Wide Area Tentative Scaling (WATS) for Quick Response in Distributed Cloud Computing

Hitoshi Yabusaki, Hiroshi Nakagoe, Koichi Murayama, and Takatoshi Kato
Yokohama Research Laboratory, Hitachi, Ltd.
Yokohama-shi, Kanagawa-ken, Japan
{hitoshi.yabusaki.vw, hiroshi.nakagoe.yo, koichi.murayama.hp, takatoshi.kato.bb}@hitachi.com

***Abstract*—As cloud computing is increasingly adopted by enterprises, a stringent service level agreement such as response time is required by the cloud in order to run business applications. This can be problematic because activity from geographically distributed terminals owing to globalization often increases the average response time. Federating various clouds enables to utilize datacenters in various geographical regions regardless of their servicers. The response time can be reduced by replicating the applications and related data at datacenters near the terminals by considering the factors of delay (e.g., data synchronization, distribution of multi-tier applications, and influence of other applications). However, it is unrealistic to accurately analyze all of the factors without any errors. We propose wide area tentative scaling (WATS) to improve the response time in a phased manner by repetitively replicate a part of the application and related data at other datacenters and selecting a better organization. The WATS approach is to repetitively change the organization to reduce the response time even if the analysis is incorrect, unlike mathematical-formulae-based approaches that require precise analysis. The drawback of this approach is that it consumes more computing resources due to repeating the replication. We therefore, applied Bayesian inference to search for a better organization with fewer trials. Evaluation results showed that WATS successfully reduced the response time in a phased manner.**

***Index Terms*—Cloud computing, replication, response time, computing cost, Bayesian inference**

I. Introduction

Business applications (e.g., CRM and ERP) in private cloud computing systems are accessed from all over the world owing to modern-day globalization. This makes it difficult for the cloud to deliver a good response time in fact, the response time can be hundreds or even thousands of milliseconds if terminals access the applications from a great distance. This exceeds the maximum response time for end users to use applications without deteriorated usability. Content delivery networks reduce the response time of static (i.e., read-only) Web content, but it does not work for dynamic (i.e., read/write) content such as business applications.

To lower the response time, applications and data should be replicated at datacenters in the same geographical regions as the terminals by federating multiple clouds. Most terminals are globally distributed, while most datacenter locations provided by a cloud servicer are typically limited to a few regions. To overcome this limitation, federation or interconnection of multiple clouds is a promising technology that enables the utilization of the multiple clouds including public clouds. Common standards and policies to provide a universal environment for clouds [1] have recently been formulated.

The open issue in today's operation of private clouds involves improving the response time when utilizing multiple datacenters, including those of public clouds. For one thing, system analysis and planning for placing applications and related data is often manual work. Moreover, measuring the response time in test environment and feedback results is required because estimated response time based on operational experience is not precise enough to guarantee the stringent service level agreements required by the business applications. Such work is one of the principle factors that make the management of private cloud structures so expensive.

Previous studies related to placement based on mathematical formulae such as a linear integer programming promise to alleviate the analysis and planning workload but do not alleviate the manual measurement and feedback. For example, placement of applications together with their data components in the cloud has been studied in [2]. Cloud platform requirements are specified in terms of processing capacity, memory, and storage. Similar requirements are formulated into mixed integer programs in [3] to find an appropriate mapping within datacenters. Various other approaches have also attempted to solve multi-fold placement constraints on CPU, memory, network bandwidth, and energy [4]–[6]. Placement problems to minimize end-to-end response time have been formulated in [7] and [8].

