Article

TRANSACTIONS OF THE INSTITUTE OF MEASUREMENT & CONTROL

Transactions of the Institute of Measurement and Control
2014, Vol. 36(1) 26–37
© The Author(s) 2013
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0142331213488903
tim.sagepub.com
SAGE

An adaptive control mechanism for access control in large-scale distributed systems

Xiaofeng Jiang, Jun Li and Hongsheng Xi

Abstract

The highly scalable infrastructure of large-scale distributed systems is very attractive for network services. However, data access is unpredictable in this environment for the reasons of loosely coupled nature and large-scale data storage of such systems. Today, an increasing number of network applications require not only considerations of computation capacity of servers but also accessibility for adequate job allocations. An effective and adaptive mechanism of access control is important in this environment. In our study, the client clustering is used to describe the behaviors of clients and the adaptive server clustering is used to divide the large-scale distributed system into relevant small-scale systems. Since the clients which are assigned to one server cluster have the similar behaviors, we can use the stochastic control and passive measurement to do reliable and adaptive accessibility estimation and client allocation in such a small-scale system. We call this adaptive mechanism of access control based on accessibility estimation and client clustering as ACEC, and the experimental results show that ACEC can significantly reduce the data access cost and guarantee the load balance and controllability of large-scale distributed systems.

Keywords

Large-scale distributed system, adaptive control, access control, accessibility estimation, client clustering

Introduction

As on-line large-scale distributed systems have significantly increased in popularity over the past few years, various data objects are stored in diverse geographical locations, for example, a variety of applications for online video and the popular cloud computing platform. In these systems, different users access the network data from different locations depending on where they are located. However, the users do not actually specify their location when they make network requests. Therefore, a key component of the distributed systems is a mechanism for determining where users can obtain good services. The prime objective of our work is to minimize the access latency of clients which is achieved by allocating each client to the most appropriate server in the distributed system.

There are a number of problems for designing such a mechanism. First of all, the characterization of accessibility which determines the most appropriate server for a client should be defined in detail, the definition must be meaningful for access control and easy to obtain. Second, there are many different users in the network, it would be extremely difficult and expensive to keep track of all users individually. Third, the providers of systems cannot control the client machine directly, it is difficult to tell the client to measure the accessibility of the servers and choose the best server itself. Forth, the best servers for many clients may be one or two servers, if the mechanism allocates all of these clients to a small amount of servers, the network would suffer load imbalance and the clients would wait a long time for services, so the mechanism should also consider load balance. Finally, the network always changes over time. Servers are being added and removed all of the time and the communication latencies are changing constantly, the mechanism of access control should be sensitive to the accessibility of network, and can be adjusted adaptively.

To solve the problems, this study develops an adaptive control mechanism of access control based on accessibility estimation and client clustering called ACEC for large-scale distributed systems.

Since the control of a client cluster is much easier than the control of the users individually, ACEC first makes effort to divide the large-scale distributed system into relevant small-scale systems with the clustering algorithm. A common clustering technique is usually used to group sets of clients based on network or geographic proximity, or administrative domain. Such algorithms require the routing information, network locations and expected load of the clients to calculate the distances between clients according to some criteria. However, such algorithms need a large amount of

Department of Automation, University of Science and Technology of China, Hefei, China

Corresponding author:
Xiaofeng Jiang, Department of Automation, University of Science and Technology of China, Hefei 230021, China.
Email: jxf@mail.ustc.edu.cn

communication traffic and time, and the client clustering may cease to be effective when a number of clients leave or join the system.

In our study, the behaviors of clients cannot be controlled by the system. We use the client clustering according to IP addresses to describe the behaviors of the clients and use adaptive control to do the server clustering to control the load imbalance caused by the behaviors of clients. Therefore, a server cluster which is a small-scale system can serve the clients with similar behaviors. Since ACEC does clustering according to IP addresses, the absence or presence of one client only affects the current server cluster, the scalability of clustering can be guaranteed. In such a small-scale system, we can use the stochastic control and passive measurement to do reliable and adaptive accessibility estimation and client allocation.

The accessibility of servers is usually defined as the round-trip time (RTT) which indicates the time it takes for a data packet to be communicated between the server and a client; alternatively, it can also use the packet loss. In our work, considering the popular environment of large-scale distributed systems, we use the request arrival time and data download time to describe the accessibility and this will be described in Section 4 in detail.

Once we get the definition of the accessibility, the measurement of accessibility also becomes a problem. A common technique to measure the network accessibility is to test the network by introducing probe packets. Probes to measure the RTT and packet loss are typically done by sending groups of back-to-back packets to a server which echoes them back to the sender. But in many applications of active probing algorithms, it is unscalable and usually introduces a large amount of traffic that is not useful for the applications. The other technique for measurement is the passive measurement which can avoid the introduction of useless probe traffic. When a client runs its request in a server, the running time will be recorded as the historical measurement by the passive measurement. In our study, the passive measurement is attractive for its relatively small overhead, and thus could be desirable for many networked applications that do not require an extremely high degree of accuracy.

Then considering a small-scale system which only includes a client cluster and its corresponding server cluster, a selection algorithm based on accessibility estimation is developed. We adopt passive accessibility measurement to do accessibility estimation and introduce an attenuation coefficient to guarantee the adaptivity of historical measurements. This can significantly reduce the communication overhead in contrast to the active probing. The selection algorithm uses the stochastic control and passive measurement to do the adaptive accessibility estimation with controlled communication cost and can guarantee the access control mechanism for a small-scale distributed system which serves the clients with similar behaviors.

Finally, a communication algorithm is proposed to control the communication overhead by *drop_condition*. Trading off reducing the overhead against the accessibility, we can get a further reduction of 60% communication overhead. It also effectively makes the selection algorithm and the clustering algorithm join together.

The rest of this paper is organized as follows. The previous studies are introduced in Section 2. Section 3 describes the system model. Sections 4, 5 and 6 develop the selection algorithm, clustering algorithm and communication algorithm, respectively. Validating simulations are presented in Section 7 to demonstrate the improved accessibility of our algorithms, followed by conclusions in Section 8.

Related work

Some algorithms for the network accessibility estimation and client clustering have been proposed in previous work. Ng et al. (2003) study lightweight measurement-based optimization techniques in the peer-to-peer environment. By analyzing thousands of Internet peers and conducting trace-based and Internet-based experiments, they propose that the peer selection is inherently a very challenging problem due to the diversity in peers, even so, lightweight measurement-based techniques such as RTT probing, 10 kB transmission control protocol (TCP) probing and BNBW probing can perform reasonably well, achieving 40–50% of optimal performance. But the main factor that limits the performance of these techniques is that while they are fairly reasonable at eliminating poor peer choices, they are not very reliable in differentiating among the top peer choices. In our work, we solve this problem by sharing the measurements among the neighbor servers.

