Cloud Pricing Models (Draft)

Zhongliu Xie, Yike Guo

Imperial College London
{zx10, yg}@doc.ic.ac.uk

Abstract

Cloud computing has kept increasingly drawing social attentions in the past decade. As explained in [37], cloud computing may be defined as “an innovative computing approach with elastic deployment of existing computing technologies under pay-per-use model”, where computing resource and service are charged based on time. Although considerable researches have been carried out in market simulation for computing resource, there is very limited work in practical pricing. This paper aims to develop a systematic theory to instruct cloud providers in developing pricing strategies to maximize profits under various circumstances, including monopoly or oligopoly, constrained or unconstrained total resources, single or multiple products, possibilities and different types of outsourcing and broking. The theory covers a wide range of fields including Cloud Computing, Utility Computing, Economics, Econometrics, Mathematical Modeling, Operations Research, Mathematical Finance, Game Theory and Industrial Organization; and it may also be applied in domains other than cloud industry. Cloud industry could be divided into three layers: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), where this paper is focused on the IaaS sector. The paper starts from an introduction to dimensions of utility pricing models and general pricing techniques, to using the basic economic concept “demand and supply” to formulate mathematical models to derive optimal pricing strategies. It also discusses a number of approaches to simulate demand and cost in a mathematical way. Finally, a spot pricing system has been developed on top of the Imperial College Cloud Platform (IC Cloud) to test the practical value of the theory.

Key Words: cloud computing, IaaS, pricing models, economics, mathematical modeling and optimization, demand simulation, cost simulation

1 Introduction

1.1 Cloud Computing

Cloud computing has kept increasingly drawing social attentions in the past decade. As explained in [37], cloud computing may be defined as “an innovative computing approach with elastic deployment of existing computing technologies under pay-per-use model”. The core of cloud industry lies on elastic provisioning of computing resources and services and pay-as-you-go mechanism, where instead of purchasing service with life-long license, cloud computing allows users to only pay the time for their use. In other words, computing resources and services are charged based on time. Cloud markets could be divided into three layers: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), by three special value-adding activities that build up the unique “production chain” of the computing industry [35]. According to [37], cloud computing could be viewed from both a computer science perspective and an economic perspective, and research could be generally divided into two parts: computing technology development and cloud economics. The first part refers to developing innovative technologies to feature clouds with more advanced computing capacity and service diversity, and the second part emphasizes on investigating the economy brought by cloud computing. This paper is focused on the economy of cloud IaaS providers and aims to provide a guidance for them to maximize economic efficiency.

1.2 Dimensions of Utility Pricing Models

General utility pricing models consist of four dimensions: product, customer, billing & payment, and price option [25]. First of all, product refers to the underlying utility such as electricity or gas. In the case of IaaS market, product is the computing capacity generated by bundled resources of CPU, memory and disk space, or in other words, virtual machine (VM). Since computing utility is charged by time, the unit of
measurement is the like of instance hour or instance day, etc. Secondly, Customers refer to the group of people who need and are also able to purchase the utility, excluding those out of physical reach. Thirdly, billing & payment involves two aspects, method and frequency. Prevailing billing method has evolved from sending by post or by telephone to e-billing, which provides a much better user experience and reduced cost, etc. Selection of frequency should suit customer needs, where common cycles include weekly billing, monthly billing, and yearly billing, etc. Finally, the most important dimension is price option. In general, it could be divided into metered and unmetered options, where the former concerns with unit charges and the latter only concerns with billing cycles. The whole paper is centered at developing optimal price options to maximize profits.

Furthermore, design of pricing models also involves three more subtle dimensions: market type, dynamism, and short/long run basis. Market type defines the type and level of competition with other suppliers, and thus determines a firm’s price making capacity. More details can be found in [37]. Dynamism refers to the concern whether market situation is dynamic or static, or in other words, whether demand and/or supply fluctuate over time. Many existing pricing models assume a static market situation, which although significantly simplifies problem solving, may not be feasible in practice. Models developed in this theory is based on a dynamic market situation. Pricing models on the short-run basis assume that a firm’s maximum supply is constant as it is impossible to make or withdraw investment in fixed assets such as data centers to increase or decrease productivity. On the contrary, each firm could easily invest or disinvest in the long run thus all these assets become variable (all costs are variable in the long run). However in reality, firms generally make decisions based on temporary situations, thus long-run models may not be useful. For that reason, the models developed in this research are based on short-run conditions.

1.3 General Pricing Techniques

Mixed Pay Schemes of Same Product

In economics, price discrimination refers to charging different customers different prices for the same product, in order to reap higher profits. Common examples include offering discounts for students, children and senior people who normally do not have income, against the major working group. It rests on the assumption that different consumers have different willingness to buy and there are barriers to prevent disadvantaged customers turning to advantaged customers regarding a purchase. An example of establishing barrier is requiring students to present student cards before offering student discount. Another example is to ask legislation to forbid reselling (large amounts of) product by law. In IaaS, the typical discriminative pricing strategy is to devise a number of price options with different billing cycles, such as pay per minute, per hour, per day, per week, per month, per year and so forth, as shown by Figure 1. Amazon’s “On-demand Instance” option [28] is a standard pay-hourly scheme. Furthermore, firms sometimes may use hybrid billing cycles. Again taking Amazon as an example, the “Reserved Instance” option is a hybrid scheme. This type of contract requires customers making a low one-time pre-payment for the contract length, after which they could enjoy a premium price on hourly basis. A typical one-year contract is a hybrid of pay-hourly and pay-yearly schemes. However, to the best of our knowledge, there is no generic discriminative pricing model that could be applied in all situations, because customer pool varies from case to case.

Figure 1: Price Discrimination
Launch of Multiple Products

In addition to price discrimination, IaaS providers usually launch a number of products to satisfy different customer needs, for example, some for CPU-intensive tasks, and some for disk-intensive tasks.

- **Extra CPU**: 4 CPU cores + 512 MB memory + 10 GB disk space
- **Extra Memory**: 1 CPU core + 4096 MB memory + 10 GB disk space
- **Extra Disk**: 1 CPU cores + 512 MB memory + 100 GB disk space

A real example is the list of Amazon EC2 products which could be found in [27]. A notable point is that these products each is associated with a separate demand curve but meanwhile are mutual substitutes, which forms a dynamic system of inter-correlated demands [37].

Spot Market

The discussion so far is centered at the standard market, in which price is settled once and for all, or at least for a considerable length of time. For that reason, firms would try their best to predict and model demand as accurate as possible. However, demand is determined by a number of factors, some of which are difficult to quantify or measure, and data sampling is also subject to bias, thereby accuracy is difficult to guarantee (see Section 5). Furthermore, real demand fluctuates over time: no matter how accurately modeled for the time being, accuracy probably diminishes in the next year, next month or even next day. Failure in properly adjusting price to cope with a changing demand cannot constantly achieve maximum profit. Also, since investment is fixed in the short run, a change of demand would probably rise the rate of idle resource.

As a remedy, Amazon introduced the so-called “Spot Instance” by the end of 2009 [26] (however the underlying pricing formulae remain to be a commercial secret). Spot instance is operated on a dynamic basis where spot price fluctuates over time w.r.t. demand-supply status. A customer bids for a spot instance with the maximal price s/he is willing to buy. Then the spot instance will be purchased and run when spot price $\leq$ bidding price, until explicitly terminated or spot price rises above bidding price. Basically, standard and spot product are the same product but launched in different markets. However, spot instance is offered with a major flaw of unreliability, i.e. it may terminate at any time. For that reason, people generally prefer standard product to spot product at the same price level, which means that the demand curve of a spot product is under that of its counterpart in the standard market. On the other hand, spot price has the potential to drop, which becomes an advantage in running non-critical tasks, thereby demand of spot product is unlikely to reach zero even charged at the same price as standard product.

There are four key advantages of spot market: First of all, spot product dynamically use spare resources left from running the standard market, thus could achieve better resource utilization. Secondly, demand curve of spot product could be directly derived based on bid statistics, instead of using complex market research which is costly and prone to errors. Moreover, techniques such as discriminative pricing strategies and launching multiple products could be further employed in the spot market. Finally, spot price could be dynamically adjusted to continuously maximize profits under changing circumstances, and to compensate the loss in the standard market, as illustrated by Figure 2 and Figure 3.

![Figure 2: Extra Profit by Spot Market](image-url)
1.4 IaaS: a System of Monopolies and/or Oligopolies

In [37], the author has explained the concepts of market types, optimal firm behaviors w.r.t. cloud markets, where the IaaS market has been defined as “a System of Monopolies and/or Oligopolies”. A brief summary is:

With a more abstract perspective, the IaaS market could be characterized as an oligopoly, because there are several companies in the market, such as Amazon, Rackspace, GoGrid, Terremark and so on (competition is far from perfect), but computing utility supplied by different companies are essentially the same. On the other hand, with a more concrete perspective, IaaS may also be classified as a monopolistic competitive market. For instance, a VM with a 2.0 GHz CPU, 1 GB memory and 40 GB disk space is slightly different from a VM with a 2.0 GHz, 2 GB memory and 100 GB disk space, as the second one is generally believed to be more valuable. A sound compromise may be treating products provided by different companies as identical if composed by the same set of resource, but as different if otherwise. In that sense, IaaS market could be divided into a number of sub-markets by products, each is either a monopoly or an oligopoly. These sub-markets are inter-related, since products are mutually substitutable.

According to [18], in a monopolistic market, the only firm dominates the market and becomes a pure price maker. To maximize profits, it should produce at the level that marginal revenue equals to marginal cost, and charge a price that is consistent with demand curve, as shown by Figure 4. In contrast, each firm in oligopoly has some price making capacity but meanwhile act as a price taker. This is because all firms in the market collectively share one demand curve. Each firm can choose a quantity to supply, but market price is determined by aggregate market supply of all firms w.r.t. demand. Any firm that sets a higher price shall lose complete market share as all its customers shall switch to alternative suppliers.
1.5 Purpose and Structure of the Paper

This paper aims to develop a systematic theory to instruct cloud providers in developing pricing strategies to maximize profits under various circumstances, including monopoly or oligopoly, constrained or unconstrained total resources, single or multiple products, possibilities and different types of outsourcing and broking. The theory covers a wide range of fields including Cloud Computing, Utility Computing, Economics, Econometrics, Mathematical Modeling, Operations Research, Mathematical Finance, Game Theory and Industrial Organization. The principle of the theory is to formulate some optimization problem with mathematical modeling of demand (demand function) and supply constraint (cost function and production capacity), then solve it using relevant techniques. For that reason, this theory could also be applied in other markets to solve pricing problems.

