2-2011

Consideration of Service Time in Placing Clients of Web-Based Services

Craig E. Willis
Worcester Polytechnic Institute, cew@cs.wpi.edu

Wei Zhang
Worcester Polytechnic Institute

WPI-CS-TR-11-02 February 2011

Consideration of Service Time in Placing Clients of Web-Based Services

by

Wei Zhang
Craig E. Wills

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

Abstract

Web-based services involving dynamic computation of the content are increasingly used on the Web. This computation may not only involve processing, but often involves access to other back-end services typically for database access. These multi-tiered Web-based services have different characteristics than traditional Web content and new models need to be considered not only how to manage them, but also how the DNS mechanism should map clients to the appropriate front-end server.

In this work we study the potential of considering back-end service times in the decision of mapping clients to servers. We do so by gathering data from deployed platforms in the Internet today from client locations both scattered around the globe and around the United States. We use these data for the simulation of straightforward policies that account for back-end service time. Our results show that in the best case our simple policies have better performance than using current DNS decisions, while in the worst case they provide comparable performance for a range of performance metrics.

1 Introduction

Web-based services involving dynamic computation of the content are increasingly used on the Web. This computation may not only involve processing, but often involves access to other back-end services typically for database access. These services may be in the same data center as the client-facing front-end server or may be located in a remote location. These multi-tiered Web-based services have different characteristics than traditional Web content and new models need to be considered not only how to manage them, but also how to map clients to the appropriate front-end server.

The traditional approach for mapping clients to one of multiple server locations for content is to simply map a client to a nearby data center when that client makes a DNS request for the given server. This is the core of the approach used by content distribution networks and has worked well for static content where the retrieval costs are largely dependent on transporting data over a TCP connection. Shorter round-trip times result in quicker ACKs and shorter response times.

However, Web-based services introduce a potentially significant component that we conjecture needs to be included in the decision of which front-end server to use for a client request. If a nearby front-end server incurs significant back-end costs to perform a service then the best front-end server for a client of that service may not be the closest and could even be one much further away. The client-to-server mapping decision made by DNS should not only consider the client and server location, but also account for the nature of the service itself. This consideration is not possible if a Web site offers all services under the same server name, such as `www.company.com`, but increasingly we see sites "expose" Web services, such as `search.company.com` allowing the desired service to be considered as part of the DNS decision.

In this work we study the potential of considering back-end service times in the decision of mapping clients to servers. We do so by gathering data from deployed platforms in the Internet today from client locations both scattered around the globe and around the United States. We use these data for the simulation of straightforward policies that account for back-end service time. Our results show that in the best case our simple policies have better performance than using current DNS decisions, while in the worst case they provide comparable performance for a range of performance metrics.

In the remainder of this paper we further motivate the problem and our approach in Section 2. We go on to define performance metrics for evaluating approaches in Section 3. We describe the study we performed in Section 4 with the results obtained from a set of clients in the United States in Section 5. We define a simple parameterized set of policies for consideration of service time in Section 6 and use our gathered data to evaluate the performance of these policies for the U.S. clients in Section 7 and a set of global clients in Section 8. We describe related work in Section 9 and conclude with a summary and future work in Section 10.

2 Motivation

Modern Internet platforms deliver computing resources to Web-based services across one or more data centers. An important decision is which data center should handle the request of a client for such a service. This decision is traditionally made at the DNS level by mapping the server name to an appropriate IP address, although we observe that some sites expose the Web-based service name as part of the server name to aid in this mapping. For example, rather than giving a search service the URL of `www.company.com/search`, the service is exposed to DNS with the URL of `search.company.com/`. This exposure allows the DNS mechanism to make a decision on where to map a client based upon the Web-based service.

Even if the nature of the Web-based service is exposed, there are still many factors involved in the decision of which data center to choose for a particular client. For example, an application can simply make the decision based on the geographic location of the client, or it may employ load balancing techniques to direct a request to the least loaded servers. Rather than focus on these up-front factors, our work is motivated by also looking at the nature of the service itself, specifically at the back-end processing costs in making client placement decisions.

As the online social networking, video content hosting, and other Web services become increasingly popular, these services are more complex as they have more back-end dependencies on other services such as database queries. When service providers deploy their services, these dependent services may not be at the same location as the front-end client-facing servers. Hence, fulfillment of a Web service may require remote database queries to performed. In addition, the resource granted to each service may not be balanced and overloaded back-end servers may have an impact on the performance of front-end application servers. As a result, different front-end application servers may have different service times due to the configuration and current load of their dependent set of back-end servers.

This observation motivates our work to examine the back-end processing costs of current Web services and seek to understand if these costs can and should be considered in the mapping of clients to front-end servers for these services. If the processing cost of the back-end server is significant then it should be considered when making a decision to choose the best front-end server for a client.

While consideration of back-end processing costs may be worthwhile, an important question is how to study this issue without direct access to deployed Web-service platforms. Rather than seek such access our approach is to use a simple observation about a typical HTTP network flow as shown in Figure 1 to estimate the back-end service time for any transaction.

