科技部補助專題研究計畫成果報告
期末報告

適用於雲端環境下的動態資源服務

計 畫 類 別 ：個別型計畫
計 畫 編 號 ：NSC 102-2221-E-004-003-
執 行 期 間 ：102 年 08 月 01 日至 103 年 07 月 31 日
執 行 單 位 ：國立政治大學資訊科學系

計 畫 主 持 人：蔡子傑
共 同 主 持 人：張宏慶
計畫參與人員：碩士班研究生-兼任助理人員：韓建淳
碩士班研究生-兼任助理人員：詹筱璇
碩士班研究生-兼任助理人員：李致賢
碩士班研究生-兼任助理人員：唐聖傑
博士班研究生-兼任助理人員：詹賀翔

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：
1.公開資訊：本計畫可公開查詢
2.「本研究」是否已有嚴重損及公共利益之發現：否
3.「本報告」是否建議提供政府單位施政參考：否

中 華 民 國 103 年 10 月 31 日

中 文 摘 要 ： 雲端運算是當下極受矚目的技術。做為在網路上提供服務的一個新選擇，雲端平台擁有很高的彈性與以使用量付費的機制，讓使用者不再需要提前做審慎的硬體採購規畫，可以隨時在需求增加時，再添購更多服務。

然而，在雲端環境中提供服務時，以固定數量的虛擬主機配置，在面對少數使用者時，將造成許多虛擬主機的閒置，而在面對數量遠大於預期的使用者時，將面臨無法有效提供服務的問題。在現有的雲端平台中，已經有 Auto Scale 技術可以以動態的方式增加虛擬主機，提供水平擴充，以處理意外的負載，以及動態減少虛擬主機，以節省成本，但是大都必須仰賴使用者自行提供的 threshold 設定，而且沒有預測未來的 workload 變化，所以動態擴充的效果可能會不如預期。

本研究的議題如下：針對雲端環境，提出一個動態式的資源服務。由於增加或減少虛擬主機有一定的時間成本，如果在某個時間內將要處理大量的使用者需求，透過 Queuing Model 與 時間序列分析來預測系統未來 workload 與 request rate 的變化，設計一套動態式的資源服務，根據所預測的結果，系望能在系統負載過高之前，就能進行水平擴充。而如果在某個時間內使用者數量不多時，系統能自動減少虛擬主機，以節省服務成本。另外，在同樣的資源配置下，本研究探討如何透過動態的 task assignment policy，使系統整體能達到更高的服務效能。

中文關鍵詞： 雲端運算、自動擴增、等候理論、時間序列分析、工作指派

英 文 摘 要 ： cloud computing, auto sacel, queueing theory, time series analysis, task assignment

英文關鍵詞： Cloud computing is one of the most important technologies today. As a new choice for hosting and delivering services over the Internet, users do not have to carefully plan ahead for hardware due to cloud computing features high elasticity and “pay as you go“ service. More computing service is purchased when demand increases.

1. Introduction

One of the biggest benefit of using cloud computer as a service is the “Pay as you go” payment method. The dynamic scalability of available resource with different costs is desirable for fast-paced software development which is massively emerging lately. Small companies can benefit from flexible payment, and start-ups can benefit from instant available resources with very low cost.

However, resource planning in cloud computing is not a trivial problem, especially in IAAS platforms where users need to rent virtual machines and setup the whole architecture themselves. It is not easy to determine the optimal resource requirement for a service. In coming workloads varies all the time. Benchmarks and simulations are not possible for immature systems. Therefore, over provisioning of resources (pay too much for unused resources) or under provisioning of resources can happen often, and may result into disasters such as service unavailability, then there will be unhappy customers.

To solve the resource provisioning problem, we hope to tackle the problem in a mathematical way. We hope to build effective models for the resource provisioning problem in cloud environments. As a contrast to benchmarking the system and building simulations of the system, our model has the following benefits:

1. Portable: Dynamically adjust to different system architecture.

2. Flexible: can be used under different demands of workloads and different level of system resources.

In resource provisioning for cloud computing, there are two major issues: auto scaling and load balancing. In this research, we aim to find a mathematical model to solve the above problems.

