A Web content serving utility

by P. Gayek
R. Nesbitt
H. Pearthree
A. Shaikh
B. Snitzer

Utility computing allows users, or customers, to utilize advanced technologies without having to build a dedicated infrastructure. Customers can use a shared infrastructure and pay only for the capacity that each one needs. Each utility offers a specific information technology service, delivered on a pay-as-you-go model. This paper describes the design and development of a content-serving utility (CSU) that provides highly scalable Web content distribution over the Internet. We provide a technology overview of content distribution and a summary of the CSU from a customer perspective. We discuss the technical architecture underlying the service, including topics such as physical infrastructure, core service functions, infrastructure management, security, and usage-based billing. We then focus on the key issues affecting the performance and capacity of both the service infrastructure and the customer Web sites it supports.

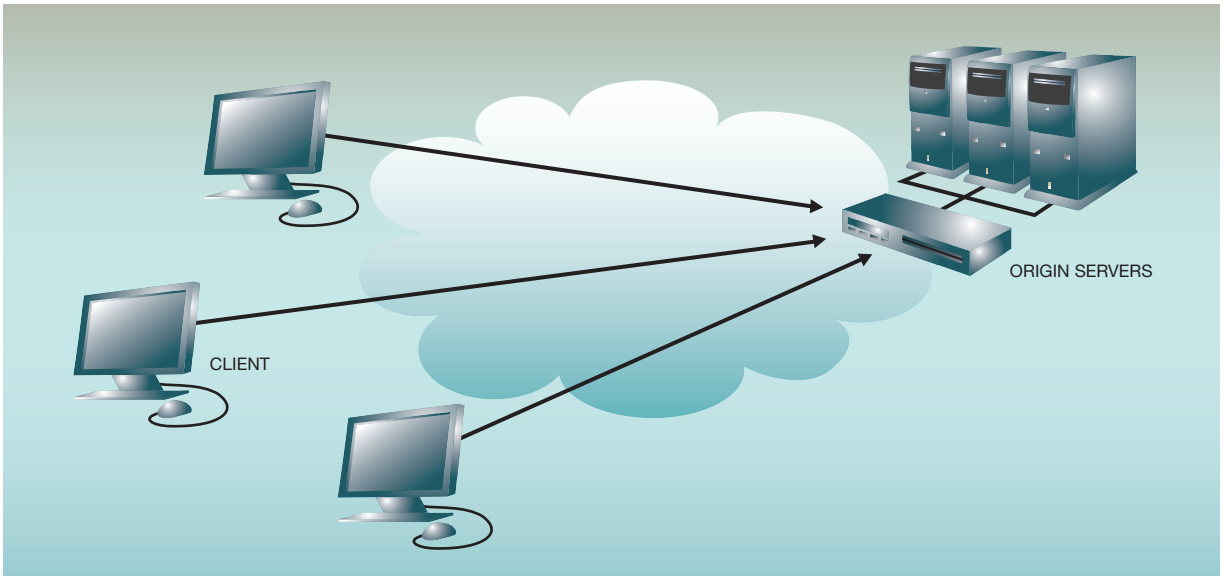
End users have long complained of excessive delays in accessing Web sites—primarily on the public Internet, but also on corporate intranets. Among the technologies developed to ease this problem is Web caching, where part of the content of a Web site is stored in locations that are closer or better connected to end users (from a network topology perspective). For example, corporate information technology (IT) staff might deploy Web caches at points where the corporate network connects to the public Internet. Internet service providers (ISPs) might deploy caches within their own network, so that subscribers can retrieve content from those caches instead of servers on other networks.

Web caching benefits different groups in the following different ways:

- End users benefit through quicker and more consistent response times because of reduced request latency and increased capacity to serve requests.
- Web site operators benefit from reduced server resource requirements because fewer client requests are reaching their Web servers. This benefit also reduces bandwidth usage at the Web site.
- ISPs that deploy Web caching benefit from reduced "upstream" bandwidth usage, since they can serve content to their subscribers on their own network rather than paying other ISPs to transfer those bits to their network.

In the late 1990s, several new companies began to offer Internet-based Web caching as a paid subscription service. These services came to be known as "content distribution" (CD) services because they focused on optimizing the delivery of Web site content to Internet end users. In the most typical business model, the content provider (i.e., Web site owner) pays the content distribution service provider (CDSP) usage-based charges for serving the content of its site from caching servers owned and operated by the CDSP. The CDSP deploys these servers in lo-

Figure 1 A centralized content-serving architecture



cations throughout the Internet, frequently inside ISP-owned networks. This overlay network of caching servers managed by a single provider is called a *content distribution network* (CDN). The key distinction between a CDN cache and an ISP-operated cache is that the former serves content only for certain content providers, namely CDN customers, whereas the latter caches content from all Web sites.

The main value proposition for CD services has shifted over time. Initially, the focus was on improving the end-user experience by decreasing response time, especially when the customer Web site met with unexpectedly high load. More recently, content providers have viewed CD services as a way to use a shared network infrastructure to handle their peak capacity requirements, thus allowing reduced investments in their own Web site infrastructure. Also appealing is the ability of a CDN to increase Web site security by placing a logical layer of trusted servers between the Web site and Internet end users. Finally, companies can use a CDSP to offer new functions, such as multimedia streaming, without new investment in the hardware or software for their own site.

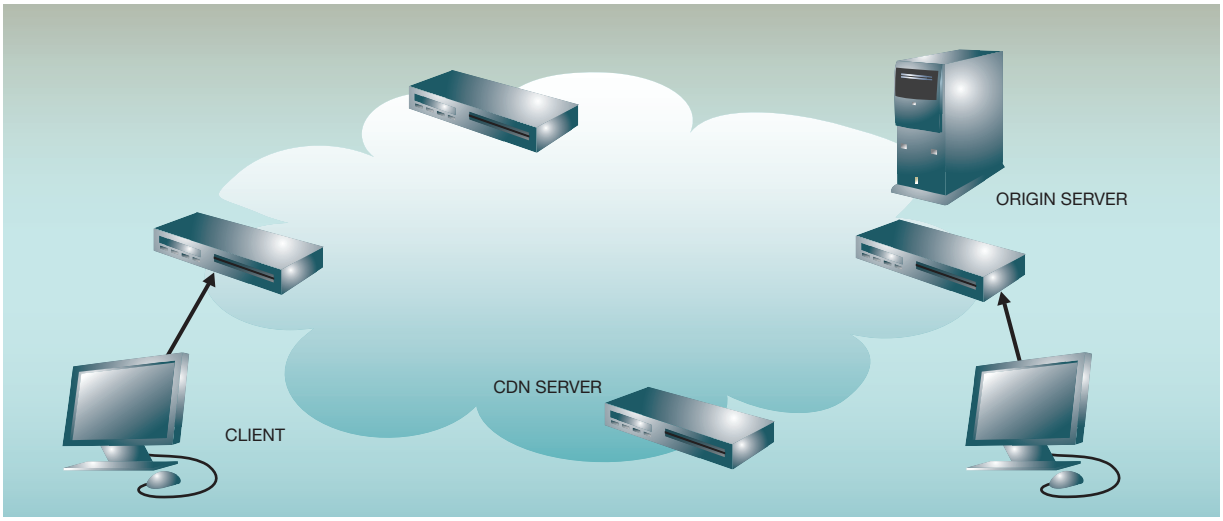
In this paper we describe the design and development of the *content-serving utility* (CSU), a utility computing service for content distribution. After an overview of content distribution technology, we detail many aspects of the CSU design, including the user- (i.e., customer-) facing functions that allow rapid service enablement and customer-controlled configuration and monitoring. We also discuss the CSU architecture and the specific features that enable improved end-user performance, Web site scalability, high availability, and security. Finally, we present performance measurements of the infrastructure and an overview of our capacity planning approach.

Content distribution technology

Before turning to the specifics of the CSU design, it is helpful to review CDN technology in general. The following is not a general discussion of Web caching, but rather its application in content distribution services.

CDN basics. In a traditional Web content-serving architecture, all clients request content from a single Web site location, as shown in Figure 1. In this approach, site capacity and performance are improved by adding servers and server off-load devices within the site, but client delays caused by network problems are not addressed. To handle surges in demand (for example, a seasonal shopping peak), many content providers overprovision their sites with respect to servers and bandwidth. Clients contact the Web site directly; thus security measures to protect

Figure 2 A distributed CDN architecture



the site from attacks are deployed within the site itself.

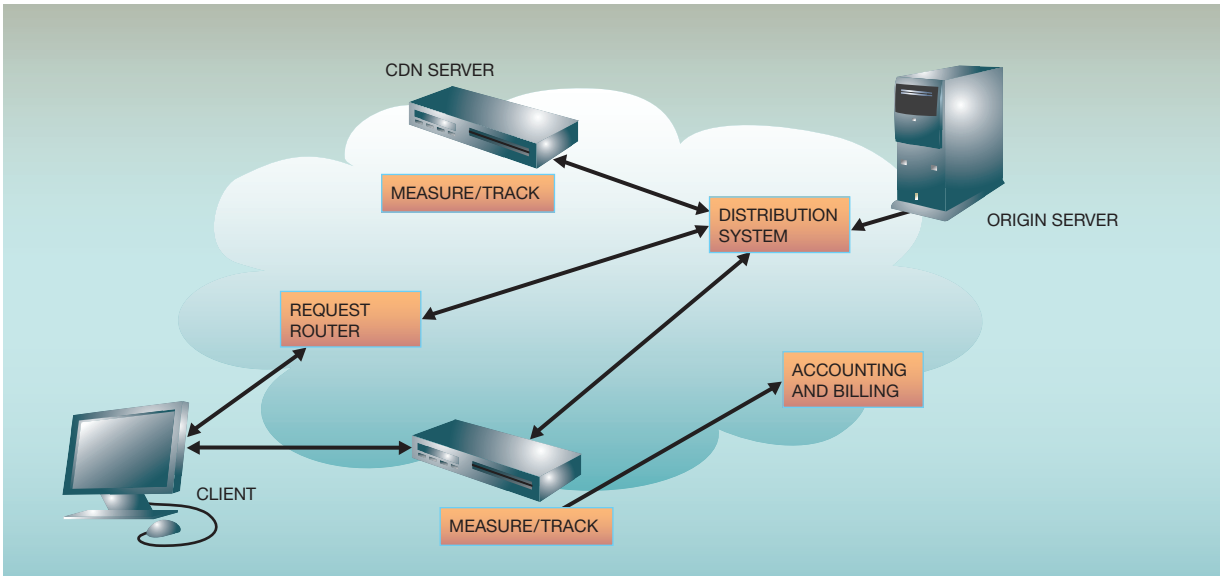
A CDN addresses poor performance caused by network congestion or flash crowds at servers by distributing popular content to servers or caches located closer to the edges of the network, as shown in Figure 2. (A flash crowd is a large number of clients simultaneously requesting content all within a short time.) A CDN is simply a network of servers or caches that delivers content to users on behalf of content providers. One of the objectives of a CDN is to serve content to a client from a CDN server such that the response-time performance is improved over contacting the origin server (i.e. server hosting the authoratative version of the content) directly. A CDN also improves the origin site scalability, because surges in demand are spread across the network of CDN servers instead of being served directly from the origin servers.

