Midterm Report & Final Project for Cloud Computing & Networking

Reporter : Ting-Wei Ou
Advisor : Hsueh-Wen Tseng

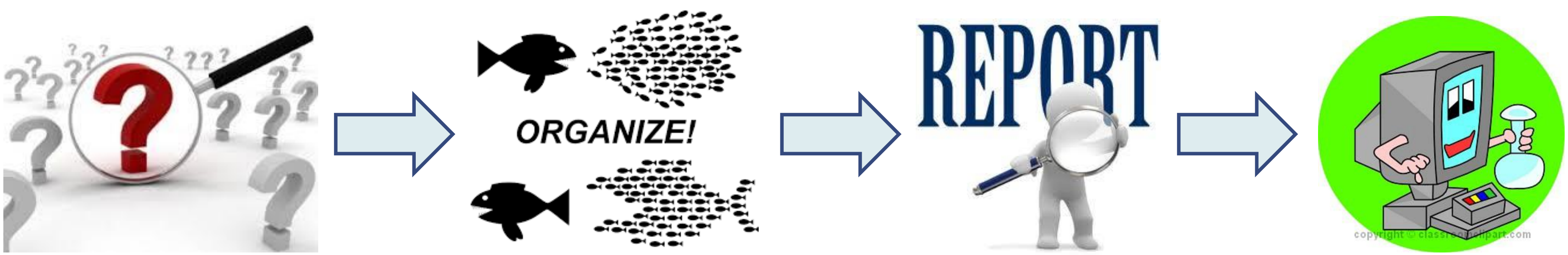
Outline

- We Can Learn ?
- How to Survey & Start ?
  - Example Topic
- Presentation
  - Midterm Report
    - Scheduling for Scalable Management in Cloud Environments
  - Final Project
    - Scalable Scheduling Services in Data Center Networks

We Can Learn ?

報告內容與期末專題相關
盡可能和自己研究主題相符

1. How to find a research topic
   - Cloud Computing & Networking
2. How to reorganize these papers
3. How to report it
4. How to design your experiment (實作)



How to Survey (ACM/IEEE)

- **References**
  - ACM Conference
    - ACM SIGCOMM
    - ACM MobiCom
    - ACM MobiHoc
    - ACM MobiSys
  - IEEE Conference
    - IEEE Infocom
    - IEEE ICC
    - IEEE WCNC
    - IEEE Globecom
  - NSDI
    - USENIX Symposium on Networked Systems Design and Implementation
  - IEEE & ACM 的 transaction on XXX 的 Journal paper

實作與報告的相關議題要盡量符合

出處!!!

~~Workshop~~

近三年文獻

Schedule first, manage later: Network-aware load balancing

Nahir, A.; Orda, A.; Raz, D.

INFOCOM, 2013 Proceedings IEEE

Year: 2013

Pages: 510 - 514, DOI: 10.1109/INFCOM.2013.6566825

IEEE Conference Publications

▸ Abstract ((html)) PDF (106 Kb) ©

ACM DL DIGITAL LIBRARY

IEEE Xplore® Digital Library

How to Know Conference Ranking

Title	Acronym	Source	Rank
IEEE International Conference on Computer Communications	IEEE INFOCOM	CORE2014	A*

- http://103.1.187.206/core/
- http://academic.research.microsoft.com/?SearchDomain=2&entitytype=2
- http://www.cs.ucsb.edu/~almeroth/conf/stats/
- https://en.wikipedia.org/wiki/List_of_computer_science_conferences

Computer Networking and Networked Systems [edit]

See also: § Concurrent, distributed and parallel computing

Conferences on computer networking:

- SIGCOMM - ACM SIGCOMM Conference [2][40][105][106][107]
- NSDI - USENIX Symposium on Networked Systems Design and Implementation [108][109]
- SIGMETRICS - ACM SIGMETRICS Conference [2][4][105][110][111]

  a.k.a. "ACM Conference on Measurement and Modeling of Computer Systems"
- CoNEXT - ACM Conference on emerging Networking EXperiments and Technologies [2]
- INFOCOM - IEEE Conference on Computer Communications [2][105][112][113][114]
- IMC - USENIX/ACM Internet Measurement Conference [2]
- ICNP - IEEE International Conference on Network Protocols [2][115][116]
- IPTPS - International Workshop on Peer-To-Peer Systems [117]
- ICSOC - International Conference on Service Oriented Computing [118]
- IWQoS - IEEE International Workshop on Quality of Service [119]

電腦網路相關的 Conference

How to Start (presentation)

- The content (最少三篇) for midterm report
  - Time : 2/3
  - About 30 pages : Outline + content + conclusion
    - The word format
      - Time New Roman (推薦)
        - Title: about 40 size
        - The content: about 20~28 size
      - Page number & reporter name
  - References
    - Authors / Institution (作者、題目、機構、年份 …)

文法 !!!

參考文獻

How to Start

- Topic choice
  - Different topics → Hard to present
  - The same topics → More easier ✓

相同的主題，較易組織與連結。

- Example Topic
  - TCP Incast
  - Energy Optimization in Data Center Network
  - Scalable multicast
  - My Current Research
    - Disaster Recovery for Distributed Storage

TCP Incast

一篇主要
兩篇參考

標示: 作者/論文題目/出處/年分
年分: 近三年內

- A Cross-Layer Flow Schedules with Dynamical Grouping for Avoiding TCP Incast Problem in Data Center Networks
  - M. Alizadeh, A. Greenberg, D.-A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," Proc. ACM SIGCOMM, pp. 63-74, Oct. 2010.
  - B. Vamanan, J. Hasan, and T.N. Vijaykumar, "Deadline-aware datacenter tcp (D2TCP)," Proc. ACM SIGCOMM, pp. 115-126, Aug. 2012.

Energy Optimization in Data Center Network

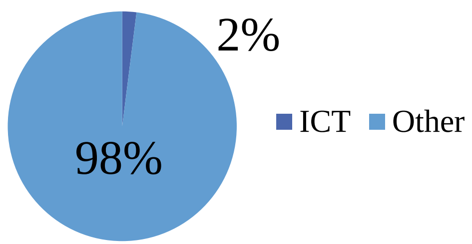
- **Energy optimizations for data center network: Formulation and its solution**
  - Shuo Fang ; Hui Li ; Chuan Heng Foh ; Yonggang Wen ;Khin Mi Mi Aung
  - Global Communications Conference (GLOBECOM), 2012 IEEE
- **Limits of energy saving for the allocation of data center resources to networked applications**
  - Leon, X. ; Navarro, L.
  - INFOCOM, 2011 Proceedings IEEE
- **HERO：Hierarchical energy optimization for data center networks**
  - Yan Zhang; Ansari, N.
  - Communications (ICC), 2012 IEEE International Conference on

Energy Optimization in Data Center Network

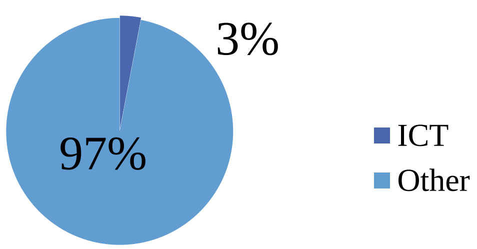
- Data centers should provide high availability and fault tolerant
  - Require high energy consumption

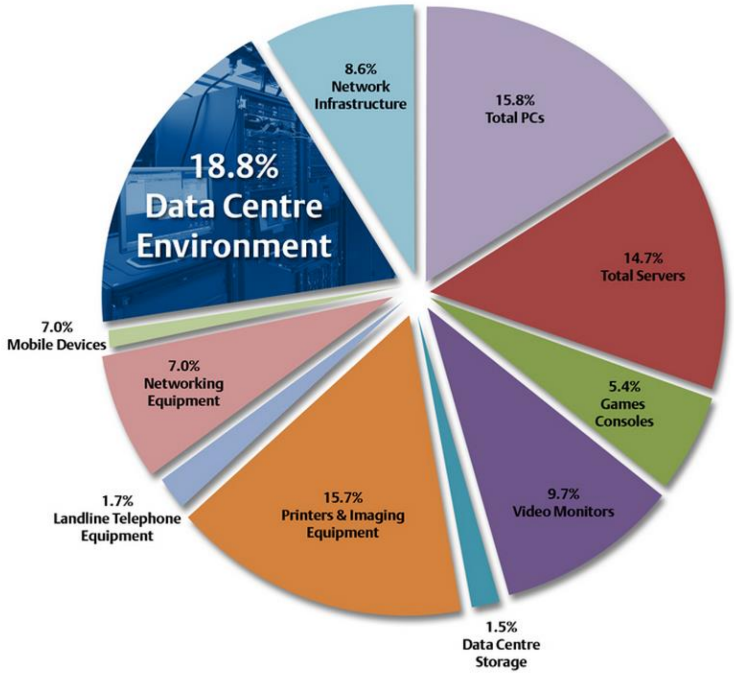
利用圖說明，能量消耗議題
在資料中心的重要性

CO2 emissions



Global energy expenditure



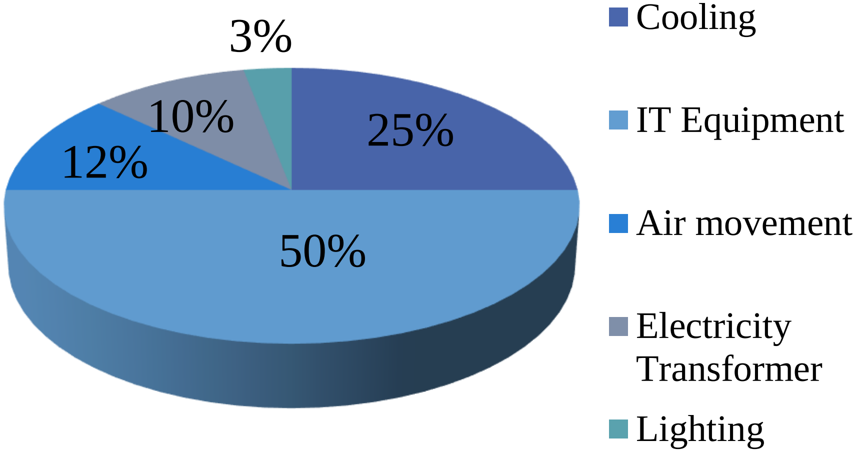


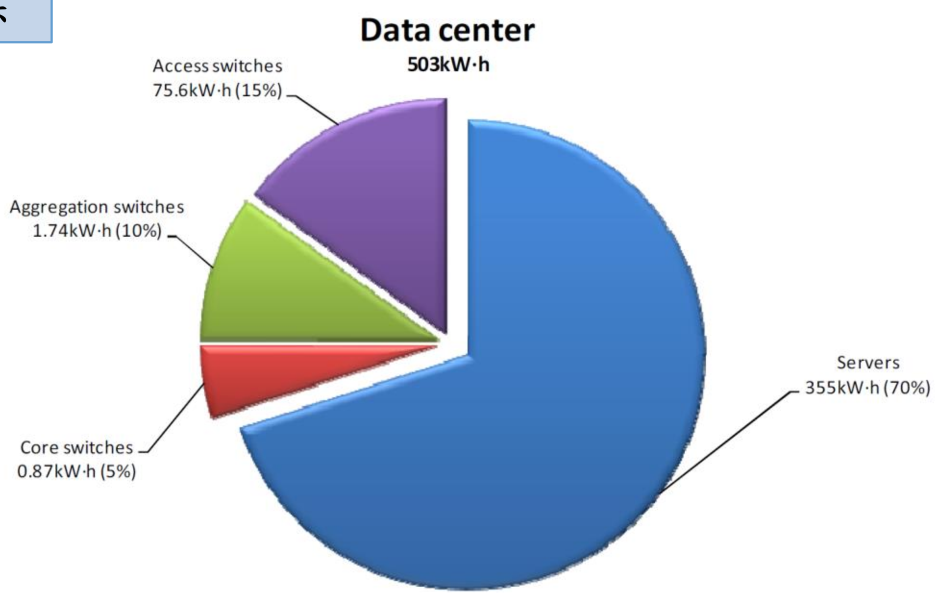
ICT Energy Consumption in Australia

Energy Optimization in Data Center Network

- Power consumption in a data center

資料中心的中的能源消耗 : server 為大宗





Nearly 30% of the total computing energy in a data center is consumed by the communication links, switching, and aggregation elements

Energy Optimization in Data Center Network

server 中的能量消耗→ CPU 為大宗

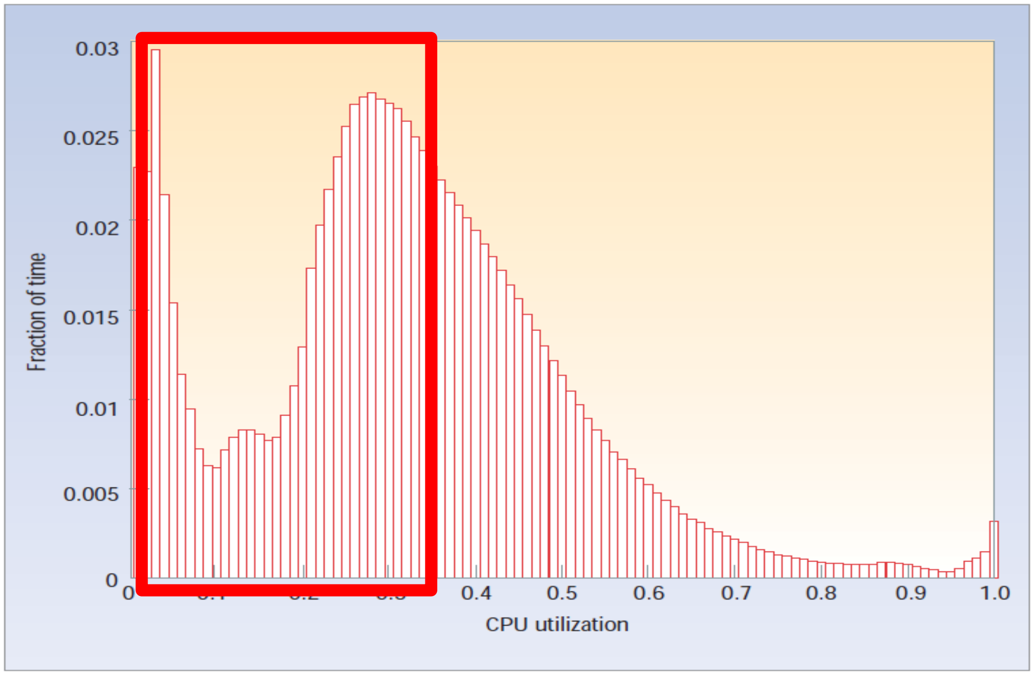
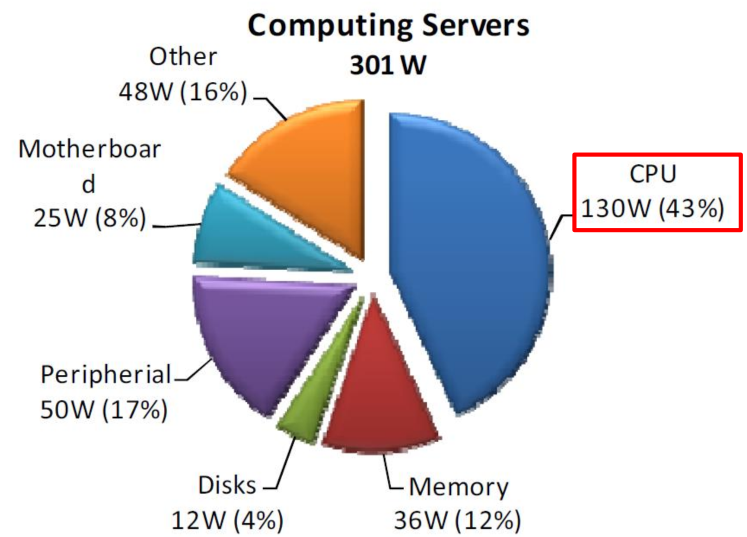


Figure 1. Average CPU utilization of more than 5,000 servers during a six-month period. Servers are rarely completely idle and seldom operate near their maximum utilization, instead operating most of the time at between 10 and 50 percent of their maximum utilization levels.