2. Research goal

Our goal is to provide mathematical model to solve the auto scale and load balancing problem. A good model can be very useful; a service built in cloud can benefit from optimal system resource provisioning (effective cost/performance), and good overall performance under certain system resource.

System resource in cloud computing services can be dynamically allocated, i.e. automatically increase the number of virtual machines to deal with more demanding workload or decrease the number of virtual machines to save money.

Auto-scaling is based on certain

rules, for an example, add three instances when an average CPU usage reaches 70%, or decrease one instance when average CPU usage drops below 20%. However, it's very difficult to determine whether this setting is "optimal" or "effective" for handling sudden increase of user's usage demand in peak period or during off peak period such as early morning, without full benchmarks or simulations. Our goal is to provide a model to capture dynamic user demand and system workloads, and forecast the system performance metric for auto scale.

In cloud computing environment, horizontal scaling of system resource is preferred to vertical scaling. Horizontal scaling in cloud computing environment is ideal because of its flexibility to handle different workload demands at different times. This will form a server of servers, and the load balance strategy, or task assignment policy, for incoming jobs will hugely affect the overall system performance. Using the right task assignment policy is particularly import when the job size has high variability in its distribution. For an example, if a poor task assignment policy is applied, short jobs may suffer very long queuing time when queued behind jobs with long service time. We propose a task assignment policy based on performance modeling of job queues, aiming to forecast the server farm's performance. With the model-driven task assignment policy, our goal is to maximize the server farm's overall performance under specific resource provisioning.

3. Related Work

Previous works in auto scale in cloud computing studies the optimal resource provisioning to fulfill QOS. Related work includes works in Queuing Model、Statistical Machine Learning、Control Theory [1].

A Queuing-based Model for Performance Management on Cloud [2]

Dynamic resource provisioning is an important issue in cloud computing. This paper proposes a model based on Queuing Model. The Web Application is modeled as a queue and the predicted performance metric is used for automatic adding or killing instances (auto scale).

Black-Box Approach to Capacity Identification for Multi-Tier Applications [3]

This research studies capacity planning for an multi-tier web application. The author uses unsupervised machine learning from system logs to predict the system capacity, and find the optimal resource to fulfill the SLA agreement.

Support vector regression for link load prediction[5]

This research uses the Support Vector Machine (SVM) to forecast link workload. The linked load prediction is based on historical data using embedded process, without overwhelming computing complexity.

4. Research Methodology

a. Auto Scale in cloud computing

To provide auto scale in cloud, we propose a statistical model combining queuing model and time series analysis. Auto scale is driven by two approach, the reactive and proactive approach. In reactive approach, the system responds to different preset rules and auto scales the system based on the rules. However, when encountering a burst of user demand or high varying workload, the auto scale policy may not catch up the demand in time. In a proactive approach, user demand and system workload is forecasted by the system, and the system can auto scale a head of time before system resource gets under provisioned or over provisioned. Our approach is a proactive approach.

System Architecture

A typical scalable Cloud Application is composed of a Load Balancer and a group of VMs to support horizontal scaling. The Load Balancer dispatches new incoming requests to VMs to balance the load of the VMs.

Our goal is to build an analytical model for the cloud auto scale system. When jobs arrives in the system, it will need some computing time and resources. A system can handle only a maximum of jobs at a time, therefore, as job arrival rate rises but system resource is finite, a queue will be formed. Queuing theory is ideal for modeling the queue.

A service hosted in cloud computing environment typically solves a large number of users. Therefore, we can assume the distribution of the arrival rate to be memory less. However, regarding to the distribution of the service rate, we cannot simply make such assumption. Job size may vary intensely, short jobs may suffer long queuing time when queued behind large jobs. In a result, we assume the distribution of the service rate to be general. For the queuing model, we model the system as a M/G/1 queue. The queue contains the incoming requests waiting in the Load Balancer in order to be dispatched to a VM. The VMs behind the Load Balancers serve the requests, we model all VMs as a whole server.