CDNs are deployed mainly on the public Internet and within corporate enterprise intranets. On the Internet, CDNs are typically built and operated by service providers offering a commercial service. The infrastructure is shared by many content providers, which amortizes costs and smooths traffic peaks by multiplexing requests for many sites. Internet CDSPs frequently use a combination of their own technology and off-the-shelf vendor components within their content delivery systems. They may place caching server clusters within a variety of ISP networks. If the CDSP is also an ISP, servers may be placed at its major peering points or access locations. Some Internet CDSPs provide corporate customers with an option to have caching servers placed just outside the Internet firewall at major corporate and business partner locations.

Within corporate intranets, a CDN is typically built and operated by corporate IT staff. Such a CDN can optimize employee access to Web-based enterprise applications such as streaming video for training. Several network equipment vendors offer product suites that enable an enterprise to build its own CDN. These packages typically include caching and load-balancing products, along with centralized management software. Enterprise CDNs can also be considerably simpler than Internet CDNs because they only have to service a single company and usually have less stringent reporting and network security requirements.

CDN architectural elements. As illustrated in Figure 3, CDNs have three key architectural elements in addition to the CDN servers themselves: a distribution system, an accounting/billing system, and a request-routing system. The distribution system is responsible for moving content from origin servers into CDN servers and ensuring data consistency. The accounting/billing system collects logs of client accesses and records CDN server usage according to

Figure 3 CDN architectural elements



the origin site for use in traffic reporting and usage-based billing. Finally, the request-routing system is responsible for directing client requests to appropriate CDN servers. In some cases (such as for very large objects), the request-routing system interacts with the distribution system to keep an up-to-date view of which content resides on which CDN servers.

In the following subsections we discuss the request-routing and distribution systems in greater detail before concluding the general CDN overview with a look at emerging CDN functions.

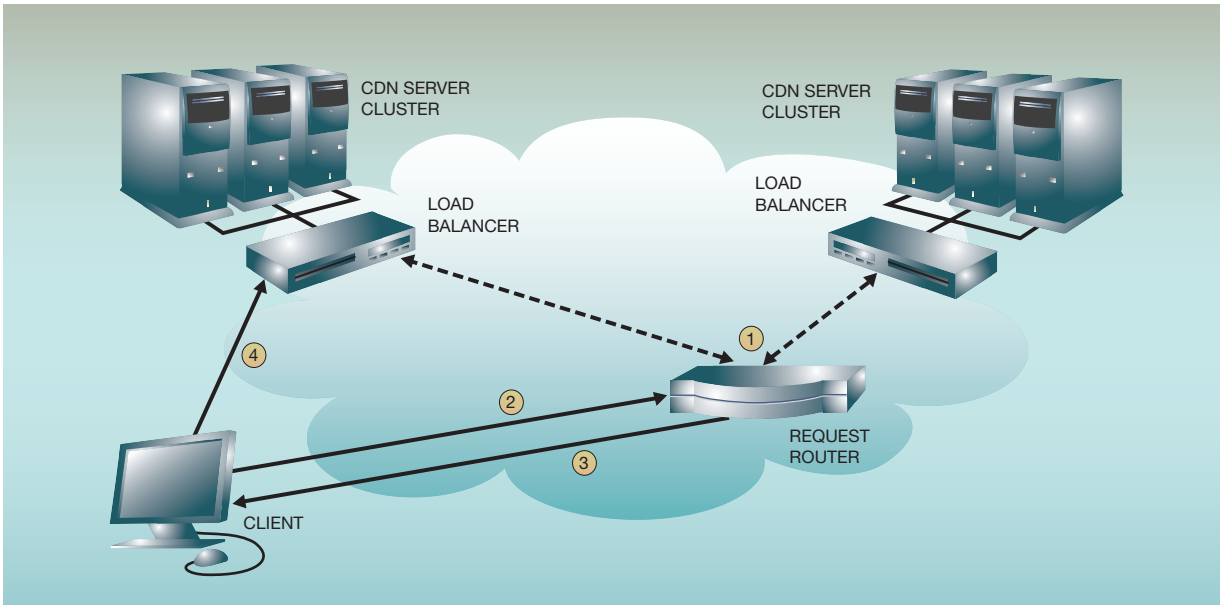
CDN request-routing. Figure 4 provides a simplified view of a typical CDN request-routing system. The objective of the system is to quickly route client requests to the best available CDN site (server cluster), and within that site to an optimal individual server. Before the arrival of any client requests, the request router regularly collects information about potential target sites and servers (step 1), including their availability and load. In some CDNs, such as that shown in Figure 4, the request-routing function is performed by a global request router that selects the target site and a local load balancer that selects the individual target server. In other CDNs, these two operations may be combined into a single system.

Clients access content from the CDN first by contacting a request router (step 2). The request router makes a site selection and possibly a physical server selection decision and returns a server assignment to the client (step 3). Finally, the client retrieves content from the specified CDN server (step 4).

Request-routing mechanics. Web request-routing techniques fall into three main categories: transport-layer methods, application-layer methods, and methods based on the Domain Name System (DNS).[1] Although the first two of these categories are often used within and between origin Web sites, DNS-based request-routing has emerged as the method most commonly used by CDSPs and global load-balancing equipment vendors. It is both simple—it requires no change to existing protocols—and general—it works with any application based on the Internet Protocol (IP) regardless of the transport-layer protocol being used.

With request-routing based on DNS, clients are directed to a CDN-controlled name server during the name resolution phase of Web access. Typically, the authoritative DNS server for the target domain or subdomain is controlled by the CDSP. In this scheme, a specialized DNS server receives name resolution requests, determines the location of the client, and returns the address of a nearby CDN server or a referral to another name server for further processing. The answer is often cached at the client side for a short time so that the request router can adapt

Figure 4 CDN request-routing



quickly to changes in network or server load. This caching is achieved by setting the associated time-to-live (TTL) field in the answer to a very small value (e.g., less than one minute).

DNS-based request-routing may be implemented with either full- or partial-site content delivery.[2] In full-site delivery, the content provider configures its DNS so that, for example, all requests for www.company.com are resolved to a CDN server, which then delivers all of the content. With partial-site delivery, the content provider modifies its content so that links to specific objects have host names in a domain for which the CDSP is authoritative. For example, links to http://www.company.com/image.gif are changed to http://cdsp.net/company.com/image.gif. In this way, the client retrieves the base HTML (HyperText Markup Language) page from the origin server but retrieves embedded images from CDN servers to improve performance. This type of uniform resource locator (URL) rewriting may also be done dynamically as the base page is retrieved, though this may increase client response time. The performance and effectiveness of DNS-based request-routing has been examined in a number of recent studies.[1–4]

CDN site and server selection. Clearly, the algorithms for selecting the target site and server have a direct impact on the performance of the CDN. Poor server selection decisions can defeat one of the key objectives of the CDN, namely to improve client response time over accessing the origin server. Thus, CDNs typically rely on a combination of static and dynamic information when choosing the best server. Several criteria are used in the server selection decision, including CDN site, server, and network conditions, client proximity to the candidate sites, and the content being requested.

It is critical to select a CDN site and server that are currently reachable, preferably a server that is lightly loaded. CDN request-routing systems determine site availability and load through regular status polling over the network. Because Web response time is heavily influenced by network conditions, it is also important to choose a CDN server that is "near" the client, where proximity is defined in terms of network topology, geographic distance, or network latency. Examples of proximity metrics include autonomous system hops or network hops, which are relatively static and simple to measure, but may not be good predictors of latency.[5,6] CDNs measure network latency in a variety of ways, including active polling of candidate servers with Internet Control Message Protocol (ICMP) echo or HyperText Transfer Protocol (HTTP) download requests, forwarding

a request simultaneously to multiple sites in a "race," or passively monitoring round-trip times between CDN sites and groups of clients. One final request-routing strategy is to direct the client to a CDN server that hosts the content being requested. This strategy is difficult with DNS-based request-routing, because the client DNS request contains only a server host name (e.g., www.company.com) and not the full HTTP URL.

It is unlikely that any one of these decision factors will be suitable in all cases. Most request routers use a combination of proximity and network or server load to make server selection decisions. For example, client proximity metrics can be used to assign a client to a "default" CDN server, which provides good performance most of the time. The selection can be temporarily changed if network conditions change or load monitoring indicates that the default server is overloaded.

Data distribution and consistency management for CDNs. There are two primary models for content distribution in CDNs: a "push" model, in which new content is moved from origin servers to CDN servers in advance of client requests, and a "pull" model, in which client requests cause CDN servers to retrieve content from origin servers. For most Web content such as HTML pages and static images, the pull model predominates. It requires little or no coordination with origin site content publishing and adjusts automatically to shifts in content popularity and audience location. For very large media objects, especially those for which a high demand can be anticipated, it makes sense to push the files in advance to CDN servers.

