TeraNode™

# How to cash in on caching

entera

Internet Content Delivery Systems

## High-Speed Information Access—The Problem at Hand

The information highway—in the past few years, the Internet has become an integral part of everyday life. Approximately 37 million adults in the U.S. alone "surf" the Web from home on a daily basis, compared to only 19 million in mid-1997. At work, the number of users is up from 19 million in mid-1997 to 32 million.[1] In today's low-cost, unlimited-access environment, the ability to keep pace with this increased demand while simultaneously providing high-speed content delivery at a reasonable price is the biggest challenge Internet Service Providers (ISPs) face.

Meeting this challenge is no easy task as networks become more and more congested, delivering agonizingly slow access, especially during peak times. Since most small ISPs can't afford to build large backbone networks, the problem is further compounded as their networks become oversubscribed. Although the delivery of e-mail messages and Web content is adequate, the service quality for users with access speeds greater then 28.8 BPS is lacking. Today's broadband technologies promise users connection speeds faster than 56K, yet networks in most U.S. areas don't support this rate.

In attempt to improve end-user access, however, many ISPs are introducing these premium-priced, high-speed DSL and cable modem technologies (128Kbps to 1.5Mbps) to residences and small businesses. Although these options promise faster rates and richer services (such as video-on-demand) DSL users can easily overwhelm a provider's upstream network connection unless additional—and expensive—bandwidth is purchased. However, even a large reserve of bandwidth isn't a guarantee against a complete shutdown. "Bandwidth hogs" using NewsBin, NewsBot, or streaming servers, and "flash crowds" like the ones requesting Victoria's Secret, hot.jobs.com, and the Clinton testimony, can cause site access failure.

The largest ISPs are tackling this problem by aggressively deploying caches to support streaming media, broadband delivery (over DSL, cable, and other mediums), and distributed applications. Cache offers better response times in poorly connected networks, supports the oversubscribing that DSL causes, and buffers against upstream outages, mass behavior, and "bandwidth hogs." However, traditional, full-featured caching is costly in terms of personnel and capital resources. This is fine for the companies that can afford the price tag, but most of the smaller ISPs are eagerly seeking a cost-effective way of staying competitive. And many of the larger ISPs would love a cache option that was both full-featured and less expensive.

That fast, cost-effective means is an "Internet Content Delivery System"—cache equipped with a relational database engine and an open API so ISPs can take advantage of new technologies as they become available and develop exciting, new, differentiated applications. This, in turn, allows them to offer end users better service while keeping their costs down and profits up.

[1] *Internet User Trends: Year-End 1998*, a bi-annual study produced by The Strategis Group

## Caching—A Review of the Current Landscape

What exactly is a cache? A cache is a server that replicates, stores, and delivers Internet content, such as Web pages, files/documents, and streaming media (Real Audio, NetShow, QuickTime 4). Caches are strategically placed close to users and/or at strategic aggregation points in order to eliminate potential bottlenecks.

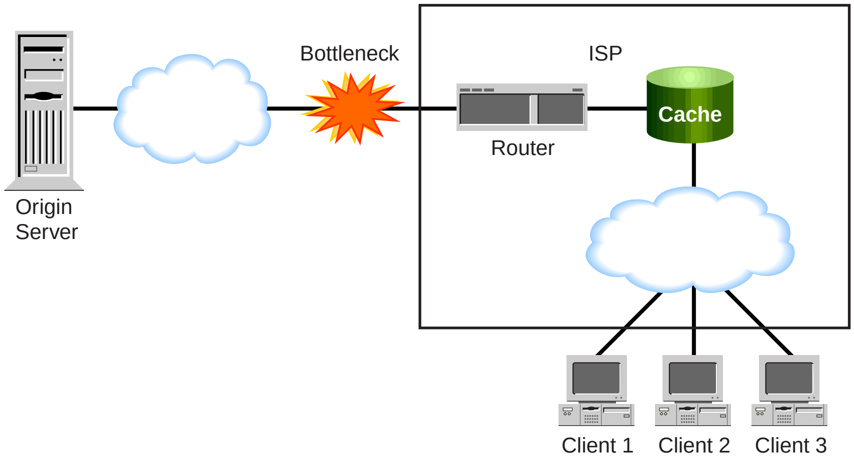

Figure 1: Cache Deployed on the Client Side of a Bottleneck

There are several ways to move content closer to clients. One option is to "mirror" all the content from the origin in another server closer to end users. For example, a U.S.-based company places a copy of its entire site on a server located in England so clients in Europe are closer to the content. In this scenario, data in the "mirror" site is installed and modified by the content provider and is a time-consuming, costly process.

Caching is different. It is a dynamic replication of origin content, based on client requests, that is strategically located on cache server(s) in a provider's network. In this instance, only content that is requested is duplicated.

Caches are beneficial to ISPs in many ways:

- Improve response times in poorly connected networks
- Allow ISPs to implement DSL
- Isolate networks from upstream failures in well-connected networks
- Buffer against mass-behavior
- Protect against "bandwidth hogs."

However, caching isn't without problems. Today's "dumb" caching is based on pull technology. It only stores what users pull into it, "spoofing" them into thinking they're accessing the origin server. This results in two key issues (both of which are covered in more detail on pages 5 – 7):

- Cache coherency—making sure cached content matches the origin content
- Content relevancy or hit ratio—whether or not a cache has the content that a client is requesting.

In addition to the coherency and relevancy concerns, "dumb" caching is also a costly venture. The leading products are expensive. The few that offer a more reasonable price tend to have poor performance. This poses problems for smaller ISPs who want to remain competitive with larger providers.

To fully understand the advantages and disadvantages of caching, a review of the key concepts is necessary. This includes:

- Types of cache
- Cache coherency methods
- Cache relevancy
- Cache hierarchies
- User redirection
- Transparent redirection (including load balancing).

