International Journal of u- and e- Service, Science and Technology
Vol.9, No. 9 (2016), pp.111-120
http://dx.doi.org/10.14257/ijunesst.2016.9.9.12

Vertical or Horizontal Scaling for Web Information Systems

Wenjuan Liu[1,2],Guosun Zeng[1] and Huanliang Xiong[1,3]

[1]*Department of Computer Science and Technology,Tongji University,Shanghai 201804, China*
[2]*State Key Laboratory of High-end Server & Storage,Technology,Jinan 250101, China*
[3]*Jiangxi Agricultural University,Nanchang 330045*
wenjuanliu_china@163.com

Abstract

As the number of Web users and the amount of information produced keep growing rapidly, information systems need to be scaled up constantly. Usually scaling methods of information systems suffer from being chosen by experience, this paper studied on quantitative optimal selection of vertical/horizontal scaling methods. According to investigation and statistics, we assumed the cost growth formula of the key performance parameters reasonably. Then five key parameters of systems were chosen to construct Kiviat graphs and the areas of Kiviat graphs were computed to evaluate system performance. Next, the quantitative function relation between the performance of information system and the cost of system scaling was established, and then three optimal selection algorithms for system scaling were proposed. Finally, some case studies demonstrate that algorithms are feasible and have a great significance for guiding system scaling.

Keywords*: Web information system; vertical scaling; horizontal scaling; optimal selection*

1. Introduction

Since 1990s, driven by the global information highway construction booms, information systems are widely built and used. Typical information systems include e-business, e-government affairs, websites of all departments, remote education systems, digital libraries, and so on. They offer convenience for people's work, study and everyday life greatly. With the rapid development of Web systems, mobile phones and mobile internet, the number of users and the amount of information created increase quickly. However, the functions and scales of information systems at present cannot meet the requirements of the growing of business and need to be upgraded and scaled urgently.

Generally speaking, the scalability [1]of a computer system is an ability that the computational performance of the system increases correspondingly when its nodes size increases, which is an important performance indicator of information systems. Scholars have been working in the research of scalability, in the respects of both hardware, such as CPU, memory, disk and network, and software.

CPU scaling. In 2008, adding more cores in a single CPU had become a main trend to improve processing performance [2], and it was very necessary to implement parallel algorithms on multiprocessors, then programming models which were suitable for multiprocessors were put forward and designed. And some researches were done to allocate and process computing loads both on CPU and GPU [3], which realized the extension from CPU to GPU and alleviated the problem of poor performance caused by imbalanced loads.

ISSN: 2005-4246 IJUNESST

Memory scaling. It includes devices and architecture. For devices, the problems of energy consumption and data bandwidth caused by memory scaling [4] were analyzed and new storage technologies were studied in 2012, including 3D-IC and non-volatile storage. For architecture, Mutlu *etc.*, [5] not only analyzed the bottleneck problems of memory systems, but also evaluated the huge costs which were caused by maintaining and enhancing the efficiency and stability of resources.

Disk scaling. In order to maintain load balancing among all the disks, RAID-6 extension has to move the existing data blocks to the newly added disks. However, because of the loop distribution of data, all the existing methods for expansion need to move all the data blocks, which leads to huge costs for RAID-6 extension. Then in 2015, a new method was proposed that can accelerate the RAID-6 extension by reducing I/O and XOR operations of disks [6]. And it also can minimize the number of data blocks which need to be moved while maintaining the distribution of data.

Software system scaling. The problems about software scalability theory, software architecture and software design [7-9] were also further studied. At present, an important goal for database systems is to provide elastic extension according to load variations. However, database systems have states. To solve the problem of state consistence, a database scaling method [10] was put forward.

Not only industry but also academia did a lot of studies on the extension of hardware and software. However, the existing scaling methods of information system usually came from previous experience, lacking relational quantitative analysis. Therefore, this paper will analyze the scalability of Web systems quantitatively and then provide algorithms for selecting vertical/horizontal scaling optimally, so as to guide the extension and upgrade of information systems.

The rest of the paper is organized as follows. Section 2 analyzes the requirements and problems of system scaling. Section 3 reviews the architecture of Web system and the cost model of scaling. Section 4 proposes a method to evaluate comprehensive performance and calculate scaling times of each component. In section 5 we present three optimal selection algorithms for different scaling scenarios, and in section 6 we analyze the three algorithms. Finally, section 7 concludes this paper.

