2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

An Innovative Self-adaptive Configuration Optimization System in Cloud Computing

Jing Jiang
Decision Systems & E-Service Intelligence Research Lab.,
Center for Quantum Computing and Intelligent System,
Faculty of Engineering and Information Technology,
University of Technology Sydney, Sydney,
P.O.Box123, Broadway, NSW2007, Australia
jing.jiang-1@student.uts.edu.au,

Jie Lu, Guangquan Zhang
Decision Systems & E-Service Intelligence Research Lab.,
Center for Quantum Computing and Intelligent System,
Faculty of Engineering and Information Technology,
University of Technology Sydney, Sydney,
P.O.Box123, Broadway, NSW2007, Australia
{jie.lu, guangquan.zhuang}@uts.edu.au,

Abstract— **Cloud computing has emerging as an extremely popular and cost-effective computational service model using pay-as-you-go executing environments that scale transparently to the user. However, cloud providers should tackle the challenge of configuring their systems to provide maximal performance while minimizing customer's cost of computing resources, which satisfy the customers' various workload requirements. To solve the above challenge, in this paper, we propose an innovative architecture of self-adaptive configuration optimization system which supports dynamic reconfiguration when workloads change. In addition, we develop an optimization algorithm by using genetic algorithm for this system. By using queuing theory and statistic techniques, we model and compute the SLAs metrics which are defined as the fitness function in the optimization algorithm. This optimization system can guide cloud customers to purchase appropriate resources and make decision of deployment configuration such as scale, scheduling and capacity.**

Keywords- cloud computing; optimization; cloud service configuration;SLAs; resource scheduling

I. INTRODUCTION

Cloud Computing will be the next generation in computation [1] . Possibility people can have everything they need on the cloud through internet. Moreover, cloud computing has emerging as an extremely popular and cost-effective computational service model for the on-demand provisioning of virtualized resources. Major commercial cloud vendors have obtained great success by providing the flexible "pay-as-you-go" or "on-demand" services [2] with a number of various service instance types and plans to meet customers computing needs, which are supported by large number of distributed physical resources. For example, Amazon EC2 provides three families (on-demand, reserved, spot), and each family includes several service instance types. Different service instance types operate at different quality of service (QoS) levels and have different price. The Amazon Elastic Computing Cloud (EC2) instance types and price for on-demand family are summarized as in table 1 [3] .

In this "pay-as-you-go" cloud model, the customer can purchase and increase or release resources by manual when they need rather than owing these devices, so that the users greatly reduce their capital expenses. However, most of these customers often do not design and construct the system. Consequently, they lack domain knowledge to monitor systems. Thus, this cloud scenario presents new challenges for customers to make decision of purchasing appreciate resources before their systems executing, especially when their application workload (i.g. application requests) is uncertain and dynamically changed, such as,

- *System configuration*: what types of cloud instances should be used? How big a system (how many CPUs, how many disks, how much network bandwidth) is needed to execute the workload within time and cost constraint?
- *Resource scheduling:* when the workload change (peak and non-peak), should increase or release the existing resources (elastic cloud)? How to reconfigure the system?
- *Workload management:* what kind of attributes of the workloads have? How to model the workload distributions and how to predict them?

TABLE 1. AMAZON EC2 INSTANCE TYPE AND PRICE

Instance Type	Sub Type	API Name	Cores	Cost ($/h) (windows)
Standard Instances	Small (default)	m1.small	1	0.12
	Large	m1.large	2	0.48
	Extra Large	m1.xlarge	4	0.96
High Memory Instances	Extra Large	m2.xlarge	2	0.62
	Double Extra Large	m2.2xlarge	4	1.24
	Quadruple Extra Large	m2.4xlarge	8	2.48
High-CPU Instances	Medium	c1.medium	2	0.29
	Extra Large	c1.xlarge	8	1.16

To address the above challenges, the cloud infrastructure service providers have to: (1) model workloads distribution well to anticipate spikes and account for diurnal patterns; (2)

model the correlation among system performance, configuration and workload to optimize the configuration and to maximize the performance and minimize cost and contention; (3) design self-adaptive configuration optimization system along with workload distribution change.