## Types of Caches

Caching can be performed at any point along the delivery path between the client that is requesting the information and the server that provides the information. Different terms are used to refer to the cache depending on where it is deployed in the delivery path. Figure 2 shows some typical locations for cache servers in a network.

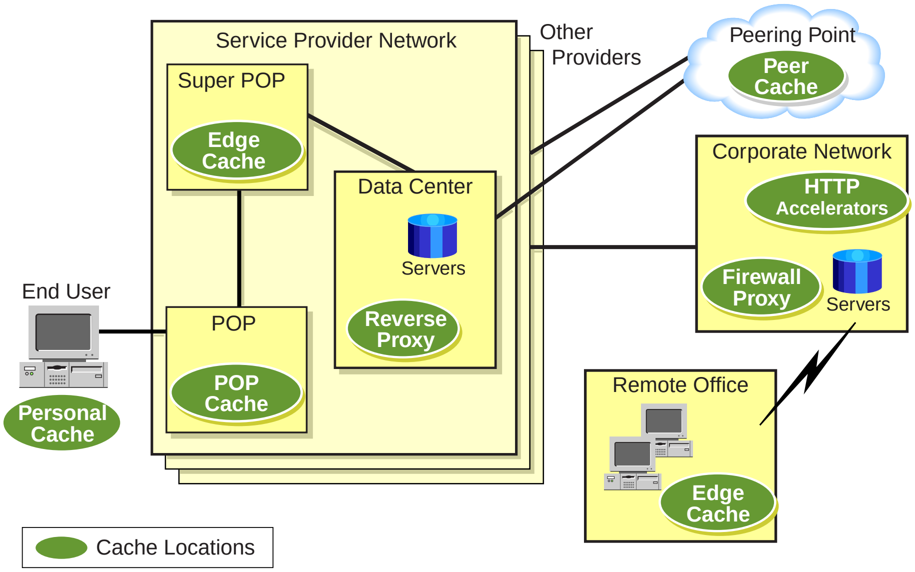

Figure 2: Typical Cache Deployments

### Personal Cache/Personal Proxy Server

A personal cache improves user performance by keeping local copies of frequently requested content on the user's system. All major Web browsers include some limited caching capability (both in features and storage). A personal cache can also attempt to anticipate possible content requests and "pre-fetch" them before they occur.

A personal proxy server extends the personal cache concept by servicing more than one client. (In most cases, two or more computers/devices are attached to the network over a single connection.) The proxy server hides that there is more than one computer attached to it by using its own address when forwarding requests to the origin server. When it receives a response back from the origin, the proxy keeps a copy of the reply in local cache storage and also sends the reply to the requester.

### Point of Presence (POP) Cache

Placing a cache in your POP improves user performance (for all users in that POP), manages bandwidth costs, and isolates networks from upstream failures in well-connected networks. It can be configured in one of two ways (both of these are reviewed in more detail later on pages 9 – 13):

- A proxy where each user specifically asks to use the cache
- A transparent cache where all requests are redirected.

### Edge Cache

A cache can also be placed in each major regional center (known as "super POPs"). This edge cache minimizes traffic across an ISP's backbone, and since most providers lease their backbone network circuits, this offers significant cost savings.

### Peer Cache

The Internet is comprised of thousands of separate networks exchanging information. This is done at peering points, which allow service providers to connect their networks to other ISP networks located at the same point. (An ISP can have connections to several different peering points.) High network-to-network traffic has completely saturated most peering points because the same piece of information is often moved across the same location thousands of times. By placing caches at these points, content is only transferred one time. For all subsequent requests, it is served from the ISP's network cache.

### HTTP Accelerator

Caches are often used to front-end originating servers. These caches are called HTTP or server accelerators. The accelerator—frequently used in conjunction with a load-balancer switch—allows a site to handle higher loads without additional servers. When a request is received, it is directed to an available cache. Most likely the cache will already have the information and be able to respond without communicating with the originating server. This significantly reduces the origin server's workload and decreases response time.

### Firewall Proxy Server

A firewall proxy server is typically found at a company's Internet interface where it performs several functions:

- Blocks unauthorized outside client access to the provider network
- Provides the ability to control employee access to the Internet
- Stores frequently requested information in cache to improve user response time and reduce network costs
- Integrates with URL databases that restrict access to sites containing material not in line with company policies.

Until recently, the primary focus of firewall proxy servers has been on access control and security, with limited use for caching.

### Reverse Proxy, Distribution Content Caching (DCC)

Any enterprise doing business on the Internet must be able to scale their service and manage user response time. DCC (more commonly known as reverse proxy) is one option. With this approach, caches are deployed at high-latency areas for a company's site, such as Japan and Europe (if the origin server is in the U.S.) or major traffic areas (large ISPs that route visitors to a company's content). When a request is received, it is redirected from the central server to the cache closest to the client making the request using a Domain Name Service (DNS) redirection.

## Cache Coherency Methods

A cache needs to check for possible discrepancies between its copy of the content and the original. This is called cache coherency. There are several methods for validating coherency and keeping cache content fresh. Choosing the best approach depends on the type of content that is being replicated and the business requirements. Most cache coherency options don't require that the original content be checked each time that it's requested. Instead, they provide a means for defining how stale a cached copy should be before it's rechecked against the original.

As mentioned earlier, one of the problems with today's "dumb" caching is coherency. Verifying the freshness of cache data in well-connected networks can take up to 60 percent of a company's bandwidth, and all that overhead can actually make response times slower than not using a cache at all. To date, none of the different methods offer a very effective means of dealing with this issue.