2. Basic Concept and Extension Requirements

In recent years, with the high-speed development of the Internet and information systems, people become more dependent on Web systems. At the same time, mobile devices emerge in endlessly, which needs the enhancement of system functions. However, the performance of existing systems cannot meet applications' requirement and need to be expanded urgently. For example, Taobao, its web system cannot handle all the users' request in a valid time because visitors increased sharply. It needs to be expanded to meet the requirement of business grows. We can achieve system scaling in two ways, vertical scaling or horizontal scaling.

Definition 1 (vertical scaling): Adding resources to the same logic unit to improve the performance of a system, such as upgrading the servers' CPU, adding more disks to the RAID/SAN storage devices, and so on.

Definition 2 (horizontal scaling): Adding multiple logic units of resources to improve the performance of a system and making them work as a whole. For example, we can add computing nodes to improve computing power of a computer cluster.

Usually, when information systems need to be scaled up, people choose a so-called appropriate scaling method from vertical and horizontal scaling just by experience rather than compare and analyze them reasonably. Therefore, in this

paper, we provide a quantitative method for calculating the relationship between the performance and the cost for system scaling so as to guide web systems to choose an appropriate scaling method. A decision rule is that we choose vertical scaling to expand the comprehensive performance of a web system by a certain time and let C^v denote the corresponding cost. Similarly, we choose horizontal scaling to expand the system by the same time and let C^h denote the cost. If $C^v > C^h$, we should choose horizontal scaling, otherwise, choose vertical scaling. Another rule is that we spend cost C to expand a web system with vertical scaling and horizontal scaling, respectively. Then the new correspondingly performances we can gain are P^v and P^h. If $P^v > P^h$, we should choose horizontal scaling, otherwise, choose vertical scaling.

3. Web System Architecture and Cost Model

3.1. Web System Architecture

Information systems include a variety of types, however, they have the same basic framework. In this paper, we take a Web system as an example (shown in Figure 1). Usually, a web system includes application servers, storage servers, and publish servers. With the popularity of mobile network and devices, people can access Web systems at any time and any place, which causes that the amount of data produced becomes larger and larger and continue to grow at a breathtaking pace. And in the near future, Web systems will face the risk of collapse. Therefore, Web systems need to be extended urgently and effectively to improve their performance.

No matter vertical scaling or horizontal scaling, we mainly care about the scaling about CPU, memory, hard disk, network and software. Thus, Web system extension can be abstracted as Figure 2. In Figure 2, two kinds of scaling are shown, vertical scaling (above) and horizontal scaling (below). Although we can achieve the goal of improving system performance by choosing any kind of scaling methods, we need carry quantitative analysis and comparison to choose a better one.

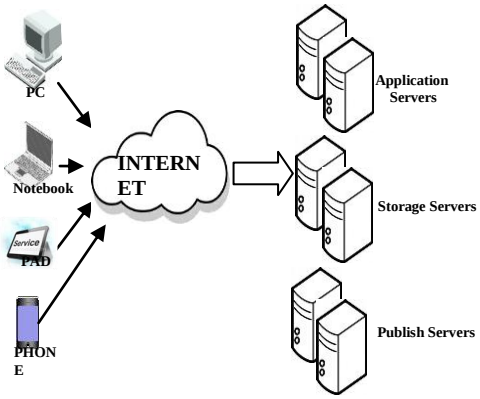

Figure 1. Web System Schematic Diagram

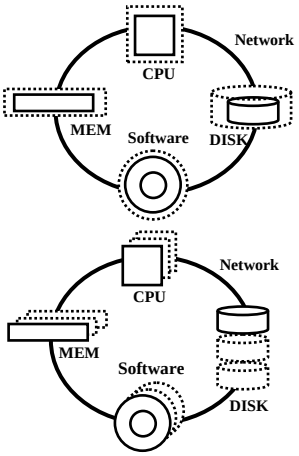

Figure 2. An Abstract Web System

3.2. Modeling

3.2.1. Cost Model of Vertical Scaling: Let C^v denotes the total cost for vertical scaling. For a Web system, when vertical scaling is carried out, five parts are mainly concerned, including CPU, memory, hard disk, network and software. Thus C^v equals to the sum cost of the five parts. Let r^v_{cpu}, r^v_{mem}, r^v_{disk}, r^v_{net} and r^v_{soft}, which can be determined by statistical

analysis, denote the cost for improving unit performance on CPU, memory, hard disk, network and software, respectively. In order to figure out C^v, we need to calculate C^v_{cpu}, C^v_{mem}, C^v_{disk}, C^v_{net} and C^v_{soft} firstly, where C^v_{cpu}, C^v_{mem}, C^v_{disk}, C^v_{net} and C^v_{soft} denote the cost of CPU, memory, hard disk, network, and software, respectively.