This paper is the first attempt to investigate the self-adaptive configuration optimization in cloud computing environment, considering multiple conflicting SLAs and supporting dynamic reconfiguration when workloads change. We propose an innovative architecture of this self-adaptive configuration optimization system. By using queuing theory and statistic techniques, we address the problem of SLAs metrics computaion and modeling based on utility function for a given workload distribution. We also develop an multiobjective genetic algorithm for this optimization system to realize the self-adaptive configuring. This optimization system can guide cloud customers to purchase appropriate resources and make decision of deployment configuration such as scale, scheduling and capacity.

The remainder of this paper is structured as follows. We first discuss some related work in Section II, including current cloud service SLAs computation and system performance metrics. In Section III, we present a high-level overview of our deployment configurationc and propose the architecture of the system and how does the self-adaptive configuration process dynamically execute with the workload interval changing (different rate requests at different interval). Next, the SLAs metrics, including performance metrics and cost, are modeled and computed, and corresponding optimization algorithm is designed in Section IV. Finally, the contributions of this paper and future work are summarized in Section V.

II. RELATED WORK

In recent days, increasing enterprises deploy their web-based applications on the flexible "pay-as-you-go" or "on-demand" cloud services, considering the significant reduction of the total cost of ownership and operating [4-7] . The cloud customers and providers often negotiate with Service Level Agreements (SLAs) which outlines all aspects of cloud service usage and obligation of both service providers and clients, including various descriptors collectively referred to as Quality of Service (QoS) [4-7]. The cloud service providers need to schedule their resources to maximize their profit with the satisfaction of SLAs, while the customers tend to minimize their cost with the satisfaction of application requirements based on the SLAs.

Typically, SLAs for traditional distributing computing and autonomic computing systems are guaranteed in term of response time and throughput [8, 9]. In cloud system, [10-12] present the selected SLAs as average response time, transaction throughput, cloud availability and cost; In [13], SLAs are defined as performance metrics (throughput, latency, usage) and cost.

[14] evaluated its performance using an embedded Markov chain model to obtain a probability distribution of response time. In [15], the percentile of response time is studied. [16] modeled the distribution of response time was obtained for a cloud center as M/M/m/m+r (i.e. the first letter M: exponential arrival process; the second letter M: exponential service time distribution; the third letter m: number of servers; the fourth letter means number of services in system including number in queue and number being served) queuing system. Analysis in the case where inter-arrival time and/or service time are not exponential is more complex. Most theoretical analysis focused on extensive research in performance evaluation of M/G/m queuing systems [17].

The objectives among these SLAs are usually conflicting. To solve the cloud resource configuration optimization based on conflicting SLAs, several heuristics algorithms have been proposed to find semi-optimal solutions in a reasonable short time [15, 18-20]. However, existing heuristics approaches do not consider SLAs [21, 22] or consider only single [23] or two SLAs [24] at a time. In addition, they give single semi-optimal solution which do not present the trade-offs between different solutions. No existing work considers multiple differentiated SLAs and supports dynamic service deployment configurations optimization in cloud computing environment. This paper will study the cloud service configuration optimization considering both system performance, resource provision and scheduling, multiple conflicting SLAs, and changing workloads.

III. DEPLOYMENT CONFIGURATION AND SYSTEM ARCHITECTURE

A. Deployment Configuration

This problem, the deployment of cloud service problem, is a combinatorial optimization problem which ensures the optimal mapping between each service of an application and cloud computing resource. Since there are a large number of various types of cloud computing instances and many services in one application, a huge number of possible deployment plans can be drawn out. For example, one possible deployment plan is represented as one of combinations plans

$$S = m \times (2^n - 1) \tag{1}$$

where m is the number of cloud computing instances types and n is the number of services in an application.