### Cache-based

Neither HTTP .9 nor 1.0 offered direct support for caches—which made it difficult for a cache to determine if its content was current—and the original HTTP standards didn't support testing for content freshness at all. As a result, many caches relied on internal information to indicate when content should be refreshed. Typically, caches associated refresh timers with content types that were often tunable by end users.

To overcome this problem, caches made use of two HTTP extensions:

- Last-updated (HTTP .9)—an HTML tag used by the cache to determine if its content was still current. Initially this method relied upon Web page authors to include the tags in their HTML documents. As this approach became more common, Web servers were updated to automatically include last-updated tags in the reply header based on file-modification times. To determine if its cached content was current, the server periodically issued HEAD commands that retrieved only the content summary information. If the data wasn't fresh, the cache had to connect a second time to retrieve the updated content.
- If-modified-since (HTTP 1.0)—a conditional GET command that includes an if-modified-since variable, implemented to avoid the "dual-connection" situation of last-updated tags (see above). When a server receives a document GET request, it always returns the HTTP header information (similar to a HEAD command), and, if the document has been modified, it also returns the updated document.

## Pragma No-cache (HTTP 1.0)

The pragma-no-cache tag tells the cache to serve up a fresh copy of the requested content. Intended for clients, the pragma no-cache feature is also useful for servers. If a document includes a pragma-no-cache tag, the cache knows to force a revalidation of the replicated content it currently has.

## HTTP 1.1 and Beyond

One of the major improvements in HTTP 1.1 was more extensive support for cache servers. With HTTP 1.1, both the client and origin server can provide information to the cache that helps it make decisions on how and when to refresh or expire replicated content.

Clients with 1.1 can now tell a cache to:

- Never cache a document
- Refresh if it's older than (n) seconds
- Refresh if it will be stale within the next (n) seconds.

Origin servers with 1.1 can now tell a cache to:

- Expire its current copy
- Don't cache this response
- Only cache this response if it's a private server.

## Pre-fetching

Several caches attempt to "pre-fetch" (or guess) the content that users might want next and download it proactively. While a good idea, there currently isn't much to ensure that a cache "guesses" correctly, and often this scheme uses more bandwidth then it saves.

## Cache Relevancy

Since “dumb” caches are based on pull technology, they only store what users “pull” into them. This give caches two options; either they rely on user behavior or they try to anticipate what users may want to see. Add to that the fact that not all network traffic is cacheable and you have several factors impacting a cache’s content relevancy or hit ratio. (A cache hit occurs when content requested by a client is located on and served by the local cache. A cache miss happens when the requested content isn’t found in the cache and must be fetched from the origin server.)

For POP cache to be cost effective, it needs to have a cache-hit ratio of fifty percent or greater. It is estimated that the maximum cache-hit ratio is approximately 40 percent if cacheable DNS, HTTP, and news traffic are cached (see Table 1 below). To raise the cache-hit ratio to 50 percent, a POP cache must handle all the protocols listed in Table 1 as well as FTP, streaming media, and other protocols.

**Table 1: Percentage of Cacheable Traffic**

| Traffic by protocol | Percent of total (est.) | Percent cacheable (est.) | Composite percent (est) |
|---|---|---|---|
| DNS | 18 | 90 | 16.2 |
| HTTP | 30 | 50 | 15.0 |
| News | 10 | 90 | 9.0 |
| Total | | | 40.2 |

Another option is to pre-populate the cache with content pushed through delivery sources, such as satellite or wireless. For example, when a request comes in from a user, it is forwarded over the Internet to a satellite provider like SkyCache. The requested content is then broadcast to all the caches on the satellite network at faster speeds and a lower cost than with terrestrial lines. What is returned? An interesting, representative sample of Internet requests closely paralleling the top requested content on the Internet—sent out-of-band, over satellite, 24 hours a day.

## Cache Hierarchies

Arranging cache servers in a hierarchy or mesh topology and searching for information among caches before directly connecting to the origin server can be beneficial in many situations, such as:

- A poorly connected network where linking to the original server is always slow compared to looking in neighboring caches
- Static content that allows the cache server to create an economical distribution mechanism
- A need to reduce redundant traffic across and between networks
- The only economical method for delivery content.

## Internet Cache Protocol (ICP)

ICP, the standard method for creating cache hierarchies, is very simple. When a cache wants to know if another cache has a copy of the information it needs, it sends a request to its defined parents. If a parent has the object, it sends the data back to the child. If the parent doesn't have the requested content, it will send a query to its siblings on behalf of the child. If the parent's siblings don't have the information, the parent will retrieve the content from the origin server and feed it back to the child who, in turn, serves it the client (see Figure 3). This type of process is referred to as a message-passing architecture. In order to determine if a given neighbor cache has the requested content, it must send the cache a message and then wait for a reply.

The current ICP standard has three significant drawbacks:

- It increases client response time because the client must wait while messages are exchanged between caches
- It uses the same network bandwidth it's trying to save and thus has limits on its growth
- It poses security and payload concerns.

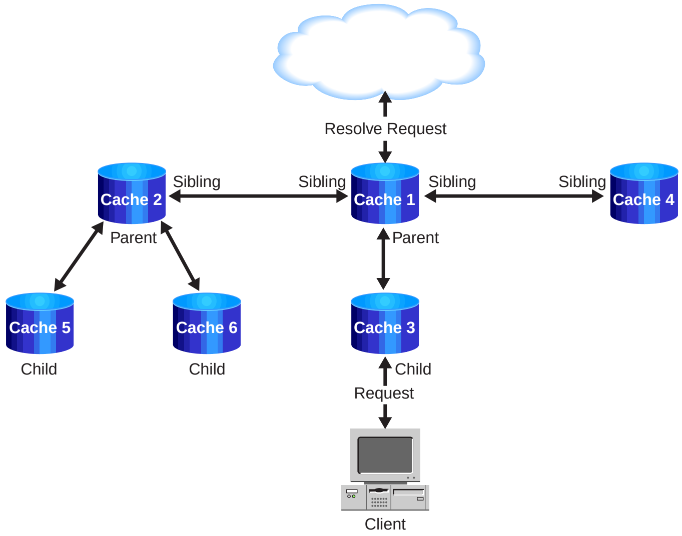

**Figure 3: Example of a Cache Hierarchy**

A cache's relationship (parent, sibling, or child) with other caches in the hierarchy is relative to the cache's position. A cache can be a sibling in one relationship with a neighboring cache, and that same cache can be a parent or child in another relationship.