These approaches to optimizing placement do not work well if the input parameters of the model are inaccurate, and it is no trivial matter to precisely understand these parameters (e.g., performance and communication delay) in the federated cloud. For example, performance in most public clouds is not guaranteed and varies depending on other applications, and communication delay between datacenters also varies. Moreover, enormous analysis is required to understand all of the behaviors and reference relationships between the components of multi-tier applications as well as the data consistency between components in a large-scale system. As a

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

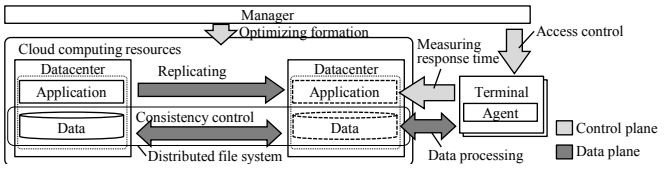

Fig. 1. Overview of wide area distributed cloud computing.

result, even after the previous studies are applied to the operation of the private clouds, the manual measurement and feedback is required to ensure sufficient performance.

Our objective is to reduce the response time by placing applications and data at geographically distributed datacenters without incurring additional operation load such as measurements in the test environment. We previously have proposed a concept of wide area distributed cloud computing system that reduces response time using geographically distributed datacenters [9]; we have prototyped automated application deployment, unified management of distributed data, and a flow control mechanism.

In this paper, we propose wide area tentative scaling (WATS) to repetitively change the placement organization of a part of applications and related data so that it can reduce response time in a phased manner (i.e., in a cycle of organization change). The previous approaches to optimizing the placement focus on improving analytical precision by considering various parameters. In contrast, WATS focuses on coping with analytical errors by tentatively measuring change organization, thereby directly tackling the difficulty of precise estimation. The drawback of this approach is that it consumes more computing resources due to repeated replication. We therefore, applied Bayesian inference to search for a better organization with a fewer trials.

II. Wide Area Distributed Cloud Computing

Fig. 1 shows an overview of wide area distributed cloud computing. In this cloud, a manager replicates applications or a part of applications such as a Web front-end and their data at datacenters so that response times can be reduced by processing data nearer the terminals. In order to evaluate end-to-end response time, including access network delays, the agents deployed in the terminals and the manager collaborate with one another. Such an agent-in-client system architecture is not unique considering that some SaaS applies this type of architecture. Specifically, the manager specifies the appropriate location for each terminal to access, the terminals measure the response time and pass on the results to the manager, and the manager then calculates the organizations of the applications and data on the basis of the response time.

A distributed file system controls the data consistency. It supports three types of consistency: strong, weak, and metadata. With the strong consistency, an application responds to the terminals after completely synchronizing the same data at different datacenters. With the weak consistency, it responds before synchronizing the data, and in this case, the distributed data eventually become consistent. With the metadata consistency, it responds after synchronizing only metadata (e.g. data location, version, and name).

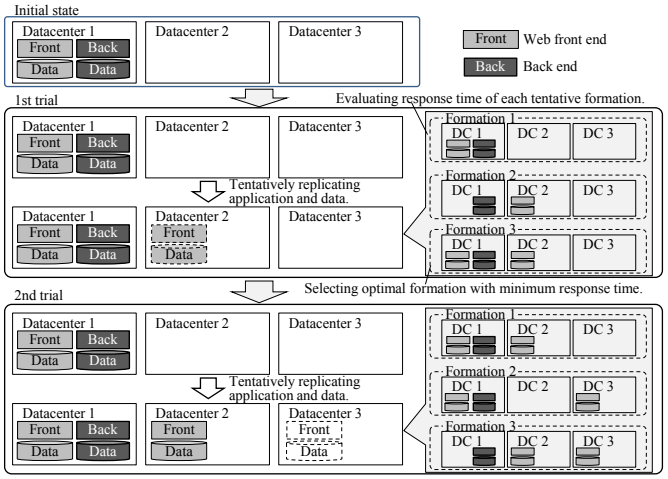

Fig. 2. Example behavior of WATS.

