Wide Area Redirection of Dynamic Content by Internet Data Centers

S. Ranjan, R. Karrer and E. Knightly
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005, USA

***Abstract*— Traditional approaches to mirroring, caching, and content distribution have an underlying assumption that minimizing network hop count minimizes client latency. However, with uncongested backbones and potentially high-latency service times for dynamic content, such techniques are of limited effectiveness.**

In this paper we present an architecture in which dispatchers at an overloaded Internet Data Center (IDC) redirect requests for dynamic content to a geographically remote but less loaded IDC. We show with both analytical modeling as well as testbed experiments that the delay savings of redirecting requests to a lightly loaded IDC can far outweigh the overhead in inter-IDC network latency. Consequently, client end-to-end delays are significantly reduced without requiring modifications to clients, servers, or DNS.

***Index Terms*— Internet Data Centers, Performance, Content Distribution**

I. Introduction

Web content providers are increasingly off-loading the task of content placement and distribution to Content Delivery Networks (CDNs) such as Akamai, Digital Island, or Mirror Image. The primary objective of CDNs, as well as of mirroring and caching strategies, is to reduce the network latency between the clients and the data they are accessing. This is done in two ways: firstly, by directing clients to the closest server [1], [2], [3], [4], [5], [6] and secondly, by placing the most popular *urls* on replicas closer to *hot-spots* [7], [8].

However, two key trends challenge the premise that minimizing network latencies minimizes a client's end-to-end delay. First, because of over-provisioning in the Internet core [9], [10], delays across network backbones are increasingly dominated by speed-of-light delays, with minimal router queuing delays. Second, web content is increasingly dominated by *dynamic* content that requires online processing of requests, e.g., e-commerce sites. We will show and exploit the fact that the processing times of dynamic content on moderately to highly loaded servers can exceed network delays by an order of magnitude.

In this paper, we introduce WARD (*W*ide *A*rea *R*edirection of *D*ynamic content), a novel architecture for redirection of dynamic content requests from an overloaded Internet Data Center (IDC) to a remote replica. The key objective is to reduce end-to-end client delays by determining at the IDC whether the sum of networking and server processing delay is minimized by servicing the request remotely (via redirection) or locally. In WARD, client requests are first routed to an initial IDC via any mechanism available today (e.g., from simple DNS round-robin to more sophisticated server selection schemes, as described in [5]). Upon arrival at the initial IDC, a *request dispatcher* uses a measurement-based delay-minimization algorithm to determine whether to forward the request to a remote or local server. Thus, unlike previous approaches, WARD performs IDC-driven request redirection, incorporates networking and server-processing delays to minimize total delay, and requires no changes to clients, DNS or web servers.

Next, we develop a simple analytical model to characterize the effects of wide area request redirection on end-to-end delay. The model consists of a system of dispatchers, M/G/1 queues that represent servers, and inter-server delays that capture the cost of redirecting to a remote IDC. With this model, we derive an expression for the optimal percentage of requests that should be dispatched to a remote IDC replica under given server and network characteristics. Moreover, we compute the expected average request response time under this dispatching policy. We then perform a systematic performance analysis to study the impact of key performance parameters such as server load, inter-IDC network latency, and measurement errors that can be expected in realistic systems.

Finally, we implemented WARD and developed a testbed consisting of (1) clients emulating an e-commerce workload based on TPC-W benchmark, (2) wide area network links emulated via Nistnet, (3) a web server tier, (4) a request dispatcher that performs remote and local redirection via the algorithms as described above, and (5) a database tier that processes requests. The experiments show that the analytical model provides a close match with experimental results for server loads up to 70%. For higher loads, effects unique to the implementation (e.g., database table conflicts) lead to a deviation between model and testbed. Regardless, both the model and implementation results indicate that WARD can achieve significant reduction in end-to-end delay. For example, for an e-commerce site with 300 concurrent clients, wide area redirection reduces the mean response time by 54%, from 5 sec to 2.3 sec.

The remainder of this paper is organized as follows. In Section II, we describe the system architecture of IDCs for wide area request redirection. In Section III, we develop a queuing model to study the architecture and in Section IV we present numerical studies of the fraction of requests dispatched remotely and the expected response times under various scenarios. Next, we describe our testbed implementation and

This research is supported by Hewlett Packard, NSF Grant ANI-0085842, and by a Sloan Fellowship. The authors may be reached via http://www.ece.rice.edu/networks.

measurements in Section V. Finally, we discuss related work in Section VI, and conclude in Section VII.

II. Redirection Architecture and Algorithm

In this section, we present background on today's IDCs, describe WARD, and present a measurement-based redirection algorithm.

A. IDC background

Figure 1 depicts the four-tier architecture prevalent in today's IDCs. To illustrate this architecture, consider the requests of an e-commerce session. First, the *access tier* routes requests to the correct server cluster and performs basic firewall functions such as intrusion detection. Second, upon arriving at the *web tier*, load balancers may parse the request's URL and route it to a web server typically according to a load-balancing policy (e.g., using round robin or more sophisticated policies as in reference [11]). If the request is for a static web page, a server in the web tier serves the requested page. If the request requires dynamic processing, it is routed to the *application tier*. The application tier orchestrates access to the *database tier* for operations such as purchase processing or maintaining the contents of the shopping cart.

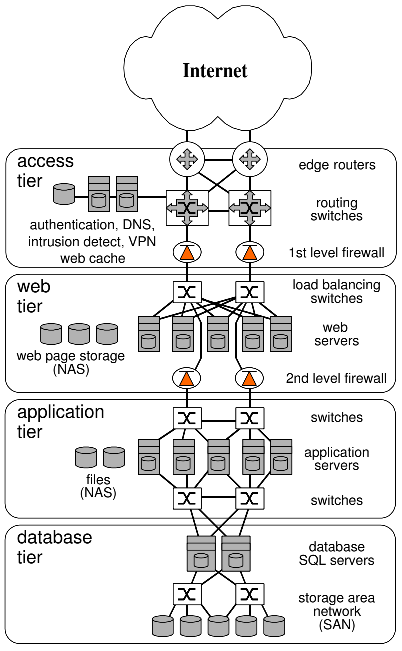

Fig. 1. IDC Four Tier Architecture

B. WARD

In WARD, services and applications are replicated across several geographically dispersed IDCs that are inter-connected via high-bandwidth links. Once a client request arrives at the initial IDC, a dispatcher as illustrated in Figure 2 can decide to service the request locally or redirect it to a less loaded remote IDC. The objective of the algorithm is to redirect requests only if the savings in the request's processing time at the remote IDC overwhelm the network latency incurred to traverse the inter-IDC links in both the forward and reverse path. In this way, end-to-end client delays can be reduced while requiring changes only to the dispatcher.

Note that *clients* cannot determine which IDC will minimize network plus server delay as clients do not have knowledge of each IDC's load and delay characteristics. Thus, clients can select the initial IDC via existing DNS round-robin techniques or they can find the IDC with the smallest network delay using delay estimation tools such as [12]. Upon arrival at the initial IDC, WARD will then minimize the remaining service time.