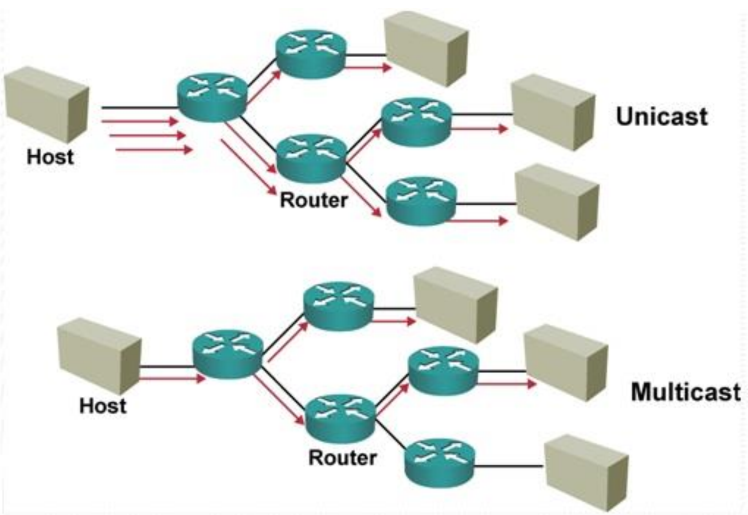
Scalable multicast

- **Exploring Efficient and Scalable Multicast Routing in Future Data Center Networks**
  - Dan Li, Jiangwei Yu, Junbiao Yu, Jianping Wu
  - INFOCOM, 2011 Proceedings IEEE
- **ESM: Efficient and Scalable Data Center Multicast Routing**
  - Dan Li, Yuanjie Li, Jianping Wu, Sen Su, Jiangwei Yu.
  - Networking, IEEE/ACM Transactions on
- **Multicast Fat-Tree Data Center Networks with Bounded Link Oversubscription**
  - Zhiyang Guo , Yuanyuan Yang
  - INFOCOM, 2013 Proceedings IEEE

Scalable multicast

- Multicast benefits data center group communication.
  - Saving network traffic.
    - Increase the throughput.
  - Reduce the task finish time of delay-sensitive applications.
    - Releasing the sender from sending multiple copies of packets to different receivers.

群播在資料中心群組溝通的優勢



Scalable multicast

- Explore network-level Multicast routing, which is responsible for building the multicast delivery tree, in future data center networks.
- Bandwidth-hungry, large-scale data center applications call for efficient and scalable Multicast routing schemes.

群播在資料中心中的優勢

Disaster Recovery for Distributed Storage

資料中心中資訊服務的趨勢：資料是重要資產

- Information services in Data Centers [1][2]
  - The explosion of Big Data
    - EMC and IDC : 44 ZB by 2020 [3]
      - A 50-fold for the beginning of 2010
    - Alibaba processes more than 50TB of data per day. [4]
      - Store more than 40PB each day
    - Storage as a Service → applications
      - e.g. Instagram, Flickr
  - Data has become the critical asset
    - High availability & reliability

ZB (zettabytes) = 10^{12} GB

[1] H. Yu; X. Xiang; Y. Zhao; W. Zheng, "BIRDS: A Bare-Metal Recovery System for Instant Restoration of Data Services," Computers, IEEE Transactions on, vol.63, no.6, pp.1392,1407, June 2014
[2] Z. Wu, M. Butkiewicz, D. Perkins, E. Katz-Bassett, and H. V. Madhyastha, "Spanstore: Cost-effective geo-replicated storage spanning multiple cloud services," in Proc. 24th ACM Symp. Operating Syst. Principles, 2013, pp. 292–308.
[3] http://www.emc.com/leadership/digital-universe/2014iview/executive-summary.htm
[4] http://www.ithome.com.tw/news/92243

Disaster Recovery for Distributed Storage

資料服務需要有高可用性及可靠性的需求

- High available & reliable data services [1]
  - Failures lead to service disruptions
  - IDC report : the average cost of downtime (disruption)
    - The Fortune 1000 : $1 million dollars per hour [5]
  - Disaster recovery (DR)
    - Protect data & resume data services quickly
    - With Big Data
      - Real-time data analytics
      - Data protection & reduce downtime

[1] Hongliang Yu; Xiaojia Xiang; Ying Zhao; Weimin Zheng, "BIRDS: A Bare-Metal Recovery System for Instant Restoration of Data Services," Computers, IEEE Transactions on , vol.63, no.6, pp.1392,1407, June 2014
[5] http://devops.com/2015/02/11/real-cost-downtime/

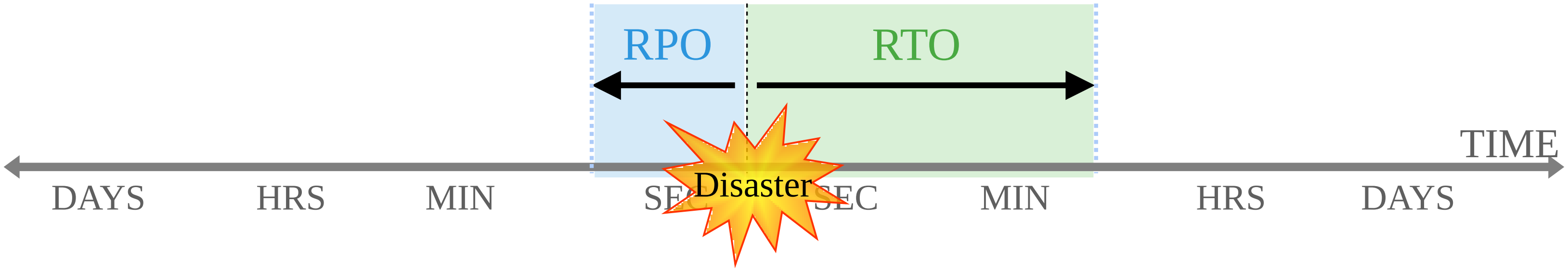
Disaster Recovery for Distributed Storage

- Disaster recovery in data centers
  - Virtualization are more and more ubiquitous
    - e.g. Xen, VMware vSphere, KVM, OpenStack
    - Server virtualization
      - One physical server runs multiple OS concurrently
        - Infrastructure utilization
    - Storage virtualization
      - Consolidation of physical storage
        - Reduced management overhead
        - Data : high availability

Disaster Recovery for Distributed Storage

Disaster Recovery 四大指標

- Disaster recovery techniques for virtualization
  - Four important goals
    - General-purpose (一般性)
    - Low overhead (低功耗)
    - Recovery point objective (RPO, less data loss)
    - Recovery time objective (RTO, less downtime)



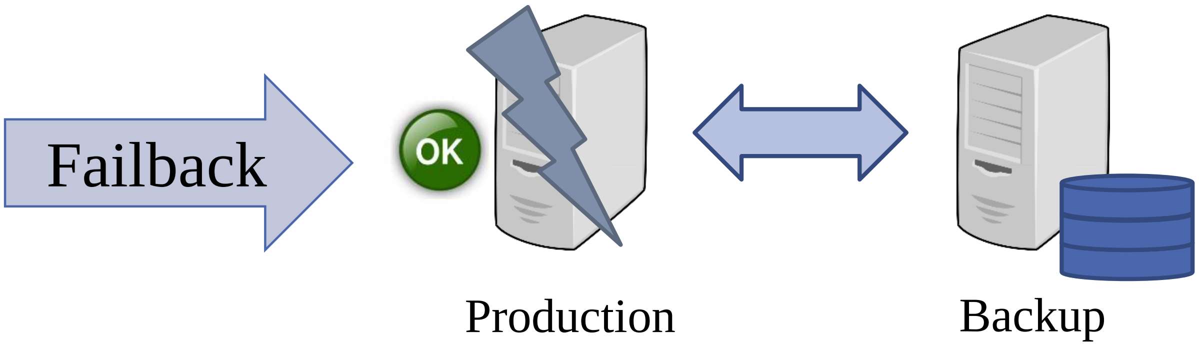
Disaster Recovery for Distributed Storage

Disaster Recovery : RTO 與 RPO 和 Cost 間的矛盾 (應以需求考量)

	Recovery Time Objective	Recovery Point Objective	What should I use it for?
Small	>1 Day to Week(s)	> 1 Day to Week(s)	Development and Testing Systems
Medium	> 4 Hours to Day(s)	> 4 Hours to Day(s)	Workgroup Applications
Large	Minutes to Hour(s)	Minutes to Hour(s)	Infrastructure Systems and Messaging Systems
“Biggie” S	ediate	Real-Time	Business-Critical Systems

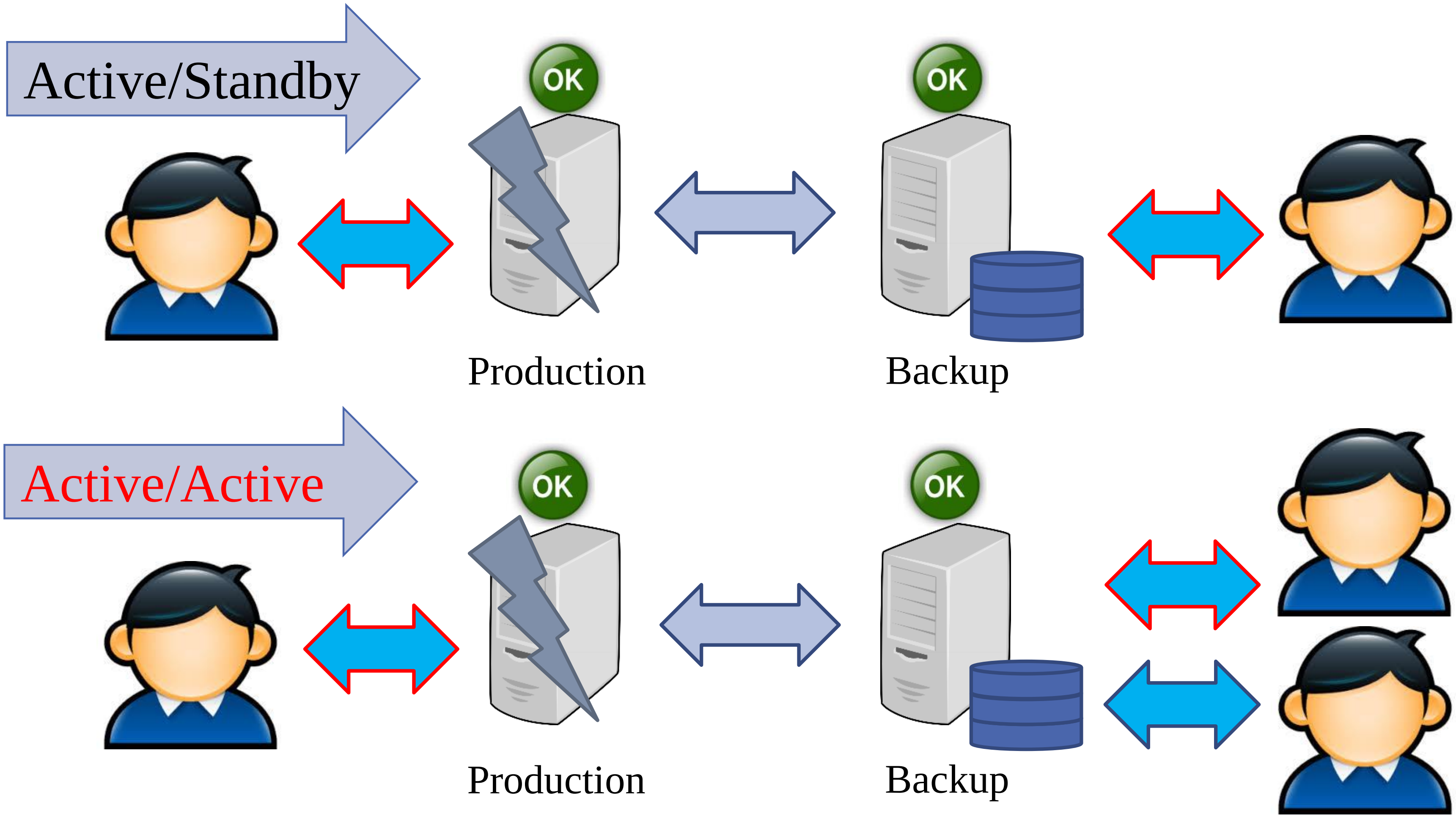
Disaster Recovery for Distributed Storage

- DR approaches (backup)
  - Failback
    - Restore services in the production site by remote backups
  - Failover – active/standby & active/active
    - The production (primary)/ backup (secondary)
      - The secondary will take over after the primary fails



Disaster Recovery for Distributed Storage

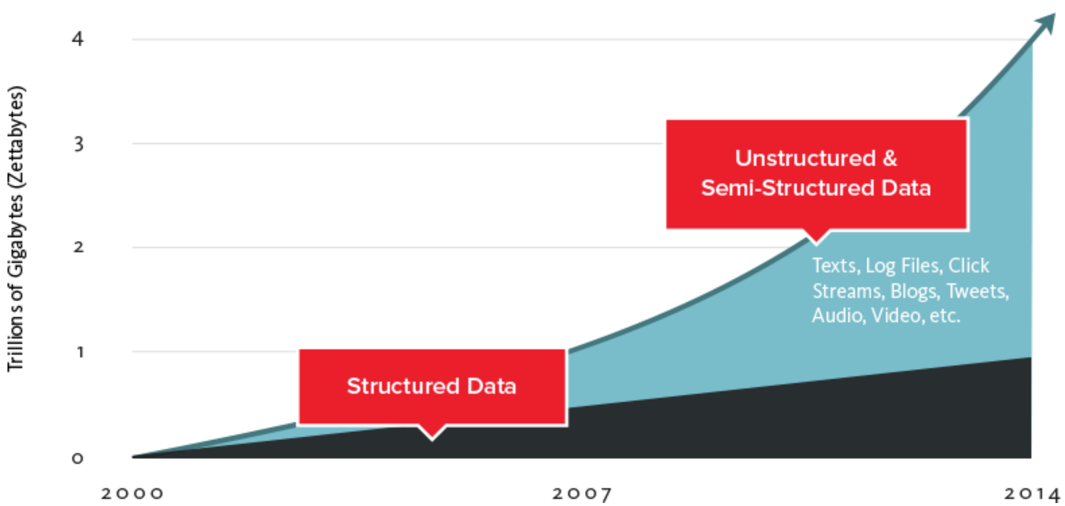
- Failover mode



Disaster Recovery for Distributed Storage

Disaster Recovery 在 Big Data 下的需求

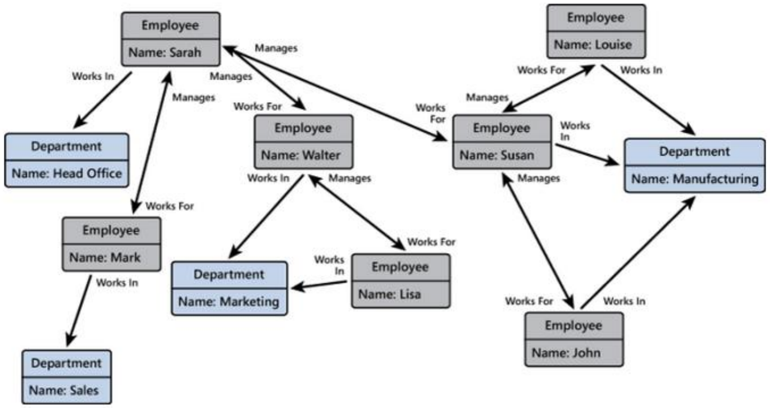
- The impact of Big Data (volume, velocity, variety)
  - Variety : storage for database systems [6]
    - 70 ~ 80% unstructured & semi-structured data
      - Traditional relational databases cannot meet the challenges on categories and scales.
      - NoSQL databases are becoming more popular for big data.



