Caching, Content Distribution and Load Balancing
Motivation

- Which optimization means do exist?
- Where should means for optimization be applied?  
  First Mile?
  Last Mile?
  Middle Mile?

Web documents recently published containing important news

Within one point in time several thousands of requests have to be handled ("flash crowd" problem)
Outline

• **Web caching and associated protocols**
  • Internet Cache Protocol (ICP)
  • Cache consistency mechanisms

• **Content Distribution Networks (CDNs)**
  • Architecture of a CDN
  • Content replication mechanisms
  • Akamai

• **Load Balancing**
  • Proxy-based Load Balancing
  • Elastic Load Balancing
Web Caches

- Mechanism for temporary storage of Web resources to reduce bandwidth usage, server load, and perceived lag
- A Web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache
- Caches are often deployed
  - by Internet Service Providers (ISPs) to cache content for own users
  - by Web server Providers to avoid regeneration of content for different users
- ISPs and Web servers run various caches in parallel which store subsets of entirely cached documents
  - Interaction between caches is important for cooperatively answer requests
Internet Cache Protocol

- Request / reply protocol used for coordinating (hierarchical) Web caches and for exchanging data between caches **on demand**
- Specified in Request for Comments 2186 and 2187
- Goals: to enable efficient inter-cache communication and to minimize the number of remote requests to the originating server
- Most important protocol message semantics:
  - ICP_OP_QUERY: Request a resource from another cache
  - ICP_OP_HIT: Indicate that a cache offers a specific resource
  - ICP_OP_MISS: Indicate that a requested resource is not available in a cache

Requested Web resource is locally not available

HTTP GET-Request

ISP Cache

ICP_OP_QUERY

ISP Cache

ICP_OP_HIT

HTTP GET-Request

HTTP Reply

Web resource is delivered to Web client and cached locally

User Datagram Protocol (UDP) is used as transport layer protocol

HTTP Reply
Internet Cache Protocol

- Using configuration options in cache systems, cache hierarchies can be built
- Sample configuration file used, e.g., in the ICP-enabled proxy server implementation “Squid” (path in Unix-based systems: `/etc/squid/squid.conf`)

```plaintext
cache_peer parentcache.example.com parent 3128 3130
cache_peer childcache2.example.com sibling 3128 3130
cache_peer childcache3.example.com sibling 3128 3130
```

TCP port for HTTP communication with cache
UDP port for ICP communication

parentcache.example.com
ISP Cache
access
Origin Web server

ISP Cache

childcache1.example.com
childcache2.example.com
childcache3.example.com

(sibling cache systems)

ISP Cache

ISP Cache

ISP Cache
Hierarchical Cache with ICP

- Caches first try to resolve requests on the same level and thus request sibling caches for a Web resource not available locally.
- If the Web resource is not available or if the validity of a Web resource has expired in all sibling caches, the parent cache is accessed.
- For minimizing interaction, only parent caches are allowed to fetch Web resources from the origin server.

Specific flag “ICP_FLAG_HIT_OBJ” is set in query, thus the object is directly returned in the response (ICP_OB_HIT_OBJ).
Cache Consistency Mechanism

- **Strong consistency**: The cached versions are consistent with a content provider’s version at all times. An update of the version of the content provider will propagate through the system and all replicas will be consistent after a fixed time period $\Delta$ (=configuration parameter).

- **Delta consistency**: The version available in a cache may or may not be consistent with the version of a content provider.