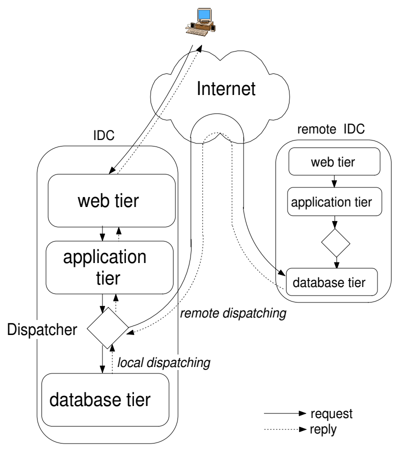

Fig. 2. IDC multi-tiered architecture

WARD therefore provides a foundation for spatial multiplexing of IDC resources. Namely, as a particular IDC becomes a hot-spot due to flash crowds [13], [14] or time-of-day effects [15], load can be transparently redirected to other IDCs while ensuring a latency benefit to clients. For example, client access patterns have been observed to follow *time-of-day* patterns where server utilization varies with a diurnal frequency. We can exploit this effect such that no IDC has to provision for the peak demand. Thus, when the workload to one IDC is peaking, the workload at an IDC several time zones away will be much lower, enabling a significant performance improvement by allowing redirection among IDCs.

C. Redirection algorithm

The objective of the redirection algorithm is to minimize the total time to service a request. Namely, if a request arrives at IDC i, then the objective is to dispatch the request to IDC j satisfying

$$\text{argmin}_j \left(\Delta_{ij} + \Delta_{ji} + T_j\right) \tag{1}$$

where Δ_{ij} denotes the network delay between IDC i and j, and T_j is the request's service time at IDC j.

In practice, T_j can be estimated by measuring the average load ρ_j at IDC j and using information about the request type. IDCs periodically exchange load information to update the load estimates of each others' processing delays. Similarly, latency values can be measured among the IDCs.

We consider two redirection policies. The first is *per-request* redirection in which each request is sent to the IDC that minimizes Equation (1). The second is *probabilistic* redirection

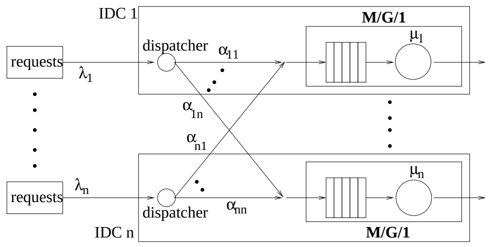

Fig. 3. IDC system model.

in which a fraction $0 \leq \alpha_{ji} \leq 1$ of requests are redirected from IDC j to IDC i. In particular, we show in the next section that under certain simplifications there is an optimal ratio of requests that should be remotely dispatched in order to minimize the delay of all requests. Once this ratio is known, the dispatcher remotely redirects requests according to the computed probability.

III. Performance Model

In this section, we develop a performance model for wide area redirection. For a given workload, mean and variance of service time, and network latency, we derive an expression for the delay-minimizing fraction of requests that a dispatcher should redirect to remote IDCs. Moreover, we compute the average total response time including service- and waiting-times and end-to-end network latency.[1] We then perform a systematic performance analysis to estimate the optimal dispatching ratios α^*_{ji} and to predict the expected request response time under different parameters, such as the server load, the end-to-end network latency and the average request service time.

Figure 3 illustrates the system model for WARD. We model request arrivals at IDC i as a Poisson process with rate λ_i and consider a single bottleneck tier modeled by a general service time distribution having mean $\overline{x}_i$ and variance σ_i^2.

We consider a redirection algorithm in which a request is redirected from IDC j to IDC i with probability α_{ji}, i.e., we consider probabilistic redirection. Denote $E[T_i]$ as the expected total delay for servicing a request at IDC i, and denote Δ_{ji} as the one-way network latency for a request sent from IDC j to IDC i.

For the general case of a system of n IDC replicas, denote $\mathbf{A} = \{\alpha_{11}, \ldots, \alpha_{ji}, \ldots, \alpha_{nn}\}$ as a matrix of request dispatching fractions, $\mathbf{E}[\mathbf{T}] = \{T_1, \ldots, T_n\}$ as the vector of all total delays at an IDC bottleneck tier and $\mathbf{D}$ as a matrix of round-trip times from IDC i to j such that $D_{ji} = \Delta_{ji} + \Delta_{ij}$. Furthermore, denote $\mathbf{L} = \{\lambda_1, \ldots, \lambda_n\}$ as a vector of request arrival rates at the IDC dispatchers, $\overline{\mathbf{X}} = \{\overline{x}_1, \ldots, \overline{x}_n\}$ as the average service time, $\mathbf{C} = \{c_1, \ldots, c_n\}$ as the vector of squared coefficient of variation for the service times, with $c_i = \sigma_i^2/\overline{x}_i^2$.

Result 1: The mean service time for the redirection policy using a dispatching fraction $\mathbf{A}$ is given by:

$$\mathbf{E}[\mathbf{T}] = \overline{\mathbf{X}} + \frac{(\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}}^2(1 + \mathbf{C}^2)}{2(1 - (\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A} \cdot \mathbf{D} \tag{2}$$

Proof: The total service time is composed of 3 durations: (i) the network latency of transferring the request to and from the remote IDC (ii) the queuing time at the IDC and (iii) the service time at the IDC.

For symmetry reasons, in the following equations we attribute the "costs" to the receiving IDC i. First, we assume that the network latency between the dispatcher and a local IDC $\Delta_{ii} = 0$ and hence, network latency is incurred only by requests dispatched to a remote IDC as given by:

$$\alpha_{ji}(\Delta_{ji} + \Delta_{ij}). \tag{3}$$

Second, consider the mean waiting time for a request in an IDC queue before being serviced. In general, the waiting time for for an M/G/1 queue is:

$$\frac{\rho\overline{x}(1 + c^2)}{2(1 - \rho)} \tag{4}$$

with $\rho = \lambda\overline{x}$.

For any IDC i, the arrival rate λ is the sum of the requests that are dispatched from all IDCs j to IDC i, i.e., $\lambda_i = \sum_j \alpha_{ji}\lambda_j$. With this λ, Equation (4) can be rewritten for a single IDC i as:

$$\frac{(\sum_j \alpha_{ji}\lambda_j)\overline{x}_i^2(1 + c_i^2)}{2(1 - (\sum_j \alpha_{ji}\lambda_j)\overline{x}_i)} \tag{5}$$

Finally, the service time for a request at IDC i is given by $\overline{x}_i$. The addition of these 3 terms for a set of IDCs yields Equation (2). ■

From Equation (2), we can compute the optimal dispatching ratios that minimize the service times over all requests. In particular, let $\mathbf{A}^* = \{\alpha^*_{11}, \ldots, \alpha^*_{nn}\}$ denote the matrix of optimal request dispatching ratios.