The active probing algorithm, passive measurement algorithm and cooperative measurement algorithm are summarized by Seshan et al. (1997), and a shared passive network accessibility discovery algorithm is proposed. But the work of Seshan et al. (1997) considers the model of web access which is very different from the large-scale distributed system, the algorithms cannot guarantee the adaptivity and load balance. The system model that is considered by Kim et al. (2009) is similar to ours. The work of Kim et al. (2009) proposes a resource selection algorithm with the consideration of data accessibility which is described by the second-hand accessibility estimation. Although the algorithm guarantees the adaptivity of data accessibility estimation, many requests from clients may select one server with the best data accessibility and this case will result in the load imbalance. The work of Wolski et al. (1999) proposes a mechanism Network Weather System (NWS) which maintains network sensors measuring network accessibility by periodic probing and statistical predictions based on probing results. This design is too expensive in a large-scale system.

Du et al. (2013) study the modeling and stability of multi-input multi-output networked control systems where the sensors output signals, the signals are encapsulated and transmitted to the controller in the network, and transmitted to the actuators. The data processing procedure is very similar to our system where the requests from users are transmitted to the distributed system, served and back to the users. However, there are too many channels in our system. This study considers the delay, data packet out-of-order and data packet loss for access control. Xu et al. (2012) developed an adaptive pinning control mechanism to reduce the control cost of previous algorithms for the parameter and structure

identification of complex delayed networks, since the algorithms need to add a controller on each node in the network. This study is similar to the accessibility estimation in our work. Orantes et al. (2006) propose a methodology based on learning and classification techniques and the information quantity measure, by the entropy concept, in order to address the problem of sensor location for fault identification. This study is similar to the work nodes allocation problem in our work. Yang et al. (2005) aimed to analyze the features of Internet transmission and build proper control architecture with time delay compensations efficiently to overcome the Internet time delay and data loss. This work tries to predict the accessibility of Internet.

Previous work (Krishnamurthy andWang, 2000; Andrews et al., 2002; Amini and Schulzrinne, 2004; Yuan et al., 2012) proposes some work about client clustering. Andrews et al. (2002) developed a methodology to identify and characterize Web traffic clusters. Amini and Schulzrinne (2004) proposed a client clustering algorithm which uses prefixes of IP addresses and is only for a server: the algorithm cannot run in the context of distributed systems. Krishnamurthy and Wang (2000) use prefixes derived from border gateway protocol (BGP) routing table snapshots as their clusters.

Yuan et al. (2012) focuses on utilizing the users' behaviors which are denoted by the trajectory data for location-based services. A clustering algorithm based on an index tree is proposed to collect the trajectory segments and improve the service performance.

Gossip-based dissemination is scalable and resilient to failure, previous work (Kermarrec et al., 2003; Kyasanur et al., 2006; Kim et al., 2011) uses it for data dissemination. It can also be used to achieve the selection middleware which will be described in Section 4.

System model

We consider a popular large-scale infrastructure for the distributed system. The distributed system consists of work nodes that provide computational resources for executing the requests from the clients and data nodes that store the data objects required by the requests. We adopt the context proposed by Kim et al. (2009) as the context of this distributed system. In the assumption of Kim et al. (2009), both work nodes and data nodes are connected in an overlay structure without any assumption of centralized entities for scalability. We do not assume any specific type of organization for the overlay, but assume that the overlay provides basic data access functionalities including search, store and retrieve.

Figure 1 shows the description of the distributed system model that we consider. When a client makes a request to access the distributed system, the request first arrives at one of the work nodes, the chosen work node analyzes the request and accesses the requested data objects which are served in data nodes, then the work node processes the request with requested data objects. There are two selection problems in this process. First, the client needs to select a work node which can access the request data objects with the minimal access cost. Second, the work node needs to select data nodes which have the minimal data access cost from the work nodes.

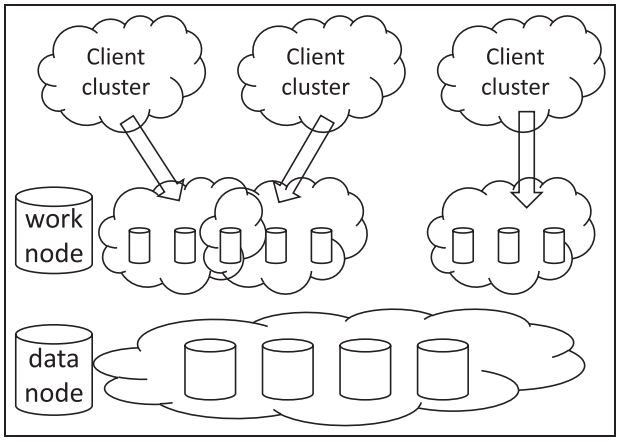

Figure 1. Distributed system model.

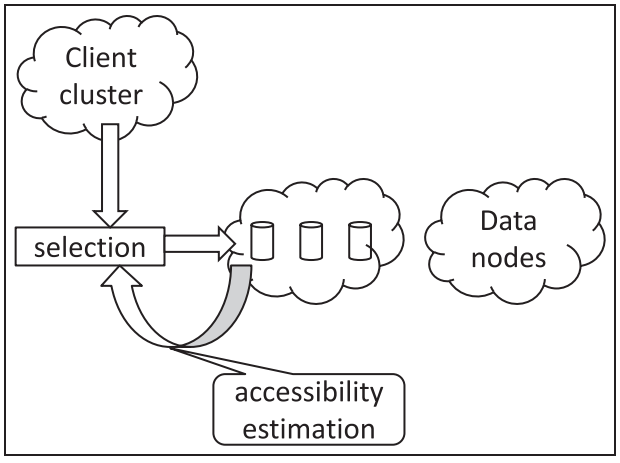

Figure 2. Small-scale distributed system model.

In our model, the clients and work nodes are both divided into many clusters, and each client cluster has a corresponding work node cluster, but the work node may be associated with each other, for example, one member of a work node cluster may also be a member of other clusters. The clients are collected into a cluster for the reason that they can access the member of corresponding work node cluster with similar access cost. We can get the following problem, some work nodes with enough load capacities and minimal access cost from the clients should be chosen and aggregated into the corresponding work node cluster for the request allocation of a client cluster.

Selection algorithm based on accessibility estimation

In this section, we show the selection algorithm based on accessibility estimation in detail, and for a clear description, we adopt the small-scale distributed system model with only one client cluster and one work node cluster. In Figure 2, we

describe the selection algorithm as a middleware between the clients and work nodes, the middleware can be achieved by an embedded device or even a communication software in front of the work nodes in a actual system, however, we will focus on this problem in Section 6.

The previous studies of Andrews et al. (2002) and Kim et al. (2009, 2011) also use passive measurement techniques to improve the network performance. Andrews et al. (2002) passively monitor incoming TCP connections of content servers to measure the distances from clients. Kim et al. (2009, 2011) perform passive network performance estimation based on the second-hand measurements which are collected by other servers without any topological restrictions. Our system differs from these studies in the following ways. First, in our study, the system model consists of two types of servers with different functions, the passive measurement of the accessibility is not only affected by the access server which serves the clients. Second, we introduce the attenuation coefficient to guarantee the adaptivity of historical measurements. Third, a communication algorithm is proposed to further implement the controllability of the passive measurement of accessibility in Section 6. Fourth, a further difference between our work is that in addition to the accessibility estimation, we use the stochastic control to adaptively predict the optimal servers to serve the clients, and the stochastic control fits well with the accessibility estimation for the reason that only a small amount of information about the accessibility of the servers can be given by the passive measurements.