III. Wide Area Tentative Scaling (WATS)

We propose WATS to repetitively change the organization of an application and related data so that it can improve the response time in a phased manner. Unlike approaches to pursuing higher analytical precision, WATS focuses on coping with analytical errors by tentatively measuring change organization. It temporarily replicates the application and data, then measures the response time and selects better organization with lower response time. It repeats this trial in trial-and-error manner, thereby improving the response time although increasing computing cost required due to the repetitive organization changes.

A. Mechanism

An example behavior is shown in Fig. 2. Specifically, Web front-ends are deployed to optimal datacenters. As a precondition, front-end is initially located at datacenter 1. The maximum number of times that front-ends can constantly co-exist at different datacenters is two excluding the tentatively replicated front-end, which is temporally created at a different datacenter to compare the response time with ones at original datacenter and is deleted after comparison if inferior in response. The maximum number of the front-end is limited so as to prevent unlimited increase of computing costs.

At the first trial, let us assume that datacenter 2 is selected to replicate the front end if needed. The manager replicates front-end to datacenter 2 and evaluates the response time in each organization: the front end in only datacenter 1, the one in only datacenter 2, and the ones in both datacenter 1 and datacenter 2. The evaluation is then performed using A/B test whereby the terminals are randomly mapped to one of the three organizations and specified an application identifier at the datacenter (e.g., URL) to access. They request data process and measure the response time of the specified application. The manager selects one optimal organization (front end in both datacenter 1 and datacenter 2) with the minimum averaged response time. Applications are stopped if they are unused in the selected organization. It then goes forward to the second trial. Trials are repeated until improvement of the response time is small enough.

B. Management functions of WATS

Fig. 3 shows the management functions of WATS. The manager is composed of four functions: optimal organization estimation, tentative organization configuration, access control, and organization decision.

The optimal organization estimation function designates the datacenter at which the application and the data are replicated at the next trial. This function reduces the number of trials, thereby cut down the computing costs incurred by repetitive trials.

The tentative organization configuration function replicates the application and the data at the designated datacenter. The access control function randomly groups terminals and then maps these groups into organizations for split testing. It notifies the URLs of the application at the datacenters nearest to the terminals. It receives the response time measured by the terminals and averages these response times for each organization.

The organization decision function decides the optimal organization with the minimum response time. It also decides whether to continue trials or not. In the rest of this paper, we focus mostly on the optimal organization estimation function, which is the core function of WATS.

C. Optimal organization estimation function

The computing cost increases as the number of the trials increases, so, the optimal organization estimation function estimates the organization with the minimum average response time, which is evaluated at the next trial. on the basis of evaluation results of past trials. Thus, it improves response time with fewer trials.

In order to estimate the organization, the optimal organization estimation function calculates all the probabilities that each organization gives the minimum response time. It calculates the conditional probabilities in order to calculate probability on the basis of evaluation results of past trials.

D. Bayesian inference algorithm

The optimal organization estimation function calculates the conditional probability using Bayesian inference. The Bayesian inference allows informative priors so that prior knowledge or results of a previous model can be used to inform the current model. Moreover, it can avoid problems with model identification by manipulating prior distributions.

The symbols used in this paper are listed in Table I. The conditional probability $P_t(y_k|x_i)$ is that fact y_k (i.e., replicating the sub-application and data to the datacenter k minimize the response time) happens, given the fact x_i (i.e., response time $r_{i,t}$ averaged by terminals accessing the sub-application and data at datacenter i was less than the threshold α_i at t trial), is calculated by

$$P_t(y_k \mid x_i) = P_t(y_k) \times P(x_i \mid y_k) / P(x_i), \quad (r_{i,t} < \alpha_i). \tag{1}$$

Likewise, given the fact $\bar{x}_i$ that the response time $r_{i,t}$ was more than the threshold α_i at t trial, the conditional probability $P_t(y_k \mid \bar{x}_i)$ that fact y_k happens is calculated by