Section 2 discusses the key theories borrowed from other disciplines as well as related work in computing market simulation and pricing. Section 3 will be focused on the monopolistic market whereas Section 4 shall analyze the situation of oligopoly including the case of a broker role. Then Section 5 and 6 will respectively discuss the simulation of demand and cost. After that, Section 7 will discuss the experiments on a spot pricing system, which is part of the Imperial College Cloud Platform (IC Cloud). Finally, Section 8 will draw a conclusion and point out some future directions.

2 Related Work

2.1 Economics, Econometrics, Mathematical Finance, Operations Research

Economics refers to the studies of efficiently allocating scarce resources to society, and its fundamental concept in economics is “demand and supply” [18]. Demand is the need to consume with a willingness to purchase, whereas supply is the capacity to provide the product or service. Both concepts are normally expressed by the whole group of customers or producers rather than by individuals. In general, demand will be higher when price is lower, and vice versa (unless it is a Giffen good), whereas supply is on the opposite. A mathematical function that represents the relationship between demand and price is called a demand function, expressed as \( Q = f(P) \), and demand curve is plotted by its inverse, i.e. \( P = f^{-1}(Q) \). Cost correlated to quantity of supply, expressed as \( C = g(Q) \). The pricing models developed in this paper shall be centered at the use of (inverse) demand and cost functions, as well as total productivity.

Econometrics is a field that specializes in measurement of economic phenomena with statistical and mathematical modeling, hypothesis testing as well as forecasting [10]. Regression analysis is the orthodox approach to analyze economic data. Researchers in this field have developed a number of regression models and methods, along with potential problems, detection tests and solutions. This makes regression analysis very practical and reliable. Also, econometrics directly link to economics, therefore both theoretical support and related work are more accessible. Econometric approach will be used in demand simulation, which is discussed in Section 5.

Mathematical Finance is a field that applies computational techniques in analyzing the phenomena of financial economics. Typical research focuses include decision making, hedging, valuation and risk management. The approaches of portfolio optimization and geometric Brownian motion [16] are borrowed in this research.

Studies of Operations Research (OR) primarily focus on formulating mathematical models and developing mathematical optimization solutions. In particular, it is highly related to Mathematical Programming, which includes Linear Programming, Integer Programming, Quadratic Programming. The proposed pricing theory heavily relies on mathematical optimization techniques in solving profit maximization problems, and majority of the models developed in this paper use a Quadratic Programming approach.

2.2 Economics of Computing

Numerous attempts have been made on simulating a market-based environment of computing capacity to optimize resource allocation. The earliest research in maximizing computing utility in a specified domain using economic incentives could date back to 1966, when Greenberger discussed ways and methods of achieving fairness in queuing systems [9]. More frequently referenced is the auction system developed by Sutherland in 1968 [31]. In this auction system, users bid for exclusive access to PDP-1, which was the first computer created by Digital Equipment Corporation in 1960. Different users hold different levels of budget, which are refilled every day and residuals could not be brought forward to the next day. All users are placed in an equal position and auction is purely based on price of bid. Following that, two auction-based resource allocation systems (i.e. Bellagio and Mirage) are suggested in [1], along with the deployment experiences on the worldwide PlanetLab research network and on a shared sensor network testbed. The high-level design centers at a resource allocation mechanism and a virtual currency policy. Users in these
systems will be distributed with virtual currencies, and resource use is associated with a price. There is a specially developed currency refilling policy, which penalizes/rewards users based on usage or lack of usage during peak time. Another good example is the market-based P2P backup system developed in [29]. In this system, users trade their resources such as disk space and bandwidth in exchange for a reliable backup service provided by other users in a local P2P network. It uses an implicit policy where each type of resource is associated with a hidden price. Cost and earning each time are directly converted to the level of backup service usable before prompted to user. This conforms to the Hidden Market Design pointed out in [30], which suggests developing some ‘Hidden Market User Interface’ that wraps around an actual market to hide its existence from unsophisticated users in the interaction, to avoid ‘decision overload’ and reduce cognitive cost. This is similar to the Adapter design pattern in object-oriented design.

In Grid computing market, authors in [11, 12] have developed a number of advanced models using a multi-agent simulation environment to analyze behaviors. Agent-based simulation could be used to model distributed computing as a process of interaction, where agents cooperate and compete with other agents with respect to their own ‘economic purposes’ [34]. It usually involves a large number of interacting and decision-making processes conducted by agents, which are difficult to be modeled otherwise. Later on, they extend the models to simulate a Markovian futures market for computing utility and demonstrated the potential of computing utility as a financial derivative [19]. According to [22], there are generally two reasons of establishing a Grid market:

1. To maximize utilization in a specified domain
2. To maximize profit for companies that possess excessive computing utility

Amazon EC2 is probably the best example as its original purpose was to better utilize its excessive computing resources left from Amazon Web Services.

All of the work have simulated a market environment to efficiently allocate computing resources, which demonstrates the feasibility of establishing a market for computing capacity and treating computing as a utility. The potential of financial activities has also been pointed out.

2.3 Other Pricing Models

There is very limited practical work that could be used for firms to develop pricing strategies. Researchers have attempted to use machine-learning based approaches. Authors in [17] have presented a pricing model using Genetic Algorithms that takes in some simple pricing functions and generates new pricing functions, which are hopefully better in the sense of offering suitable prices. The sense of “suitable” refers to more stable and quicker in matching sellers and buyers and settling a deal. However, it does not give a clue on what price a firm should charge for using one unit of computing utility in order to maximize profit. In fact, electricity pricing theories could provide considerable inspirations in pricing computing utility. Authors in [5] have suggested two approaches of electricity pricing: the average approach and the marginal approach. The former refers to equally dividing and distributing cost to customers in order to cover some sunk cost. Whereas the latter refers to charging customers for the cost that s/he actually brings to the company on marginal basis. The average approach could be used in the standard market as price is fixed, whereas the marginal approach could be used in the spot market as spot price is dynamically adjusted based on cost (and also demand).

3 Monopolistic Pricing Models

For better understanding, the analysis starts with a monopolistic market. Supposing the monopolist is called CloudNexus and is a dedicated IaaS provider who may launch a single product or multiple products. Complexity increases dramatically from the case of single product to the case of multiple products.

3.1 Single-Product Profit Maximization

Supposing there is only one product: its demand curve is expressed by \( P = f(Q) \), where \( P \) stands for price and \( Q \) stands for quantity; and its cost function is \( TC = g(Q) \), where \( TC \) stands for Total Cost, the sum of variable cost \( VC \) and fixed cost \( FC \). Due to the assumption of short run basis, fixed cost is not considered, \( TC = VC \). For simplicity, let us choose the linear model \( P = \alpha \cdot Q + \beta \) and the quantity-proportional cost function \( TC = \gamma \cdot Q \). According to short-run conditions, there is a constraint on \( Q \) such that \( 0 \leq Q \leq K \) (where \( K \) is a known constant). There are three solutions from different theories discussed in [36], which are respectively Naive Marginal Revenue-Cost Solution, Elasticity Calculus Solution and Mathematical Optimization Solution. Naive Marginal Revenue-Cost Solution is only applicable when there
is no constraint in total productivity, and Elasticity Calculus Solution rests on the assumption that there is zero or minuscule cost. Only Mathematical Optimization Solution serves as a generic pricing solution, therefore is discussed here. In mathematical terms, the problem is defined as below:

\[
\text{Maximise } \Pi = TR - TC = P \cdot Q - g(Q) \\
\text{Subject to } 0 \leq Q \leq K
\]

where \( K \) is a given constant

Since \( f(Q) = \alpha \cdot Q + \beta \) and \( g(Q) = \gamma \cdot Q \), the first order derivative of profit w.r.t. quantity is

\[
\frac{d\Pi}{dQ} = \left[ f'(Q) \cdot Q + f(Q) \right] - \gamma = (\alpha \cdot Q + \alpha \cdot Q + \beta) - \gamma = 2\alpha \cdot Q + \beta - \gamma
\]

Recall that demand curve is downward-sloping curve, therefore \( \alpha < 0 \). Also, when \( P = 0 \), i.e. product is free, there must be a (high) positive demand, thus \( \beta > 0 \). Since \( \alpha < 0 \) and \( Q > 0 \), profit \( \Pi = \alpha \cdot Q^2 + \beta \cdot Q - \gamma \cdot Q \) will be non-negative only if \( \beta \geq \gamma - \alpha \cdot Q \), from which we could derive \( \beta \geq \gamma \) must be satisfied. Then since \( \frac{d\Pi}{dQ} = 2\alpha \cdot Q + \beta - \gamma \), \( \alpha < 0 \) and \( \beta \geq \gamma \), we could obtain that \( \frac{d\Pi}{dQ} \geq 0 \) iff \( Q \leq \frac{\gamma - \beta}{2\alpha} \). In other words, as long as \( Q < \frac{\gamma - \beta}{2\alpha} \), providing an additional unit of computing utility will increase total profit, i.e. marginal profit is positive. Now we look at the constraint \( Q \leq K \). This constraint divides the optimal condition into two cases:

1. If \( K < \frac{\gamma - \beta}{2\alpha} \), \( Q \) will be bounded by \( K \), where the optimal condition is that \( Q^* = K \), \( P^* = \alpha \cdot K + \beta \) and \( \Pi_{\text{max}} = \alpha \cdot K^2 + (\beta - \gamma) \cdot K \).

2. If \( K \geq \frac{\gamma - \beta}{2\alpha} \), \( Q \) will not be bounded by \( K \), thus the optimal condition is \( Q^* = \frac{\gamma - \beta}{2\alpha} \), \( P^* = \frac{\gamma - \beta}{2} \) and \( \Pi_{\text{max}} = \frac{\gamma - \beta^2}{4\alpha} - \gamma \cdot \frac{\beta - \gamma}{2\alpha} = \frac{\beta - \gamma}{4\alpha} \).