Definition of accessibility estimation

The selection middleware maintains a list of the work nodes in the cluster, and makes decisions according to the accessibility estimation of each work node. Just like the description about accessibility definition in the section of introduction, in our model, the access time consists of two parts: the request arrival time that denotes the time spent for request messaging from the client to work node, and the data download time that denotes the time spent for downloading requested data objects from the data nodes. After the access time, the work node can serve the request. As a result of clustering, the estimation of request arrival time of a client in a cluster is similar with each other, so we only take care of the download time in this section. Let w_i and t_i denote the ith work node in the cluster and its download time, respectively. Let s_i denote the size of the downloaded data object, and we can get the download speed v_i of w_i from

$$v_i = s_i/t_i \tag{1}$$

Enough records of v_i can estimate the accessibility.

With equation (1), we can get a historical record of download speed of w_i, and use the historical records to estimate the accessibility u_i according to the mean criterion just like

$$u_i = 1/T \sum_{t=1}^{T} v_i^t \tag{2}$$

where T denotes the number of the historical records. However, in many cases, a work node may be removed or added to the system, the new work node may not have any record or only a few records, the estimation of such a work node must be not accurate. To solve this problem, we can utilize the accessibility estimation from the work node's neighbor nodes in the network overlay for its estimation. We assume that w_i has N neighbor nodes, let nei_j^i denote the jth neighbor node of w_i. If nei_j^i has historical records, we can get the neighbor estimation $u_{i,j}$ from equation (2). Considering the accessibility of itself and neighbor nodes, we can get the new value of u_i from

$$u_i = \rho * v_i + (1-\rho)/N \sum_{j=1}^{N} u_{i,j} \tag{3}$$

In equation (3), ρ is an equilibrium coefficient between the estimations of self and neighbor nodes, and $0 \leqslant \rho \leqslant 1$. It denotes the importance of self-estimation, and will be tested in our experiments. However, in some cases, the number of neighbors may be too large, we should restrict the value of N, and 5, 8, or 10 may be suitable, 100 must be too large.

Because the Internet always changes over time, work nodes and data nodes are being added and removed all the time and the communication latencies are changing constantly, the historical estimations will become useless over time. To guarantee the adaptivity of ACEC, we add an attenuation coefficient δ to accessibility estimation and $0 \leqslant \delta \leqslant 1$. Let u_i^k denote the accessibility estimation of w_i when the work node cluster has served $k-1$ requests, we can get the next accessibility estimation u_i^{k+1} from

$$u_i^{k+1} = (1-\delta)v_i + \delta * u_i^k \tag{4}$$

The first term and second term of equation (4) describe the impacts on the next accessibility estimation of current estimation and historical estimation, respectively. We can see that the impact of historical estimation decreases exponentially with time. The value of δ will also be tested in our experiments.

Selection algorithm

To reduce the overhead for accessibility estimation, the passive measurement is used to design the selection algorithm. This will bring a hard problem, since an accessibility estimation of a work node can be obtained only when the node is chosen to serve a request, and we can only get a small amount of information about the accessibility of the entire work node cluster. We assume that the work node cluster consists of M nodes, and assume that the accessibility of one work node has two states denoted by $S(t) = [S_1(t), \ldots, S_M(t)]$: good and bad. If the accessibility u_m is larger than a fixed value uf, we think that work node w_m has the good accessibility and $S_m(t) = 1$, otherwise it is bad and $S_m(t) = 0$. Owing to the randomness of network, we can illustrate that the state transition of a work node evolves as a two-state discrete time Markov chain in Figure 3.

For making the optimal selection, a sufficient statistic of work nodes should be given. We denote this information by $p(t) = [p_1(t), \ldots, p_M(t)]$, where $p_m(t)$ is the conditional

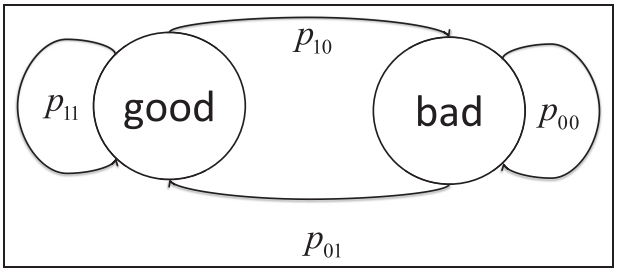

Figure 3. State Transition of Work Node.

probability that accessibility of work node w_m is good at time t given all past states and selections. The selection at time t is also denoted by $a(t)$, if $a(t) = m$, the work node w_m is selected. Owing to Markovian nature of work nodes, if a request arrive at time t, given $p(t)$ and $a(t)$, $p(t+1)$ can be obtained from

$$p_i(t+1) = \begin{cases} p_{11} & \text{if } s(t) = i \text{ and } u_i \geqslant uf \\ p_{01} & \text{if } s(t) = i \text{ and } u_i < uf \\ p_i(t)p_{11} + (1 - p_i(t))p_{01} & \text{if } s(t) \neq i \end{cases} \tag{5}$$

At time t, only the accessibility of work node indexed by $a(t)$ can be estimated, in the case $u_{a(t)} \geqslant uf$, $p_{a(t)}(t+1) = P\{S_{a(t)}(t+1) = 1 | S_{a(t)}(t) = 1\} = p_{11}$, in the case $u_{a(t)} < uf$, $p_{a(t)}(t+1) = P\{S_{a(t)}(t+1) = 1 | S_{a(t)}(t) = 0\} = p_{01}$, for other work node w_i whose accessibility can't be estimated, $p_i(t+1) = P\{S_i(t+1) = 1 | p_i(t)\} = p_i(t)p_{11} + (1 - p_i(t))p_{01}$. For the greed approach, $a(t)$ is decided to maximize the probability of good accessibility, the index $a(t)$ of selected work node is given by

$$a(t) = \arg \max_{1 \leqslant i \leqslant M} (p_i(t)) \tag{6}$$

Experiments for testing equilibrium and attenuation coefficients

In our lab which is dedicated to the research for network communication and control, a large-scale system is designed and has been running for 4 years in Shanghai to provide video on demand services. To test the values of equilibrium coefficient ρ and attenuation coefficient δ, we intercept a system log from video caching module of the system and choose 10,000 caching records for the experiments. We use *cache.*$*$ to express one member $*$ of the caching record. Since the log has recorded the caching beginning time, caching ending time and cache size of a video which are denoted by *cache.begin*, *cache.end* and *cache.size*, respectively, we can get the measurement of accessibility *cache.speed*, and it equals *cache.size*/(*cache.end* − *cache.begin*). In the experiments for accessibility estimation, the values of ρ and δ are both changing from 0 to 1 with interval 0.01. For a fixed value pair of (ρ, δ) (e.g. $(0.81, 0.87)$), we observe the hit rates which are expressed by the percentage of measurement records that have error of less than 10% with the estimations.

