

# Maximizing Profit and Pricing in Cloud Environments

FACULTY OF  
ENGINEERING &  
INFORMATION  
TECHNOLOGIES

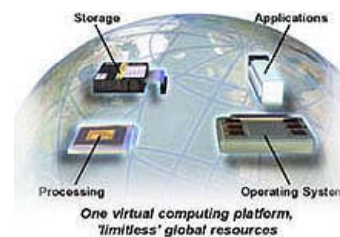
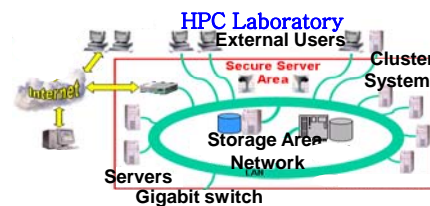
**Albert Y. Zomaya**

Chair Professor of High Performance Computing & Networking  
Centre for Distributed and High Performance Computing  
School of Information Technologies  
sydney.edu.au/it/~zomaya



## Centre for Distributed and High Performance Computing

- › A 40+ member group with more than \$12M in funding in the last five years. Current funding is from Australian Research Council, CISCO, ERICSSON, IBM, Microsoft, Sun, Smart Internet CRC, NICTA, and DSTO.
- › The Centre's mission is to establish a **streamlined research, technology exploration and advanced training program**. It will be a leading centre to undertake collaborative multi-disciplinary research in support of *distributed* and *high performance computing* and related industry to enable advances in information technology and other application domains.
- › The Centre focuses currently on several themes which build on existing strengths at Sydney University:
  - **Algorithmics and Data Mining**
  - **Cloud Computing and Green ICT**
  - **Internetworking**
  - **Service Computing**
  - **Distributed Computing Applications**





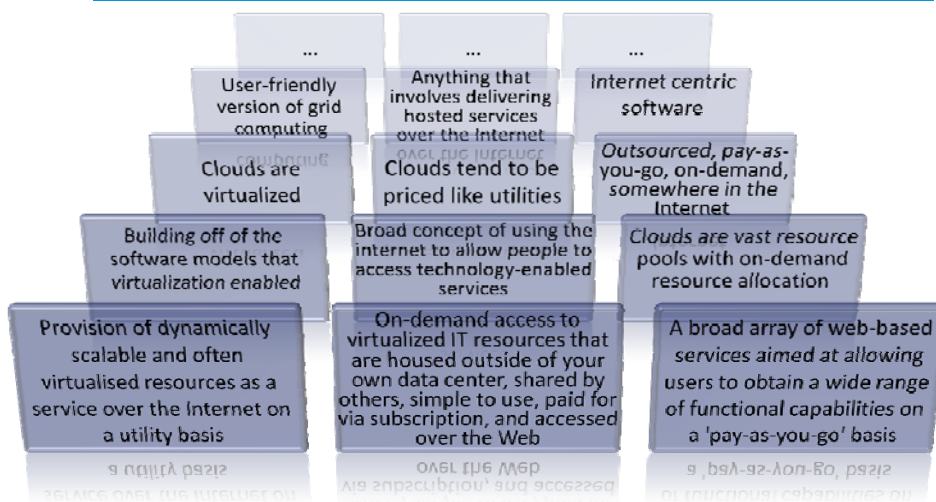
## Outline

- › **Cost Efficiency of the Cloud**
  - Cost reductions and profit increases
  - Pay-as-you-go pricing
- › **Implications of multi tenancy**
  - Resource virtualization → Resource contention
  - Current SLAs: only availability (performance?)
- › **Scheduling and resource allocation as a cost efficient solution**
  - Exploitation of application characteristics
  - Explicit consideration of user experience/satisfaction


3



## Definitions



4



# Definitions

...

User-friendly version of grid computing

Clouds are **virtualized**

Building off of the software models that **virtualization** enabled

Provision of dynamically **scalable** and often **virtualised** resources as a **service** over the Internet on a **utility basis**

...

Anything that involves delivering hosted **services** over the Internet

Clouds tend to be priced like **utilities**

Broad concept of using the Internet to allow people to access technology-enabled **services**

On-demand access to **virtualized** IT resources that are housed outside of your own data center, shared by others, simple to use, **paid for via subscription**, and accessed over the **Web**

...


Internet centric software

Outsourced, **pay-as-you-go**, on-demand, somewhere in the Internet

Clouds are **vast resource pools with on-demand** resource allocation


A broad array of **web-based services** aimed at allowing users to obtain a wide range of functional capabilities on a **'pay-as-you-go' basis**

5




# Key Terms Ordered by Frequency

- Internet
- Utility basis/pay-as-you-go
- Virtualization
- Services
- Elasticity/Scalability/Flexibility






6




# Cloud Computing

› Two key enabling technologies:

- Resource virtualization






- Utility computing
- Pay-as-you-go



Service type	Pricing
Amazon EC2 On-Demand Large	\$0.34 - \$0.40 per hour
Data transfer in	\$0.10 per GB
Data transfer out	\$0.127 - \$0.201 per GB
Amazon EBS Volumes	\$0.10 - \$0.12 per GB-month of provisioned storage \$0.10 - \$0.12 per 1 million I/O requests

7



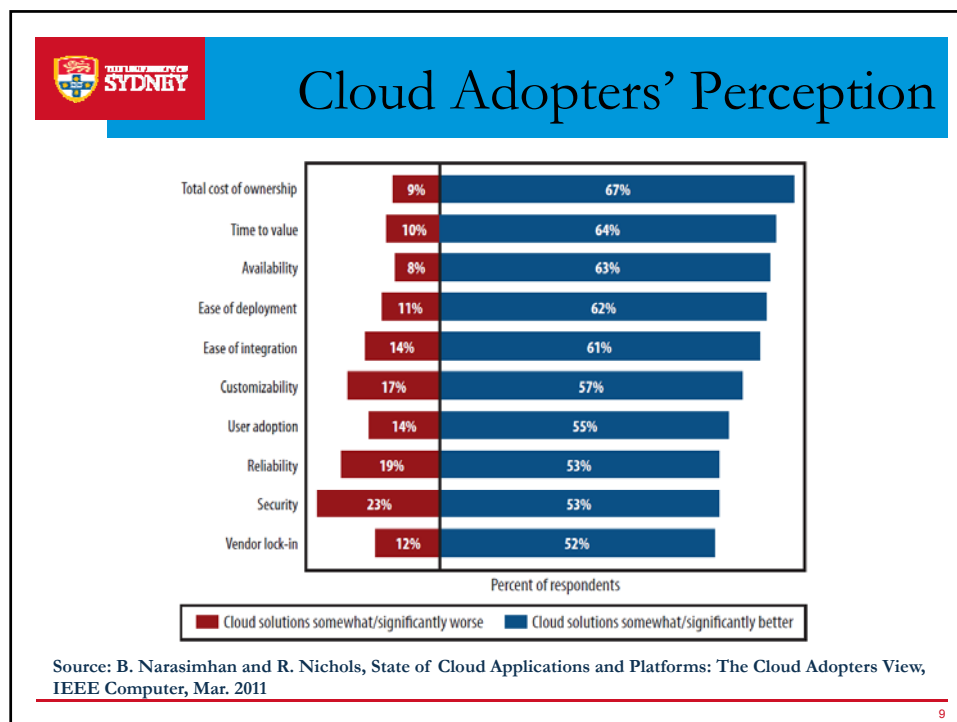
# Cloud Computing

› Motivation: Efficient resource use

- Utilization of typical data centers: below 10-30%
  - Typical enterprise DCs have a PUE of 2.0 or higher
  - DCs with best practices: 1.4 – 1.5
- Average lifetime of servers: approx. 3 years (CapEx)
- Excessive operating costs (OpEx)
  - Staffing
  - Maintenance (HW & SW)
  - Energy (both for powering and cooling)
- Offering resources as a service much enabled by virtualization technology

$$PUE = \frac{\text{Total\_data\_center\_power}}{\text{IT\_equipment\_power}}$$

8



9

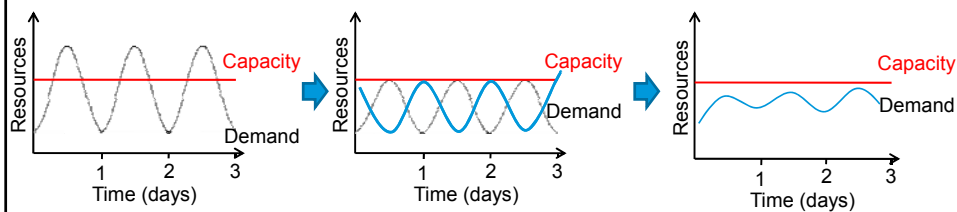
## Cost Efficiency: Provider's Perspective

- › **Cost Reductions (TCO)**
  - Economies of scale prevails
    - Cloud service providers can bring 75% - 80% cost reduction by bulk purchases
  - Efficient resource management practices
    - Utilization improvement (server consolidation)
    - Automated processes (reduction in staffing cost)
- › **Profit Increases**
  - Increase in market demand
  - Quality of service (performance)