## Cache Digest

Cache digests represent an improvement to the ICP message-passing architecture. With this approach, each neighboring cache broadcasts a list of its contents to other caches in the mesh (similar to broadcast- and table-based routing algorithms, such as Route Information Protocol.) This information is then used to build a local lookup table that each cache uses for quickly determining if a neighbor has the requested content. If the appropriate cache can be identified, the data is retrieved with a single request-and-reply sequence and doesn't burden the network with additional traffic.

### Cache Array Protocol (CARP)

To address some of the problems of ICP, Netscape and Microsoft jointly created the Cache Array Protocol. Like ICP, CARP allows the administrator to define relationships and create a hierarchy or mesh topology. However, it doesn't rely on message passing to determine which server has the requested content. Instead, a replicated piece of data is always assigned to the same server. The server it's assigned to is decided with a "hash" that's based on the server and path portions of the requested URL.

In addition to solving ICP's performance and scaling issues, CARP also addresses security and payload concerns. However, CARP is not an ideal solution either. The biggest issue with CARP is an inability to perform load balancing. Since requests for a given document are always directed to the same server, there's no opportunity to distribute frequent requests amongst more than one server.

## User Redirection

For a cache to work, a client's request must be directed to the cache versus the origin server. This is accomplished in one of two ways: the user can elect to use the cache or all users can be transparently redirected. This section covers the ways users can redirect their systems.

### User Proxy Selection

In this instance, the user enters the IP address or host name of the cache/proxy server in their Web browser preferences. For example, in Microsoft Internet Explorer (I.E) 4.0, the user configures the browser to access the Internet via a cache by selecting "Access the Internet using a proxy server" in the connection dialog box. If the user clicks the "Advanced" tab, s/he can provide additional configuration information for each protocol.

Note: If the user doesn't specify a proxy server for a given protocol, most browsers will bypass the cache and connect directly to the origin server.

### Auto Proxy Configuration

I.E. also offers an "Automatic Configuration" option that uses a JavaScript (created by a system administrator) to determine which cache it should connect through. The script always includes a "FindProxyForUrl" function that the browser calls for each new URL request, and can be written to point the client towards a primary or secondary cache or directly to the origin server. With auto proxy configurations, users don't have to know the specific information for each protocol that they wish to use.

Auto proxy configuration scripts can also handle load balancing and failover situations (please refer to load balancing in Transparent Redirection).

## Transparent Redirection and Load Balancing

User-based redirection may not be an option due to security restrictions, administration burdens, or other such reasons. When this is the case, administrators need to reroute traffic transparently to the proxy server.

There are several ways to do this, all of which rely on the same basic technique. The device performing the redirection looks at the sending and destination addresses of each incoming IP packet and determines if the packet should be passed on or redirected to a cache. If the packet is redirected to a cache, it's delivered just as if the cache was a router on the network. The cache then processes the packet and takes appropriate action.

### Router

Generally, there's a router that connects a local network to a remote network. This same router can be configured to handle transparent redirects, and is sufficiently powerful and cost-effective enough for most installations. The drawback, in this case, is the router can't detect a failed cache and configure around it, nor can it distribute workloads across multiple caches.

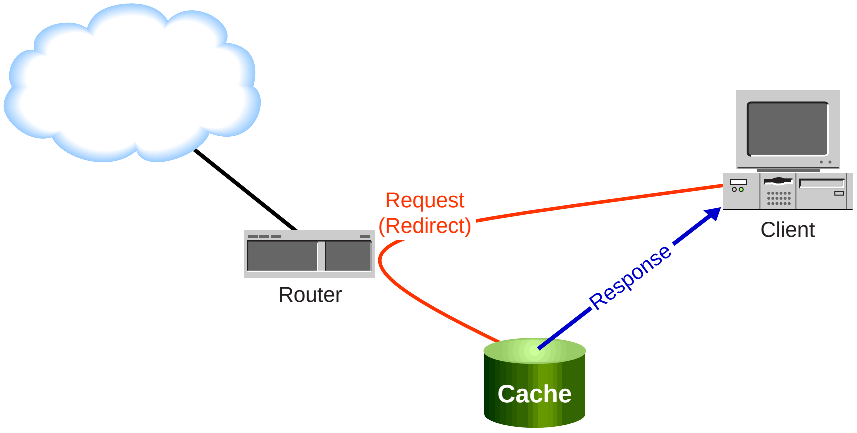

Figure 4: A Router Configured to Handle Transparent Redirect.

### Layer Four and Layer Seven Switches

Some installations require a more robust redirect option than a router offers, such as redundant caches, automatic failover, and other similar features. Layer Four and Layer Seven switches support more robust methods.

A Layer Four switch either replaces the local hub/switch or sits between the caches and the router to intercept and reroute traffic. It detects failed caches and transparently routes around them or forwards all requests to the network if no caches are available. Most Layer Four switches perform load balancing in several ways, including least number of connections, round robin, least busy (as determined by the switch), or hash algorithms. Load balancing between Layer Four switches during busy periods is also possible.