[6] Chen, Min, et al. "Big data storage." *Big Data*. Springer International Publishing, 2014. 33-49.
http://www.couchbase.com/nosql-resources/what-is-no-sql

Disaster Recovery for Distributed Storage

- NoSQL databases [7] ("Not Only SQL")
  - Massive data storage across distributed servers
  - Advantages
    - Easier deployment & Large-scale data applications
    - Horizontal scaling/Scale-out
  - Categories (features) for storage
    - Key-value store (e.g. Redis, RAMCloud)
    - Column family stores (e.g. Cassandra)
    - Document stores (e.g. MongoDB)
    - Graph database (e.g. Pregel)



https://chtseng.wordpress.com/2014/05/19/nosql%E8%88%87mongodb/

[7] Chandra, Deka Ganesh. "BASE analysis of NoSQL database." *Future*

Disaster Recovery for Distributed Storage

點出目標 : DR 在 Big Data 的需求

- DR (disaster recovery) with Big Data
  - For cloud providers
    - Storage : RDBMS → NoSQL databases
      - Key-value store
      - Real-time, low-latency, and scalability
    - Cost (backup operation) & recovery performance (RTO)
      - Active/active mode option
      - Maintain data high availability
        - Data protection
        - Smaller RTO (recovery time objective)

Midterm Report –
Scheduling for Scalable Management in Cloud Environments

主題：在雲端環境下可擴充式的排程

REPORTER : 歐庭瑋
ADVISOR : 曾學文

Outline

- Introduction
  - Scalability
  - Auto-scaling
- Scalable management
  - Web services
  - Internet applications
  - Distributed scheduling
- Conclusion & Comparison

介紹三篇共通環境與主題

簡介三篇方法實驗部分
重點考量環境

統整特性差異

Reference

參考文獻依年份順序整理，叫好發揮

題目
作者
出處

- **Web scaling frameworks: A novel class of frameworks for scalable web services in cloud environments**
  - Fankhauser, T.; Qi Wang; Gerlicher, A.; Grecos, C.; Xinheng Wang
  - Communications (ICC), 2014 IEEE International Conference on
- **Automatic Scaling of Internet Applications for Cloud Computing Services**
  - Zhen Xiao; Qi Chen; Haipeng Luo
  - Computers, 2014 IEEE Transactions on
- **Schedule first, manage later: Network-aware load balancing**
  - Nahir, A.; Orda, A.; Raz, A.
  - INFOCOM, 2013 Proceedings IEEE

Introduction

- What is Scalability?
  - Scalability is a term used to describe how the application will handle increased loads of traffic volume.
  - Scalability requirements
    - Increase in performance
    - Operational efficiency
    - Resource elasticity

主題介紹

需求

把重點用紅字標示

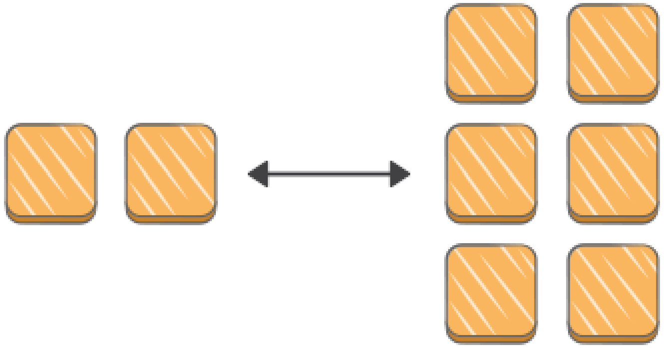
Introduction

- Resource elasticity
  - Utilization
  - On demand
  - Scale up and down
  - **Costs** should not scale – **Auto-scaling**
    - VMs or Applications (Instances)

資源彈性：可以依需求調整

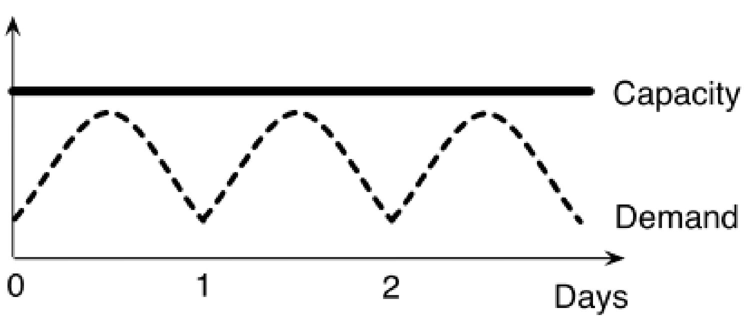
點出主題

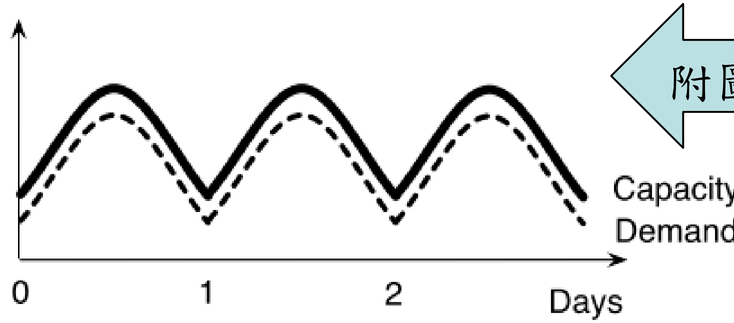
目的



Introduction

- Auto-scaling 主題介紹
  - It builds upon the idea of load balancing
  - Resource usage can be scaled up & down automatically.
    - Maintain your instance availability
    - On demand
    - It provides flexible resource allocation with cost savings.
  - The users are charged only for what they actually use.
    - “pay as you go”





附圖說明

Introduction

範例介紹

- Auto-scaling practices
  - Amazon Web Services (AWS)
  - Microsoft's Windows Azure
  - Google Cloud Platform (GAE)
  - Facebook
  - OpenStack

Google Cloud Platform Live

openstack™ CLOUD SOFTWARE

一個在雲端環境對於可擴充的網頁服務 framework

33

Web scaling frameworks: A novel class of frameworks for scalable web services in cloud environments

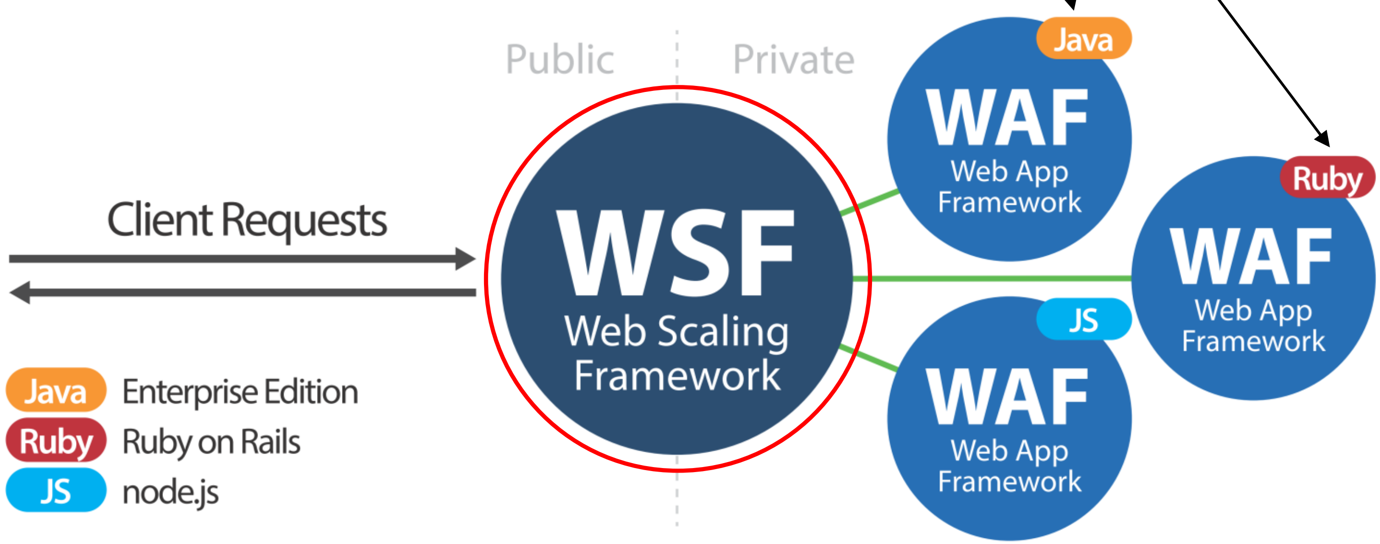
Fankhauser, T.; Qi Wang; Gerlicher, A.; Grecos, C.; Xinheng Wang
Communications (ICC), 2014 IEEE International Conference on

報告的論文題目
作者、出處、年份

Web Services

- The Web Scaling Framework (WSF)
  - Web Application Frameworks (WAFs) do not offer
    - Automatic resource-provisioning
    - Elastic caching
    - Guaranteed maximum response times

前情題要



The relationship between the WSF and WAFs

Web Services

- The capability of WSF
  - WAFs – incorporation & connection
    - Separate the service logic
  - Horizontal scaling – instant ability
    - Adapt infrastructure to fit the required performance
    - Use SaaS or machine-cluster components transparently (透明)
  - Pay-per-use

主軸、目的

Web Services

方法概念簡述

- Architecture, Cache, Data Store
  - Auto-Scaling
    - Job deadlines
    - Resource utilization

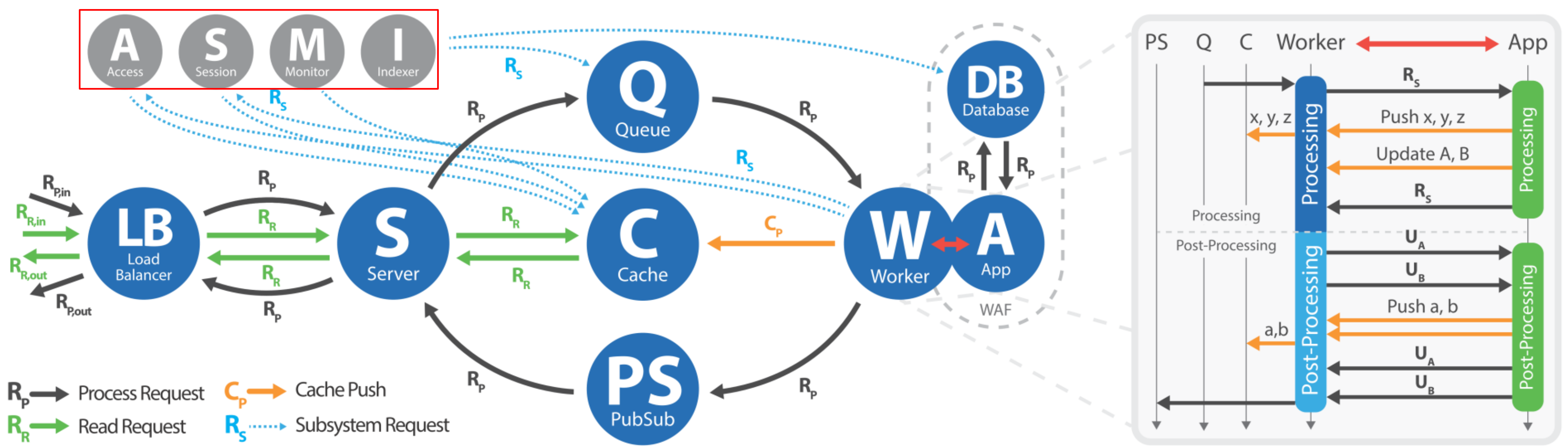
考量層面及目標

- A scalable and predictable system
  - A mathematical model
    - Relieve (延緩) WAF from scaling → COST !!!
    - Utilize both IaaS & SaaS as components

要處理與解決的部分

Request flow – Scaled WSF

- Ratio between R_P (Processing) & R_R (Read requests)



The flow of requests through the proposed prototype with a detail view of the processing & synchronous post-processing

Web Services

- Request flow (V_N vs. V_S)

The normal app version : V_N
The scaled app version : V_S

Model 分析

- V_N
  - $C_N = \{ LB, A \}$ The number of machines used for a single component c_x
  - $RFPS_N = \min_{x \in C_N} \{RFPS_{N,x} \cdot m_x\}$
- V_S – considering *CPR*
  - $C_{S,R} = \{ LB, S, C \}$
  - $C_{S,P} = \{ LB, S, Q, W, PS \}$
  - $RFPS_S = CPR \cdot RFPS_{S,R} + (1 - CPR) \cdot RFPS_{S,P}$

Component: $c_x \in \{C_N, C_S\}$
$V_N : C_N = \{LB, APP\}$ & $V_S : C_S = \{LB, S, C, Q, W, APP, PS\}$

RFPS : Maximum Request Flow Per second
CPR : Cache/Processing Ratio

Web Services

- Linear total machines regression (V_N vs. V_S)
  - The growth of the machine demand
  - The slope (斜率) of V_N
    - $ms_N = \dfrac{\sum_{x \in C_N} \frac{RFPS_{N,max}}{RFPS_{N,x}}}{RFPS_{N,max}}$

      $RFPS_{N,max} = \max_{x \in C_N} \{RFPS_{N,x}\}$

      - Add up the number of the machines needed for all components
    - $M_{N,Reg} = \lceil ms_N + |C_N| \rceil$
  - The slope of V_S
    - $ms_S = CPR \cdot ms_{S,R} + (1 - CPR) \cdot ms_{S,P}$
    - $M_{S,Reg} = \lceil ms_S + |C_S| \rceil$

