CloudGPS: A Scalable and ISP-Friendly Server Selection Scheme in Cloud Computing Environments

Cong Ding[1], Yang Chen[2,†], Tianyin Xu[3,†], and Xiaoming Fu[1]
[1]Institute of Computer Science, University of Goettingen, Germany
[2]Department of Computer Science, Duke University, USA
[3]Department of Computer Science and Engineering, U.C. San Diego, USA
Email: {cong, fu}@cs.uni-goettingen.de, ychen@cs.duke.edu, tixu@eng.ucsd.edu

***Abstract*—In order to minimize user perceived latency while ensuring high data availability, cloud applications desire to select servers from one of the multiple data centers (i.e., server clusters) in different geographical locations, which are able to provide desired services with low latency and low cost. This paper presents CloudGPS, a new server selection scheme of the cloud computing environment that achieves high scalability and ISP-friendliness. CloudGPS proposes a configurable global performance function that allows Internet service providers (ISPs) and cloud service providers (CSPs) to leverage the cost in terms of inter-domain transit traffic and the quality of service in terms of network latency. CloudGPS bounds the overall burden to be linear with the number of end users. Moreover, compared with traditional approaches, CloudGPS significantly reduces network distance measurement cost (i.e., from $O(N)$ to $O(1)$ for each end user in an application using N data centers). Furthermore, CloudGPS achieves ISP-friendliness by significantly decreasing inter-domain transit traffic.**

I. Introduction

Nowadays, the Internet has evolved to the cloud computing era, which provides an elastic and stable infrastructure for hosting online applications such as web searching, e-mailing, instant messaging, online social networking, and online gaming. *Cloud infrastructure providers* (CIPs) allocate numerous computation nodes to form geographically distributed data centers, which becomes to a popular infrastructure for hosting Internet applications, according to its attractive features such as service-level agreement (SLA), auto scaling, cheap/accountable pricing, and negligible front-end investment. For a *cloud service provider* (CSP), in order to serve the huge and still increasing number of users distributed all over the world, reserving computation nodes from data centers located at various geo-locations is a desirable solution. This solution can minimize the users' perceived latency and increase reliability in terms of reducing service outages [13]. Given a set of reserved service-hosting nodes, a CSP is facing the challenge of *server selection problem*, i.e., how to arrange the mapping between service-hosting nodes and end users in a scalable way to satisfy both end users' and CSPs' needs. On one hand, the selection should improve users' quality of experience. Since end users would expect low end-to-end latency in various applications, such as Voice over IP and online/mobile gaming [4], [19], the CSP should arrange every user a close service-hosting node. On the other hand, such selection should provide a load-balancing among service-hosting nodes, in order to effective utilize the CSP's investment. Moreover, the traffic generated by CSPs should be ISP-friendly, since these globally distributed service-hosting nodes would be located within different ISP domains and the *inter-domain transit traffic* bandwidth is limited and expensive [10].

There are several existing ways for server selection. One straightforward method is to completely grant the choice rights to the users. Intuitively, a user will choose its "closest" server for obtaining the lowest access latency. However, such selection does not consider servers' workloads, which may lead to an outage of some over-capacity servers. This weakness can hardly be fixed by placing servers according to a certain distribution because users' online and offline behaviors are highly dynamic and unpredictable. Moreover, the method imposes high burdens on user clients, especially mobile devices with constrained energy and bandwidth. Although service-hosting nodes from the same data center are co-located, still the number of data centers can be large. Emerson Network Power has reported there were 509,147 data centers worldwide as of December 2011. Let's take a single vendor as an example, there are more than 1,158 server clusters in Akamai in 2008 [15]. These numbers are still increasing. Thus for a particular user, it takes a long time to do the measurement. Based on our test by Synaptic Package Manager on Ubuntu version 10.04 LTS using a desktop computer located in University of Goettingen, it takes about 45 seconds to select the closest Ubuntu mirror, which consists of 348 servers. The time cost is much more than acceptable for most cloud computing services, especially those real-time applications.

Another method is to let CSPs or public CIPs handle the server selection. For example, Akamai uses a centralized hierarchical stable-marriage algorithm for pairing clients with its CDN servers [23]. The centralized architecture causes an

† This work was done when Yang Chen and Tianyin Xu were with Institute of Computer Science, University of Goettingen, Germany.