An example deployment configuration is shown as Figure 1. Each deployment configuration can have arbitrary number of deployment plans and each deployment plan can process arbitrary number of service by which the operation

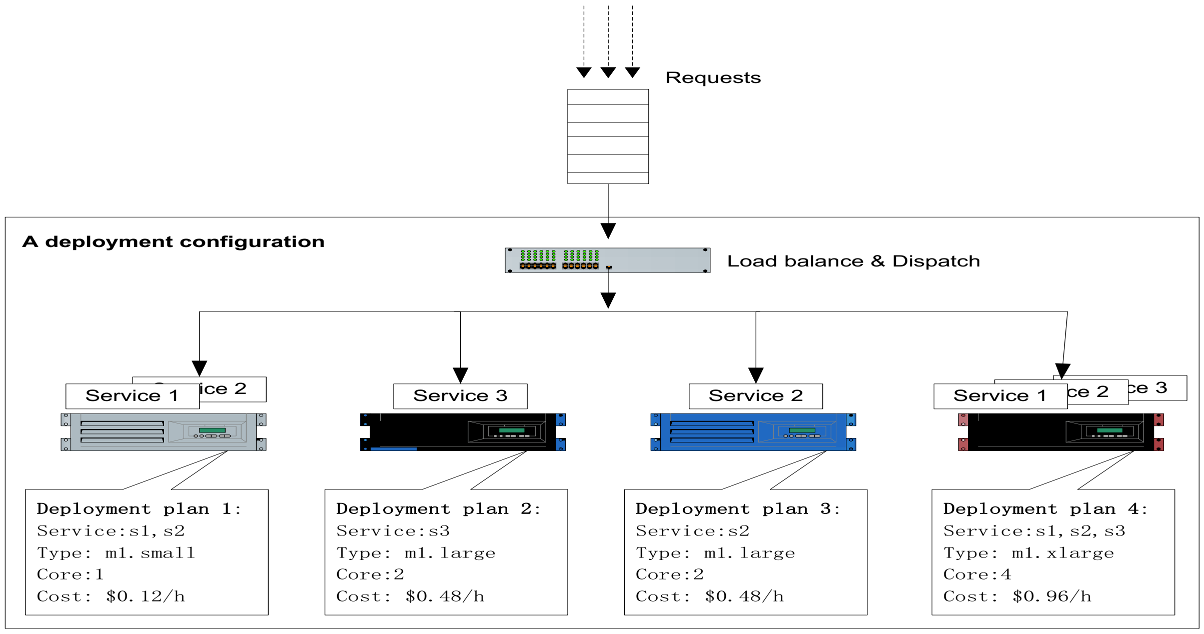

Figure 1. An Example Deployment Configuration

of services' throughput and latency is improved. Our research work supports that multiple same services are not operated on the same deployment plan, since multiple same services deployed on one machine do not contribute to the performance improvement.

B. *Architecture of Self-adaptive Configuration Optimization System*

The architecture of our proposed system, its components and their interaction are shown in Figure 2. The system has four main components: workload analysis and prediction model, SLAs computation model, optimization and scheduling model, and resource configuration model. This system is a closed loop self-adaptive control model. When the request workload is changing, the optimization and scheduling model executes the optimization algorithm to determine the best configuration for the system at regular interval (i.e. reconfiguration interval), considering the previous execution results and cloud computing resources information (as shown in Figure 3.)

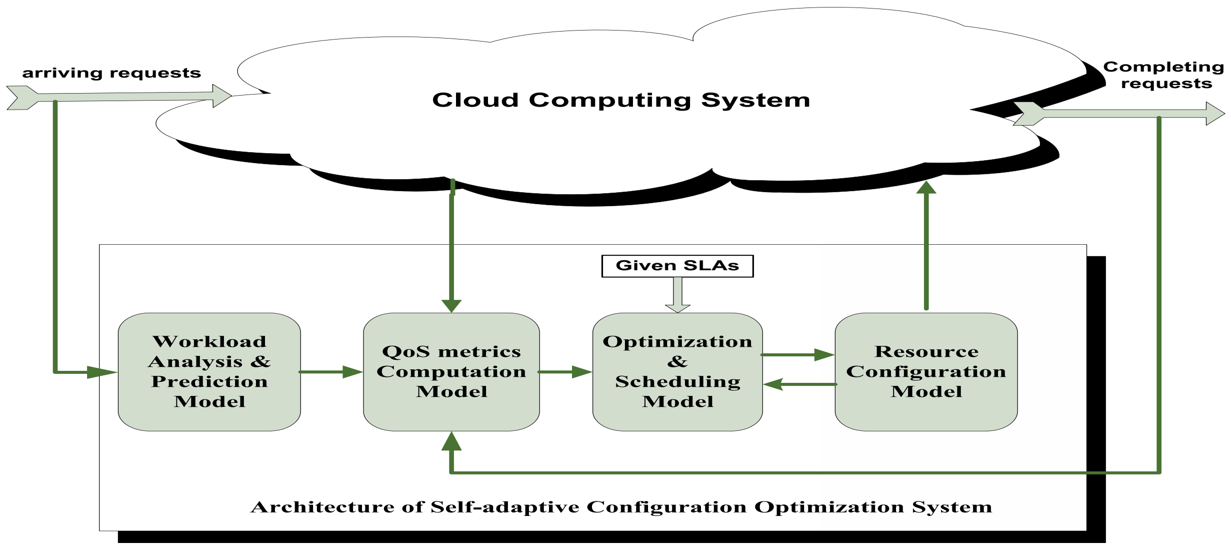
arriving requests
Cloud Computing System
Completing requests
Given SLAs
Workload Analysis & Prediction Model
QoS metrics Computation Model
Optimization & Scheduling Model
Resource Configuration Model
Architecture of Self-adaptive Configuration Optimization System