Result 2: The optimal dispatching ratios $\mathbf{A}^*$ are given by:

$$\frac{\partial}{\partial\alpha}(\overline{\mathbf{X}} + \frac{(\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}}^2(1 + \mathbf{C}^2)}{2(1 - (\mathbf{A} \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A} \cdot \mathbf{D}) = 0 \tag{6}$$

with $\mathbf{E}[\mathbf{T}]$ defined in Equation (2).

To solve Equation (6) for all α_{ji}, we make a set of simplifying assumptions to reduce the number of unknowns (for more general solutions, see e.g. [16], [17]). First, we clearly have that $\sum_j \alpha^*_{ji} = 1$. Second, we assume that all IDCs have equal processing times, i.e., $\overline{x}_i = \overline{x}_j$. Third, $\lambda_i \geq \lambda_j \implies \alpha^*_{ji} = 0$, i.e., when considering 2 IDCs with different λ, under steady-state conditions, no requests will be dispatched from the IDC with a smaller arrival rate to the IDC with a higher arrival rate.

The optimal dispatching ratios $\mathbf{A}^*$ can be used to predict the average request service time for a system of IDC replicas.

Result 3: The expected request service time under optimal dispatching ratios is given by:

$$\mathbf{E}[\mathbf{T}^*] = \overline{\mathbf{X}} + \frac{(\mathbf{A}^* \cdot \mathbf{L})\overline{\mathbf{X}}^2(1 + \mathbf{C}^2)}{2(1 - (\mathbf{A}^* \cdot \mathbf{L})\overline{\mathbf{X}})} + \mathbf{A}^* \cdot \mathbf{D} \tag{7}$$

[1]Throughout, we use the term end-to-end latency to describe the server processing time plus the network round-trip time from the local IDC to the remote and back. We do not consider the client-to-IDC delay.

Proof: Equation (7) follows from Result 1 and by using the optimal dispatching ratios from Equation (6). ■

IV. Numerical results

In this section, we first show that wide area redirection is able to reduce the total access delay. Then, we study the effects of the key performance factors that affect the total delay.

We consider a system of 2 IDCs with replicas having the same average request service time $\overline{x}$. Furthermore, we assume a symmetric network with wide area latency between two IDCs: $\Delta := \Delta_{12} = \Delta_{21}$, so the IDC round-trip time is $D = 2 \cdot \Delta$. Finally, we set $\lambda_2 = 0$, which satisfies $\lambda_1 > \lambda_2 \implies \alpha_{21} = 0$, and denote $\lambda := \lambda_1$ and $\alpha^* := \alpha^*_{12}$ for simplicity.

The dispatching ratio is computed based on Equation (2):

$$E[T_1] = \overline{x} + \frac{(1-\alpha)\lambda\overline{x}^2(1+c^2)}{2(1-(1-\alpha)\lambda\overline{x})} \tag{8}$$

$$E[T_2] = \overline{x} + \frac{\alpha\lambda\overline{x}^2(1+c^2)}{2(1-\alpha\lambda\overline{x})} + \alpha D \tag{9}$$

Equations (8) and (9) are solved according to Result 2 to obtain the optimal dispatching ratio α^*. Henceforth, we refer to the term α^* as the *(remote) redirection* ratio, i.e. the fraction of requests dispatched to the remote IDC, and $1-\alpha^*$ is the fraction of requests processed on the local IDC. Then, according to Result 3, the expected total delay of the IDC system is given by:

$$\begin{aligned} E[T^*] = \overline{x} + \frac{(1-\alpha^*)\lambda\overline{x}^2(1+c^2)}{2(1-(1-\alpha^*)\lambda\overline{x})} + \\ \overline{x} + \frac{\alpha^*\lambda\overline{x}^2(1+c^2)}{2(1-\alpha^*\lambda\overline{x})} + \alpha^* D \end{aligned} \tag{10}$$

If not otherwise stated, we use the following default values: $\overline{x} = 42.9$ msec, $\sigma = 40.1$ msec, where these values were obtained from our testbed and one-way network latency $\Delta = 36$ msec, which corresponds to a speed-of-light latency for two IDCs separated by 6 time-zones at 45^o latitude. We will use $\rho = \lambda\overline{x}$ to denote the total load on *all* IDCs. For IDCs without redirection, ρ corresponds to the server load on the bottleneck tier, whereas WARD can split this load among the local and remote IDCs. To obtain a given value of ρ, the arrival rate λ will be scaled, with $\overline{x}$ remaining fixed.

A. The case for wide area redirection

First, we provide evidence that wide area redirection is able to decrease the user-perceived total delay. We calculate the total delay of WARD using Equations (6) and (7) and compare it to the total delay of an IDC without redirection. Figure 4 shows the total delay as a function of the network latency Δ for different system loads ρ.

For a load $\rho = 0.5$, improvements are achieved only when the network latency $\Delta \leq 25$ msec. For $\Delta > 25$ msec, the redirection cost exceeds the processing time so that all requests are serviced locally. However, a significant improvement is achievable under higher loads. For a moderate load of $\rho = 0.75$ and a latency $\Delta < 50$ msec, the total delay is reduced from 0.16 sec to <0.13 sec using WARD, an improvement of $>$ 18%. For a heavily loaded system with $\rho = 0.9$ and $\Delta < 50$ msec, the total delay is reduced from 0.38 sec without redirection to <0.15 sec using WARD, an improvement of $>$ 60%. Moreover, for loads $\rho > 0.9$, still higher improvements are predicted by the model.

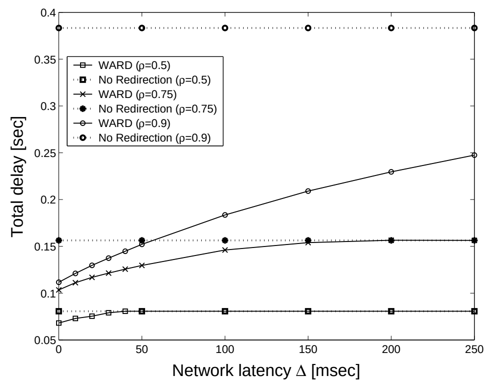

Fig. 4. Comparison of the total IDC delay with and without wide area redirection

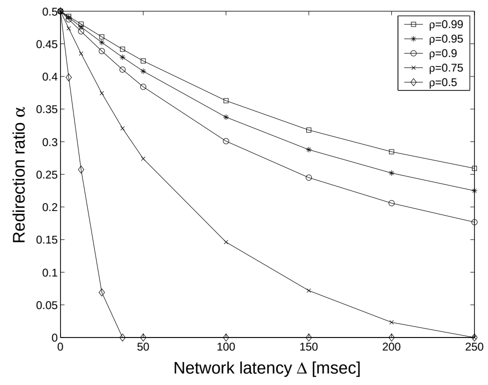

Fig. 5. Redirection ratio α for different network latencies Δ and server loads ρ.

B. Server load versus network latency

Next, we study the influence of server load and network latency on the redirection decision. An increased network latency implies a higher overhead to send a request to a remote IDC and here we quantify the network latencies for a given server load, until which we can expect gains out of redirection. The influence of the network latency Δ for different server loads ρ on the optimal redirection ratio (Equation (6)) is shown in Figure 5. Each curve depicts a value of server load ρ and the x-axis denotes the network latency Δ between two IDCs.