978-1-4673-1298-1/12/$31.00

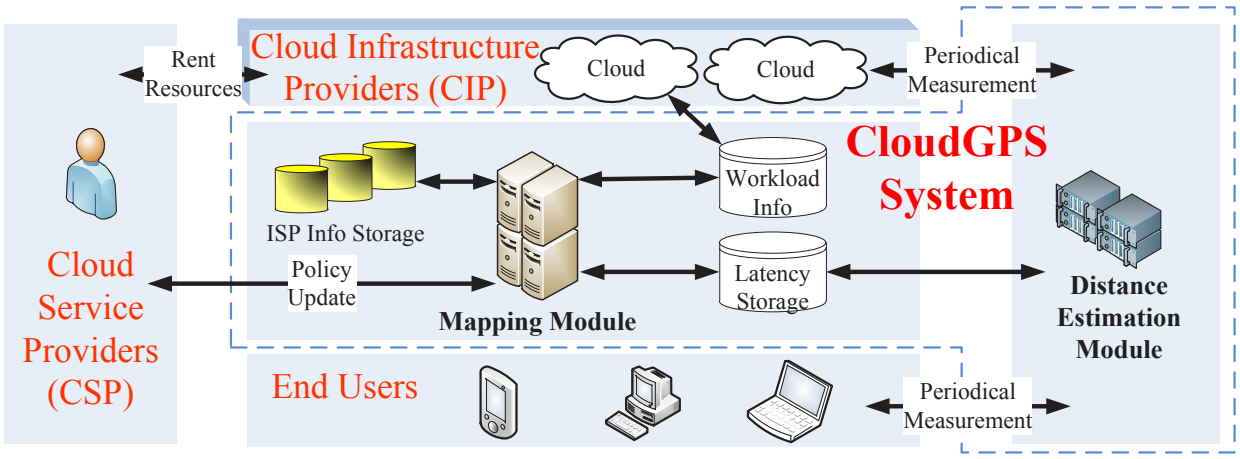


Fig. 1. System Architecture

extremely large overhead, adds additional delay, and makes the systems less responsive to sudden changes in client request rates (i.e., flash crowds), so it is unsuitable to act as an outsourcing system to provide services for all the commercial clouds [23]. YouTube uses a hash-based mechanism to share static loading, and a location-aware DNS resolution to perform a semi-dynamic approach for its video sharing [3]. However, both of them put unnecessary burdens on CSPs than outsourcing to a third organization because even the simplest approach requires a whole distributed mapping system including distributed domain name servers (DNSs) and mapping nodes [23]. To solve the above-mentioned problems, some outsourcing mapping systems (e.g., DONAR [23]) made up of dedicated nodes are newly proposed. For example, DONAR considers both proximity and server load in their selection policy. However, this selection scheme has the following two problems. Firstly, it requires a full measurement to all related data centers, thus it cannot scale to larger amount of server locations due to long measurement time. Secondly, it is not an ISP-friendly solution since it does not consider the ISP operational cost due to the inter-domain transit traffic [7], [24].

In this paper, we propose CloudGPS, a scalable and ISP-friendly server selection scheme to address the above challenges. In CloudGPS, our goals include: (1) *Scalability:* The CloudGPS system has to be scalable with the rapid-growing of the applications, as well as the huge user scale and wide user distribution. (2) *ISP Friendliness:* In CloudGPS, we take the ISP's economic profit into serious consideration. We aim at reducing the inter-domain transit traffic, while still letting users to access their nearby servers. Client nodes should give preferences to server clusters in the same ISP or peering ISPs in addition to proximity. (3) *Decentralization:* Centralized systems introduce a single point of failure, as well as an attractive target for attackers. As a result, a decentralized system for server selection is desired. CloudGPS builds a bridge between end users and CSPs for better user-server mapping. By considering different behaviors of the stable servers and dynamic users, we enhance the existing network coordinate (NC) techniques for an accurate latency prediction. Moreover, CloudGPS proposes a novel global performance optimization for the server selection, by considering the ISP-friendliness. In summary, the paper makes the following three contributions:

1) We propose a novel server selection middleware for cloud applications named CloudGPS, which performs efficient user-server mapping in a decentralized way to scale up to wider server and user distributions, involving several important issues such as proximity, server workload, as well as ISP-friendliness.
2) We design a novel distance estimation component in CloudGPS, which takes the advantage of two different network coordinate (NC) techniques – distributed NC and landmark-based NC to position the user clients with high resilience to churn.
3) We evaluate CloudGPS based on the real-world Internet traces collected by the Meridian project [2]. The results verify that CloudGPS matches our design goals by satisfying all involved entities (i.e., CSPs, CIPs, ISPs, and end users).

The rest of the paper is organized as follows. We elaborate CloudGPS's system design in Section II including the mapping system and the distance estimation system. Section III provides the performance evaluation of CloudGPS when applied in the most popular network topology dataset. Section IV concludes the paper and envisions the future work.

II. CloudGPS: System Design

There are four kinds of entities in the cloud computing application architectures: *Internet service providers* (ISPs), *cloud infrastructure providers* (CIPs), *cloud service providers* (CSPs), and *end users*. CIPs, who use ISPs' Internet connection services, own the cloud computing infrastructure and lease

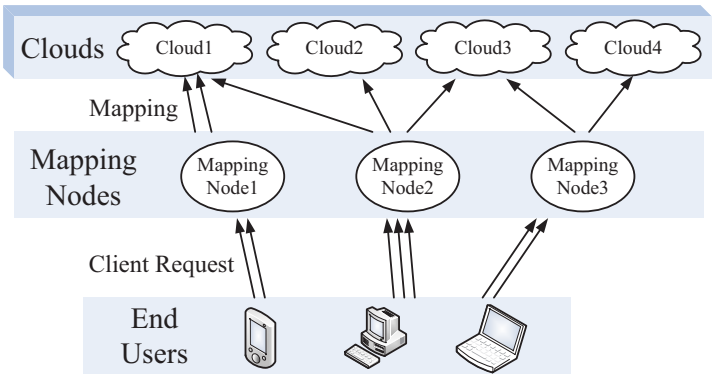


Fig. 2. Mapping Module

the resources as instances to CSPs. CSPs use the resources reserved from the CIPs to serve end users.

As shown in Fig. 1, CloudGPS builds a middleware to provide outsourcing selection service, aiming at improving network performance, balancing server workload, and reducing inter-domain transit traffic.