### 3.2 Multiple-Product Profit Maximization

Supposing CloudNexus now launches a number of products to attract more customers, for example,

- Small: 1 CPU core + 512 MB memory + 10 GB disk space
- Medium: 1 CPU core + 1024 MB memory + 20 GB disk space
- Large: 2 CPU cores + 1024 MB memory + 30 GB disk space

As mentioned above, these products are all different, thus each is associated with a demand curve. Suppose the inverse demand functions are respectively

\[
P_S = \alpha_1 \cdot Q_S + \beta_1 \quad P_M = \alpha_2 \cdot Q_M + \beta_2 \quad P_L = \alpha_3 \cdot Q_L + \beta_3
\]

Since all the three products use CPU, memory and disk space, CloudNexus may choose to build a single resource base to better manipulate production and more efficiently utilize resources. For example, the resource base may contain 100 virtual CPU cores, 100 GB memory and 999 GB disk space. Also, let us assume the total cost function is \( TC = \gamma_1 \cdot Q_S + \gamma_2 \cdot Q_M + \gamma_3 \cdot Q_L \) for simplicity. Taking all these factors into consideration, what is the optimal price for each product? This sounds incredibly awful at the first glance. Fortunately, it can be divided into two simpler scenarios to help readers understand the notion of Portfolio Optimization [16] at a slower pace.

#### 3.2.1 Scenario I: Unconstrained Total Resources

Supposing CloudNexus has unlimited resources thus can provide any quantity of any product as wanted. In a mathematical form, the problem is described as below:

\[
\text{Maximise } \Pi = TR - TC = (P_S \cdot Q_S + P_M \cdot Q_M + P_L \cdot Q_L) - (\gamma_1 \cdot Q_S + \gamma_2 \cdot Q_M + \gamma_3 \cdot Q_L)
\]

Since there is no constraint on total resources, the quantity of each product is unbounded. The solution above could also be used to solve this problem by coping with different products separately using partial differentiation, and divides the problem into three sub-problems.

\[
\frac{\partial \Pi}{\partial Q_S} = \frac{\partial}{\partial Q_S} [ (P_S \cdot Q_S + P_M \cdot Q_M + P_L \cdot Q_L) - (\gamma_1 \cdot Q_S + \gamma_2 \cdot Q_M + \gamma_3 \cdot Q_L)] = \frac{\partial (P_S \cdot Q_S - \gamma_1 \cdot Q_S)}{\partial Q_S}
\]
Since \( \partial_t \) till another \( \partial_t \) otherwise. In mathematical terms, the problem could be expressed by:

\[
\alpha_i \cdot \frac{\partial^2 (Q^2 + \beta_1 \cdot Q - \gamma_i \cdot Q)}{\partial Q^2} = 2 \alpha_1 \cdot Q + \beta_1 - \gamma_i
\]

Similarly, \( \frac{\partial \Pi}{\partial Q_M} = 2 \alpha_2 \cdot Q + \beta_2 - \gamma_2 \) \( \frac{\partial \Pi}{\partial Q_L} = 2 \alpha_3 \cdot Q + \beta_3 - \gamma_3 \)

Since \( \alpha_1, \alpha_2, \alpha_3 \leq 0 \) and \( \beta_1 \geq \gamma_1, \beta_2 \geq \gamma_2, \beta_3 \geq \gamma_3 \) (by similar reasoning as above),

\[
\frac{\partial \Pi}{\partial Q_S} \geq 0 \quad \text{iff} \quad Q_S \leq \frac{\gamma - \beta_1}{2 \alpha_1}
\]

\[
\frac{\partial \Pi}{\partial Q_M} \geq 0 \quad \text{iff} \quad Q_M \leq \frac{\gamma - \beta_2}{2 \alpha_2}
\]

\[
\frac{\partial \Pi}{\partial Q_L} \geq 0 \quad \text{iff} \quad Q_L \leq \frac{\gamma - \beta_3}{2 \alpha_3}
\]

For that reason,

\[
Q^*_S = \frac{\gamma - \beta_1}{2 \alpha_1} \quad Q^*_M = \frac{\gamma - \beta_2}{2 \alpha_2} \quad Q^*_L = \frac{\gamma - \beta_3}{2 \alpha_3}
\]

and \( P^*_S = \frac{\gamma + \beta_1}{2} \quad P^*_M = \frac{\gamma + \beta_2}{2} \quad P^*_L = \frac{\gamma + \beta_3}{2} \)

\[
\Pi_{\text{max}} = -\frac{(\beta_1 - \gamma_1)^2}{4 \alpha_1} - \frac{(\beta_2 - \gamma_2)^2}{4 \alpha_2} - \frac{(\beta_3 - \gamma_3)^2}{4 \alpha_3}
\]

**Extension to N Products**

When there are \( N \) products, there will be \( N \) partial derivatives \( \frac{\partial \Pi}{\partial Q_i} \) (\( i \in N \)) that need to be worked out and optimized. There is no need for a more detailed explanation.

### 3.2.2 Scenario II: Constrained Total Resources

In this scenario, we encounter a situation that total resources are limited. A constraint is binding if there are not enough resources to optimize all the product, and is not binding (indifferent from Scenario I) if otherwise. In mathematical terms, the problem could be expressed by:

Maximise \( \Pi = (P_S \cdot Q_S + P_M \cdot Q_M + P_L \cdot Q_L) - (\gamma_1 \cdot Q_S + \gamma_2 \cdot Q_M + \gamma_3 \cdot Q_L) \)

Subject to

- \( \text{cpu} \cdot Q_1 + \text{cpu}_2 \cdot Q_2 + \text{cpu}_3 \cdot Q_3 \leq T_{\text{cpu}} \)
- \( \text{memory}_1 \cdot Q_1 + \text{memory}_2 \cdot Q_2 + \text{memory}_3 \cdot Q_3 \leq T_{\text{memory}} \)
- \( \text{disk}_1 \cdot Q_1 + \text{disk}_2 \cdot Q_2 + \text{disk}_3 \cdot Q_3 \leq T_{\text{disk}} \)
- \( Q_1, Q_2, Q_3 \geq 0 \)

Since \( \frac{\partial \Pi}{\partial Q_i} \) changes as \( P_i \) or \( Q_i \) changes, and since all the products share a constrained resource base, there may be many different combinations. To the best of our knowledge, there is no analytical solution. However, one may use an algorithm that starts with adjusting the price (quantity) of the product with the highest \( \frac{\partial \Pi}{\partial Q_i} \), ranks higher, then iterate the procedure time after time until it converges. This moves to the field of **Mathematical Programming**. In fact, the problem above is a standard Quadratic Programming (QP) problem, which by definition, is a problem of optimizing a quadratic function of several variables subject to linear equality and/or inequality constraints on these variables [13]. There are a number of algorithms existed to solve QP problems, including Active Set algorithm, Interior Point algorithm and so forth [13, 14]. Also, there are many off-shelf QP solvers in the market. A typical example is the Quadprog function in the Matlab Optimization Toolbox, and users could use it straightaway. Taking the example mentioned above:

CPU: \( \text{cpu}_1 = 1, \text{cpu}_2 = 1, \text{cpu}_3 = 2, T_{\text{cpu}} = 100 \)

Memory (in GB): \( \text{memory}_1 = 0.5, \text{memory}_2 = 1, \text{memory}_3 = 1, T_{\text{memory}} = 100 \)

Disk (in GB): \( \text{disk}_1 = 10, \text{disk}_2 = 20, \text{disk}_3 = 30, T_{\text{disk}} = 999 \)
Now, the only missing data are inverse demand functions and cost functions for the three products. Supposing by whatever means, CloudNexus obtains the following:

\[ P_1 = -0.005 \cdot Q_1 + 2.1 \quad P_2 = -0.006 \cdot Q_2 + 2.3 \quad P_3 = -0.006 \cdot Q_3 + 2.5 \]

\[ TC = 0.01 \cdot (Q_1 + Q_2 + Q_3) \]

Using Matlab Quadprog, we could easily obtain the optimum \( <Q^*_1 = 95.7462, Q^*_2 = 2.0769, Q^*_3 = 0> \), and with the demand functions, we could obtain the optimal prices are \( <P^*_1 = 1.6212, P^*_2 = 2.2875, P^*_3 = 2.5> \), thus the total profit is

\[ \Pi = P_1 \cdot Q_1 + P_2 \cdot Q_2 + P_3 \cdot Q_3 - \gamma \cdot (Q_1 + Q_2 + Q_3) = 159.0031 \]

Interpretation: CloudNexus should set prices to 1.6212 \$/per Small instance hour, 2.2875 \$/per Medium instance hour and 2.5 \$/per Large instance hour, in which case it shall sell 95.7462 Small instances, 2.0769 Medium instances and 0 Large instances, making a profit of 159.0031 \$/per hour. A notable point is that the sales here primarily rely on Small instance. This is because resources are very limited and Small instance is most profitable. If CloudNexus increases the size of its resource base to 500 virtual CPU cores, 500 GB memory and 5000 GB disk space, the result would be \( <Q^*_1 = 169.8592, Q^*_2 = 125.5986, Q^*_3 = 26.3146> \), \( <P^*_1 = 1.2507, P^*_2 = 1.5464, P^*_3 = 2.3421> \) and the total profit becomes 468.3019 \$/per hour. Expanding the resource base may always increase profit until constraint become not binding.

**Extension to N Products**

Extension to N products is very obvious thus not detailed.

### 3.3 Application on Standard and Spot Markets

In practice, the firm should conduct some market research in the first place, then based on demand conditions and resource constraints, optimize the pricing strategy for standard product. After operations in the standard market, there are may be spare resources, which could be used to launch spot product. The residual resources, demand functions of spot product and cost functions again could formulate an optimization problem, and be solved accordingly. A notable point is that since demand changes over time but price of standard product cannot be changed, sales in the standard market may grow up and down (and may not be optimal), which changes the residual resource structure. In that sense, spot price is dynamically adjusted to maximize profits under changing circumstances, and to compensate the (temporary) loss in the standard market.