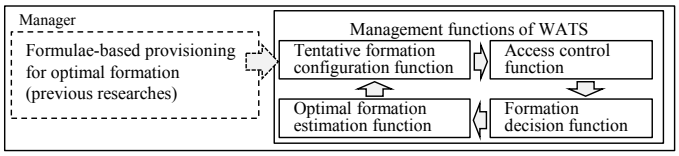

Fig. 3. Management functions of WATS

TABLE I. SYMBOLS USED IN BAYESIAN INFERENCE

Symbol	Explanation
$r_{i,t}$	Average response time ($\rho_{k,i}$) of terminals accessing the application at datacenter i, measured at t ($t \in T$) trial.
x_i	Fact that response time $r_{i,t}$ is less than threshold α_i.
$\bar{x}_i$	Fact that response time $r_{i,t}$ is equal to or larger than α_i.
y_k	Fact that replicating a sub-application and related data at datacenter k minimizes the response time.
$P_t(x)$	Probability that a fact x happens at t trial.
$P_t(y\|x)$	Conditional probability that a fact y happens, given that a fact x has happened.
$DC_{opt,t}$	A datacenter where the sub-application and data are replicated at t trial.

$$P_t(y_k \mid \bar{x}_i) = P_t(y_k) \times P(\bar{x}_i \mid y_k) / P(\bar{x}_i), \quad (r_{i,t} \geq \alpha_i). \tag{2}$$

Probability $P_t(y_k)$ should be calculated on the basis of Bayesian updating. If the application and data are replicated at datacenter i in t-1 trial, $P_t(y_k)$ is updated as,

$$P_t(y_k) = P_{t-1}(y_k \mid x_i), \quad (r_{i,t-1} < \alpha_i, \quad t \geq 2), \tag{3}$$

$$P_t(y_k) = P_{t-1}(y_k \mid \bar{x}_i), \quad (r_{i,t-1} \geq \alpha_i, \quad t \geq 2). \tag{4}$$

The probabilities $P(x_i)$, $P(\bar{x}_i)$, $P_t(x_i \mid y_k)$ and $P_t(\bar{x}_i \mid y_k)$ are given as statistical information calculated from execution history or simulation. The optimal organization estimation function selects $DC_{opt,t}$ whose conditional probability is the largest among datacenter k that have not been replicated in past trials.

IV. MODELS OF RESPONSE TIME AND COST

A. Response time

To evaluate the response time and the cost, we modeled the parameters bellow. The symbols used in modeling the response time are listed in Table II. The response time is calculated as the sum of communication delay between a terminal k and a datacenter i ($\rho_{k,i}$'), communication delay between datacenters i and j ($\rho_{i,j}$''), and the processing time at a datacenter i (ρ_i''').

(1) Communication delay between terminal and datacenter

The communication delay includes overhead time and time to transfer data. The delay $\rho_{k,i}$' is calculated as

$$\rho_{k,i}' = d_{i,k} \times s + \delta / b_{i,k}. \tag{5}$$

(2) Communication delay between datacenters

The communication delay between datacenters changes depending on the required data consistency. In this paper, we consider three types: strong consistency, weak consistency, and metadata consistency. In the strong consistency, the communication delay between datacenter i and j is calculated as the sum of the overhead time and the time to transfer data, as

TABLE II. SYMBOLS USED IN MODELING RESPONSE TIME

Symbol	Explanation
$\rho_{k,i}$	Response time when terminal k accesses the application at datacenter *i*, measured at *t* trial.
$\rho_{k,i}'$	Communication delay between terminal *k* and datacenter *i*.
$\rho_{i,j}''$	Communication delay between datacenter *i* and *j*.
ρ_i'''	Processing time of an application at datacenter *i*.
$d_{i,k}$	RTT between datacenter (or terminal) *i* and datacenter *k*.
$b_{i,k}$	Network bandwidth from datacenter (or terminal) *i* to datacenter *k*
s	Overhead of communication (e.g., fragmented communication and session establishment).
δ	Amount of data to transfer.
I	A set of datacenters where sub-application hasn't been replicated.
x	CPU utilization ratio.