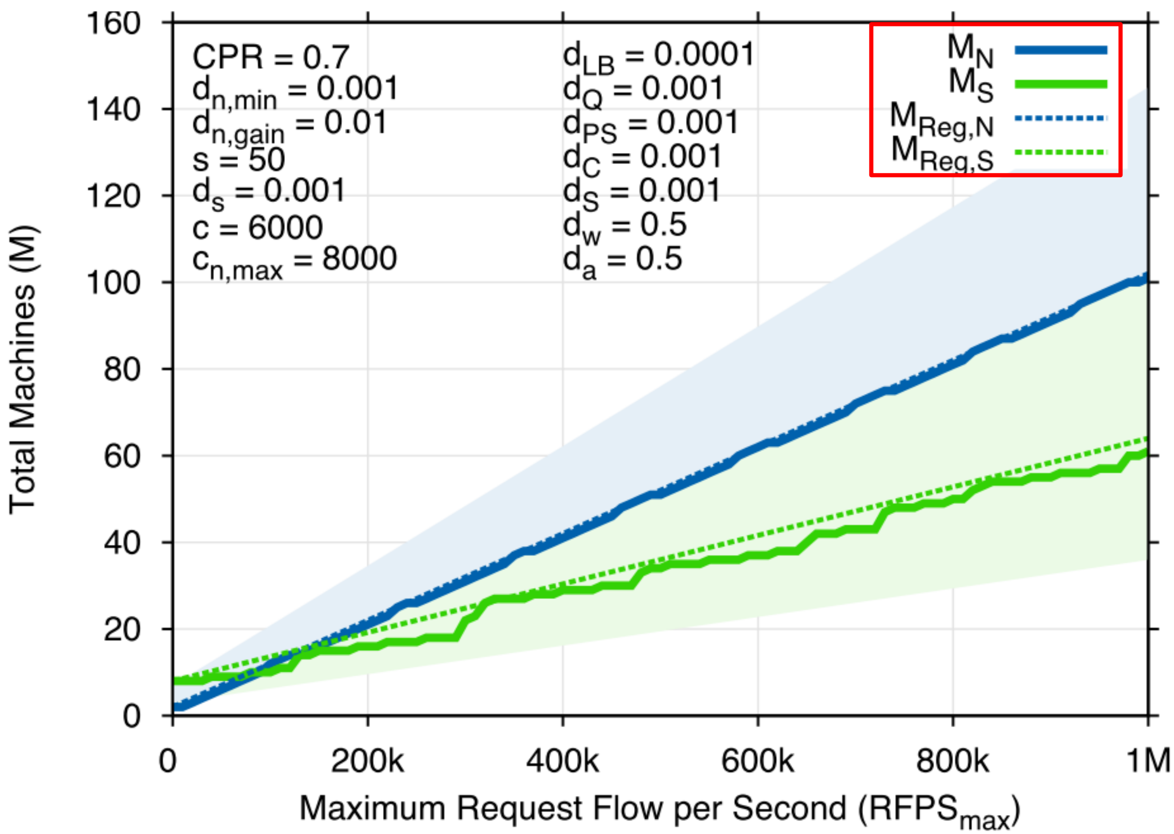
Model 分析需要幾台

$|C_N|$ or $|C_S|$: Current machine number for compenents

Web Services

- The results of V_S are compared to V_N.

模擬



Total Machine comparison (prediction and linear regression).
V_S constantly needs 44.44% fewer machines than V_N (lower is better).

實驗結果

Relative average machine reduction

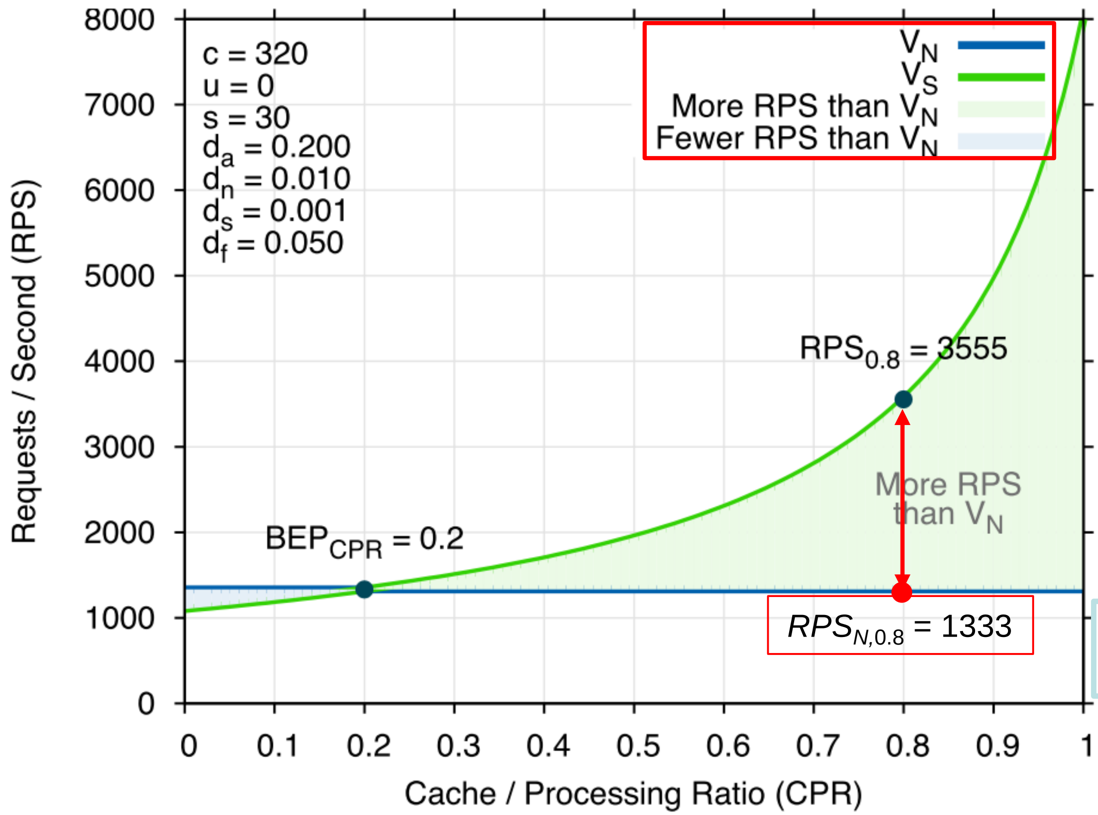
$$RAMR = (1 - \frac{ms_S}{ms_N}) \cdot 100$$

講述 X 軸與 Y 軸的關係 !!!

$RAMR = 44.44\%$

Real systems (Applications)

實作



$$\frac{3555}{1333} - 1 = 166.6\%$$

CPR dependent *RPS* development for S_2. The *CPR* where V_S starts to perform better than V_N is given by break-even $BEP_{CPR} = 0.2$. At $CPR = 0.8$, V_S generates 166.6% more *RPS* than V_N .

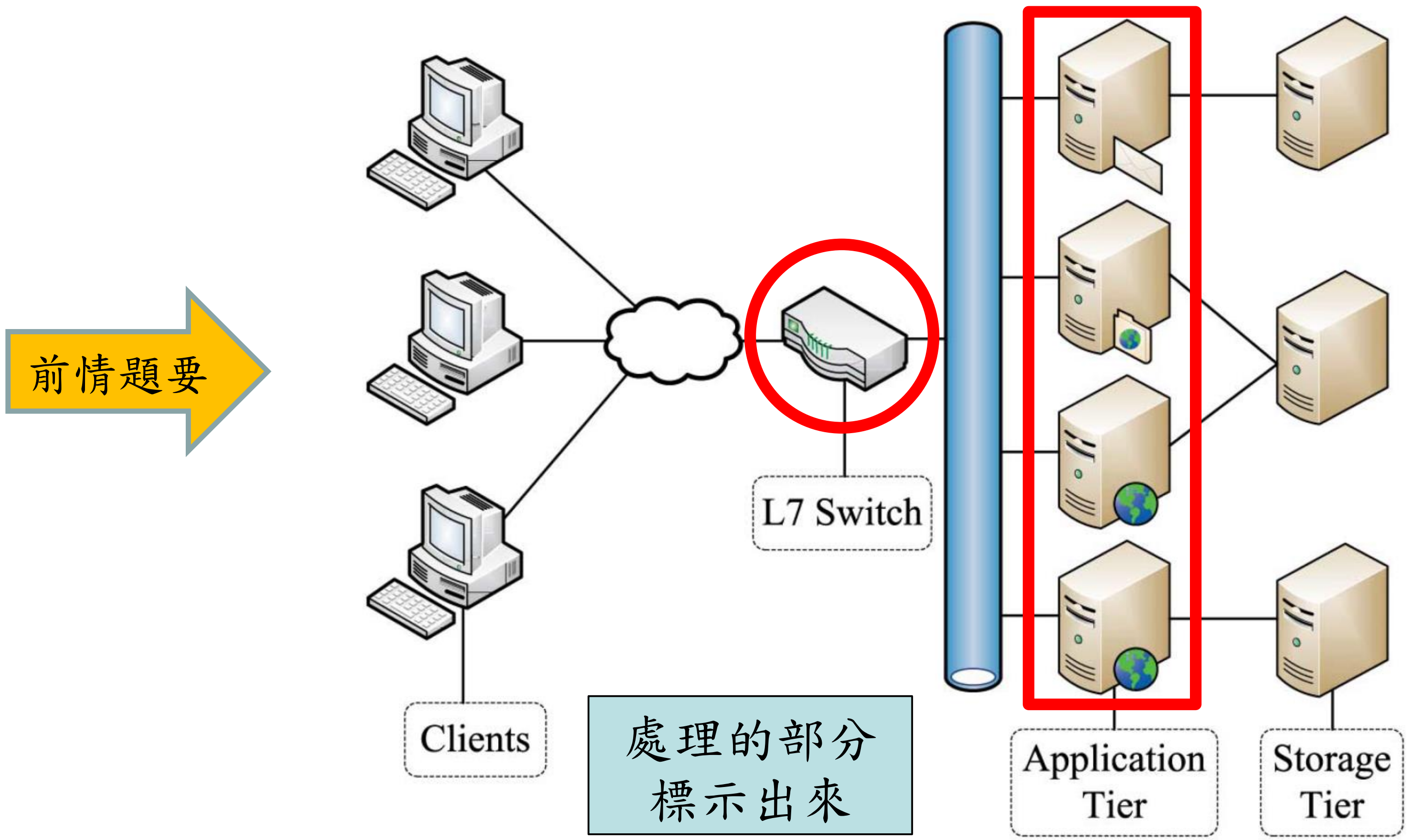
Automatic Scaling of Internet Applications for Cloud Computing Services

Zhen Xiao; Qi Chen; Haipeng Luo;

Computers, IEEE Transactions on, 2014

Internet Applications

- The architecture of data center servers (Logic)



Two-tiered architecture for Internet applications.

Internet Applications

背包問題

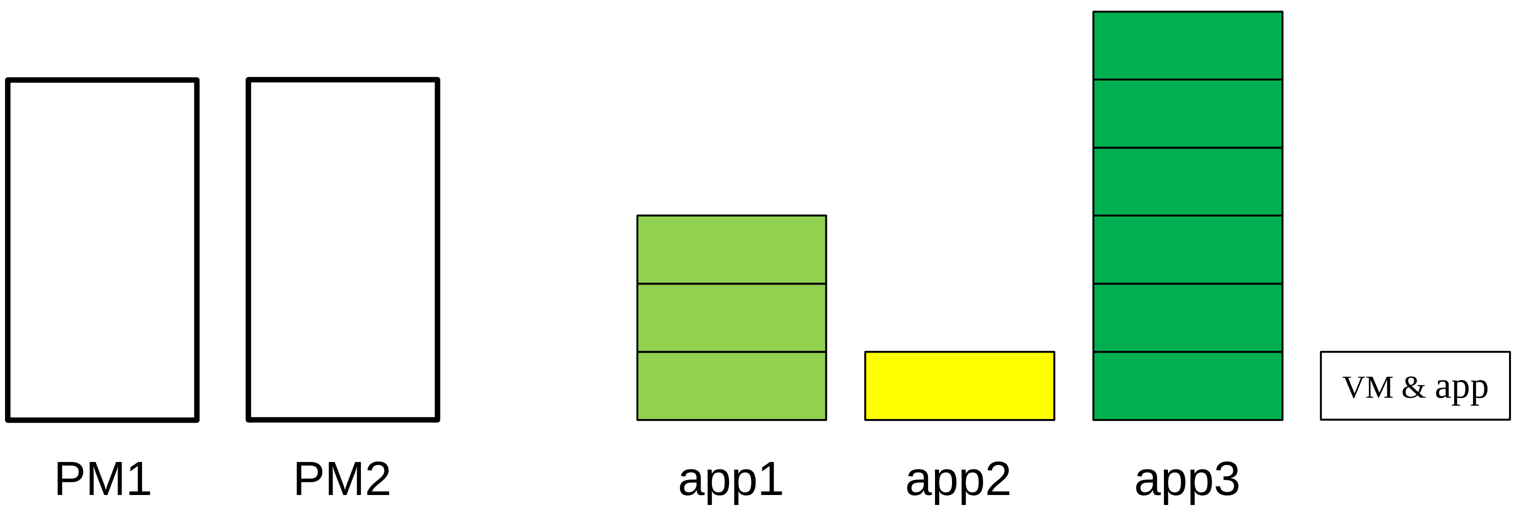
方法概念簡述

- Class Constrained Bin Packing (CCBP) problem
  - Each server is a **bin**
  - Each **class** represents an application.
    - To find a feasible placement of all the items in a minimal number of bins