There are two main components in the CloudGPS system, *mapping module* (MM) and *distance estimation module* (DEM). MM allows CSPs to choose different selection policies. MM obtains server workload information, ISP information, and latency information between servers and clients from three corresponding databases (ISP Info Storage, Workload Info, Latency Storage) respectively. Based on these information, it makes server selection decisions using the algorithm described in Section II.A. The ISP Info Storage database stores the AS (typically an ISP) relationships (i.e., peering or not) and IP-AS mapping information. CloudGPS obtains the daily snapshot of AS relationships published by UCLA Internet Topology Collection project [25], and collect the IP-AS mapping information from the Oregon RouteViews project [1]. The UCLA Internet Topology Collection project [25] publishes latest AS-level topology graph by collecting the information from BGP routing tables and routing updates, route servers, and looking glasses; the Oregon RouteViews project [1] is designed for Internet operators to obtain real-time information about the global routing system from the perspectives of several different backbones and locations around the Internet. The real-time server workload information (Workload Info database) is updated by the servers, and the latency information (Latency Storage database) are updated by DEM. DEM measures network distances among servers and distances between every client and a fixed number of servers which are chosen randomly. Moreover, DEM calculates and estimates the latency information between clients and other servers based on these measurement results and stores them to the Latency Storage database. The partial measurements use tools such as King [14], while calculation and estimation use the algorithm described in Section II.B. King [14] is a tool to estimate the round-trip time (RTT) between any two arbitrary hosts in the Internet by estimating the RTT between their domain name servers. Both Workload Info database and Latency Storage database store local information only, and the decentralized mapping algorithm ensures each local mapping node to make the server selection decision based on the local workload and latency information from local storages and global information from other mapping nodes.

A. Mapping Module

Mapping Policy. We propose to use a global performance function to minimize the network cost, and introduce *inter-domain transit traffic penalty coefficient* to balance the reduction of inter-domain transit traffic and the minimization of user-server latency. There are two parameters should be configured and updated in CloudGPS, the inter-domain transit traffic coefficient and the capacity of each server cluster. The CSPs update the inter-domain transit penalty coefficient directly with any mapping node, to determine the degree of pairing a client with a server in the same domain. CSPs also update the capacity information of each server cluster to the MM periodically. MM is composed of a cluster of mapping nodes distributed in multiple geographical locations. Each mapping node obtains ISP information from ISP Info database, current workloads of all the local server clusters from local Workload Info database, and network distances between servers and local clients from local Latency Storage database, respectively. Global server workload information (only a constant number of values, will be described later) are shared among mapping nodes by gossip-based dissemination algorithms [12]. Each mapping node pushes the global server workload information to its neighbors every 10 minutes. [16] proves that if there are M nodes and each node gossips to $\log(M) + c$ other nodes on average, the probability that everyone gets the message converges to $e^{-e^{-c}}$, which is very close to 1 without considering failures. The number of gossip rounds necessary to spread global workload information to all the mapping nodes respects $\log(n)/\log(\log(n))$ [11],

which shows that it takes at most a logarithmic number of steps to reach every mapping node. For example, there are $M = 100$ mapping nodes, it cost about 6 rounds (i.e., 1 hour) to spread the global workload information to all the mapping nodes if each mapping node spread its information to 5 neighbors on average. In this case, each mapping node solves the global performance optimization problem using the parameters obtained through the policies above, and thus pairs each client to the expected server cluster, as shown in Fig. 2. The decentralized server selection algorithm guarantees the selection of local mapping nodes are global optimal.

Server Selection Algorithm. Three important metrics should be considered in the global server selection problem: (1) network performance, (2) server workload balance, and (3) inter-domain transit traffic. Large latency between clients and servers causes poor user experience and network performance; imbalanced workload of servers may cause a large overhead for specific servers, which increases the risk of server breaking down. The inter-domain transit traffic produces unnecessary ISP operational cost. Our goal is to minimize the network cost, balance client requests across servers, and reduce inter-domain transit traffic. However, improving one of these components typically comes at the expense of the others. Thus, we allow our customers (i.e., CSPs) to configure the parameters to satisfy their willingness for the trade-off among these three factors.

An objective function is desired for seeking an optimal user-server mapping. Subject to the pre-configured load balancing requirement, we try to minimize the latency between every user and the selected server. Furthermore, we introduce inter-domain transit traffic penalty coefficient, to reflect the ISPs' economic profit in the objective function. The following global performance optimization problem describes the goals stated above:

$$\begin{aligned} \texttt{minimize} \quad & \sum_{c\in\mathcal{C}}\sum_{i\in\mathcal{I}} R_{ci}\cdot cost(c,i) \\ \texttt{subject to} \quad & B\cdot P_i \le B_i, \forall i. \end{aligned} \tag{1}$$

where

$$cost(c,i) = \begin{cases} D(c,i) & ISP(c) \texttt{ and } ISP(i) \\ & \texttt{are same or peering} \\ penalty(D(c,i)) & \texttt{others} \end{cases} \tag{2}$$

where $D(c,i)$ denotes the latency from client c to server i, and $penalty(D(c,i))$ is the penalty function for inter-domain transit traffic. Generally we use

$$penalty(D(c,i)) = k\cdot D(c,i), \tag{3}$$

where k is inter-domain transit traffic penalty coefficient [22]. The inter-domain transit traffic penalty coefficient $k = 1$ means the CSP does not want to distinguish inter- and intra-domain traffic, and $k = +\infty$ denotes the CSP forbids inter-domain transit traffic by setting the cost penalty to an infinite value. $\mathcal{C}$ and $\mathcal{I}$ are the set of clients and servers. Moreover, R_{ci}, which denotes the proportion of traffic mapped to server i from client c, satisfies $\sum_{i\in\mathcal{I}} R_{ci} = 1$ and $R_{ci} \ge 0$ for any c. B is the total amount of traffic, a constant parameter that can be calculated by summing the traffic observed by all the servers. B_i is the capacity of server i, and P_i is the proportion of requests directed to server i, so that

$$P_i = \frac{\sum_{c\in\mathcal{C}} R_{ci}}{\sum_{c\in\mathcal{C}}\sum_{i\in\mathcal{I}} R_{ci}} = \frac{\sum_{c\in\mathcal{C}} R_{ci}}{number\_of\_clients}. \tag{4}$$

There are several algorithms and tools to solve this linear programming problem [8], e.g., the simplex algorithm and the criss-cross algorithm. Moreover, the algorithms can be decentralized by enabling each mapping node to perform a smaller-scale local optimization system based on its own view of clients and the aggregated global information collected from other mapping nodes [23].