Regardless of the data distribution model, content consistency between CDN sites and the origin servers is essential. Content providers generally wish to avoid the situation where users in one location see one version of the content while users elsewhere see a different version. One key requirement of CDNs is to give the content provider control over freshness and ensure that all CDN sites are consistent. Some of the methods CDNs use to provide this control also extend to Web caches outside the CDN (for example, those operated by ISPs or enterprises on behalf of their end users), whereas some methods control only CDN caches.

Some of the methods CDNs use to provide content consistency and freshness are:

- Standard HTTP response header controls—The content provider configures its origin Web servers to provide instructions to caches about what content is cachable, how long different content is to be considered fresh, when to check back with the origin server for updated content, and so forth.
- CDN-specific cache policy distribution—Rather than configure content-caching policies at its origin servers, the content provider specifies them in a format unique to its CDN service provider or vendor. The CDN propagates the rule set to its caches, which then apply the rules.
- Vendor-specific cache heuristics—In cases where the content provider does not wish to develop complex cache policies for the content, some caching products have logic to "learn" over time how frequently different sets of content change at the origin server and tune their behavior accordingly.
- Active content pushing—The origin server publishes new content to a CDN control point, which pushes it into CDN servers. This is not frequently done except for cases where pre-positioning of specific content is worth the expense of bandwidth and administration.
- Active content invalidation—The content provider manually or programmatically initiates invalidation of specified sets of content. The CDN propagates the invalidation request to all its caches, and subsequent client requests force the cache to retrieve the fresh content from the origin server.

Of these mechanisms, HTTP response header controls are the most effective in ensuring cache content consistency because they have the greatest reach. CDN policy distribution may in some cases be simpler to administer but is limited in its effects. Cache heuristics are a good CDN feature for content providers who do not want to think about caching policies, but heuristics will not deliver the same results as well-planned policy controls. Active content pushing and invalidation are not generally applicable as steady-state control mechanisms, and they can cause control traffic to consume bandwidth and processor resources that could otherwise be used for serving content.

Emerging CDN functions. Content providers use CDNs primarily for serving static content such as images, and for streaming stored multimedia objects and live events. More recently, CDNs have begun to deliver additional application functions. Some examples of emerging features include:

- Dynamic content caching—CDNs can cache and serve dynamically generated content that an application server marks as cachable. Although this content is program-generated, it has value to more than one end user (for example, a sports score is reusable, whereas an individual stock portfolio listing is not).
- Page assembly—Web pages in markup languages such as HTML can be broken down into page fragments that are served independently by Web servers and assembled by a CDN into complete pages. Conditional page assembly instructions such as those defined by the Edge Side Include (ESI) language[7] allow different page fragments to be served to different end users based on request parameters and location.
- Localized content—CDNs can enable serving different content based on the location of the end user. The location can be inferred by the CDN site selected or more directly through (imperfect) geographical mapping of client IP addresses. [8]
- Application off-load—One eventual goal in CDN functionality is to off-load parts of server-based applications to the CDN. For example, IBM WebSphere* Application Server enables applications to be split into an "edge" component and a "server" component.[9] CDNs can serve as generalized platforms for hosting the edge components of applications that are divided in this way.
- CDN Internetworking (also called CDN peering)—Any single CDN is limited in its capacity and network coverage. Significant industry work has been done to allow CDNs to off-load work to each other on demand, and the Internet Engineering Task Force (IETF) has developed a general model for Content Delivery Internetworking.[10]

Customer view of content serving utility

Content distribution service customers experience little of the technology complexities just discussed. They purchase a service that provides certain functions and uses an infrastructure which is largely out of their direct control. Customers interact with the service in a limited number of ways, such as enabling their Web content to be served, viewing traffic reports, and receiving usage-based billing. The following topics summarize the customers' view of the CSU.

HTTP caching. The CSU delivers HTTP content from customers' Web sites to Internet end users with support for both full-site and partial-site content delivery. Customers are responsible for minor DNS configuration changes and any Web site content changes needed to enable their content to be served. No changes to customer origin-site hardware or software are required to enable the service. The entire process, including customer changes, can be completed within a few hours if necessary, but the normal enablement time is three to four days. Following the enablement of the HTTP caching function, customers simply experience a decrease in HTTP Web traffic to their origin site infrastructure, dependent on the percentage of cachable content served by the CSU, freshness time-out values, and other factors.

HTTPS caching. The CSU can also deliver Web content using HTTP over the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) Protocols, collectively termed HTTPS. As with HTTP, the CSU supports both the HTTPS delivery of an entire Web site or of selected objects. To help off-load CPU-intensive HTTPS server traffic, the CSU can cache and deliver common graphics that are embedded in an HTTPS Web page.

There is some additional complexity for enabling HTTPS because of the browser requirement that host names and host-installed digital certificates match. We provide common infrastructure host names that match certificates already installed on CSU-caching servers for partial-site caching. Customers also have the option of providing their own host names and host certificates.

Alternate site content. Although the infrastructure is highly available, a customer's origin Web site is still subject to outages. The CSU can monitor a customer's origin Web site for outages and serve alternate content to Internet end users in the event that the origin site is no longer accessible. To enable this function, the customer provides the alternate Web site content (which is relatively small and simple) to the CSU in advance, along with instructions formatted in XML (eXtensible Markup Language) for its use, on the customer's own Internet-accessible server. Once enabled, the alternate site content function runs automatically to detect site outages, serves the alternate content, and switches back to the origin site when it becomes available again.

Traffic reporting. With no infrastructure to look at and touch, customers rely on reports to know what a utility computing service is doing for them. The CSU reports show a number of statistics, including the number of end-user client requests, the number of cache hits and misses, and the number of bytes served. Report information is broken down by geo-

graphical region of the serving site, which corresponds roughly to client network location. For example, the user can see how much traffic was served from the eastern United States versus the western United States.

Response time reporting. In addition to seeing traffic levels, CDN customers often want to gauge the end-user response time benefits of using the service. This measurement is provided by third-party monitoring services such as Keynote Systems, Inc.[11] Gómez Performance Network,[12] or a service from the IBM Enterprise Monitoring Solutions Lab (EMSL). These services can regularly monitor the response time to retrieve specified Web objects from several probes placed around the Internet, both through the CSU and directly from the origin server.

Content management. At times, it may be necessary to force cached content out of the CSU infrastructure and trigger a "refresh" from origin servers, for example, if content is published in error with a long expiration time. Authorized users can use an HTML form to specify a set of objects to be deleted and initiate the invalidation action. The CSU then carries out the action, typically within several minutes.

Log delivery. Many companies analyze Web server logs to gain business insight into their end-users' browsing behavior. When using a CDN, many Web requests are handled directly without involving the Web servers of the origin site, and information about these requests does not appear in the Web server logs. The CSU has the ability to retain log data from its own caching servers and transfer it to a customer periodically in one of several standard log formats. The customer may then combine this log data with its own origin site data before processing by using Web analytics software or a Web analytics service (e.g., IBM SurfAid*).

Note that even this form of combined log analysis does not capture end-user requests served from caching servers that belong to ISPs and enterprises, which are outside the control of a CDN. More accurate statistics can be gathered by counting references to uncachable objects (placed on each Web page) because requests for uncachable objects must flow to the origin server (and cannot be served from any cache).

Service level agreements. The CSU aims to provide a service level agreement (SLA) that ensures 100 percent availability; that is, content will always be available through the CSU if it is available from a customer's origin servers. In the extremely unlikely event that all CSU sites become unavailable simultaneously, this SLA would be violated. For end-user performance, the service has a service level objective (SLO) that the average response time to retrieve a reference object from multiple locations through the CSU will be less than retrieving it from the customer's origin server. This is achieved based on our experience that most of the probes are likely to be closer (topologically) to a CSU caching location than to the origin server.

Usage-based billing. The CSU is designed to charge customers on the basis of their use of *virtual server units* (VSUs). A virtual server unit corresponds roughly to an amount of HTTP request traffic that a small Web server can handle in one hour. Our on demand pricing model allows customers a number of options for fixed or variable billing. A customer can pay at a set monthly rate for a predicted base level of traffic. If the customer's traffic exceeds that base level, the customer pays for that excess traffic at a higher rate, but only for the hours in which the traffic peaks occurred. The customer could also set the predicted base level to zero, yielding a 100 percent usage-based bill. We also provide the option to pay in advance at a nonpeak rate for a number of hours in which customer traffic may exceed the base level without peak charges. This feature provides the customer with protection against unexpected peak charges.