### 4 Oligopolistic Pricing Models

#### 4.1 Standard Market: Game Theory and Industrial Organization

In oligopoly, all firms collectively share a single demand curve. If cooperative, they could negotiate and reach an agreement to collectively output at the monopoly optimal level, so that the seller group will receive the highest benefits. However, upon such an agreement, each supplier has an incentive to defect, because market price is higher than cost and providing an extra quantity could bring extra profit. As a consequence, all firms shall choose to defect, reaching the worst outcome. The worst come is called a Nash Equilibrium, which is the core concept of Game Theory. General Game Theory involves four typical features: 1. cooperative or non-cooperative game: whether players play jointly or separately; 2. static or dynamic: whether game is once-off or sequentially with post information available; complete or incomplete: whether players know the payoffs of others; 3. perfect or imperfect: whether players know all the historical data or only partly [4]. Industrial Organization is a field that relies on Game Theory as one of its principal theories. Models developed in this field are normally considered to be more practical, for instance, the Cournot Quantity Setting Models, Bertrand Price Setting Models [4]. Industrial Organization makes more emphasis on industry structure, and one of the most important theories is 'Structure, Conduct and Performance', which investigates industry monopoly level and firm performances [15]. Research in both fields could be used as basics for oligopolistic pricing models for standard product. A notable point is that different firm may perceive different demand curves, thus pricing under this condition would be much more complicated. This paper is focused on oligopolistic spot pricing and leave the standard market problem as a future direction.
4.2 Spot Pricing Models in Oligopoly

Despite the complication in the standard market, pricing spot product is still similar to that in monopoly, because demand curve is observable by bid statistics. It will also be analyzed on a scenario-by-scenario basis in the following. Furthermore, a notable point in oligopoly is that outsourcing may be possible. Outsourcing means a firm purchases from other providers to support its own provisioning of service. In reality, there would be a number of technical problems regarding outsourcing such as how to seamlessly combine remote resources with local resources and how to cope with contingencies when remote source suddenly become unavailable. The mathematical models here abstract away all the technical complexities, and simply assume outsourcing can be done at any time and CloudNexus uses remote resources as if they are local. In addition, outsourcing can be divided into two forms:

- **Instance-based Outsourcing**: outsources end-products from other providers. Although price of same product remains the same, others may provide different products. CloudNexus could choose from a list of products to cover its resource shortage in the most cost-effective way.

- **Resource-based Outsourcing**: instead of purchasing end products, CloudNexus may negotiate with other providers regarding a direct purchase of desired resources, at probably a much lower price. In addition, as more and more providers use this strategy, a provider-level resource market may come into existence. This Provider Intranet may only be accessible to providers, helping them operate the spot market.

4.2.1 Scenario I: Unconstrained Resources

With unlimited resources, it is not much different from the monopolistic model, since there is no need to outsource anything from any party.

4.2.2 Scenario II: Constrained Resources without Outsourcing

If total resources are constrained and outsourcing is not an option, the situation is also indifferent from that in monopoly.

4.2.3 Scenario III: Constrained Resources with Resource-based Outsourcing

As mentioned above, the provider intranet offers a standard resource market for a firm to directly outsource desirable resources to cover its temporary shortage. For that reason, a set of standard market prices for CPU, memory and disk space shall probably come into existence at last, which, for instance, may be respectively $P_c$ per virtual CPU core, $P_m$ per GB of memory and $P_d$ per GB of disk space, and the quantities are denoted $<Q_c, Q_m, Q_d>$. If values of $<P_c, P_m, P_d>$ and total resources are given, demand and cost functions are respectively $P_i = \alpha_i \cdot Q_i + \beta_i$ and $Cost_i = \gamma_i \cdot Q_i$ for $i = 1, 2, ..., N$, then the problem becomes

**Maximise** $\Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma_i \cdot Q_i - (P_c \cdot Q_c + P_m \cdot Q_m + P_d \cdot Q_d)$

**Subject to**

$\sum_{i=1}^{N} cpu_i \cdot Q_i - Q_c \leq T_{cpu}$

$\sum_{i=1}^{N} memory_i \cdot Q_i - Q_m \leq T_{memory}$

$\sum_{i=1}^{N} disk_i \cdot Q_i - Q_d \leq T_{disk}$

$Q_i \geq 0$ for $i = 1, 2, ..., N$

Again, $<Q_1^*, Q_2^*, ..., Q_N^*, Q_c^*, Q_m^*, Q_d^*>$ can be worked out by standard QP Solvers like Matlab Quadprog without difficulty, then $<P_1^*, P_2^*, ..., P_N^*>$ can be found by corresponding demand functions, and thus the total profit.
4.2.4 Scenario IV: Constrained Resources with Instance-based Outsourcing

For instance-based outsourcing, CloudNexus should conduct a market research to find all other IaaS providers and their products in the first hand, then take into account all these products to build up the set of outsourcing options (duplicates only considered once). If CloudNexus offers \( N \) products, with quantities and prices respectively denoted \( Q_1, Q_2, \ldots, Q_N \) and \( P_1, P_2, \ldots, P_N \); and there are \( M \) options to outsource, with quantities and prices respectively denoted \( Q_{o1}, Q_{o2}, \ldots, Q_{oM} \) and \( P_{o1}, P_{o2}, \ldots, P_{oM} \) (outsourcing prices are all known), the problem becomes

\[
\text{Maximise} \quad \Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma_i \cdot Q_i - \sum_{j=1}^{M} P_{oj} \cdot Q_{oj}
\]

Subject to

\[
\sum_{i=1}^{N} \text{cpu}_i \cdot Q_i - \sum_{j=1}^{M} \text{cpu}_{oj} \cdot Q_{oj} \leq T_{cpu}
\]

\[
\sum_{i=1}^{N} \text{memory}_i \cdot Q_i - \sum_{j=1}^{M} \text{memory}_{oj} \cdot Q_{oj} \leq T_{memory}
\]

\[
\sum_{i=1}^{N} \text{disk}_i \cdot Q_i - \sum_{j=1}^{M} \text{disk}_{oj} \cdot Q_{oj} \leq T_{disk}
\]

\[
Q_{oj}, Q_i \geq 0 \quad \text{for } i = 1, 2, \ldots, N \text{ and } j = 1, 2, \ldots, M
\]

Using QP solvers, one could easily obtain the values of \( <Q_1^*, Q_2^*, \ldots, Q_N^*> \) and \( <Q_{o1}^*, Q_{o2}^*, \ldots, Q_{oM}^*> \), then \( <P_1^*, P_2^*, \ldots, P_N^*> \) and total profit.

4.3 Pricing Models for Broker

As mentioned before, discriminative pricing reaps higher profits. The precondition is the existence of barriers to prevent disadvantaged people from purchasing at a premium price and prevent people offered premium price to re-sell to those with disadvantages. However, barriers are not always easy to establish, most of the time there would be chance for a premium price holder to resell product to others, making a profit free of charge. In finance, this kind of behavior is called arbitrage. The person makes profits by reaping price gap and links customers to service providers is called a broker. An example in IaaS could be that a person issues a three-year contract with Amazon to reserve a large number of VM instances. After prepayment, he is offered a premium price on hourly basis. Then he could build a new company reselling the VMs at a higher price (but not higher than Amazon) to end customers who have disadvantages in purchasing from Amazon directly. Alternatively, if the broker is capable, s/he could purchase instances from others, disassemble them and build his/her own resource base, develop and offer new products, and create a new market. The following models could instruct a broker on how to develop pricing strategies with different options of outsourcing to maximize profits.

4.3.1 Scenario I: Broking with Instance-based Outsourcing

Suppose CloudNexus now is a broker who does not have any resource but can make instance-based outsourcing. To maximize profits, the first thing to do is find out all the possible sources (not only Amazon) to purchase products. Again, for example there are \( M \) choices to outsource, each is associated with a known unique price. Then, CloudNexus find the demand and cost functions of its own products (supposing there are \( N \) products), and formulate a standard QP problem as below:

\[
\text{Maximise} \quad \Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma_i \cdot Q_i - \sum_{j=1}^{M} P_{oj} \cdot Q_{oj}
\]

Subject to

\[
\sum_{i=1}^{N} \text{cpu}_i \cdot Q_i - \sum_{j=1}^{M} \text{cpu}_{oj} \cdot Q_{oj} = 0
\]

\[
\sum_{i=1}^{N} \text{memory}_i \cdot Q_i - \sum_{j=1}^{M} \text{memory}_{oj} \cdot Q_{oj} = 0
\]

\[
\sum_{i=1}^{N} \text{disk}_i \cdot Q_i - \sum_{j=1}^{M} \text{disk}_{oj} \cdot Q_{oj} = 0
\]
The reason for constraints to be equalities rather than inequalities is because a broker does not possess any resource and everything is outsourced from others on demand. A remarkable point is that all the outsourcing prices \( P_{o_j} \) are known constants. In this particular case, an analytical solution could be developed, which borrows ideas from Markowitz Model in portfolio optimization theories\[16\]. The solution is derived by the following steps: First of all, form a Lagrangian Function as below:

\[
L = \left[ \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \bar{P}_i \cdot Q_i - \sum_{j=1}^{M} P_{o_j} \cdot Q_{o_j} \right] + \lambda \cdot \left[ \sum_{i=1}^{N} \text{cpu}_i \cdot Q_i - \sum_{j=1}^{M} \text{cpu}_{o_j} \cdot Q_{o_j} \right] + \mu \cdot \left[ \sum_{i=1}^{N} \text{memory}_i \cdot Q_i - \sum_{j=1}^{M} \text{memory}_{o_j} \cdot Q_{o_j} \right] + \omega \cdot \left[ \sum_{i=1}^{N} \text{disk}_i \cdot Q_i - \sum_{j=1}^{M} \text{disk}_{o_j} \cdot Q_{o_j} \right]
\]

Then find the partial derivative for each \( Q_i \), each \( Q_{o_j} \), \( \lambda \), \( \mu \) and \( \omega \), and set them equal to zero:

\[
\frac{\partial L}{\partial Q_i} = 2a_i \cdot Q_i + k_i + \lambda \cdot \text{cpu}_i + \mu \cdot \text{memory}_i + \omega \cdot \text{disk}_i = 0
\]

\[
\text{where \quad } k_i = \beta_i - \gamma_i
\]

\[
\frac{\partial L}{\partial Q_{o_j}} = -P_{o_j} + \lambda \cdot \text{cpu}_{o_j} + \mu \cdot \text{memory}_{o_j} + \omega \cdot \text{disk}_{o_j} = 0
\]

\[
\frac{\partial L}{\partial \lambda} = \sum_{i=1}^{N} \text{cpu}_i \cdot Q_i - \sum_{j=1}^{M} \text{cpu}_{o_j} \cdot Q_{o_j} = 0
\]

\[
\frac{\partial L}{\partial \mu} = \sum_{i=1}^{N} \text{memory}_i \cdot Q_i - \sum_{j=1}^{M} \text{memory}_{o_j} \cdot Q_{o_j} = 0
\]

\[
\frac{\partial L}{\partial \omega} = \sum_{i=1}^{N} \text{disk}_i \cdot Q_i - \sum_{j=1}^{M} \text{disk}_{o_j} \cdot Q_{o_j} = 0
\]

Solving the system of equations, one could obtain optimal \(<Q_1^*, Q_2^*, ..., Q_N^*>\) and \(<Q_{o1}^*, Q_{o2}^*, ..., Q_{oM}^*>\), and derive \(<P_1^*, P_2^*, ..., P_N^*>\) by corresponding inverse demand functions. Interpretation: CloudNexus should outsource \(<O_1^*, O_2^*, ..., O_M^*>\) of the \( M \) products from others at the prices of \(<P_1^*, P_2^*, ..., P_M^*>\) to build its temporary resource base, in order to supply \(<Q_1^*, Q_2^*, ..., Q_N^*>\) of its \( N \) products to customers at prices of \(<P_1^*, P_2^*, ..., P_N^*>\), making a profit of \( \Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \bar{P}_i \cdot Q_i - \sum_{j=1}^{M} K_j \cdot O_j \).