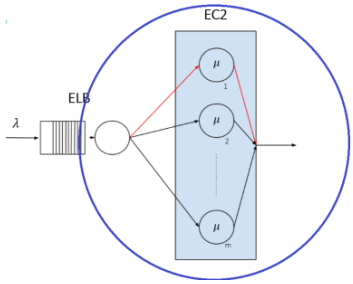


Figure 1: The Queuing System for a Web Application hosted on Amazon EC2

In a cloud system, system workload may intensely vary. The M/G/1 model assumes the service rate of the queue to be general, which is ideal to use in this scenario. However, since the M/G/1 captures the variation of the system service rate, a time window needs to be specified. We use the time series analysis to estimate the ideal window size for estimating the variation of the service rate.

Parameter Estimation

Using the Latency Metric from Amazon Cloud Watch, we have:

$$S = \{s_{t-1}, s_{t-1}, \dots s_0\},\ C_v{}^2 = \frac{Var(S)}{E[S]^2}$$

In a M/G/1 queue, the average waiting time depends on the service time of the customer being served in the system. Therefore, the waiting time is correlated with the variance of the service time of the customers. However, from the queuing model alone, it is unclear how to select the length of the service time's distribution in order to estimate an accurate E[S]. Since we need to find the most correlated service times, we use Time Series Analysis to find the lag by AR model. After obtaining the lag, we use the lag to build a new series of the service times that is correlated with the newest service time. We have:

$$S_{t,k} = \{s_{t+1}, s_t, s_{t-1}, s_{t-2}, \dots s_{t-k-2}\},\ C_v{}^2 = \frac{Var(S_t{}')}{E[S_{t,k}]^2},$$

where k=lag in AR model

To estimate s_{t+1} and k, we use the differenced autoregression of order p to derive the predictive value of s_{t-1} based on the observations $\{s_t - s_{t-1}, s_{t-1} - s_{t-2}, \dots, s_2 - s_1\}$. The order p of autoregressive (AR) model is obtained with Bayesian information criterion (BIC). It is the value minimizing the BIC, which is a criterion for model selection among a finite set of models and defined by

$$\mathrm{BIC}(p) = -2 \ln \hat{L}(p) + p \ln n,$$

where $\hat{L}(p)$ is the maximized value of the likelihood function of the model AR(p) and n is the number of observations. In this work, the set of candidate models is specified as

$$\{\mathrm{AR}(p),\ p = 1, 2, \dots, 5\}.$$

2.4 Proposed Hybrid Model

We can adjust the M/G/1 model to:

$$\mathrm{E[T]} = E[s_{t+1}] + \mathrm{E}[\, S_{t,k}] \left(\frac{{C_v}^2+1}{2}\right)\left(\frac{\rho}{1-\rho}\right),$$

where $E[s_{t+1}]$ is the estimated service time for the customer, $\mathrm{E}[\, S_{t,k}]$ is the new estimated expected system service time by using ARMA model, and ${C_v}^2$ is the squared coefficient of $\mathrm{E}[\, S_{t,k}]$.

Finally, we still need one more parameter for our model, namely the traffic intensity. The traffic intensity models the intensity of the system's utilization in a period of time. System utilization cannot be acquired easily, therefore we make an estimation using historical data. For traffic intensity ρ, we estimate ρ by maximizing the log-likelihood function:

$$\log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi {\sigma_i}^2/n_i}} exp\left[-\frac{\left(\hat{T}_i - T_i\right)^2}{2{\sigma_i}^2/n_i}\right]$$

Which is equivalent as minimizing the weighted sum of square error as:

$$\sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{{\sigma_i}^2/n_i}$$

We can then estimate the parameter ρ by optimizing:

$$\hat{\rho} = \underset{\rho}{\operatorname{argmin}} \sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{{\sigma_i}^2/n_i}$$

