Many-objective Evolutionary Optimization of Web 3-tier Systems Resource Allocation in the Inter-Cloud Environment

Atsushi Saito[1] **and Masaharu Munetomo**[2]
[1] Graduate School of Information Science and Technology
[2]Information Initiative Center, Hokkaido University, Sapporo, Hokkaido, Japan

Abstract - *This paper presents a many-objective optimization framework for web 3-tier systems resource allocation in the inter-cloud environment considering seven objective functions. The selection and design of those functions is based on collaboration with real-world web systems engineers. Due to the large number of objective functions, we employ NSGA-III, a multi-objective evolutionary algorithm capable to solve many-objective optimization problem, and discuss obtained Pareto optimal solutions through numerical experiments.*

Keywords: many-objective optimization, cloud computing, web 3-tier systems, cloud broker, inter-cloud environment

1 Introduction

Currently, cloud service providers offer a wide variety of cloud services globally, and it is difficult for application engineers to choose the best cloud services/resources deployment/configuration from them. In this paper, we propose a new mathematical model for many-objective optimization to find the best web 3-tier system configuration satisfying web application engineers' requirements in a cloud broker system.

A series of papers have been published related to cloud broker services. S. Sundareswaran et al. [1] built a system that could find a cloud instance that matches the engineer's needs based on CSP-index, which is encoded into a binary string. They used CSS-query algorithm based on Hamming distances between the string of the user's request and that of cloud services. This method quickly finds a solution; however, that solution corresponds to only one instance closest to the user's request.

Garg at el. [2] evaluate cloud instances using Service Measurement Index (SMI), one of the evaluation criteria of cloud service proposed by CSMIC. To evaluate cloud services using SMI, they employ Analytic Hierarchy Process (AHP). AHP is a decision-making method based on a hierarchical measurement. AHP can find cloud services according to the user's requests; however, this method does not consider a whole web system consisting of multiple instances. It is necessary to deal with whole web systems consisting of multiple instances to build practical application systems.

Kawakatsu et al. [3] focused on web 3-tier systems employing resources in the global inter-cloud environment. As objective functions, they consider not only the deployment cost for the users but also the energy conservation, which is important from the viewpoint of data center management.

Pawluk et al. [4] proposed a cloud broker system architecture and formulated the optimization problem for it; however, their objective functions and optimization techniques are not sufficient to cover the whole Pareto front because they employed a weighted sum method that can only find one of the solutions from the front.

In this paper, we discuss a new mathematical model formulated as a many-objective optimization problem in cloud broker systems, where the objective functions are based on discussions with IT corporations building practical web-systems. In order to obtain Pareto optimal solutions of the problem, we employ a multi-objective evolutionary algorithm NSGA-III (Non-dominated Sorting Genetic Algorithm III) that can solve many-objective optimization problems. We demonstrate the effectiveness of our approach through numerical experiments.

2 Mathematical Model

The problem that a cloud broker system has to solve is a constrained multi-objective optimization problem involving a number of objectives such as cost, performance, availability, and also constraints such as regions to deploy virtual machines according to available resources.

When we define the number of the variable dimension as *n*, the number of objective functions as *m*, a solution of the constrained multi-objective optimization problem is represented as an integer vector of service (cloud resources such as virtual machines) selection $\boldsymbol{x} = (x_1, x_2, \cdots x_n)^T \in I^n$, their feasible region is $S \subset I^n$, where x_i denotes a service ID to be selected for system component i. As we define objective functions as

$\boldsymbol{f} = (f_1, f_2, \cdots, f_m)^T$, we can formulate the multi-objective optimization problem as follows.

$$\min or \max \boldsymbol{f_i}(\boldsymbol{x}) \quad (i = 1,2,\cdots,m), \quad \boldsymbol{x} \in S$$

In this paper, we define seven objective functions through discussions with engineers of web systems. There are two categories of objective functions in a cloud broker system. The first one is for users of the target application. The second one is for the developer to deploy and manage it. Users generally want their requests to be quickly and certainly processed, whereas engineers need to ensure availability of the system and minimize the cost to deploy it.