Results with $\Delta = 0$ therefore correspond to an IDC with 2 local servers. Observe that in this case, the redirection ratio α is 0.5 independent of the server load. Since no latency costs are incurred, the optimal strategy is to equally balance the load on the two local servers. We make two further observations: Firstly, for a given network latency, the model redirects a greater number of requests as the server load increases. Secondly, for a given server load, the redirection

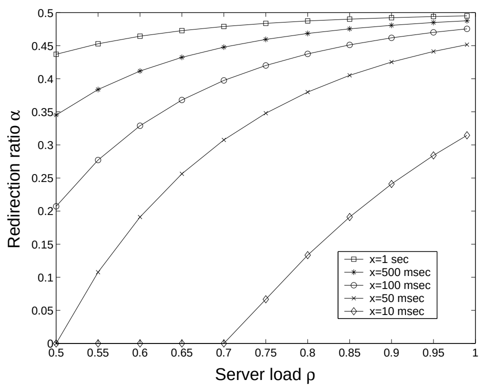

Fig. 6. Redirection ratio α for different service times $\overline{x}$ and server loads ρ.

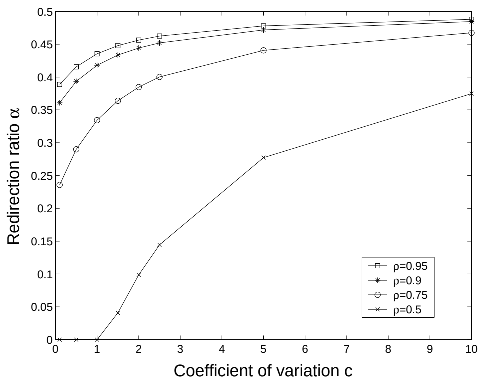

Fig. 7. Redirection ratio α for different coefficient of variations c and server loads ρ.

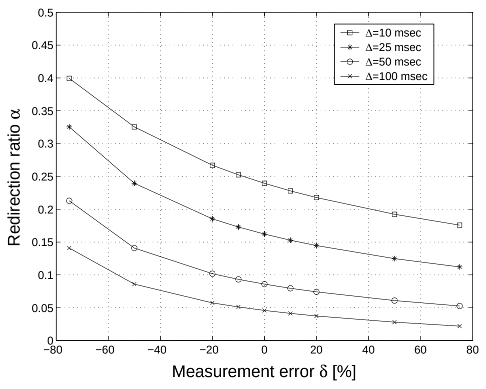

Fig. 8. Redirection ratio α for different network measurement errors δ.

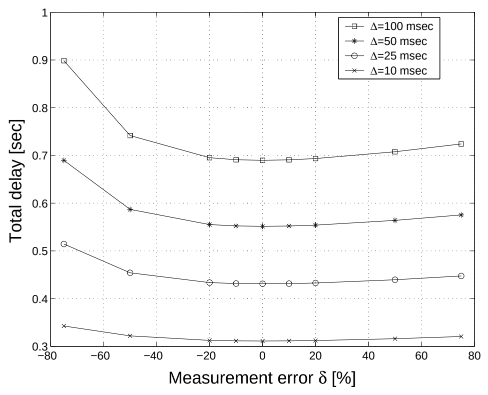

Fig. 9. Total delay for different network measurement errors δ.

ratio decreases as the network latency is increased and goes to zero when the cost of network latency exceeds the reduction in server processing time due to redirection. For a non-heavily loaded system ($\rho = 0.5$), it is advantageous to redirect only when the network latency $\Delta < 40$ msec, while for a moderate load of $\rho = 0.75$, redirection is advantageous for network latency as high as 250 msec. For $\rho \geq 0.9$, the model predicts redirection ratios of 0.2 for latencies even higher than 250 msec.

C. Service time

Figure 6 illustrates the effect of the request service time mean $\overline{x}$ on the redirection ratio. The x-axis denotes the server load ρ and each curve denotes a different mean service time. For a small service time $\overline{x} = 10$ msec, the redirection ratio α is 0 for server loads up to $\rho = 0.7$. Only above this high load does the dispatcher send requests to a remote IDC because the redirection costs exceed the service time on the local IDC.

When the request service time $\overline{x}$ increases, the redirection ratio increases for a given server load ρ. For service times of 50 msec, which is close to the testbed value, an increase in the redirection ratio starts at $\rho = 0.5$. Finally, for $\overline{x} = 1$ sec, the ratio stays above 42%.

Figure 7 shows the effect of the coefficient of variation c on the redirection ratio. It can be expected that an increase in variance leads to an increase in the redirection ratio. Figure 7 shows the greatest influence of the variance when c increases from < 1 to 1 for $\rho \geq 0.75$.

Hence, our model predicts an increase in redirection ratios when either the service time or the coefficient of variation increases. This allows us to predict that future e-commerce sites designed with greater complexity in their dynamic content would achieve still higher performance gains from using wide area redirection.

D. Measurement errors

The dispatcher bases its redirection decision on 2 measured values: the network latency Δ and the server load ρ. So far, we have assumed that perfect information is available for this decision. In this section, we study the impact of measurement errors on the effectiveness of the algorithm and quantify it in terms of *error tolerance*. defined as the percentage error $\pm\epsilon$ that increases the total delay by at most 2%.

First, we study the impact of network latency measurement errors as follows. Let Δ denote the true latency from a local to a remote IDC and back, and $\widehat{\Delta} = \Delta + \delta$ the measured value, and $\widehat{\mathbf{D}}$ the corresponding round-trip time matrix. The dispatcher calculates the dispatching ratios using $\widehat{D}$ (Equation (2))

$$\mathbf{E}[\mathbf{T}] = \overline{\mathbf{X}} + \frac{(\mathbf{A}\cdot\mathbf{L})\overline{\mathbf{X}}^2(1+\mathbf{C}^2)}{1-(\mathbf{A}\cdot\mathbf{L})\overline{\mathbf{X}}} + \mathbf{A}\cdot\widehat{\mathbf{D}} \qquad (11)$$

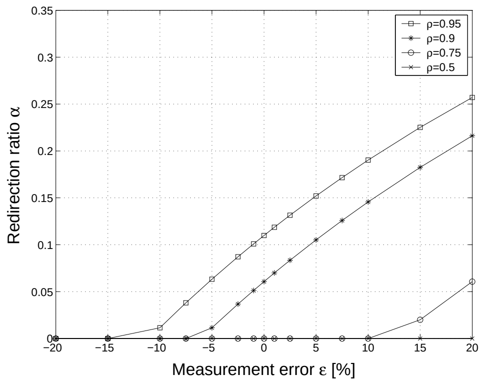

Fig. 10. Redirection ratio α for different server load measurement errors ϵ.