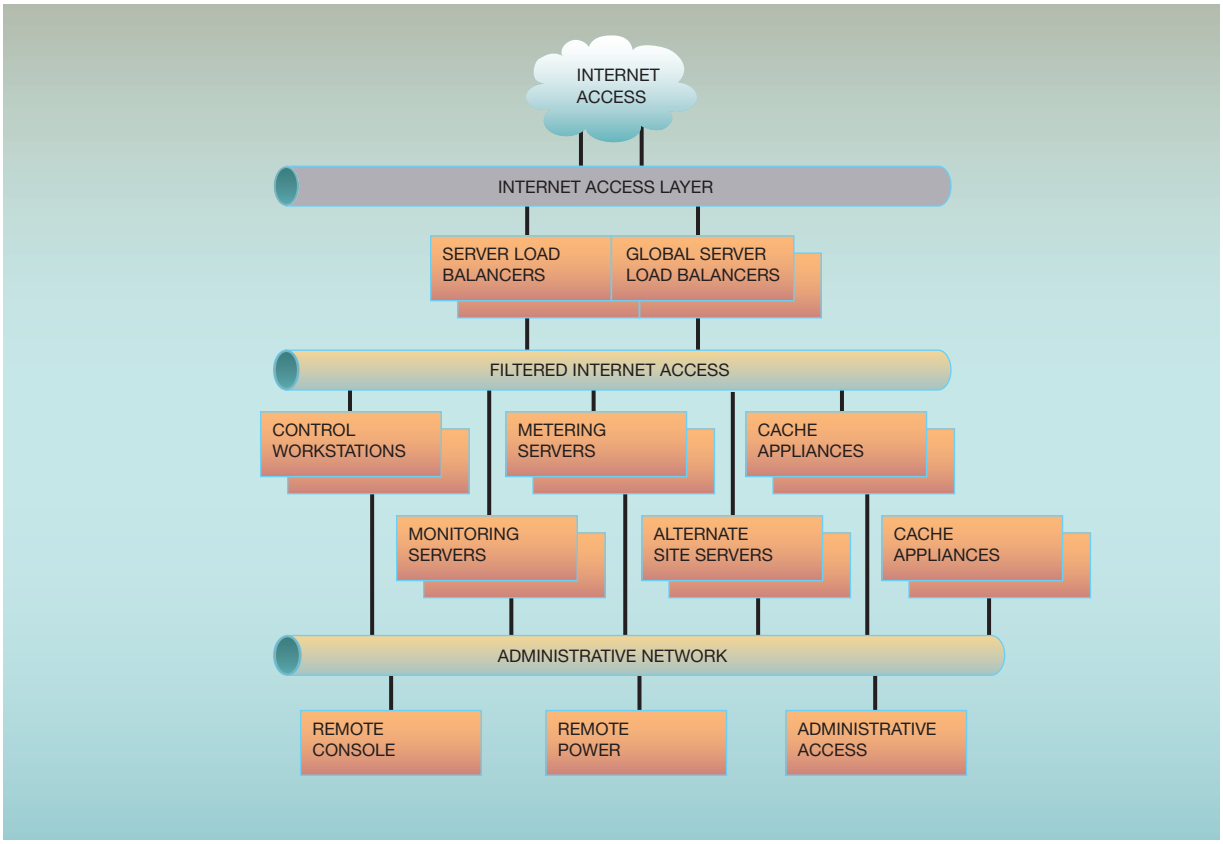
Architecture and implementation

In this section we discuss the architecture of the service and its implementation, which includes the infrastructure, service functions, security, infrastructure management, and service provisioning.

Physical infrastructure. The physical infrastructure of the content serving utility is built around four key components: network connectivity, global and server load balancing, caching, and management infrastructure. These components are collected into a logical entity we call a caching PoD (point of distribution). For purposes of redundancy and performance for end users, multiple PoDs are used. Each PoD also has internal redundancies for high availability. Figure 5 shows the logical components for each PoD.

Within each PoD, the physical architecture begins with dual ISP connectivity. Each ISP is connected to a separate router, with failover configured between

Figure 5 Logical components of a caching PoD



them. Those routers are in turn connected to the main PoD switches. Logical infrastructure elements include the switches to establish base connectivity, the global load-balancing devices to handle wide-area request-routing, and the server load-balancing devices to balance traffic across local PoD servers. In practice, all three of these components were delivered using a vendor's single device, configured in a redundant pair. This consolidation simplified administration and reduced points of failure within the PoD design.

Sitting logically behind the load-balancing switches are the caching devices. These devices are vendor appliances dedicated specifically to serving cached content. Surrounding these devices is the management infrastructure of the PoD, consisting of IBM xSeries* servers running the Linux** operating system. The servers perform management functions, including monitoring, log collection and distribution, software distribution, serving of alternate site content, and distributed file system access.

Our design criteria identified these required functions as well as the capacity requirements of our PoDs. We defined two PoD sizes, a large PoD with the capacity of 1.2 Gbps and a small PoD with the capacity of 300 Mbps. These sizes were based on available bandwidth in PoD locations, that is, the 1.2 Gbps solution was sized on dual ISP (655 Mbps each) OC-12 (i.e., Optical Carrier level 12) connections, and the 300 Mbps solution was sized on dual ISP (155 Mbps each) OC-3 connections. Our design criteria included the ability to swap components in and out of the architecture based on growth, product end-of-life, or future requirements. Because of redundancy at the switch, caching, and server layers, a single

switch failure only reduces the serving capacity of a PoD by 50 percent.

We deployed management servers within each PoD in redundant pairs. The main components of the management infrastructure are listed below.

- Control workstations: These workstations are the main administrative servers. They provide remote ssh (secure shell) access for PoD administration, DNS service for all PoD devices, RADIUS (Remote Authentication Dial In User Service) authentication for the switch/load balancing and caching devices, NTP (Network Time Protocol) for time synchronization, AFS (Andrew File System) shared file systems and authentication for administrative access, and secure loggers for all devices and servers within the PoD.
- Metering servers: These servers are responsible for collecting the usage logs from all cache appliances and sending these logs to central distribution points before delivery to a customer. Additionally, these servers parse the caching appliance logs and insert summary data into metering and reporting databases every six minutes. This provides a near real-time view of log data.
- Monitoring servers: These servers are responsible for monitoring and sending alerts for all elements of the PoD. They also provide real-time health views of critical devices and applications and long-term trending of resources. Additionally, they serve as external DNS servers. Though the global load balancers handle all DNS requests, they rely on standard DNS servers to obtain the original DNS data. Monitoring servers also provide the common control point for administration of all network and cache appliances.
- Alternate site servers: These servers host a customer's alternate site content if the customer has chosen the alternate site service. They routinely query a customer's servers for the alternate site content XML file and populate the alternate site Web servers with the customer's content.
- Out-of-band management devices: These devices include the xSeries out-of-band adapters, remote console devices, remote power control units, and remote modem access in case of complete ISP access failure.

Request-routing. The CSU uses DNS-based global server load balancing (GSLB) to route end-user requests to PoD sites. The GSLB system routes requests to the best responding site, while shifting traffic away from sites that are unreachable or near capacity. The CSU functions as an autonomic, self-healing system; if some sites are unusable, it moves traffic to the others with no manual intervention. Within a PoD site, requests also flow around failed individual components.

To choose the target PoD for an end-user request, the GSLB system evaluates specific metrics reflecting the state of the CSU infrastructure and network conditions between the end user's assumed location (based on network address range of his or her DNS server) and the CSU sites. It checks the following policies until one test determines the site that is best for the end-user request:

- Are the virtual IP (VIP) addresses that represent the target customer Web site available and responding within defined latency thresholds?
- Are the session capacities and new connection arrival rates of the target PoD sites within tolerance?
- Is the round-trip time for connection establishment from the user's network within tolerance?
- Is the end user located on the same continent as the target PoD?
- Is one of the target sites preferred, according to administrative settings?
- Which PoD site has the fastest response time to the GSLB system?

The domain name servers for the CSU are the request-routing devices themselves. The infrastructure has four sites with GSLB servers. Usually, the GSLB server that is closest by way of natural IP routing is the GSLB server that responds to the end user first. A few examples given below can offer a good understanding of how the GSLB servers direct end-user requests to the best site.

Scenario 1—A DNS request comes in from an end user whose network address range has not yet been served by the GSLB infrastructure. In this scenario, the GSLB does not know the round-trip time to the end user's network from any of the serving PoDs. If it is assumed that all candidate target PoDs are available and have capacity, the GSLB system checks for continent location and administrative preference (a value used by operations staff to manually prefer or avoid a site). If there is still more than one candidate target PoD, the GSLB system selects the PoD with the fastest overall response time to health check requests and returns its IP address to the end user's DNS server.

Scenario 2—A DNS request arrives from an end user whose network range has already been served by the GSLB infrastructure. If it is assumed that all candidate target PoDs are available and have capacity, the GSLB system routes the request to the PoD with the lowest previously recorded round-trip times to the end-user network. In order to continuously learn whether a different PoD might provide even better response (because of changing network conditions), the GSLB system routes a small percentage of requests from the network to the second-best responding PoD.

Scenario 3—A DNS request comes in from an end user whose network range has already been served, but what was once the best site is now heavily utilized. The GSLB system checks the availability values and knows all sites are responding within thresholds. It then checks the capacity values and identifies one site as heavily utilized. The GSLB system eliminates this site from further consideration and selects from among the remaining sites on the basis of historical round-trip times to the end user's network.

In addition to sending the user to the best-responding site, another key feature of the request-routing mechanism is the ability to automatically bypass a failed site. Bypassing occurs in two ways. First, each PoD is in constant communication with the GSLB servers, giving information about its health and current load. Any failure to reach a PoD is rapidly detected by the GSLB servers, which then prevent any new requests from being sent to the questionable site until connectivity is re-established. Also, because the GSLB system performs health checks on the individual VIPs in each PoD, a failure to reach a specific VIP at one of the PoD sites will remove that PoD from the candidate list for that specific VIP.

Another concern arises from users who have already completed their DNS resolution and are tied to a PoD that then fails. In this case, we rely on the underlying capabilities of the client's Web browser. On failure to receive content from an IP address that had previously been responding, typical browser behavior is to send another DNS request to resolve the Web site name. By requesting the IP address for the Web site, the existing client is treated as a new client, triggering its re-evaluation against the GSLB policies and assignment to a new PoD. Typically, this entire interaction occurs without the end user being aware of the change. In the worst case, the end user experiences a broken image or a single failed connection and, on retry, is successfully able to retrieve the entire Web site. Our experience from running event Web sites on the CSU infrastructure is that very few clients are aware of movement to new PoDs, and complaints stemming from this movement are indistinguishable from normal Internet connectivity complaints. In practice, we have observed a site failure, subsequent movement of its existing traffic to other sites, and automatic site recovery in 41 seconds (for the 2003 Australian Open tennis event).

Caching. The caching function in the CSU infrastructure is provided by vendor-supplied caching appliances—dedicated-function products consisting of both hardware and software. Large and small caching appliances provide a range of throughput (measured in hundreds of Mbps) and new HTTPS connection capacity (measured in hundreds of SSL key negotiations per second). We deployed several small caching appliances in each small PoD, where lower overall capacity was needed, and deployed several large caching appliances in each large PoD.