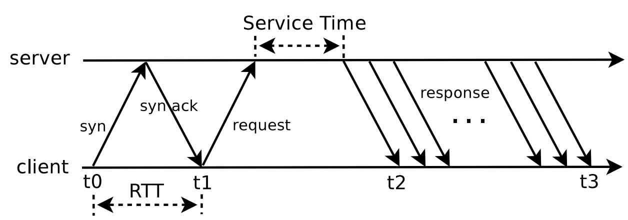


Figure 1: Network Flow of Typical HTTP Transaction

As shown in the figure, $t_1 - t_0$ is the time to set up the TCP connection and is a typical estimate of the round-trip time (RTT) between client and server. An immediate HTTP GET request is then sent from the client at t_1 with t_2 the time in which the first byte of the HTTP response is returned to the client. Data continues to flow to the client (corresponding TCP ACKs from client to server are not shown) until t_3 when the last byte of the response is received. The time $t_2 - t_1$ contains both a RTT and the time taken at the server before it generates any data to return to the client. The server could begin sending data (such as HTTP response headers) before all servicing is done, but in preliminary testing we found this situation does not occur frequently and when observed the HTTP data arrives within 10ms of the HTTP headers so we did not try to distinguish these cases in subsequent testing. Rather, we consider $(t_2 - t_1) - RTT$ as an estimate of the service processing time. That is,

$$T_{rtt} = t_1 - t_0$$
$$T_{service} = (t_2 - t_1) - T_{rtt}$$

While this approach does not allow us to distinguish *why* there is a service processing delay, it does allow us to estimate its magnitude. The delay could be caused by some combination of front-end service delays, back-end service delays or access to remote back-end services. Obviously it is possible that if the subsequent HTTP GET request is delayed in the network then the estimate value of $T_{service}$ will incorrectly assume the delay is attributed to service time rather than network delay. However if we make multiple requests for a Web service and the requests consistently yield significant estimated service times then we can safely assume that the delays are indeed caused by servicing.

As a starting point to understand the magnitude of the service time, we used a simple client running on our campus to retrieve the home page of 1300 popular Web sites that have been used in previous work [5]. The home pages of popular sites are increasingly dynamic, where the content is built for each request. Figure 2 shows a CDF of the estimated service times for these popular "home page services". As shown, half of the sites have median service times under 100 milliseconds. However, 10% of them have median service times of several hundred milliseconds and 95%-tile service times over a second.

Considering that the RTT from a client to front-end server is typically less than 100 milliseconds within a country and a couple of hundred milliseconds across oceans, these preliminary results indicate that the service time can be an important component in the performance of Web-based services. We examine the impact of this component in our study.

3 Performance Metrics

Before looking to measure Web-based service performance, it is important to consider what is meant by performance. The traditional measure of performance for a Web object is the time from when a user requests an object until that object is received in its entirety. Referencing Figure 1 this is the value

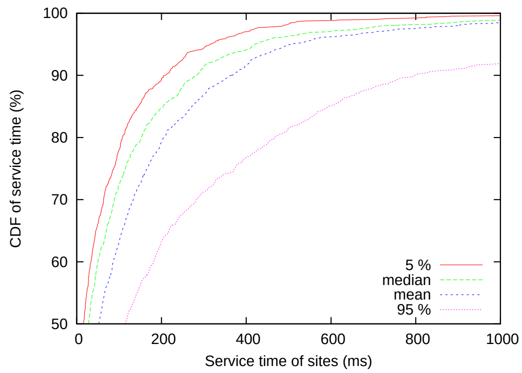


Figure 2: CDF of Service Times of Home Pages for 1300+ Popular Web Sites

of $t_3 - t_0$, which includes TCP connection setup as well as the time to make the HTTP request and receive all of the data. However modern browsers begin to process data as they are received for purposes of executing embedded JavaScript, retrieving embedded objects and rendering the contents. This "as data are available" approach argues that it also important to consider how long it takes for the *first* byte of data to arrive ($t_2 - t_0$). These two basic metrics indicate when all data from the service are received and when the client has data to begin work.

Given the typical settings of most browsers to support persistent TCP connections, we can also extend each basic metric to consider persistent connections. With persistent connections, the client does not have to re-establish the connection resulting in more efficiency. This extension leads to four possible performance metrics to consider:

$$T_{total} = t_3 - t_0$$
$$T_{total-pers} = t_3 - t_1$$
$$T_{firstbyte} = t_2 - t_0$$
$$T_{firstbyte-pers} = t_2 - t_1$$

In our work, we compute all four metrics, but focus primarily on $T_{firstbyte}$ as that represents the time that a client can begin processing data and is a metric that is independent of the amount of data returned by a service. We also show some results for T_{total} as appropriate. While service time is an even bigger contributing component of requests made over persistent connections we assume that Web-based service requests will primarily be made as the first request to a front-end application and thus will likely incur the cost to create a TCP connection. We also focus only on the time for the invocation of the Web-based service and not on any subsequent retrieval of embedded objects as those are often done in parallel with retrieval and parsing of the base content.

While our work evaluates the *quality* of the DNS decision we do not explicitly account for the cost of this lookup in the work. In a small number (1%) of cases in our study we also observe the use of HTTP redirection to direct a client to an alternate server. Whenever a redirection occurs, we measure the time for the redirected server so that t_0 is reset to the time the client initializes a connection to this server.

4 Study

To study this problem we first identified a set of popular Web sites providing a range of services as candidates for testing from a couple of platforms of clients. We then performed an initial set of tests to narrow this candidate set to a smaller number. Once this set is determined we describe the methodology used in studying the services provided by these sites and how we analyzed the collected data.

4.1 Candidate Web-Based Services for Study

In selecting the set of services to study, we wanted to focus on a relatively small set of popular Web sites that supported for more than one service for study. Using Alexa [1], we choose 20 popular Web sites as candidate service providers to study. For each site, we choose four types of services to represent the whole site, namely, *Home Page*, *Image*, *Search*, and *Video Page*. In total, we initially examined 80 Web-based services. We perform an initial examination on these candidate set of services and then focus our study on a smaller set of sites.