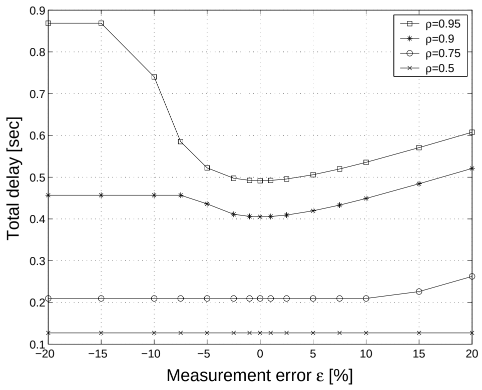

Fig. 11. Total delay for different server load measurement errors ϵ.

For the calculation of average total delay, Equation (7) is used with the true latency values $\mathbf{D}$. The effects of measurement error in network latency on the redirection ratio α and the resulting average request response time are shown in Figures 8 and 9 respectively. Each curve denotes a different (true) latency Δ, and the x-axis denotes the error δ, in percent of Δ. A value of 0 on the x-axis corresponds to perfect end-to-end latency information. The server load is set to $\rho = 0.95$.

Figure 8 shows that the redirection ratio changes more for negative δ than for the corresponding positive δ. The reason is that the redirection ratio does not grow linearly with the end-to-end latency, as shown in Figure 5. As a consequence of the asymmetry, the total delay increases more for negative δ, as shown in Figure 9. Note, however, that the response times are not highly sensitive to latency measurement errors and the the error tolerance is quite high at $\pm 20\%$.

Likewise, we consider a scenario when the dispatcher has inaccurate server load measurements, e.g., due to delays in receiving the measurements. In this scenario, the measured load at the dispatcher is given by $\widehat{\rho} = \widehat{\lambda}\overline{x}$, with $\widehat{\lambda} = \lambda + \epsilon$ (where ϵ is in percent of the correct load ρ) and the corresponding measured arrival rate by $\widehat{\mathbf{L}}$. The one-way network latency is set to 36 msec.

First, consider the case of measurement error $\epsilon > 0$, when the dispatcher assumes the server load to be higher than what it is and hence it redirects more requests than the optimal. Figure 10 shows that the redirection ratio increases almost linearly for $\rho \geq 0.9$, while for $\rho = 0.75$ the ratio doesn't start increasing until $\epsilon \geq 10\%$. The extra redirections incur additional network latencies and hence the total delay also increases linearly in Figure 11. In particular, for $\rho \geq 0.9$, the error tolerance is $+1.5\%$. Next, consider negative ϵ, when the dispatcher assumes the local server load to be less than the actual value and hence redirects pessimistically. As a result, the load on the local server incurs greater processing times at the local IDC. As expected, Figure 11 shows that at high server loads $\rho \geq 0.9$, the total delay is more sensitive for negative ϵ with an error tolerance of $\approx -0.5\%$.

Thus, comparing the impact of latency and server measurement errors, the error tolerance for network latency is high at $\pm 20\%$ while that for server load is an order of magnitude lower at $+1.5, -0.5\%$. We thus conclude that (1) greater accuracy is needed in server load measurements than network latency and (2) IDC-driven redirection can have greater robustness than existing client-driven schemes as IDCs can obtain accurate server load information when compared to clients, client-side proxies, or DNS.

V. Testbed implementation and experiments

In this section, we describe our prototype implementation and testbed experiments of the WARD architecture. Our results provide a proof-of-concept demonstration of wide area IDC-driven redirection, explore the testbed's key performance factors, and validate the performance model.

A. WARD Testbed

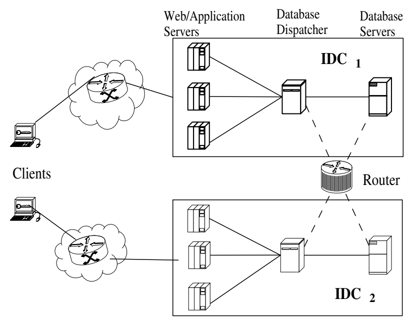

Fig. 12. Testbed

The testbed, depicted in Figure 12, consists of a cluster of Intel Pentium IV 2.0 GHz processor machines running Linux 2.4.18-14, with 512 MB SDRAM, and a 30 GB ATA-66 disk drive. One machine is configured as a router and runs Nistnet [18], an IP-layer network emulation package. The router separates the remaining machines into 3 domains, 2 for IDCs and 1 for the client. This setup allows variation of the network conditions (delay and bandwidth) between the client and the IDCs as well as between IDCs.

We developed a system architecture depicted in Figure 2. At the web tier, we use an Apache web server and dynamic content is coded using PHP scripts at the application tier. Access to the 4 GB database is provided by a MySQL server.[2]

[2]See apache.org, php.net, and mysql.com respectively.

The workload is driven by a client emulator implementing an e-commerce browsing mix workload characterizing an online bookstore site [19] following the TPC-W benchmark.[3] The client emulator opens a set of n user sessions which last 15 minutes. Each session opens a persistent HTTP connection to the web server and sends a sequence of requests to the IDC. Between two requests, the user waits for a configurable parameter termed *think time* before the next request is generated. The mean think time, which we set to 7 sec, together with the number of users, defines the request arrival rate at the IDC.

At the IDC, the PHP scripts specified in the request are executed by the web and application tier. Every PHP script may generate one or multiple database queries that are executed sequentially by the application tier. These queries arrive at the dispatcher with the arrival rate λ used in Section III.

The dispatcher first determines the query type: 95% of the queries in the browsing mix are read queries while 5% account for write queries. To ensure consistency among all IDCs, the dispatcher follows a *read-one write-all* dispatching strategy combined with an identical *total ordering* of writes at all database servers. That is, all write queries must be executed at every IDC in the same logical order they arrived at the dispatcher. To maintain the order, every query is assigned a unique sequence number. To execute a write query with number n, all previous queries $k < n$ must be terminated and all subsequent queries $l > n$ must be queued until n is finished. Read-one write-all implies that the dispatcher is free to send a read query to *any* IDC whereas write queries must be processed at *all* IDCs.

The redirection algorithm therefore works as follows. If a write query arrives, it is sent to all IDCs and executed in total order. When a read query arrives, the dispatcher checks for IDCs which have outstanding write requests. These IDCs are not considered for redirection because the read query cannot be processed immediately. Of the remaining IDCs, the dispatcher selects an IDC by using either the *per-query* or, the *probabilistic* redirection policy. In the per-query redirection policy, the dispatcher calculates the expected response time by using measured loads of database tiers to determine if the latency overhead incurred by remote dispatching is outweighed by the savings in server processing time. In the probabilistic policy, the dispatcher uses the optimal redirection ratio computed by the model and dispatches queries with that probability. We implement the probabilistic policy such that given a number of clients, it is configured for the redirection ratio predicted by the model and hence it doesn't use the online server load measurements.

The described consistency issues have to be addressed in the real WARD implementation, but they are not accounted for in the performance model. While they limit the ability of the dispatcher to achieve an optimal redirection ratio, the above consistency implementation has been shown to scale to higher throughputs than other techniques [20]. Nevertheless, we expect that the dispatcher is still able to redirect queries because (i) the vast majority of the queries are non-conflicting read queries and (ii) read queries have a larger service time than write queries [21].