- **Weak consistency**: To provide cache consistency, three categories of solutions exist:
  - Server-driven consistency: The content provider notifies the caches every time the content changes (e.g. **Web Content Distribution Protocol**)
  - Client-driven consistency: The clients request updates managed Web resources in regular intervals (polling)
  - Leases approach: A lease is a promise by the server that it will push updates only a specified time period
Web Content Distribution Protocol

- HTTP and XML-based protocol for updating replicated content
- Supports strong, delta, and weak consistency

Example for strong consistency mechanism:

Before a commit request is received by the replication / cache server, every request to the invalidated Web object is forwarded to the origin server.
Web Content Distribution Protocol

- Example for an invalidation request sent by the origin server to the replication cache server:

  POST /invalidate_request HTTP/1.1
  Content Length: 512

  <?xml version="1.0"?>
  <!DOCTYPE InvRequest "wcdp.dtd">
  <invalidation sequence_number=100>
    <identifier>
      <object_invalidation_id=9876/>
      <object_invalidation_group_id=12345/>
    </identifier>
    <action>
      <invalidate=immediate/>
      <refresh=yes/>
    </action>
    <type atomic=no/>
    <consistency require_commit=yes/>
  </invalidation>

  HTTP header
  - Sequence number used, e.g., in later commit request
  - Identification of the invalidated object
  - Due to strong consistency, the invalidation should be done immediately
  - atomic = no: a single object and not the whole group is updated
  - Commit is sent after all subscribers have sent a response
Leases-based Caching

- Server issues lease on first request and sends notification until expiry
  - Clients need to renew lease upon expiry time if necessary
- Smooth tradeoff between state and message exchange
  - Zero lease duration: polling, infinite leases: server-push
- Lease duration may be determined based on various factors:
  - **Age-based leases**: granting short leases to frequently modify objects and long leases to long lived objects
  - **Renewal-frequency-based leases**: proxies at which an object is popular get longer leases
  - **Server load based leases**: shorter leases are issued during heavy server load
Regular Web Caches - Problems

- Caching proxies are mainly deployed by Internet Service Providers (ISP) in order to handle requests close to users.
- Central Problems:
  - Proxies serve either only ISP clients or Web servers they have been deployed for.
  - No network-wide optimization to avoid congestion or to improve latency issues.
  - Content is not cached proactively – only Web resources requested in regular intervals are located in proxy caches.
  - System is still vulnerable to “flash crowd” problem.
Development of CDNs

Increased functional abilities of Internet and Web Applications, improved performance, amount of content increases

Pre-CDN period
- Hierarchical caching
- Server farms
- Caching proxy deployment
- Efficiency improved Web server

First Generation CDNs
- Static and dynamic content

Second Generation CDNs
- Video on Demand, media streaming

Community-based CDNs
- Combination of CDNs with peer-to-peer-based infrastructures

Beginning of 90's
Late 90's
2005
Today
Future?

When have approaches been developed?

These in practice very important topics are mainly covered in this lecture

Paradigm change: Responsibility shifts further to end-users (convergence to peer-to-peer-based solutions)

Minor technological gap
Content Distribution Networks

• A Content Distribution Network (alternative: Content Delivery Network) is a system of computers containing replicated data placed at various nodes of a network
• Intended to improve access to data by increasing access bandwidth, redundancy and reducing access latency
• Uses techniques for content replication (=proactive placement of content) and content caching
• Comparison to Proxy Servers:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Proxy Server</th>
<th>CDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key practice</td>
<td>Web caching</td>
<td>Content replication and caching</td>
</tr>
<tr>
<td>Cached content</td>
<td>Dynamically changes; content requested by users of Website / ISP</td>
<td>Predefined content distributed by the CDN-supported content providers</td>
</tr>
<tr>
<td>Scalability</td>
<td>Low to medium</td>
<td>high</td>
</tr>
<tr>
<td>Performance</td>
<td>Vulnerable to “flash crowd” events</td>
<td>Stable; suitable for resource-hungry applications (e.g. streaming media)</td>
</tr>
</tbody>
</table>
Architectural Components of a CDN

Web servers delivering content to end users via a CDN

Content Distribution Network

Request Routing System

Monitoring and Distribution System

Accounting System

Billing Organization

"Edge Servers" offering content replicated from origin servers
Architectural Components of a CDN