The CSU caching system supports three of the consistency management methods previously described for CDNs in general: standard HTTP response header controls, active content invalidation, and vendor-specific cache heuristics. The caching appliance provides the base support for managing content freshness, and the CSU infrastructure provides a necessary supporting role. For example, supporting time-of-day-based response headers requires accurate time synchronization, via the Network Time Protocol (NTP), throughout the CSU infrastructure. Content invalidation requires a facility for propagating invalidated messages from the user interface application to all caching appliances in all PoDs.

If a CSU customer chooses not to set expiration policies on its Web site content, software in the caching appliances applies heuristics to manage content freshness. For example, the cache may examine information from the origin servers such as the date and time of last change to a piece of content, or it may track how frequently the content is requested. For example, a common heuristic in such cases is to set the document expiration to a small percentage (e.g., 10 percent) of the elapsed time between the last document modification time and the current time. The cache can check its estimate of expiration time for the content with the origin servers periodically to determine whether it needs to be refreshed.

Security. The CSU infrastructure was designed with the following security goals in mind:

- Protect customer Web content on the infrastructure from unauthorized changes.
- Ensure the integrity of usage information for billing.
- Protect the CSU infrastructure itself from attacks that might disrupt availability.
- Help protect customer origin servers from some attacks, such as denial of service.

Rather than rely on an "eggshell" defense, with protection only from an outer layer of firewalls, security was built into each component of the infrastructure. The network layer provides initial protection from unauthorized access and common denial of service attacks. Firewalls, intrusion detection, encryption of administrative flows, and automated notifications provide additional security at every level of the technical implementation. A number of automated procedures and audit points protect usage data from tampering as it is collected and consolidated for usage billing purposes.

Because of its distributed nature, the CSU can protect origin server sites from a number of attacks. Recall that for the portion of customer content handled by the CSU, all end-user requests flow first to CSU PoDs. The only requests for this content that reach the origin servers are from CSU caching servers for objects that are expired or not in cache. Typical attacks or scans from the Internet are filtered out at the CSU PoDs. Denial of service attacks, in which the attacker tries to overwhelm the Web site with a high volume of requests, can in some cases simply be absorbed by the high capacity of the CSU, or they can be deflected at its networking layer. More sophisticated denial of service attacks can be localized to a specific PoD by using global load balancing, leaving the others unaffected and able to serve the customer Web site.

Alternate site content. To serve alternate customer Web site content when the customer origin site is unavailable, the CSU infrastructure includes HTTP servers (separate from caching servers) in a number of the PoDs. These servers must be loaded with the alternate content at all times, so that they are ready to serve it when needed. As previously mentioned, the customer supplies XML-formatted instructions specifying where the alternate content can be obtained. A small Java** application runs at the customer-specified frequency to retrieve that content from specified customer locations. A publishing application then "pushes" the content to the HTTP servers located in the PoDs.

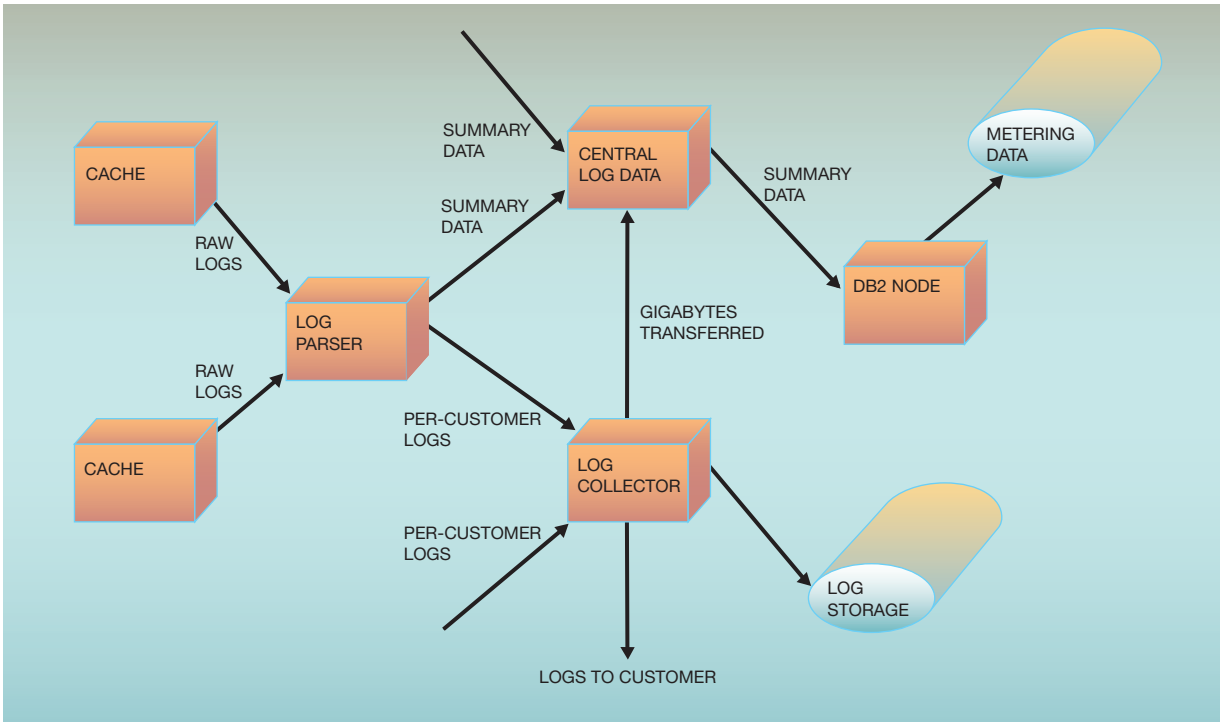
If the origin server persistently fails to respond to health checks or HTTP requests, failover to the alternate site content is triggered. The failover is effected by having the CSU caching servers begin to pull site content from the CSU HTTP servers, rather than from customer origin site HTTP servers. CSU caching servers continue to serve content to end users. The failure threshold is set low enough to prevent most end users from experiencing site failure, yet not so low that every momentary Internet or origin server glitch causes a flip to alternate content. Generally we seek to restrict the potential failure window to less than a couple of minutes, though it varies with the volume of traffic to the site. Heavily trafficked sites might move in only a few seconds, whereas more lightly visited ones might be switched only by health checks after several minutes of consecutive origin server failure.

After the switch to alternate content has been made, it is important to continue monitoring the origin server to determine when to switch back to the original content again. In addition to continued health-checking, we also use a default expiration time of five minutes for all alternate site content. Health checks ensure that the cache will revert to the origin server content as soon as it is available again, and short expiration times ensure that end users will be able to take advantage of the return of the origin server within only a few minutes.

Metering and log collection. The CSU needs to collect accurate information about customer traffic data served, for both near real-time statistical reporting and monthly usage-based billing. The source for all this information is the server logs maintained in each caching appliance. The CSU includes a sophisticated system that collects, processes, summarizes, reformats, and stores this log information. This facility also supports the delivery of raw log data files to those customers who want to perform their own statistical analysis (in combination with origin Web server logs).

The caches supply the raw logging data in a custom format to the log processing system. The logs are transferred from the cache device on six-minute intervals, and then they are subsequently processed by a log parser that can process millions of hits per minute across the physical locations. The parser gathers statistics for each customer's individual Web sites

Figure 6 Metering and log data flow



and collects the data into five-minute interval "buckets." At the end of the interval, all of the gathered data are submitted to both a primary and secondary aggregator as an XML document. Local disk queuing of interval data is supported in case one or both of the aggregators cannot be reached.

As part of this highly available design, the parsers can be configured to send the interval data to any number of aggregators. Data are then placed in multiple databases, and at the end of the month, a set of reconciliation scripts are executed to report any discrepancies. In practice, with over a year of database inserts across two aggregation systems, there have been no discrepancies.

Customers who request raw log delivery specify a preferred format, delivery interval, and protocol. After the logging data have been parsed, the data are formatted into the customer's desired format and stored on the local disk of the parsing node. At certain intervals, log collection clients execute and transfer the individual customer logs to a central location over HTTPS. There, the log files are gathered into a compressed collection and subsequently sent to the customer at the requested interval using SCP (Secure Copy), SFTP (ssh File Transfer Protocol), or FTP.

Figure 6 illustrates the data flow from the cache devices, through the log parser, to the log aggregator, and on to the metering database. Note that the parser communicates with multiple central log aggregators, each of which connects to its own IBM DB2 Universal Database* copy. At the log parser node, a parser, a formatter, and a log collection client run. A log collection server and a log delivery application execute at a central log collection node.

Traffic reporting and content invalidation. To support the requirement for customer traffic reporting, the CSU includes an Internet-connected Web site for access by customer personnel. This Web site is centrally deployed along with connectivity to the central log data DB2 databases. Customer personnel normally reach the CSU reporting Web site through the e-business Hosting* Connection portal, which provides a common launch point to both CSU traffic reports and the response time reporting functions men-

tioned below. Customer personnel administer their own user IDs through IBM's Web Identity platform, but administrative linkages are required with portal and service authorization databases.

The CSU reporting Web site provides standard and custom reports of customer traffic statistics on demand, with data normally available from the metering system within 10 minutes of the traffic being served by the CSU caching servers. Users can also register for regular e-mail delivery of standard reports. The reporting Web site is implemented with WebSphere Application Server using JSP** (JavaServer Pages**) and servlets, with JDBC** (Java Database Connectivity) connections to the DB2 metering data server.

To support the requirement for active content invalidation, this Web site also provides an HTML form where authorized users can specify a set of content and submit a request to invalidate. The presentation server validates the request itself, then submits it to a content invalidation system. This system distributes the request to each PoD location, where an invalidation agent submits cache vendor-specific requests to the appropriate caching devices.

Response time and availability monitoring. The CSU service requirements for a 100-percent availability SLA and a response-time-measurement option both require Internet-based probing of customer content. Rather than build this capability into the CSU itself, the development team leveraged existing capabilities of the IBM Global Services (IGS) Enterprise Monitoring Solutions Lab (EMSL). EMSL has a number of probe servers installed on the Internet and provides end-to-end availability and response time monitoring services for a variety of transaction types (e.g., Web access, e-mail, file transfer).

