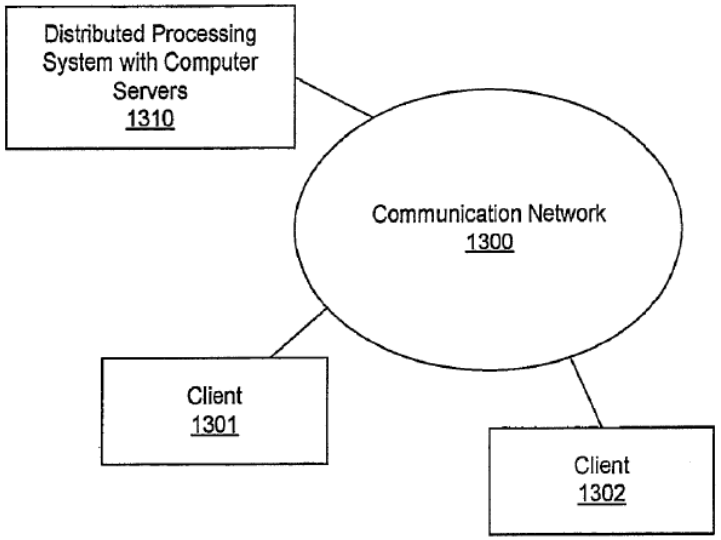
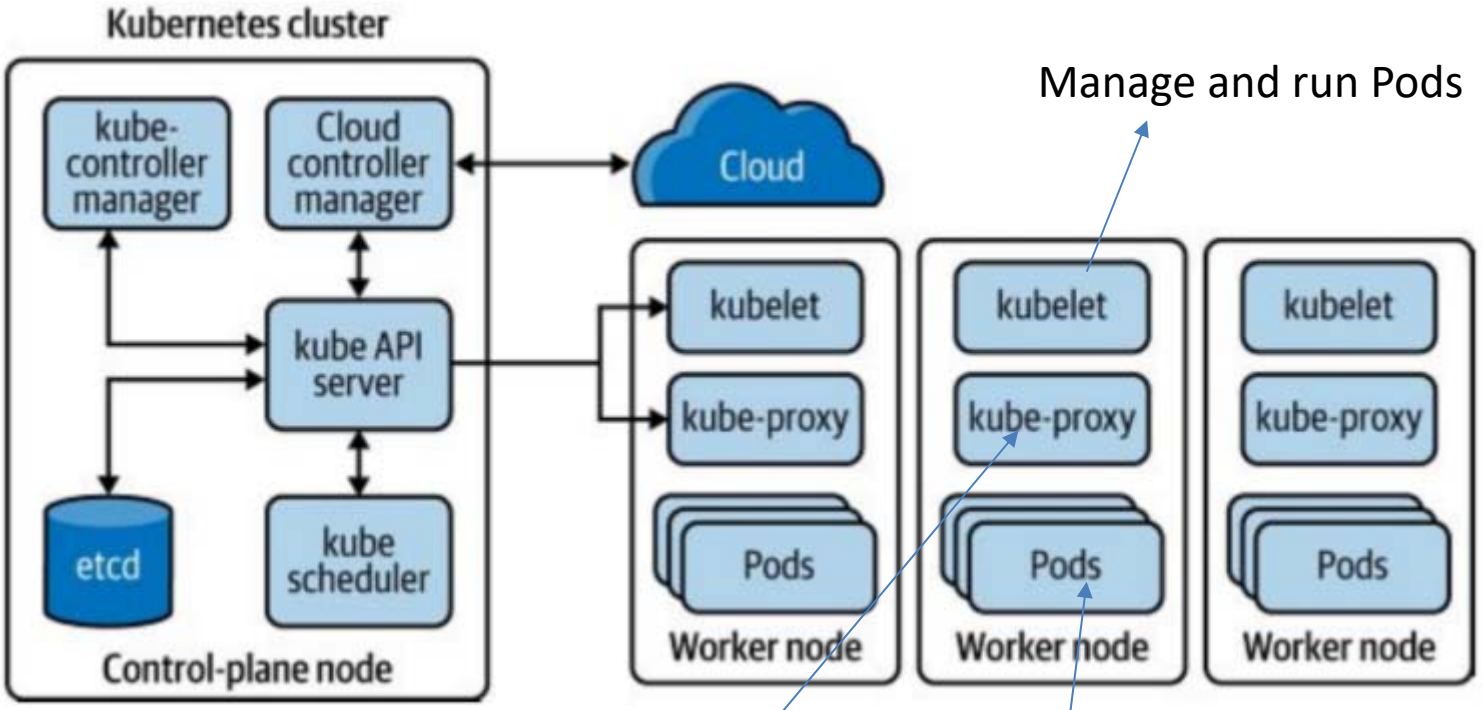


FIG. 13 An example computer system in a communication network that provides distributed processing

How to Manage Services in Data Center?