Evaluation

Currently there do not exist a standard for benchmarking cloud services. Most of the benchmark system relies on model generated user workloads, since we proposed a dynamic model, we hope to evaluate our model in a realistic scenario. We use Wikibench[12] and scaled Wikipedia traces to generate the workload, and use MediaWiki for the web application. The Wikibench uses scaled real user data log from the Wikipedia website. The testbed is hosted on Amazon EC2 infrastructure and services, built on one Elastic Load Balancer to load balance incoming requests to three m1.large VMs hosting the MediaWiki web service, and one m1.large Amazon RDS for the MYSQL database holding the Wikipedia pages. The Wikibench workload generator is hosted on three m1.medium instances. All the instances run in the same availability zone(us-west 2a). Evaluation shows our proposed hybrid model has improved accuracy compared to using M/G/1 alone.

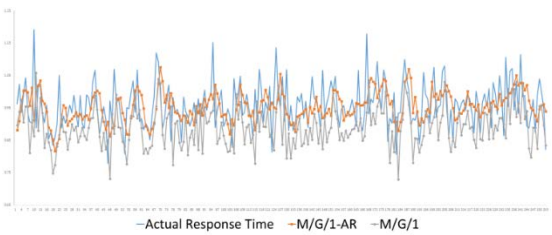


Figure 3. Predict response time with M/G/1 model and proposed approach of M/G/1 and AR model

From the emulation results, our

hybrid model out performs the M/G/1 standalone model. Since the workload do not vary very intensely in our emulation, our model did not greatly outperform the M/G/1. In our future work, we plan to test our model in a work demanding scenario, such as in Map-Reduce applications.

b. Model Drivel Task Assignment Policies in cloud computing environment

Cloud computing is a distributed environment. A service hosted in the cloud is formed by a farm of instances, or virtual machines. When there are many jobs to be processed in a distributed environment, task assignment policies are needed. We propose an analytical model to model the server farm and a dynamic task assignment policy based by the prediction metrics of the analytical model.

When assigning tasks to servers, it is desirable to keep a low overall mean service time for the jobs, e.g. users will not need to wait for a long time when querying in a website, and short jobs are not blocked behind long jobs for a long time. When dealing with complicated workload, which happens often in the cloud computing environment, wrong task assignment policy can lead to very long service times, e.g. when short jobs suffer from starvation when queued behind long jobs, or some servers are overwhelmed by too much works while some servers are idle with no jobs at all, wasting system resources.

In a farm of servers with jobs being served in a First-Come-First-Service manner, there's some common policies for the task assignment issue. First of all, consider RANDOM, where each job is assigned to one if the servers with equal probability. This aims to equalize the expected number of jobs in each server. Second, a common policy would be ROUND-ROBIN, also aims to equalize the expected number of jobs in each server. Another policy is JSQ (Join-the-Shortest-Queue), each job is assigned to the host with the fewest job to process (shortest queue of jobs) when job arrives. The JSQ tries to equalize the number of jobs in each server in a dynamical manner.

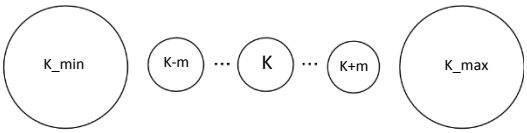
The performance of Task Assignment policies varies under different type of workloads. If the size of jobs is nearly the same all the time, all three policies can keep the system well load balanced with equal amount of works for each server. However, when dealing with complicated workloads, such as "heavy-tailed" workloads, a dynamic task assignment policy is needed.

To tackle the heavy tailed workload

in cloud computing environment, we propose a Least-Work-left based task assignment policy, in which each job goes to the queue where it will achieve the lowest possible response time. In practical applications, it's usually not possible to obtain the job size before the task assignment, therefore we propose a model to predict the server's expected response time, then assign incoming jobs to the server that has the lowest expected response time. The expected response time can also be predicted by Queuing Model such as M/G/1 or G/G/1 model, but queuing model gives a long term estimation, which may not be able to adapt to the high time varying characteristic of the system performance. Our model aims to model the system's performance in a shorter term.