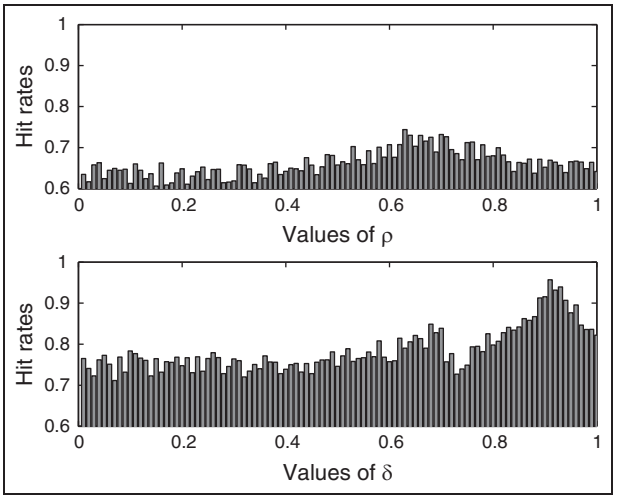

Figure 4. Test results of equilibrium and attenuation coefficients.

We first test the values of equilibrium coefficient ρ in the case that does not consider δ. In the video on demand system, 10 servers are in one group, we think that each server has 10 neighbor servers. In top part of Figure 4, we can see that the hit rates are near with 0.65 when the accessibility estimation algorithm only considers local historical records or neighbors' records, and the hit rate curve gets the vertex 0.73 when ρ obtains the value 0.67. The hit rate 0.73 is much lower than its vertex in the web service system for the reason that the video services occupy more resources, therefore, the network environment has much larger changes when video service requests join or leave the system. Then, we fix the value of ρ as 0.67, and test the values of attenuation coefficient δ. In the bottom part of Figure 4, we can see that the hit rate curve becomes higher with the growth of δ over $0 \leqslant \delta \leqslant 0.92$, and gets its vertex 0.95 when δ is 0.92, then the curve becomes lower over $0.92 \leqslant \delta \leqslant 1$. This shows us that the historical records still provide great help to accessibility estimation, but the hit rate will be down to 0.83 if we do not attenuate the impact of older historical records.

Clustering of ACEC

If all clients and work nodes are classified into one cluster, the selection middleware will have to withstand heavy loads. It will become a new performance bottleneck of the distributed system, and affect the scalability, so a clustering algorithm is essential for the mechanism of access control.

The previous studies of Andrews et al. (2002), Amini and Schulzrinne (2004), Krishnamurthy and Wang (2000) and Yuan et al. (2012) also use clustering techniques to cluster clients and assign the client cluster to the optimal server. The clustering techniques in these studies attempt to group sets of clients based on network or geographic proximity, or administrative domain. The clients can be grouped according to the network distance measured, but this algorithm needs a large amount of communication traffic and time, meanwhile the clients are more volatile than the work nodes, they may leave

or enter the distributed system at any time, so this kind of algorithm is not practical. Our clustering algorithm differs from these studies in the following ways. First, a client cluster is usually assigned to multiple servers in a large-scale system, not one server in the common environment. Second, we only use the client clustering to describe the clients' behaviors which cannot be controlled by the system. Third, in addition to the client clustering, we focus on using adaptive control to do server clustering for batch-client access.

Adaptive clustering

Fortunately, it is the case in today's Internet that many addresses with similar prefixes are near each other in the network; this is done to facilitate routing. We use the standard classless inter-domain routing (CIDR) notation, a.b.c.d/n, to represent the cluster of IP addresses that match a.b.c.d in the first n bits. Since addresses which differ only in the last 8 bits normally belong to the same tightly coupled network, we classify the clients according to the first 24 bits of the IP address, for example, two clients with IP 211.86.144.220 and 211.86.144.209 should be assigned to one client cluster 211.86.144.0/24.

With the clusters of clients, we should determine the clusters of work nodes. We first randomly select S clients from each client cluster, and use the timing of the three-way TCP handshake to give us a measure of the RTT from the selected clients to each work node, where the value of S should be a small positive integer, for example, 3, 5 and 10 may be suitable. Then, each client cluster has S lists of message RTT that denotes the time spent for messaging from the client to each work node, let $rtt^k_{i,m}$ denote the message RTT from selected client i of client cluster k to work node w_m. Then, we can calculate a list $MeanT_k$ of mean time for each client cluster k, and $MeanT_k = \{mt^k_1, mt^k_2, \ldots, mt^k_M\}$, where mt^k_m denotes the mean message RTT from client cluster k to work node m:

$$mt^k_m = 1/S \sum_{i=1}^{S} rtt^k_{i,m} \tag{7}$$

We reorder the values of mean message RTT in $MeanT_k$ in ascending order, and assign previous $clustersize_k$ work nodes in the list to a work node cluster for the client cluster k, the value of $clustersize_k$ is determined by the work loads of the client cluster and the number of clients in the cluster. If one selected work node is also assigned to other clusters, the $clustersize_k$ + 1th work node in the list should be added to the cluster, and the value of $clustersize_k$ adds 1.

Adaptive clustering algorithm based on large deviations

Because the clients are more volatile than the work nodes, they may leave or enter the distributed system at any time, the work loads of a client cluster may be changing constantly, the corresponding work node cluster should change its size adaptively to guarantee service capacity. Based on the large deviations theory, we can get more information about the trend of the work load to make a more appropriate decision. Weiss et al. (1995) introduced some large deviations techniques that have been used for analyzing models of communication networks and this paper provides a theoretical basis for our work. Yang et al. (2011) used large deviations to perform adaptive media playout control, the problem and model that they consider are similar to ours.

First, we define a random process $X(\omega, t)$ as the average of random processes of the work nodes' load percentages:

$$X(\omega, t) = 1/m \sum_{m=1}^{M} \xi_m(\omega, t) \tag{8}$$

where $\xi_m(\omega, t)$ denotes the random process of the load percentage of w_m, a sample of $\xi_m(\omega, t)$ denotes the observation of load percentage of w_m at time t. Then, we set t as a sequence of future time $\{t_1, t_2, \ldots, t_N\}$, and $\{X(\omega, t_1), X(\omega, t_2), \ldots, X(\omega, t_N)\}$ is a sequence of independent and identically distributed random variables. We define a random variable X_{mean} as the average of the sequence

$$X_{mean} = 1/N \sum_{n=1}^{N} X(\omega, t_n) \tag{9}$$

The probability distribution of X_{mean} can clearly describe the trend of work load of work node cluster before time t_N. With large deviations theory, we can get the probability that mean load percentage is larger than ε:

$$\begin{aligned} &P(X_{mean} \geqslant \varepsilon) \\ &= P\left(\frac{X(\omega, t_1) + X(\omega, t_2) + \ \ldots \ + X(\omega, t_N)}{N} \geqslant \varepsilon\right) \\ &= \exp^{-N * l(\varepsilon) + O(1/N)} \end{aligned} \tag{10}$$

$$l(\varepsilon) = \sup_{\theta} (\theta * \varepsilon - \log g_{X(\omega, t_n)}(\theta)) \tag{11}$$

$$g_{X(\omega, t_n)}(\theta) = E\{\exp^{\theta X(\omega, t_n)}\} \tag{12}$$

To get the probability distribution of $X(\omega, t_n)$, we repeatedly retrieve the mean load of the work node cluster, and choose recent k results which is denoted by $load^1, load^2, \ldots, load^k$. The approximate probability distribution can be given as follows:

$$P(X(\omega, t_n) = load^i) = 1/k \tag{13}$$

$$g_{X(\omega, t_n)}(\theta) = 1/k \sum_{i=1}^{k} \exp^{\theta load^i} \tag{14}$$

$$l(\varepsilon) = \sup_{\theta} (\theta * \varepsilon - \log (1/k \sum_{i=1}^{k} \exp^{\theta load^i})) \tag{15}$$

For $l(\varepsilon)$, we calculate its derivative with respect to θ, and make the derivative equal to zero. We can obtain:

$$\varepsilon = \sum_{i=1}^{k} load^i \exp^{\theta load^i} / \sum_{i=1}^{k} \exp^{\theta load^i} \tag{16}$$

Using mathematical software, such as Matlab, we can get a numerical solution of equation (16) which is denoted as $\theta^+(\varepsilon)$.