### 4.3.2 Scenario II: Broking with Resource-based Outsourcing

If a provider intranet exists and if CloudNexus could access to this intranet, the firm will be able to outsource raw resources at a much cheaper price. In that case, CloudNexus could formulate the following model

Maximise \( \Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \bar{P}_i \cdot Q_i - (P_c \cdot Q_c + P_m \cdot Q_m + P_d \cdot Q_d) \)

Subject to \( \sum_{i=1}^{N} \text{cpu}_i \cdot Q_i - Q_c = 0 \)

\( \sum_{i=1}^{N} \text{memory}_i \cdot Q_i - Q_m = 0 \)

\( \sum_{i=1}^{N} \text{disk}_i \cdot Q_i - Q_d = 0 \)

A similar solution could be used to work out the optimum, and one could obtain \(<Q_1^*, Q_2^*, ..., Q_N^*>\), \(<Q_c^*, Q_m^*, Q_d^*>\), and then \(<P_1^*, P_2^*, ..., P_N^*>\) as well as total profit.

## 5 Demand Simulation Models

In general, demand is determined by a number of factors, including price, customer average income level, customer taste and expectation of future trends, and price of substitute, price of complement as well as total number of buyers in the market, etc. Among them, price is the principal factor a firm uses to adjust demand (along demand curve). The rest of the factors collectively play the role of an environment. There are a number of demand simulation approaches to model real situations as precise as possible.
5.1 Standard Market: Econometric Approach with Regression Analysis

Although exact demand curve could be in any shape, practical analysis usually tend to use a smooth curve to reduce complexity. In fact, when there is a large customer pool, demand curve normally becomes smoother. The basic model is a straight line, represented by a linear demand function $Q = \alpha \cdot P + b$ (or the inverse $P = \alpha \cdot Q + \beta$). With adequate data, one can run a cross-sectional regression (to be explained later) to find out the demand function based on the factors mentioned above, except for taste and expectation, which are difficult to quantify or measure. A simple linear regression could be:

$$
\hat{Q} = a_0 + a_1 \cdot P + a_2 \cdot Income + a_3 \cdot No.\_Buyers + a_4 \cdot Substitues + a_5 \cdot Complements
$$

All other factors not modeled are represented by an error term $\epsilon$ (or idiosyncratic term), which follows an independent and identically distribution (i.i.d.), for instance $\epsilon \sim N(0, \sigma^2)$. Once values of coefficients are identified, we could input the real data and construct the demand function. For example, supposing we would like to analyze the demand of IaaS market in London: assume everybody in London is a (potential) customer, where there are 7,825,200 in total; average income is £33,384 per year; and there is no substitute.

The basic model is a straight line, represented by a linear demand function $P = \alpha \cdot Q + \beta$. In fact, when there is a large customer pool, demand curve normally becomes smoother. However, model fit does not necessarily increase with increasing variables. In principle, there are three types of regression, respectively based on three types of data, i.e. cross-sectional, time-series and panel data [10]: the first one refers to running regressions based on data from different sections (such as regions) within one period, the second type refers to using data from the same section but at different periods of time, and the third type is the combination of both. Furthermore, methods to run regressions include Ordinary Least Squares (OLS), Weighted Least Squares (WLS), and Feasible Generalized Least Squares (FGLS), among which, OLS is most widely used [7]. Moreover, there are problems such as multicollinearity (i.e. determinants are interdependent) and heteroscedasticity (error terms are not i.i.d) that might occur, and detection methods include Tolerance Value, Variance Inflating Factor for multicollinearity and Park Test, Breusch-Pagan-Godfrey Test and White Test for heteroscedasticity, which have been discussed in [10, 24], along with solutions including dropping less significant regressors and transforming them into proper forms for multicollinearity as well as WLS and FGLS mentioned above for heteroscedasticity.

Another notable point is that there are many other options of regression models. For instance, one may introduce the diminishing/increasing effect on price by adding additional parameters such as $P^2$, $\frac{1}{Q}$ or $log P$ to enhance expressiveness. However, model fit does not necessarily increase with increasing variables. Model fit may be examined by a number of techniques, including F-Test, $R^2$ and adjusted $R^2$ indices for general models, and tests such as Wald Test that specialize increasing/decreasing regressors. [33, 24]. However, there is no perfect model for all, and the fittest model may only be developed by iterations of a construct-compare-refine process, because it is extremely dependent on sampling data and choosing of regressors [7, 33]. In addition, as the type of demand function changes, revenue function will be different as well:

- **Example 1:** if $P = \alpha \cdot Q + \beta$, then $R = P \cdot Q = \alpha \cdot Q^2 + \beta \cdot Q$
- **Example 2:** if $P = \alpha \cdot Q^2 + \beta \cdot Q + \gamma$, then $R = P \cdot Q = \alpha \cdot Q^3 + \beta \cdot Q^2 + \gamma \cdot Q$
- **Example 3:** if $P = \frac{\alpha}{Q}$, then $R = P \cdot Q = \alpha$, i.e. revenue is a constant no matter how to set price or quantity
- **Example 4:** if $P = \frac{\alpha}{Q} + \beta$, then $R = P \cdot Q = \alpha + \beta \cdot Q$

Since the objective function is $\Pi = Revenue - Cost$, if demand function is $P = \frac{\alpha}{Q} + \beta$ and there is no cost, it becomes $\Pi = \alpha + \beta \cdot Q$, where pricing becomes a Linear Programming problem [13], which far simplifies the problem. Alternatively, demand function may be $P = \alpha \cdot Q^2 + \beta \cdot Q + \gamma$, thus $\Pi = \sum_{i=1}^{N} (\alpha \cdot Q^3 + \beta \cdot Q^2 + \gamma \cdot Q)$, and it becomes a ‘Cubic Programming’ problem. A highlight is that increasing the degree of objective function would dramatically increase the difficulty, therefore is usually restricted as much as possible.

**Limitations and Remedies**

The econometric approach with regression analysis is useful to model demand functions for general markets with complex demand situations. However, regression analysis is based on a large data set sampled from
the real world, thereby a sound sampling method is critical to the reliability of this approach. Traditional sampling methods have been suggested and critically assessed in [21]. However, sometimes there is a difficulty in obtaining a large set of data, especially for small firms. In this case, Monte-Carlo sampling methods such as Gibbs Sampling may be used to generate new data based on original data, details of which could be found in [3].

5.2 Simulation and Forecast Models for Spot Product

Demand data are much easier to obtain in the spot market due to the bidding system. Supposing there are 100 customers in total, and they collectively bid for 99 instances with a price above 0.5 $ per instance hour, 202 instances above 0.4 $ per instance hour, 310 instances above 0.3 $ per instance hour, 401 instances above 0.2 $ per instance hour and 491 instances above 0.1 $ per instance hour, a demand curve could be plotted based on bids statistics, as shown in Figure 5, and a regression may be run to obtain demand function, such as the linear and logarithmic functions in the figure. For simplicity, let us simply choose the linear model for analysis.

Another important feature is that users could make further orders and terminate existing ones at any time, which may dramatically change demand situation from time to time. In particular, as current price is shown, customers tend to bid somewhere close, which could significantly amplifies fluctuation of demand. To some extent, this is similar to fluctuation of stock price in exchange market. As bid statistics change frequently, demand function derived by regressing over current statistics may no longer be feasible in the next period. For that reason, forecasting demand based on current (and may be also past) situation is required. There are a number of models could be used to model demand fluctuation and forecast the demand in the next period.

Shift of Demand Curve

A remarkable point is that the shape of demand curve may also change during fluctuation, so does the structure of the underlying demand function. For example, a linear function may become quadratic or cubic rather than staying linear. In economics, researcher normally use the concept “shift of demand curve” to explain demand changes and reduce complexity. For that reason, here we merely consider demand fluctuations as shifts of demand curve, but no change in the shape. In the case of $P = \alpha \cdot Q + \beta$ for example, demand fluctuation shall only change the value of $\beta$ but not $\alpha$. Also the intercept $\beta$ must only be a positive value, otherwise the demand function makes no sense.

5.2.1 Model I: Timely Optimized Static Regression

The first solution is directly using the result of regression analysis above. This sounds ironic as high fluctuation of demand has been emphasized time after time. The reason is when spot price is optimized frequently, for example every minute, then it may not be necessary to do forecast. This introduces the concept of **spot effective period** and **spot fluctuation rate**: the former indicates the length of time for a spot price remains constant, whereas the latter refers to the rate of update (e.g. every day, every hour or every minute, etc). Both spot effective period and fluctuation rate may either be constant or variable. When
spot effective period is small, demand is less likely to change. Even there is a change, it is more likely to be insignificant, and spot price shall be updated shortly to deal with the change. However, it is difficult to claim under what fluctuation rate can a firm assume a static demand and unnecessary to forecast. Further research is required.

5.2.2 Model II: Time Series Regression

In fact, there are a number of dynamic regression models that may be used [10]. Dynamic models could be divided into two classes: Distributed Lags Models and Auto-regressive models. An example of the first class is \( Q_{t+1} = a_1 \cdot P_t + a_2 \cdot P_{t-1} + a_3 \cdot P_{t-2} + b \), where \( t \) is the time index. This means that quantity at one time is influenced by price in the last three periods. An example of the second class is \( Q_{t+1} = k \cdot Q_t + a \cdot P_t + b \). This means quantity at one time is influenced by that in the last period, i.e. it is path dependent and its effect will be everlasting since its presence, although weakened each time. Auto-regressive models are widely used in econometrics to measure and analyze macroeconomic impacts.