When the simple load balancing methods used by Layer Four switches aren't sufficient, Layer Seven switches can be used. Layer Seven switches work at the application layer of the OSI stack, normally intercepting DNS requests and determining which cache should receive the requests. Decisions are based on information (server load, type of request, and so forth) that the Layer Seven switch gathers from the network and servers.

## Load Balancing

There are several load balancing approaches to choose from:

- Round robin—caches are grouped together based on their IP (and possibly port) addresses. As requests come in, they are passed to the caches in an ordered sequence starting with the first server. When the last server is reached, the process starts over. In addition to a straight round robin, some methods support weighing caches, in which case, the administrator decides what proportion of connections should go to each cached based on perceived size.

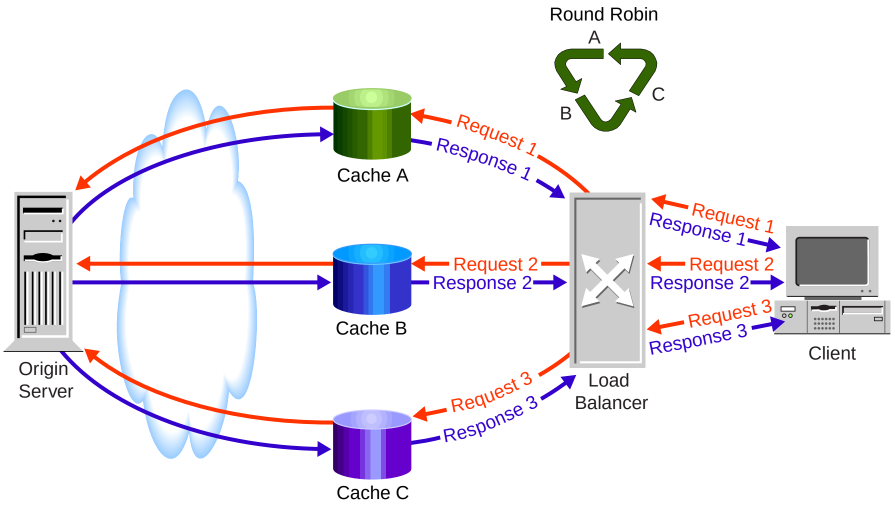

Figure 5: Round Robin Load Balancing

- Connection-based—this scheme is based on how many current connections a cache is handling. If caches A and B have three connections and cache C only one, the next two connections would go to C. In the event that all the caches have the same number of connections, then round-robin load balancing is used. Most connection-based schemes also allow you to weight the servers, in other words, cache A would get two connections for every one that cache B receives.

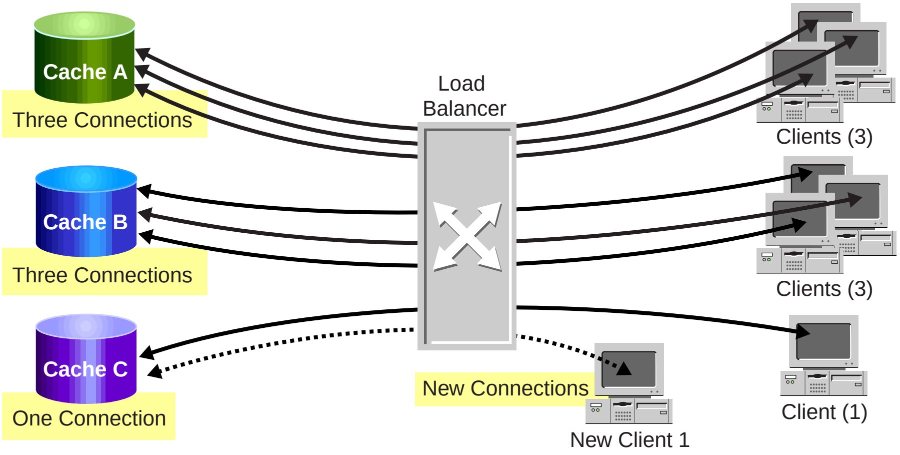

Figure 6: Connection-based Load Balancing

- Hash-based—a hashing algorithm can improve cache hits in multiple-cache clusters because it assigns individual content requests (www.entera.com) to specific caches. All requests for a specific URL are then directed to its associated cache. This is very different from a round robin where requests are passed in an ordered sequence from cache to cache even if the subsequent requests are for the same piece of information. In round robin, a specific request is not necessarily served from cache the second time it is received, and copies of the requested data—all of which may be different—could exist on multiple caches.

Hash-based load balancing indexes the content and assigns every given URL to one of the available caches. If that cache is not available, the request will be forwarded to another cache for processing. Hash algorithms can increase the cache hits ratio above the typical 40 percent. However, they can also limit cache group's total capacity since only one cache is allowed to respond to any given request.

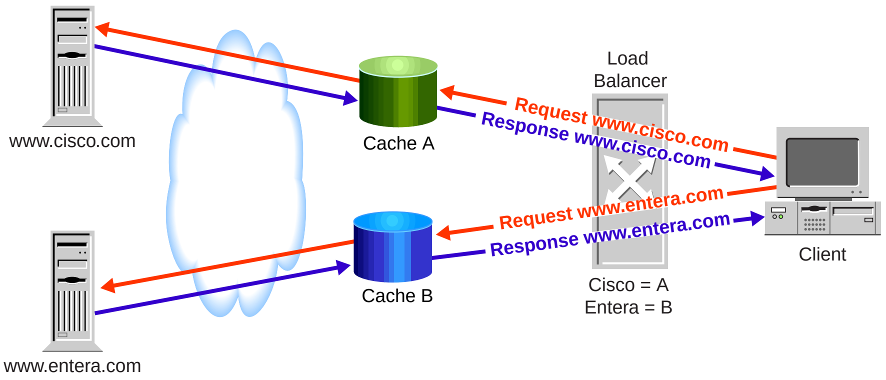

Figure 7: Hash-based Load Balancing

- IP traffic statistics—this method computes a weight-based load metric for each cache as determined by connection-creation time, bytes transferred, network errors (like dropped packets), and network latency. When a Layer Four or Layer Seven switch receives a request, it attempts to direct the request to the least-loaded cache. The cache stores the results of each request and continually updates its "has" table, which the switch uses to determine the least-loaded cache.

