Cooperative Web Cache

Tomas Cerny, Petr Praus, Slavka Jaromerska, Lubos Matl
Faculty of Electrical Engineering, Czech Technical University
Charles square 13, 121 35, Prague, Czech Republic
Email: {tomas.cerny, prauspet, jaromsla, matllubo}@fel.cvut.cz
Jeff Donahoo
Baylor University - ECS
One Bear place, 76798 Waco, TX, USA
Email: jeff_donahoo@baylor.edu

***Abstract*—Internet web service delivery expectations grow every year but the underlying technology does not scale naturally to deal with greater demand for services. A simple service, once becoming popular, requires the service provider to invest in powerful hardware in order to deal with sudden spike in client interest. A standard solution involves employing a content distribution network resulting in spiralling costs with increasing load. In this paper we argue it is possible to scale web service delivery more naturally assuming clients take part in content replication. We propose a design and a prototype implementation of a P2P overlay network that shows optimistic preliminary results. An extensive simulation is provided in order to establish performance possibilities of such a network with no extra related costs or demands on hardware.**

***Index Terms*—Peer to peer computing, Communication networks, Web and internet services**

I. Introduction

Contemporary web content distribution model places the entire burden on the service provider. With increasing popularity of his content, service provider is forced into perpetual upgrade cycle or has to invest into a content distribution network. In both cases he effectively buys two commodities – bandwidth and hardware, which a client has in abundance. We propose engaging the client in distribution of content she herself consumes. In addition our solution is locality-aware. It prefers retrieving content from nearest available neighbors because local bandwidth is relatively cheap. Apart from lowering server load we aim for decrease in content download time for the client. We naturally preserve autoscaling ability of modern peer-to-peer networks such as BitTorrent[1]. The more popular the content becomes the more available it is.

II. Proposed solution

In this paper we propose an extension to the established web service delivery scheme in the form of cooperative low latency P2P network that can effectively locate the closest neighbor with the desired content. The overlay network consists of collaborating nodes which are consumers and providers at the same time. Server unaware of existence of such an overlay will experience decreased load, while a client with the overlay membership will experience decreased download time. To attain this behavior two network mechanisms are essential – anycast for content discovery and multicast for content invalidation. We build on top of peer-to-peer application substrate Pastry [2] that can effectively form a decentralized and self-organizing overlay network. Abovementioned network mechanisms are achieved by Scribe [3], [4] based on Pastry. Scribe is a scalable application-level multicast/anycast infrastructure. It supports large numbers of groups, with a potentially large number of members per group. For each cacheable *unique* content object (e.g. a file) we create a group and any node capable of supplying the given object becomes its member. A group is identified by object's hashed URL. These underlying technologies have a topology-aware routing – anycast requests are served by the topologically close nodes. They also guarantee message travels at most through 4 nodes in average case.

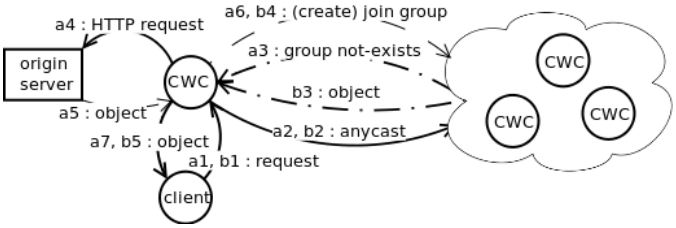

Fig. 1. Cooperative Web Cache request life-cycle

The life-cycle of CWC is shown in Fig. 1. It recognizes static (inherently cacheable) and dynamically generated, non-cacheable content objects. In the simplest case client requests dynamic object, CWC recognizes it as such (currently based on file extensions) and retrieves it directly from the server. Request (a1, b1) recognized as for static object is anycasted [4] (a2, b2) to the CWC network. If it is not contained in the network (a3) the requesting node downloads the object from the server (a4, a5) and creates and joins (a6) a new group [3]. If the node retrieves the object from the CWC network (b3), the group already exists and node only joins (b4). In both cases the node becomes a provider (a7, b5) of this object.