10

**UNIVERSITY OF SYDNEY** Cost Efficiency: Provider's Perspective

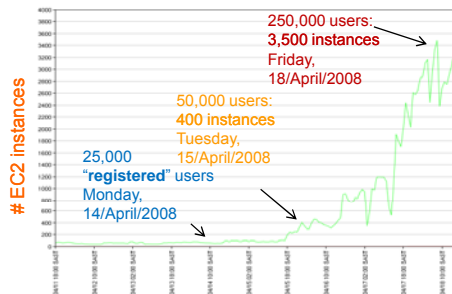
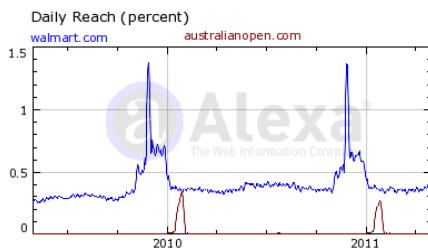
› Smoothing/flattening out workloads by effectively managing demand and capacity




**UNIVERSITY OF SYDNEY** Cost Efficiency: User's Perspective

› Elasticity

- Utilization may often be bursty





## Cost Efficiency: User's Perspective

› Elasticity


- 1 machine for 1000 hours or 1000 machines for 1 hour

### The New York Times

- In late 2007, the New York Times faced a challenge: making its entire archive of articles (11 million) available online
- 4TB of TIFF images: poorly suited to the web (multiple TIFFs for a single article)
- Solution: 24 hours of Amazon S3 and EC2 usage
- 100 EC2 instances and storage service from S3
- Cost: USD240 (i.e., 10¢ x 100 instances x 24 hours)

---

13



## Cost Efficiency: User's Perspective

› Pay-as-you-go pricing

- Cloud services may cost more than on-premises data centers
- A single server in a 50,000 node data center costs **\$112.42/month**
- Amazon large EC2 instance costs  
\$0.41/hour x 24 hrs x 30 days = **\$295.20/month**
- However, usage may not be on the 24/7 basis  
\$0.41/hour x 8 x 20 = **\$65.60/month**

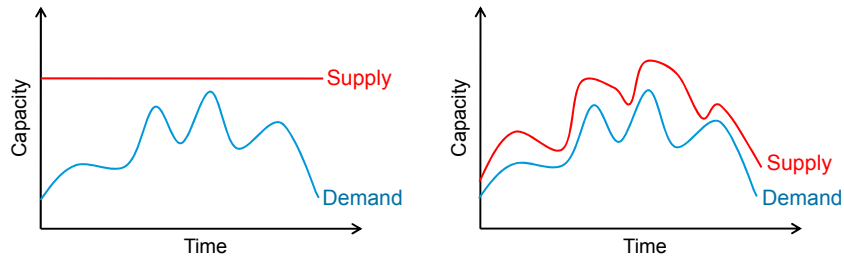
---

14



## Cost Efficiency: User's Perspective

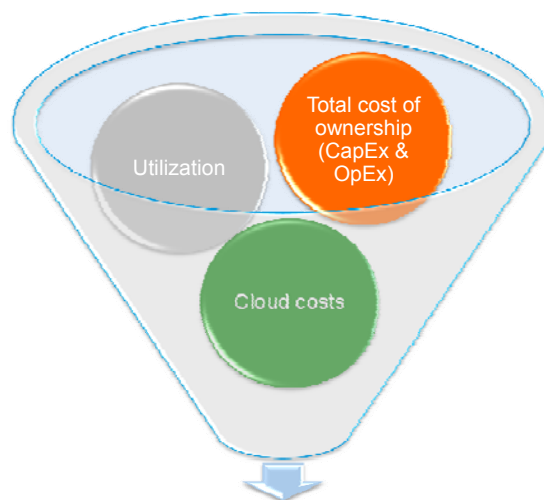
### › Dynamic provisioning



15



## Cost Efficiency: User's Perspective



**Cost efficiency solutions**

16





## Implications of Multi Tenancy

- › Limitations of resource virtualization
  - No complete resource isolation (performance interference)
  - Currently, some resources must be shared (e.g., network bandwidth, disk and last level cache)
- › Resource contention is natural
  - ‘Noisy neighbors’
- › Current SLAs only support “availability”



### Amazon EC2 Service Level Agreement

Effective Date: October 23, 2008

This Amazon EC2 Service Level Agreement ("SLA") is a policy governing the use of the Amazon Elastic Compute Cloud ("Amazon EC2") under the terms of the Amazon Web Services Customer Agreement (the "AWS Agreement") between Amazon Web Services, LLC ("AWS") and users of AWS' services ("you"). This SLA applies separately to each account using Amazon EC2. Unless otherwise provided herein, this SLA is subject to the terms of the AWS Agreement and capitalized terms will have the meaning specified in the AWS Agreement. We reserve the right to change the terms of this SLA in accordance with the AWS Agreement.

#### Service Commitment

AWS will use commercially reasonable efforts to make Amazon EC2 available with an Annual Uptime Percentage (defined below) of **at least 99.95%** during the Service Year. In the event Amazon EC2 does not meet the Annual Uptime Percentage Commitment, you will be eligible to receive a Service Credit as described below.

at least  
99.95%

17



## Implications of Multi Tenancy


- › Performance variability
  - A factor of **200 higher** than that in a non-virtualized and dedicated system\*
  - Sending a packet of data between two internal nodes within Amazon may vary from 0.3ms to 7241ms (**7 secs**)\*\*



\* Schad, J. et al. "Runtime measurements in the cloud: observing, analyzing, and reducing variance," VLDB, 3(1-2), 2010.

\*\* Has Amazon EC2 become over subscribed?, [http://alan.blog-city.com/has\\_amazon\\_ec2\\_become\\_over\\_subscribed.htm](http://alan.blog-city.com/has_amazon_ec2_become_over_subscribed.htm)

18



## Cost Efficiency: Profit Maximization

Resource utilization


User experience

Profit-driven scheduling & resource allocation

Scale

Scale

19

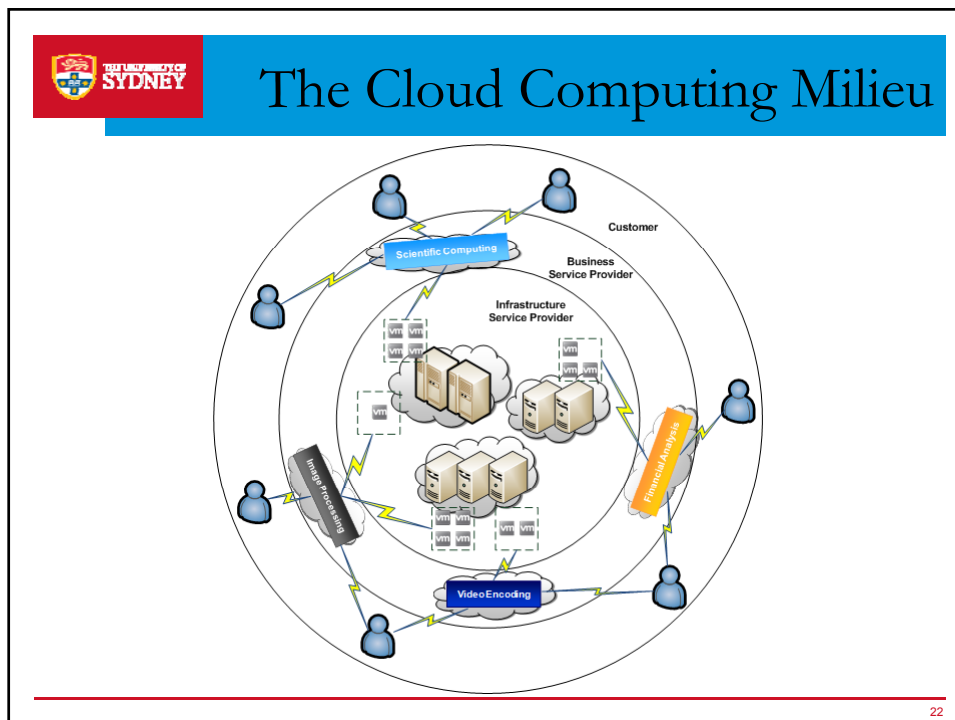
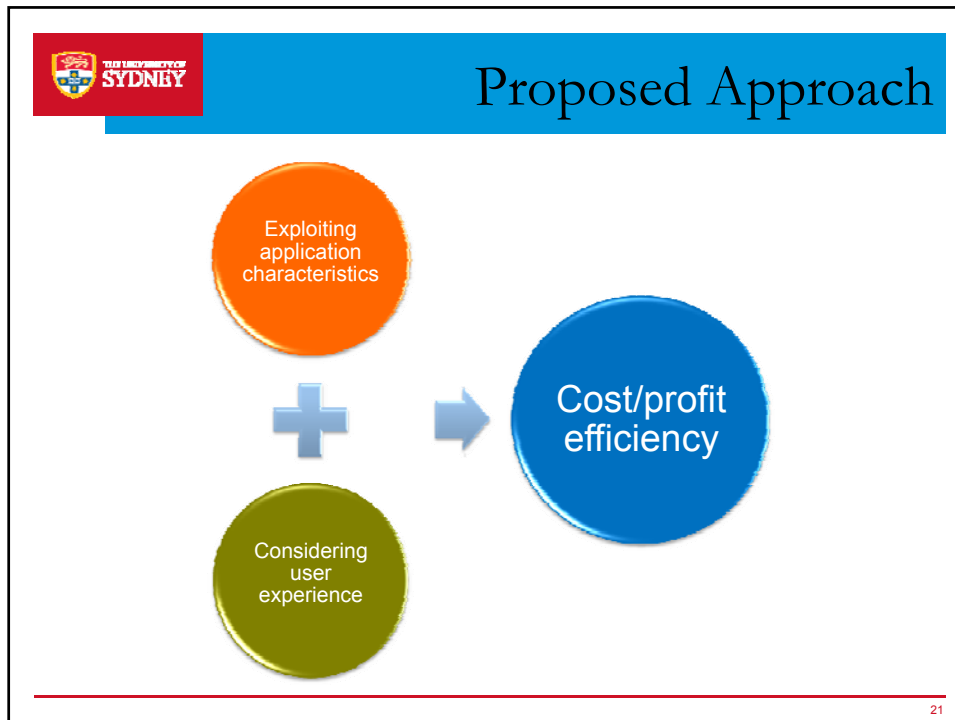