We set the seven objective functions with two categories as follows:

- From web systems engineers' point of view : *Response time, Reject rate*
- From cloud providers' point of view : *Operation cost, Configuration cost, Throughput, Availability, Complexity*

A) Operation cost

Application systems developers need to minimize the running cost of the system. f_{op_cost} is the cost to maintain a system consisting of cloud services $\boldsymbol{x}$ for a month. The cost of Web servers, Application (App) servers, Database (DB) servers, and Load Balancers (LB) are denoted by $P_{Web}(\boldsymbol{x})$, $P_{App}(\boldsymbol{x})$, $P_{DB}(\boldsymbol{x})$, $P_{LB}(\boldsymbol{x})$, respectively. The Monitoring cost is denoted by $P_{\text{watch}}(\boldsymbol{x})$. f_{op_cost} is described below.

$$f_{op\_cost}(\boldsymbol{x}) = P_{Web}(\boldsymbol{x}) + P_{App}(\boldsymbol{x}) + P_{DB}(\boldsymbol{x}) + P_{LB}(\boldsymbol{x}) + P_{watch}(\boldsymbol{x})$$

B) Configuration cost

When developers configure (or re-configure) the systems, they need to consider the cost to configure or re-configure them. The configuration cost is defined as follows by the sum of the cost of each component $C_{Web}(\boldsymbol{x})$, $C_{App}(\boldsymbol{x})$, $C_{DB}(\boldsymbol{x})$, $C_{LB}(\boldsymbol{x})$ of Web, App, DB, and LB.

$$f_{conf\_cost}(\boldsymbol{x}) = C_{Web}(\boldsymbol{x}) + C_{App}(\boldsymbol{x}) + C_{DB}(\boldsymbol{x}) + C_{LB}(\boldsymbol{x})$$

C) Throughput

Developers expect that the system can process as many requests as possible. f_{perf} represents how many requests the system can process in a second. In this paper, we calculate f_{perf} by using a queuing network.

Harada et al. [5] did a performance prediction of multi-tier systems by using queuing networks. Urgaonkar et al. [6] predicted performance of multi-tier system with 95% accuracy. We use the analytical model based on a Mean Valued Algorithm (MVA) [6] to predict the performance of a web 3-tier system. The characteristics of this analytical model consider that by setting a limit of simultaneous sessions, user requests are rejected by the system. The probability for this event is defined as P_m^{Drop} where m is a tier number ($m=1$ for Web, $m=2$ for App, and $m=3$ for DB tier). By using P_m^{Drop}, we can summarize the performance accurately.

We have to take the following three steps to calculate throughput τ from this analytical model.

1. Throughput λ of the requests to the web system by using the Mean Value Algorithm (MVA), we set $P_m^{Drop} = 0$ if there is no limits of a simultaneous sessions.
2. $P_m^{Drop}(\boldsymbol{x})$ is calculated by using a limited open queue Q_x ($M/M/s/K_m$) and an arrival rate λV_m, where K_m is the maximum length of the queue and V_m is visit ratio of the queue in the m-th tier.
3. We calculate the throughput τ by using the MVA considering $P_m^{Drop}(\boldsymbol{x})$.

From the above, we set f_{thrpt} by using the obtained throughput as follows:

$$f_{thrpt}(\boldsymbol{x}) = \tau.$$

D) Response time

Response time is a key factor for the users' satisfaction of the target application. MVA can predict not only the throughput but also the response time $\overline{R(\boldsymbol{x})}$ based on the average response time of the system $\boldsymbol{x}$ as follows:

$$f_{rsp\_time}(\boldsymbol{x}) = \bar{R}(\boldsymbol{x}) = \sum_{m=1}^{M} \bar{R}_m,$$

where $\bar{R}_m$ is the average delay at each tier queue.

E) Reject rate

Users satisfaction may also degrade when their requests are rejected. We calculate the rejection rate P_m^{Drop} and calculate the sum of P_m^{Drop} to consider all the rejections in the system. Hence, the objective function f_{reject} is calculated as follows:

$$f_{reject}(\boldsymbol{x}) = \sum_{x=1}^{M} P_m^{Drop}(\boldsymbol{x}).$$

F) Availability

Developers need to construct a fault-tolerant system. The objective function f_{avail} calculates availability of the system c_i, which is defined as the proportion of working time that the system works properly.
In this paper, we use a Bayesian network for analyzing the probabilities of availability. The probability that the system $\boldsymbol{x}$ is calculated as follows, where $T=T(\boldsymbol{x})$ means the proportion of working time:

$$f_{avai}(\boldsymbol{x}) = P(T(\boldsymbol{x})).$$

A characteristic of web 3-tier systems is that all tier (Web tier, App tier and DB tier) have to work correctly for the system. We set the probability that Web, App, DB tiers works are *P(W)*, *P(A)* and *P(D)*, respectively. If all tiers work, the probability becomes the following:

$$P(X) = P(W)P(A)P(D).$$

We set the probability that each tier doesn't work as $P(\bar{W})$, $P(\bar{A})$ and $P(\bar{D})$. $P(\bar{W})$, $P(\bar{A})$ and $P(\bar{D})$ are analyzed by a Bayesian network. We have to consider 3 cases. First, the probability that the tier halts by effect of the same layer. This probability is $P(\overline{W_U})$. Second, the probability that the tier halt by effect of the lower layer. This probability is $P(\overline{W_U})$. Third, the probability that the tier halt by effect that the lower layer effects another upper layer. This probability is $P(\overline{W_{U\to S}})$. The probability of $P(\bar{W})$ can be described as follows:

$$P(\bar{W}) = P(\overline{W_S}) + P(\overline{W_U}) - P(\overline{W_{U\to S}}).$$

Next, $P(\overline{W_S}), P(\overline{W_U}), P(\overline{W_{U\to S}})$ are analyzed. At the layer of event of tiers, if there is a fault on App tier, the Web tier may not present information to users. If there is a fault on DB tier, App tier cannot process user requests and the Web tier will not show correct information. In short, the Web tier depends on the App tier, and the App tier depends on the DB tier. We define the probability that the Web tier and App tier fails independently according to the probability *P(w)* for Web and *P(a)* for App. Then we can calculate the following probabilities.

$$P(\overline{W_S}) = P(w) + P(\bar{W}|\bar{A})P(\bar{A})$$
$$P(\overline{A_S}) = P(a) + P(\bar{A}|\bar{D})P(\bar{D})$$
$$P(\overline{D_S}) = P(d)$$

Next, the layer of event of instances presents the probability that if instances in a tier are down, how much effect this will cause on the tier. Tiers are considered as failure if some instances are down. How many instances can maintain the system is a critical issue. If r instances in n instances are needed to maintain the Web tier, the probability that the Web tier works is as follows:

$$P(\bar{W}_U) = 1 - \left(\sum_{i\in\{r,\dots,n\}} \left(\sum_{S\in C(n,i)} \left(\prod_{s\in S} p(w_s) \right) \left(\prod_{t\in\{1,\dots,n\}\setminus S} p(\overline{w_t}) \right) \right) \right)$$

.

In this paper, we leave sharing an instance out of consideration, because our Web 3-tier model needs to separate tiers. Therefore, the value $P(\overline{W_{U\to S}}) = 0$.
Next, we have to consider the layer of event of availability-zone. Availability-zone means the area of data-centers which are situated in the same area like the state of Virginia. At the same availability-zone, the probability that instances in the area are down simultaneously is high, because of a natural disaster and similar causes. We set the probability $P(w_1)$ that an instance w_1 is down. We can calculate $P(w_1)$ as follows:

$$P(w_1) = 1 - P(\overline{w_1}),$$
$$P(\overline{w_1}) = P(\overline{w_{1S}}) + P(\overline{w_{1U}}) - P(\overline{w_{1U\to S}}).$$

Because we don't consider sharing instances, $P(\overline{w_{1S}}) = 0$ and $P(\overline{w_{1U\to S}}) = 0$.
We define the probability that instance w_1 is down independently as $P(w_1)$. Then we can describe $P(\overline{w_{1U}})$ as follows:

$$P(\overline{w_{1U}}) = P(\overline{w_1}|\overline{Az_1})P(\overline{Az_1}) + P(\overline{w_1}),$$

where $P(\overline{Az_1})$ is the probability of stopping in an availability-zone Az_1.