Figure 2. Architecture of Self-adaptive Configuration Optimization System in Cloud Computing

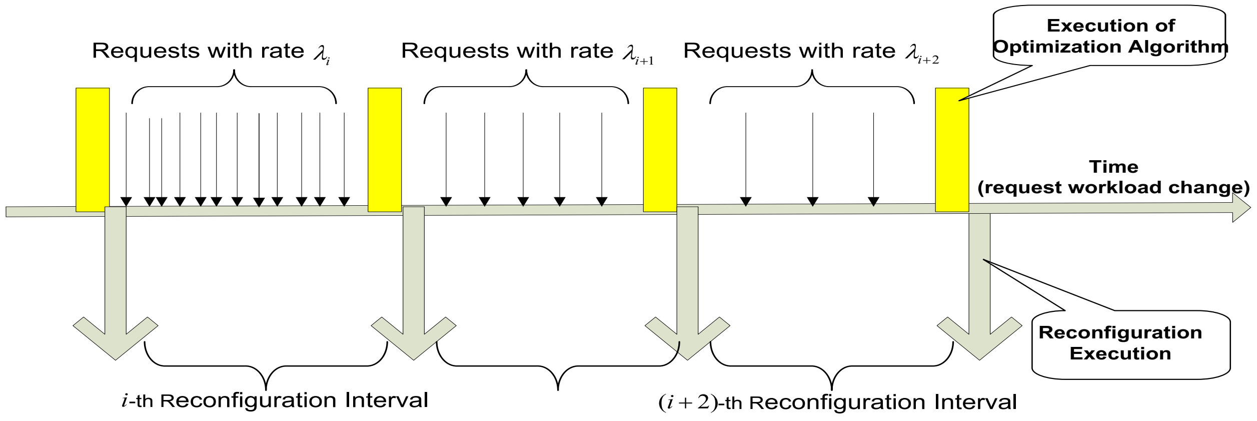
Execution of Optimization Algorithm
Requests with rate λ_i
Requests with rate λ_{i+1}
Requests with rate λ_{i+2}
Time (request workload change)
Reconfiguration Execution
i-th Reconfiguration Interval
$(i+2)$-th Reconfiguration Interval

Figure 3. Self-adaptive Reconfiguration Interval

The SLAs computation component collects utilization data from the cloud computing system (e.g., CPU and disks) as well as the account of completed requests, which allows the component to compute the SLAs objectives (e.g. throughput, latency, cost). The service demand of a request, i.e., the total average service time of a request at a resource, can be computed as the ratio between for resource utilization and the system throughput. The values of SLAs objectives computed by this component are used as one of the input parameters to the optimization and scheduling model.

The workload analysis and prediction model analyzes the stream of arriving requests, computes and stores statistics for the workload intensity, such as average arrival rate, and applys statistical approaches [25, 26] to model and predict the intensity of the workload in the next reconfiguration execution interval. The current or predicted values of workloads intensity computed by this component are also used as input parameters of SLAs computation model.