As well known, for vertical scaling, it's not a simple linear function relation between the cost for CPU scaling and the multiple of performance improved for CPU. This is because, during the process of improving the performance of CPU, the manufacturing technology and process is becoming more and more complex and difficult, which causes that the cost increases exponentially. Therefore, we have reasons to assume that:

$$C^v_{cpu} = e^{v_1 \times x_1} \times r^v_{cpu} \quad (1)$$

Where v_1 is a constant and x_1 is the multiple of performance improved for CPU.

Also, the function relation between the cost for memory scaling and the multiple of performance improved for memory is not linear. We can also assume that:

$$C^v_{mem} = e^{v_2 \times x_2} \times r^v_{mem} \quad (2)$$

Where v_2 is a constant and x_2 is the multiple of performance improved for memory.

To achieve the goal of improving disk performance, original disks are replaced by larger capacity of disks during disk vertical scaling. In general, the function relation between the cost and the capacity of disks is roughly linear. Thus, for vertical scaling, the cost trend of disk scaling appears as a certain linear form and we assume that:

$$C^v_{disk} = v_3 \times x_3 \times r^v_{disk} \quad (3)$$

Where v_3 is a constant and x_3 is the multiple of performance improved for disk.

Due to the characteristics of vertical scaling, network scaling does not change network typology structure of the system, and we only increase network bandwidth to improve the performance of network system. In general, there is a linear function relation between the cost used to increase network bandwidth and the amount of bandwidth increased. Thus, we assume that

$$C^v_{net} = v_4 \times x_4 \times r^v_{net} \quad (4)$$

Where v_4 is a constant and x_4 is the multiple of performance improved for network.

Similarly, during software vertical scaling, we do not change the software architecture, but increase the concurrent users to improve the software performance. Also, the relation between the cost used to enhance software performance and the increased concurrent users is roughly linear. Thus, for vertical scaling, the cost trend of software scaling appears as a certain linear form and we assume that:

$$C^v_{soft} = v_5 \times x_5 \times r^v_{soft} \quad (5)$$

Where v_5 is a constant and x_5 is the multiple of performance improved for software.

At last, if vertical scaling is chosen and performance is improved by k times, the total cost is expressed as: $C^v = e^{v_1 \times k^v_1} \times r^v_{cpu} + e^{v_2 \times k^v_2} \times r^v_{mem} + v_3 \times k^v_3 \times r^v_{disk} + v_4 \times k^v_4 \times r^v_{net} + v_5 \times k^v_5 \times r^v_{soft}$, where k^v_1, k^v_2, k^v_3, k^v_4 and k^v_5 denote the multiple of performance improved for CPU, memory, disk, network, and software, respectively.

3.2.1. Cost Model of Horizontal Scaling: Let C^h denotes the total cost for horizontal scaling. For Web systems, horizontal scaling also mainly cares about the five parts: CPU, memory, hard disk, network and software. Therefore, C^h equals to the sum cost of the five parts. Let r^h_{cpu}, r^h_{mem}, r^h_{disk}, r^h_{net} and r^h_{soft}, which can be determined by statistical analysis, denote the cost for improving unit performance on CPU,

memory, hard disk, network and software, respectively. In order to figure out C^h, we need to calculate C^h_{cpu}, C^h_{mem}, C^h_{disk}, C^h_{net} and C^h_{soft} firstly, where C^h_{cpu}, C^h_{mem}, C^h_{disk}, C^h_{net} and C^h_{soft} denote the cost of CPU, memory, hard disk, network, and software, respectively.

Similar to vertical scaling, we analyze the characteristics of horizontal scaling and investigate the cost on the market, and we assume the cost trends of the five parts that: $C^h_{cpu} = h_1 \times y_1 \times r^h_{cpu}$, $C^h_{mem} = h_2 \times y_2 \times r^h_{mem}$, $C^h_{disk} = h_3 \times y_3 \times r^h_{disk}$, $C^h_{net} = h_4 \times (y_4)^2 \times r^h_{net}$, and $C^h_{soft} = e^{h_5 \times (y_5)^2} \times r^h_{soft}$, where h_1, h_2, h_3, h_4, and h_5 are constants, and y_1, y_2, y_3, y_4, and y_5 represent the multiple of performance improved for CPU, memory, disk, network, and software, respectively.