We place this value into equation (10) and give a simplified notation P^{ε}:

$$P(X_{mean} \geqslant \varepsilon) = \exp^{-N*\sum_{i=1}^{lrnum} load^i \exp^{\theta load^i} / \sum_{i=1}^{lrnum} \exp^{\theta load^i}} = P^{\varepsilon} \tag{17}$$

Case 1. In this case ε^+ and P^+ are the load and probability thresholds that are used to determine whether the work node cluster is overloaded, and $0\% < \varepsilon^+ < 100\%$, $0 < P^+ < 1$. Values of 80% and 0.7 may be suitable for their values. In the algorithm, if $P^{\varepsilon^+} > P^+$, which means that the probability that the load percentage of work node cluster will be more than 80%, is more than 0.7, the work node cluster should add a new work node according to the message RTT list $MeanT_k$.

Case 2. In this case ε^- and P^- are the load and probability thresholds that are used to determine whether the work node cluster is too light, and $0\% < \varepsilon^- < 100\%$, $0 < P^- < 1$. Values of 20% and 0.7 may be suitable for their values. First, we want to get the value of $P(X_{mean} \leqslant \varepsilon^-)$ which denotes the probability that the load percentage of work node cluster will be less than 20%:

$$P(X_{mean} \leqslant \varepsilon^-) = 1 - P(X_{mean} \geqslant \varepsilon^-) = 1 - P^{\varepsilon^-} \tag{18}$$

In the algorithm, if $1 - P^{\varepsilon^-} < P^-$ which means that the probability that the load percentage of work node cluster will be less than 20% is more than 0.7, the work node cluster should delete a work node according to $MeanT_k$.

Communication algorithm of selection middleware

In above sections, we describe the function of selection middleware, and declare that it can be achieved by a communication software in front of the work nodes. Here, we will give the specific communication algorithm of the software called CASM.

With the clustering algorithm running on clients and work nodes, the large-scale distributed system is divided into a hierarchical structure, and this solves two natural problems of most of the message dissemination algorithms: (1) they rely on each peer having knowledge of the global membership and (2) being oblivious to the network topology, they can impose a high load on network links when applied to wide-area settings. In CASM, each work node only needs to have knowledge of the membership of the cluster, and since the messages are spreading only in a work node cluster, the growth of system size can merely bring the linear growth of communication overhead, rather than the exponential growth.

Algorithm CASM:

Actions of work node w_m:

$getpe(v_m)$, $receive(message\ mes_j^m)$, $send(message\ mes_m^j)$

$getpe(v_m)$

Precondition:

$v_m \neq NULL$

Effect:

Use the values of v_m and the neighbor nodes' accessibility estimations stored in $receive_{buff}$ to calculate the new accessibility estimation of w_m according to equations (2) and (3). With u_m, we judge the value of $drop\_condition$, if $drop\_condition = false$, put u_m into $send_{buff}$. Then clear $receive_{buff}$ and set v_m as $NULL$.

$receive(messagemes_j^m)$

Precondition:

Receive a message mes_j^m which includes the accessibility estimation $u_{i,j}$ of neighbor node nei_j^i.

Effect:

Add message mes_j^m to $receive_{buff}$, and replace the member with the same index j.

$send(messagemes_m^j)$

Precondition:

$send_{buff} \neq NULL$

Effect:

Broadcast message mes_m^j to any nei_j^i, and mes_m^j includes new accessibility estimation u_m, then clear $send_{buff}$.

In the description of algorithm CASM, we can see that work node w_m performs action $getpe(v_m)$ when it gets a measurement of itself v_m. Then with this measurement and the neighbor's accessibility estimation, w_m calculates the accessibility estimation of itself u_m. With the new accessibility estimation, w_m calculates the value of $drop\_condition$ to judge whether should the new accessibility estimation be disseminated to its neighbor nodes in the work node cluster. We define $drop\_condition$ as follows:

$$drop\_condition : \frac{|u_m - u'_m|}{u_m} > \tau$$

Here u'_m is set as the old accessibility estimation, τ is a threshold for describing the similarity between old and new accessibility estimations, and $0 \leqslant \tau \leqslant 1$. There are two special cases: (1) if $\tau = 1$, $drop\_condition$ is always $false$, and all of the estimations will be broadcast to the neighbor nodes of w_m; (2) if $\tau = 0$, $drop\_condition$ is always $true$, and the estimations will never be broadcast to the neighbor nodes of w_m. So we can control message dissemination among work nodes by adjusting the value of τ, the following experiments show us that the value 0.80 of τ can give ACEC a good performance.

We test the impact of communication algorithm CASM on the accessibility of ACEC. Here, we adopt the simulation environment which will be introduced in Section 7, and generate a task request trace to run the selection algorithm of ACEC. In the top part of Figure 5, we can see that the accessibility of ACEC has the linear growth with the growth of τ which can control the number of messages in the system. We set the number of all estimations as the total number of messages, and the number of messages in the system is expressed

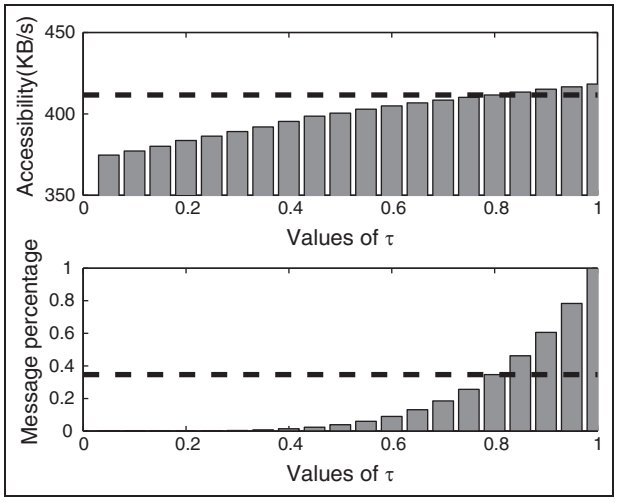

Figure 5. The impact of the communication algorithm on accessibility.

as the percentage in bottom part of Figure 5. We can see that the estimations disseminated in 60% messages have error of less than 0.2 when $\tau = 0.8$ and the accessibility is similar with the best. Therefore, $\tau = 0.8$ is a suitable value for the communication algorithm.