We propose an analytical model for the server farm using the discrete time Markov Chain. For each server, we have a Markov Chain, where the states represents the current length of the queued jobs in the server, and the transition probability standing for the probability of moving from a number of queued jobs to another number of queued jobs. With the Markov Chain, we can model the probability of each transition state of moving from a queue length to another queue length. Then, from Little's result and average arrival rate, we can predict the expected queuing time of the server.

Below is a diagram of the Markov Chain.



The Proposed Markov Chain

When a server has a queue of jobs with length K, since the system resource of the server is finate, the server's state in the next time window is limited. In other words, in the next time window, the state should be constrained to k-m and k+m. For the other states, we merge the states that are very unlikely to reach as K_max and K_min.

From a history of time stamp and queue length, we can construct the transition matrix of the Markov Chain of the server is P, and with π P= π , we can solve the Markov Chain and obtain the expected queue length of the server. With Little's result and the mean arrival rate, we can obtain the expected queuing time.

We have: $\mathrm{E[T]} = E[T_q] + \mathrm{E[\,S]}$, where $E[T_q]$ is the expected queuing time in the server for a job calculated by our Markov Chain, and E[S]is the expected service time of a job. With E[T], we can assign the incoming job to the server with the lowest expected response time. If our model is accurate, our task assignment policy can always assign job to the "fastest" server that will handle the job in an shortest amount

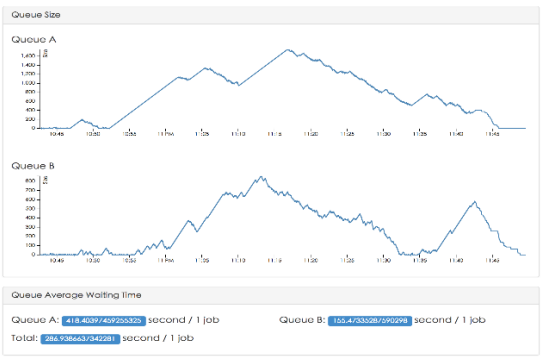
of time.

Evaluation

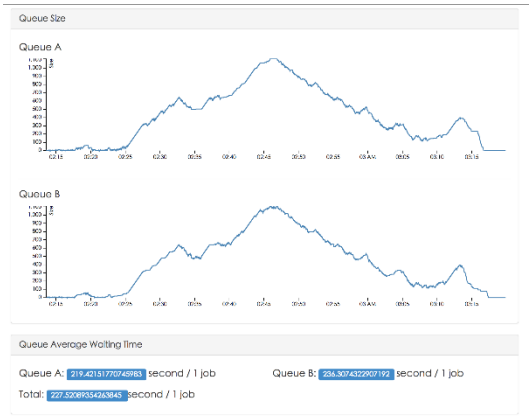
We evaluate our model in a simulation. The simulation is built by Ruby on Rails, Redis, and Python SciPy package. The simulation is run on a MacBook Pro I7 16G ram. We run the simulation by using the Pareto Distribution with alpha=1.1, which is used in the SURGE web workload generator.

We compare our model with two common task assignment policies: the ROUND ROBIN and the JSQ (Shortest job first) policy. Job arrival rate follows the Pareto Distribution with alpha=1.1, assigned into two servers. Simulation results shows our model out performs the conventional task assignment policies.

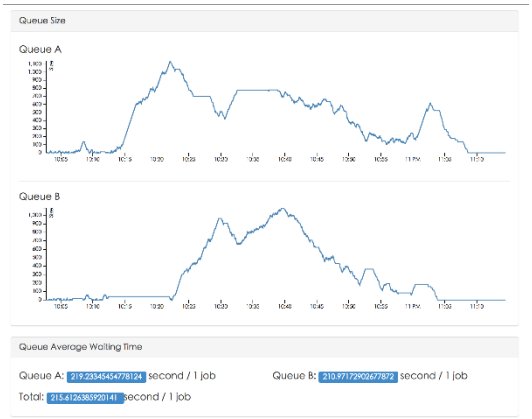
If ROUND ROBIN policy is used, since the policy static, jobs may suffer from very long queuing time. This can be observed in the queue A of the result.