At the beginning of each reconfiguration interval (see Figure 3), the optimization and scheduling model runs the optimization algorithm. This algorithm takes into account the desired SLAs objective goals, the arrival and departure processes, and performs a multiobjective genetic algorithm to search a trade-off configuration in the state space of possible deployment configurations. Once the resource configuration model determines the best configuration for the workload intensity level provided by the workload analysis and prediction model, it will send reconfiguration commands to the cloud computing systems. Then system is realized with self-adaptive configuration optimization.

IV. SLAs METRICS MODELLING AND OPTIMIZATION ALGORITHM

In this paper, we study a given workload distribution with the same rate request in the reconfiguration interval. In the section, we present the SLAs metrics modeling and propose a computational method of Stochastic SLAs Metrics. Then we develop a corresponding optimization algorithm (genetic algorithm).

A. SLAs Metrics Modelling using Utility Function

In our research work, we define the SLAs as performance metrics (throughput, latency, CPU usage) and cost. These four objectives are conflicting and variously weighted by different clients. Therefore the optimization value is usually a trade-off optimal SLAs rather than a single optimal result. To handle this multiobjective optimization problem, we use the notion of utility function to determine the combined usefulness of cloud services as a function of the various SLAs. Utility functions are used quite often in economics and have been extensively used in autonomic computing [27]. However, the use of utility functions to determine the optimal mix of SLAs in cloud computing as presented here is novel. We introduce the following notation to formalize the problem of optimal selection of SLAs. Let objectives tuple in SLAs is: $SLAs = \{T, L, U, C\}$.

where

- T (throughput) : SLA on transaction throughput.
  $T = (T_1, T_2, \ldots T_n)$, T_i is the i-th interval throughput.
- L (latency) : SLA on response time or waiting time of per service instance processing.
  $L = (L_1, L_2, \ldots L_n)$, L_i is the i-th interval average response time.
- U (CPU Usage) : SLA on CPU utilization.
  $U = (U_1, U_2, \ldots U_n)$, U_i is the i-th interval CPU utilization.
- C (cost): SLA on resource utilization cost.
  $C = (C_1, C_2, \ldots C_n)$, C_i is the i-th interval resource utilization cost.

- w_1, w_2, w_3, w_4 : is the weights associated to throughput, latency, CPU usage, and cost, respectively, used to indicate the preference of objectives in SLAs for cloud clients.

$$w_1 + w_2 + w_3 + w_4 = 1 \tag{2}$$

The *Utility Function* formulas of $SLAs = \{T, L, U, C\}$ metric uses a following the type of sigmoid curve [28-30]:

$$Uti = K\frac{e^{\alpha(\beta-\varepsilon)}}{1+e^{\alpha(\beta-\varepsilon)}} \tag{3}$$

Where K is the normalizing factor. α is a sensitivity parameter that defines the sharpness of the curve and β is the associated SLAs goal. The sign of α determines whether the sigmoid decrease (α f 0) with ε or increase (α p 0) with ε . The maximum value of the utility function defined above is 1. These values occur at $\varepsilon = 0$ for latency and cost, $\varepsilon = 1$ for CPU usage, and for $\varepsilon \to \infty$ for throughput. A decreasing utility function is used for latency and cost, and an increasing one is used for throughput and availability. The utility function of SLAs is following:

$$\begin{aligned} Utility(SLAs) &= w_1 Uti(T) + w_2 Uti(L) \\ &+ w_3 Uti(U) + w_4 Uti(C) \end{aligned} \tag{4}$$

The first term $Uti(T)$ in Eq. (4) is the throughput utility function, the following terms are latency, CPU usage and cost utility function, respectively. Since each value of these individual utility functions belongs to the [0, 1] range and $w_1 + w_2 + w_3 + w_4 = 1$, $Utility(SLAs) \in [0,1]$.