Experimental results

We construct a large-scale distributed system that consists of 170 work nodes and 50 data nodes to observe the performance of ACEC. The data objects distributed in data nodes have 5 sizes: 5, 20, 50, 100 and 200 MB. We then generate a series of random requests for downloading different objects, and distribute them to the work nodes according to the access control mechanism.

Since the large-scale distributed systems have many different characteristics from the conventional system, we observe the performance of ACEC with the following considerations: the impact of clustering algorithm, the performance over time, the impact of historical record size, the impact of neighbor size, the impact of data size, the impact of multi-object access, the impact of the loss of work nodes. To show the effectiveness of ACEC, we also compare the performance of ACEC with the following existing algorithms:

- Random algorithm (RAN): randomly selects servers for requests from clients.
- Latency-based algorithm (LA): selects the servers according to their latencies.
- Optimal algorithm (OP): selects the best servers for all clients, and we ignore the time for finding the servers.
- Network-aware clustering (NA) (Krishnamurthy and Wang, 2000): uses the information available from BGP routing table snapshots to do client clustering, and moves content to groups of clients.
- Clustering and Server Selection using Passive Monitoring (CSPM) (Andrews et al., 2002): performs the client clustering with the information from monitoring the TCP connections and assigns each cluster to an optimal content server.
- Clustering based on network proximity and traffic modeling (CNT) (Amini and Schulzrinne, 2004): assigns groups of clients to servers according to the network location and expected load generated by these clients.
- Trajectory-clustering (TC) (Yuan et al., 2012): performs the client clustering with the trajectory data for location-based services.
- Resource selection using passive network performance estimation (RSPE) (Kim et al., 2011): exploits second-hand measurements collected by other servers without any topological restrictions to do passive performance estimation for resource selection.

Impact of the clustering algorithm

Since the clustering algorithm divides the large-scale system into relevant small-scale systems, the adaptivity of the clustering algorithm has a major impact on the adaptivity of the access control mechanism. In addition, the load balance is also ensured by the clustering algorithm, we should investigate the performance of the clustering algorithm. We send a series of task requests from a client cluster to the larger-scale distributed system and then observe the changes of mean load percentage and the number of work nodes. These two measurements can clearly show the adaptivity and load balance of the corresponding work node cluster.

In the initial step, we set the request arrival rate (i.e. number of arrival task requests in unit time) as 1000, and the number of work nodes is 50. The clustering algorithm makes the decision to add or delete a work node in each time step, and each time step denotes 100 task requests that have been sent from the client cluster. We adjust the request arrival rate per 100 time steps to test the adaptivity of the algorithm, and the sequence of arrival rates is set as $\{1000, 500, 700, 1400, 300, 800, 900, 200, 1300, 400\}$. We can see that the expected load of the client cluster changes dramatically. Here, we compare the clustering algorithm of ACEC with NA, CSPM, CNT and TC which also focus on the clustering algorithm, and use the notation $AC(\varepsilon^+, \varepsilon^-)$ to denote the clustering algorithm of ACEC with the key parameters $P^+ = P^- = 0.7$ and $(\varepsilon^+, \varepsilon^-)$.

In Figure 6, we can observe the changes of the clustering algorithms' curves per 100 time steps to examine the adaptivity. NA only assigns the client cluster to the work nodes and does not change the number of work nodes, the mean load and number of work nodes remain the same. The adaptivity of NA cannot be guaranteed. Since CSPM and CNT use the passive and active information which is relevant to the expected load generated by the clients to perform clustering, respectively, they can improve the adaptivity of the work node cluster. Here, the trajectory which is used in TC is denoted by the trajectory of data access, consequently, TC also uses the passive measurement to do clustering, however, this information is not directly relevant to the load, which makes TC worse at balancing the load.

ACEC is more sensitive to the changes of request arrival rates than CSPM, CNT and TC. In the beginning of each 100 time steps, if the mean load is larger than ε^+, ACEC will add

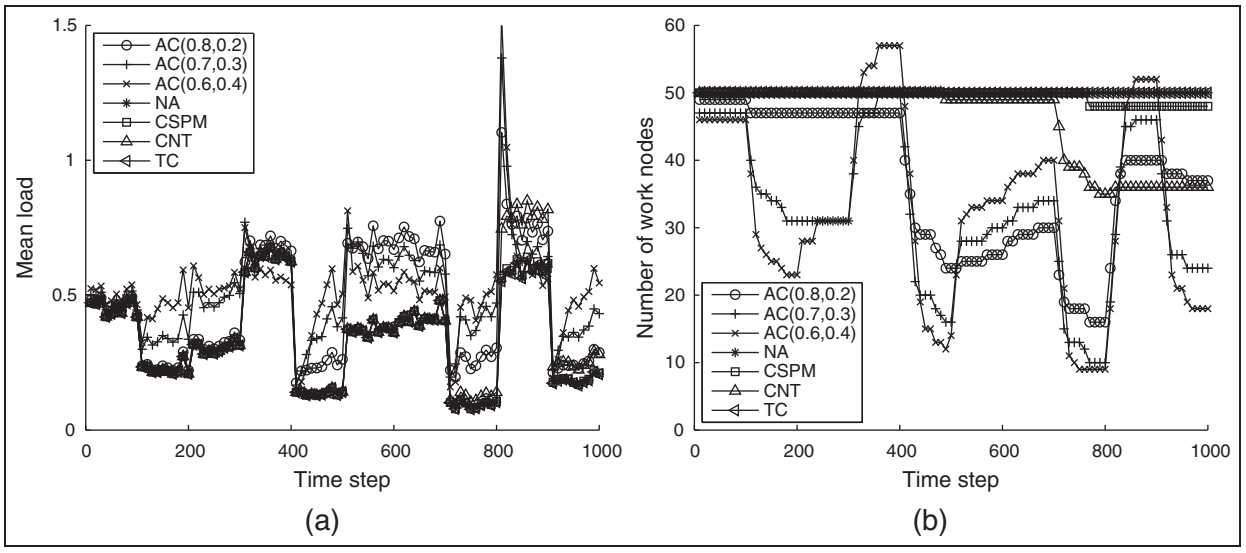

Figure 6. Test results of the clustering algorithm: (a) mean load of the work node cluster; (b) number of work nodes.

work nodes to decrease the mean load, if the mean load is smaller than ε^-, ACEC will delete work nodes to increase the mean load. We can see that the smaller the value of ε^+ the faster the operations for adding work nodes, and the larger the value of ε^- the faster the operations for deleting work nodes. However, the mean load becomes 150% in 810th time step when $\varepsilon^+ = 0.6$, the work load cluster has been too overloaded for the reason that the growth of work nodes is lower the growth of requests, we can solve this problem by adjusting the step size or judgement frequency of the clustering algorithm.

Performance over time

We present the performance comparison over time to see the basic performance and stability of the access control mechanisms. As the default, we set the historical record size and the neighbor size to 50 and 10, respectively (and they are applied to all of the following experiments, unless otherwise mentioned). In the following, we make performance comparisons between ACEC and RAN, LA, OP, CSPM, CNT, TC, RSPE. Since NA is always weaker than CSPM and CNT, we omit its comparison. To express the performance, here, we use the mean download speed which is measured within current 10 seconds, therefore, it has dramatic fluctuations.