The *Home Page* service is the front page of a Web site, which typically has dynamic content. Based on our preliminary results, we expect some amount of service time is required for this type of service.

We choose a relatively small image (typically on the home page) for each site as the *Image* service. While we expect these static images to be cached on the servers and require minimal service time, we include this "service" to verify the expected results.

A *Search* service is generally involved with sending queries that likely require contacting a back-end server, particularly if the query is not cached by a front-end server. Given that the back-end server could be accessed remotely this service may involve a large service time.

The last service we include is for a *Video Page*. While this service is not directly serving video content its content is dynamic in generating the current set of popular videos.

4.2 Client Sets

We used two sets of clients to study the performance of Web-based services on two different scales. In all cases we selected clients for their geographic location from the set of PlanetLab [9] nodes available to use. When multiple PlanetLab nodes were available in a geographic area we selected specific nodes based upon low load and the fewest number of active slices.

The *global* client set consists of 10 PlanetLab nodes located in different continents of the world. Two of them are in USA (USW(est) and USE(ast)), three of them are in Europe (UK, FR and DE), another two are in Asia (KR and JP) and the others are located in Africa (EG), Australia (AU), and South America (BR) respectively. We choose this set to represent a geographic distribution around the world.

The *U.S.* clients set consists of 8 PlanetLab nodes geographically dispersed around the United States. These clients are located in the states WA, CA, CO, TX, IL, FL, MD and MA. We choose this set to understand performance variation of services in a particular region of the world in which clients for many of the 20 Web sites are concentrated.

4.3 Web-Based Services for Study

Given this set of clients and candidate services, we wanted to focus our study on a smaller set of services with two properties. First, we wanted services for which the service time is non-trivial and exhibits some amount of variance indicating that the service time may be a consideration in the overall performance for that service. Second, we wanted services where the size of the "platform", measured in terms of the number of unique IP addresses offering this service, provided multiple opportunities for selection by the DNS mechanism of the service.

To determine a set of services with these properties we determined a URL to retrieve for each of our 80 candidate services. For services involving a search string me made sure to use a different set of search terms on each retrieval. We then performed an initial measurement study from each of the clients in our global set. The study consisted of 100 retrievals from each client spaced at intervals of 10 minutes where on each retrieval a DNS lookup was made (with all returned IP address saved) and the URL contents retrieved using the Perl LWP library.

For the first property, we calculated the service time $T_{service}$ for each of the 80 services in our candidate set for each retrieval from all 10 global clients. Figure 3 shows the rank order all 80 candidate services.

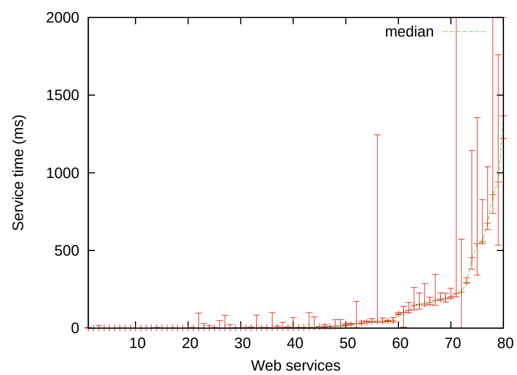


Figure 3: Rank Order of Median Service Time for Each Candidate Service (with 5%/95% Error Bars)

As shown, most of the services have negligible service times—many of these are the Image service. However, over 10% of them have median service times over 200 milliseconds, and several of them have medians over half second. In addition, many services with larger median service time performance also exhibit much variation as shown by the 5% and 95% error bars in the figure.

For the second property, we used these preliminary results to collect the list of *all* IP addresses returned to all ten clients. In some cases multiple

IP addresses were returned for a single DNS lookup. In some cases the same IP address(es) were returned to different clients. We did not try to distinguish if multiple IP addresses were associated with the same data center or represented multiple data centers. We merged all IP addresses for all clients of services for each of the 20 sites in our candidate set to determine the "platform" of addresses for each site. We found the size of the platform for the 20 Web sites ranged from 1 to 200 IP addresses for a single site.

In considering the two desired properties for selecting services of sites to study we focused on sites with a larger number of IP addresses and with larger and more variable service times in Figure 3. Based on this consideration we selected the four services of the four sites `foxnews.com`, `msn.com`, `yahoo.com` and `cnbc.com` for focused study.

Details about these 16 services are shown in Table 1 where we show the server name providing each service, the authoritative time-to-live (ATTL) for DNS lookups of this server name as well as the number of distinct IP addresses found by our set of global and U.S. clients (we performed similar work to obtain this set for these clients). Note that some of services share the same domain names (e.g. `www.foxnews.com`), which means the authoritative DNS is unable to make separate decisions for mapping clients of different services.