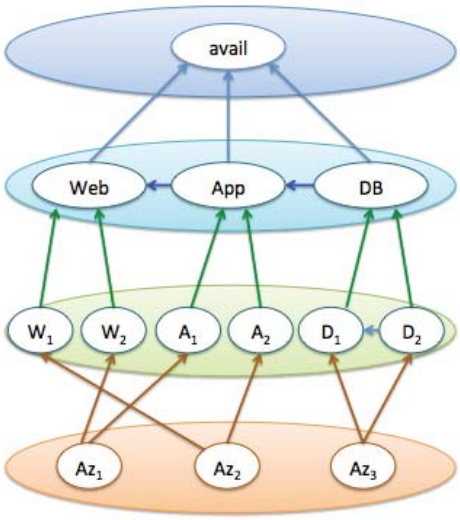

Fig. 1. A Bayesian network to model of availability

G) Complexity

Engineers do not prefer complex systems, because they need more effort for maintenance and management. In this paper, we define the objective function of complexity f_{cmpl} as the number of servers $S_n(\boldsymbol{x})$ and the difference of instances $D_n(\boldsymbol{x})$. And w_1 and w_2 are weights for complexity.

$$f_{cmpl}(\boldsymbol{x}) = \sum_{n=1}^{N} w_1 * S_n(\boldsymbol{x}) + w_2 * D_n(\boldsymbol{x})$$

3 Multi-objective Evolutionary Algorithm

We employ NSGA-Ⅲ[7], which is an evolutionary multi-objective algorithm to efficiently solve many-objective optimization problems of the cloud broker systems.

NSGA-Ⅲ is an extended version of NSGA-II which is one of the most popular multi-objective genetic algorithms based on non-dominated sorting using a crowding distance scheme to obtain the Pareto front. NSGA-Ⅲ uses a reference point selection scheme instead of the crowding distance scheme, because it requires a lot of computational cost, and the reference point selection can greatly reduce such cost. Fig.2 shows the algorithm of NSGA-III.

Input: H reference points Z^r, parent population P_t
Output: P_{t+1}
1: $S_t = \emptyset$, $i = 1$
2: Q_t = Recombination+Mutation(P_t)
3: $R_t = P_t \cup Q_t$
4: $(F_1, F_2, \ldots)$ = Non-dominated-sort(R_t)
5: **repeat**
6: $S_t = S_t \cup F_i$ and $i = i + 1$
7: **until** $|S_t| \geq N$
8: Last front to be included: $F_l = F_i$
9: **if** $|S_t| = N$ **then**
10: $P_{t+1} = S_t$, break
11: **else**
12: $P_{t+1} = \cup_{j=1}^{l-1} F_j$
13: Points to be chosen from F_l: $K = N - |P_{t+1}|$
14: Normalize objectives
15: Associate each member **s** of S_t with a reference point: $[\pi(\mathbf{s}), d(\mathbf{s})]$ =`Associate`(S_t, Z^r) % $\pi(\mathbf{s})$: closest reference point, d: distance between **s** and $\pi(\mathbf{s})$
16: Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j)\ ?\ 1\ :\ 0)$
17: Choose K members one at a time from F_l to construct P_{t+1}: `Niching`$(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
18: **end if**

Fig. 2. Algorithm of NSGA-Ⅲ (from [7])

A fast non-dominated sorting counts up dominant and dominated gene simultaneously. Then if there is no dominant gene for a specific a gene, such specific gene is ranked as 1. And if genes of rank1 are removed, there is no dominant gene by these genes, they are rank 2. We repeat this operation until there is no gene. This operation achieves faster sorting compared with NSGA-II.

Elitism is the method that certainly leaves better solutions from parent and child population. Parent population is used for preservation. Child population is used for searching. At generation t, parent population P_t, child population Q_t, and compound population R_t made from P_t and Q_t. Q_t is renewal by genetic operation (selection, crossover, mutation). Then Q'_t and P_t are combined, it makes R_t. P_{t+1} are selected from R_t. Then go to next generation.

Reference Points are arranged regularly in the Pareto front. The number of selected genes depends on the result of fast non-dominated sorting. The number of reference points H is set as $H={}_{M+p-1}C_p$, where M is the number of objective functions, and p is a parameter to be set as H becomes the same as the population size. In reference point selection, we use a niche count ρ. The niche count ρ is set on each reference point. ρ counts the number of nearest genes compared with the distance from other reference points. Genes are selected from those with smallest ρ. By using a reference point selection, even if the number of objective functions increases, we can get a wide range of optimal solutions and its computational cost remains low.