Therefore, if horizontal scaling is chosen and performance is improved by k times, the total cost is expressed as: $C^h = h_1 \times k_1^h \times r^h_{cpu} + h_2 \times k_2^h \times r^h_{mem} + h_3 \times k_3^h \times r^h_{disk} + h_4 \times (k_4^h)^2 \times r^h_{net} + e^{h_5 \times (k_5^h)^2} \times r^h_{soft}$, where k^h_1, k^h_2, k^h_3, k^h_4 and k^h_5 denote the multiple of performance improved for CPU, memory, disk, network, and software, respectively.

4. Comprehensive Performance Model

In order to find out the function relation between k times improved on system performance, and k^v_1, k^v_2, k^v_3, k^v_4 and k^v_5 (k^h_1, k^h_2, k^h_3, k^h_4 and k^h_5) times improved on performance of five components, we do quantitative analysis for them.

4.1. Area Method for Evaluating Comprehensive Performance

We know that the performance of computer system is mainly reflected by join action of CPU, memory, disk, network and software. Thus, in this paper, an area measurement method is proposed to reflect the performance of system. Let P_0 denote the comprehensive performance of system, and P_1, P_2, P_3, P_4, and P_5 denote the corresponding performance of CPU, memory, disk, network, and software, respectively. We take these components as indicators to construct Kiviat graph, as shown in Figure 3.

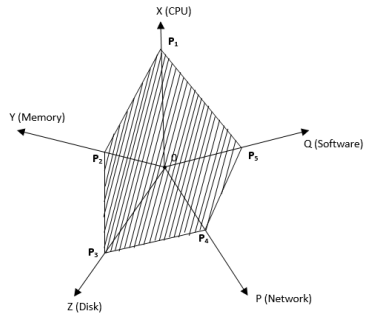

Figure 3. A Kiviat Graph of Comprehensive Performance (Left)

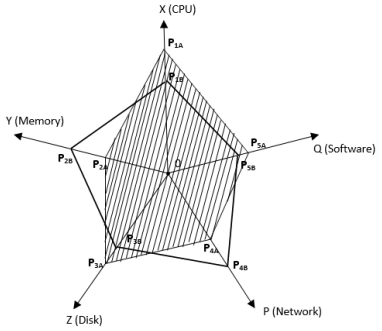

Figure 4. Comparison of Comprehensive Performances (Right)

A mathematical formula is used to express the Figure 3, as follows:

$$S = S(P_1, P_2, P_3, P_4, P_5) = (P_0)^2 \quad (6)$$

Where S is the area of Kiviat graph constructed by P_1, P_2, P_3, P_4 and P_5. Obviously, the comprehensive performance of system can be reflected by the area of the polygon. Given the comprehensive performance P^A_0 of system A and P^B_0 of system B, the corresponding area of polygons are S^A and S^B. In Figure 4, the followings are known: (1) $S^A = S^B$, namely, $P^A_0 = P^B_0$, which shows that system A and B have the same performance basically; (2) $S^A > S^B$, namely, $P^A_0 = P^B_0$, which shows that the performance of system A is

superior than B; (3) $S^A<S^B$, namely, $P_0^A<P_0^B$, which shows that the performance of system B is superior than A. Therefore, whether to choose vertical scaling or horizontal scaling can be decided by comparing the areas of corresponding polygons.

4.2. Calculation of Each Component

In section 4.1, we evaluate the comprehensive performance of system by calculating the area of polygon. On the contrary, given area S or performance P_0, we need to calculate each component of comprehensive performance to implement particular and effective scaling operations.