4.4 Methodology

Once we narrowed our study to these 16 services, we set up another round of data collection where each client in our respective client sets was used to gather data about each service. Each client ran a script every 10 minutes where it invoked each of the 16 Web-based services (by retrieving the given URL) from *each* of the IP addresses known to the respective global- and U.S.-based client sets. We also did a DNS lookup of the corresponding server name for each service to know which IP address would be selected by the DNS mechanism at the time. These lookups and retrievals were made using a modified Perl LWP library to not only be able to send requests to the DNS returned IP but also to send requests to all IP addresses in our known set. Each group of requests was repeated 100 times over a 16-hour period. We dropped requests that took more than 10 seconds, which occurred for 0.03% of retrievals for the U.S. clients (and 6.5%, primarily from KR and AU, for the global clients). We repeated this methodology multiple times over a one-

Table 1: Site Servers, Services and Server Information

Server Name	Service	ATTL (sec)	Global #IP	U.S. #IP
www.foxnews.com	Home Image Search	20	30	32
video.foxnews.com	Video	20	34	49
www.msn.com	Home	300	3	3
col.stb.s-msn.com	Image	300	201	101
www.bing.com	Search Video	20	32	81
www.yahoo.com	Home	60	7	5
l.yimg.com	Image	300	29	24
search.yahoo.com	Search	1800	12	3
video.yahoo.com	Video	300	2	2
www.cnbc.com	Home Video	3600	3	3
media.cnbc.com	Image	20	30	26
search.cnbc.com	Search	600	1	1

month period for our two client sets with the results shown from last round of data gathering.

4.5 Analyzing Results

In analyzing the gathered data we distinguish the results from three particular server IP addresses within each group of results for a service. These three server IP addresses are S_{dns}, $S_{best-rtt}$, and $S_{best-service}$.

The server identified by S_{dns} is the address chosen by the DNS mechanism during this group of requests. It represents the case that the client always uses the DNS-returned server—the expected behavior for an actual client of this service from the given client location. Note that the ATTL for most of the servers in Table 1 is less than our 10min (600 sec.) retrieval interval so an authoritative DNS lookup will be performed for these servers on each retrieval in the absence of other user activity.

The server identified by $S_{best-rtt}$ is the address of the server machine with the lowest RTT (T_{rtt}) for a group of requests and represents the 'closest' server. While this server cannot be determined ahead of time, it represents a benchmark for the importance of RTT when considering the best performing server.

Similarly, the value of $S_{best-service}$ is the address of the server machine with the lowest service time ($T_{service}$) for a group of requests and represents the 'fastest' server once it receives the request. Again this value cannot be determined ahead of time, but it represents a benchmark for the importance of service time when considering the best performing server.

The goal in comparing the performance of the server returned by the DNS mechanism S_{dns} with the server providing the best RTT $S_{best-rtt}$ and the server providing the best service time $S_{best-service}$ is to understand how well the current DNS mechanism is doing in making decisions relative to best-case uses of RTT and service time. The best performance is also dependent on the performance metric from Section 3 that is used. If better performance is obtained via one of these best-case approaches on a consistent basis then that result points that the current DNS mechanism for placing clients of these Web-based services could be improved.

In comparing the performance of these approaches for selecting a server relative to a particular performance metric we choose not to show performance directly, but rather show the difference of between the performance of an approach and the best performance observed for that metric within a group of requests.

That is, for each group of requests, and for each metric, we find out the server with the best performance. We identify the server providing the best performance for each of the four metrics denoted as $S_{best-firstbyte}$, $S_{best-total-pers}$, $S_{best-firstbyte}$, and $S_{best-firstbyte-pers}$. In showing the results we compute the time differences between the best server for a metric and the servers identified by each of the three approaches using $\Delta_{metric}(approach)$ notation. For example,

$$\Delta_{firstbyte}(S_{dns}) = T_{firstbyte}(S_{dns}) - T_{firstbyte}(S_{best-firstbyte})$$

$\Delta_{firstbyte}(S_{dns})$ shows the difference between the *first byte time* of the server S_{dns} and the best server $S_{best-firstbyte}$. If it is 0, it implies that the DNS-returned server is indeed the best server ($S_{dns} \equiv S_{best-firstbyte}$) in terms of this performance metric. Using the best server performance for a metric allows us to focus on the relative differences between the performance of various servers because having a situation where all servers perform well or all perform poorly for a client is not a server selection problem.

5 Base Results for U.S. Client Set

We initially focus our attention on the results from the U.S. client set. Analysis of the results obtained from the experiments of the 16 services described in the previous section validated our expectation that the four Image services all demonstrated negligible services times and the DNS-identified server was often the same as the server with the best RTT. For the 'CNBC Search' service we observe only one server IP in our test so there is no notion of "selection". The 'FoxNews Search' service both yielded large service times for the first server accessed by a client as well as what appeared to be back-end caching effects that resulted in inconsistent results. As a result we dropped this service as well as the other Search and four Image services and concentrated our analysis on the remaining ten services.

Figure 4 shows the base performance results comparing DNS performance with best RTT and best service time for all tests for all clients of the ten remaining services using the first byte performance metric. As described in Section 4.5, a difference of zero indicates that the given approach yields the best performance time for a client from amongst the set of all possible servers and shows that all approaches attain this result in close to 50% of the tests.