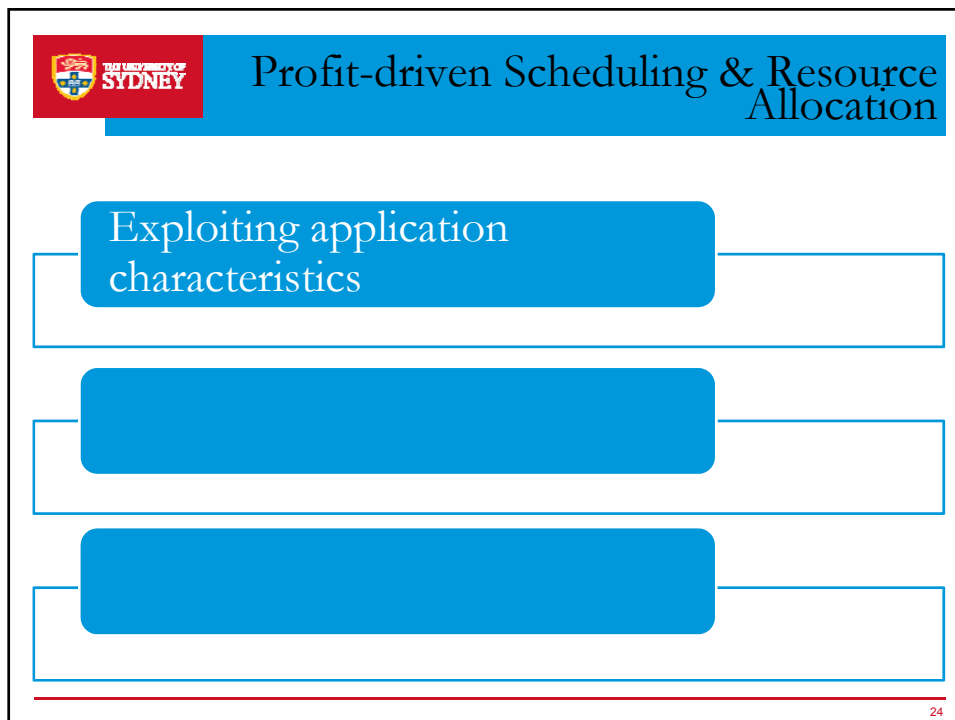
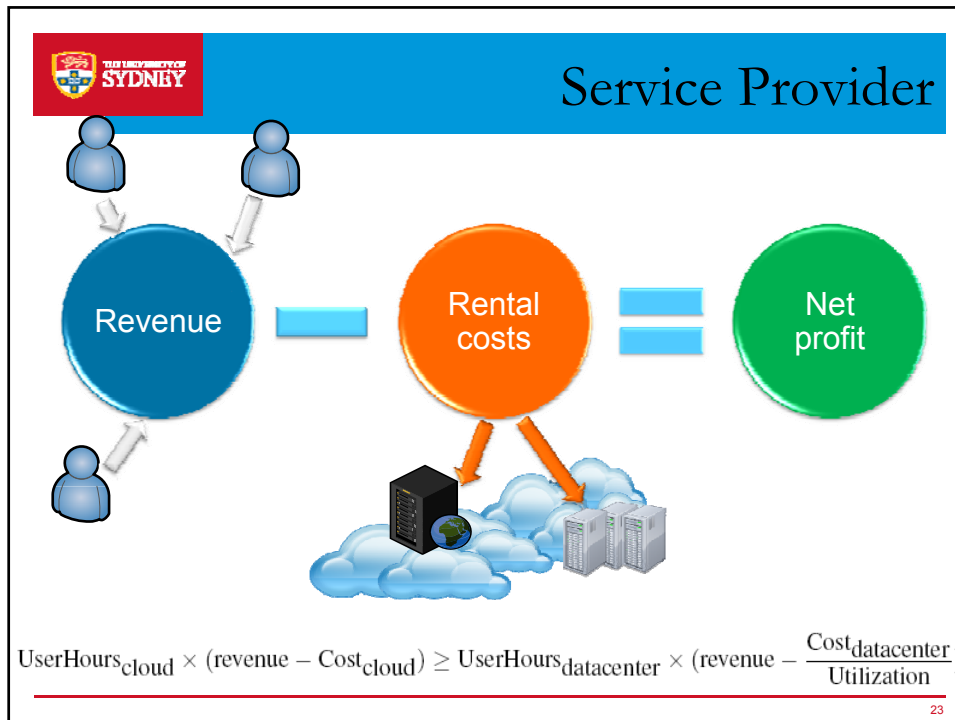
## Profit-driven Scheduling & Resource Allocation


To solve the above problem comprehensively.

- Exploiting application characteristics
- Incorporating user satisfaction into resource allocation
- Application profiling

20








# Profit Driven Scheduling


**Consumer 0**




S<sub>2</sub>  
S<sub>4</sub>

S<sub>0</sub>  
S<sub>1</sub>  
S<sub>3</sub>  
S<sub>2</sub>


**Cloud infrastructure provider**





**Service provider**

**Consumer 1**




SLA contains

- $V$ : max value for an application
- $\alpha$ : decay rate (penalty)
- $\lambda$ : request/incoming rate

Lee, Y.-C., Wang, C., Zomaya A.Y., and Zhou, B.B., 2010, "Profit-Driven Service Request Scheduling in Clouds," *10<sup>th</sup> IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'2010)*, May 17–20, pp. 15–24, Melbourne, Australia

25



# SLA Parameters

› Key SLA parameters of an application  $A_i$  are:

- $V$ : maximum value

$$V_i^{lower} = \sum_{j=1}^n w_j u \qquad V_i^{extra} = TMIN_i d_i u$$

$$V_i^{max} = V_i^{lower} + V_i^{extra}$$

- $\alpha$ : value decay rate

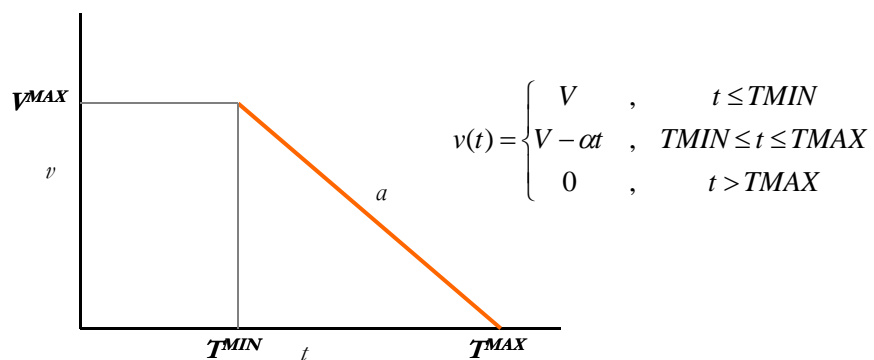
$$\alpha_i = \frac{V_i^{extra} + V_i^{lower} \left(1 - \frac{1}{(1+e)}\right)}{TMIN_i d_i}$$

26



## Profitability

- › Value (profit)  $v$  is inversely related to processing time  $t$



27



## Conflicting Objectives

- › Service providers: maximize profit (return on investment)

- Maximize revenue:

- #applications

- performance

- › Minimize resource rental costs:

- service instance utilization

- #service instances

- › Consumers: minimize expenses and meet response time requirements

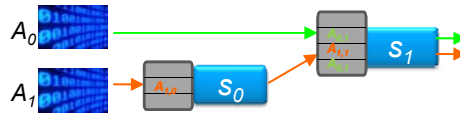
$$p^{net} = \sum_{i=1}^N v_i - \sum_{j=1}^L c \tau_j$$