TABLE III. CONSTANTS AND ERRORS IN PROCESSING TIME MODEL

Cores	a	b	c	Error rate
1 core	7.94	-1.56	45.17	0.08
2 core	3.89	2.44	14.96	0.3
3 core	5.12	1.27	30.85	0.22
4 core	14.09	-7.74	71.69	0.23

$$\rho_{i,j}'' = d_{i,j} \times s' + \delta' / b_{i,j}. \quad (6)$$

In the weak consistency, the communication delay between datacenters does not affect response time. Therefore the delay between datacenter *i* and *j* is calculated as

$$\rho_{i,j}'' = 0. \quad (7)$$

In the metadata synchronization, $d_{i,j} \times s' >> \delta' / b_{i,j}$ because, in wide area networks (WANs), RTT is tens or hundreds of milli-seconds and network bandwidth is more than 100 kbps, while the size of typical metadata is several bites. The communication delay between datacenters is calculated as

$$\rho_{i,j}'' = d_{i,j} \times s'. \quad (8)$$

(3) Processing time in datacenter

We modeled the processing time on the basis of the experimental data in [13]. Because the processing time increases as the CPU utilization increases, we formulated the processing time as

$$\rho_i''' = \exp(a \times x + b) + c, \quad (9)$$

where *a*, *b*, and *c* are constants calculated by the least squares approach. Table III shows the constants and relative errors of the formulation compared with the experimental results. The formulation and the experimental results matched within a relative margin of error rate of 0.08 in the case of a single core server, and 0.22~0.3 in the case of a 2–4 core server.

The response time, thus is calculated as

$$\rho_{k,i} = \rho_{k,i}' + \text{maximum}(\rho_{i,j}'', \text{for all } j \in \text{I}) + \rho_i'''. \quad (10)$$

TABLE IV. SYMBOLS USED IN MODELING COMPUTING COST

Symbol	Explanation
C	Total cost of computing resource.
Ci_i	Instance cost in datacenter *i*.
Cs_i	Storage cost in datacenter *i*.
$Cc_{i,j}$	Network cost between datacenter *i* and *j*.
$Cc'_{k,i}$	Network cost between terminal *k* and datacenter *i*.
N_i	Number of reserved instances at datacenter *i*.
N'_i	Number of on-demand instances at datacenter *i*.
Pi_i	Price of a reserved instance at datacenter *i*.
Pi'_i	Price of an on-demand instance at datacenter *i*.
T_i	Average time to use on-demand instances at datacenter *i*.
Ps_i	Price of storing a certain amount of data at datacenter *i*.
Ps'_i	Price of I/O transactions at datacenter *i*.
As_i	Amount of data to store at datacenter *i*.
Pn_i	Price of network cost of datacenter *i*.
Aio_i	Number of I/O transactions at datacenter *i*.
$An_{k,i}$	Amount of communication data between terminal *k* and datacenter *i*.
C_{total}	Total cost including computing cost and cost for repetitive trials.
Cr_t	Cost for *t* trial (tentative organization changes).

B. Computing cost

The symbols for modeling computing cost are summarized in Table IV. The computing cost of cloud computing includes instance cost (Ci_i), storage cost (Cs_i) and network cost. The network cost includes the cost between terminals and a datacenter ($Cc_{i,j}$) and the cost between datacenters ($Cc'_{k,i}$).

(1) Instance cost

We modeled each cost on the basis of a price model of a major cloud servicer. The instance cost is calculated as production of the number of instances and the price per instance, as

$$Ci_i = N_i \times Pi_i + N'_i \times T_i \times Pi_i, \quad (11)$$

where the first term is the cost of reserved instance and the second term is that of on-demand instance. The on-demand instances are for the tentative use.