We encode a string $\boldsymbol{x}$ as a vector of integers each of which represents a service ID selected and deployed for a virtual machine for Web, App, and DB instance of the web 3-tier system as

$$(x_1^{Web}, x_2^{Web}, \cdots, x_{k_{Web}}^{Web}, x_1^{App}, x_2^{App}, \cdots, x_{k_{App}}^{App}, x_1^{DB}, x_2^{DB}, \cdots, x_{k_{DB}}^{DB})$$

where k_{Web} is the number of virtual machines for Web servers, k_{App} is that for App servers, and k_{DB} is that for DB servers. We employ one-point crossover and simple mutation applied to the integer vectors.

4 Experiments

We perform numerical experiments of the proposed framework employing NSGA-III. As for cost, performance, and availability measures, we use instance metadata of the Amazon Web Service (AWS) and set parameters of the GA, MVA, Availability, Complexity as shown in Table 1.

Table 1. Parameters setting in the experiments

GA	MVA	Availability	Complexity
Pop. size: 1000	100 Users	p(w), p(a), p(d): 0.001	w1: 0.5
1000 generations	Think time: 7.5s	$p(w_x)$: 0.01	w2: 0.5
Pcross: 0.01	Average arrival time: 1 req/s	$p(w_x \mid AZ_{JP})p(AZ_{JP})$: 0.008	
Pmutation: 0.03	Average service rate: 3.318/vCPU	$p(w_x \mid AZ_{US})p(AZ_{US})$: 0.04	
	Reject limit: 100 req	$p(w_x \mid AZ_{EU})p(AZ_{EU})$: 0.03	

Using the parameters shown in Table 1, we get the last population and its evaluated value by objective functions. Figure 6 to 11 show the obtained population after performing NSGA-III from the viewpoint of 7 different objective functions (All horizontal axes are operation cost). The white on black circles are the last population and the red large crosses are rank 1.

4.1 Configuration Cost

The vertical axis of Figure 6 represents the configuration cost. First of all, one can see that all genes are red crossed. This means that all genes are rank 1 and Pareto optimal solutions. One may think that apparently on this graph inferior solutions are shown. However, considering other objective functions, those are all Pareto optimal.
Both, operation cost and configuration cost have to be minimized, therefore many solutions must gather at the lower left corner. This characteristic can be well observed from Figure 6.

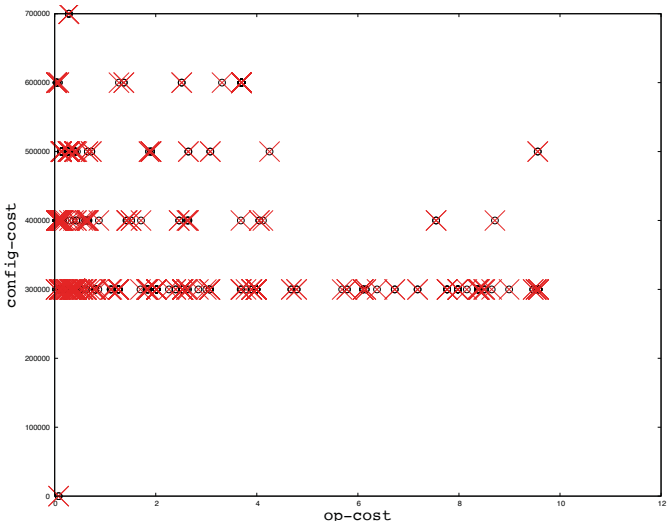

Figure 6. Configuration cost - Operation cost

4.2 Throughput

The vertical axis of Figure 7 represents the configuration cost. As in Figure 6, all solutions are rank 1. The characteristic of Figure 7 is that Figure 7 draws a curved line at from the low left corner to high right corner. This represents a trade-off between operation cost and throughput. This means that inexpensive systems have a low performance, which seems reasonable.

4.3 Response time

The vertical axis of figure 8 is response time. Figure 8 draws a curved line at low right to high left. It is a trade-off between operation cost and response time.