28

## Exploiting Application Characteristics

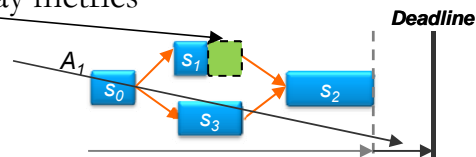
### › A pricing model based on processor-sharing

- Each of  $n$  requests receives  $1/n$  of the service's capacity
- Queuing delay is embedded in processing time



### › Allowable delay metrics

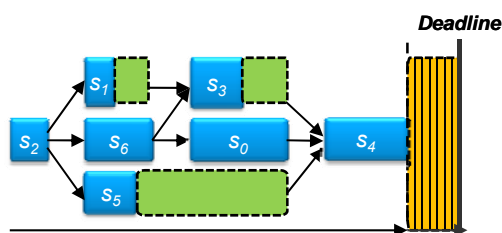
- service-wise
- application-wise



29

## Exploiting Application Characteristics

- › Application-wise AD (**AAD**): determined by consumer specified extra time allowed
- › Service-wise AD (**SAD**): determined by dependencies of services
  - Aggregative SAD (**ASAD**) = **SAD** + portion of **AAD**
  - Cumulative SAD (**CSAD**) = ASAD + ASADs of predecessors



30



## Proposed Algorithms

- › Maximum profit algorithm (**MaxProfit**)
  - focuses explicitly on net profit
  - takes into account not only the profit achievable from the current service, but also the profit from other services being processed on the same service instance
- › Maximum utilization algorithm (**MaxUtil**)
  - focuses more on utilization
  - an indirect way of reducing costs to rent resources

31



## MaxProfit

- › Time complexity:  $O(I_j)$ 
  - $I_j$ : # service instances for service  $s_j$
  - $s$ : # services being processed on the instance  $s_j$
- › Calculation of profit increase
- › Identification of an instance with largest profit increase
- › Create a new service instance
- › Service assignment


```

1. Let  $max\_pi = \emptyset$ 
2. Let  $s_{i,s} = \emptyset$ 
3. Let  $s_j^*$  = the first service to be scheduled
4. for  $\forall s_{j,k} \in I_j$  do
5.   Let  $p_i = \emptyset$ 
6.   Let  $p_i^c = \emptyset$ 
7.   for  $\forall s_{j,k}^j$  running on  $s_{j,k}$  do
8.     Let  $ap_{j,s}^i = a_j^i$  of  $s_{j,k}^j$  without considering  $s_j^*$ 
9.     Let  $ap_{j,s}^{i*} = a_j^i$  of  $s_{j,k}^j$  with considering  $s_j^*$ 
10.    Let  $c_i^{(s_{j,k}^j)} = ap_{j,s}^i + cost_{j,s}^i$ 
11.    if  $ap_{j,s}^{i*} > c_i^{(s_{j,k}^j)}$ , then // possible loss
12.      Go to Step 4
13.    Let  $p_i = p_i + c_i^{(s_{j,k}^j)} - ap_{j,s}^i$ 
14.    Let  $p_i^c = p_i^c + c_i^{(s_{j,k}^j)} - ap_{j,s}^{i*}$ 
15.    Let  $p_i^c = p_i^c + c_i^{(s_{j,k}^j)} - ap_{j,s}^i$  // include  $s_j^*$ 
16.    if  $p_i^c > p_i$  then
17.      Let  $\Delta p_i = p_i^c - p_i$ 
18.      if  $\Delta p_i > max\_pi$  then
19.        Let  $max\_pi = \Delta p_i$ 
20.        Let  $s_{i,s} = s_{j,k}$ 
21.    if  $s_{i,s} = \emptyset$  then
22.      Create a new service instance  $s_{j,new}$ 
23.    Let  $s_{i,s} = s_{j,new}$ 
24.    Assign  $s_j^*$  to  $s_{i,s}$ 

```

32






## MaxUtil

- › Time complexity:  $O(I_j^2)$ 
  - $I_j$ : # service instances for service  $s_j$
  - $s$ : # services being processed on the instance  $s_j$
  
- › Calculation of profit
  
- › Identification of an instance with lowest utilization
  
- › Create a new service instance
  
- › Service assignment

1. Let  $min\_util = 1.0$
2. Let  $s_r = \emptyset$
3. Let  $s_j$  = the first service to be scheduled
4. **for**  $\forall s_{j,k} \in I_j$  **do**
5. **for**  $\forall s_{j,k}$  **running on**  $s_{j,k}$  **do**
6. Let  $app_{j,k}^s = aff$  of  $s_{j,k}$  without considering  $s_j$
7. Let  $app_{j,k}^{s_j} = aff$  of  $s_{j,k}$  with considering  $s_j$
8. Let  $clp_{j,k}^s = app_{j,k}^s + cost_{j,k}$
9. **if**  $app_{j,k}^s > clp_{j,k}^s$  **then** // possible loss
10. | Go to Step 4
11. Let  $util_{j,k}$  = utilization of  $s_{j,k}$
12. **if**  $util_{j,k} < min\_util$  **then**
13. | Let  $min\_util = util_{j,k}$
14. | Let  $s_r = s_{j,k}$
15. **if**  $s_r = \emptyset$  **then**
16. | Create a new service instance  $s_{j,new}$
17. | Let  $s_r = s_{j,new}$
18. Assign  $s_j$  to  $s_r$


33



## Experimental Settings

- › 105,000 (21,000 for each algorithm) simulations
  - 6 different maximum widths (2 to 64)
  - 5 different numbers of services per app.U(10, 80)
  - 7 different simulation durations (between 2,000 and 30,000)
  
- › Performance metrics
  - Net profit rate
  - Utilization
  - Response rate

34




## The Experiments

- ›  $EFT^{profit}$ 
  - Greedy algorithm without using processor-sharing
  - Create a new instance whenever no service instance is readily available
- ›  $MaxProfit$  and  $MaxUtil$ 
  - Profit calculation using ASAD (i.e. Aggregative SAD (**ASAD**) = **SAD** + portion of **AAD**)
  - Profit conservative
- ›  $MaxProfit^{csad}$  and  $MaxUtil^{csad}$ 
  - Profit calculation using CSAD (i.e. Cumulative SAD (**CSAD**) = ASAD + ASADs of predecessors)
  - Utilization conscious

---

35



## Performance Evaluation


- › Overall comparative results


algorithm	net profit	Utilization	response rate
$EFT^{profit}$	31%	29%	100%
$MaxUtil$	34%	51%	70%
$MaxUtil^{csad}$	37%	54%	64%
$MaxProfit$	<b>52%</b>	50%	87%
$MaxProfit^{csad}$	40%	<b>56%</b>	79%

Dynamic instance creation captures the trade-off between utilization and profit


---

36

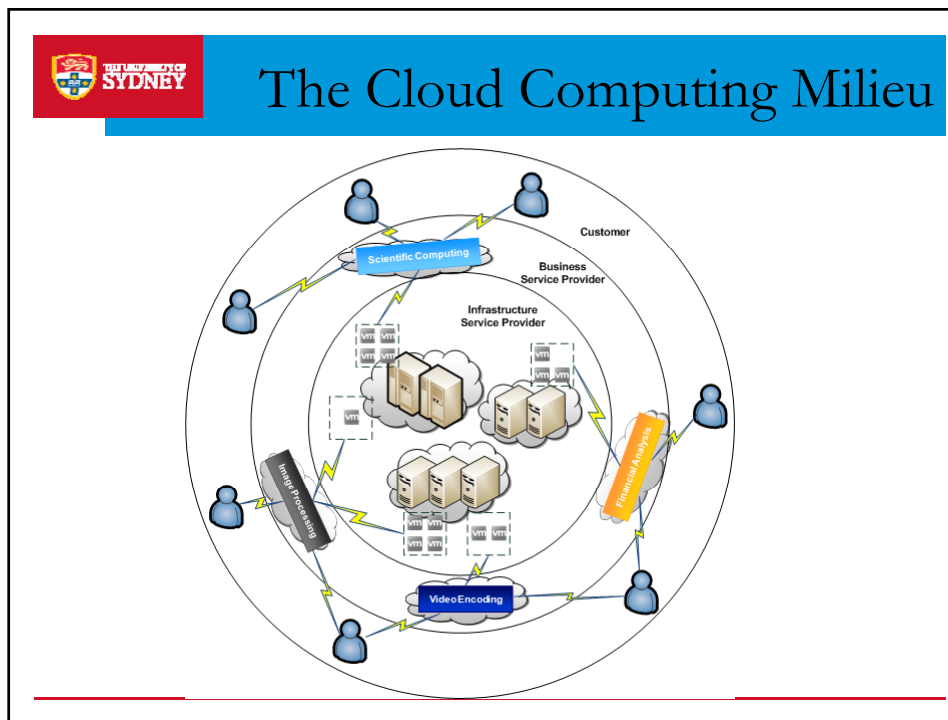
 Profit-driven Scheduling & Resource Allocation



Incorporating user satisfaction into resource allocation



37





## Problem Description

- › The service provisioning problem of the business service provider
  - How to rent VMs to build an appropriate resource set and schedule service requests
  - Business objectives
    - Maximize service profit
    - Maintain customer satisfaction
  - Constraints
    - Constraints of downstream customers
      - different customer preferences
    - Constraints of upstream infrastructure service providers
      - various types of VM instances that differ in capacity and prices
      - price fluctuations

J. Chen, C. Wang, B.B. Zhou, L. Sun, Y.C. Lee, A.Y. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud," 20<sup>th</sup> ACM HPDC, San Jose, June 8-11, 2011.