In Figure 7, we can see that ACEC has the best performance among the existing algorithms (OP ignores the time for making decisions). RAN has poor performance and large variations. LA performs better than RAN. CSPM, CNT and TC make effort to cluster the clients based on the load information, and have better performance than LA. Since RSPE needs some historical records to perform performance estimation, it has worse performance than CSPM and CNT at the beginning time, it performs better after 4000 seconds. ACEC also needs historical records to do accessibility estimation, however, the work nodes can help each other, a slight tilt of performance appears at the beginning time.

Impact of historical record size

The historical record size (T, introduced in Section 4.1) has a major impact on the accessibility estimation before allocating clients, therefore, we investigate the impact of historical record size on the access control mechanisms. Here, the performance is denoted by the mean download speed which is measured within 10,000 seconds (and this definition is applied to all the following experiments). Figure 8 shows that the performance of RAN and LA does not change with respect to the historical record size. The performance of CSPM, CNT, TC, RSPE and ACEC increase as the historical record size increases. However, the performance of CSPM, CNT, TC and RSPE decreases when the historical record size is larger than 100 for the reason that the old historical records become useless for allocating clients. Since ACEC attenuates the impact of old historical records, ACEC not only works better than the exiting algorithms across historical record sizes, but also improves as the historical record size increases.

Impact of neighbor size

The historical records in neighbor nodes are very useful for the nodes with only a few prior observations in the large-scale distributed system which has a large amount of operations for adding or deleting servers. We investigate how the access control mechanisms work over different neighbor sizes. Figure 9 shows that the performance of RAN, LA, CSPM, CNT and TC does not change with respect to the neighbor size. The performance of RSPE and ACEC increases as the neighbor increases. This result implies that the proposed algorithm can work better when the nodes have more neighbors.

Impact of data size

The size of accessed data objects can vary depending on the requirements of clients in the large-scale systems.

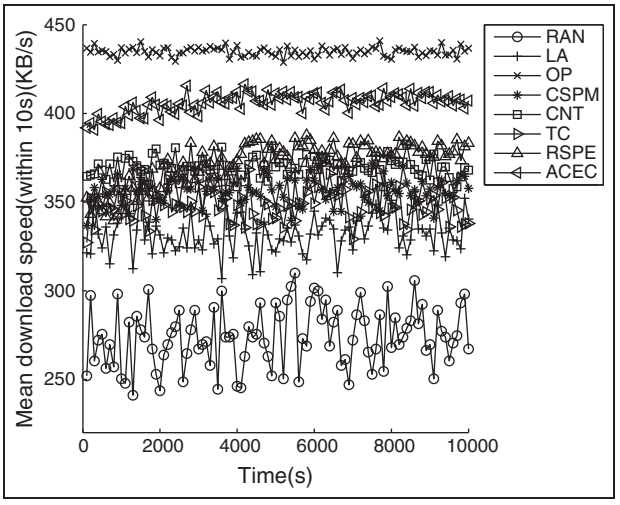

Figure 7. The performance over time.

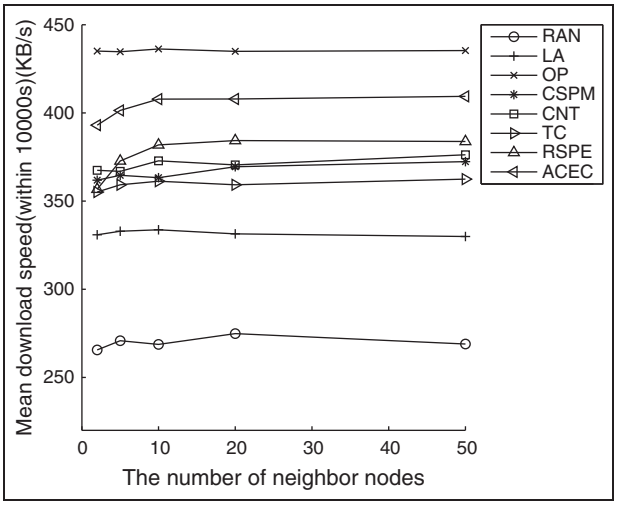

Figure 9. Impact of neighbor size.

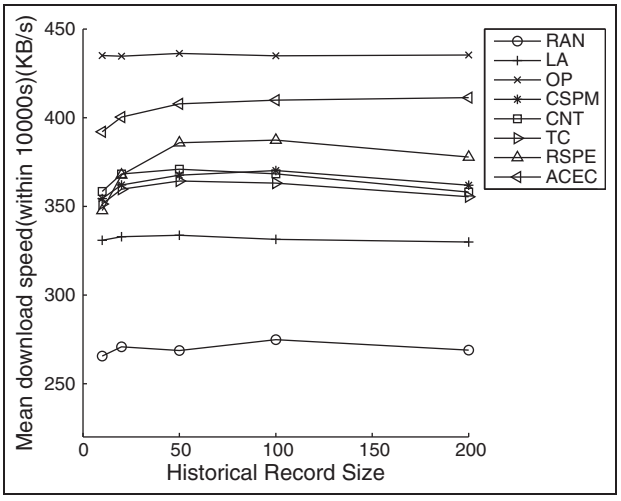

Figure 8. Impact of historical record size.

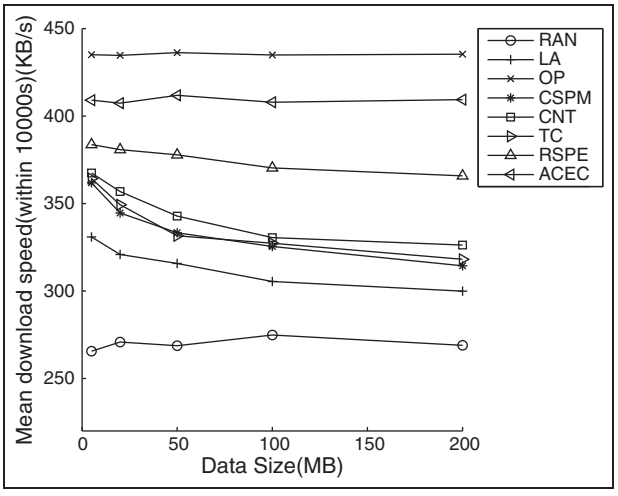

Figure 10. Impact of data size.

We investigate how the access control mechanisms respond across different data sizes. Five task request traces with different data sizes are generated respectively to run the access control mechanisms in the large-scale system, and the performance comparisons are shown in Figure 10. We can see that the performance of RAN does not change with respect to the data size, the performance of LA, CSPM, CNT, TC and RSPE decreases as the data size increases. In contrast, ACEC gets better performance than the other algorithms and keeps consistent with the growth of data sizes. This result implies that the proposed algorithm can work consistently across different data sizes, and it is useful for intensive data requirements.

Impact of multi-object access

In the large-scale distributed system, many clients request multiple data objects, which means that multiple data objects are accessed to complete one task. The performance may decrease for the main reason that the most appropriate servers cannot meet the requirements and some tasks have to access the worse servers. In Figure 11, we investigate the impact of multi-object access. We can see that the performance of all of the algorithms decreases as the number of data objects increases. However, the performance of ACEC decreases more slowly, and it approaches OP. The result implies that the proposed algorithm can work consistently across different numbers of accessed objects.