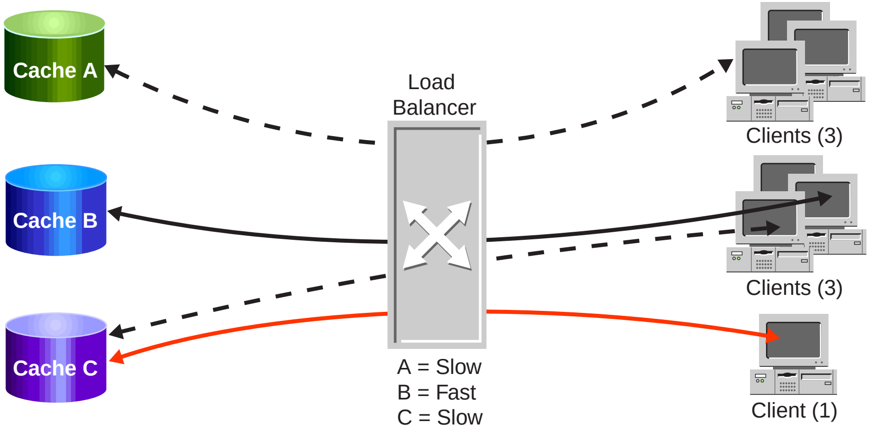

Figure 8: IP Traffic Load Balancing

- Agent instrumentation—a common problem with the above approaches is that they all establish their metrics using only network-level information. While that's good, it may not provide an accurate picture of cache-level activity. To resolve this problem, some vendors created agents that run on each cache and gather statistics on memory, CPU, disk, and network utilization. This information is fed back to a Layer Seven switch, which uses the information to make loading decisions.

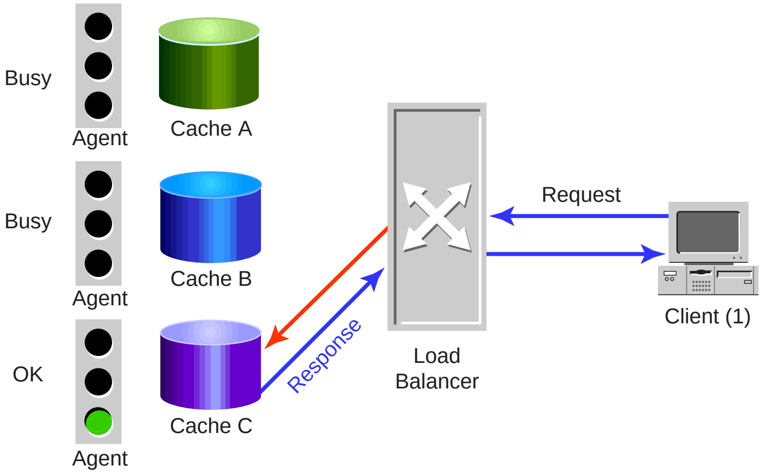

Figure 9: Agent Instrumentation Load Balancing

## Internet Content Delivery System—The Wave of the Future

With an open API and a relational database engine, an Internet Content Delivery System (ICDS) takes caching to the next level. It offers all the benefits of caching at a much lower cost while delivering the flexibility and intelligent decision-making capabilities needed to overcome the current drawbacks (such as "spoofing" and an HTTP focus) and position ISPs to take advantage of new technologies as they become available.

So what should ISPs expect from intelligent caches with ICDS capabilities? A lot.

- Support for high-speed services, such as DSL, while minimizing upstream costs
- Inexpensive installation and the ability to expand the hardware with memory, storage, and gigabit networking as business grows
- Buffering from mass-behavior events like Victoria's Secret or the Clinton testimony as well as "bandwidth hogs" (NewsBot, NewsBin, and so forth)
- Better service quality in streaming media for customers
- The ability to develop and add new services easily with third-party plug-ins or their own
- Proven and reliable hardware, fail-safe design, and commercial support

## What to Look for in a Caching Product

There are several key attributes that service providers should look for when evaluating caching technology—all designed to give them a competitive edge.

### Affordability

Since smaller ISPs tend to run lean on cash, minimal capital outlays are essential. The entry point for a basic caching system should be low, and leasing programs should also be available. However, leasing shouldn't be a substitute for low entry prices, and payback calculations should be straightforward. If a complicated payback calculation is required, a provider is most likely paying too much.

While larger ISPs can afford more expensive caching solutions, most have only been able to add them to their data centers. Those larger ISPs could enhance service quality and user experience by distributing caches to the edges of their networks, at smaller POPs. An affordable, entry-point caching system would make this possible.

### Customization

To survive in the competitive ISP market, service providers need every opportunity to differentiate themselves. That's where a flexible network infrastructure can help. "Dumb" caches only store objects that a user requests. Intelligent caches, on the other hand, move ISPs closer towards an ICDS by offering more customization features, such as the ability to edit parameters (policy customization) and extend services via open API.

### Software Expandability

Cache software should grow with an ISP, both in performance and protocol support. In order to do this, the cache must have a modular design, open API, and standard SQL interfaces.