39

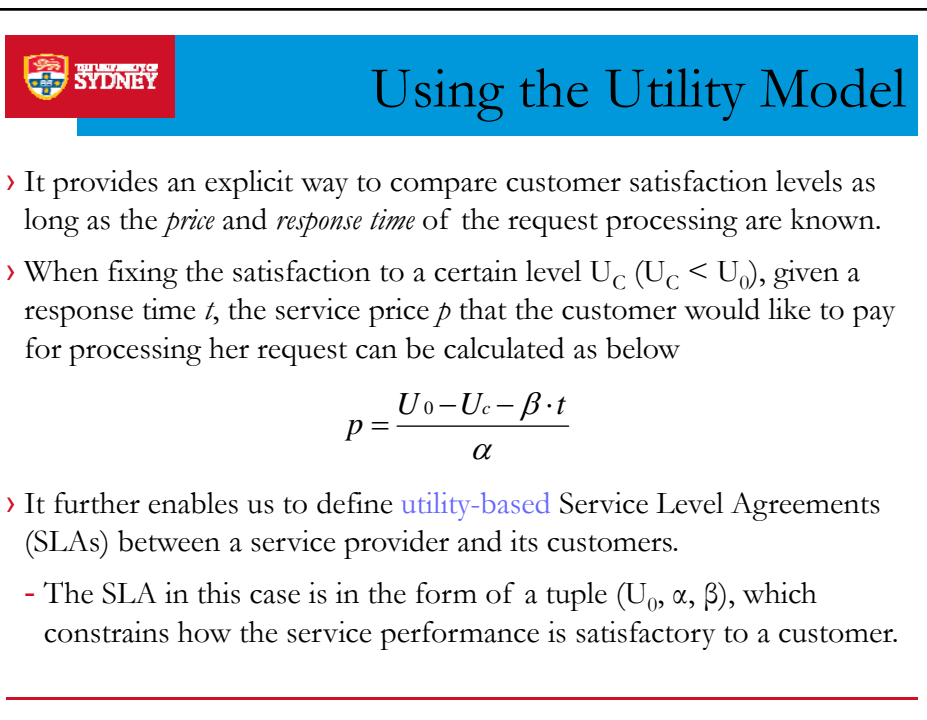
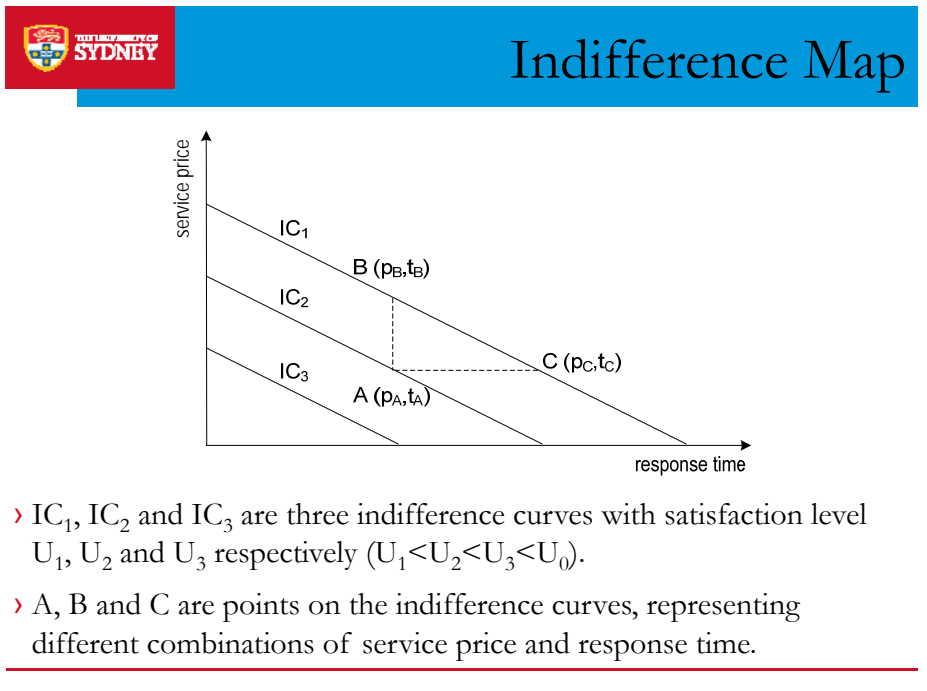


## Modeling Customer Satisfaction

- › Based on **Utility Theory** in economics, we model a customer's satisfaction (or utility) of using a service as a function of the service price  $p$  and the response time  $t$

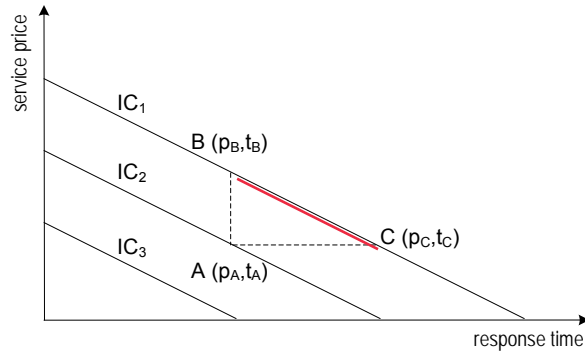
$$U(p, t) = U_0 - \alpha \cdot p - \beta \cdot t$$

- ›  $U_0$ : the maximum utility that the service delivers to the customer.
- ›  $\alpha/\beta$  (or  $\beta/\alpha$ ): known as **marginal rate of substitution** in economics, denoting the rate at which the customer is willing to give up response time (or service price) in exchange for service price (or response time) without any satisfaction change





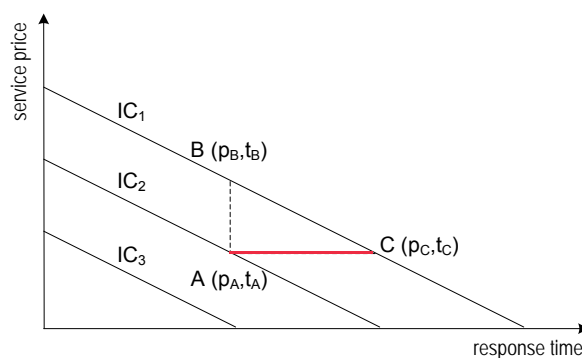
## Efficient Resource Use with Utility-based SLAs - 1



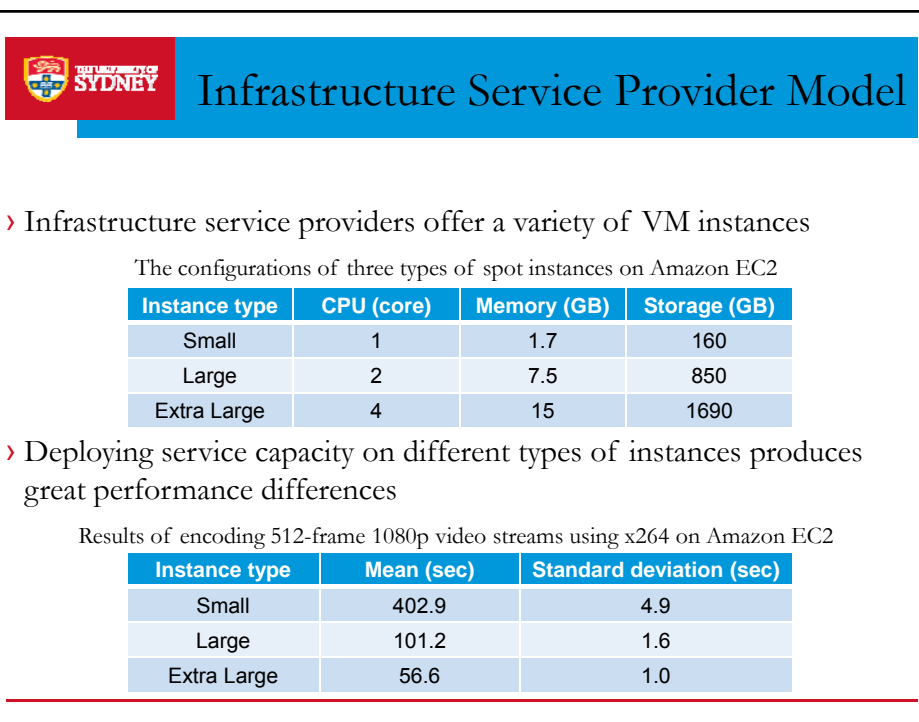
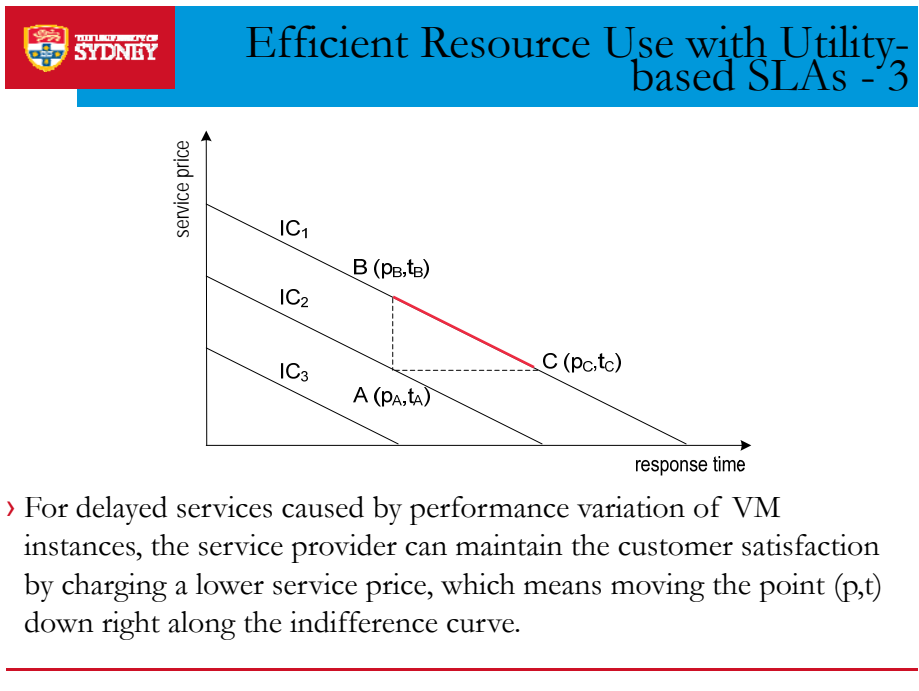
- › While maintaining a certain level of customer satisfaction, the service provider is enabled to optimize profit by reducing the response time and charging a higher service price, which means moving the point (p,t) up left along the indifference curve.