Impact of loss of work nodes

Since the work nodes in large-scale distributed systems are not very reliable, it is important to observe how the access control mechanisms respond across different scales of system failure. Here, we randomly make some work nodes leave the system

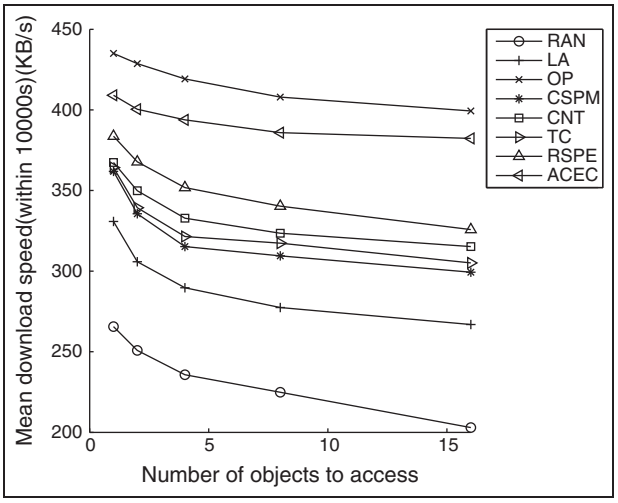

Figure 11. Impact of multi-object access.

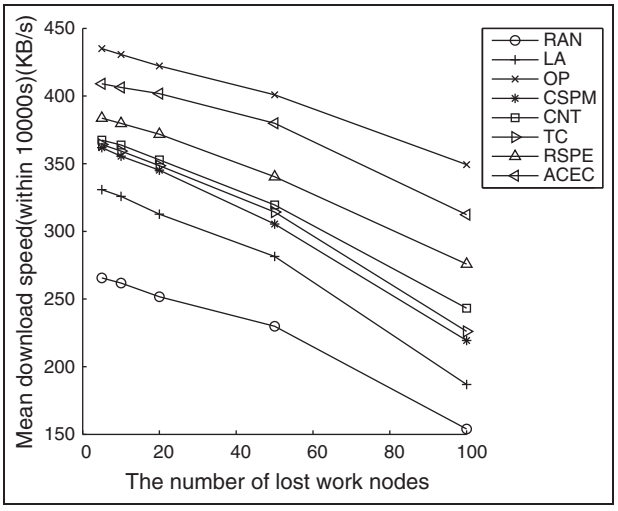

Figure 12. Impact of loss of work nodes.

per 1000 seconds to simulate the loss of work nodes, these work nodes will rejoin the system after 100 seconds. In Figure 12, we investigate the impact of the loss of work nodes. We can see that the performance of all of the algorithms decreases as the number of lost work nodes increases as expected. It is different from Figure 11, the decline rate of the performance increases as the number of lost work nodes increases. The result implies that the proposed algorithm works better across different numbers of lost work nodes.

Conclusion and future work

An effective mechanism of access control can help a large-scale distributed system reduce the access time of its clients and improve the performance of services. This work has described a mechanism of access control based on accessibility estimation and client clustering: ACEC. The description of ACEC algorithm shows that both selection and clustering algorithms can guarantee load balance and adaptivity. The experiments test the key coefficients of selection algorithm in the environment of online video services and compare the performance of ACEC with other algorithms. The experimental results show that ACEC performs better.

There are still several main studies in future. We first want to work for better clustering algorithm with the consideration of semantic analysis. Then we want to design a collaborative algorithm between work nodes and data nodes for further access control. At last, we want to implement the ACEC algorithm in the practice multimedia distributed system for access control of multimedia data.

Funding

This work was partly supported by the National Natural Science Foundation of P.R. China (grant numbers 61074033 and 61233003) and the Doctoral Fund of the Ministry of Education of P.R. China (grant number 20093402110019).

References

Amini L and Schulzrinne H (2004) Client clustering for traffic and location estimation. In: *Proceedings of the 24th International Conference on Distributed Computing Systems*, pp. 730–737.

Andrews M, Shepherd B, Srinivasan A, Winkler P and Zane F (2002) Clustering and server selection using passive monitoring. In: *Proceedings of IEEE INFOCOM*, pp. 1717–1725.

Du D, Fei M and Jia T (2013) Modelling and stability analysis of MIMO networked control systems with multi-channel random packet losses. *Transactions of the Institute of Measurement and Control* 35(1): 66–74.

Kermarrec AM, Massoulie L and Ganesh AJ (2003) Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems* 14(3): 248–258.

Kim J, Chandra A and Weissman JB (2009) Using data accessibility for resource selection in large-scale distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 20(6): 788–801.

Kim J, Chandra A and Weissman JB (2011) Passive network performance estimation for large-scale, data-intensive computing. *IEEE Transactions on Parallel and Distributed Systems* 22(8): 1365–1373.

Krishnamurthy B and Wang J (2000) On network-aware clustering of web clients. In: *Proceedings of SIGCOMM '00*, pp. 97–110.

Kyasanur P, Choudhury R and Gupta I (2006) Smart gossip: an adaptive gossip-based broadcasting service for sensor networks. In: *Proceedings IEEE International Conference Mobile Ad Hoc and Sensor Systems*, pp. 91–100.

Ng TSE, Chu YH, Rao SG, Sripanidkulchai K and Zhang H (2003) Measurement-based optimization techniques for bandwidth-demanding peer-to-peer systems. In: *Proceedings of IEEE INFOCOM*, pp. 2199–2209.

Orantes A, Kempowsky T and Le Lann M-V (2006) Classification as an aid tool for the selection of sensors used for fault detection and isolation. *Transactions of the Institute of Measurement and Control* 28(5): 457–479.

Seshan S, Stemm M and Katz RH (1997) SPAND: shared passive network performance discovery. In: *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems*, pp. 135–146.

Weiss A (1995) An introduction to large deviations for communication networks. *IEEE Journal on Selected Areas in Communications* 13(6): 938–952.

Wolski R, Spring N and Hayes J (1999) The Network Weather Service: a distributed resource performance forecasting service for

metacomputing. *Journal of Future Generation Computing Systems* 15: 757–768.

Xu J, Zhang J and Tang W (2012) Parameters and structure identification of complex delayed networks via pinning control. *Transactions of the Institute of Measurement and Control*, in press.

Yang J, Hu H, Xi HS and Hanzo L (2011) Online buffer fullness estimation aided adaptive media playout for video streaming. *IEEE Transactions on Multimedia* 13(5): 1141–1153.

Yang SH, Chen X, Tan LS and Yang L (2005) Time delay and data loss compensation for Internet-based process control systems. *Transactions of the Institute of Measurement and Control* 27(2): 103–118.

Yuan G, Xia S, Zhang L, Zhou Y and Ji C (2012) An efficient trajectory-clustering algorithm based on an index tree. *Transactions of the Institute of Measurement and Control* 34(7): 850–861.

Copyright of Transactions of the Institute of Measurement & Control is the property of Sage Publications, Ltd. and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.