1. Origin servers replicate their content proactively to the Content Distribution Network.
2. Content is forwarded to different replication servers via the Distribution Subsystem.
3. Information about the placement of content is provided to the Request Routing Subsystem.
4. Requests of users for specific Web resources are directed to the Request Routing Subsystem which selects an appropriate Replication Server based on proximity and current network characteristics (load, latency, ...).
5. After forwarding the user requests to an appropriate Replication Server, the requested Web resource is delivered to the user.
6. The amount of content is transmitted for billing purposes to the Accounting System.
7. The customers are billed based on the amount of transferred data and usage of the CDN.
Comparison between P2P networks and CDNs

<table>
<thead>
<tr>
<th></th>
<th><strong>CDNs</strong></th>
<th><strong>Peer-to-Peer-Networks</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Category</strong></td>
<td>A collection of networked computers spanning the Internet</td>
<td>File retrieval networks formed by ad-hoc aggregation of resources</td>
</tr>
<tr>
<td><strong>Constitution</strong></td>
<td>Distribution of replication servers to the edge of the Internet</td>
<td>Collaboration among peers</td>
</tr>
<tr>
<td><strong>Main goal</strong></td>
<td>Reducing latency during content delivery in the Web</td>
<td>File sharing among peers</td>
</tr>
<tr>
<td><strong>Integrity</strong></td>
<td>Integrity should be guaranteed between replication servers</td>
<td>Not addressed</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>Strong cache consistency between cache servers</td>
<td>Weak consistency between cached content</td>
</tr>
<tr>
<td><strong>Autonomy</strong></td>
<td>None</td>
<td>Autonomous peers</td>
</tr>
<tr>
<td><strong>Administration</strong></td>
<td>Individual companies, proprietary in nature</td>
<td>Self-interested end users / peers</td>
</tr>
</tbody>
</table>
Combining replication and caching in CDNs

Replication Server (Storage capacity: \( c + r = 100\% \))

- **Cached Content in Dynamic Cache**: \( c\% \)
  - Filled by classical cache replacement policies such as:
    - Least Recently Used (LRU)
    - Least Frequently Used (LFU)

- **Replicated Content in Static Cache**: \( r\% \)
  - Filled by replication strategy such as il2p

- Total storage capability of a replication server has to be distributed to the dynamic and static cache

  \( \rightarrow \) Central question: Which percentage \( c \) of the total storage capacity should be distributed to the dynamic cache and which percentage \( r \) to the static cache?
Combining replication and caching in CDNs

By combining a huge amount of statically replicated content with a solid amount of content adapted to user behavior, very good performance is achieved.

Configuration turns system into a pure proxy cache with all drawbacks regarding scalability and vulnerability to “flash crowd” events.
Exemplary Replication strategy: il2p

- Latency and load object placement (il2p) = Cooperative push-based replication strategy taking into account network’s latency and the Web resources’ load
- Problem of optimal replication is divided into two sub-problems:
  - Selection of the best replication server for each object (phase 1)
  - Prioritizing the potential replications by taking the object popularity into account (phase 2)
- Phase 1 and 2 are repeated until all objects are distributed and the replication servers do not have any free storage space left

- By taking the popularity of the Web objects in the 2rd phase into account, it can be avoided to place many popular objects on one server thus increasing the probability to overload this server
The first phase determines a distribution of objects to replication servers that minimizes the overall latency.

Goal: Minimize the function $\text{cost}(x)$ where $x$ defines the placement of an object.

Cost of a placement is high if
- a server has got a high request rate,
- the probability that a user will request an object or
- the distance from many replication servers to the object is high.