CSU response time measurements required no new EMSL function. As part of CSU enablement, a customer provides the name of an object on the customer's Web site to be monitored. EMSL is configured to regularly retrieve that object from several different probe sites, both through the CSU and directly from the customer's origin server. Web-based EMSL reports show the customer the relative response times achieved, over time.

CSU availability monitoring did require new functionality, however, because the SLA defined an outage as the object not being available through CSU, *whereas it was still available from the customer's origin site.* In other words, failure of the customer's origin site would not result in CSU SLA noncompliance. To perform this monitoring, EMSL probe logic was modified from simply alerting appropriate personnel of a single site outage, to alerting them when CSU had an outage and the origin site did not. EMSL continued to use Tivoli Event Correlation (TEC) to correlate these alert events from multiple probe sites and to notify the appropriate personnel of an outage. In any real CSU outage, however, the CSU operations team would already have been notified of any infrastructure problems through its own management system. The EMSL system is used primarily for formal SLA tracking and adjudication of any outage reports.

The use of EMSL by CSU is an example of one utility service building on another rather than duplicating an existing capability. As the IGS portfolio of on demand services increases, one architectural goal is to facilitate this type of service bundling and reuse through programmatic provisioning interfaces.

Infrastructure management and monitoring. The CSU was designed to be managed remotely from anywhere by a central operations team. All customer boarding information resides in a configuration file. Boarding refers to the process of configuring the shared infrastructure to handle the Web content of a new customer; that is, the service is provisioned for the customer. This file is used to identify all values required to board a customer and is defined from the initial customer engagement process. All tools required to configure and manage utility computing use this file to generate the incremental configurations required to configure the caching, load-balancing, metering, monitoring, and reporting elements.

There are primarily two monitoring views: a customer view and an operations view. Whereas the customer view provides data unique to an individual customer, the operations view displays the health of the whole infrastructure, including historical trends for capacity planning. Each PoD monitors its own systems in addition to cross-monitoring all other PoDs. Individual PoDs report their health to two central PoDs where the results are displayed and alerts are generated in real time. This provides a fully redundant operations monitoring solution.

The operations view monitors all elements of the infrastructure. Data that are less time-sensitive are monitored every few minutes, whereas more time-sensitive data are monitored every minute. Data with

relaxed time constraints primarily consist of the status of supporting software. Examples of these include the authentication servers, network time servers, DNS servers, out-of-band devices, and the normal operating system monitors such as CPU, disk, and memory usage of all support servers. Time-sensitive data are monitored every minute for purposes of alerting and thresholding and are displayed in real time. Examples of the real-time data collected include cache, CPU, and memory utilization, concurrent sessions, new sessions per minute, bytes in and out of the infrastructure, traffic distribution among sites, requests per second, cache hits and misses per second, and thread utilization of all caching devices.

Service provisioning. Although request-routing, caching, security, and management systems of the CSU are quite complex in their function, the implementation and boarding of a new customer on the utility is very simple and has been completed in less than one hour. The customer simply identifies the origin Web server host name where the caches will retrieve content, and updates its DNS with a CNAME pointing to the utility-hosted host name. Requirements are defined in a customer repository file, from which all boarding tools gain the information required to board a customer. Once properly defined, the boarding tools perform the following tasks:

- Configure PoD caches with the customer boarding details. These details identify the customer Web site URL, the back-end servers from which to retrieve content, and the alternate site servers if the customer has chosen the alternate site service.
- Determine the IP address to assign to the customer URL and update the external DNS servers with this information.
- Configure the server load balancers to allow only the specified protocols (either HTTP or HTTPS or both) to the customer caches.
- Configure the global load balancers with the boarded customer URLs.
- Configure the metering and log delivery elements.
- Configure the monitoring elements to ensure 100 percent availability.
- Test and verify customer URL boarding functionality.
- Notify the customer that it is boarded and ready for DNS to be pointed to the utility.

We configure all caches in a PoD to serve customer requests but send customer requests only to a pair of caches unless the customer traffic is expected to exceed 100 000 requests per minute. In this way only two caches per PoD will have to retrieve content, minimizing cache misses and requests to the origin Web servers. In the case where a site originally configured for a lower traffic level starts to exceed the threshold, we simply update the configuration of the server load balancer to route the requests to additional pairs of caches.

Billing interface. As mentioned previously, charges to CSU customers are based on usage measured in VSUs, which in turn are based on HTTP/HTTPS request rates. One purpose of the metering function previously described is to collect the raw per-customer statistics required to calculate the number of VSUs used. At the end of the billing month, a CSU-specific accounting application extracts each customer's traffic statistics from the centralized metering database, compares them with the customer's ordered traffic levels, and calculates the number of VSUs to be billed. The accounting application then passes its results to a general-purpose billing engine (shared by many services) using a standard batch interface.

The billing platform handles the rating of the customer's usage, integrating it with any fixed monthly or one-time charges and generating invoices. Functions such as electronic bill presentment and payment handling are also standard and did not require custom development for the CSU.

Relationship to the Universal Management Infrastructure. IBM Global Services is actively working on a framework of general-purpose utility functions to facilitate the rapid development of on demand services such as the CSU. Some of these functions include: a common metering function and data store, a common reporting engine, and a common billing system. These functions are collectively called the Universal Management Infrastructure (UMI) and are being developed and deployed in stages. Internal analysis of the effort required to develop these systems suggests that UMI should significantly reduce the development effort and expense of future on demand services.

Performance and capacity considerations

End-to-end Web response time performance is affected by numerous factors such as client and server network connectivity, network loss and delay, server load, HTTP version, and DNS resolution time. The content-serving architecture has a significant positive impact on some of these factors, as well as on important aspects unrelated to performance, such

as cost, availability, and ease of management. In this section we describe some of the performance considerations in designing and deploying the CSU.

Architecture and capacity. In a traditional centralized content-serving architecture, performance and scalability are typically improved by adding servers, or server-side caches, to off-load some operations from the servers such as static content serving or SSL processing. This approach does not provide the ability to address poor performance caused by problems in the network, however, and can be expensive because the site must often be overprovisioned to handle unexpected surges in demand. At the other end of the spectrum is edge caching provided by CDSPs who deploy hundreds or even thousands of caches in a large number of ISP networks. Although this approach offers very good response time performance by locating CDN servers as close to clients as possible, it suffers from high infrastructure costs and management complexity.

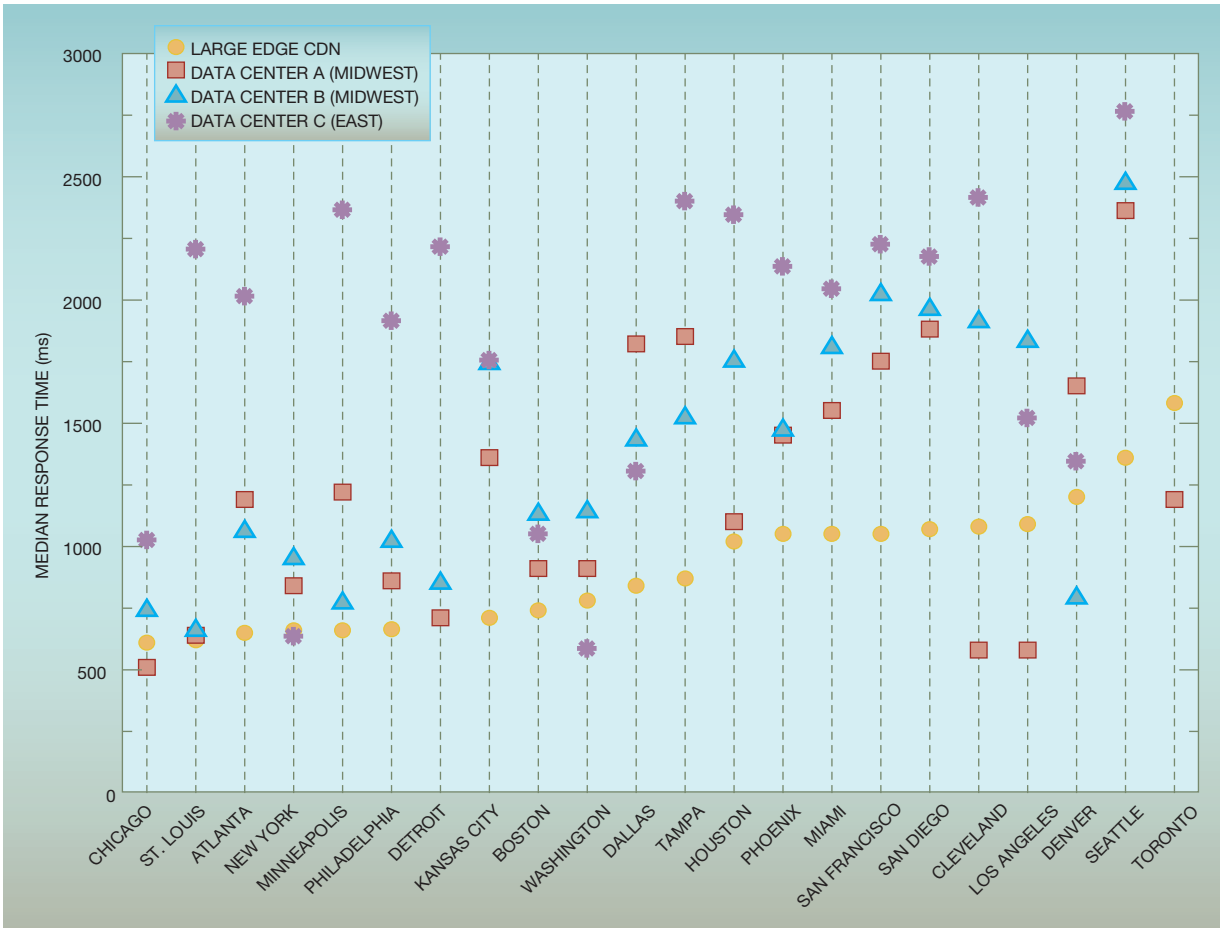
In designing the CSU, the approach taken was to deploy a *regional* caching utility with the ability to increase performance and availability with more moderate infrastructure costs. Hence, an early requirement was to examine the performance implications of a regional versus edge solution. As part of a larger performance study conducted in late 2001, we deployed a number of cache appliances with identical content at several commercial data center locations. We also instrumented the content to be served from a large commercial edge CDN. Our test content was the top-level page from the 2001 U.S. Open tennis tournament site, consisting of about 29 embedded objects, totaling roughly 104 KB. We collected performance measurements using Keynote agents[11] distributed around the world and connected to a variety of ISPs. (In this paper, we focus primarily on data collected by agents located in North America.) The Keynote agents used in our study are well-connected and thus are more representative of enterprise or university clients, rather than consumer clients using dial-up or broadband access.