Now, a cloud client is faced with the problem of selecting the appropriate resource deployment that maximizes the utility function subject to SLA constraints. This non-linear constraint optimization problem shown below:

$$\max Utility(SLAs) = f(T, L, U, C) \tag{5}$$

Subject to

$$T_{\min} \le T \le T_{\max}$$
$$L_{\min} \le L \le L_{\max}$$
$$U_{\min} \le U \le U_{\max}$$
$$C_{\min} \le \sum C_i \le C_{\max}$$

B. Stochastic SLAs Metrics Computation

This section describes the stochastic SLAs computation method used in the modeling of SLAs utility function optimization. To estimate the performance of distributed systems such as cloud system, queuing theory is a well-established method which has been studied by a number of research work [31]. Our research work uses queuing theory and statistic techniques to estimate SLAs objectives that are throughput, latency (i.e. waiting time), CPU usage and cost.

We model cloud server cluster as a M/G/m queuing system which indicates that the inter-arrival process of requests is exponential inter-arrival times (i.e. given Poisson arrival distribution). The Poisson distribution is shown as:

$$f(k;\lambda) = \sum_{k=0}^{m} \frac{\lambda^k e^{-\lambda}}{k!} \tag{6}$$

where

- *e* is the base of the natural logarithm (*e* = 2.71828...).
- *k* is the number of occurrences of an event — the probability of which is given by the function.
- *k*! is the factorial of *k*.
- *λ* is a positive real number, equal to the expected number of occurrences during the given interval

Assume that only one service runs on a deployment plan with n CPU cores, considering the following notations:

- μ : the maximum *number* of processed requests per unit time for single cores.
- n : the number of cores.
- s_i : the cost computation in i -th interval time.

From Eq. (6), the Expectation is $E(f(k;\lambda)) = \lambda$, that means the average number of arrival requests per reconfiguration interval. Since we support given distribution with same rate requests, μ is known. Then the throughput can be computed as:

$$T = \begin{cases} \lambda, \lambda \le n\mu \\ n\mu, \lambda \text{ f } n\mu \end{cases} \tag{7}$$

And the approximation of CPU usage is :

$$U = \rho = \frac{\lambda}{n\mu} \tag{8}$$

The probability that latency distribution function for M/G/m queuing system is [8, 32] :

$$P_r(W \ge t) = (\frac{np_n}{n-\rho})e^{-(n\mu-\lambda)t}, t \ge 0 \tag{9}$$

Where,

$$p_n = p_0(\frac{\rho^n}{n!})$$

And

$$p_0 = \left[\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{(n-1)!(n-\rho)}\right]^{-1}$$

Therefore, the calculation of the latency is the expectation shown as in Eq. (10)

$$L = E(P_r(W \ge t)) = \frac{np_n}{(n-\rho)(n\mu-\lambda)} \tag{10}$$

Then the i -th interval resource utilization cost is

$$C_i = ns_i \tag{11}$$

C. Optimization Algorithm

In this section, we develop a multi-objective genetic algorithm for configuration optimization in cloud computing. The detailed algorithm is specified in Algorithm 1. This algorithm is designed to seek individuals that satisfy given SLAs and exhibit optimal utility value considering trade-off among QoS objectives in SLAs. At each generation, two parents, p1and p2 are selected with binary tournaments. They reproduce two offsprings by performing the crossover operator with one-point crossover on genes. Then, the offsprings's genes are mutated. A mutation operator is designed to increase or decrease resources. In order to provide an ability to dynamic change a deployment plan to a deployment configuration, the mutation of operator first adds an empty deployment plan (random selected), i.e. there is no service deployed on it. After execute mutations, the mutation operator examines each deployment plans and release empty deployment plans form a deployment configuration. Therefore, it realizes the self-adaptive on-demand cloud service deployment. This reproduction process is repeated until arriving at the MaxGeneration.

In order to fulfill both the requirements, this multiobjective genetic algorithm for configuration optimization in cloud computing uses the utility function of SLAs (shown as in Eq.(5)) as the fitness function which satisfy the given SLAs constraints. The fitness function for feasible individuals is designed to encourage them to improve their SLAs values in all objectives and maintain diversity in their population of SLAs values.