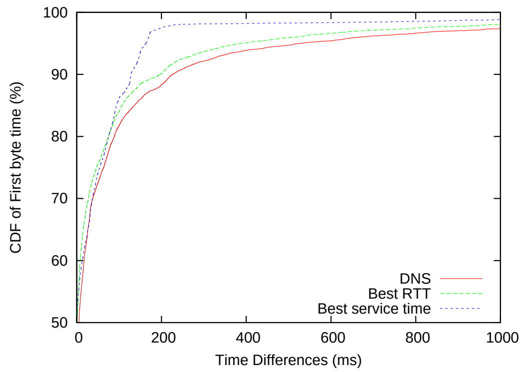


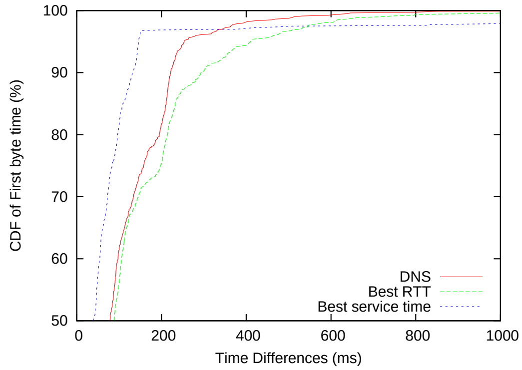
Figure 4: Base Performance for All Services for All U.S. Clients (First Byte Time)

In general sense, at most of the time, most of them are doing reasonable jobs using the DNS returned server. However, at the 85%-tile there begins to be a noticeable performance difference between the DNS-returned server and the best possible server for this metric. The server providing the best RTT closely tracks the DNS-returned server performance, but about the 85%-tile the servers providing the best service time yield noticable better first byte time performance. This difference does not necessarily mean that a selection policy that takes into account service time will yield better decision as this is a best-case situation, but it does indicate that consideration of such an approach is warranted.

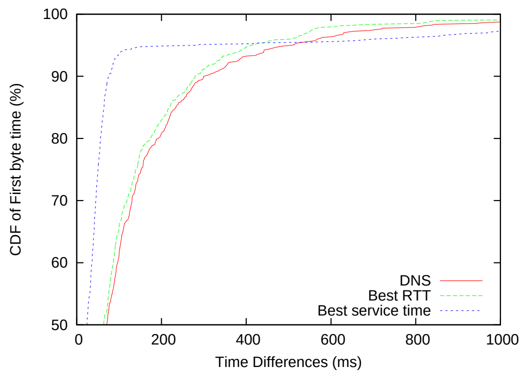
Figure 5 shows similar CDFs for three specific services across all test locations where Figure 5(a) is for the 'FoxNews Home Page' service, Figure 5(b) is for the 'MSN Video Page' service and Figure 5(c) is for the 'Yahoo Search' service.

In the first two graphs of the figures the server with best service time consistently provides the best first byte performance results while in the third figure the DNS and best RTT servers perform similarly and consistently better than best service time server.

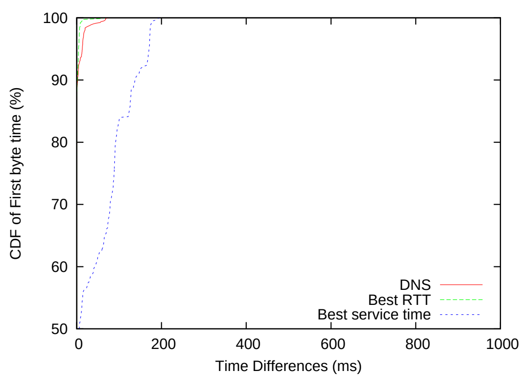
These results show there is not a universally good single metric in terms of choosing the best server. However, these results do indicate that consider-



(a) FOXNews Home Page



(b) MSN Video Page



(c) Yahoo Search

Figure 5: Base Performance for Specific Services for All U.S. Clients (First Byte Time)

ation of *both* RTT and service time in making a server selection for a client may result in a better decision. We examine this approach in the following.

6 Consideration of Service Time in Server Selection

In Section 5 we found that clients for some services are not being directed to their best servers, that is the DNS decision making could be improved. We also found that in some, but not all, cases consideration of the service time could improve the decision making. The question is how to implement a policy that can consider the service time without ignoring the RTT between client and server, which is still an important component in the decision.

Rather than use a single approach for combining these two values, we define a simple parameterized function $T(N)$, which accounts for T_{rtt} and $T_{service}$ with an integer parameter $N \geq 0$. Specifically the function is defined as

$$T(N) = T_{service} + N * T_{rtt}$$

where the function includes the service time plus an integer multiplier on the RTT. As written this function assumes the service time and RTT are already known. Instead we define $ExpT'(N)$ as the next expected $T'(N)$ using an exponential weighted average to compute $ExpT(N)$ over time. We choose to use a value of $\alpha = 0.7$ in our work (we also tried $\alpha = 0.5$ with little difference) so after each retrieval we update the expected value as:

$$ExpT'(N) = 0.7 * ExpT(N) + 0.3 * T(N)$$

Finally, we define the policy P_N as the one that always chooses the server with the minimum expected $T'(N)$, i.e.,

$$P_N = ChooseMin(ExpT'(N))$$

Intuitively, policy $P_{N=0}$ always choose the server with the lowest 'Expected Service Time', while $P_{N=\infty}$ selects the server with the lowest 'Expected RTT'. In general, a policy with a smaller N favors 'Service Time' over 'RTT', while a policy with a larger N favors servers with a lower 'RTT'. In our work we specifically examine policies for $N = 0, 1, 2$ and ∞.

7 Simulation of Policies for U.S. Client Set

We evaluated the use of varying combinations of expected RTT and service time for different Web-based services using the collected data via trace-driven simulation. With 100 retrievals for each service, we used the first 10 retrievals to initially compute expected RTT and service times and then used the remaining 90 retrievals to both evaluate each policy as well as continue to update the expected value.