5.2.3 Model III: Parametric Random Walk

In computational finance and econometrics, an important mathematical model is Random Walk Model, which extends the auto-regressive model \( f(t_{k+1}) = f(t_k) + a \) by replacing the constant additive term \( a \) with a normally distributed random variable [16]:

\[
z(t_{k+1}) = z(t_k) + \epsilon(t_k) \cdot \sqrt{\Delta t}
\]

where \( t_{k+1} = t_k + \Delta t \) and \( \epsilon(t_k) \sim N(0,1) \)

A notable point here is that the amount of time elapse now is taken into consideration (represented by \( \Delta t \)), and could be customized by the unit of measurement for \( \Delta t \). The term \( \epsilon(t_k) \cdot \sqrt{\Delta t} \) is a random step from \( z(t_k) \) to \( z(t_{k+1}) \) during the period from \( t_k \) to \( t_{k+1} \), where \( \epsilon(t_k) \) follows a standard normal random distribution. Supposing one unit of time is an hour, then if \( \Delta t = 0.5 \), \( \epsilon(t_k) \cdot \sqrt{\Delta t} \sim N(0, \sqrt{0.5}) \). However, random walk model may encounter negative value \( z(t_{N+1}) \), as there is no way to prevent a series of \( N \) steps \( \sum_{i=1}^{N} \epsilon(t_i) \cdot \sqrt{\Delta t} \) from being a negative value. Since the intercept cannot be negative, we move to a modified version: Parametric Random Walk Model.

First of all, CloudNexus uses the bid statistics and run a static linear regression and obtains \( P = \alpha \cdot Q + \beta \).

Supposing spot fluctuation rate is a constant and the effective period is \( \Delta t \), naively introducing a random step term into the model would be:

\[
P(t_{k+1}) = P(t_k) + \epsilon(t_k) \cdot \sqrt{\Delta t} = \alpha(t_k) \cdot Q(t_k) + \beta(t_k) + \epsilon(t_k) \cdot \sqrt{\Delta t}
\]

Since the intercept may only be a positive value, \( [\beta(t_k) + \epsilon(t_k) \cdot \sqrt{\Delta t}] > 0 \) must be satisfied. To do so, we introduce another parameter \( \omega \in (0,1) \), which indicates the proportion of maximal step away from \( \beta(t_k) \):

\[
P(t_{k+1}) = \alpha(t_k) \cdot Q(t_k) + \beta(t_k) + \omega \cdot \beta(t_k) \cdot \epsilon(t_k) \cdot \sqrt{\Delta t}
\]

The \( \omega \cdot \beta(t_k) \) in the step term makes sure a step is bounded below \( \beta(t_k) \), unless \( \sqrt{\Delta t} > 1 \). Then we could easily prevent \( \sqrt{\Delta t} > 1 \) by limiting spot effective period to be smaller than one unit of time. In this case, fluctuation may be insignificant when \( \Delta t \) is small. We could amplify it by replacing \( \sqrt{\Delta t} \) with \( \Delta t \):

\[
P(t_{k+1}) = \alpha(t_k) \cdot Q(t_k) + \beta(t_k) + \omega \cdot \beta(t_k) \cdot \epsilon(t_k) \cdot \Delta t = \alpha(t_k) \cdot Q(t_k) + \beta'(t_k)
\]

where \( \beta'(t_k) \sim N(\beta(t_k), \beta(t_k) \cdot \omega \cdot \Delta t) \)

In fact, \( \beta'(t_k) = \beta(t_k) \cdot [1 + \omega \cdot \epsilon(t_k) \cdot \Delta t] \) is a derivative of the multiplicative model: \( x_{t+1} = a \cdot x_t \). In this model, the value of \( \omega \) is determined by the user. As \( \omega \) gets smaller, the model is closer to Model I and vice versa. Besides, this model is not an auto-regressive model, because at any time \( t_k \), the actual outcome \( Q(t_k) \) may be slightly different from the forecasted \( Q(t_k) \) at \( t_{k-1} \), and forecast of \( Q(t_{k+1}) \) only uses the actual \( Q(t_k) \) and drops the forecasted \( Q(t_k) \). Actually, this property is recognized as Markov property, a.k.a. memorylessness, which means that future is conditional on current status but not correlated to the past [8]. In that sense, this model may be called Parametric Markov Process.

5.2.4 Model IV: Geometric Brownian Motion

As described in [16], a random walk model, by taking the limit \( \Delta t \to 0 \), becomes a Wiener Process or Brownian Motion:

\[
z(t_{k+1}) = z(t_k) + \epsilon(t_k) \cdot \sqrt{\Delta t} \implies z(t_{k+1}) - z(t_k) = \epsilon(t_k) \cdot \sqrt{\Delta t}
\]
\[
\lim_{\Delta t \to 0} \left[ z(t_{k+1}) - z(t_k) = \varepsilon(t_k) \cdot \sqrt{\Delta t} \right] \implies d z = \varepsilon(t_k) \cdot \sqrt{d t}
\]

An important extension is Generalized Wiener Process: \(dx(t) = a \cdot dt + b \cdot dz\), which means that the systematic change over time \(a \cdot dt\) and the random change \(b \cdot dz\) are also taken into account. Furthermore, Ito Process (named after Kiyoshi Itô) is even more generalized:

\[
dx(t) = a(x,t) \cdot dt + b(x,t) \cdot dz
\]

This indicates that both the rate of systematic change and the amplitude of fluctuation (or oscillation) are functions w.r.t. \(x\) and \(t\).

Next, again we look at the multiplicative model \(x_{t+1} = a \cdot x_t\). In this model, \(x_{t+1}\) is a geometric series that proportional to \(x_t\). By taking logarithm on both sides, it becomes \(ln(x_{t+1}) = ln(a \cdot x_t) = ln(a) + ln(x_t)\), which is an additive model. Applying multiplicative model to the intercept of demand function \(P = \alpha \cdot Q + \beta\) derived from regression, we could obtain \(\beta(t_{k+1}) = \mu \cdot \beta(t)\), and by taking logarithm, it transforms to \(ln[\beta(t_{k+1})] = ln[\beta(t_k)] + ln(\mu) = ln[\beta(t_k)] + w\). Supposing \(dln(\beta)\) is an Ito Process such that \(dln(\beta) = dw = v \cdot dt + \sigma \cdot dz\), where \(dz\) is a standard Wiener Process, and \(<v, \sigma>\) depends on \(ln(\beta)\) and \(t\), then

\[
d\beta = \frac{d\beta}{\beta} = dln(\beta) = v \cdot dt + \sigma \cdot dz
\]

However, this solution is problematic as in fact \(\frac{d\beta}{\beta} \approx dln(\beta)\) rather than \(\frac{d\beta}{\beta} = dln(\beta)\). Ito’s lemma indicates that the actual result should be \(\frac{d\beta}{\beta} = \left(v + \frac{1}{2} \sigma^2\right) \cdot dt + \sigma \cdot dz\), where the proof could be found in [16]. Finally, by rearrangement, this becomes

\[
d\beta = \beta \cdot \left[ \left(v + \frac{1}{2} \sigma^2\right) \cdot dt + \sigma \cdot dz\right]
\]

This means that the change of \(\beta\) is proportional to \(\beta\) itself, and the rate of change depends on a systematic change \(v\) and a variance \(\sigma\).

**Example**

Supposing at time \(t_k\), \(P(t_k) = \alpha(t_k) \cdot Q(t_k) + \beta(t_k)\), and by observing historical data, CloudNexus discovers that the demand curve gradually shifts to the right at a rate of 1% per hour on the intercept with a standard deviation of 5%. Supposing spot effective period \(\Delta t\) is a constant of 0.5 hour, then forecast of demand at \(t_{k+1}\) could be carried out by

\[
\Delta \beta(t_k) = \beta(t_k) \cdot \left[ \left(v + \frac{1}{2} \sigma^2\right) \cdot \Delta t + \sigma \cdot \Delta z\right] = \beta(t_k) \cdot (0.005625 + 0.035355 \cdot \varepsilon)
\]

\[
P(t_{k+1}) = P(t_k + \Delta t) = P(t_k) + \Delta \beta(t_k) = \alpha(t_k) \cdot Q(t_k) + \beta(t_k) + \Delta \beta(t_k)
\]

\[
= \alpha(t_k) \cdot Q(t_k) + 1.005625 \cdot \beta(t_k) + 0.035355 \cdot \varepsilon \cdot \beta(t_k)
\]

### 5.3 Alternative Demand Simulation Methods

There are a number of machine-learning based approaches that may be used to simulate demand situations. The most typical are Bayesian Net [23, 3], Decision Trees, Artificial Neural Network and Case-based Reasoning [20]. These system-based or network-based simulation approaches could be used to make inference on demand in the next period, based on past experience and probabilistic reasoning. However, they cannot be easily used to deduce a demand function. Whereas, the econometric approach has been widely used to model dependency in function forms. Nevertheless, these alternatives have not been critically assessed, further investigation is required.

### 6 Cost Simulation Models

#### 6.1 Cost Factors and Possible Simulation Models

There are two types of cost frequently quoted: opportunity cost and accounting cost. The former refers to the loss of performing some activity instead of choosing the best alternative; whereas the latter refers to the monetary expenditure of performing the activity [18]. The relationship between them is that opportunity cost includes accounting. In general, pricing models only consider accounting information rather
than alternative business opportunities, unless in extreme cases such as spot operation cost, which is to be discussed in Section 6.2. For now, we only consider accounting cost factors. Authors in [32] have identified four factors of supplying electricity, which are respectively capacity cost, transmission & distribution cost, energy-consumption cost and customer-related cost. In IaaS, cost factors are similar to some extent but also a bit different, which are summarized as below.

1. Capacity Cost: cost incurred to support the level of computing capacity, i.e. the investment in CPU, memory, hard disk and technology, etc.

2. Energy Cost: cost of energy consumption in the data centers, including electricity consumption and cooling.

3. Transmission Cost: cost of network traffic, and may be further divided into upload cost and download cost.

4. Maintenance Cost: cost of maintaining the operations of IaaS service, including cost of manpower and equipment replacement

5. Customer-related Cost: cost not covered in the above, mainly metering and billing expenses.

In our model, pre-investment (capacity cost) is a sunk cost [18] due to short-run conditions, thus is ruled out. As the whole cloud industry uses e-metering and e-billing technologies, once invested, there is no significant further monetary outflow, thus customer-related cost is neglected. For that reason, only energy cost, transmission cost and maintenance cost are considered. These costs are generally dependent on number of VM instances running, thereby could be collectively considered as a function of quantity supplied. For example, it may be a simple proportional function \( C = \gamma \cdot Q \). Different products may be associated with different cost functions, thereby for \( N \) products, the cost is \( C = \sum_{i=1}^{N} \gamma_i \cdot Q_i \). More complex models include \( C = \gamma_1 \cdot Q^2 + \gamma_2 \cdot Q \), which introduces a fixed cost regardless of supply; \( C = \gamma_1 \cdot Q^2 + \gamma_2 \cdot Q + \gamma_3 \), which specifies an additional increasing/diminishing effect in provisioning large numbers of instances; and \( C = \gamma_1 \cdot Q^2 + \gamma_2 \cdot Q + \gamma_3 \), which combines these two models; etc.