5 格

v = 5
c = 2

2 類



Load unit = 20 %

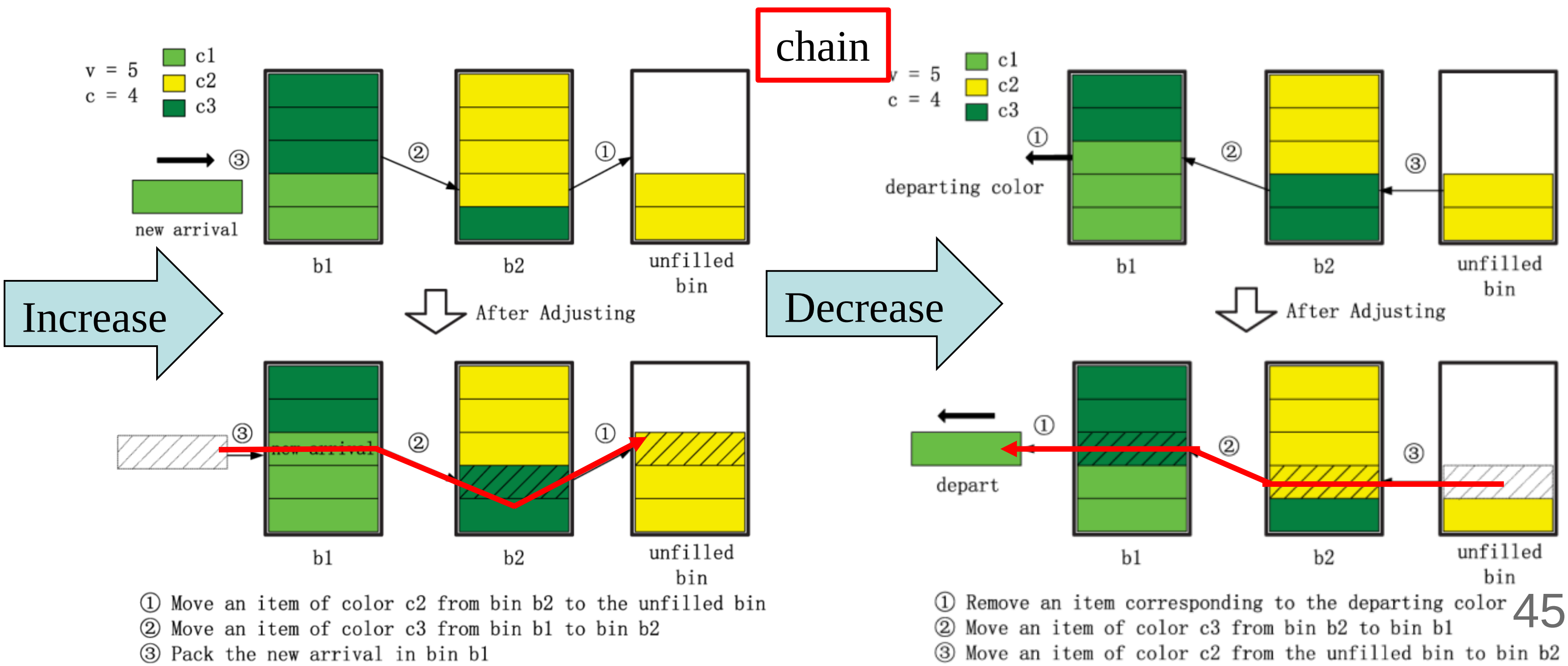
Application placement & load distribution

NP-hard

Internet Applications

- Application Load
  - Increase & Decrease
    - A bin (e.g. b1) contains **both colors**
    - Joins & Leaves – at most one unfilled color set

PM without needing add new applications



Internet Applications

Approximation Ratio

The capacity of a bin

$$R = \frac{A(\sigma)}{OPT(\sigma)} = \min(2, 1 + \frac{v - 1}{c \cdot L}).$$

The average load for each application

The class constraint

目的

- High load
  - Achieve good **satisfaction ratio**
- Low load
  - A small number of servers to satisfy all application demands

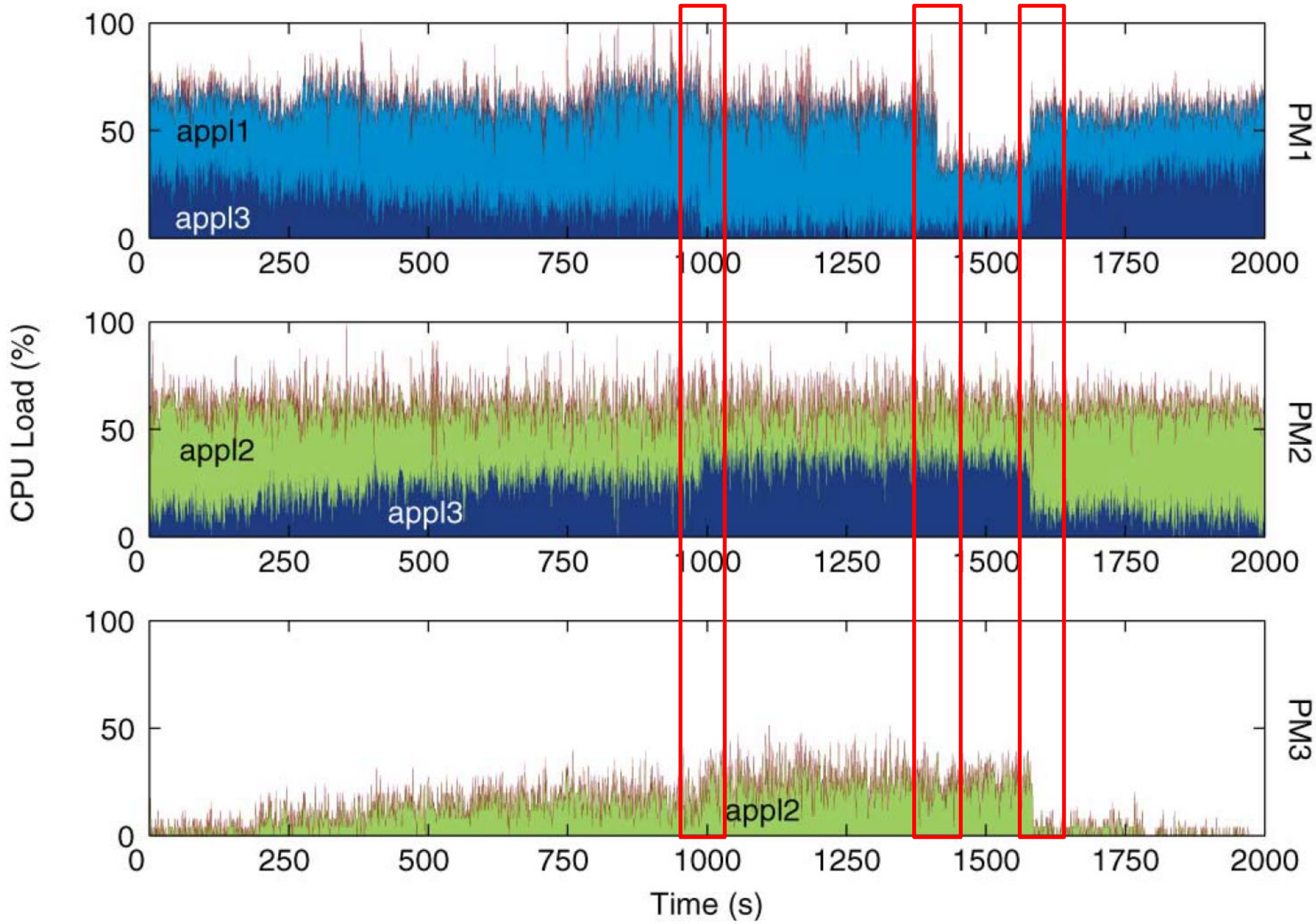
σ : the list of the input sequence
$A(\sigma)$: the number of bins used under the A algorithm (proposed)
$OPT(\sigma)$: the number of bins used under the optimal algorithm

Internet Applications

- Practical Considerations
  - Server Equivalence Class
    - Hardware settings – the same unit size
  - Class Constraint
  - Load Change – Invoke periodically
    - The arrivals or departures of several items
    - The granularity (we can capture) is limited by the load unit.
  - Optimization
    - At most one unfilled bin
    - Allowing temporary violation of the color set property

Internet Applications

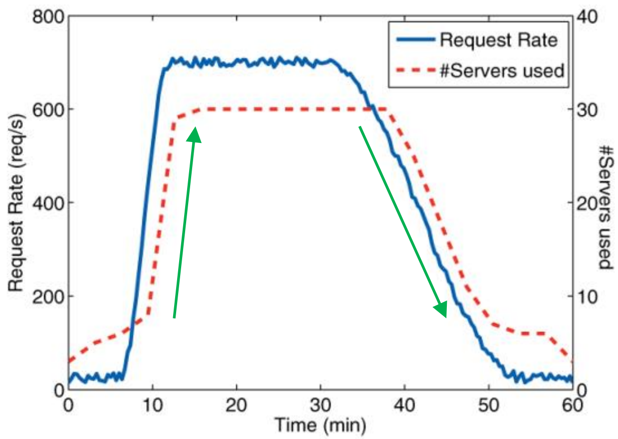
Load Shifting



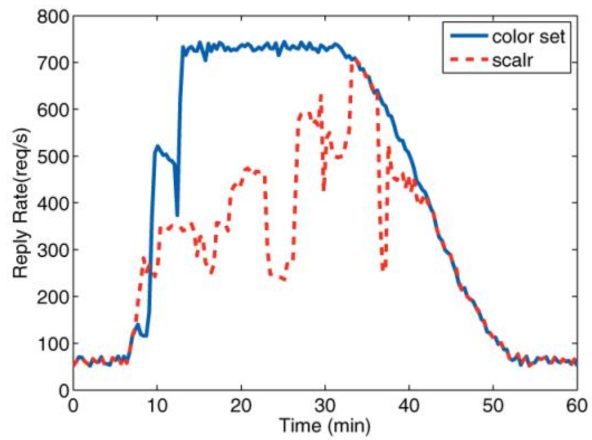
Internet Applications

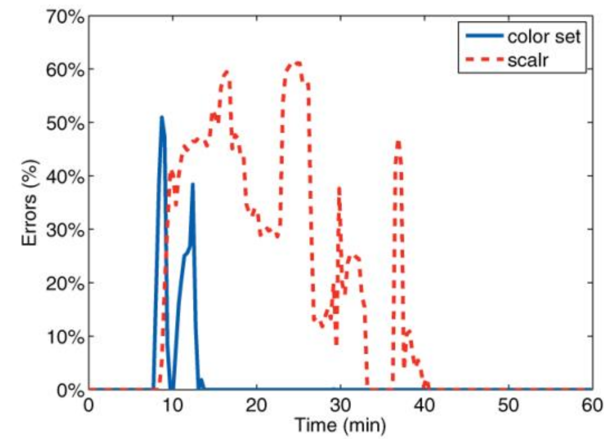
Auto Scaling

The Scalr open source implement in Amazon EC2

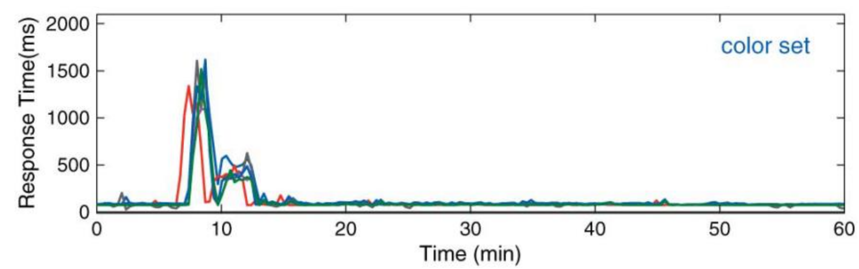


Request rate & servers used

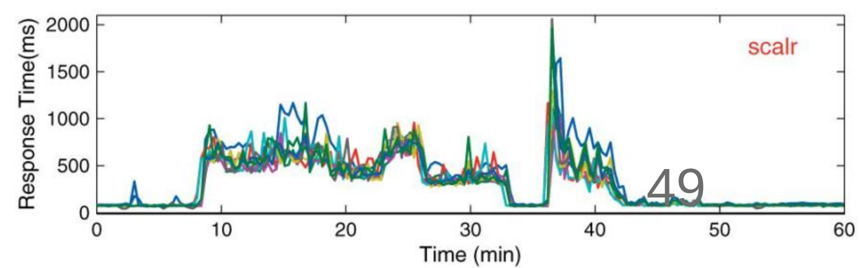




9 applications
30 servers



"Flash crowd" event



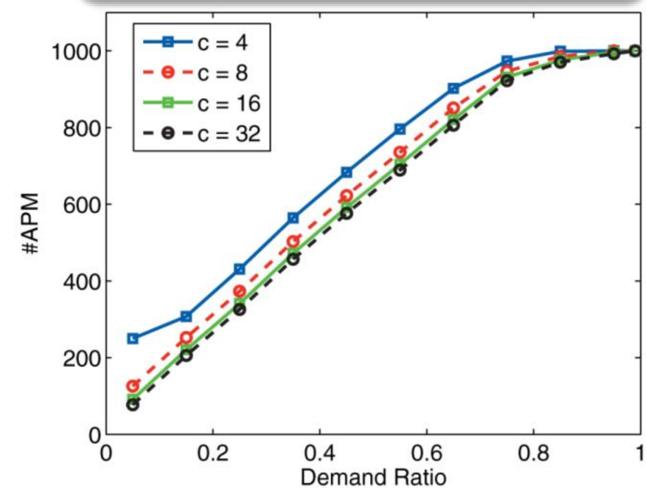
Internet Applications

Simulator – Demand

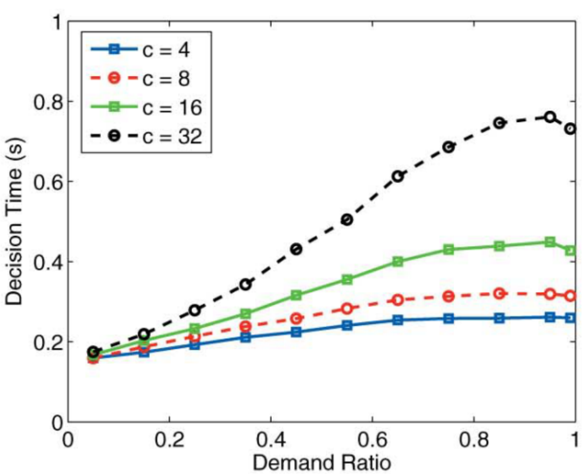
"Demand ratio" – between
(i) The aggregate demand of all applications
(ii) The aggregate capacity of all servers

1000 servers with 1000 applications
Demand ratio (D) : 0.05 to 0.99

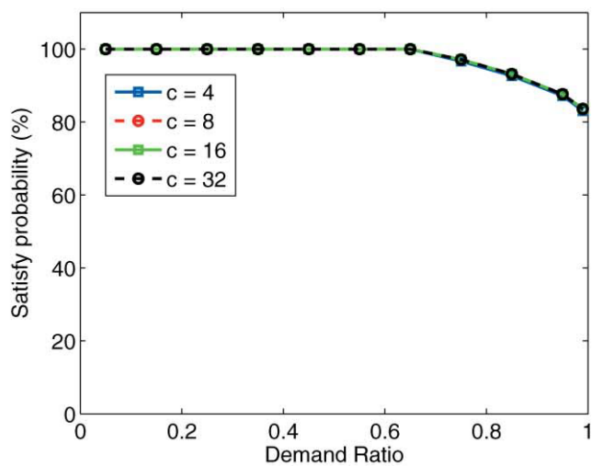
APM
(Active Physical Machine)