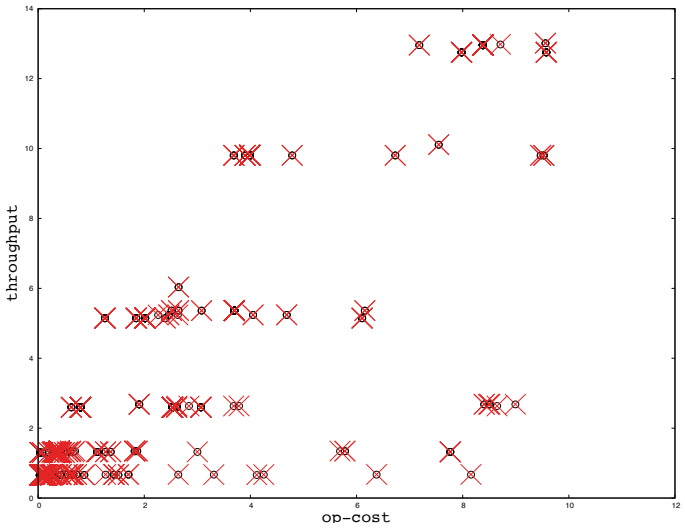

Figure 7. Throughput - Operation cost

4.4 Reject rate

The vertical axis of Figure 9 is reject rate. Same as Figure 6, all solutions are rank 1. A trade-off exists between operation cost and reject rate.

4.5 Availability

The vertical axis of Figure 10 is availability. Same as Figure 6, all genes are rank 1. The characteristic of Figure 10 is that there seems to be no tradeoff between operation cost and availability. Because a system can get high availability when such system has a lot of cheap instances, high availability and low operation cost are compatible.

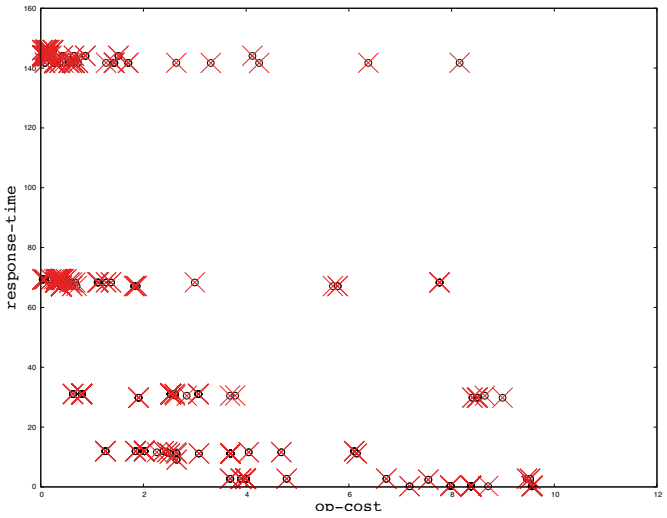

Figure 8. Response time - Operation cost

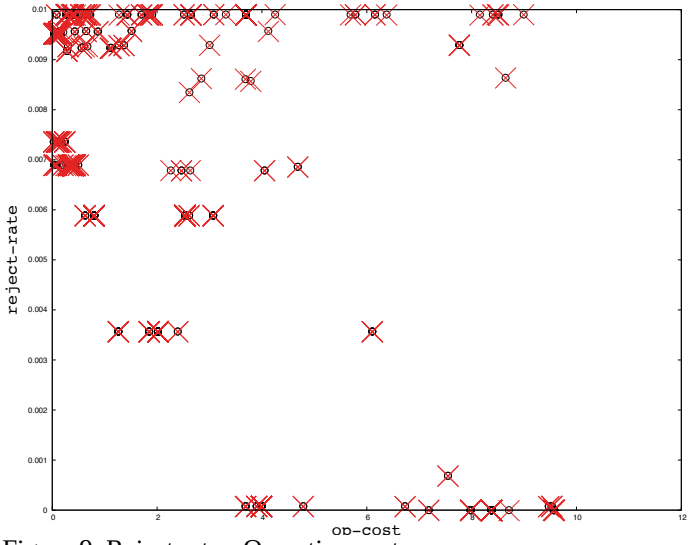

Figure 9. Reject rate - Operation cost

4.6 Complexity

The vertical axis of Figure 11 represents complexity. As in Figure 6, all genes are rank 1. The characteristic of Figure 11 is that, as in Figure 10, there is no tradeoff shown between operation cost and complexity. Even for low operation cost, a gene can generate a complex system using cheap instances.