According to the formula(6), there is a more specific analytical expression: $S = \frac{1}{2} \times (P_1 \times P_2 + P_2 \times P_3 + P_3 \times P_4 + P_4 \times P_5 + P_5 \times P_1) \times \sin\theta$, where θ is the angle between two adjacent coordinate axes. Obviously, this formula, having more unknown variables than constraint equations, usually has an infinite number of solutions, so we can randomly select P_1, P_2, P_3, P_4 and P_5, which just need to meet the above formula. In particular, if the five components of system are equally important, which means that they do the same contribution to the comprehensive performance of system, the five components are $P_1=P_2=P_3=P_4=P_5= \sqrt{\frac{2S}{5\sin\theta}}$

However, in one information system, the five components have different effects on the comprehensive performance of the system. For vertical scaling, CPU and memory have the greatest effect on it, while for horizontal scaling, software also has great effect on it because applications require to work in parallel. In a word, we can use proportionality coefficient to reflect the different effects, namely, P_1: P_2: P_3: P_4: $P_5= \lambda_1 : \lambda_2 : \lambda_3 : \lambda_4 : \lambda_5$. According to the formula (6), we obtain that $P_1= \lambda_1 M$, $P_2= \lambda_2 M$, $P_3= \lambda_3 M$, $P_4= \lambda_4 M$ and $P_5= \lambda_5 M$, where $M= \sqrt{\frac{2S}{(\lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_4 + \lambda_4\lambda_5 + \lambda_5\lambda_1) \times \sin\theta}}$

4.3. Semantics of Extension by k Times

Regarding to vertical scaling or horizontal scaling, system scaling usually refers to the improvement of comprehensive performance. In another word, system scaling by k times means we improve the comprehensive performance by k times. While the improvement of comprehensive performance reflects in the improvement of the five components, we can expand each component at a certain times to achieve the objective of improving the comprehensive performance by k times.

Before scaling, let P_0 denotes the comprehensive performance, and P_1, P_2, P_3, P_4 and P_5 denotes the performance of CPU, memory, disk, network, and software, respectively. After scaling by k times, let P_0' denotes the comprehensive performance after system scaling, and $P_0'=k\times P_0$. Let P_1', P_2', P_3', P_4' and P_5' denote the performance of CPU, memory, disk, network, and software after scaling, and k_1, k_2, k_3, k_4 and k_5 denote their corresponding multiple of performance improved for five components. There are two kinds of conditions: (1) if five components scale up by the same times at the same time, then $k_1=k_2=k_3=k_4=k_5=k$. (2) if not, then they need to satisfy the following equation:

$$\begin{aligned}&(k)^2 \times (P_1 \times P_2 + P_2 \times P_3 + P_3 \times P_4 + P_4 \times P_5 + P_5 \times P_1)\\ &= (k_1 \times P_1 \times k_2 \times P_2 + k_2 \times P_2 \times k_3 \times P_3 + k_3 \times P_3 \times k_4 \times P_4 + k_4 \times P_4 \times k_5 \times P_5 + k_5 \times P_5 \times k_1 \times P_1)\end{aligned} \quad (7)$$

5. Optimal Selection Algorithms

In this section, we put forward three kinds of optimal selection algorithms for different extension scenarios.

5.1. Same Extension for Each Component and Less Cost Chosen First (SELC)

According to the demand of reality and applications, many enterprises tend to choose a scheme that is simple and easy to be implemented in order to save time and effort. Thus, this paper proposes SELC algorithm. The main idea is that each performance component is scaled up by the same times in vertical and horizontal scaling, respectively. Then the two kinds of cost will be compared and a lower one will be chosen. The corresponding pseudo-codes are as follows:

Algorithm 1 SELC *algorithm*

input: k; h_1, h_2, h_3, h_4, h_5; $r^h_{cpu}, r^h_{mem}, r^h_{disk}, r^h_{net}, r^h_{soft}$; v_1, v_2, v_3, v_4, v_5; $r^v_{cpu}, r^v_{mem}, r^v_{disk}, r^v_{net}, r^v_{soft}$;

// k denotes the time of system scaling, other parameters just as the definition in this paper;

output: 'v' , 'h'; // v denotes vertical scaling, h denotes horizontal scaling;

Steps:

{ $C^{v*}_{cpu} \leftarrow e^{v_1 \times k} \times r^v_{cpu}$; //according to statistical facts, CPU cost for vertical scaling;

$C^{v*}_{mem} \leftarrow e^{v_2 \times k} \times r^v_{mem}$; // memory cost;

$C^{v*}_{disk} \leftarrow v_3 \times k \times r^v_{disk}$; // disk cost;

$C^{v*}_{net} \leftarrow v_4 \times k \times r^v_{net}$; // network cost;

$C^{v*}_{soft} \leftarrow v_5 \times k \times r^v_{soft}$; // software system cost;