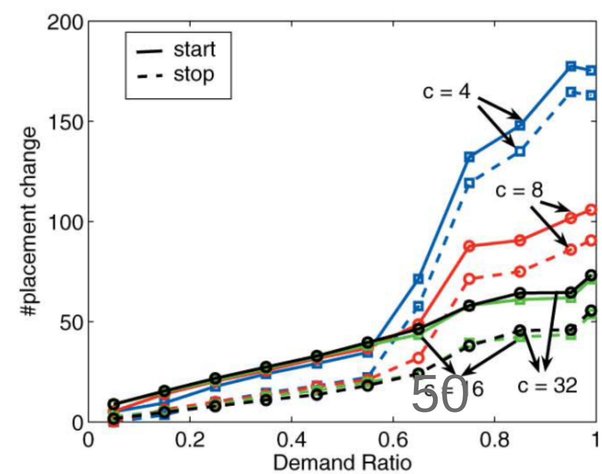
Decision time



Satisfy probability



Placement change



Internet Applications

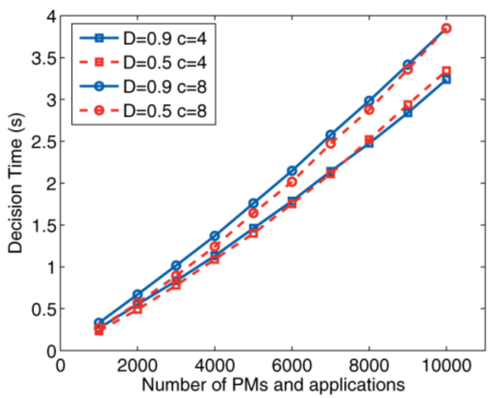
- Simulator

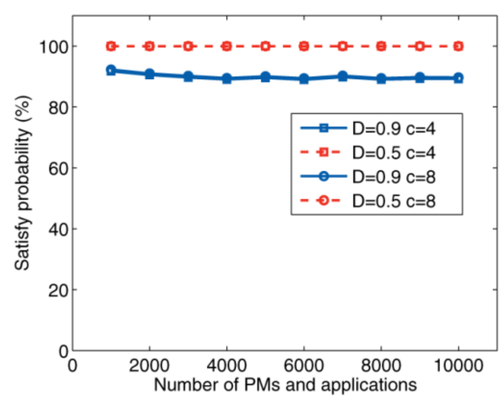
1000 servers & applications → **10000**

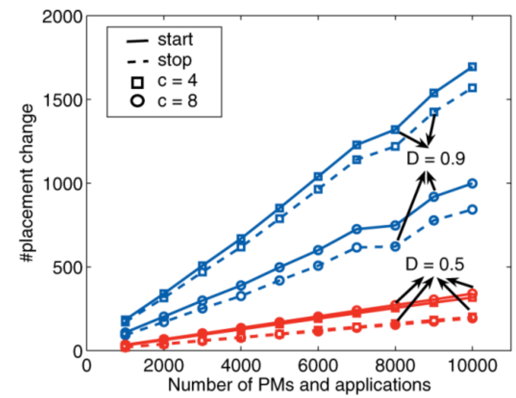
Decision time

Satisfy probability

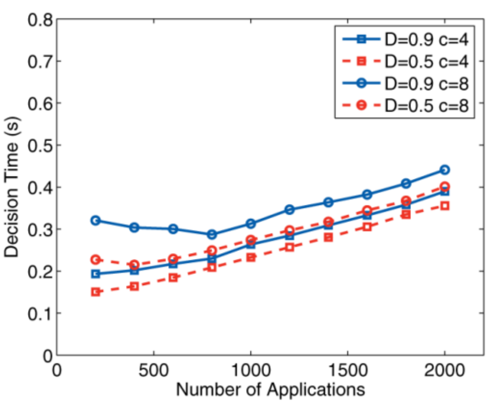
Placement change

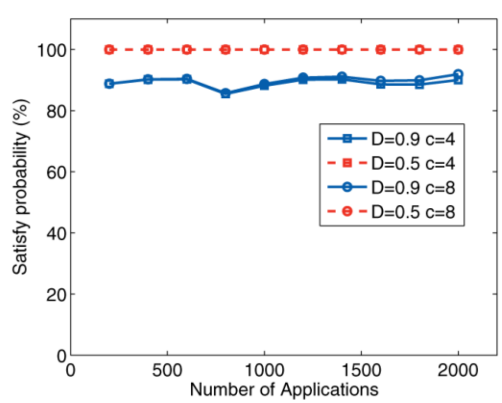


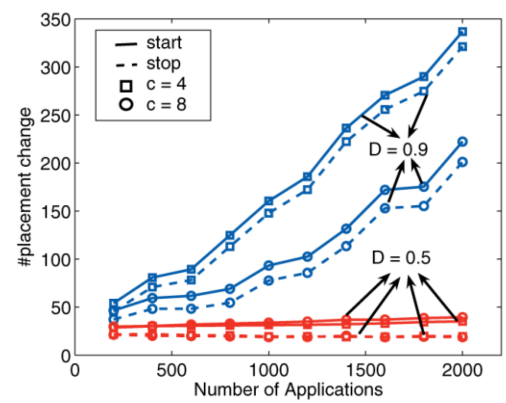




Scalability







Application number

Applications : 200→2000 & server : 1000

網路負載平衡（大型分散式排程）

52

Schedule first, manage later: Network-aware load balancing

Nahir, A.; Orda, A.; raz, d.

INFOCOM, 2013 Proceedings IEEE

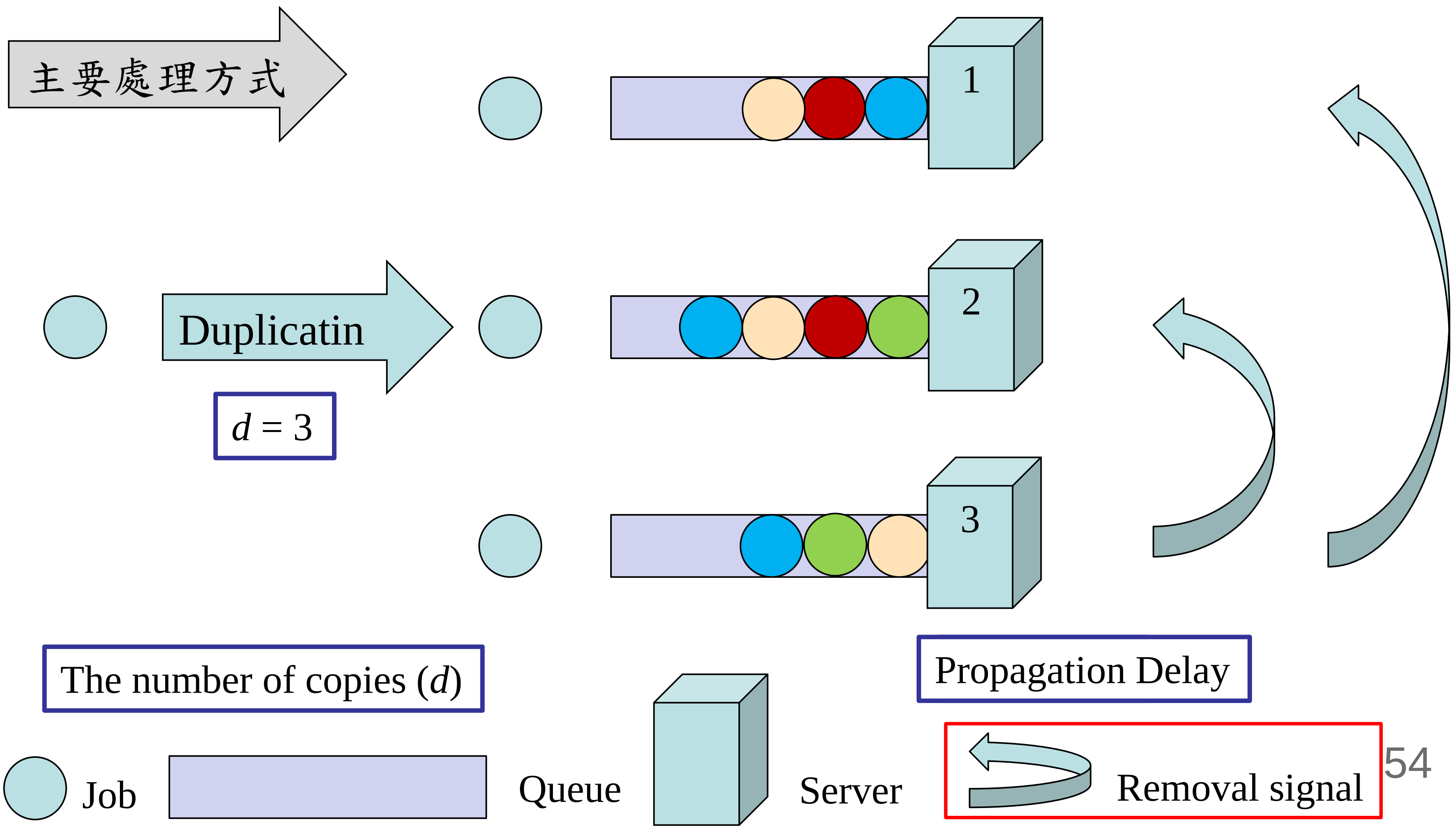
Distributed scheduling

動機

- Job (Request) allocation
  - Schedule first, manage later
    - The data collection process hinders
      - The selection of the executing server
      - The arrival of the job to that server.
    - No communication overhead
    - Without decision-making delay
  - Distributed load balancing techniques
    - Minimize the time a job spends till being assigned to a server.
    - Scale well & suit for existing data center
      - Duplication (複製)

Distributed scheduling

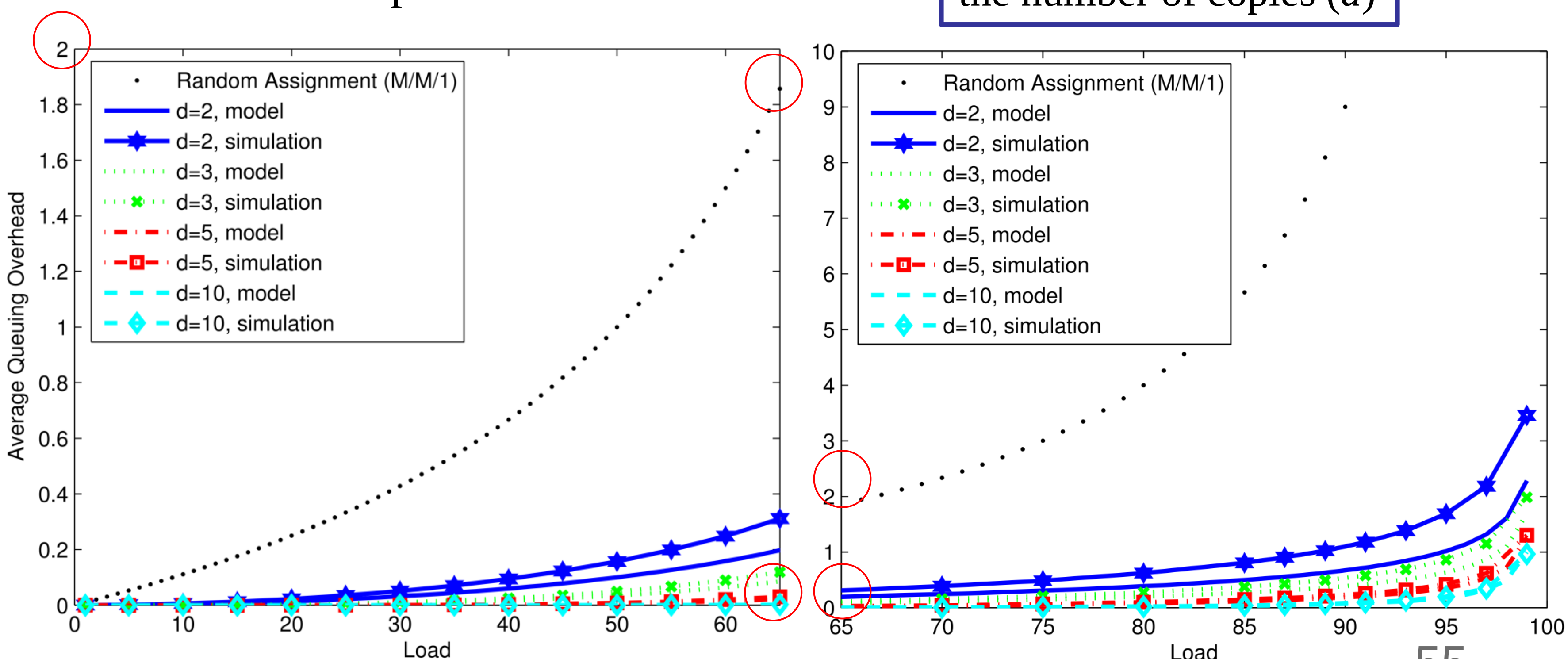
- Schedule first, manage later (Network-aware)



Implementation

- Performance analysis without Delay
  - When *d* grows, so does the accuracy of the model.
    - The improvement

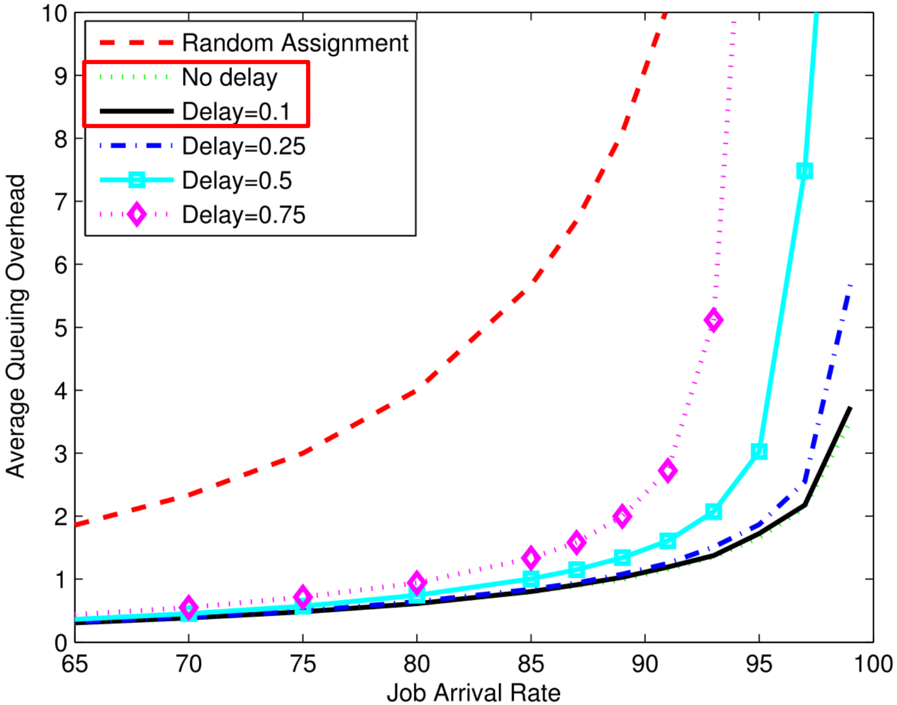
the number of copies (*d*)