## Efficient Resource Use with Utility-based SLAs - 2



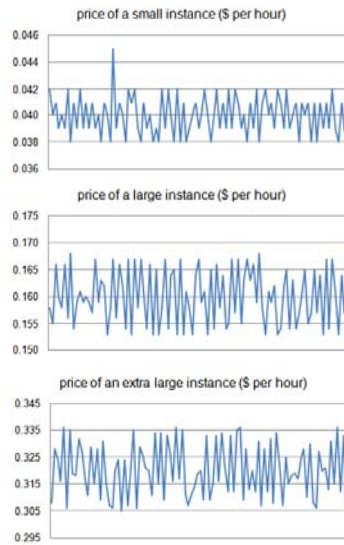
- › While keeping a profit target, the service provider can improve customer satisfaction by reducing the response time, which means moving point (p,t) left horizontally from an indifference curve to another indifference curve with higher satisfaction level.



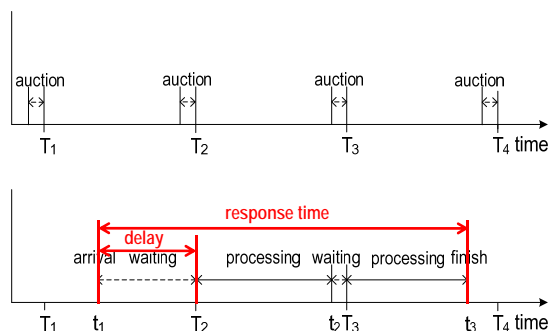


## Price Fluctuations of VM Instances

- › Prices of VM instances are determined and changed periodically according to a certain market-based mechanism, e.g., an auction
- › Price history of three instance types on Amazon EC2 (Linux, California, US, Jan 1 – Jan 15, 2011).



## The Semi-Online Scheduling Process







## Measuring Performance Difference

- › We normalize the request processing capacity of various instance types against that of a standard instance, and we call the normalized capacity *Performance Index (PI)*
- › Let  $w_0$  and  $w_k$  denote the workload that a standard instance and a type  $i_k$  instance can process in a time interval respectively, the performance index of instance type  $i_k$  is defined as

$$PI_k = \frac{w_k}{w_0}$$

- › Suppose a standard instance uses time  $t_0$  to process a request, a type  $i_k$  instance shall normally need time  $t_0/PI_k$  to process the same request.



## Portfolio Strategies for Renting Resources

- › We use  $I = \{i_1, \dots, i_m\}$  to denote the set of instance types and  $R = \{r_1, \dots, r_n\}$  to denote the requests in the waiting queue attached to the service provider.
- › For each request  $r_j \in R$ , the following variables are defined to describe its state
  - *cost<sub>j</sub>*: the accumulated cost of instance renting for processing  $r_j$ . *cost<sub>j</sub>* is updated every time interval because the cost of instance renting is charged per time interval by the infrastructure service provider.
  - *revenue<sub>j</sub>*: the revenue that a service provider expects to generate by serving  $r_j$ . The revenue is realized only when  $r_j$  is finished processing, i.e., the service provider charges the customer only when her request is finished.
  - *rpt<sub>j</sub>*: the remaining processing time (on a standard instance) of  $r_j$ . It is also updated every time interval. The initial value of *rpt<sub>j</sub>* equals the request size *size<sub>j</sub>*.



## Portfolio Strategies for Renting Resources

- › Based on the utility model and performance indexes of various instance types, we develop *portfolio strategies* for a service provider to rent an appropriate set of VM instances to serve its customers.
    - At the end of each time interval, the service provider makes decisions on what types of instances and how many instances to bid for.
    - When deciding which type of instance to choose for processing a request in an auction session, our strategies calculate expected profit (or satisfaction) for all types of VM instances, and then choose the type with maximum expected profit (or satisfaction).
- Note: Due to the price fluctuations, the instance type chosen for processing the same request may be **different** in different auction sessions.
- 



## Profit Optimization under a Satisfaction Target

Suppose the service provider aims to maintain a minimal satisfaction level  $U_{min}$  ( $U_{min} < U_0$ ).

In an auction session, for each request  $r_j$  in the queue, if it is scheduled to an instance of type  $i_k \in I$ ,

the expected remaining processing time

$$rpt_{jk} = \frac{rpt_j}{PI_k}$$

the expected accumulated cost

$$cost_{jk} = cost_j + rpt_{jk} \cdot p_k$$

the expected response time

$$resp\_time_{jk} = current\_time - arrival\_time_j + rpt_{jk}$$


---



## Profit Optimization under a Satisfaction Target

With the expected response time and  $U_{min}$ , the expected revenue is

$$revenue_{jk} = \frac{U_0 - U_{min} - \beta \cdot resp\_time_{jk}}{\alpha}$$

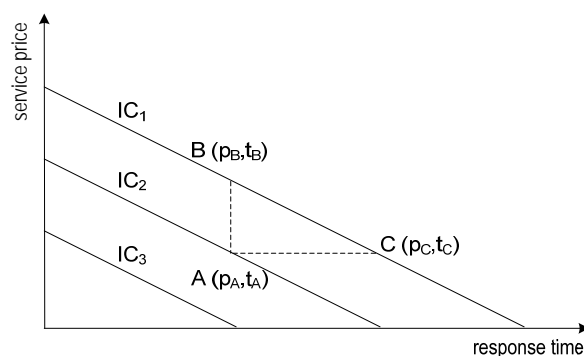
Then the expected profit is

$$profit_{jk} = revenue_{jk} - cost_{jk}$$

Finally, among all instance types, the instance type that produces the maximum expected profit is selected for processing  $r_j$



## Indifference Map



- ›  $IC_1$ ,  $IC_2$  and  $IC_3$  are three indifference curves with satisfaction level  $U_1$ ,  $U_2$  and  $U_3$  respectively ( $U_1 < U_2 < U_3 < U_0$ ).
- › A, B and C are points on the indifference curves, representing different combinations of service price and response time.



## The FirstFit-profit Algorithm

---

### Algorithm 1: FirstFit-profit algorithm

---

**Input:** market prices  $p_1, \dots, p_m, U_{min}$

```

1: for each request  $r_j \in R$  do
2:   Update  $rpt_j, cost_j$ 
3:    $profit_j = 0, instance_j = 0$ 
4:   for each instance type  $i_k \in I$  do
5:      $rpt_{jk} = \frac{rpt_j}{PI_k}$ 
6:      $cost_{jk} = cost_j + rpt_{jk} \cdot p_k$ 
7:     Calculate  $revenue_{jk}$  with  $U_{min}$  and  $resp\_time_{jk}$ 
8:      $profit_{jk} = revenue_{jk} - cost_{jk}$ 
9:     if  $profit_j < profit_{jk}$  then
10:       $profit_j = profit_{jk}$ 
11:       $instance_j = i_k$ 
12:     end if
13:   end for
14:   bid for an instance of type  $instance_j$  for processing
   request  $r_j$ 
15: end for

```

---



## Satisfaction Optimization with a Profit Bound

Suppose the service provider aims to keep a minimal unit profit  $profit_{min}$  for each request (unit profit is defined as  $profit/size$ ).