$C^{v*} \leftarrow C^{v*}_{cpu} + C^{v*}_{mem} + C^{v*}_{disk} + C^{v*}_{net} + C^{v*}_{soft}$; // total vertical scaling cost for expanding k times;

$C^{h*}_{cpu} \leftarrow h_1 \times k \times r^h_{cpu}$; // CPU cost for horizontal scaling;

$C^{h*}_{mem} \leftarrow h_2 \times k \times r^h_{mem}$;

$C^{h*}_{disk} \leftarrow h_3 \times k \times r^h_{disk}$;

$C^{h*}_{net} \leftarrow h_4 \times (k)^2 \times r^h_{net}$;

$C^{h*}_{soft} \leftarrow e^{h_5 \times (k)^2} \times r^h_{soft}$;

$C^{h*} \leftarrow C^{h*}_{cpu} + C^{h*}_{mem} + C^{h*}_{disk} + C^{h*}_{net} + C^{h*}_{soft}$; // C^{h*} denotes the total horizontal scaling cost;

if ($C^{h*} > C^{v*}$) //compare C^{v*} and C^{h*};

return ('v') ;

else

return ('h'); }

5.2. Same Cost for Extension and Better Performance Chosen First (SCBP)

In many cases, companies want to pay a certain cost to expand their information system effectively. However, they do not have a clear idea that which scaling method is better. In this section, a selection algorithm is proposed. Compare the maximum performances that the two scaling methods can achieve under a given certain cost, and choose a better one. The corresponding pseudo-codes are as follows:

Algorithm 2 SCBP *algorithm*

input: C; h_1, h_2, h_3, h_4, h_5; $r^h_{cpu}, r^h_{mem}, r^h_{disk}, r^h_{net}, r^h_{soft}$; v_1, v_2, v_3, v_4, v_5; $r^v_{cpu}, r^v_{mem}, r^v_{disk}, r^v_{net}, r^v_{soft}$;

// C denotes cost expected, other parameters just as the definition in the paper;

output: 'v' , 'h'; // v denotes vertical scaling, h denotes horizontal scaling;

Steps:

{ ……; } //the process is similar to algorithm 1, no repeat here;

5.3. CPU Preference and Less Cost Chosen First (CPLC)

Due to the special requirements of information systems, they may pay special attention to a particular component. For example, computational information systems mainly concern with the performance of CPU, and storage ones mainly care about the performance of disk, *etc*.,. Under these circumstances, a method satisfying special performance preference is given. The main idea is to compare the two kinds of cost under the given prerequisite of satisfying both certain extension times of comprehensive performance and a special component, and to choose the one with lower cost. Our algorithm takes CPU as an example here. It can also be applied to other components. The corresponding pseudo-codes are as follows:

Algorithm 3 CPLC *algorithm*

input: $k, k_{cpu}; P_1, P_2, P_3, P_4, P_5; h_1, h_2, h_3, h_4, h_5; r^h_{cpu}, r^h_{mem}, r^h_{disk}, r^h_{net}, r^h_{soft}; v_1, v_2, v_3, v_4, v_5; r^v_{cpu}, r^v_{mem}, r^v_{disk}, r^v_{net}, r^v_{soft}$ //k denotes the scaling times of system performance, k_{cpu} denotes the scaling times of CPU performance, P_1, P_2, P_3, P_4, P_5 denotes the performance of CPU, memory, disk, network and software performance, respectively; other parameters just as the definition in the paper;

Num ; //number of desired extension schemes;

output: '*v*' , '*h*';

Steps:

{ $Para \leftarrow \emptyset$, i=0; // Initialization;

while(i<*Num*)

{ $(k_1, k_2, k_3, k_4, k_5) \leftarrow$*find_a_random_solution*($k, k_{cpu}, P_1, P_2, P_3, P_4, P_5, Num$); //find a random solution by using (7);

if(k_1>=1$\wedge k_2$>=1$\wedge k_3$>=1$\wedge k_4$>=1$\wedge k_5$>=1)

{ $Para \leftarrow Para + \{(k_1, k_2, k_3, k_4, k_5)\}$;

i++; } }

$min\_cost\_C^v \leftarrow \infty$;

for $\forall$ $(k_1, k_2, k_3, k_4, k_5) \in Para$

{ $C^v \leftarrow e^{v_1 \times k_1} \times r^v_{cpu} + e^{v_2 \times k_2} \times r^v_{mem} + v_3 \times k_3 \times r^v_{disk} + v_4 \times k_4 \times r^v_{net} + v_5 \times k_5 \times r^v_{soft}$;