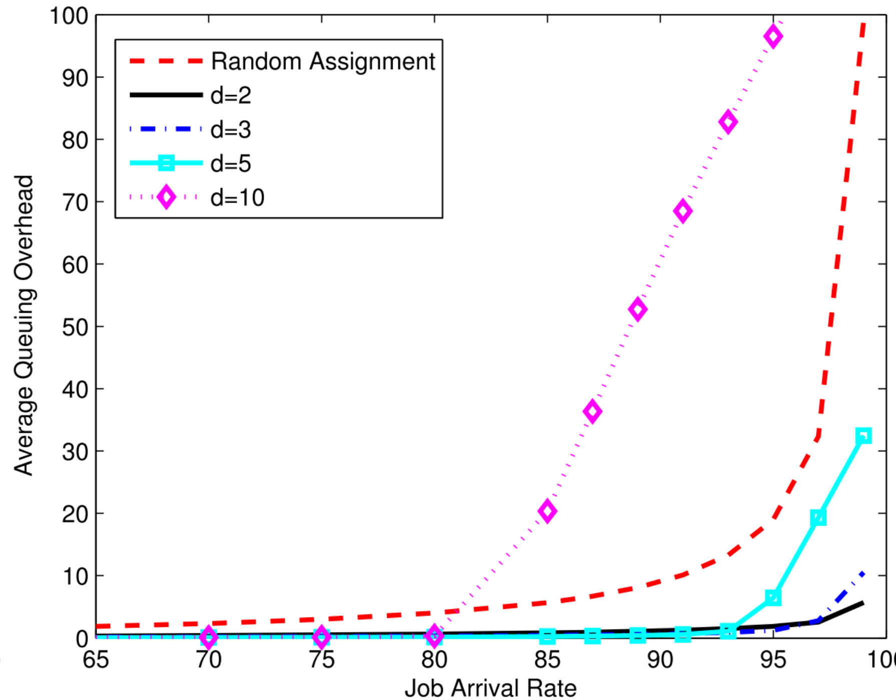


Average Queuing Overhead, no delays, high loads

Distributed scheduling

- The effect of delays analysis
  - The rise with **Delay** & ***d***

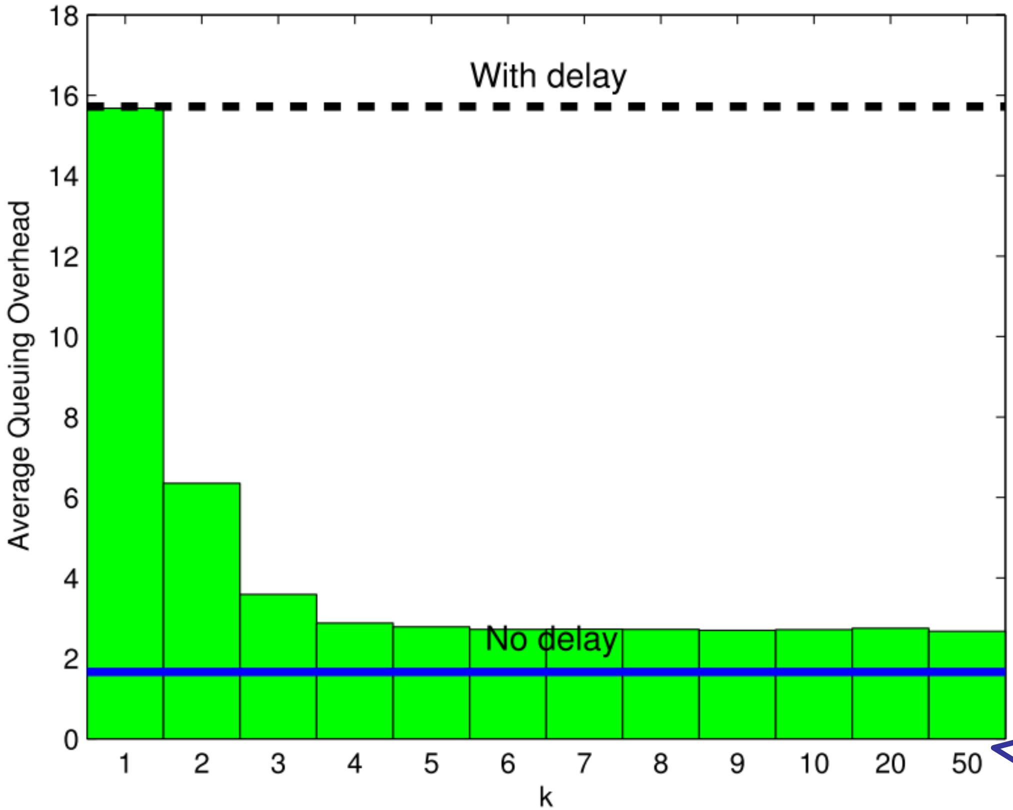




Expected service rate in presence of signal propagation delays, $d = 2$ (左), $T_d = 0.25$ (右), high load values

Distributed scheduling

- Addressing delays

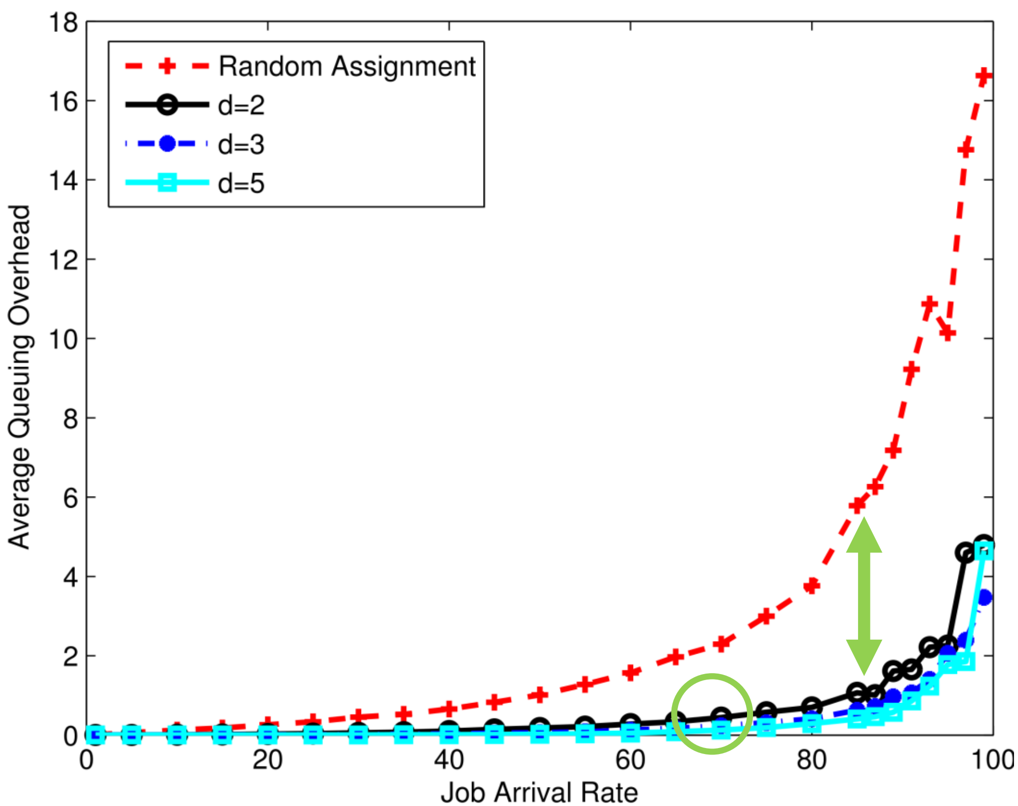


To select the next job from its queue for processing, the server would select, among the first ***k*** jobs.

Average queuing overhead with location in vector selection, d = 2 , T_d = 0.75 , load = 95% , with starvation-prevention

Distributed scheduling

- Real-world implementation & Evaluation
  - Amazon EC2



Average Queuing Overhead, AWS implementation

實作所考量的部分

- Execution component
- Queuing component
- Job-removal component

Conclusion & Comparison

- Web Scaling Frameworks (WSF)
  - More flexible prediction of performance
- A color set algorithm that decides the application placement and the load distribution achieves high **satisfaction ratio**.
- Distributed load balancing
  - Duplicating for the queuing time
  - Increasing scalability (large-scale)

比較這三篇的特色與差異

Conclusion & Comparison

	Auto-Scaling	Request Classification	Load Balance	Performance (Satisfaction ratio)	Delay Constraint
[1]	●	●			●
[2]	●		●	●	
[3]			●		●

比較這三篇的特色與差異

Final Project – Scalable Scheduling Services in Data Center Networks

與期中內容主題一致

實作部分：實驗 → 說故事

Reporter: 歐庭瑋

Advisor: 曾學文 教授

Outline

- Introduction
  - Scalability
  - CloudSim
- Scenario
  - Resource Provisioning
  - Scheduling Policy
- Conclusion
- Reference

帶一下之前的期中內容，主題一致

實驗的環境以及如何模擬

實驗的主要內容介紹，你做了什麼部份

Introduction

- Scalability
  - Scalability is a term used to describe how the application will handle increased loads of traffic volume.
  - Resource provisioning
    - Utilization
    - On demand
  - How to simulate ?
    - CloudSim

Introduction

實驗的環境介紹 (1)

- CloudSim [1]
  - Motivation
    - A generalized and extensible simulation framework
    - University of Melbourne (墨爾本)
    - CloudSim Toolkit 3.0.3

[1] http://www.cloudbus.org/**cloudsim**/

Introduction

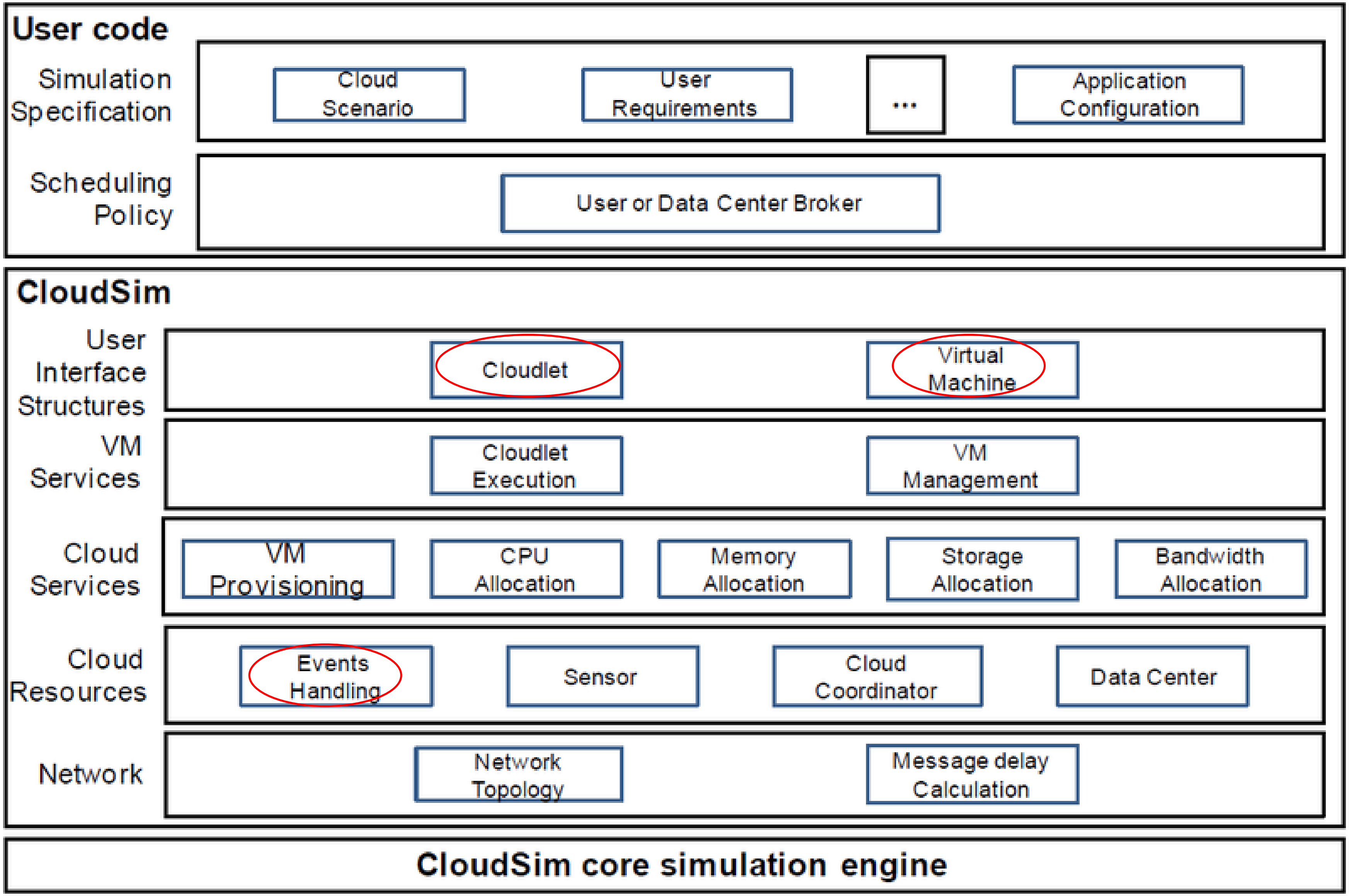
實驗的環境介紹 (2)

- Features
  - Cloud resource provisioning
  - Energy-efficient management of data center resources
  - Optimization of cloud computing
  - Research activities

  - Limitation: No Graphical User Interface (GUI)

- Prerequisites
  - Java & OOP

系統環境的整體架構


User code
Simulation Specification
Cloud Scenario
User Requirements
...
Application Configuration
Scheduling Policy
User or Data Center Broker
CloudSim
User Interface Structures
Cloudlet
Virtual Machine
VM Services
Cloudlet Execution
VM Management
Cloud Services
VM Provisioning
CPU Allocation
Memory Allocation
Storage Allocation
Bandwidth Allocation
Cloud Resources
Events Handling
Sensor
Cloud Coordinator
Data Center
Network
Network Topology
Message delay Calculation
CloudSim core simulation engine

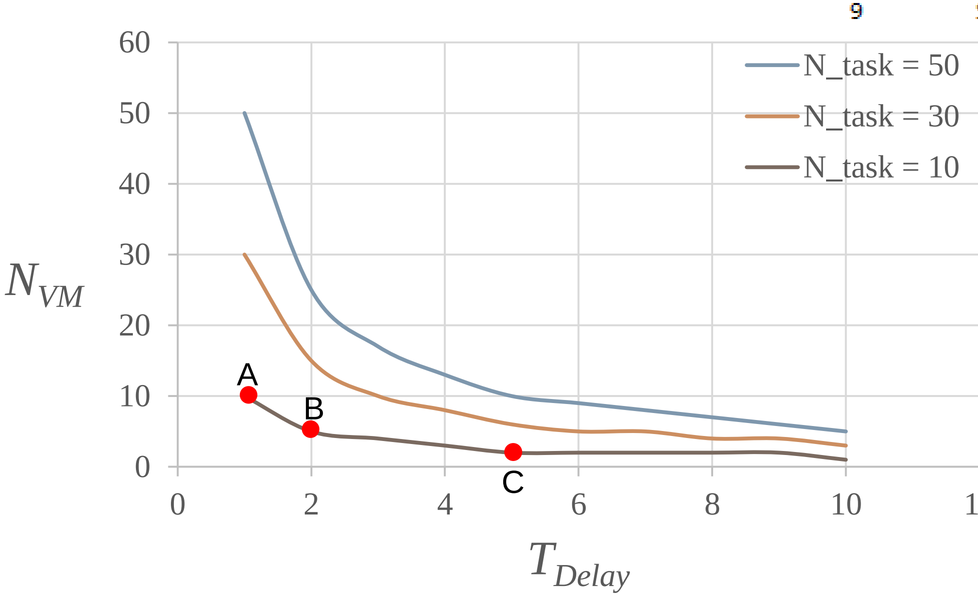
Resource Provisioning

- Auto-scaling [2] – Delay-constrained

$$N_{VM} = \frac{N_{task}}{T_{Delay}}$$

A

```
========== OUTPUT ==========
Cloudlet ID    STATUS      VM ID      Time    Start Time    Finish Time
    0          SUCCESS     0          1       0.2           1.2
    1          SUCCESS     1          1       0.2           1.2
    2          SUCCESS     2          1       0.2           1.2
    4          SUCCESS     4          1       0.2           1.2
    3          SUCCESS     3          1       0.2           1.2
    5          SUCCESS     5          1       0.2           1.2
    6          SUCCESS     6          1       0.2           1.2
    7          SUCCESS     7          1       0.2           1.2
    8          SUCCESS     8          1       0.2           1.2
    9          SUCCESS     9          1       0.2           1.2
```