Figure 7 summarizes the measurement results, comparing the median response time performance from several individual hosting centers with the large edge CDN from Keynote agents located in a number of metropolitan areas. The graph suggests that even a single regional location can deliver good performance to a number of clients, in some cases comparable to or better than a very-large-footprint CDN. For example, data center A, located in the Midwest, is able to deliver very good performance to clients in much of the region (including Chicago, Detroit, Cleveland, St. Louis, and Minneapolis). These results confirm our view that a well-connected regional data center can provide competitive performance with a CDN for a large set of clients.

Though our focus was initially on North American clients, we also collected performance measurements for clients connecting from international locations in Europe and Asia-Pacific. These results (not shown) make it clear that placing content-serving locations only in North America is insufficient to provide good performance to most international clients. The transoceanic network links introduce high latency that results in poor response time. A similar strategy of deploying a few well-connected content-serving locations adequately serves international clients.

The experimental validation of the regional caching approach was a critical part of the CSU design phase; however, it was not feasible to conduct large-scale experiments for every data center. Yet, it was still necessary to gain some insight into the performance and availability improvement as a function of the number and location of CSU sites. For this task, we created a parameterized planning tool that was simple but useful for exploring options. The tool was based on a model in which clients were divided into a small number of regions (e.g., New England, the West Coast, and the South) and a latency estimate was assigned for communication between each client region and a number of candidate CSU sites. These latency estimates were based on experiments in which we measured inter-region backbone network delays using a set of well-connected probe points. We also included a parameter to assign a weight to a client region in order to reflect relative importance (e.g., on the basis of the level of traffic expected). Using this model, we could then estimate the average delay over all client regions when deploying a given number of sites from the candidate set, as well as the expected performance improvement when growing the deployment. Since these delays reflected backbone latency, we extended the model to include delays from a tunable set of client access technologies (e.g., 40 percent dial-up and 20 percent broadband). Finally, the tool also included an analysis of application-level response time using the delays above in conjunction with a model of Web response time.[13] With this tool, we were able to analyze the resultant performance from a variety of deployment scenarios and reinforce our initial design

Figure 7 Performance of a large edge CDN versus caches deployed in individual data centers



choices. The planning tool was useful to illustrate the advantages of the regional caching approach, namely, realization of the majority of performance benefits of edge caching at a reduced cost.

CSU production environment performance. The CSU has handled a number of high-volume special-event Web sites since its deployment in mid-2002. One of the early events for which we collected a large amount of performance measurement data was the Wimbledon 2002 tennis tournament. We collected data similar to that described above, using Keynote agents to download and measure the performance of the top-level page (approximately 35 objects, 110 KB total).

In Figure 8A and 8B we show the response time performance categorized by ISP and metropolitan area respectively. In addition to the median response time, the graph also gives an idea of the variability with bars indicating the 10th and 90th percentiles of the observed measurements. Note that we excluded erroneous measurements in which one or more page elements were not downloaded correctly. We also limited the presented results to those metropolitan regions and ISPs with two or more measurement agents to avoid biases caused by one very high- or low-performing agent. From the graphs we can see fairly consistent median response time between 1.5 and 2 seconds across ISPs, as well as across metropolitan regions. The variability exhibits wider differ-

Figure 8A Response time performance by client ISP

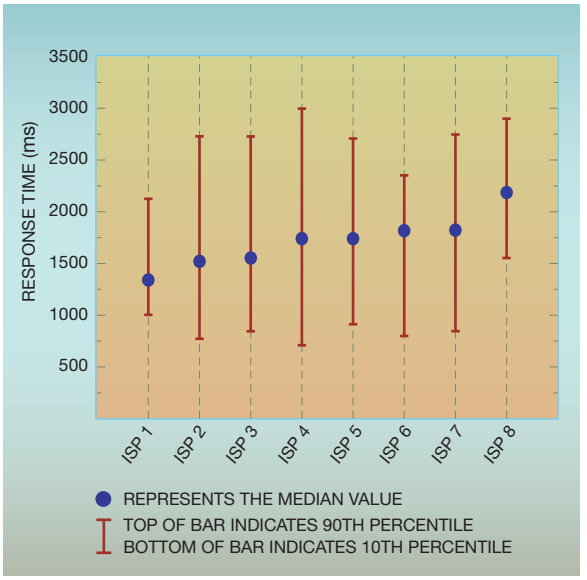
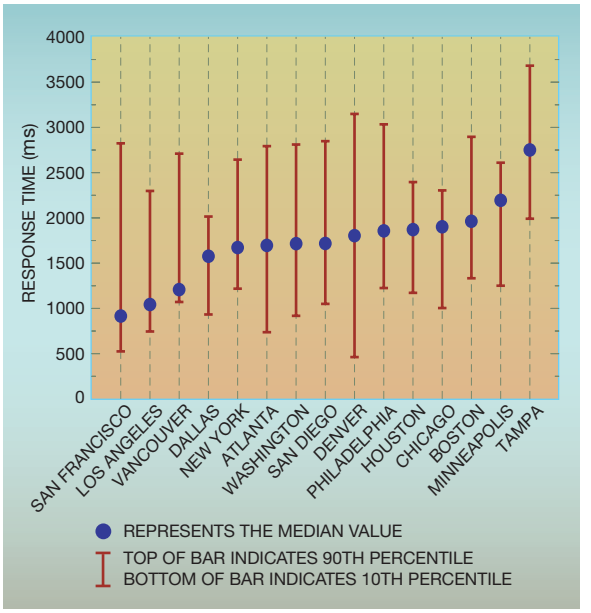


Figure 8B Response time performance by client metropolitan area



ences, however. One limitation of these measurements is that Keynote agents use one connection for each embedded object. Thus the performance does not benefit from using persistent connections (or other HTTP version 1.1 features). Hence, the negative effects of packet loss in the network are exacerbated.

The overall response time performance can be further broken down into component operations. In order to view a Web page, the client typically must first resolve the Web site host name to an IP address using the DNS and establish a TCP connection to the Web server. Then the client issues a HTTP GET request for the base Web page and waits for the server to respond. After receiving the base page, the client parses the page to extract links to any embedded objects and then requests each object. The last phase may itself include additional DNS requests and connection establishments to retrieve objects located on different servers. In Figure 9A we show the median time to complete each of these operations categorized by agents on different ISPs. As might be expected, the response time is dominated by the time to fetch the objects, though this time is inflated somewhat by the lack of persistent connections. It is interesting to note that the height of the other components (i.e., DNS, connection establishment, server response) are relatively similar across most ISPs, consistent with our observation that agents on different ISPs experience roughly equal performance.

Finally, we collected data about the distribution of client requests to the CSU in order to evaluate the effectiveness of the DNS-based wide-area request-routing mechanism. We show the distribution of client requests to four of the CSU sites in Figure 9B. Request-routing is based primarily on network delay and site capacity. Because network delay roughly correlates with geographic distance, we expect that clients in California, for example, would be directed primarily to the West Coast site. From the figure we see that the request routers generally direct a majority of client requests to the nearest site, though other sites are also explored. In some instances, clients in a particular metropolitan region are directed in roughly equal proportions to two sites to balance load, or because the network delay is similar.

Increased reach through peering. In some cases content providers may wish to expand the reach or capacity of the CSU by taking advantage of other CDN networks. For example, by using a regional caching service in conjunction with an edge-caching CDN for

Figure 9A Total response time broken into its components

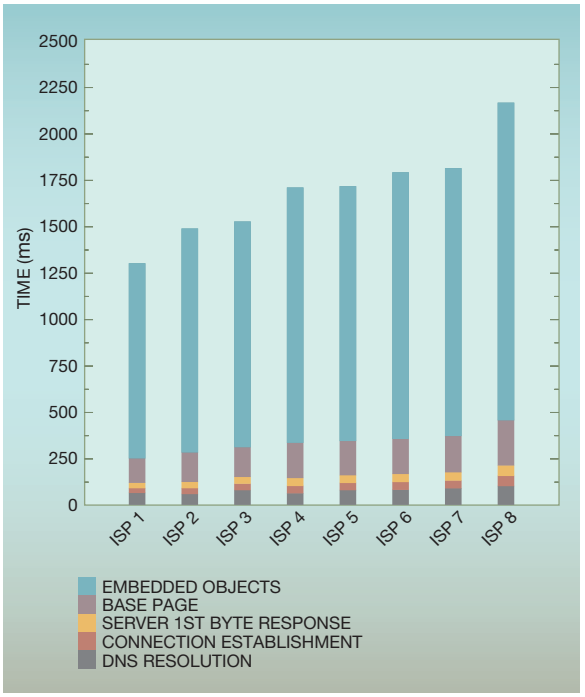
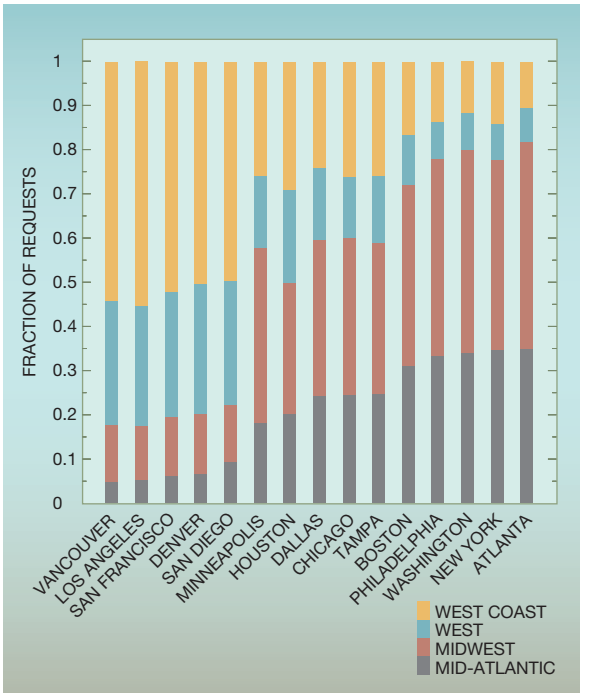