$min\_cost\_C^v \leftarrow min(C^v, min\_cost\_C^v)$; } // the minimal cost for vertical scaling;

$min\_cost\_C^h \leftarrow \infty$;

for $\forall$ $(k_1,k_2,k_3,k_4,k_5) \in Para$

{ $C^h \leftarrow h_1 \times k_1 \times r^h_{cpu} + h_2 \times k_2 \times r^h_{mem} + h_3 \times k_3 \times r^h_{disk} + h_4 \times (k_4)^2 \times r^h_{net} + e^{h_5 \times (k_5)^2} \times r^h_{soft}$;

$min\_cost\_C^v \leftarrow min(C^v, min\_cost\_C^v)$;} // the minimal cost for horizontal scaling;

if ($min\_cost\_C^h > min\_cost\_C^v$)

return ('*v*');

else

return ('*h*');}

6. Case Studies

6.1. Actual Parameters

Based on the comprehensive survey of devices, functions and prices of a large amount of information systems, we take company A and B as example to do specific analysis. Their configuration are: CPU(mononuclear*1.6GHz), memory (1*2G), hard disk(256G), network(bandwidth:5M) and software system(concurrent users:500).

Table 6.1. Cost of Company A and B for Scaling (Unit：RMB)

Times of scaling	Company A (Vertical Scaling)					Company B (Horizontal Scaling)				
	CPU	Memory	Disk	Network	Software System	CPU	Memory	Disk	Network	Software System
2	850	750	3800	6000	1000	1000	800	900	2500	11000
4	1000	2600	8000	12000	2000	2000	1600	1800	9500	12000
8	1500	1300	15000	24000	5000	4000	3200	3500	38000	15000
16	3500	3000	32000	48000	8000	8000	6400	7000	155000	22000
32	17000	15000	65000	95000	16000	16000	13000	14000	600000	50000
64	420000	360000	128000	195000	32000	32000	26000	29000	2500000	250000
128	254000000	217000000	2888000	384000	64000	64000	51000	58000	9800000	6000000

After fitting and analyzing the data of vertical scaling in table 6.1, we had the following results. For company A, the cost of CPU and memory increase exponentially and match with $y = e^{0.1 \times x} \times 700$, $y = e^{0.1 \times x} \times 600$, respectively. And the cost of disk, network, and software increase linearly and match with $y = 2.5 \times x \times 800$, $y = 3 \times x \times 1000$, $y = 5 \times x \times 100$, respectively. According to the assumptions in section 3.2, the actual parameters of the system are: v_1=0.1, r^v_{cpu}=700, v_2=0.1, r^v_{mem}=600, v_3=2.5, r^v_{disk}=800, v_4=3, r^v_{net}=1000, v_5=5, r^v_{soft}=100.

After fitting and analyzing the data of horizontal scaling in table 6.1, we had the following results. For company B, the cost of CPU, memory, and disk increase linearly and match with $y = 0.5 \times x \times 1000$, $y = 0.5 \times x \times 800$ and $y = 0.5 \times x \times 900$, respectively. And the cost trends of network and software fit with $y = 0.6 \times (x)^2 \times 1000$, $y = e^{0.05 \times x} \times 10000$, respectively. According to the assumptions in section 3.2, the actual parameters of the system are: h_1=0.5, r^h_{cpu} =1000, h_2=0.5, r^h_{mem}=800, h_3=0.5, r^h_{disk}=900, h_4=0.6, r^h_{net}=1000, h_5=0.05, r^h_{soft}=10000.

Therefore, the statistics and analysis in the above case studies show that our assumptions in section 3.2 are reasonable.

6.2. Applications

Company C prefers the performance of CPU and memory, so the performance of the two components should be first meted during system scaling. In this paper, we analyze the cost when k_1=k_2=5k and k_1=k_2=10k, and we can obtain the corresponding k_3, k_4, and k_5 by algorithm 3. The results in Figure 7, and Figure8, show that the cost for vertical/horizontal scaling increases along with the extension times of comprehensive performance increases. We call the intersection of the two lines as equivalent point, and its value is about 9 and 5 in Figure 7, and Figure 8, respectively. On the left side of equivalent point both in Figure 5, and Figure 6, the cost of vertical scaling is less than the cost of horizontal scaling, and on the other side, the result is opposite. The results show that: in general, it is better to choose vertical scaling when the times of performance extension is smaller (on the left side of equivalent point). And the value of equivalent point decreases with the increase of k_1 and k_2, and the range for choosing vertical scaling becomes narrow while the range for choosing horizontal scaling becomes widen.