### 6.2 Practical Issue: Operation Cost of Spot Instance

Analysis above only considered the general cost factors in IaaS sector. However, in practice, spot products experience some operation cost, which is far more significant than all other costs and was significantly underestimated before practical trial. In general, spot instance may run or stop at different periods of time as spot price fluctuates over time. This is the key feature of spot product and is where practical issues arise. When a spot instance is paused, the system must ensure that data in the memory is retained so that when resumed, the instance is able to recover its original status. There are two basic methods to do so:

- **Method I**: If there is a large or unlimited size of total memory, the system could simply freeze the data in memory when instance is stopped. Then when the instance is resumed, it could directly access the data in memory as if nothing has happened. However, in reality memory resource is usually limited, taking this solution may be risky.

- **Method II**: On the other hand, the system could choose to dump out the data in memory and write them into disk when the instance is paused. However, it takes quite a while for this operation, and when the instance is resumed again, it also takes a lot of time to recover the status as it was. In other words, there is an operation cost associated with this move. It does not involve monetary outflow. However, when the system is spending a considerably long time pausing/resuming instances, it loses the opportunity to use the resources to make profit. For that reason, operation cost is part of opportunity cost, rather than accounting cost.

In reality, Method II takes too much time. Our experiments have shown that recovering data from disk for a single VM with only 1GB memory takes several minutes, let alone recovering hundreds of VMs. When spot fluctuation rate is low, there may be a lot of resumed VMs getting paused before they can complete the process of memory recovery. Operation cost is present when there are instances paused or resumed, thus it is correlated with \( \Delta Q \), i.e. the change of quantity, as well as the size of memory. Therefore operation cost may be defined as \( OC = \Delta Q \cdot m \), where \( m \) stands for the size of memory in each instance. Alternatively, we could also use \( OC = \Delta Q^2 \cdot m \) or \( OC = \Delta Q^2 \cdot m^2 \). Since operation cost primarily explains opportunity cost, it should be defined to justify the actual situation, thus its modeling requires extreme attentions and should be developed case-by-case. A notable point is that the profit considered will become economic profit rather than accounting profit.
6.2.1 Solution I: Freezing Memory

If memory is freezeed when an instance is paused, there shall be no operation cost. However, the freezeed memory may not be used by other instances. In other words, the corresponding memory size shall be deducted from total available memory resource.

Single-Product Case

If there is only one product, there is no trouble at all, as memory control is like a valve that determines how many instances should run and how many should not. This is because instances are queued up in descending order of bidding price.

Lemma: When there is only one spot product, if an instance \(i\) is offered with memory segment \(M_i\), there is no other instance may be allocated with \(M_i\)

Proof: Supposing \(i\) is the index in the the queue and the queue is in descending order, if there is another instance indexed by \(j\), there are two cases in comparing the priorities between \(i\) and \(j\): either \(j\) is placed before \(i\) in the queue or behind. If \(j > i\), i.e., \(j\) has higher priority, then when \(i\) is offered memory segment \(M_i\), \(j\) is already running, which inherently indicates it has already been allocated with another piece of memory \(M_j\), therefore freezing \(M_i\) does not affect \(j\). If \(i > j\), whenever \(j\) is able to use a piece of memory \(M_j\), \(i\) shall already be running and \(M_i\) is not freezed.

For that reason, the freezing mechanism works perfectly in the single-product case.

Multiple-Product Case

If there are a number of products, applying this solution need to take extra care. Suppose there are three products with quantities and prices denoted \(Q_1, Q_2, Q_3\) and \(P_1, P_2, P_3\). As described in the single-product case, instances of the same product will never reach in conflict, therefore such conflict may only occur between instances of different products. Remember, freezing memory shall only occur by pausing some running instances, for that reason,

\[ \text{Total Locked Memory} = \text{Freezed Memory} + \text{Memory in Use} \]

In fact, Total Locked Memory equals to the sum of historically maximal memory usage of each product. In other words, for each product, once quantity raises to \(K\), the memory section allocated to these \(K\) instances will always be locked since then. Suppose the maximum quantities in the history are \(Q_{1,\text{max}}, Q_{2,\text{max}}, Q_{3,\text{max}}\), freezing mechanism could be used as long as \(Q_{1,\text{max}} + Q_{2,\text{max}} + Q_{3,\text{max}} \leq \text{Total Memory}\). For that reason, the optimization problem becomes:

\[
\begin{align*}
\text{Maximise} \quad & \Pi = (P_1 \cdot Q_1 + P_2 \cdot Q_2 + P_3 \cdot Q_3) - (\gamma_1 \cdot Q_1 + \gamma_2 \cdot Q_2 + \gamma_3 \cdot Q_3) \\
\text{Subject to} \quad & \text{cpu}_1 \cdot Q_1 + \text{cpu}_2 \cdot Q_2 + \text{cpu}_3 \cdot Q_3 \leq T_{\text{cpu}} \\
& \text{memory}_1 \cdot \max(Q_1, Q_{1,\text{max}}) + \text{memory}_2 \cdot \max(Q_2, Q_{2,\text{max}}) + \text{memory}_3 \cdot \max(Q_3, Q_{3,\text{max}}) \leq T_{\text{memory}} \\
& \text{disk}_1 \cdot Q_1 + \text{disk}_2 \cdot Q_2 + \text{disk}_3 \cdot Q_3 \leq T_{\text{disk}} \\
& Q_1, Q_2, Q_3 \geq 0
\end{align*}
\]

After each run, \(\langle Q_{1,\text{max}}, Q_{2,\text{max}}, Q_{3,\text{max}} \rangle\) will be updated with the new \(\langle Q_1^*, Q_2^*, Q_3^* \rangle\). Since there is a special \(\max(Q, Q_{\text{max}})\) operator, the problem cannot be solved by standard QP algorithms. The solution needs further investigation.

Problems

This solution could waste memory as paused instances may never be resumed and freezeed memory may never be used if spot price never drops below the corresponding bidding price. Supposing there are A, B, C three products, if at certain period demand of A is high while demand of B and C is low, then CloudNexus shall set a lower price for product A. However, if later on both products B and C become more popular and more profitable than A, the company shall cut supply of product A (by rising its price) while focus on provisioning B and C. Nevertheless, by freezeing memory of type A instances, instances of B and C shall never be able to use the memory once used by type A instances. In the above model, the problem could be expressed as \(Q_{A,\text{max}}\) is very large while current \(Q_a\) is very small.
6.2.2 Solution II: Writing>Loading Memory Into>From Disk

This solution does not waste memory, however inevitably incurs operation cost, which should be taken into consideration during dynamical pricing. Supposing operation cost function is \( OC_i = \xi \cdot \text{memory}_i \cdot \Delta Q_i \), a naive problem formulation for the general N-Product model with constrained resources becomes

\[
\begin{align*}
\text{Maximise} \quad & \Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma \cdot \text{memory}_i \cdot Q_i - \sum_{i=1}^{N} \xi \cdot \Delta Q_i,
\text{Subject to} \quad & \sum_{i=1}^{N} \text{cpu}_i \cdot Q_i \leq T_{\text{cpu}}, \\
& \sum_{i=1}^{N} \text{memory}_i \cdot Q_i \leq T_{\text{memory}}, \\
& \sum_{i=1}^{N} \text{disk}_i \cdot Q_i \leq T_{\text{disk}}, \\
& Q_i \geq 0 \quad \text{for } i = 1, 2, \ldots, N
\end{align*}
\]

However, \( \Delta Q_i \) is unknown until the actual \( Q_i \) is worked out and compared with \( Q_i' \), i.e. the quantity in the last period. To cope with this, we use the definition of \( \Delta Q_i \) that \( \Delta Q_i = |Q_i - Q_i'| \) and substitute it into the naive model. Then the new objective function becomes

\[
\Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma \cdot Q_i - \sum_{i=1}^{N} \xi \cdot \text{memory}_i \cdot |Q_i - Q_i'|. \quad \text{However, existing QP solvers such as Matlab Quadprog do not support absolute values. A remedy is to replace } \Delta Q_i \text{ in the cost function with } (\Delta Q_i)^2, \quad \text{i.e. } OC_i = \xi \cdot \text{memory}_i \cdot (\Delta Q_i)^2 = \xi \cdot \text{memory}_i \cdot (Q_i - Q_i')^2.
\]

Then \( \Pi \) could become an objective function in the standard QP form by the following reduction:

\[
\begin{align*}
\Pi &= \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} \gamma \cdot Q_i - \sum_{i=1}^{N} \xi \cdot \text{memory}_i \cdot (Q_i - Q_i')^2 \\
&= \sum_{i=1}^{N} \left[ P_i \cdot Q_i - \gamma \cdot Q_i - \xi \cdot \text{memory}_i \cdot (Q_i - Q_i')^2 \right] \\
&= \sum_{i=1}^{N} \left[ (\alpha_i - \xi \cdot \text{memory}_i) \cdot Q_i^2 + (\beta_i - \gamma + 2 \cdot Q_i' \cdot \xi \cdot \text{memory}_i) \cdot Q_i - \xi \cdot \text{memory}_i \cdot Q_i'^2 \right] \\
&= \sum_{i=1}^{N} \left[ (\alpha_i - \xi \cdot \text{memory}_i) \cdot Q_i^2 \right] + \sum_{i=1}^{N} \left[ (\beta_i - \gamma + 2 \cdot Q_i' \cdot \xi \cdot \text{memory}_i) \cdot Q_i \right] - \sum_{i=1}^{N} \left[ \xi \cdot \text{memory}_i \cdot Q_i'^2 \right]
\end{align*}
\]

Inputting the values of \( <Q_1', Q_2', \ldots, Q_N'> \), we could use standard QP solutions to work out \( <Q_1, Q_2, \ldots, Q_N> \), and then \( <P_1', P_2', \ldots, P_N'> \) and total profit.

Problems

This solution brings in operation cost at each time unless there is no change in the quantity of each product, thus significantly decreases profit and delays instance resuming time.

6.2.3 Hybrid Solution I

More advanced solutions would probably take into account both solutions above. For example: when there is sufficient memory, use the freezing solution; and when there is insufficient memory, release all the frozen memory. In this case, use the original model that neglects memory freezing effects, and meanwhile maintain a record of \( Q_{i,\text{max}} \) which is updated after each run. When \( \sum_{i=1}^{N} Q_{i,\text{max}} > T_{\text{memory}} \), the system performs an explicit memory releasing task to the current status, i.e. \( Q_{i,\text{max}} = Q_i \). However, in that special case, there is an operation cost \( \sum_{i=1}^{N} (Q_{i,\text{max}} - Q_i)^2 \cdot \text{memory}_i \), which reduces the profit.