During every iteration the parameters for calculating the cost are adapted.

$$\text{cost}(x) = \sum_{i=1}^{N} \sum_{k=1}^{K} \left( \left( p_k \cdot \lambda_i / \sum_{j=1}^{N} \lambda_j \right) \cdot D_{ik}(x) \right)$$

$i, j =$ replication servers  
$k =$ object (Web resource)  
$\lambda_i =$ request rate for replication server $i$  
$p_k =$ probability that a user will request object $k$
il2p – Phase 2

• For every placement determined in phase 1, a “utility value” is calculated which takes the load of an object into account:

\[ Utility \_ value_k = load_k \times latency_k \]

where \[ load_k = access \_ rate_k \times s_k \]

• \( latency_k \) = the latency that the object \( k \) produces if it is replicated to the replication server which has been determined by the previous step
• \( load_k \) = total load of object \( k \)
• \( access \_ rate_k \) = number of accesses of object \( k \) per unit time
• \( s_k \) = the size of object \( k \)
• Only the object with the highest “utility value” is replicated to the server selected in phase 1, afterwards phase 1 starts again
• During the next iteration, servers already containing an object are excluded from the optimal placement selection in phase 1 for this object
• As a result of the algorithm objects with high loads may be placed to various replication servers

Potential result: Objects with high loads are assigned to various replication servers

These replication servers are selected reducing the latency during object access
Excursus: Domain Name System

- The Domain Name System (DNS) is a distributed, hierarchical database enabling the translation of domain names to IP addresses.
- Every name server contains a set of databases ("zone files") describing translation rules ("resource records") for domain names it is assigned for.
- Various types of resource records are available, such as:
  - Type A: describes a mapping between a domain name and an IPv4 address.
  - Type NS: maps a domain name to another name server assigned for this domain.
  - Type CNAME: specifies that the domain name is an alias of another - the DNS lookup will continue by retrying the lookup with the new name.
- Example resource records:
  
<table>
<thead>
<tr>
<th>Resource Record</th>
<th>IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>mail.example.com. A</td>
<td>141.76.40.23</td>
</tr>
<tr>
<td>ftp.example.com. A</td>
<td>141.76.40.24</td>
</tr>
</tbody>
</table>

- The CDN "Akamai" uses the CNAME mechanism to route requests into the Akamai network and thus to Akamai replication servers.
Akamai

- Akamai is the largest CDN worldwide with more than 15,000 replication servers deployed in 60 different countries

Contains CNAME entry for www.example.com

Query: www.example.com

Remote DNS Server

2

Query: www.example.com

Remote DNS Server

3

4

a1312.g.akamai.net

Akamai root DNS server

5

za.akamaitech.net

Akamai high-level DNS server

6

n0g.akamai.net

Akamai low-level DNS server

7

8

Return IP address of replication server

Local DNS Server

Local network

http://www.example.com

Replication / Edge server

Replication / Edge server

Replication / Edge server

Akamai network
For resolving domain names to IP addresses, a local domain server is accessed in a first step. The request is forwarded by the local DNS server to a server on a higher hierarchical level which maps the requested domain to another domain (due to a CNAME entry in the resource records). The newly received domain (www.example.com.edgesuite.net) is resolved via a further name server that directs the request to a domain server deployed by Akamai. The Akamai domain server forward the request to a domain server of a sub-network assigned responsibility to serve the request. The IP address of the responsible replication server is returned to the local DNS server thus the user client can send a request to this server in step.
Request Routing System in Akamai

- Forwarding requests to a replication server is done in a two-stop-approach for scalability purposes:
  - After selecting an appropriate cluster close to the user / his local DNS server, a concrete replication server is determined within this cluster.
- Selection of cluster is based on IP address range of user / local DNS server and various historic and real-time data gathered from a monitoring system (network load, log files, ...) and based on configuration options.
- Selection of a concrete replication server is done on historic and real-time monitoring data plus configuration options.

![Diagram of Request Routing System in Akamai](image-url)
Community Networks

- Community networks are networks mainly formed by decentralized users contributing to the services and the content available in such a network.
- In community networks, the decentralized characteristics of the Internet are explored and means are applied to enable a direct interaction of end user systems.
- For this purpose peer-to-peer-based protocols are used.
- Thus, in sum network infrastructures result which combine CDN systems with peer-to-peer infrastructures.