B. Distance Estimation Module

DEM in a Nutshell. We propose to use NC techniques as the basic infrastructure for network distance estimation. However, several technical challenges arise in building effective and scalable NC systems in cloud server selection scenario. The high dynamics and the constraints on bandwidth/radio coverage (e.g., mobility, unstable links) of "thin" client devices such as smartphones leads to high churn rate [19], which greatly deteriorates the performance of traditional decentralized NC systems. One client cannot have stable neighbor nodes as reference points. Moreover, based on the measurement in mobile IP addresses on 3G networks, the exposed IP addresses of individual mobiles may change frequently, even in a few minutes [5]. Similar to node churn, this is also harmful to the overall prediction accuracy of a decentralized NC system. As a result, current NC systems with flat structures [6], [9], [20], [21] are no longer applicable for positioning the clients in such client-server model.

DEM's design leverages the two features of server selection in cloud computing: (1) The stability and availability of cloud nodes within clouds are relatively high [17]; (2) Only the distance estimation between clients and servers are required, while inter-client distances are not useful, because only the client-server distance matters in client-server model. Fig. 3 demonstrates the architecture of DEM. DEM is a two-layer distance estimation system which includes two different kinds of NC systems: the intra-cloud NC system for positioning cloud nodes and the client-cloud NC system for orienting client nodes. For the intra-cloud NC, stable cloud nodes are self-coordinated using decentralized NC systems (e.g., Vivaldi [9], Phoenix [6]) to obtain accurate servers' coordinates for satisfying the large scale of cloud nodes. Further, these cloud nodes can serve as the landmarks (i.e., reference points) to orient the NC estimation of the clients. Since the clients are not involved in the NC calculation of the cloud servers, the overall estimation accuracy of DEM will not be impacted by the high churn of client nodes. Consequently, network distances between clients and servers

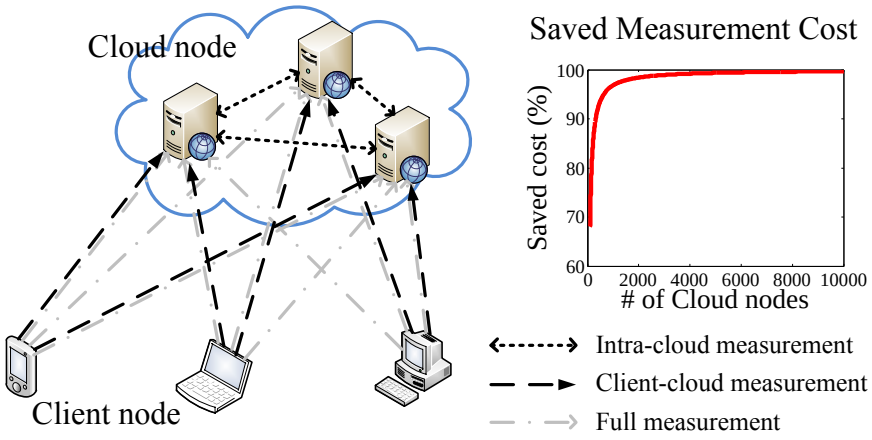


Fig. 3. Distance Estimation Module

are calculated based on these NCs. In this way, DEM reduces each user's measurement cost from $O(N)$ to $O(1)$ with satisfying estimation accuracy, where N denotes the number of servers within the cloud.

Intra-Cloud NC System. DEM proposes to employ the decentralized NC system (DNCS) to determine the coordinates of cloud nodes for DNCS's high estimation accuracy. Due to the stability of cloud nodes [17], cloud nodes can serve as landmarks for typical "thin" clients of cloud services, which cause few node churn to impact the overall estimation accuracy. Serving as landmarks, cloud nodes should have stable network coordinates. Unstable landmark coordinates lead to poor estimation performance due to the subsequent high churn of client's coordinates. To satisfy the accurate estimation and stable coordinate goals, we choose Phoenix [6] as the intra-cloud NC system in DEM for the following two considerations: (1) Phoenix utilizes a matrix factorization model, which gets rid of the triangle inequality violation of the Euclidean space and achieves higher accuracy. (2) The regularization component in Phoenix makes its coordinates stable, which is critical for landmarks. Serving as landmarks, these cloud nodes enlighten the distance estimation between client nodes and cloud nodes. Different from the existing landmark-based NC systems like GNP [21] and IDES [20], DEM gets rid of a centralized algorithm to calculate the NCs of the landmarks, achieving high scalability within the cloud.

Client-Cloud Distance Estimation. Landmark-based NC systems have good performance under the high churn of client nodes (not landmarks), which is critical for client-cloud coordinate calculation because of the mobility and dynamics of "thin" client devices (e.g., smartphones) which is common in the cloud-based networks. Each DEM client randomly selects a subset of the landmarks, and calculates its own NC by referring to the NCs of these landmarks. Rendezvous point (RP) is used to randomly select landmarks from all the cloud nodes in DEM, which ensures the decentralization property. Each end users select different group of landmarks to ensure the overhead balance among all the cloud servers. As in the typical landmark-based matrix-factorization NC system IDES [20], linear least squares are utilized for the NC calculation. In this way, the network distances between all the end users and servers can be obtained and stored to those local Latency Storage databases.