Result for ROUND ROBBIN



Result for JSQ



Result for Proposed Model

In the result of the proposed model, it can be observed that the system assigns jobs to the server with shorter waiting time, while the server with heavy load temporarily receives no new incoming jobs.

5. Conclusion & Discussion

Our research covers the performance modeling of auto scale and task assignment policies in cloud environment. In the first year, we confirm our proposed model by running simulations, but the performance of the model should be fine - tuned in future work.

For our hybrid model in auto scale combining the M/G/1 queuing model and Time Series Analysis, result looks promising, but in practical application, the different arrival rates may not be easily captured by Time Series Analysis, e.g. the series is not stationary, making our model not general for practical applications. We hope we can come up with a more general model to capture the optimal time window for estimating the variance of S in M/G/1.

For our work in the task assignment problem in cloud computing, we did not greatly out perform the JSQ policy. In the heavy tailed workload, it is not an easy task to capture the workload. We consider to improve our model by finding the optimal truncate length for our markov chain, and the time window for the history data.

6. Bibliography

[1] Cloud computing: state-of-the-art and research challenges

[2] Hao-pen Chen, Shao-chong Li, A Queuing-based Model for Performance Management on Cloud, in Advanced Information Management and Service (IMS), 2010 6th International Conference on, 2010

[3] Waheed Iqbal, Matthew N. Dailey, David Carrera, Black-Box Approach to Capacity Identification for Multi-Tier Applications Hosted on Virtualized Platforms, in 2011

[4] Peter Bod´ık, Rean Griffith, Charles Sutton, Armando Fox, Michael Jordan, David Patterson. Statistical Machine Learning Makes Automatic Control Practical for
Internet Datacenters in HotCloud'09 Proceedings of the 2009 conference on Hot topics in cloud computing, 2009 International Conference on Cloud and Service Computing, 2011

[5] Poala Bermolen, Dario Rossi, Support vector regression for link load prediction in Computer Networks

[6] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, The Eucalyptus Open-source Cloud-computing System in 9th IEEE/ACM International Symposium on Cluster Computing and the Grid

[7]RUBIS: Rice University Bidding System: http://rubis.ow2.org/

[8]ALEX J. SMOLA and BERNHARD SCHO¨ LKOPF, A tutorial on support vector regression, Statistics and Computing 14: 199–222, 2004

[9]Yin-Wen Chang, Chih-Jen Lin, Feature, Ranking Using Linear SVM in JMLR: Workshop and Conference Proceedings 3: 53-64, 2008

[10] Jing Bi1, Zhiliang Zhu1, Ruixiong Tian, Qingbo Wang, Dynamic Provisioning Modeling for Virtualized Multi-tier Applications in Cloud Data Center, in 2010 IEEE 3rd International Conference on Cloud Computing, 2010

[11] Ching-Chi Lin,Jan-Jan Wu,Jeng-An Lin,Li-Chung Song,Pangfeng Liu,

Automatic Resource Scaling Based on Application Service Requirements in 2012 IEEE Fifth International Conference on Cloud Computing
[12] Oliver Nieh¨orster, Alexander Krieger, Jens Simon, Andr´e Brinkmann, Autonomic Resource Management in GRID '11 Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing
[13] Emmanuel Cecchet,Julie Marguerite,Willy Zwaenepoel, Performance and Scalability of EJB Applications in OOPSLA '02 Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications
[14]Qi Zhang, Alma Riska, Wei Sun, Evgenia Smirni, and Gianfranco Ciardo, Workload-Aware Load Balancing for Clustered Web Servers in IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, 2005
[15]Vaquero LM, Rodero-Merino L, Buyya R. Dynamically scaling applications in the cloud. SIGCOMM Computer Communication Review 2011
[16]Hussin M, Lee YC, Zomaya AY. Efficient energy management using adaptive reinforcement learning-based scheduling in large-scale distributed systems. Presented at the International Conference on Parallel Processing (ICPP), 2011.