[3]See tpcw.org.

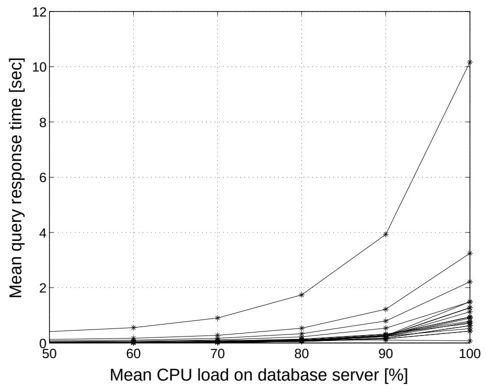

Fig. 13. Mean database query response times vs mean database CPU load for the 30 read-only MySQL queries in the browsing mix of the TPC-W benchmark.

Finally, we note that while WARD provides a general redirection mechanism that can be applied to all IDC tiers, our implementation only redirects before the database tier. This implementation decision is motivated by studies that found that the database tier is often the bottleneck due to the substantial processing demands of complex database queries [20].

B. Experimental Setup

The input parameters in our experimental study are the inter-IDC link latency and the number of clients. The latency is varied through the Nistnet module at the router. The parameters we measure are *request response time* as perceived by the clients, *query response time* as perceived by the database dispatcher and *remote redirection ratio* as achieved by the database dispatcher. Request response time is defined as the time elapsed between the generation of a request and the return of the last byte of the response to the client. Query response time is defined as the time period between the sending of a query by the dispatcher to the database and the reception of the response by the dispatcher. We measure the mean- and 90-%ile request (query) response time for all requests (queries) generated during the entire duration of an experiment. The redirection ratio is defined as the fraction of the number of queries sent by the dispatcher to a remote database server.

C. Experiments

Here, we first present the offline technique to configure our *per-query* redirection policy with the *query response time characteristics*. Second, we quantify the performance benefits of the WARD architecture by exploring the trade-off between the load on the local database server and wide area link latency. Third, we compare performance gains predicted by the analytical model of section III with those obtained via testbed measurements.

1) *Offline measurement of query response time characteristics:* In these experiments, we measure the response time as a function of CPU load, a key input to the per-query redirection policy. We use one IDC with access to one local database server. The execution time for a query depends on

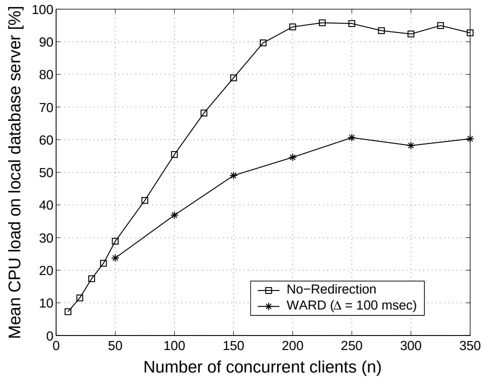

Fig. 14. Mean database CPU load as a function of the number of concurrent clients (n).

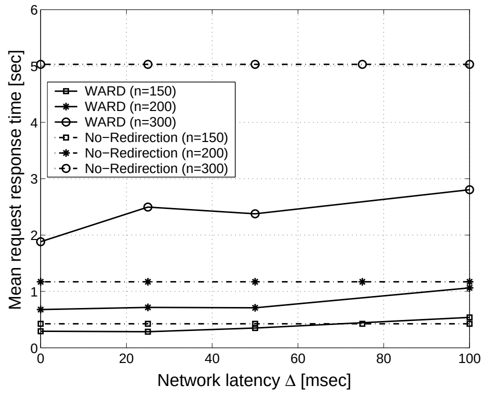

Fig. 15. Mean request response time as a function of the network latency Δ and the number of concurrent clients n.

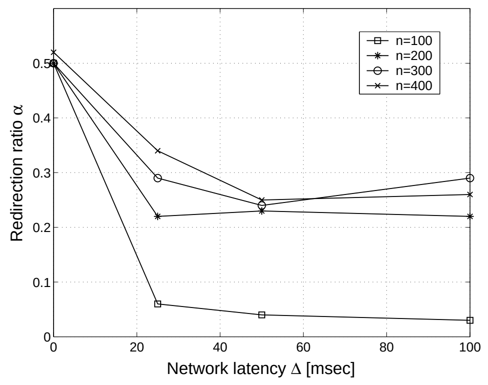

Fig. 16. Redirection ratio as a function of the network latency Δ and the number of concurrent clients n.

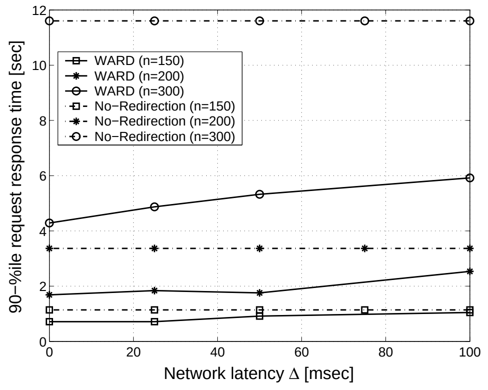

Fig. 17. 90 %-iles of response time as a function of the network latency Δ and the number of concurrent clients n.

the number and type of other queries executing at the same time on the database server, which can be abstracted as the workload entering the system. Hence, we vary the CPU load on the database server by increasing the number of clients. In each case, we measure the mean execution time for each of the 30 read-only MySQL queries. The resulting delay-load curve as illustrated in Figure 13 is then used in the *per-query* redirection policy.

2) WARD Performance Gains: In this experiment we quantify the delay reductions achieved by the *per-query* redirection policy by considering the trade-off between the two parameters of wide area link latency and CPU load on the local database server.

We compare the following two architectures: (1) *No-Redirection* architecture with two IDCs, each of which has access to one local database server and doesn't employ wide area redirection and (2) *WARD* architecture with two IDCs, each of which has access to one local and one remote database server and the latency between the two IDCs is Δ varied as 0, 25, 50 and 100 msec. In both architectures, the workload arrives at only one IDC, termed "local," whereas the workload of the remote IDC is solely created by dispatched requests. We expect such low-load conditions on remote IDCs several time-zones away due to the *time-of-day* effects.

Figure 14 compares the local server CPU of an IDC without redirection to WARD with an inter-IDC latency of 50 msec. In the No-Redirection architecture, the CPU load on the database tier reaches 90% for 200 concurrent users. In contrast, WARD keeps the local database server load below 60% even for 350 concurrent users.

The high CPU load on the database server in the *No-Redirection* architecture increases the mean request response time, as shown in Figure 15. For 300 concurrent users, the mean request response time reaches 5 sec. In contrast, the mean request service time of WARD is 2.3 sec for an inter-IDC latency of 50 msec, a 54% reduction. Figure 16 shows that WARD's redirection policy dispatches 24% of the database queries to the remote IDC in this case.