(2) Storage cost

The storage cost includes the cost for storing data and the cost for I/O transactions. It is calculated as

$$Cs_i = As_i \times Ps_i + Aio_i \times Ps'_i. \quad (12)$$

(3) Network cost

The network cost between terminals and datacenters includes access network (e.g., LTE and FTTH) cost, WANs (e.g., the Internet and leased lines) cost and network cost at the entrance of datacenters. We should eliminate access network and WANs costs from the communication cost because these costs are incurred not by application providers but by end-users. The network cost is thus calculated as

$$Cc'_{k,i} = An_{i,k} \times Pn_i, \quad (13)$$

and the network cost between datacenter *i* and *j* is calculated as

TABLE V. ENVIRONMENTAL CONDITIONS OF EVALUATION

Item	Explanation
Locations of datacenters	Africa: Algeria, Burkina Faso Asia: China, India, Japan, Jordan, Malaysia, Pakistan Europe: German, Italy, Switzerland North America: Canada, USA South America: Bolivia, Brazil
RTT	RTT between datacenters and terminals were exactly equal to the measurement data in [10].
Network bandwidth	Network bandwidth between datacenters and terminals was exactly equal to the measurement data (TCP throughput) in [10].
Locations of terminals	Ten sites were randomly selected for each experiment case from 70 sites. The position was not changed during the each case.

TABLE VI. SPECIFICATION OF IT EQUIPMENTS

Item	Explanation
Processor (CPU)	2.4 GHz, 1 core
Memory	8 GB
Number of instances	1
Variation coefficient of CPU (σ)	13.79%
Variation of CPU	±3σ (degree of credence: 99.7%)

$$Cc_{i,j} = An'_{i,j} \times \left(Pn_i + Pn_j\right) \tag{14}$$

The computing cost excluding the cost for repetitive trials (cost incurred by WATS) is calculated as

$$C = \sum_{i \in I} \left(Ci_i + Cs_i\right) + \sum_{i \in I} \sum_{j \in I, j \neq i} Cn_{i,j} + \sum_{k \in I} \sum_{i \in I} Cn'_{k,i}. \tag{15}$$

(4) Cost including repetitive trials (cost incurred by WATS)

The cost Cr_t for t trial includes: network cost for transferring applications and their data; network cost between tentatively replicated data for consistency; and instance and storage cost due to tentative replication. The data transfer costs are calculated using Eq. 14. In the cost for replication, $An'_{i,j}$ is calculated as the data size of the transferred application and data. The instance cost is calculated using Eq. 11; note that all instances are on-demand instances. The storage cost is calculated using Eq. 12. The total cost including the computing cost and cost for repetitive trials is calculated as

$$C_{total} = C + \sum_{t \in T} Cr_t. \tag{16}$$

V. EVALUATION

A. Conditions

We evaluated the response time and the resource computing cost of WATS in a simulated cloud environment based on the model described in section IV and measurement results in real environment as described below. We set the location of the datacenters so that they are distributed to every geographical region and similar to that of a major cloud servicer. The locations of datacenters and terminals, and network conditions among them were set as shown in Table V. The specifications of servers in datacenters were set as shown in Table VI. We set variation coefficient of the CPU to 13.79% as measured in [11] for negative effects of other applications on processing time.

TABLE VII. CHARACTERISTICS OF APPLICATION [12]

Item	Explanation
Data amount of application	Input: 4291 MB, output: 7970 MB, logs: 40 MB
Storage	I/O transactions: 7176 times, storage size: 5 GB
Communication overhead	1.5 times
Data amount received by terminal	1 MB
Consistency	Randomly selected from strong consistency, weak consistency, and metadata synchronization for each case.