In an auction session, for each request  $r_j$  in the queue, if it is scheduled to an instance of type  $i_k \in I$ ,

the expected remaining processing time

$$rpt_{jk} = \frac{rpt_j}{PI_k}$$

the expected accumulated cost

$$cost_{jk} = cost_j + rpt_{jk} \cdot p_k$$

the expected response time

$$resp\_time_{jk} = current\_time - arrival\_time_j + rpt_{jk}$$


---



## Satisfaction Optimization with a Profit Bound

With the accumulated cost and  $profit_{min}$ , the expected revenue is

$$revenue_{jk} = profit_{min} \cdot size_j + cost_{jk}$$

Then the satisfaction is

$$satisfaction_{jk} = U_0 - \alpha \cdot revenue_{jk} - \beta \cdot resp\_time_{jk}$$

Finally, among all instance types, the instance type that produces the maximum satisfaction is selected for processing  $r_j$



## The FirstFit-satisfaction Algorithm

---

### Algorithm 2: FirstFit-satisfaction algorithm

---

**Input:** market prices  $p_1, \dots, p_m, profit_{min}$

- 1: **for** each request  $r_j \in R$  **do**
- 2:   Update  $rpt_j, cost_j$
- 3:    $satisfaction_j = 0, instance_j = 0$
- 4:   **for** each instance type  $i_k \in I$  **do**
- 5:      $rpt_{jk} = \frac{rpt_j}{PI_k}$
- 6:      $cost_{jk} = cost_j + rpt_{jk} \cdot p_k$
- 7:      $revenue_{jk} = profit_{min} \cdot size_j + cost_{jk}$
- 8:     Calculate  $satisfaction_{jk}$  with  $revenue_{jk}$  and  $resp\_time_{jk}$
- 9:     **if**  $satisfaction_j < satisfaction_{jk}$  **then**
- 10:        $satisfaction_j = satisfaction_{jk}$
- 11:        $instance_j = i_k$
- 12:     **end if**
- 13:   **end for**
- 14:   bid for an instance of type  $instance_j$  for processing request  $r_j$
- 15: **end for**

---



## Performance Evaluation

- › We evaluate our algorithms through simulation based on the performance data of different types of Amazon EC2 instances and their price history.

Parameter	Value
Number of runs	10
Number of requests	10,000
Request arrival rate $\lambda$	15 per time interval
Minimum request size	2 time intervals
Maximum request size	50 time intervals
Maximum utility $U_0$	equals request size
$\alpha/\beta$	9, 3, 2, 1, 1/2, 1/4, 1/8
Instance types	<i>small, large, extra large</i>
Instance prices	Amazon spot instances price history



## Performance Metrics

- › The following performance metrics are used to evaluate our algorithms:

- Average unit profit

$$profit = \frac{\sum_{j=1}^n \frac{revenue_j - cost_j}{size_j}}{n}$$

- Profit loss rate
- Average satisfaction

$$satisfaction = \frac{\sum_{j=1}^n satisfaction_j}{n}$$

- Satisfaction loss rate
- Number of instances
- Utilization rate

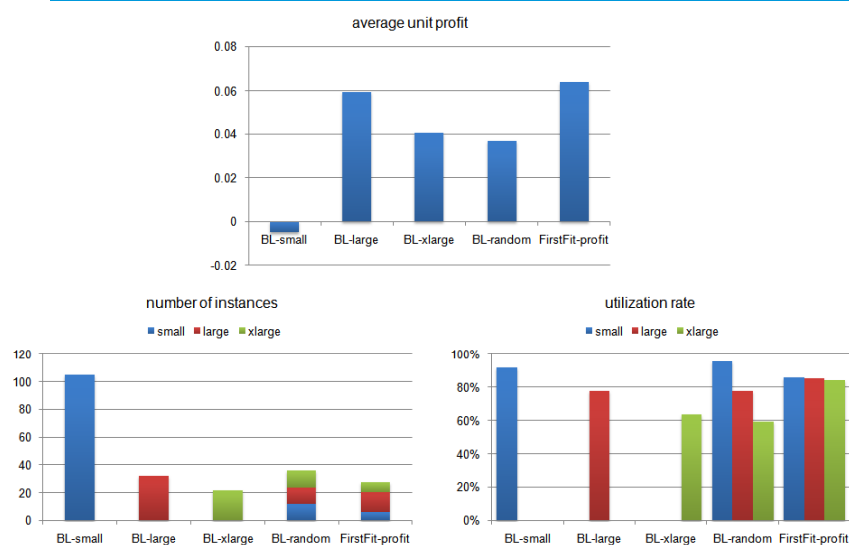


## Experiment 1

- › We first evaluate the effectiveness of using VM instances of different types for service request processing.
  - Compare with four baseline algorithms that use homogeneous instances, BL-small, BL-large, BL-xlarge and BL-random.
  - Marginal rate of substitution  $\alpha/\beta$  is randomly selected from 9, 3, 2, 1, 1/2, 1/4 and 1/8 for each request.



## Experiment 1 - results



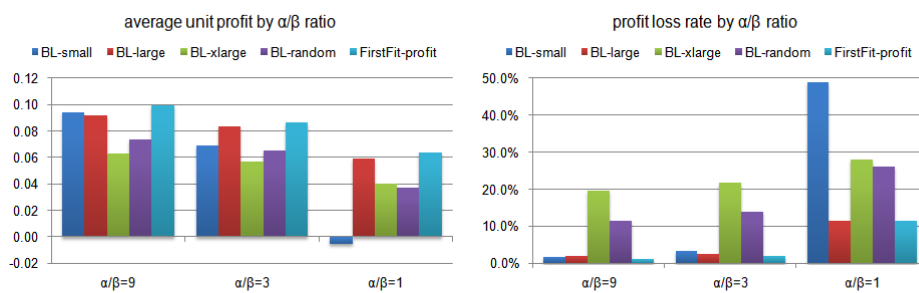


## Experiment 2

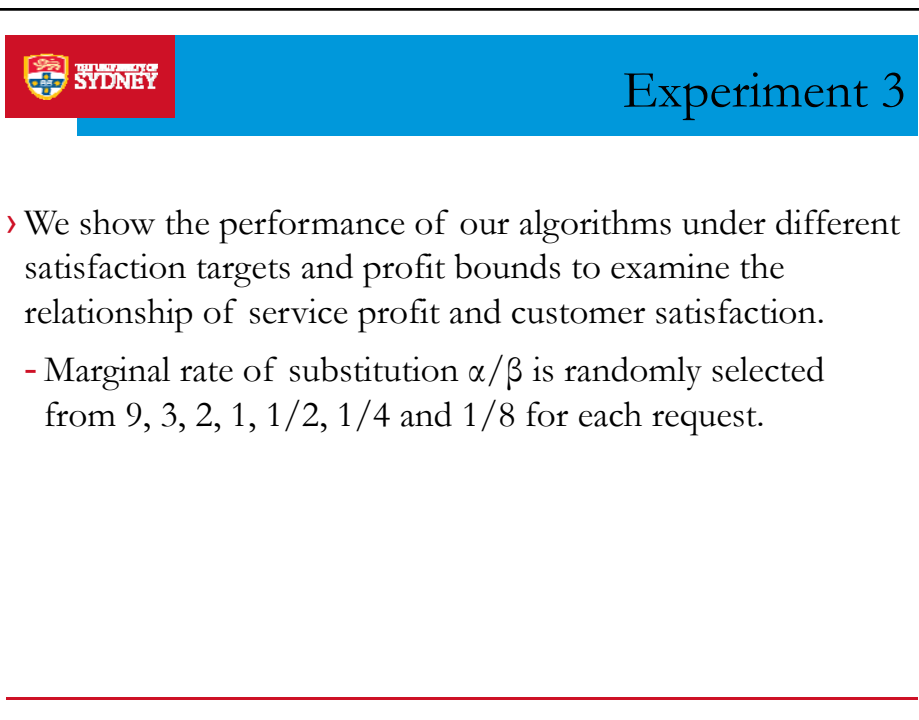
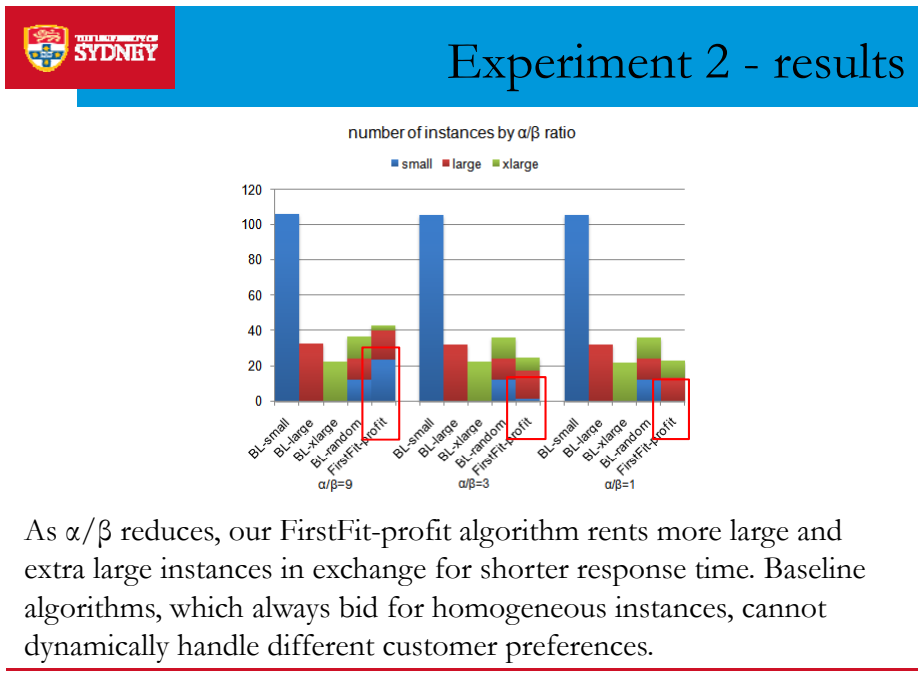
- › We then evaluate the results of our scheduling algorithms on handling different customer types defined by different  $\alpha/\beta$  ratios.
  - Marginal rate of substitution  $\alpha/\beta$  is set to 9, 3, 1 for each request respectively.