The delay reductions of using wide area redirection increases with increasing CPU load on the local database server. For 150, 200 and 300 concurrent users, the delay reduction is 17%, 40% and 54%. In this case, 150 users corresponds to a moderate load (80%) while 300 users corresponds to a heavily loaded local server (92%) with reference to the figure 14. Hence, we conclude that wide area redirection is of advantage for both long-term provisioning of resources when an IDC operator wants to maintain a moderate load and short-term bottlenecks due to flash-crowds. Similarly, the redirection ratio (Figure 16) as well as the 90-%ile response times (Figure 17) increase with the number of concurrent users. Therefore, WARD achieves a higher throughput than a system without redirection, as shown in Figure 18.

Thus, this experiment quantifies the performance gains of

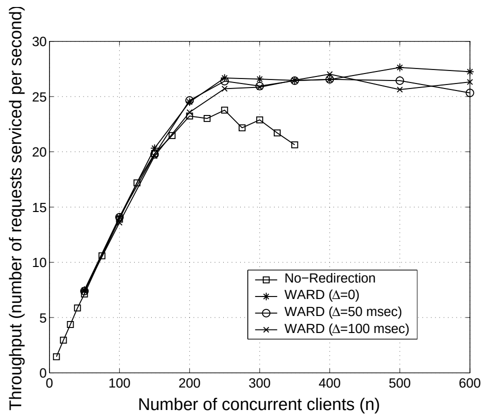

Fig. 18. Throughput obtained for various system configurations.

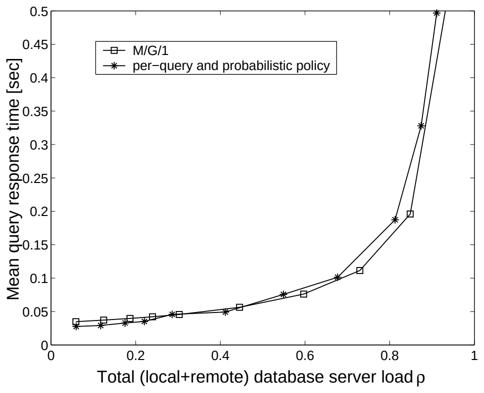

Fig. 19. Model verification: 1 IDC system

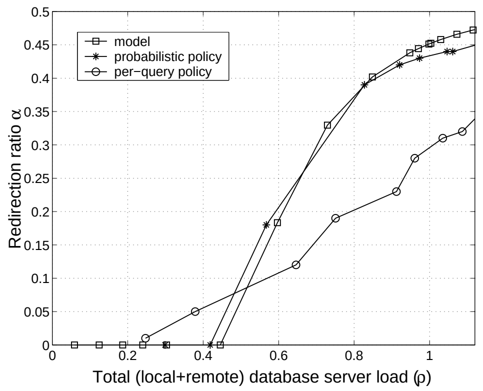

Fig. 20. Model verification: redirection ratio

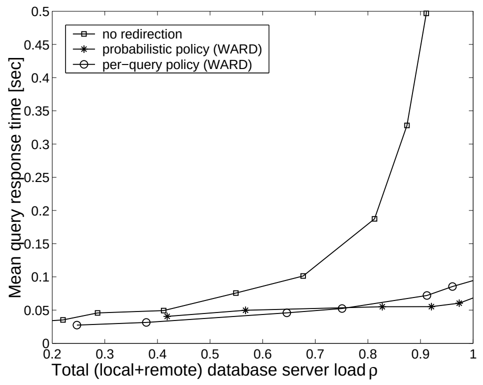

Fig. 21. Model verification: mean response time

wide area redirection of dynamic content. Once a local server is overloaded, remote dispatching can be highly effective for network delays as high as 100 msec.

3) Model validation and redirection policies: We validate the analytical model of Section III with testbed results for both redirection policies. In particular, we compare the redirection ratios and total response times of a system with two IDC replicas. For the model, we use Equations (8), (9) as well as Equation (10) from Section IV with $\overline{x} = 42.9$ msec and $\sigma = 40.1$ msec, as measured on an unloaded database server in our testbed.

Figure 19 compares the mean query response time of the model and the implementation on a single IDC, as a function of the server load ρ. The figure indicates that the model matches the measured query response time for $\rho < 0.7$ within $\pm 10\%$. Beyond this load, the model deviates from the implementation because: (1) our M/G/1 model doesn't take read-write conflicts into account due to which queries may take longer to process that what the model predicts and (2) at high loads there are more queries and thereby greater number of conflicts.

Next, we compare the model with the two implemented redirection policies: (1) *probabilistic*, and (2) *per-query*. The per-query policy receives the CPU load measurements every 5 sec and we set the inter-IDC latency to be 25 msec in all the experiments.

Figures 20 and 21 compare the redirection ratio and query response time as a function of the system load. The redirection ratios of the model and the probabilistic policy are close because this policy bases itself upon the optimal values predicted by the model. On the other hand, the per-query policy begins redirecting earlier and redirects more queries until $\rho < 0.5$ compared to both the model and probabilistic policy. The reason for this behavior is that heavy queries are more sensitive to load as shown in Figure 13, and hence it is of increasing value to redirect them at comparatively lower system loads. Hence we observe a lower mean response time for the per-query policy for $\rho < 0.5$ in Figure 21. When $\rho > 0.5$, the probabilistic policy redirects more queries than the per-query policy and hence yields lower response times. We attribute this difference to the probabilistic policy's better balancing of requests at fast time-scales.

VI. RELATED WORK

Approaches to minimize web access times can be separated into different groups: resource vs. request management and, for the latter, client-side vs server-side redirection.

One approach to minimizing web access times is to ensure that enough resources are available at IDCs. *Server migration* assigns servers that are unused or lightly loaded *within* an IDC to hosted applications that are suffering from high usage [15]. Server migration involves transfer of the application state from an existing server to a new server and hence migration times are on the order of 10 minutes. Therefore, server migration is a means to avoid bottlenecks over a long period of time (minutes or hours), e.g., following time-of-day patterns. Our approach is not only able to address long-term bottlenecks (at the additional redirection costs), but can also address short-term bottlenecks, e.g., due to flash-crowds. *Server sharing*, as applied to content distribution, e.g., [22], is similar to server migration, except that a fraction of the resources are assigned. This option is not applicable to our architecture because we assume that only entire servers, but not fractions of them, are assigned to individual sites. However, both server migration and sharing are orthogonal approaches to request redirection, and we advocate a combination of the mechanisms.

A significant body of research has focused on *client-side* mechanisms such as request redirection in CDNs [5], [23], server selection techniques [24], [3], caching [25], mirroring, and mirror placement [26], [7]. Such techniques are based on the premise that the network is the primary bottleneck. However, we have shown that serving dynamic content shifts the bottleneck onto the IDC. Thus, while such schemes can be applied to finding the best initial IDC, WARD's IDC-driven redirection is essential to jointly incorporating server and network latencies.