Algorithm 1: a multiobjective genetic algorithm for cloud resource configuration optimization

```
Initial:
  MaxGeneration <- Max generation count;
  u <- population size;
  P <- Random u individuals;
  g <- 0;
  repeat until (++g > MaxGeneration)
  {
     /* Assign fitness value to the population P based on the
utility function */
    For each p in P
    {
       t <- Throughput with configuration p;
       l <- latency with configuration p;
       u <- CPU Usage with configuration p;
       c <- Cost with configuration p;
       u <- utility(t,l,u,c);
       p.fitness <- u;
    }
         /* Keep the fixed size of population P */
    if size of(P) > u then
       P <- Top u individuals of P based on fitness value;

    /* Produce the next generation */
    Q <- new empty population;
    repeat until (size of(Q) == u)
    {
       r,s <- random 2 individuals from P;
       p1 <- max(r,s) based on fitness value;
       r,s <- random 2 individuals from P;
       p2 <- max(r,s) based on fitness value;

       tmp1,tmp2 = Crossover(p1,p2);
       if (tmp1, tmp2) are not better than (p1,p2) then
          tmp1,tmp2 <- p1,p2;

       tmp1 <- mutation(tmp1);
       if tmp1 is better than p1,p2 && tmp1 is not exist in Q
then
          add tmp1 to Q;
       tmp2 <- mutation(tmp1);
       if tmp2 is better than p1,p2 && tmp2 is not exist in Q
then
          add tmp2 to Q;
    }
     /* Combine the new generation with the old generation */
    P <- P + Q;
  }
```

V. CONCLUSIONS AND FURTHER STUDY

This paper addressed new challenges in the "pay-as-you-go" cloud service scenario for providers' services by which customers make decision of purchasing appreciate resources before their systems executing, especially when their workload is uncertain and dynamically changed.

This paper first studies the cloud service configuration optimization considering both system performance, resource provision and scheduling, multiple conflicting SLAs, and changing workloads. We developed the self-adaptive cloud service configuration optimization considering multiple conflicting SLAs and supported dynamic reconfiguration when workloads change. Our designed cloud service configuration optimization system and algorithm can be applicable to any cloud computing platforms.

There are still several future works we will do. The experimental results of the optimization algorithm and performance comparison with other similar research works should be drawn out. Considering different attributes of different cloud platforms and other workload distributions, another SLAs metrics modeling approaches, rather than queuing theory and the statistic approached used in this paper, can be designed to obtain high quality values. In this paper, we did not consider the network connections and geographic distributed resource centers. Therefore, our future work will be extended to capture more aspects of real-world attributes and improve the applicability of our research work to real commercial cloud computing platforms.

VI. ACKNOWLEDGMENT

The work presented in this paper was supported by the Australian Research Council (ARC) under discovery grant DP110103733.

REFERENCES

[1] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus Toolkit for Market-Oriented Cloud Computing," *Cloud Computing, Proceedings,* vol. 5931, pp. 24-44, 2009.

[2] C. F. Li, "Cloud Computing System Management Under Flat Rate Pricing," *Journal of Network and Systems Management,* vol. 19, pp. 305-318, 2011.

[3] Amazon. (2011). http://aws.amazon.com/ec2/.

[4] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," *Hpcc 2008: 10th Ieee International Conference on High Performance Computing and Communications, Proceedings,* pp. 5-13, 2008.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems-the International Journal of Grid Computing-Theory Methods and Applications,* vol. 25, pp. 599-616, 2009.

[6] E.-K. Byun, Y.-S. Kee, J.-S. Kim, E. Deelman, and S. Maeng, "BTS: Resource capacity estimate for time-targeted science workflows," *Journal of Parallel and Distributed Computing,* vol. 71, pp. 848-862, 2011.

[7] W. H. Tian, "Adaptive Dimensioning of Cloud Data Centers," *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, Proceedings,* pp. 5-10, 2009.

[8] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 22, pp. 931-945, 2011.

[9] H. Topcuoglu, S. Hariri, and M. Wu, "Performance-efficective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems,* vol. 13, pp. 260-274, 2002.

[10] H. Wada, P. Champrasert, J. Suzuki, and K. Oba, "Multiobjective Optimization of SLA-Aware Service Composition," in *Services - Part I, 2008. IEEE Congress on*, 2008, pp. 368-375.

[11] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "Multi-Objective Genetic Algorithms for SLA-aware Service Deployment Optimization Problem," *Services Computing, IEEE Transactions on,* vol. PP, pp. 1-1, 2011.