...
Load Balancing

• To ensure high availability and performance of Web applications, load balancing between replicated instances of a Web server is applied

• Example for easily applicable approach:
  • *DNS Round Robin*: IP addresses of redundant servers are published as A records for one domain; DNS server delivers addresses in changing order (“round robin”)

![Diagram showing DNS Round Robin with example IP addresses for www.example.com]

• Selected drawbacks if DNS-based approach is applied without extension:
  • Does not enable dynamic delegation to servers depending on their current load
  • For every redundant server, a public IP address has to be provided
  • DNS replies are cached on client side and thus follow-up requests are always sent to the same server

→ Drawbacks are avoided by applying proxy-based approaches
Proxy-based Load Balancing

- Approach: Software proxy ("front end") intercepts requests and delegates them to one out of a group of identical servers ("back ends") running the Web application.

  - Balancing algorithm determines which server to select (e.g. round robin or priority-based selection).
  - Delegation is realized using Network Address Translation (NAT) or on application level.

  • Example configuration for front end and back ends based on syntax of HAProxy (http://www.haproxy.org/)

    ```
    frontend www
    bind *:80
    acl url_blog path_beg /blog
    use_backend blog-backend if url_blog
    default_backend web-backend
    
    backend blog-backend
    balance roundrobin
    mode http
    server blog1 192.168.0.10:80 check
    server blog2 192.168.0.11:80 check
    ```

    - Balancing strategy
    - Health of server is continuously checked by proxy
Elastic Load Balancing

- Elastic Load Balancing / autoscaling enables automatic creation or termination of virtual machines (VMs) based on the system’s or application’s state.
- Implementation examples:
  - Amazon Elastic Compute Cloud
  - OpenStack Heat

Virtual machines report in regular intervals selected load/usage statistics (e.g. memory).

Typically provided via configuration file.

<table>
<thead>
<tr>
<th>Physical Machine 1</th>
<th>Physical Machine 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM 1: Web Application</td>
<td>VM 2: Web Application</td>
</tr>
<tr>
<td>VM 3: Web Application</td>
<td>VM 4: Web Application</td>
</tr>
</tbody>
</table>

Delegation of requests to virtual machines is done via Load Balancing Proxy.

Create/delete virtual machine

Update proxy configuration
Conclusion

- Various protocols are available for inter-cache communication and Web content distribution, e.g.:
  - Internet Cache Protocol (ICP)
  - Web Content Distribution Protocol (WCDP)
- Possible consistency guarantees:
  - Strong consistency
  - Delta consistency
  - Weak consistency

Central problems regarding caching solutions:
- Caches are only responsible for selected Web servers or serve only customers of one Internet Service Provider
- Content is not cached proactively – only Web resources requested in regular intervals are located in proxy caches

**Content Distribution Networks** solve this problem

- Core mechanism: Redirection of users’ requests to one of the replication servers (e.g. by using CNAME entries in DNS servers)
  - Mechanism applies as well for streaming CDNs
- CDNs combine strategies of caching content and replicating it proactively

For achieving high availability and performance of Web applications, **Proxy-based Load Balancing** can be applied and potentially extended by automatic scaling mechanisms (**Elastic Load Balancing**)
References


Website of the Web Cache “Squid”: http://www.squid-cache.org/

Book about CDNs: “Content Delivery Networks”, Lecture Notes in Electrical Engineering, Vol. 9 Buyya, Rajkumar; Pathan, Mukaddim; Vakali, Athena (Eds.), 2008


Website of Akamai Technologies, Inc.: http://www.akamai.com/

Amazon EC2 Load Balancing: http://aws.amazon.com/de/documentation/elasticloadbalancing/

OpenStack Heat: https://wiki.openstack.org/wiki/Heat