Architectures may also combine client-side and server-side redirection [27], [28], [29]. These architectures are most useful if the bottleneck is not clearly identified or varying over time. The server-side redirection mechanism may redirect entire web requests if the CPU utilization exceeds a certain threshold. They conclude that server-side redirection should be used selectively. In contrast, we see server-side redirection as a fundamental mechanism for current and future IDCs. Our redirection mechanism is not threshold-based, but is able to optimize IDC response times for all CPU utilization values. Moreover, [27], [28] design policies which consider network proximity and server load in isolation while our redirection policy integrates the two.

VII. Conclusions

In this paper we presented WARD, an architecture for wide-area request redirection of dynamic content by Internet Data Centers. The objective of WARD is to minimize the end-to-end latency of dynamic content requests by jointly considering network and server delays. We developed an analytical model and proof-of-concept implementation that demonstrated significant reductions in average request response times. For example, for our implementation of an IDC running an e-commerce site and serving 300 concurrent users, WARD can reduce the average response time by 54% from 5 sec to 2.3 sec. Moreover, the model predicts that the performance improvements will further increase when the complexity of dynamic content processing in web requests increases.

WARD is especially suited to prevent increased response times due to short-term bottlenecks, e.g., caused by flash crowds. If the latency costs of redirection are not excessively high, WARD can also be used to exploit long-time-scale trends such as time-of-day driven workloads, and thereby avoid expensive over-provisioning of IDCs. Finally, WARD is an orthogonal solution to client-side redirection and server migration policies and can therefore be seamlessly integrated with such approaches.

VIII. Acknowledgements

The authors are grateful for the discussions and insightful comments from Cristiana Amza (U. Toronto), Rich Friedrich (HP Labs), Jerry Rolia (HP Labs), Willy Zwaenepoel (EPFL), and members of the Rice Networks Group.

References

[1] J. Guyton and M. Schwartz, "Locating nearby copies of replicated internet servers," in *Proceedings of ACM SIGCOMM'95*, Cambridge, MA, Aug. 1995.

[2] A. Shaikh, R. Tewari, and M. Agrawal, "On the effectiveness of dns-based server selection," in *Proceedings of IEEE INFOCOM'01*, Anchorage, AK, Apr. 2001.

[3] Z. Fei, S. Bhattacharjee, E. Zegura, and M. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proceedings of IEEE INFOCOM'98*, San Francisco, CA, Mar. 1998.

[4] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *ACM Symposium on Theory of Computing*, May 1997, pp. 654–663.

[5] L. Wang, V. Pai, and L. Peterson, "The effectiveness of request redirection on CDN robustness," in *Proceedings of OSDI'02*, Boston, MA, Dec. 2002.

[6] M. Karlsson and M. Mahalingam, "Do we need replica placement algorithms in content delivery networks," in *Proceedings of the 7th International Workshop on Web Content Caching and Distribution (WCW'02)*, Boulder, CO, Aug. 2002.

[7] Y. Chen, R. Katz, and J. Kubiatowicz, "Dynamic replica placement for scalable content delivery," in *Proceedings of IPTPS'02*, Cambridge, MA, Mar. 2002.

[8] L. Qiu, V. Padmanabhan, and G. Voelker, "On the placement of web server replicas," in *Proceedings of IEEE INFOCOM'01*, Anchorage, AK, Apr. 2001.

[9] A. Odlyzko, "Data networks are mostly empty and for good reason," *IT Professional*, vol. 1, no. 2, pp. 67–69, Mar. 1999.

[10] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level Traffic Measurement from the Sprint IP Backbone," *IEEE Network Magazine*, vol. 17, no. 6, pp. 6–16, Nov. 2003.

[11] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable content-aware request distribution in cluster-based network servers," in *Proceedings of the USENIX 2000 Annual Technical Conference*, San Diego, CA, June 2000.

[12] E. Ng and H. Zhang, "Predicting Internet network distance with coordinates-based approaches," in *Proceedings of IEEE INFOCOM 2002*, New York, June 2002.

[13] V. Padmanabhan and K. Sripanidkulchai, "The case for cooperative networking," in *Proceedings of IPTPS'02*, Cambridge, MA, Mar. 2002.

[14] J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites," in *Proceedings of the International World Wide Web Conference*, Honolulu, HI, May 2002.

[15] S. Ranjan, J. Rolia, H. Fu, and E. Knightly, "QoS-driven server migration for internet data centers," in *Proceedings of IWQoS'02*, Miami, FL, May 2002.

[16] J. Sethuraman and M. Squillante, "Optimal stochastic scheduling in multiclass parallel queues," in *Proceedings of ACM SIGMETRICS'99*, Atlanta, GA, May 1999.

[17] S. Borst, M. Mandjes, and M. van Uitert, "Generalized processor sharing with heterogeneous traffic classes," *ACM SIGMETRICS Performance Evaluation Review*, vol. 29, no. 3, Dec. 2002.

[18] "NISTNET: Network Emulation Package," http://snad.ncsl.nist.gov/itg/nistnet/.

[19] C. Amza, A. Cox, and W. Zwaenepoel, "Conflict-aware scheduling for dynamic content applications," in *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Seattle,WA, Mar. 2003.

[20] C. Amza, A. Cox, and W. Zwaenepoel, "Distributed versioning: Consistent replication for scaling back-end databases of dynamic content web sites," in *Proceedings of the 4th ACM/IFIP/Usenix Middleware Conference*, Rio de Janeiro, Brazil, June 2003.

[21] C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel, "Specification and implementation of dynamic content benchmarks," in *Proceedings of the 5th IEEE Workshop on Workload Characterization (WWC-5)*, Austin, TX, Nov. 2002.

[22] D. Villela and D. Rubenstein, "Performance analysis of server sharing collectives for content distribution," in *Proceedings of IWQoS'03*, Monterey, CA, June 2003.

[23] J. Kangasharju, K. Ross, and J. Roberts, "Performance evaluation of redirection schemes in content distribution networks," *Computer Communications*, vol. 24, no. 2, pp. 207–214, Feb. 2001.

[24] R. Carter and M. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of IEEE INFOCOM'97*, Kobe, Japan, Apr. 1997.

[25] D. Karger, A. Sherman, A. Berkhemier, B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins, and Y. Yerushalmi, "Web caching with consistent hashing," in *Proceedings of the 8th International World Wide Web Conference*, May 1999.

[26] S. Jamin, C. Jin, A. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," in *Proceedings of IEEE INFOCOM'01*, Anchorage, AK, Apr. 2001.

[27] V. Cardellini, M. Colajanni, and P. Yu, "Geographic load balancing for scalable distributed web systems," in *Proceedings of MASCOTS'00*, San Francisco, CA, Aug. 2000.

[28] M. Rabinovich, Z. Xiao, and A. Aggarwal, "Computing on the edge: A platform for replicating internet applications," in *Proceedings of the 8th International Workshop on Web Content Caching and Distribution*, Hawthorne, NY, Sept. 2003.

[29] V. Cardellini, M. Colajanni, and P. Yu, "Request redirection algorithms for distributed web services," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 4, pp. 355–368, Apr. 2003.