## Experiment 2 - results

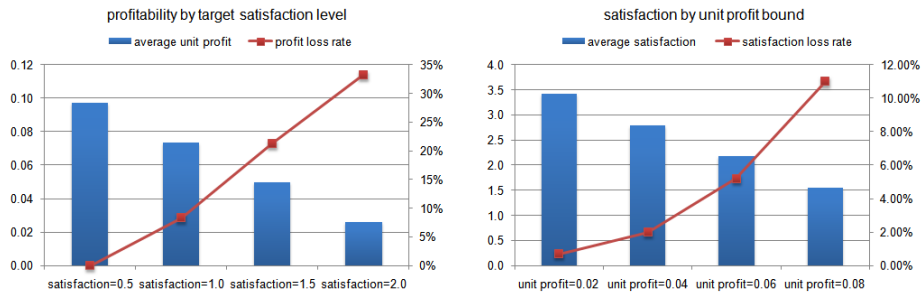








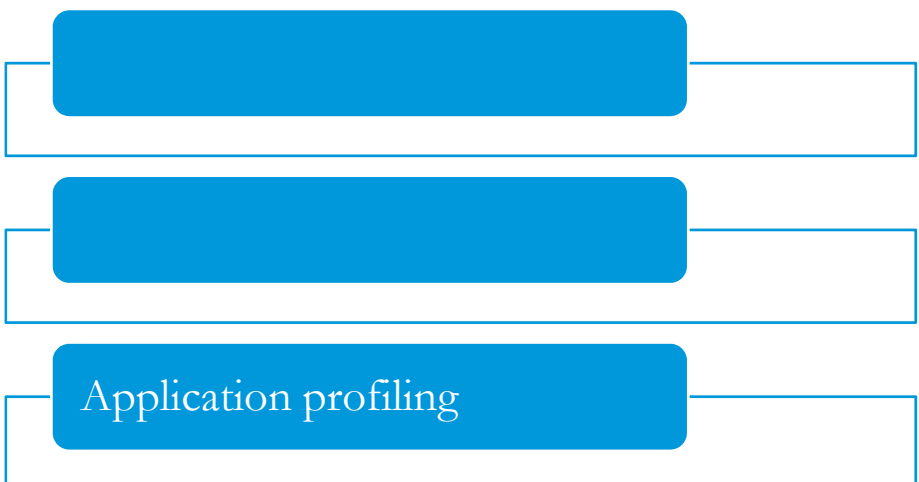
## Experiment 3 - results



It can be concluded that service profit and customer satisfaction have a negative correlation. The service provider needs to pay the cost of profit reduction for the improvement of customer satisfaction, and vice versa.



## Profit-driven Scheduling & Resource Allocation





## Objectives

- › Correlation between resource usage & performance
- › Pattern detection
- › Prediction model
- › Eventually, better VM placement/server consolidation

A. V. Do, J. Chen, C. Wang, Y. C. Lee, A. Y. Zomaya, and B. B. Zhou. Profiling Applications for Virtual Machine Placement in Clouds. In *Proceedings of the 4<sup>th</sup> International Conference on Cloud Computing (IEEE CLOUD)*, July 4-9, Washington, DC, 2011.

69



## The Approach

- › **Environment:** Xen Hypervisor 3.4
- › **Benchmark applications:** Postmark (I/O), Stream (memory), Scimark (CPU)
- › **Input** (feature metrics): No. of transactions, no. of VMs...
- › **Output** (performance metrics): I/O speed, CPU speed, power consumed...

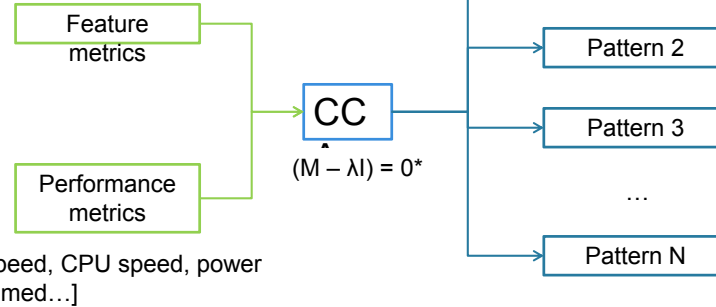
70



## The Approach (cont.)

### > Canonical Correlation Analysis

[no. of VMs, transactions, ...]



[I/O speed, CPU speed, power consumed...]

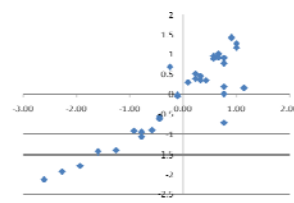
\* M: correlation matrix,  $\lambda$ : latent root, I: identity matrix

71

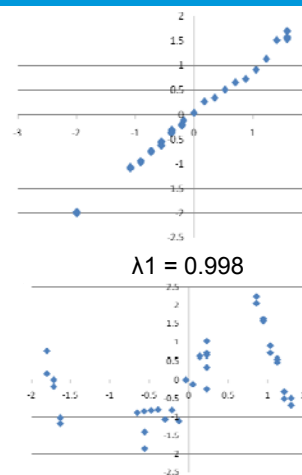


## Results & Discussion (cont.)

### > I/O intensive profiles



$\lambda_2 = 0.884$



$\lambda_3 = 0.413$

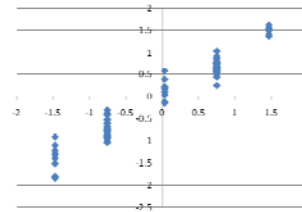
Horizontal axis: feature coefficient. Vertical axis : performance coefficient

72



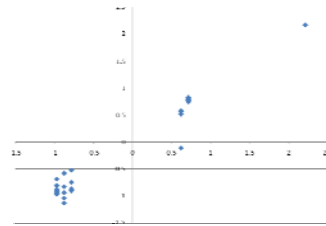
## Results & Discussion (cont.)

› Memory intensive profiles



$\lambda_1 = 0.998$

› CPU intensive profiles



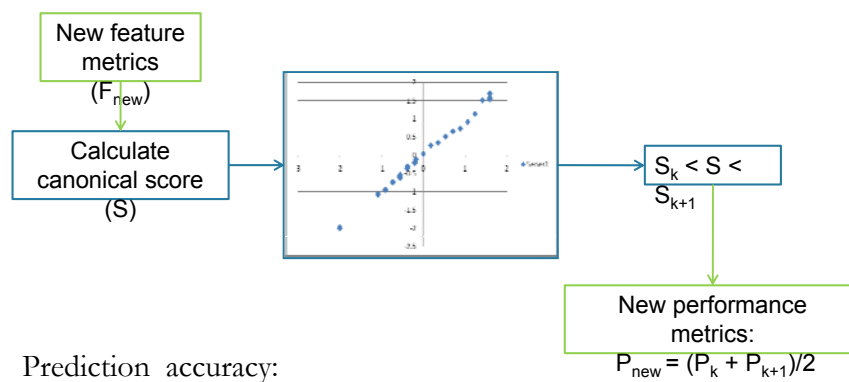
$\lambda_1 = 0.985$

73




## Results & Discussion (cont.)

› Prediction model



Prediction accuracy:  
**90.5%**

74




## Limitation & Future Work

- › Limitation
  - No access to real clouds
  - Limited-function power meter
- › Future work
  - Combination of test profiles
  - Consolidation strategies

---

75



## Open Issues

- › Measurement of actual cost savings
- › Balance between QoS and resource utilization
- › Compatibility between services offered by different service providers
- › Reliability of cloud services
- › Accountability

---

76



## Finally

- › Liberation of innovative ideas from resource constraints
- › Energy efficiency
- › Economical solution to ever increasing computing needs
- › Pricing models explicitly incorporating and effectively balancing various considerations will better leverage the proliferation of cloud computing
- › Services should be more accountable and secure

77



# Thank you



78