4.7 Examples of solutions obtained

Table 2 displays a solution example, and Table 3 shows the evaluation of it. This solution has only one instance in each tier. It certainly minimizes configuration cost and complexity. However, it probably decreases availability. So, this solution may take ap-northeast-1 (Tokyo region). This region has a relatively high availability. In addition, high performance is obtained by instance type (g2.8xlarge, 32vCPU). Figure 12 shows that this solution is one of the Pareto optimal solutions on 3 objective functions.

Table 4 also displays a solution example. Table 5 shows the evaluation of this example. This solution has a low operation cost and a high availability. This is because 2 to 3 cheap instances are used. Users who don't need high performance may choose this system. Figure 13 shows that this solution is Pareto optimal for operation cost and availability.

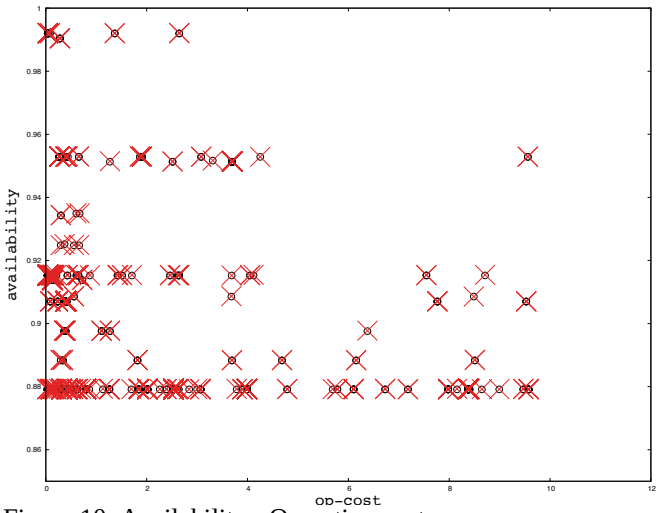

Figure 10. Availability - Operation cost

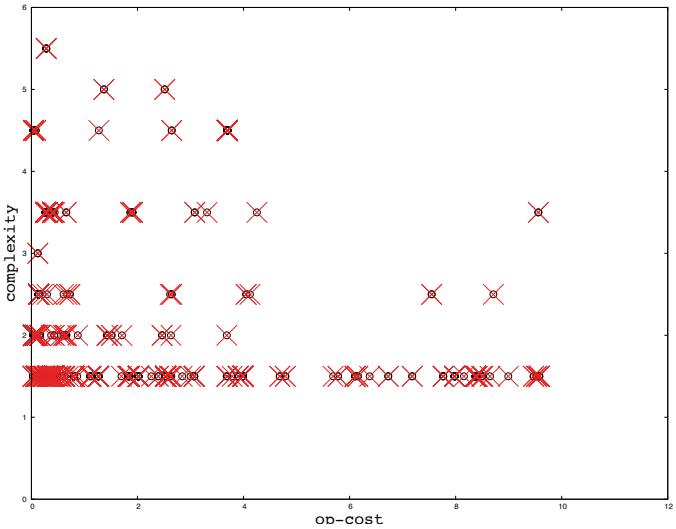

Figure 11. Complexity - Operation cost

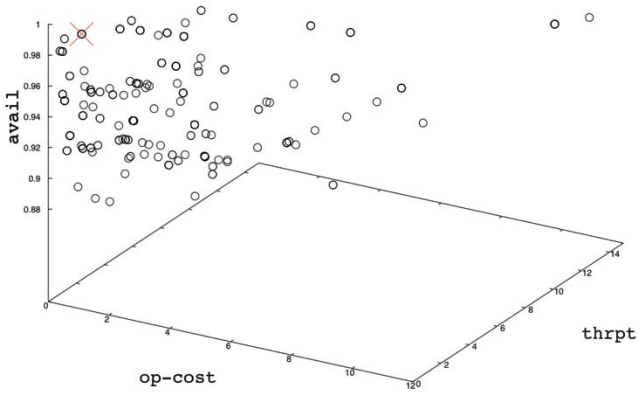

Figure 12. 3D Plot of solution #1 (large red cross)