Problems

This solution is still not good enough. Suppose there are three products: product A consumes large disk space, whereas product B or C does not. When CloudNexus is suddenly able to sell a lot of product A in the standard market, there will be very little disk space left. In that sense, in the spot market, the optimum may suddenly change from \( <\text{large } Q_A, \text{small } Q_B, \text{small } Q_C> \) to \( <\text{small } Q_A, \text{large } Q_B, \text{large } Q_C> \). A significant memory clean-up work must be carried out, which brings a significant operation cost. As profit

\[
\Pi = \text{Revenue} - \text{Maintenance Cost} - \text{Operation Cost}, \quad \text{profit may even be negative.}
\]
6.2.4 Hybrid Solution II

The problem of Hybrid Solution I is a thorough memory clean-up mechanism at the time \(\sum_{i=1}^{N} Q_{i,\text{max}} > T_{\text{memory}}\). In the example above, a better strategy may be a moderate drop of \(Q_A\) to fit the disk need in the standard market rather than a sharp decrease. Optimal level of drop must be subject to demand functions, maintenance cost functions and up-to-date resource availability. Among them, operation cost function is most tricky, as there is operation cost only when there is insufficient memory. Assuming \(OC_i = \xi \cdot memory_i \cdot (\Delta Q_i)^2\), the operation cost function may be defined as below:

\[
X(Q_i, Q_{i,\text{max}}) = \begin{cases} 
\sum_{i=1}^{N} \max(Q_{i,\text{max}}, Q_i) \leq T_{\text{memory}} & \sum_{i=1}^{N} \max(Q_{i,\text{max}}, Q_i) > T_{\text{memory}} \\
0 & \sum_{i=1}^{N} \xi \cdot memory_i \cdot (Q_i - Q_{i,\text{max}})^2
\end{cases}
\]

With this special operation cost function, the problem could be formulated as the following:

Maximise \(\Pi = \sum_{i=1}^{N} P_i \cdot Q_i - \sum_{i=1}^{N} g_i(Q_i) - X(Q_i, Q_{i,\text{max}})\)

Subject to

\[
\sum_{i=1}^{N} \text{cpu}_i \cdot Q_i \leq T_{\text{cpu}}
\]

\[
\sum_{i=1}^{N} memory_i \cdot Q_i \leq T_{\text{memory}}
\]

\[
\sum_{i=1}^{N} \text{disk}_i \cdot Q_i \leq T_{\text{disk}}
\]

\(Q_i \geq 0 \quad \text{for } i = 1, 2, \ldots, N\)

Once solving for \(<Q_1^*, Q_2^*, \ldots, Q_N^*>\), total memory release, operation cost and total profit could be worked out without difficulty.

Problems

Unfortunately, existing QP solutions do not support the function \(X(Q_i, Q_{i,\text{max}})\). Further investigation is required to sort this out.

7 Practice: IC Cloud and Spot Pricing System

IC Cloud is the abbreviation of Imperial College Cloud Platform, which is a commercial system that provides IaaS service [6]. Previously, IC-Cloud has already launched a number of standard products with different pay schemes, including pay-hourly, pay-daily, pay-weekly, pay-monthly and so forth. However, the system lacks a high-level pricing system. In particular, it needs a spot pricing mechanism before it could launch spot product. The spot pricing system is therefore developed to meet this requirement. Figure 6 and Figure 7 respectively demonstrate the system’s monitor panel and pricing engine control panel. The spot pricing system has been used to run a number of hypothesis testing tasks to assess the theory proposed in the paper.

7.1 Hypothesis Testing

For a pricing theory to be useful and practical, various aspects must be assessed. Since there are no enough data to carry out econometric analysis for standard prouct, this paper only tests the capability of the developed spot pricing models to support dynamic optimization. There is no significant difference between the application of spot pricing in monopoly and oligopoly without outsourcing. However, outsourcing cannot be tested as today’s technology does not support seamless outourcing in practice. Below are three fundamental hypotheses centered at spot dynamic optimization:

- **H1:** If there is a change of demand in the spot market while holding situation in the standard market unchanged, then the spot mechanism shall dynamically adjust price (and also quantity) of spot product to achieve a new optimum.
Figure 6: Administrator Interface 1: Monitor Panel

Figure 7: Administrator Interface 2: Spot Pricing Engine Control Panel
- **H2**: If there is a change of demand in the standard market while holding situation in the spot market unchanged: since price in the standard market is fixed, available resources left for spot product will be different; then spot price shall be dynamically adjusted to maximize profits under the new constraints.

- **H3**: If there is a change of demand that occurs in both markets, then spot mechanism shall take into account all the changes and adjust spot price properly so that profit is maximized under new circumstances.

A number of experiments have been run on the spot pricing system (for details refer to [36]), and a conclusion has been drawn that the developed theory supports all the three hypotheses.

### 7.2 Limitations

However, there are a number of limitations in the developed theory. The most significant is that optimization problem solving highly relies on Mathematical Programming techniques. If demand functions become more complex, for example in polynomial form such as $P_i = \alpha_i \cdot Q_i^2 + \beta_i \cdot Q_i + \gamma_i$, then the optimization problem becomes a 'Cubic Programming' problem that cannot be solved by QP algorithms, as pointed out in Section 5.1. In fact, there is no such a field termed Cubic Programming, probably because problems become too complex. Instead, there is a more general form of Mathematical Programming coined Non-linear Programming [2], which primarily focuses on tackling problems with nonlinear constraints. As the degree of objective function increases, complexity grows exponentially. Moreover, existing QP solvers like Matlab QuadProg cannot solve problems with special operators such as absolute value and special functions, as suggested in Section 6.2.1, 6.2.2 and 6.2.4. Further investigation is required in solving complex Mathematical Programming problems.

### 8 Conclusion

This paper has proposed a systematic theory with a number of mathematical models to guide cloud IaaS providers in developing pricing strategies. The theory covers a wide range of fields, including utility computing, mathematical modeling, economics, econometrics, mathematical finance, operations research, game theory and industrial organization. It started with a brief introduction to dimensions of utility pricing models, general pricing techniques and a brief explanation of IaaS market type as well as rational firm behaviors. The most important pricing technique is to launch spot product and create a secondary market. After that, the paper analyzes optimal pricing behaviors under various circumstances.

In the case of monopoly, the only firm becomes a price maker. If the firm only launches one product, we could use first order optimization to work out the optimum. However, if there are multiple products, this solution may not be applicable as there may be too many possibilities of constructing a product portfolio. For that reason, the approach moves to Mathematical Programming.

In the case of oligopoly, situation in the standard market changes a lot. As all the firms in the market collectively share a single demand curve and collectively determine market supply. This becomes a classic game theory and industrial organization problem. The best strategy for the whole seller group is to make an agreement to collectively output at the level equal to the optimum of monopoly. However, each firm is always better off defecting rather than cooperating, falling into a Nash Equilibrium, and the outcome is the worst. On the other hand, spot pricing in oligopoly does not encounter this problem and is not significantly different from the case of monopoly. This ascribes to the bidding system in the spot market. Furthermore in oligopoly, firms have the option to outsource from others in order to cover their temporary resource shortage. Models with and without outsourcing have been developed. More specifically, outsourcing may be divided into two forms, instance-based outsourcing and resource-based outsourcing. In addition, two pricing models for brokers have been developed, one for instance-based outsourcing and the other for resource-based outsourcing. A broker is a special role that sells computing services to end users based on other provider’s products, rather than deploying resources on his/her own. Profits of brokers come from the price gap between premium price enjoyed by the broker and disadvantaged customers.

The principle of all the models developed in this paper is formulating some optimization problem using demand function and production constraints (including cost function and productivity). An econometric approach has been adopted in simulation of demand in the standard market, along with analysis of benefits and drawbacks. Careful analysis has also been conducted on demand simulation in the spot market with the suggestion of four models, which are respectively Timely Optimized Static Regression, Time Series Regression, Parametric Random Walk Model and Geometric Brownian Motion.

As for cost simulation, five primary cost factors in the IaaS industry have been identified and analyzed, along with some possible simulation models. It is also pointed out that in practice, operation cost is the primary cost in the spot market. Operation cost occurs when there is a procedure of reading/writing data...
between memory and disk when switching status of spot instance between running and paused. It is negligible if one chooses to freeze memory when pausing spot instance from working. Four solutions have been suggested and critically analyzed.

Finally, a spot pricing system has been implemented on top of the Imperial College Cloud Platform (IC Cloud) to test relevant hypotheses regarding the practicality of the spot pricing mechanism under the circumstance of monopoly and oligopoly without outsourcing.

**Future Work**

This theory developed in this paper faces a number of limitations and requires further investigation. The most significant ones have been listed in Section 7.2. Furthermore, the hypotheses tested in this paper only support spot dynamic pricing in the case of monopoly and oligopoly with no outsourcing. Future research may explore the possibility of outsourcing-based spot pricing, including the broker models. Moreover, although starting point has been suggested, this paper did not develop practical models for oligopolistic pricing in the standard market. Further work may use frameworks from Game Theory and Industrial Organization and based on the models developed in this paper to address the problem of pricing standard product in oligopoly. In addition, the paper has also pointed that different products in the IaaS market are mutual substitutes, which forms a dynamic system of inter-correlated demands. Supposing there are \( N \) products, the mutual substitutable property would lead to a much more complex system of demand function like below:

\[
Q_1 = a_0 + a_1 \cdot P_1 + a_2 \cdot P_2 + \cdots + a_N \cdot P_N \\
Q_2 = b_0 + b_1 \cdot P_1 + b_2 \cdot P_2 + \cdots + b_N \cdot P_N \\
\vdots \\
Q_N = k_0 + k_1 \cdot P_1 + k_2 \cdot P_2 + \cdots + k_N \cdot P_N
\]

This means that changing the supply of any product will influence the demands of the rest. The models developed in the paper abstract away such complexity and assume they are independent. Future work may need to tackle the dynamic system of inter-correlated demands. Finally, the paper suggests four models to capture demand fluctuations in the spot market. However, no test could be carried out to verify the fitness of each model and thus to choose the most suitable solution. Empirical data of bid statistics from real IaaS providers are needed, because fitness may only be verified in modeling real data.

**References**