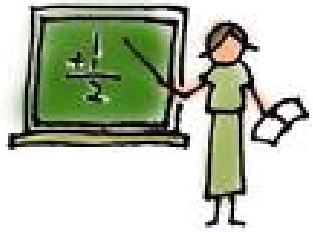
- Virtualization
  - “Hardware as Software”
- Containerization (ex: Docker)
  - Packaging system as OCI-compatible images
- Container Orchestration (ex: K8S)
  - Platform that runs OCI-compatible images
- Massive configuration management
  - Infrastructure as Code (IaC)

Kubernetes: a high-level view



How this course is going to be conducted

- Lecture
  - Slides and blackboard drawing
  - For basic knowledge: midterm 25%



- Lab (25%)
  - Short report and presentation
  - Hands-on assignment
  - All assignment have to be finished in-course (and in Time)
- Group project (40%)
  - Introducing new distributed system technologies
  - Developing and evaluate distributed applications
- Bring your laptop, notebook, and pen !

Syllabus

Course Evaluation

- Participation 10%
- In-course individual/group assignment 25%
- 期中考 25%
- 學期專案 40%
  - 期中書面15%: 小組技術報告
    - 分散式技術解析與論述
  - 期末口頭25%: 分散式技術應用開發與展示

找開源的專案

照要領去做，就能有很棒的分數

做一個專案並展示

期中報告: 分散式技術解析與論述

理解與描述
不是第一行看到最後一行
了解精髓而非瑣碎之事物

- 目的
  - 善用學期間學得的分散式系統知識，了解最新業界進展
- 條件:
  - 建議專案 (列表在下頁)
  - 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

Apache 的品質優良

圖一

- 期末展示要包含所選定的專案技術

建議專案

lines of code

- 690183 • Apache Kafka https://kafka.apache.org/
- 2274099 • Apache Pulsar https://pulsar.apache.org/
- 9652 • Aedes https://github.com/moscajs/aedes
- 2052881 • Apache Camel https://camel.apache.org/
- 1728182 • Apache Ignite https://ignite.apache.org/
- 1438245 • Apache Mina https://mina.apache.org/
- 53793 • Apache Vysper https://mina.apache.org/vysper-project/
- 187298 • Apache Zookeeper https://zookeeper.apache.org/
- 140491 • Apache OpenMeetings https://openmeetings.apache.org/
- 410944 • Apache Helix https://helix.apache.org/
- 1328808 • Apache Spark https://spark.apache.org/
- 845375 • Apache TomEE https://tomee.apache.org/

Aedes(Mosca) https://github.com/moscajs/aedes
簡易物聯網工具

1. 是 node.js 的 MQTT 代理
2. MQTT協定的訊息內容很精簡，用於處理器資源及網路頻寬有限的物聯網裝置
3. 可以快速啟動物聯網的訊息佇列(訊息會存放在佇列中，不會立即得到結果直至被取用)，能做到簡易的測試家庭自動化
4. 支援 Node-RED

建議專案

lines of code	
781564	• Spring Integration https://spring.io/projects/spring-integration
273729	• Spring Batch https://spring.io/projects/spring-batch
53207	• CoreDNS https://coredns.io/
208430	• Etcd https://etcd.io/
15687	• OpenTelemetry https://opentelemetry.io/
699953	• Envoy https://www.envoyproxy.io/
281992	• Prometheus https://prometheus.io/

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:
    - 在什麼場何用來解決什麼問題?
    - 沒有它又如何?

最好能有清楚的例子

- 架構解析
  - Explain how this technology works

如何運作

    - 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

安裝 Server 的過程要記錄，
可放報告內(1, 2頁)

圖二

- 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

期末Demo

- 目的
  - 就期中小組所報告的技術，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個位於不同機器的節點　必須為分散式
    - 可以同一台機器，三個Processes，但只透過網路溝通
- 要求
  - 非功能性: 實作至少一項維運功能 (系統偵測、自動回復、隨需伸縮…)
  - 功能性: 要具有意義的應用 (不能是hello word)
- 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

圖三

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
  - 2/28 繳交分組名單 (一組6-8人)
  - 4/18 Midterm
  - 4/25 期中報告due
  - 6/13 期末demo
  - 6/20 期末demo原始碼/slide/影片 上傳期限

圖四

期末不用另外寫報告
僅PPT，原始碼，Github，影片 Demo

Q & A