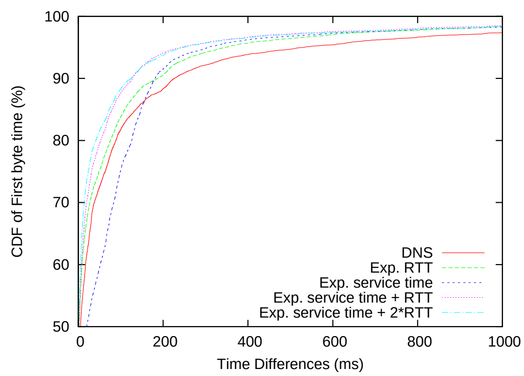
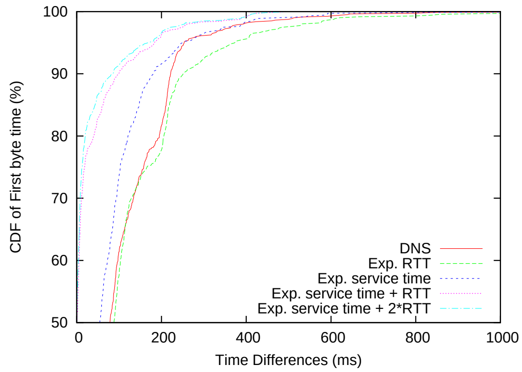


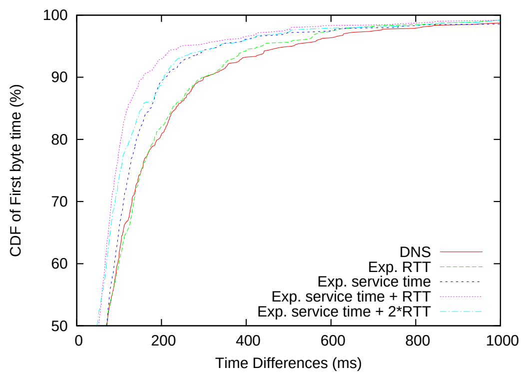
Figure 6: Simulation of Policies for All Services for All U.S. Clients (First Byte Time)

Figure 6 shows the simulation results for the chosen 10 services. The results show that consideration of only the expected service time ($P_{N=0}$) results in the worst performance over a range of the CDF while use of only the expected RTT ($P_{N=\infty}$) performs comparable or a bit better than the DNS decision. However the $P_{N=1}$ and $P_{N=2}$ policies consistently perform as well or better than all of the other policies. This result is particularly encouraging as it occurs with a simple policy on a deployed production platform.

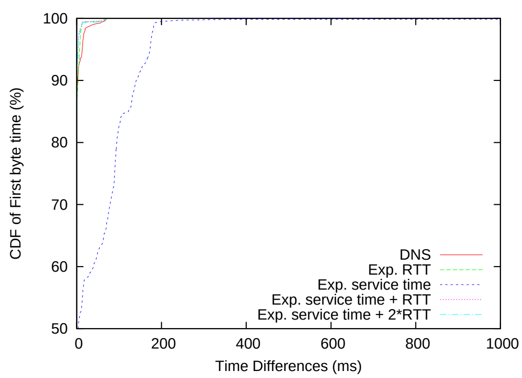
Figure 7 shows a subset of the results from Figure 6 for the three specific services shown in Figure 5. These simulation results show that a combination of expected RTT and service time (with $N = 1$ or $N = 2$) provides the best overall results for the first two services and comparable performance to DNS for 'Yahoo Search'.



(a) FoxNews Home Page



(b) MSN Video Page



(c) Yahoo Search

Figure 7: Simulation of Policies for Specific Services for All U.S. Clients (First Byte Time)

Figure 8 provides another view of simulation results for the Fox News Home Page Service by showing the 90%-tile results on a per-client basis where the clients are shown in a west to east order. These results show that there is variation in performance amongst the policies even for different client locations of the same service, although the $P_{N=2}$ generally performs the best or close to the best for each client.

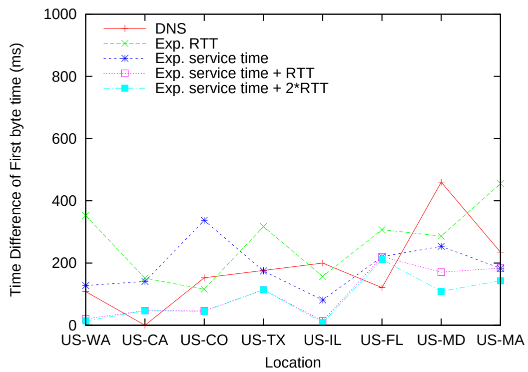


Figure 8: Simulation of Policies for FoxNews Home Page Service for Specific U.S. Clients (90%-tile First Byte Time Results)

In our final two sets of results for the U.S. clients we change the performance metric used to evaluate clients from "First Byte Time" to "Total Time". Figure 9 shows results are comparable to those shown in Figure 6 except the performance metric has changed. The total time reflects retrieval of all data for a service thus the effect of a larger RTT could delay data reception due to the TCP ACK mechanism. Thus it is not surprising that consideration of only the expected service time is not a good policy for this metric, but as shown in Figure 9 the other policies are competitive with DNS. The $P_{n=2}$ and $P_{n=\infty}$ policies even provide slightly better performance than DNS-selected server.

Figure 10 shows the same results as Figure 8 except using the total time metric. As for the first byte time results, the $P_{N=2}$ policy provides the best results for the majority of clients while consideration of only the expected RTT or expected service time generally yield the worst performance.