Performance expansion requires some type of clustering or client/server scaling. This can be accomplished in several ways, including:

- Process-level, client/server clustering
- Load sharing through Layer Four and Layer Seven switches
- Message-passing, cache-hierarchy protocols, such as ICPv2.

Message-passing protocols are adequate for least-common-denominator interoperability, but add significant overhead and latency—arguably eliminating many potential benefits. Layer Four and Layer Seven switches perform better, but can be expensive, create redundant cache stores, and reduce efficiency. Service providers will probably use all three approaches simultaneously, though only the more sophisticated cache products have modular software that can be distributed across many CPUs using a client/server clustering architecture.

Given the rapid evolution of the Internet, it's perhaps more important that cache servers support an easy way of adding new protocols and standards. An ISP's success is predicated on how well it can find new niches, develop new applications, and create new ways of acquiring, engaging, and delighting customers. Products with open APIs allow ISPs, third parties, and the research and development community to develop new protocols for cache.

Since there is already more than just HTTP on the network, caches should support USENET News, streaming media, DNS, RADIUS, and other types of cacheable protocols. If the cache doesn't currently support a particular protocol, consider how fast and easy it is for the ISP or cache vendor to add the protocol.

Dynamic content is also becoming more the rule than the exception on the Internet. As such, a cache that supports SQL-based database storage and an open API for custom application development is essential to transporting that content.

### Installation and Setup

ISPs are as pressed for time as they are money, and installation is typically the most time-intensive phase of implementing a new product. This is where appliances have a tremendous advantage over software-only cache solutions. An ISP who is tight on time, with a small staff, should look for plug-and-play appliances that are easy to install and require minimal setup for basic functionality. A cache should be up and running in less than 30 minutes.

### Commercial Support and Reliability

Most ISPs can't afford the luxury of extra staff just to repair outages when they occur. Therefore, product reliability and manufacturer support are critical to an ISP's success. Most caches on the market are based on freeware software (either Squid or Harvest code bases). While an inexpensive start for many small ISPs, freeware usually ends of costing more in time, effort, reliability, and support on the back end.

For businesses that depend on fast, reliable support to back up their staff and maintain profitability, freeware isn't the answer. The extra reliability and assistance available from commercial products can make all the difference—maintaining uptime and freeing technical resources to focus on new applications and services.

When purchasing a commercial product, ISPs most also be careful. Many so-called commercial appliances are simply freeware software (with slight modifications or user interfaces added) sold on proprietary, cost-reduced hardware platforms. There are only a small number of cache products on the market today built from the ground up for commercial use.

### Hardware Upgradability

Caching appliances should allow for easy and inexpensive upgrading so ISPs can take advantage of constant and rapid advances in chips, memory, and disk. Small ISPs may want to start with a single 10/100Mbps connection and then add additional 10/100Mbps connections for redundancy, dual homing, or additional bandwidth at a later date. A caching product based on PCI-bus Intel architecture makes this very straightforward.

As an ISP grows its network or expands and adds new data centers, upgrading from 10/100Mbps to 1Gbps Ethernet LAN connections may be necessary. As such, the ISP should make sure it can add 1Gbps LAN cards later.

## Integration with Business Systems and Applications

The pure-dial-up ISP market is crowded, and there is very little preventing customers from switching providers. ISPs must offer additional services to differentiate themselves and to develop customer relationships. As a result, more and more ISPs are becoming Application Service Providers (ASPs). Outsourcing applications adds value to their services, and generates long-term customer loyalty.

Caches must integrate with an ISPs outsourced services, business applications, and any other services it wants to offer, including the delivery of dynamic content, response-time commitments, and distributing Web database applications.

## Sparing and Repair

ISPs must maintain 24/7 availability. Closed-box cache appliances are easy to set up, but leave ISPs with few options in case of hardware failure. While completely redundant spare units can be kept onsite, it's not likely to be cost-effective. ISPs should look for solutions that are easy to fix or replace. A good option is a pre-configured, PC-based appliance with the software already loaded. If sparing is a concern, look for an open appliance that supports standard Intel-compatible processors, memory, cards, and disk for maximum uptime.

## Management Tools

Most small ISPs don't have the luxury of setting up sophisticated alert properties or generating reports. However, when trouble strikes easy access to management tools such as Web-based monitoring or a command-line interface is a plus. If the cache doesn't have these options currently, it should at least be part of the planned features. For larger ISPs that have operational staff using SNMP management features, a cache should offer at least basic SNMP trap support.

Standard reporting is also important (vendor-prepared charts and statistics on disk usage and availability, transaction history, and CPU usage). For true management flexibility, however, ISPs need to generate their own reports. This can be done by processing log files, but a more sophisticated/ effective way of tracking Web traffic is to query the cache store itself and generate reports from the results. A cache that stores its objects using an SQL-compatible database would allow this.

## What to Buy

Based on the "What to Look for" section, below is a summary of the general strengths and weaknesses of the most prevalent types of caches used in smaller POPs, as compared to an ICDS for reference.

| | Freeware | Ordinary Cache Appliances | ICDS |
|---|---|---|---|
| Customizable and configurable | ● | | ● |
| Easy to add new protocols and features | | | ● |
| Management tools | | ● | ● |
| Easy to install and set up | | ● | ● |
| Commercial support and reliability | | ● | ● |
| Ability to scale and upgrade hardware incrementally | ● | | ● |
| Affordable for small POPs | ● | | ● |
| Easy and affordable to keep spares and repair | ● | | ● |
| Integrates with business systems and applications | | | ● |

## *The ABCs of Implementing Caching*

After reviewing what to look for in a cache and what to buy, you've made your purchase and are ready to implement your solution. Following are some basic steps for setting up a plug-n-play cache appliance.