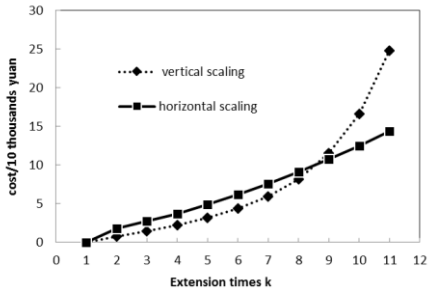

Figure 5. Cost Trend of System Vertical/Horizontal Scaling (k_1=k_2=5k) (Left)

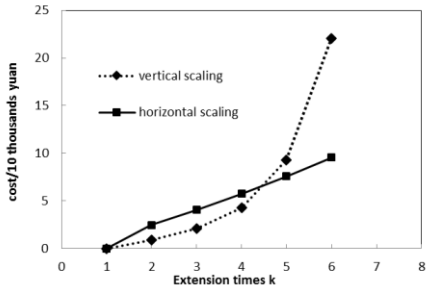

Figure 6. Cost Trend of System Vertical /Horizontal Scaling (k_1=k_2=10k) (Right)

7. Conclusion

In this paper, we built a quantitative function relation between the cost and comprehensive performance. We also have designed three optimal selection algorithms to choose the reasonable scaling methods. The case studies show that the formulas we assumed are reasonable, and the algorithms are feasible and have guiding significance for information systems scaling. We can modify the function between the cost and times of performance extension according to actual situations and our algorithms are still valid. In the future, we will study on how to mix the two scaling methods rather than only choose vertical or horizontal scaling when extending our information systems.

Acknowledgement

This work was supported by the National High-Tech Research and Development Plan of China under grant No. 2009AA012201; the National Natural Science Foundation of China under grant No. 61272107 and No. 61202173; the Program of Shanghai Subject Chief Scientist under grant No. 10XD1404400; the Huawei Innovation Research Project under grant No. IRP-2013-12-03; the State Key Laboratory of High-end Server & Storage Technology under grant of 2014HSSA10; the Natural Science Foundation of Jiangxi Province under grant No. 20151BAB207040; Scientific Research Foundation of the Education Department of Jiangxi Province under grant No. GJJ150426.

References

[1] L. Chen, "Parallel computer architecture", Beijing: Higher Education Press, **(2002).**
[2] M. D. McCool, "Scalable programming models for massively multicore processors", Proceedings of the IEEE, vol. 96, no. 5, **(2008)**, pp. 816-831.
[3] Y. Su and D. Ye, "Accelerating inclusion-based pointer analysis on heterogeneous CPU-GPU systems", Proceedings of the 20th International Conference on High Performance Computing, **(2013)** pp. 149-158.
[4] S. Hong, "Semiconductor memory scaling and beyond", Proceedings of IEEE Solid State Circuits Conference, **(2012)**, pp. 5-8.
[5] O. Mutlu, "Memory scaling: A systems architecture perspective", Proceedings of the 5th IEEE International Conference on Memory Workshop, **(2013)**, pp. 21-25.
[6] G. Zhang and K. Li, "Accelerate RDP RAID-6 scaling by reducing disk I/O and XOR operations", IEEE Transactions on Computers, vol. 64, no. 1, **(2015)**, pp. 32-44.
[7] A. Sun and Z. Jin, "Review on software architecture", Journal of Software, vol. 13, no. 7, **(2002)**, pp. 1228-1237.
[8] H. Mei and R. Shen, "Progress of research on software architecture", Journal of Software, vol. 17, no. 6, **(2006)**, pp. 1257-1275.
[9] L. Zhang and H. Gao, "Software architecture evaluation", Journal of Software, vol. 19, no. 6, **(2008)**, pp. 1328-1339.
[10] U. F. Minhas and R. Liu, "Elastic scale-out for partition-based database systems", Proceedings of the 28th IEEE International Conference on Data Engineering Workshop, **(2012)**, pp. 281-288.