B

```
========== OUTPUT ==========
Cloudlet ID    STATUS      VM ID      Time    Start Time    Finish Time
    0          SUCCESS     0          1       0.1           1.1
    1          SUCCESS     1          1       0.1           1.1
    2          SUCCESS     2          1       0.1           1.1
    4          SUCCESS     4          1       0.1           1.1
    3          SUCCESS     3          1       0.1           1.1
    5          SUCCESS     0          1       1.1           2.1
    6          SUCCESS     1          1       1.1           2.1
    7          SUCCESS     2          1       1.1           2.1
    9          SUCCESS     4          1       1.1           2.1
    8          SUCCESS     3          1       1.1           2.1
```

C

```
========== OUTPUT ==========
Cloudlet ID    STATUS      VM ID      Time    Start Time    Finish Time
    0          SUCCESS     0          1       0.1           1.1
    1          SUCCESS     1          1       0.1           1.1
    2          SUCCESS     0          1       1.1           2.1
    3          SUCCESS     1          1       1.1           2.1
    4          SUCCESS     0          1       2.1           3.1
    5          SUCCESS     1          1       2.1           3.1
    6          SUCCESS     0          1       3.1           4.1
    7          SUCCESS     1          1       3.1           4.1
    8          SUCCESS     0          1       4.1           5.1
    9          SUCCESS     1          1       4.1           5.1
```

[2] T. Fankhauser, Wang Qi, A. Gerlicher, C. Grecos, and Wang Xinheng, "Web scaling frameworks: A novel class of frameworks for scalable web services in cloud environments," *Communications (ICC), IEEE International Conference on* , pp.1760,1766, 10-14 June 2014

探討與期中報告相關的部分進行分析及模擬 (2)

Resource Provisioning

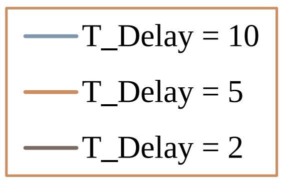
$$N_{VM} = \frac{N_{task}}{T_{Delay}}$$

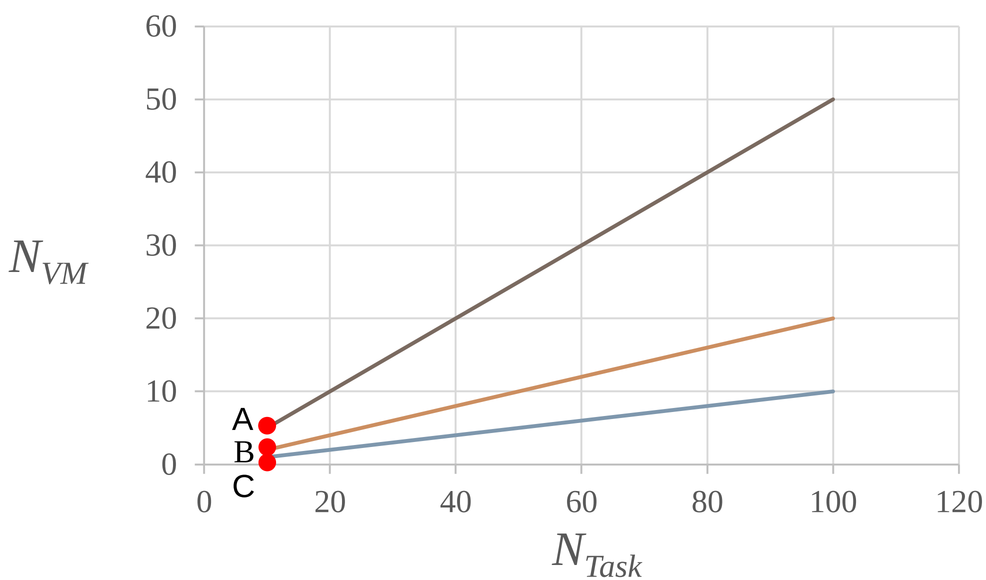
- Auto-scaling [2]

A

========== OUTPUT ==========

Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
0	SUCCESS	0	1	0.1	1.1
1	SUCCESS	1	1	0.1	1.1
2	SUCCESS	2	1	0.1	1.1
4	SUCCESS	4	1	0.1	1.1
3	SUCCESS	3	1	0.1	1.1
5	SUCCESS	0	1	1.1	2.1
6	SUCCESS	1	1	1.1	2.1
7	SUCCESS	2	1	1.1	2.1
9	SUCCESS	4	1	1.1	2.1
8	SUCCESS	3	1	1.1	2.1





B

========== OUTPUT ==========

Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
0	SUCCESS	0	1	0.1	1.1
1	SUCCESS	1	1	0.1	1.1
2	SUCCESS	0	1	1.1	2.1
3	SUCCESS	1	1	1.1	2.1
4	SUCCESS	0	1	2.1	3.1
5	SUCCESS	1	1	2.1	3.1
6	SUCCESS	0	1	3.1	4.1
7	SUCCESS	1	1	3.1	4.1
8	SUCCESS	0	1	4.1	5.1
9	SUCCESS	1	1	4.1	5.1

C

========== OUTPUT ==========

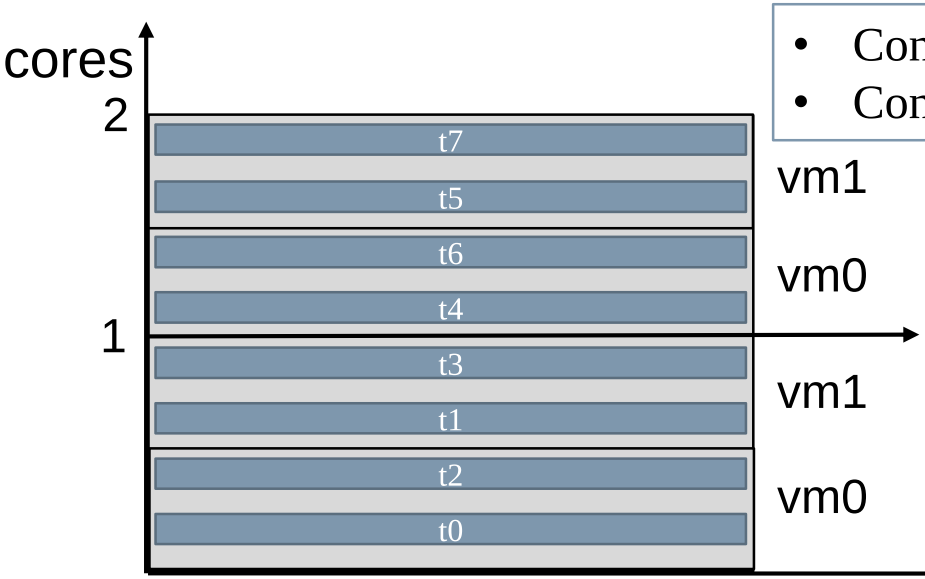
Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
0	SUCCESS	0	1	0.1	1.1
1	SUCCESS	0	1	1.1	2.1
2	SUCCESS	0	1	2.1	3.1
3	SUCCESS	0	1	3.1	4.1
4	SUCCESS	0	1	4.1	5.1
5	SUCCESS	0	1	5.1	6.1
6	SUCCESS	0	1	6.1	7.1
7	SUCCESS	0	1	7.1	8.1
8	SUCCESS	0	1	8.1	9.1
9	SUCCESS	0	1	9.1	10.1

[2] T. Fankhauser, Wang Qi, A. Gerlicher, C. Grecos, and Wang Xinheng, "Web scaling frameworks: A novel class of frameworks for scalable web services in cloud environments," *Communications (ICC), IEEE International Conference on* , pp.1760,1766, 10-14 June 2014

Scheduling Policy

- Time shared for VMs and tasks [3]

- Concurrently shared by the VMs
- Concurrently divided among the task units



time

探討與期中報告相關的部分 (3)
Requests 的運作處理方式 –
平行化處理

========== OUTPUT ==========

Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
0	SUCCESS	0	4	0.1	4.1
2	SUCCESS	0	4	0.1	4.1
4	SUCCESS	0	4	0.1	4.1
6	SUCCESS	0	4	0.1	4.1
1	SUCCESS	1	4	0.1	4.1
3	SUCCESS	1	4	0.1	4.1
5	SUCCESS	1	4	0.1	4.1
7	SUCCESS	1	4	0.1	4.1

VM No. : 2
Tasks No. : 8

[3] Buyya, R., Ranjan, R., and Calheiros, R. N. “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In High Performance Computing & Simulation,” 2009. HPCS'09. International Conference on (pp. 1-11). IEEE

Resource Provisioning

探討與期中報告相關的部分 (4)
根據硬體的限制來管理與開啟VM

- In data centers
  - Parameters

VM number : 20
Task number: 40

Virtual Machine	
Image Size	10000 MB
Memory (RAM)	512 MB
MIPS	1000
Bandwidth	1000
No. of CPUs	1

Data Center host		Original	Modified
MIPS	Host 1 (quad-core)*2	1000	2000
	Host 2 (dual-core) *2		
Memory (RAM)		2048 MB	4069 MB
Storage		100 GB	100 GB
Bandwidth		100000	100000

Resource Provisioning

探討與期中報告相關的部分 (4)
實驗結果

Original

========== OUTPUT ==========

Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
4	SUCCESS	4	3	0.2	3.2
16	SUCCESS	4	3	0.2	3.2
28	SUCCESS	4	3	0.2	3.2
5	SUCCESS	5	3	0.2	3.2
17	SUCCESS	5	3	0.2	3.2
29	SUCCESS	5	3	0.2	3.2
6	SUCCESS	6	3	0.2	3.2
18	SUCCESS	6	3	0.2	3.2
30	SUCCESS	6	3	0.2	3.2
7	SUCCESS	7	3	0.2	3.2
19	SUCCESS	7	3	0.2	3.2
31	SUCCESS	7	3	0.2	3.2
8	SUCCESS	8	3	0.2	3.2
20	SUCCESS	8	3	0.2	3.2
32	SUCCESS	8	[illegible]	[illegible]	[illegible]
10	SUCCESS	10	[illegible]	[illegible]	[illegible]
22	SUCCESS	10	[illegible]	[illegible]	[illegible]
34	SUCCESS	10	[illegible]	[illegible]	[illegible]
9	SUCCESS	9	3	[illegible]	[illegible]
21	SUCCESS	9	[illegible]	[illegible]	[illegible]
33	SUCCESS	9	[illegible]	[illegible]	[illegible]
11	SUCCESS	11	[illegible]	[illegible]	[illegible]
23	SUCCESS	11	[illegible]	[illegible]	[illegible]
35	SUCCESS	11	3	0.2	3.2
0	SUCCESS	0	4	0.2	4.2
12	SUCCESS	0	4	0.2	4.2
24	SUCCESS	0	4	0.2	4.2
36	SUCCESS	0	4	0.2	4.2
1	SUCCESS	1	4	0.2	4.2
13	SUCCESS	1	4	0.2	4.2
25	SUCCESS	1	4	0.2	4.2
37	SUCCESS	1	4	0.2	4.2
2	SUCCESS	2	4	0.2	4.2
14	SUCCESS	2	4	0.2	4.2
26	SUCCESS	2	4	0.2	4.2
38	SUCCESS	2	4	0.2	4.2
3	SUCCESS	3	4	0.2	4.2
15	SUCCESS	3	4	0.2	4.2
27	SUCCESS	3	4	0.2	4.2
39	SUCCESS	3	4	0.2	4.2

VM number : 12→20
Task per VM : 3 ~ 4 → 2
Average finish time : 3 ~ 4 → 2

Modified

========== OUTPUT ==========

Cloudlet ID	STATUS	VM ID	Time	Start Time	Finish Time
0	SUCCESS	0	2	0.2	2.2
20	SUCCESS	0	2	0.2	2.2
1	SUCCESS	1	2	0.2	2.2
21	SUCCESS	1	2	0.2	2.2
2	SUCCESS	2	2	0.2	2.2
22	SUCCESS	2	2	0.2	2.2
4	SUCCESS	4	2	0.2	2.2
24	SUCCESS	4	2	0.2	2.2
6	SUCCESS	6	2	0.2	2.2
26	SUCCESS	6	2	0.2	2.2
8	SUCCESS	8	2	0.2	2.2
28	SUCCESS	8	2	0.2	2.2
10	SUCCESS	10	2	0.2	2.2
30	SUCCESS	10	2	0.2	2.2
[illegible]	[illegible]	12	2	0.2	2.2
[illegible]	[illegible]	12	2	0.2	2.2
[illegible]	[illegible]	3	2	0.2	2.2
[illegible]	[illegible]	3	2	0.2	2.2
[illegible]	[illegible]	5	2	0.2	2.2
[illegible]	[illegible]	5	2	0.2	2.2
[illegible]	[illegible]	7	2	0.2	2.2
[illegible]	[illegible]	7	2	0.2	2.2
[illegible]	[illegible]	9	2	0.2	2.2
29	SUCCESS	9	2	0.2	2.2
11	SUCCESS	11	2	0.2	2.2
31	SUCCESS	11	2	0.2	2.2
13	SUCCESS	13	2	0.2	2.2
33	SUCCESS	13	2	0.2	2.2
14	SUCCESS	14	2	0.2	2.2
34	SUCCESS	14	2	0.2	2.2
15	SUCCESS	15	2	0.2	2.2
35	SUCCESS	15	2	0.2	2.2
16	SUCCESS	16	2	0.2	2.2
36	SUCCESS	16	2	0.2	2.2
17	SUCCESS	17	2	0.2	2.2
37	SUCCESS	17	2	0.2	2.2
18	SUCCESS	18	2	0.2	2.2
38	SUCCESS	18	2	0.2	2.2
19	SUCCESS	19	2	0.2	2.2
39	SUCCESS	19	2	0.2	2.2

Conclusion

前後呼應，扣緊主題

- Scalability
  - Resource provisioning
- Task characteristics (Request classifier)
  - Various application
    - Flexible allocation & management
    - Achieve high satisfaction ratio
  - On-demand
    - Auto-scaling → Elasticity

Thank you

祝大家期中期末報告都可以順利過關~

Reference

- [1] http://www.cloudbus.org/**cloudsim**/
- [2] Buyya, R., Ranjan, R., and Calheiros, R. N. “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In High Performance Computing & Simulation,” 2009. HPCS'09. International Conference on (pp. 1-11). IEEE
- [3] Buyya, R., Ranjan, R., and Calheiros, R. N. “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In High Performance Computing & Simulation,” 2009. HPCS'09. International Conference on (pp. 1-11). IEEE