[12] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-Based Resource Allocation for Multi-tier Cloud Computing Systems," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, 2011, pp. 324-331.

[13] D. A. Menasce and P. NGo, "understanding cloud computing:experimentation and capacity planning," in *Proc. 2009 Computer Measurement Group Conf.*, Dallas, TX, 2009.

[14] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "Evolutionary deployment optimization for service-oriented clouds," *Software-Practice & Experience,* vol. 41, pp. 469-493, Apr 2011.

[15] H. Khazaei, J. Misic, and V. Misic, "Performance Analysis of Cloud Computing Centers Using M/G/m/m + r Queueing Systems," *Parallel and Distributed Systems, IEEE Transactions on,* vol. PP, pp. 1-1, 2011.

[16] X. Kaiqi and H. Perros, "Service Performance and Analysis in Cloud Computing," in *Services - I, 2009 World Conference on*, 2009, pp. 693-700.

[17] B. Yang, F. Tan, Y. Dai, and s. Guo, "performance evaluation of cloud serviece considering fault recovery," presented at the First International Conference on Cloud Computing, CloudCom 2009, Beijing, China, 2009.

[18] H. Khazaei, J. Misic, and V. B. Misic, "Modelling of Cloud Computing Centers Using M/G/m Queues," in *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, 2011, pp. 87-92.

[19] M. Escobar, A. R. Odoni, and E. Roth, "Approximate solution for multi-server queueing systems with Erlangian service times," *Computers & Operations Research,* vol. 29, pp. 1353-1374, Sep 2002.

[20] B. N. W. Ma and J. W. Mark, "Approcimation of the mean queue length of an M/G/c queueing system," *Operations Research,* vol. vol.43, pp. 158-165, 1998.

[21] I. Moser, S. Mostaghim, and Ieee, "The Automotive Deployment Problem: A Practical Application for Constrained Multiobjective Evolutionary Optimisation," presented at the 2010 IEEE Congress on Evolutionary Computation, 2010.

[22] H. Wada, J. Suzuki, and K. Oba, "Queuing Theoretic and Evolutionary Deployment Optimization with Probabilistic SLAs for Service Oriented Clouds," presented at the 2009 IEEE Congress on Services, 2009.

[23] G. Pandey, V. N. Rao, and A. K. Srivastava, "current cloud scenario review and cost optimization by efficient resource provisioning," presented at the ACM conference on Compute'11, Bangalore, India, 2011.

[24] M. Macias, J. O. Fito, and J. Guitart, "Rule-based SLA management for revenue maximisation in Cloud Computing Markets," in *Network and Service Management (CNSM), 2010 International Conference on*, 2010, pp. 354-357.

[25] A. Andrzejak, D. Kondo, and Y. Sangho, "Decision Model for Cloud Computing under SLA Constraints," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, 2010, pp. 257-266.

[26] W. Linlin, S. K. Garg, and R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 195-204.

[27] Archana S. Ganapathi, Yanpei Chen, Armando Fox, Randy H. Katz, and D. A. Patterson. (2009, November 30). *Statistics-Driven Workload Modeling for the Cloud*. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-160.html

[28] T. Gerald, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," *Internet Computing, IEEE,* vol. 11, pp. 22-30, 2007.

[29] Y.-q. Zhao, L.-b. Lu, and S.-f. Fang, "Stochastic Linear Optimization for Modeling Uncertainty in Aggregate Production Planning," in *Autonomic and Autonomous Systems, 2006. ICAS '06. 2006 International Conference on*, 2006, pp. 31-31.

[30] L. G. A. Sung, J. W. Wong, and Ieee, *Autonomic resource management for a cluster that executes batch jobs*, 2006.

[31] D. A. Menasc, J. M. Ewing, H. Gomaa, S. Malex, and P. Sousa, "A framework for utility-based service oriented design in SASSY," presented at the Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, San Jose, California, USA, 2010.

[32] C. Stewart, T. Kelly, and A. Zhang, "Exploiting nonstationarity for performance prediction," presented at the ACM European Conference on Computer Systems, Lisbon, portugal, 2007.