Selective Measurement. The objective of server selection is to direct client nodes to choose their closest or near-closest cloud nodes within the cloud. However, as reported in [18], the closest neighbor loss of NC systems is significantly large (exemplified by Vivaldi [9]). Thus, it is inaccurate to simply use current NC systems to estimate distances between servers and clients.

We integrate *selective measurement* (SM) [26] into DEM to improve the accuracy for closest server selection, i.e., each client performs another K measurements to the top K closest servers selected by previous NC-based algorithm, and then updates the distance information of all the K links using the real distance obtained by selective measurement. For an N-server cloud, we assume that the probability of hitting the actual closest server by selecting the i-th closest server based on the estimated distance is p_i, where $\sum_{i=1}^{N} p_i = 1$. Then, the probability of selecting the actual closest server by simply employing NC-based system is $p = p_1$, and the probability by introducing SM is $p = \sum_{i=1}^{K} p_i$. By introducing SM with a little extra measurement cost, the probability of selecting the actual closest server is greatly improved. We show the effect of SM in Section III.

Measurement Cost Analysis. Measurement cost is an important metric in server selection. High measurement cost makes large bandwidth waste and is impractical for "thin" clients of cloud services. In DEM, the measurement cost for each client is $(L+K)$ `TM` (we define the cost for measuring the distance between two nodes is 1 *time measurement*, `TM` for short),

where L denotes the number of landmarks used in DEM and K is the number of servers for selective measurement. Notice that this measurement cost is a constant number; it will not increase with the growing scale of clients and servers. Comparing with the *full measurement* with (N) TM cost for each end user, DEM's measurement is much cheaper – DEM reduces the cost from $O(N)$ to $O(1)$ compared with the full measurement. The full measurement means measuring the network latency between each end users and servers directly by measurement tools such as King [14], which cost N TM for each end users. Fig. 3 shows the saved measurement cost compared with the full measurement. On the other hand, the measurement cost for each server is (K) TM, which is trivial for cloud servers.

III. Performance Evaluation

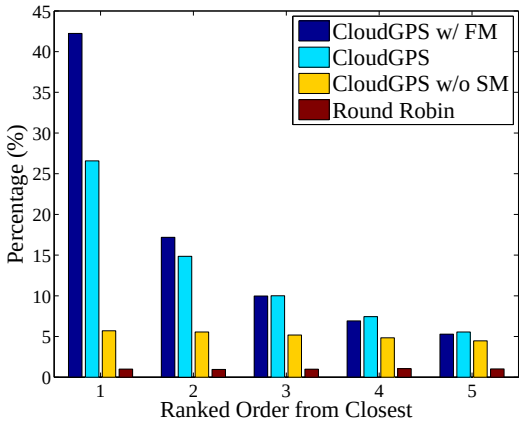
A. Simulation Settings

InetDim dataset [2] is a real-world network latency dataset with autonomous system (AS) topology. It contains all pairs RTTs between 2385 hosts annotated with IP addresses, generated from the raw King [14] measurements made by the Meridian Project [2]. Based on our observation, the proportion of ASes containing only 1 node in InetDim dataset is 70%. One node cannot reflect the internal topology of the corresponding AS, so we choose the nodes belonging to the largest 10 ASes from InetDim dataset, which reflects the inter-domain transit traffic properly. We evaluate the performance of CloudGPS via simulation using the chosen InetDim dataset, which includes the RTTs between 509 nodes. The impact of the inter-domain transit traffic penalty is analyzed in a specific way. We compare CloudGPS with CloudGPS without selective measurement (denoted as CloudGPS without SM), full measurement (FM) version CloudGPS (denoted as CloudGPS with FM), and Round Robin algorithm, to show the performance improved by selective measurement, and latency estimation. Note that when inter-domain transit traffic penalty coefficient $k = 1$, CloudGPS with FM is exactly the DONAR system [23]. We also analyze the results when inter-domain transit traffic penalty coefficient k changes, and show the performance when the number of servers increases. 100 randomly chosen nodes serve as servers and the rest as clients; inter-domain transit traffic penalty coefficient k is set to 2 in our simulation unless explicitly specified else. The other simulation parameters are set as the most popular values [6], [20], [26]: The number of servers for selective measurement is set as $K = 16$, while the number of landmarks used in IDES is set as $L = 16$ and the network coordinate dimension is $D = 10$. According to Section 2, each client's measurement cost is $L + K = 32$ TM, i.e., CloudGPS saves $(1 - 32/100) \times 100\% = 68\%$ measurement cost compared with the full measurement (the cost of full measurement is $N = 100$ TM). We run 100 times for each simulation to mitigate the effect of randomness and report the average result values. It takes 3.05 seconds on average for 409 clients in the 100-server environment in our simulation on MATLAB 2008b, Windows 7 Professional 64bit, Intel(R) Core(TM)2 Duo CPU E8400 @3.00GHz, 4GB RAM.