TABLE VIII. PRICES OF COMPUTING RESOURCE

Item		Explanation
Instance	On-demand	$ 0.4 / hour
	Reserved	$ 20.25 / month
Storage	Storing data	$ 0.1 / GB month
	I/O transaction	$ 0.3 / million transactions
Network	Input	$ 0.1 / GB
	Output	$ 0.17 / GB

We selected Montage as an application whose parameters are precisely measured using a cloud service in [12]. It's characteristics are shown in Table VII. We assumed all of the input/output data and log data were migrated when the application was replicated. The replicated application referred to ones at the other datacenters when terminals requests data in case of strong consistency and metadata synchronization. The prices of computing resource were set as shown in Table VIII, which are similar to the prices of major cloud servicers.

In each case, we randomly selected the locations of terminals and the strength of consistency, and then ran WATS. Note that the terminal positions and the consistency types were not given to WATS, assuming a situation in which systems were not completely analyzed. Because the evaluation results were affected by the distribution of terminals and the types of consistency, we evaluated 10 million cases to obtain the average response time and average computing cost. Before the evaluation, we ran an initial evaluation to calculate the statistical information of Bayesian inference (i.e., $P(x_i)$, $P(\bar{x}_i)$, $P_t(x_i \mid y_k)$, and $P_t(\bar{x}_i \mid y_k)$).

B. Results

Fig. 4 shows the average response time and average computing cost in the organization with the minimum response time. We omitted the cost incurred by WATS to show the upper limit in this environment. In the weak consistency and the metadata consistency, the average response time decreased as the maximum number of replicas increased, the response time was almost completely converged—i.e., to less than 6 %— when the number of replicas was more than 4. In strong consistency, the response time was almost completely constant regardless of the maximum number of replicas. The computing cost was increased in every strength of consistency. This figure

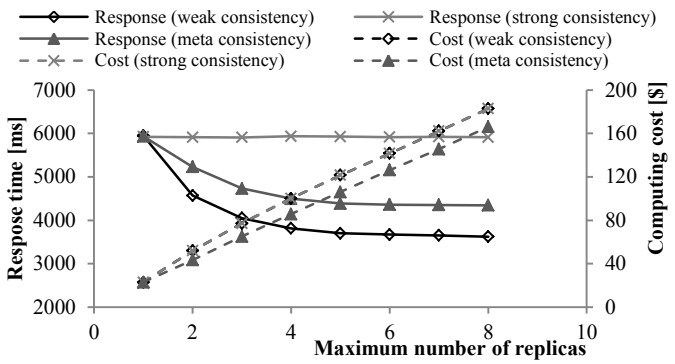

Fig. 4. Average response time and average computing cost excluding cost incurred by repetitive trials in optimal organization.

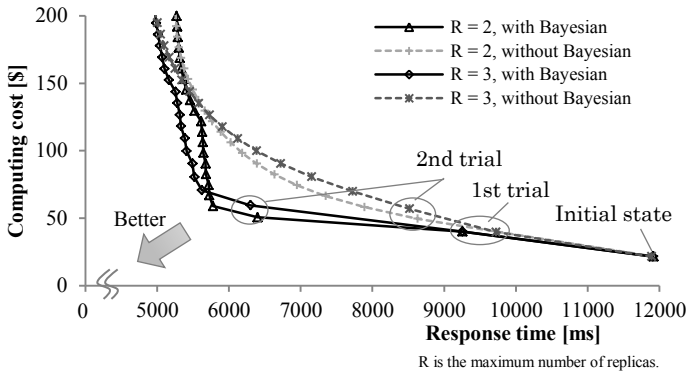

Fig. 5. Comparison on averaged response time and averaged computing cost with WATS.

leads to two findings: (a) the increase of the replicas does not always improve the average response time but it does increase the computing cost, and (b) the optimal number of replicas depends on the strength of the data consistency.