1. Enter the basic IP address information
   - The cache's IP address and host name
   - The IP address of your default gateway
   - Your IP subnet mask
   - The IP address of your Domain Name Server
   - Any SNMP information (if you're using SNMP)

2. Configure transparent routing (this can be done in one of two ways)
   - Use the policy routing on your router
     - Create a new access list that lets any Web traffic through, but blocks your cache appliance to avoid loops.
     - Create a route map that permits all your usual traffic from going through, but does some additional matching based on the access list you just created.

    - Set your local Ethernet interface to use the route map you just created as a matter of policy on that interface. This tells the router to use this map on all traffic coming from the local Ethernet interface.
    - The best way to understand the function of this procedure is to read the steps above in reverse order. On incoming packets, the router performs step three, than two, and then one.
- Use a Layer Four or Layer Seven switch (like a Foundry ServerIron Layer Four switch)
    - Ensure the software version is 3.1.08 or higher.
    - Set the IP and mask address for the switch.
    - Set default gateway address for the switch.
    - Turn redirection on for all ports.
    - Configure one or more caches.
    - Turn redirection off for the cache.
    - Turn redirection off for the router.

3. Check your upstream bandwidth utilization
    - Wait a day after implementing so that the cache fills up, and check your router's utilization.
    - You should notice:
        - Lower bandwidth utilization on your upstream link
        - Buffering against mass-behavior events such as Victoria's Secret and Clinton's testimony
        - Buffering for users against outages and Border Gateway Protocol (BGP) churn in your upstream connection
        - Better service quality for your customers.

## *Conclusion*

Caching answers an ISP's immediate needs for implementing DSL and streaming services for customers—without worrying about a site crash due to upstream outages, "flash crowds" (Victoria's Secret and Clinton's testimony), or "bandwidth hogs" (NewsBin and NewsBot). However, most small ISPs can't afford the price tag (both in technical resources and cash outlay). Even if they could, "dumb" caching isn't a cure all. Based on pull technology, it only stores what users request, spoofing them into thinking they're accessing the origin server. This results in cache coherency and relevancy issues for all ISPs, regardless of size.

ISPs need an intelligent cache option that is fast and economical. One that is equipped with a relational database and an open API. One that will help keep their networks up, lower their costs, and boost their profits. Bottom line—they need a solution that allows them to satisfy their customers.

## *Glossary*

| | |
|---|---|
| agent | Software running on a cache or server that gathers statistics, such as memory, CPU, disk, and network utilization. |
| cache | A server that replicates, stores, and delivers Internet content. |
| cache accelerator | A configuration in which a switch is used to front-end one or more caches. (See also layer four switch and load balancing.) |
| cache hit | Copy of requested content is found in cache. |
| cache miss | Copy of requested content is not found in cache. |
| CARP | Cache Array Protocol—lets administrators define neighboring cache (parent, sibling, child) to create a hierarchy or mesh topology (i.e., ICPv2) so caches can communicate with one another. |
| coherency | Degree to which the source content on the origin server matches or is up to date with the content stored in the cache. |
| content | A file, data stream, HTML document, image, applet, or any other object addressable by a single URL. |
| DNS | Domain Name Server—stores an index of IP addresses and domain names, and resolves or translates between the two. |
| dynamic content | Data generated “on the fly” by a Web-based application, such as CGI, Perl script, JavaScript, or Java Applet. |
| firewall | A barrier set up at a network interface to prevent transmission of data between networks. A simple firewall can be set up with a router; more sophisticated firewalls use proxy servers. |
| FTP | File Transfer Protocol—an Internet Engineering Task Force (IETF) protocol commonly used for transferring files over the Internet. |
| hash | A function that creates a hash table—a group of pairs, each with a key value and a data value. The same key will always return the same data value. |
| hash-based load balancing | Load balancing based on matching a specific URL to a specific cache and storing the pairs in a hash table. |
| HTTP | Hyper Text Transfer Protocol—a commonly used protocol for World Wide Web documents. |
| ICP | Internet Cache Protocol—An IETF standard for communication between caches. |
| latency | The time between a request for and the receipt of data. |
| layer four switch | A switching device that operates from data contained in the transport layer of the OSI 7-layer protocol. |
| layer seven switch | A switching device that operates from data contained in the application layer of the OSI 7-layer protocol. |

| | |
|---|---|
| MAE | A service mark of MCI WorldCom (originally an abbreviation for Metropolitan Area Exchange). See also peering point. |
| neighbor cache | Either a parent, child, or sibling cache when the distinction between the three is not significant. |
| NAP | Network Access Point. See also peering point. |
| origin server | The Web server that hosts the data, such as a Web page. |
| payload | The portion of the packet that contains the actual data and does not include the overhead fields used for transmission control. |
| peer cache | A cache placed at a peering point. |
| peering point | Multi-network connection point for exchanging data between a service provider's networks (also known as NAP or MAE). |
| POP | Point of Presence—an access point to a service provider's network. |
| proxy (server) | An intermediary server that accepts and forwards requests from clients. |
| server | A program that accepts and services requests from clients. A server may be an origin server or a proxy server. |
| server accelerator | A configuration which uses a cache to front-end one or more origin servers to speed data retrieval. Also called reverse proxy. |
| sibling cache | A cache at the same level in a cache hierarchy. |
| transparent redirection | Redirection without the knowledge of client software. |
| URL | Uniform Resource Locator—a World Wide Web resource address, such as http://www.entera.com. |
| user | A person usually using the client software to view and obtain content over the network. |

entera

*Internet Content Delivery Systems*

6150 Stoneridge Mall Drive, Suite 115
Pleasanton, CA 94588
925-730-2200
925-730-2271 fax
1-877-4-ENTERA toll-free
1-800-726-5762 toll-free fax
info@entera.com
www.entera.com