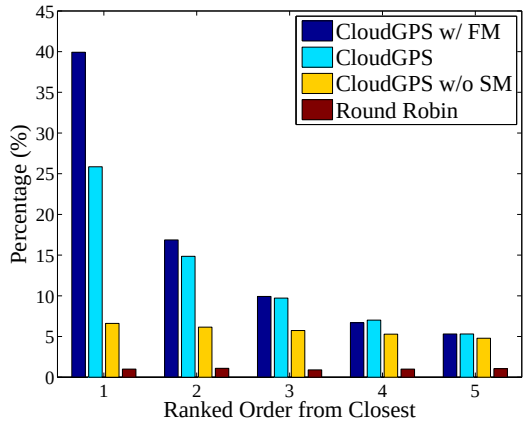
B. Simulation Results

Ranked Order from Closest. We evaluate CloudGPS's performance by comparing inter-domain transit traffic penalty coefficient $k = 2$ and $k = 1$ (without inter-domain transit penalty). We measure *ranked order from closest* (ROFC) [23] as the principal metric to evaluate the performance of the whole selection. ROFC reflects the satisfying degree of user experience. For example, if a client is oriented to the closest server, the selection gives the best user experience. We record the actual ranked order from 1 to 5, i.e., up to the 5th-closest cloud node. With the ranked order, the proportion of selecting the closest server is calculated. Fig. 4 shows that the proportion of selecting closest server (ROFC $= 1$) by CloudGPS with FM, CloudGPS, CloudGPS without SM, and Round Robin is 42.24%, 26.58%, 5.71%, 1.00%, respectively, considering inter-domain transit penalty, and 39.92%, 25.86%, 6.61%, 1.00%, respectively, without considering inter-domain transit penalty. The proportion of choosing the closest server in CloudGPS is changed from 26.58% to 25.86%, with only 0.72 percentages decreased. Comparing with the reduced inter-domain transit traffic (we will show later), it is more than valuable. The reduced proportion from CloudGPS with FM to CloudGPS is also valuable, comparing with the reduced measurement cost, which is 68% in this simulation, and will be much larger when the number of servers increases.

Accuracy in Closest Server Selection. We measure the selection accuracy of CloudGPS in closest server selection (CSS), i.e., the overhead of all the servers are unlimited. Specially, we randomly select a number of nodes from our dataset as the cloud node, and measure the *stretch* of using estimation to select the closest server. The stretch is defined as the distance to the closest cloud server cluster selected based on estimation, divided by the distance to the actual closest server cluster [26]. The result of FM is the exact closest server definitely, so the CSS stretch of CloudGPS with FM without inter-domain transit traffic penalty is always 1. Fig. 5 shows the average stretch for choosing different number of servers (same as reported in [26], the average stretch of each curve increases with the number of servers increasing). We see the average stretch value of CloudGPS is very close to the CloudGPS with FM value and outperforms the other two, and the average stretch value of CloudGPS is very close to the CloudGPS without inter-domain transit traffic penalty value. When the number of servers is 50, the average stretch of CloudGPS is 1.35, much smaller than of CloudGPS without SM (16.04) and round robin (16.03). CloudGPS's stretch for 140 servers is only 2.78, much smaller than CloudGPS without SM (24.61) and round robin (19.97). The average stretch of CloudGPS without inter-domain transit traffic penalty is 1.347, and of CloudGPS is 1.350, respectively; when the number of servers is 40, they are almost the same. From the analysis above, we conclude that comparing with the reduced measurement cost and inter-domain transit traffic, the performance is significant.