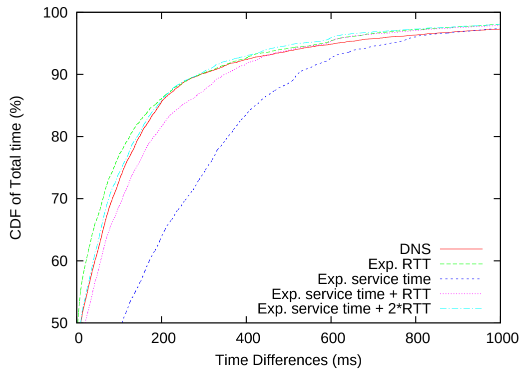


Figure 9: Simulation of Policies for All Services for All U.S. Clients (Total Time)

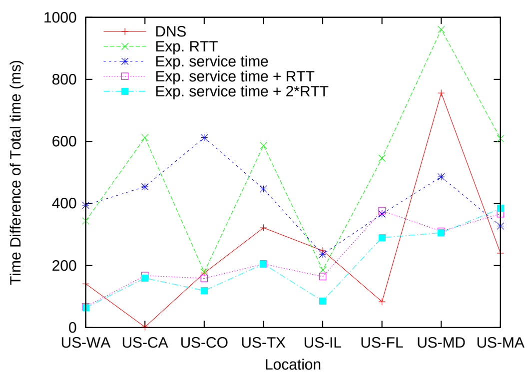


Figure 10: Simulation of Policies for FoxNews Home Page Service for Specific U.S. Clients (90%-tile Total Time Results)

8 Simulation of Policies for Global Client Set

As indicated in Section 4, we performed a similar study for a set of global clients and show a summary of results from these clients in this section. Figure 11 shows the first byte time metric results for all 10 services for all of the 10 global clients. This figure is the same as Figure 6 except for a different client set. The results shown in Figure 11 are similar to those shown in Figure 6 except there is a clear performance penalty for only considering expected service time in the decision. Otherwise the $P_{N=1}$ and $P_{N=2}$ policies are comparable, if not better, results than obtained via DNS and expected RTT policies. The close performance between DNS and expected RTT indicates that RTT is the dominant component in current DNS decision making.

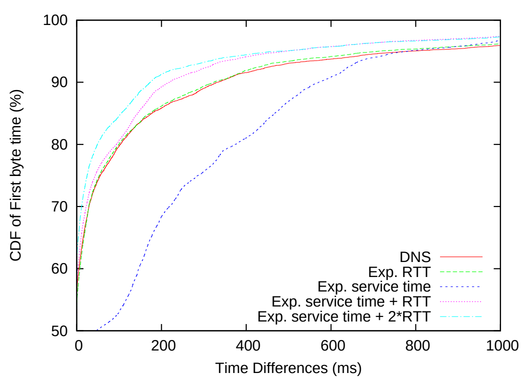


Figure 11: Simulation of Policies for All Services for All Global Clients (First Byte Time)

Figure 12 shows the total time performance metric results for all services and all global clients. On a global scale, the impact of RTT on the retrieval of data ($t_3 - t_2$ in Figure 1) has an even larger impact and it is necessary to weight the contribution of expected RTT performance higher in order to obtain the best performance for this metric.

Our final set of results are shown in Figures 13 and 14, which are comparable to the 90%-tile results shown in Figures 8 and 10 except for the global set of clients, again ordered from west to east starting with the U.S. West

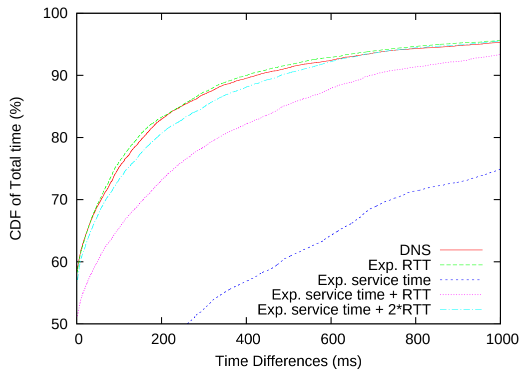


Figure 12: Simulation of Policies for All Services for All Global Clients (Total Time)

coast. These results show that a mix of expected RTT and service time provide comparable results to DNS in most regions for first byte time results. Even for total time results, these policies provide better performance than DNS for the US East, Asia and Australia clients.

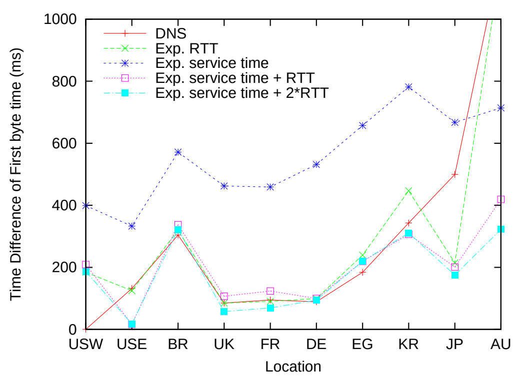


Figure 13: Simulation of Policies for FoxNews Home Page Service for Specific Global Clients (90%-tile First Byte Time Results)

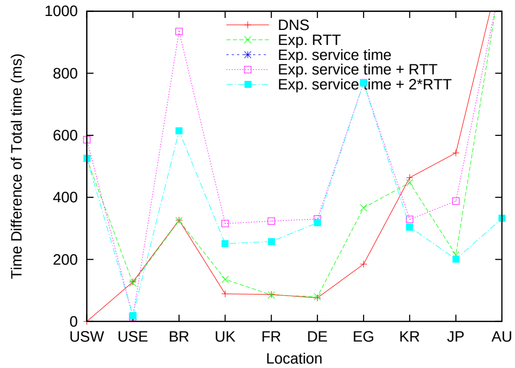


Figure 14: Simulation of Policies for FoxNews Home Page Service for Specific Global Clients (90%-tile Total Time Results)