Figure 9B Client request distribution to CSU sites



certain clients, the customer can transparently improve performance and capacity (with an additional cost). To enable this option, we developed a DNS-based request router to facilitate content internetworking[10] (i.e., content peering). The peering DNS server is deployed within the CSU infrastructure and is configured to be authoritative for a special domain name for peering. Clients contacting the customer site issue a DNS request that is handled (perhaps after a referral) by the peering DNS server. On the basis of customer-specified policies, the peering DNS responds with a referral to the CSU or one of the partner networks. Each participating CDN (including the CSU) then conducts its own internal request-routing to ultimately return a server address to the client.

The peering request router enables a number of simple policies. For example, customers could opt to use peering to increase capacity or reach during certain major events by specifying a peering policy that becomes active at a particular date and time. Another useful policy allows the customer to designate peering only for clients connecting from specified geographic regions. Additionally, we provided the ability to use various partner networks in combination by specifying that one network, for example, handle 30 percent of the requests and another 70 percent. Finally, these policies can be combined to compose more complex and flexible peering rules.

Concluding remarks

The content-serving utility enables improved Web site performance and availability without requiring customers to build the infrastructure or manage the complexity themselves. In its design and implementation, the CSU exemplifies a number of the characteristics of utility computing services. It uses open standards to manage the environment, such as XML for configuration, and standard protocols for managing content expiration. The CSU provides a virtualized caching service in which its implementation as a shared, distributed infrastructure is transparent to customers. Finally, the CSU exhibits several features of autonomic computing. If a PoD becomes unavailable (e.g., because of network outages or scheduled maintenance), traffic is shifted from the affected PoD and distributed to the other PoDs, generally without impact or visibility to the end users. The CSU also optimizes performance over time by

learning which PoD delivers the best performance to a particular group of end users.

We also found that many of the components needed by the CSU infrastructure could themselves be turned into utility computing services. By taking advantage of open standards wherever possible, CSU components are designed to interoperate with other services, making it possible to compose more complex utility computing services. Monitoring, provisioning, reporting, and billing interfaces are just a few examples of core elements that were developed for the CSU. In the future these components could be provided by using common utility computing services.

Acknowledgments

The design and development of the content serving utility was a broad collaborative effort involving people from the following IBM teams: e-business Hosting Offerings, Portal Development, EMSL, the South Service Delivery Center, Special Events, e-Technology Center Development, and Research.

In addition, we are grateful to Brent Miller and Brian O'Connell of IBM Global Services, and Lisa Amini, Dinesh Verma, and Dakshi Agrawal of IBM Research for assistance in the preparation of the paper.

Cited references

1. A. Barbir, B. Cain, R. Nair, and O. Spatscheck, *Known Content Network Request-Routing Mechanisms,* Request for Comments (RFC) 3568, Network Working Group, Internet Engineering Task Force (July 2003).
2. B. Krishnamurthy, C. Wills, and Y. Zhang. "On the Use and Performance of Content Distribution Networks," *Proceedings of ACM SIGCOMM Internet Measurement Workshop* (November 2001).
3. A. Shaikh, R. Tewari, and M. Agrawal, "On the Effectiveness of DNS-Based Server Selection," *Proceedings of IEEE INFOCOM*, Anchorage, AK (April 2001), pp. 1801–1810.
4. Z.M. Mao, C. D. Cranor, F. Boughs, M. Rabinovich, O. Spatscheck, and J. Wang, "A Precise and Efficient Evaluation of the Proximity Between Web Clients and Their Local DNS Servers," *Proceedings of USENIX Annual Technical Conference* (June 2002), pp. 229–242.
5. M. E. Crovella and R. L. Carter, "Dynamic Server Selection in Internet," *Proceedings of the IEEE Workshop on the Architecture and Implementation of High-Performance Communication Subsystems (HPCS '95)* (1995), pp. 158–162.
6. K. Obraczka and F. Silva, "Network Latency Metrics for Server Proximity," *Proceedings of IEEE GLOBECOM* (2000), pp. 421–427.
7. M. Tsimelzon, B. Weihl, and L. Jacobs, *ESI Language Specification 1.0,* ESI (Edge Side Includes) Technical Specification (2001), http://www.esi.org/spec.html.
8. V. N. Padmanabhan and L. Subramanian, "An Investigation of Geographic Mapping Techniques for Internet Hosts," *Proceedings of ACM SIGCOMM*, San Diego, CA (August 2001).
9. *Caching Proxy Administration Guide,* WebSphere Application Server, IBM Corporation, (November 2002), from links at: http://www.ibm.com/software/webservers/appserv/doc/v50/ec/infocenter/index.html.
10. M. Day, B. Cain, G. Tomlinson, and P. Rzewski, *A Model for Content Internetworking* (*CDI*), Request for Comments (RFC) 3346, Network Working Group, Internet Engineering Task Force (February 2003).
11. Keynote Systems, Inc., San Mateo, CA, http://www.keynote.com.
12. Gómez Performance Network, Gómez, Inc., Waltham, MA, http://www.gomez.com.
13. D. Agrawal, J. Giles, and D. Verma, "On the Performance of Content Distribution Networks," *Proceedings of the International Symposium On Performance Evaluation Of Computer And Telecommunication Systems (SPECTS '01)*, Orlando, FL (July 2001).

General references

B. Krishnamurthy and J. Rexford, *Web Protocols and Practice HTTP/1.1, Networking Protocols, Caching and Traffic Measurement,* Addison-Wesley, Boston (2001).

M. Rabinovich and O. Spatscheck, *Web Caching and Replication,* Addison-Wesley, Boston (2002).

D. C. Verma, *Content Distribution Networks: An Engineering Approach,* John Wiley & Sons, New York (2002).

Accepted for publication August 25, 2003.

Peter Gayek *IBM Global Services, P.O. Box 12195, Research Triangle Park, North Carolina, 27709 (pwgayek@us.ibm.com).* Mr. Gayek is a Senior IT Architect in the e-Technology Center, which provides worldwide technology and development services for offerings within IBM Global Services (IGS). He received a B.S. degree in computer science from Cornell University in 1982. Much of his career at IBM has focused on the design and development of networking products, including satellite controllers, local-area network (LAN) bridges and gateways, routers, and switches. He also did extensive standards and architecture work on the integration of Systems Network Architecture (SNA) and Transmission Control Protocol/Internet Protocol (TCP/IP) networking. Since joining IGS, Mr. Gayek has designed the architecture for a number of solutions and new services in support of both e-business Hosting and the wider Global Service Delivery community. One of his areas of specialty has been content and application distribution, and he served as the lead architect of the content serving utility discussed in this paper.

Richard Nesbitt *IBM Global Services, P.O. Box 12195, Research Triangle Park, North Carolina, 27709 (nesbittr@us.ibm.com).* Mr. Nesbitt is a senior software engineer in the IBM Special Events Web Solutions group. During his career at IBM, he has worked on projects such as IBM's Internet Connection Server, the Lotus Domino® Go Webserver, and IBM HTTP Server. For the last four years, Mr. Nesbitt has developed custom applications for hosting of high-volume Web sites. They include tools in the areas of stress testing, intrusion detection, and log analysis, as well as

custom, high-performance Web server plug-ins for sports Web serving. He received a B.S. degree in computer engineering from Virginia Polytechnic Institute and State University in 1992.

Herbie Pearthree *IBM Global Services, P.O. Box 12195, Research Triangle Park, North Carolina, 27709 (hpear3@us.ibm.com).* Mr. Pearthree is the technical lead of IBM's Special Events Infrastructure. Since joining IBM in 1999, he has served as a technical webmaster responsible for the end-to-end technical leadership of the Special Events Infrastructure, including the 2000 Olympic Games in Sydney. Mr. Pearthree was the lead technical designer and implementer of the content serving utility discussed in this paper.

Anees Shaikh *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (aashaikh@watson.ibm.com).* Dr. Shaikh is a research staff member in the Networking Software and Services group in IBM Research. He received a Ph.D. degree in 1999 from the Department of Computer Science and Engineering at the University of Michigan, and BSEE and MSEE degrees in 1994 from the University of Virginia. Since joining IBM, his research has focused on Internet services infrastructure, particularly the areas of content distribution, Web and network performance, and Internet measurement. He has also published a number of papers on load-sensitive routing, middleware for real-time communication, and multicast routing.

Brian Snitzer *IBM Global Services, P.O. Box 12195, Research Triangle Park, North Carolina, 27709 (snitzer@us.ibm.com).* Mr. Snitzer is the manager of networks, security, and architecture for IBM's Special Events Infrastructure. He received a B.A. in computer information sciences and in philosophy from Temple University in 1997 and is a candidate for the degree of Master of Business Administration from Duke University's Fuqua School of Business in May 2004. Since joining IBM, he has worked in emergent technologies as part of IBM's WebAhead initiative, served as a technical webmaster providing overall end-to-end technical leadership for event Web sites, served as technical lead for hosting NBCOlympics.com for the 2000 Olympic Games in Sydney, and has served as manager of the network, security, and architecture group for the Events Infrastructure since 2000.