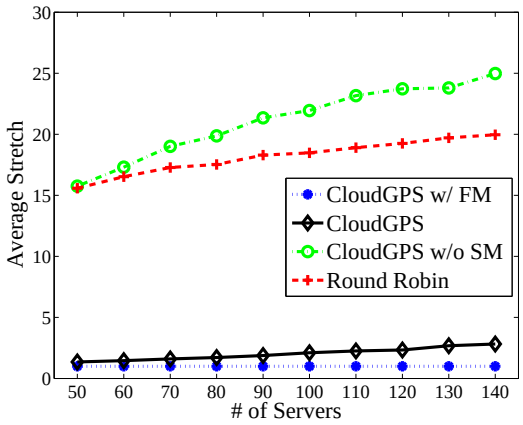


(a) CloudGPS without Inter-domain Transit Penalty

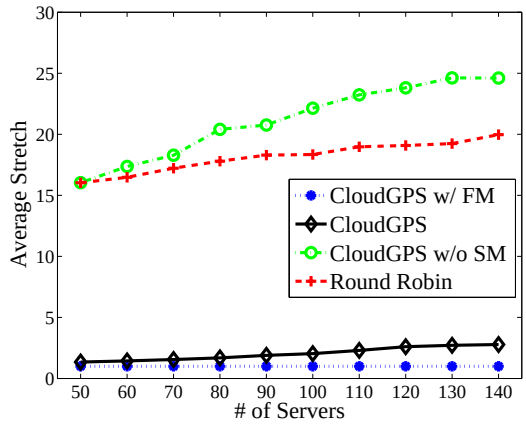


(b) CloudGPS

Fig. 4. Ranked Order from Closest



(a) CloudGPS without Inter-domain Transit Penalty

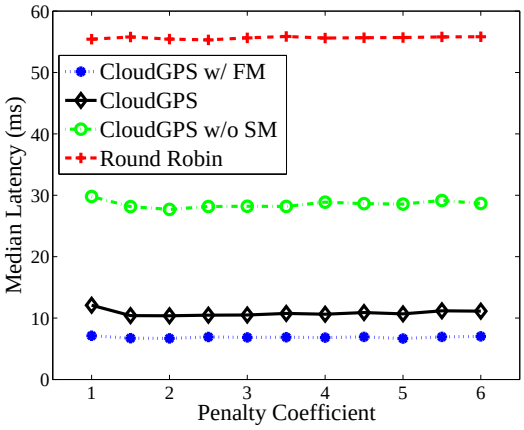


(b) CloudGPS

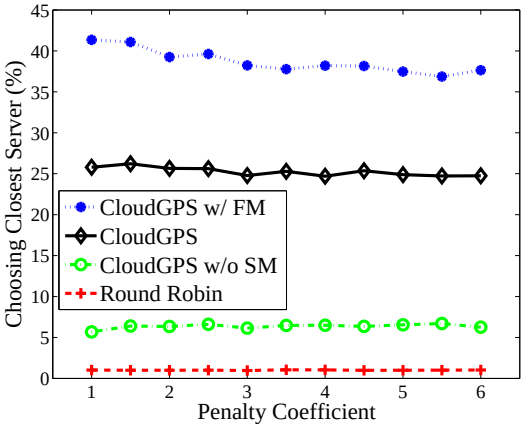
Fig. 5. Accuracy in CSS

Impact of Inter-domain Transit Penalty Coefficient. We measure the proportion of choosing closest server, inter-domain transit traffic, and median latency, when the penalty coefficient varies. With a randomly chosen 100 nodes serving as cloud servers, we change the penalty coefficient from 1 to 6 with the step of 0.5. Inter-domain transit traffic penalty coefficient $k = 1$ means the CSP does not consider inter-domain transit traffic penalty, and a large value means the CSP strongly prefer to choose a server in the same ISP. Fig. 6(a) shows the median latency is not influenced by the value of inter-domain transit traffic penalty coefficient. Fig. 6(b) shows that the proportion of choosing closest server decreases a little when the inter-domain transit traffic penalty coefficient increases. From Fig. 6(c), we find that the inter-domain transit traffic decreases with the value of inter-domain transit traffic penalty coefficient increasing, and the absolute slope is decreasing. In real application, the penalty coefficient k should be set based on the balance between user-server proximity and inter-domain transit traffic cost, because different applications' sensitiveness of proximity and inter-domain transit traffic cost varies a lot. With the large amount of saved measurement cost and reduced inter-domain transit traffic, CloudGPS performs very close to CloudGPS with FM, and outperforms DONAR (CloudGPS with FM with inter-domain transit traffic penalty coefficient $k = 1$).

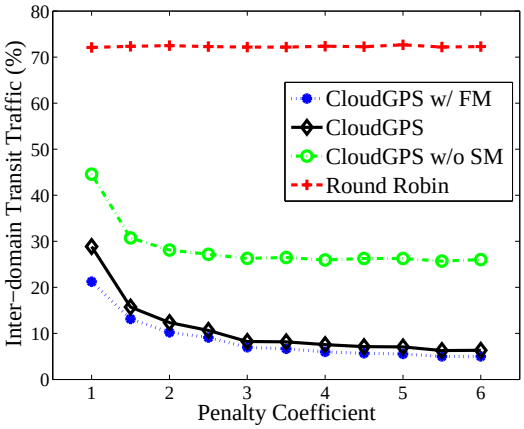
Scalability of Mounting Servers. We measure the proportion of choosing closest server, inter-domain transit traffic, and median latency, when the number of servers increases. Same as the paragraphs above, we randomly select a number of nodes from our dataset as the cloud node, with inter-domain transit traffic penalty coefficient $k = 2$. Fig. 7(a) shows the median latency decreases with the increasing number of server in a fixed number of ISPs, which means deploying more servers improves the user experiences. To the same, the proportion of choosing closest server and inter-domain transit traffic decreases when the number of servers increases, in Fig. 7(b) and Fig. 7(c). When the number of servers increases, the measurement overhead of each client is the same, which shows the scalability in terms of the increasing number of servers.



(a) Median Latency

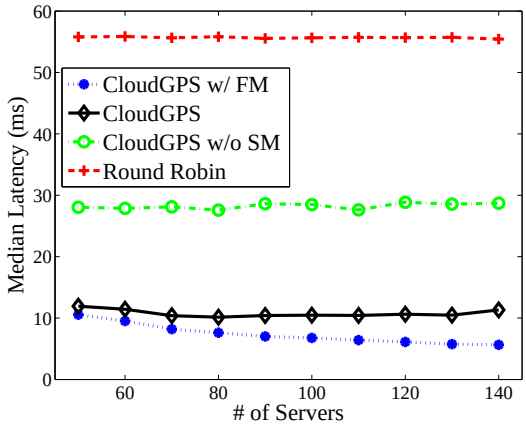


(b) Closest Server

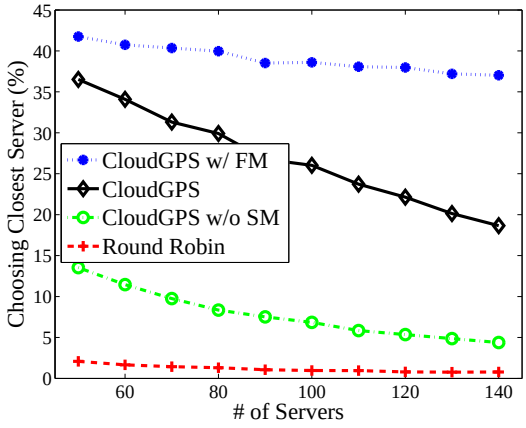


(c) Inter-domain Transit Traffic

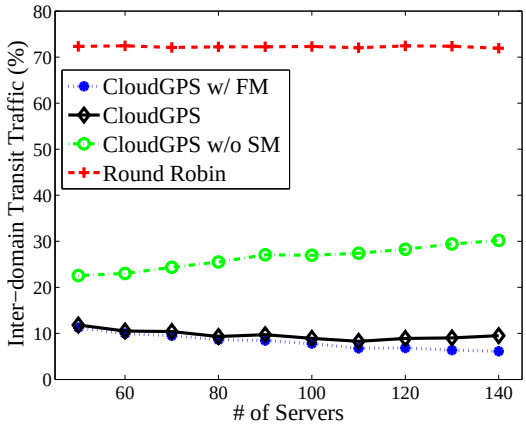
Fig. 6. Impact of Inter-domain Transit Penalty Coefficient



(a) Median Latency



(b) Closest Server



(c) Inter-domain Transit Traffic

Fig. 7. Scalability of Mounting Servers

C. Evaluation Summary

From the simulation results, we have the following observations:

1) CloudGPS performs a little worse[1] than CloudGPS with FM, but saves a large amount of measurement cost.
2) CloudGPS performs much better than CloudGPS without SM, with similar measurement cost.
3) The proportion of inter-domain transit traffic decreases when the inter-domain transit penalty coefficient K increases, but when the coefficient K increases to larger than 3, the decreasing speed becomes very slow.
4) CloudGPS is scalable with a constant number of mea-

[1]Worse and better here are in terms of the proportion of choosing the closest cloud node, compared to the decision with exact full global network and server knowledge.

surement cost for each client when the number of servers increases.