附件四

國科會補助專題研究計畫項下出席國際學術會議心得報告

日期：103 年 2 月 25 日

計畫編號	NSC 102-2221-E-004-003, 102K216		
計畫名稱	國科會計畫, 資科系碩士在職專班鼓勵學術活動		
出國人員姓名	蔡子傑	服務機構及職稱	政治大學資訊科學系 副教授
會議時間	102 年 11 月 3 日 至 102 年 11 月 8 日	會議地點	Barcelona, Spain
會議名稱	(中文) (英文)MOBIWAC 2013 : The 11th ACM* International Symposium on Mobility Management and Wireless Access		
發表論文題目	Popularity Spray and Utility-based Forwarding Scheme with Message Priority Scheduling in Delay Tolerant Networks		

一、參加會議經過

本次會議在西班牙的巴塞隆納舉行，因為會議第一天是 workshop session。而我的論文剛好排在第一天，第一個 session 的第四篇。第一天早上主要是註冊報到時間，所以趕了一大早先去大會報到，接著就到會場了，參加了大會主席的開幕致詞。之後就開始了第一個 session。我的 session 主題是 opportunistic communications。第一篇作者講的是在大型網路機會通訊下的選擇與散播的方式，由 Texas A&M Univ 所發表，第二篇講的是臨機路由的候選人數目的選定，用一些數

學模組去計算出最佳的候選人數字。這篇較為理論的模組，是由加拿大的 U of Ottawa 發表。第三篇是從認知網路的角度去探討臨機網路的鄰近結點的發現機制，是由德國 RWTH Acehen Univ 發表。接下來就是我上場發表了，我們的主題是利用 popularity 受歡迎的程度去決定散播的方式，並透過 utility 的計算來轉傳，同時在排成上面會綜合考量優先權，以達到最佳的效能。發表完後，有人針對我們的 popularity 的想法表示肯定，另外主持人也問了一些尖銳的問題，可能就是 delivery ratio 為何那麼小，是否設定參數上面不是很完整。我的回答大概是如此，因為 delivery tolerant network 在實務運作上可能密度就不是很高，所以有可能產生如此結果。我很高興聽眾都很專心聽我報告，從問題的發問就可得到他們的回饋。

二、與會心得

接下來幾天有不同的主題，我選擇性地聽了幾場，其中第一場 keynote 是西班牙的 Prof. Arturo Azcorra，講的主題是 green wireless。Green 的議題是目前蠻夯的議題，他是從能源消耗角度切入，試著能以最小的能量傳送位元。但是要省電通常涵蓋率就較侷限。因此是個挑戰性的問題。講者將這個研究的目標，把他公式化，並且也同時有不同因素交叉考量能源的效率，進而也包含了系統的協定的能源賠償等問題。

第二場 keynote 是由加拿大的 Prof. Robert Schober 主講，講題是我們如何從無線轉傳網路的 buffer 得到多少。這個議題的方向，一直都是我所研究的重點，所以我很有興趣來學習。他是從 fading link 來切入這個議題，並加入考量 adaptive transmission schedule 的機制，來做 relay 的。但基本上，relay 雖然得到較佳的

吞吐量，但也導致延遲的增加。所以才採用了可適性的鏈結選擇，以及/或者可適性的傳輸模式選擇。除了 relay 網路選擇外，也考慮了較複雜的多天線 relay 網路，和雙向的 relay 網路。聽完之後，視野大增，真的從基本上去探討 relay 的特性，比起傳統的 relay 協定，在一些狀況下，有 buffer 的 relay 協定確能增加效能。巴塞隆納是西班牙重要城市，不僅風景優美，也深具人文氣息，在這邊參加國際會議，大都來自歐美，亞洲人幾乎是很少，歐洲人在做研究跟亞洲人甚至是美國人有大大的不同，他們會很基礎地很仔細的去思考每個環節，作的慢卻品質很計較，有點慢工出細活的感覺，加上歐洲的文化氣息，即便是科技的研究，總比我們多了一點人的味道，這是我們國內目前一直在追求的跨領域科技與人文的研究，這點我倒是有些收穫。