9 Related Work

Today, multiple data centers are commonly used to serve content. for any large-scale Web site. Generally, in these systems, clients are redirected to different servers for the reasons of (i) improved client latency and (ii) providing load balancing. This is mostly done by using DNS to return server IPs of the closest data center to the clients. The effectiveness of this approach is studied in [7, 8]. This approach is widely deployed on the current Internet by various providers. However, one of the limitations for this approach is the need for name exposure. Using the same domain name for different services would result in making the same decision for clients requesting different services. Another limitation for prior work is the round-trip time and front-end server load are usually the dominant factors to consider when making decisions. Our work suggests Web sites to expose their service names in domain names for better refined decisions, and we also propose the consideration of service time in mapping clients to servers, which implicitly related to the back-end server load.

Content Distribution Networks are a popular means to serve static content for web sites. Work in [4] had preliminary measurements on the performance of CDN content delivery. Google, in [6], studied the performance of its own CDN. They built a system called *WhyHigh* to find out the causes for some large latencies appearing in their CDN. They found queueing delays and routing inefficiencies are the root causes for the high latency in their CDN. Our work has a different point of view. We observe the performance from a client perspective and propose a means to account for significant back-end server response time.

In addition to mapping clients to data centers, resource provisioning is another research direction for improving the Web performance. While not directly related, our work was partially inspired by the work of dynamic resource provisioning between multiple dependent services [2, 11, 12]. Intuitively, allocating more resource to the back-end server may improve the 'Service Time', while putting more front-end servers closer the clients may reduce the RTT. Although our approach does not reallocate any resource, we focus on how to make better use of current servers. Our work may, in return, help others work on resource provisioning decisions.

Other related work has been focused on measuring the performance of Web services. One uses analytic models to understand the performance for multi-tier services [10]. Work in [3] proposed an infrastructure can be used to rank Web sites according to download time and monitor DNS server availability.

10 Summary and Future Work

In this work we have explored the explicit use of back-end service time for Web-based services in mapping clients of these services to specific client-facing servers that provide them. We gathered data from currently deployed server platforms of popular Web sites and used the decisions of the existing DNS mechanism as a benchmark for mapping clients to specific front-end servers. Using two distinct set of clients—one set located in the U.S. and another set located around the world—we first performed a best-case analysis to see if the front-end server providing the best service time (or best RTT) to the client would have given better performance than the server selected via DNS. We then went on to propose simple policies for the combination of expected RTT and service time. Evaluating these policies for a range of performance metrics with a trace-driven simulation over the collected data, we observed better performance by our policies compared with DNS for some services and comparable performance with DNS for the remaining services. Given the increasing use of Web-based services, these results are important in showing that client-perceived performance can be improved by consideration of back-end service time costs.

Moving forward, there is much opportunity for future work. We plan to extend our methodology to study the potential improvement of other currently deployed Web-based services to understand the applicability of our techniques. We also plan to examine the reasons that service times for a particular service are large when it is provided by some front-end servers, but not for others. There are multiple explanations such as a poorly performing front-end server, an over-loaded local back-end server or the need to retrieve data from a remote back-end server. There is also need to study this problem not only from the client perspective, but also to set up controlled server platforms so that we can measure performance on a platform with a known configuration for how Web-based services are provided.

References

[1] Alexa: Most popular web sites. `http://www.alexa.com/`.

[2] D. Jiang, G. Pierre, and C.H. Chi. Autonomous resource provisioning for multi-service web applications. In *Proceedings of the 19th international conference on World Wide Web*, pages 471–480. ACM, 2010.

[3] Balachander Krishnamurthy, Harsha V. Madhyastha, and Oliver Spatscheck. Atmen: a triggered network measurement infrastructure. In *Proceedings of the 14th international conference on World Wide Web*, pages 499–509, New York, NY, USA, 2005. ACM.

[4] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, pages 169–182, San Francisco, November 2001.

[5] Balachander Krishnamurthy and Craig E. Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the World Wide Web Conference*, Madrid, Spain, April 2009. ACM.

[6] R. Krishnan, H.V. Madhyastha, S. Srinivasan, S. Jain, A. Krishnamurthy, T. Anderson, and J. Gao. Moving beyond end-to-end path information to optimize CDN performance. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 190–201. ACM, 2009.

[7] Jeffrey Pang, Aditya Akella, Anees Shaikh, Balachander Krishnamurthy, and Srinivasan Seshan. On the responsiveness of dns-based network control. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 21–26, New York, NY, USA, 2004. ACM.

[8] Anees Shaikh, Renu Tewari, and Mukesh Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of the IEEE Infocom 2001 Conference*, Anchorage, Alaska USA, April 2001.

[9] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24, 2006.

[10] Bhuvan Urgaonkar, Giovanni Pacifici, Prashant Shenoy, Mike Spreitzer, and Asser Tantawi. An analytical model for multi-tier internet services and its applications. In *Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 291–302, New York, NY, USA, 2005. ACM.

[11] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3(1):1–39, 2008.

[12] BangYu Wu, Chi-Hung Chi, and Zhe Chen. Resource allocation based on workflow for enhancing the performance of composite service. In *Services Computing, 2007. IEEE International Conference on*, pages 552 –559, jul 2007.