Fig. 5 shows the average response time and the average computing cost. As the number of trials increased, the response time decreased although the computing cost increased. It took 3 trials to reduce the response time by half for WATS with Bayesian inference while it took more than 10 trials for that without Bayesian inference (i.e., randomly selecting datacenters). In addition, the reduction of the response time with Bayesian inference at the 2nd trial was larger than the one at the 1st trial although the one at the 1st trial should be largest from a standpoint of statistics as the same with the case without Bayesian inference; this implies that WATS with Bayesian inference utilized the measurement result at the 1st trial in order to search for a better organization at the 2nd trial. These findings indicate that WATS successfully reduced the response time in a phased manner.

On the other hand, Fig. 5 also shows limitations of WATS: (a) the completely-converged response time was obtained after more than 20 trials even with the inference although the reduction of the response time at the first few trials was dominant; and (b) the computing cost was unlimitedly increased as the number of trials increased. Although the average response time monotonically decreased as the number of trials increased, the response time of each case did not in such a try-and-error approach to replicating applications. The first limitation implies there is some room to improve the inference for the complete convergence with fewer trials. The second one indicates to stop trials at adequate timing is important for quicker response with lower computing cost but is not trivial matter. These would be interesting future work.

VI. Conclusion

We proposed wide area tentative scaling (WATS) to repetitively replicate an application and related data to reduce response time in a phased manner. Unlike the previous approaches to optimizing placement with one trial, WATS focuses on coping with analytical errors by measuring tentative change organization thereby tackling the difficulty of the precise estimation. We also proposed Bayesian inference to search a better organization with fewer trials. The evaluation results showed that WATS successfully reduced the response time in a phased manner. It took 3 trials to reduce the response time by half for WATS with Bayesian inference while it took more than 10 trials for that without Bayesian inference.

References

[1] B. Rochwerger *et al*. "The RESERVOIR Model and Architecture for Open Federated Cloud Computing," *IBM J. of Research and Development*, vol. 53, no. 4, pp. 1–11, July 2009.

[2] Z. I. M. Yusoh and M. Tang, "A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud," in *Proc. of IEEE CEC*, 2010, pp. 1–8.

[3] X. Zhu *et al*. "Automated application component placement in data centers using mathematical programming." *Int. J. of Network Management*, vol. 18, no. 6, pp. 467–483, Nov./Dec. 2008.

[4] J. Anke *et al.* "A planning method for component placement in smart item environments using heuristic search." *DAIS, ser. LNCS*, vol. 4531, pp. 309–322, 2007.

[5] B. Urgaonkar *et al*. "Application placement on a cluster of servers." *Int. J. Found. Comput. Sci.*, vol. 18, no. 5, pp. 1023–1041, Jan. 2007.

[6] B. Li *et al.*, "Enacloud: An energy-saving application live placement approach for cloud computing environments," in *Proc. of IEEE CLOUD*, 2009, pp. 17–24.

[7] S. Agarwal *et al.*, "Volley: Automated data placement for geo-distributed cloud services," in *Proc. of USENIX Symp. on Networked Systems Design and Impl. (NSDI)*, 2010, pp. 2–2

[8] F. Chang *et al.*, "Placement in clouds for application-level latency requirements," in *Proc. of IEEE CLOUD*, 2012, pp. 327–335.

[9] H. Yabusaki *et al.*, "Flow Control for Higher Resiliency in Wide Area Distributed Cloud Computing," in *Proc. of IEEE COMPSACW*, 2013, pp. 760–764.

[10] "The PingER Project", Internet: www-iepm.slac.stanford.edu/pinger/, Feb. 03, 2014, [Feb. 21, 2014].

[11] R. Talaber *et al.*, "Using virtualization to improve data-center efficiency, Internet: www.thegreengrid.org/, Feb. 03, 2014, [Feb. 21, 2014].

[12] G. Juve *et al.*, "Scientific workflow applications on Amazon ec2," in *Proc. of Workshop on Cloud-based Services and Applications in conjunction with e-Science*, 2009, pp 59–66.

[13] R. Hashemian *et al.*, "Improving the scalability of a multi-core web server," in *Proc. of ACM ICPE*, 2013, pp. 161–172.