三、考察參觀活動(無是項活動者略)

無

四、建議

無

五、攜回資料名稱及內容

大會論文集 USB 隨身碟一個

六、其他

無

科技部補助計畫衍生研發成果推廣資料表

日期:2014/10/30

科技部補助計畫	計畫名稱: 適用於雲端環境下的動態資源服務
	計畫主持人: 蔡子傑
	計畫編號: 102-2221-E-004-003- 學門領域: 計算機網路與網際網路

無研發成果推廣資料

102 年度專題研究計畫研究成果彙整表

計畫主持人：蔡子傑	計畫編號：102-2221-E-004-003-
計畫名稱：適用於雲端環境下的動態資源服務	

成果項目			量化			單位	備註（質化說明：如數個計畫共同成果、成果列為該期刊之封面故事...等）
			實際已達成數（被接受或已發表）	預期總達成數(含實際已達成數)	本計畫實際貢獻百分比		
國內	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	1	1	100%		
		研討會論文	0	0	100%		
		專書	0	0	100%		
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（本國籍）	碩士生	4	4	100%	人次	
		博士生	1	1	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		
國外	論文著作	期刊論文	0	0	100%	篇	
		研究報告/技術報告	0	0	100%		
		研討會論文	2	1	100%		
		專書	0	0	100%	章/本	
	專利	申請中件數	0	0	100%	件	
		已獲得件數	0	0	100%		
	技術移轉	件數	0	0	100%	件	
		權利金	0	0	100%	千元	
	參與計畫人力（外國籍）	碩士生	0	0	100%	人次	
		博士生	0	0	100%		
		博士後研究員	0	0	100%		
		專任助理	0	0	100%		

其他成果 (無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。)	實際採用雲端平台進行實驗，以驗證所開發的模組，其結果經過實際雲端負載資料的驗證，實務應用價值高。

	成果項目	量化	名稱或內容性質簡述
科教處計畫加填項目	測驗工具(含質性與量性)	0	
	課程/模組	0	
	電腦及網路系統或工具	0	
	教材	0	
	舉辦之活動/競賽	0	
	研討會/工作坊	0	
	電子報、網站	0	
	計畫成果推廣之參與（閱聽）人數	0	

科技部補助專題研究計畫成果報告自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估

■達成目標

□未達成目標（請說明，以 100 字為限）

□實驗失敗

□因故實驗中斷

□其他原因

說明：

2. 研究成果在學術期刊發表或申請專利等情形：

論文：□已發表 □未發表之文稿 ■撰寫中 □無

專利：□已獲得 □申請中 ■無

技轉：□已技轉 □洽談中 ■無

其他：(以 100 字為限)

目前尚在整理中，預計將投稿至適當的國際學術會議。

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）(以 500 字為限)

雲端服務的效能,需要能 auto scale 以及 load balance 雙方面來動態配置資源,才能達到最佳的使用效能。本計畫的創新,在於利用 task workload 預測模型,來做 auto scale 的依據;另一方面,監控 VM 的負載情形,建立動態的 truncated Markov chain 模型，做為 task 指派的依據。這在雲端架構的資源配置相關的議題，有很大的應用價值。不只對雲端業者，對一般的雲端服務使用者，可以用較經濟的方式去使用雲端資源，達到最佳效率。

雲端運算服務日漸普及，最大的特色就是彈性資源、以量計費。但是，如何將資源做最佳化的規劃，並不是簡單的事情，尤其現在的軟體應用開發速度越來越快，如果要進行資源規劃，在穩定上線之前，不容易做準確的 benchmark。本研究提出可以針對動態資源需求的服務，預測系統效能的 model，並且以真實的 data(維基百科 trace)進行驗證。因此，在實務上架設雲端服務時，有其應用價值。

針對 model 方面，以往等候理論的效能分析適用於較長的時間範圍，我們嘗試使用 truncated Markov Chain, 試圖以較短的時間範圍來預測系統效能，在

實務上有更大的利用價值。