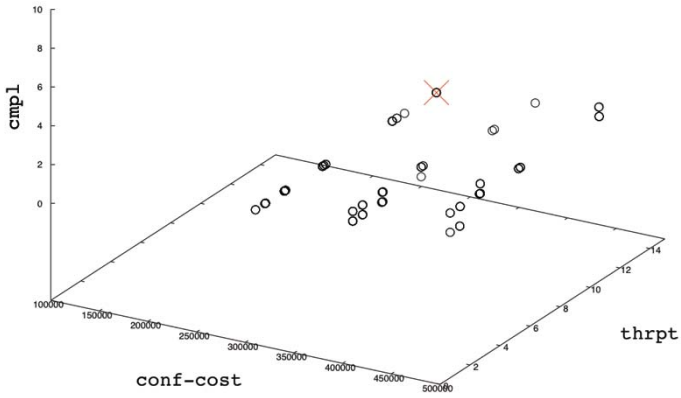

Figure 13. 3D Plot of solution #2 (large red cross)

Table 2. Solution #1, location and instance type

	location	type
Web	ap-northeast-1	g2.8xlarge
App	ap-northeast-1	g2.8xlarge
DB	ap-northeast-1	g2.8xlarge

Table 3. Evaluation of solution #1

op-cost	conf-cost	thrpt	resp-time	rjct-rate	avail	cmplx
10.77	300000	12.75	0.342	$8.5*10^{-23}$	0.970	1.5

Table 4. Solution #2, location and instance type

	location	type	location	type	location	type
Web	ap-northeast-1	t2.medium	ap-northeast-1	t2.medium		
App	ap-northeast-1	t2.medium	ap-northeast-1	t2.medium	ap-northeast-1	t2.medium
DB	ap-northeast-1	t2.medium	ap-northeast-1	t2.medium		

Table 5. Evaluation of solution #2

op-cost	conf-cost	thrpt	resp-time	rjct-rate	avail	cmplx
0.480	600000	1.340	67.1	0.0099	0.987	5.0

5 Conclusion

We proposed a mathematical model of resource optimization to deploy Web 3-tier systems in the inter-cloud environment. In our mathematical model, seven objective functions are considered based on discussions with web systems engineers to realize a practical cloud broker system. Through numerical experiments employing NSGA-III to solve the many-objective optimization problem, we could get a variety of Pareto optimal solutions that can be a guidance to web system engineers to select optimal configuration of cloud resources from a variety of services configurations such as instance types and regions.
As future work, we will extend our framework to a cloud broker system for general-purpose systems deployment that may also consider other cloud services offering given by the cloud service providers such as DB (database) as a service, storage services, bigdata processing and machine learning services provided though web service APIs. We believe that our framework is effective in selecting optimal configurations and combinations of such wide-spectrum of cloud services.

6 Acknowledgement

We thank the engineers of SCSK cooperation in Japan for discussion on requirements of the web application systems.

7 References

[1] Sundareswaran, Smitha, Anna Squicciarini, and Dan Lin. A brokerage-based approach for cloud service selection, *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. pp.558-565, IEEE, 2012.

[2] Garg, Saurabh Kumar, Steve Versteeg, and Rajkumar Buyya. A framework for ranking of cloud computing services, *Future Generation Computer Systems*, vol.29, no.4, pp.1012-1023, 2013.

[3] Takashi Kawakatsu, Masaharu Munetomo. Multi-objective resource allocation of web systems in distributed cloud environment considering, *IPSJ SIG Reports*, Vol. 2013-MPS-96, No. 9, pp. 1-6, 2013. (in Japanese)

[4] P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski. Introducing Stratos: A cloud broker service, *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 891–898, 2012.

[5] Masashi Harada, Miyu Kawamura. Performance prediction of Web 3-tier systems based on queueing network model, *Proceedings of the IPSJ national conference*, pp. 289-290, 2010. (in Japanese)

[6] Urgaonkar, Bhuvan, et al. An analytical model for multi-tier internet services and its applications. *ACM SIGMETRICS Performance Evaluation Review*, Vol. 33. No. 1, pp.291-302, 2005.

[7] Jain, Himanshu, and Kalyanmoy Deb. An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization, *International Conference on Evolutionary Multi-Criterion Optimization*, Springer Berlin Heidelberg, pp.307-321, 2013.