To summary, CloudGPS reaches the scalability and ISP-friendliness goals, as well as carries on the features in existing server selection systems.

IV. Conclusion and Future Work

In this paper, we propose CloudGPS, a novel server selection scheme for cloud-based applications. CloudGPS is scalable to deal with the explosively-increasing numbers of clouds as well as user clients based on its DEM component. DEM takes the advantages of two different kinds of NC techniques (i.e., the distributed NC and landmark-based NC) to position both the clouds and users, achieving the measurement cost reduction from $O(N)$ to $O(1)$ for clouds with N server clusters. In addition, CloudGPS is ISP-friendly that effectively reduces inter-domain transit traffic leading to low ISP operational costs and improve end users' quality of service based on its MM component which makes a balance between the closest server selection and the inter-domain transit traffic, in the limitation of servers' capacity.

Our future work includes the following two aspects. First, we expect to release a complete application program interface (API) for quick and convenient configurations. Second, we would like to deploy CloudGPS on a real-world cloud computing environment to demonstrate its feasibility and efficiency.

Acknowledgements

We are grateful to Zhuo Chen and Long Jin from Tsinghua University for helpful discussions. We also thank anonymous reviewers for their valuable comments and suggestions.

References

[1] The routeviews project. http://www.routeviews.org/.

[2] B. Abrahao and R. Kleinberg. On the internet delay space dimensionality. In *ACM SIGCOMM IMC*, 2008.

[3] V.K. Adhikari, S. Jain, and Z.L. Zhang. Where do you tube? uncovering youtube server selection strategy. In *IEEE ICCCN*, 2011.

[4] S. Agarwal and JR Lorch. Matchmaking for online games and other latency-sensitive p2p systems. In *ACM SIGCOMM*, 2009.

[5] M. Balakrishnan, I. Mohomed, and V. Ramasubramanian. Where's that phone?: geolocating ip addresses on 3g networks. In *ACM SIGCOMM IMC*, 2009.

[6] Y. Chen, X. Wang, C. Shi, E. Lua, X. Fu, B. Deng, and X. Li. Phoenix: A weight-based network coordinate system using matrix factorization. *IEEE Transactions on Network and Service Management*, 8(4):334–347, 2011.

[7] D.R. Choffnes and F.E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *ACM SIGCOMM*, 2008.

[8] T.H. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[9] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM*, 2004.

[10] A. Dhamdhere and C. Dovrolis. The internet is flat: Modeling the transition from a transit hierarchy to a peering mesh. In *ACM CoNEXT*, 2010.

[11] P.T. Eugster, R. Guerraoui, A.M. Kermarrec, and L. Massoulié. From epidemics to distributed computing. *IEEE Computer*, 37(5):60–67, 2004.

[12] A.J. Ganesh, A.M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2):139–149, 2003.

[13] A. Greenberg, J. Hamilton, D.A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM CCR*, 39(1):68–73, 2008.

[14] K.P. Gummadi, S. Saroiu, and S.D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *ACM SIGCOMM IMW*, 2002.

[15] C. Huang, A. Wang, J. Li, and K.W. Ross. Measuring and evaluating large-scale cdns. In *ACM SIGCOMM IMC*, 2008.

[16] A.M. Kermarrec, L. Massoulié, and A.J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3):248–258, 2003.

[17] A. Li, X. Yang, S. Kandula, and M. Zhang. CloudCmp: comparing public cloud providers. In *ACM SIGCOMM IMC*, 2010.

[18] E.K. Lua, T. Griffin, M. Pias, H. Zheng, and J. Crowcroft. On the accuracy of embeddings for internet coordinate systems. In *ACM SIGCOMM IMC*, 2005.

[19] J. Manweiler, S. Agarwal, M. Zhang, R. Roy Choudhury, and P. Bahl. Switchboard: a matchmaking system for multiplayer mobile games. In *ACM Mobisys*, 2011.

[20] Y. Mao, L.K. Saul, and J.M. Smith. IDES: An internet distance estimation service for large networks. *IEEE Journal on Selected Areas in Communications*, 24(12):2273–2284, 2006.

[21] T.S.E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *IEEE INFOCOM*, 2002.

[22] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, and V.V. Vazirani. How many tiers? pricing in the internet transit market. In *ACM SIGCOMM*, 2011.

[23] P. Wendell, J.W. Jiang, M.J. Freedman, and J. Rexford. DONAR: decentralized server selection for cloud services. In *ACM SIGCOMM*, 2010.

[24] H. Xie, Y.R. Yang, A. Krishnamurthy, Y.G. Liu, and A. Silberschatz. P4p: provider portal for applications. In *ACM SIGCOMM*, 2008.

[25] B. Zhang, R. Liu, D. Massey, and L. Zhang. Collecting the internet as-level topology. *ACM SIGCOMM CCR*, 35(1):53–61, 2005.

[26] R. Zhang, C. Tang, Y.C. Hu, S. Fahmy, and X. Lin. Impact of the inaccuracy of distance prediction algorithms on internet applications-an analytical and comparative study. In *IEEE INFOCOM*, 2006.