III. Case Study

Our research based on Alexa Top Sites list [5] of world's most frequently visited sites suggests that average contemporary page consists from about *90% of static and relatively*

TABLE I
A TYPICAL SETUP OF TESTING ENVIRONMENT – SERVER AND NODE BANDWIDTH

Type	Download [Mbps]	Upload	#Tests
Server	100	100	–
USA	3	0.6	–
Prague, CZ	9.3	2.2	6467
UPC ISP, CZ	10.5	1.3	13195

easily cached content such as images, stylesheets and scripts. This number is based on analysis of top 500 websites appearing on Alexa Top Sites on January 1, 2011. All our testing pages respect this ratio.

Median time period for establishing a TCP connection (round-trip time) with the top 500 websites through a backbone connection located in Prague, Czech Republic was 114 milliseconds. There seems to be no research on latency of real world consumer links. Therefore we chose 20 milliseconds based on presumption that most traffic will be between geographically clustered nodes that are selected by anycast with distance metrics. Based on these observations we consider a testing environment with the RTT values of 114 ms for server-to-peer and 20 ms for peer-to-peer to be a typical setup. However, server-to-peer measurement is based on a sample of a relatively popular websites that almost always have a global presence – usually in the form of content distribution network. This could skew RTT measurements a little bit.

Consumer-level link speeds are somewhat better documented, but they can vary significantly based on locations of such measurements or type of connectivity. Our testing environment uses three sets of bandwidth values. The value for the U.S. household is based on Speed Matters report [6]. Average bandwidth across all providers in Prague, Czech Republic and bandwidth average of state-wide provider UPC are taken from speed measuring website rychlost.cz on January 24th, 2011. Summary of all bandwidth values can be seen in Table I. Based on brief research of several commercial offers for server placement in data centers we decided to cap speed of server link in simulation to 100 Mbit for upload and download.

A. Testing environment

Our testbed environment is built around an idea of *Node Manager* which proceeds by the predefined script dictating how an environment and its nodes should behave at a certain time. In our case, environmental behavior mainly consists of network conditions. Two networking modules of Linux kernel proved to be essential for our simulated environment. The first is *NetEm* [7] that allowed us to artificially create delay, packet loss and many other conditions commonly occurring in computer networks. The second is *ifb* which allowed us to shape ingress traffic. *Node Manager* itself does not make requests to its nodes. Instead, it controls remote *Node Tester* over the network. The reason is twofold. First, making requests should be logically separated from managing nodes and their environment which is what *Node Manager* does. The second reason is related – it puts additional load on the testing machine which is undesirable as it would skew our figures.

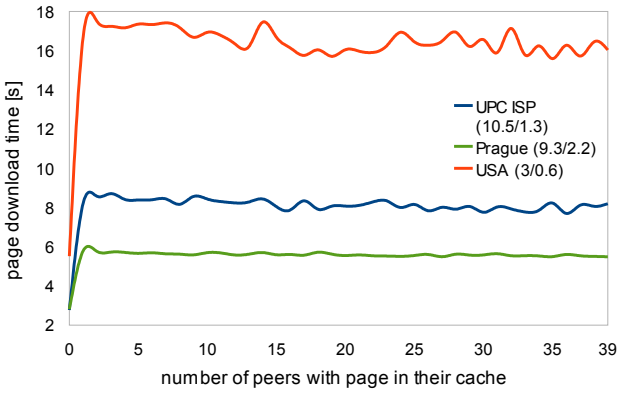

Fig. 2. Progressive download of 40 peers

B. Optimistic experiments in homogeneous network

In this set of experiments we set up a homogeneous network – bandwidth of all peers as well as RTT between each couple of peers are identical. Geometrical representation of peer distribution based on delay would be a circle with peers on its perimeter. The diameter of the circle represents the delay between any two peers, because all communication passes through the circle's center. Server would be placed on a circumference centered in the same point but with a different radius.

1) Progressive download: In this experiment we initialized a network of 40 CWC peers that successively downloaded the web page. Values on axis x in Fig. 2 represent number of peers with all sources downloaded already in their local caches. Logically, the first peer (x = 0) could not download the data from the CWC network as there were none. Instead it had to download all (dynamic as well as static) sources from the server. Axis y represents total download time of a peer. Fig. 2 assumes that low bandwidth of peers creates a significant bottleneck of a CWC network. The first Prague peer and UPC peer were of the same speed because their download capacity was nearly identical. Later peers of these setups had the same download conditions, however, they differed in upload significantly. Due to the fact they were downloading static sources from CWC network the higher upload capacity gave major advantage to the Prague peers.

2) Server load: The previous experiment is focused around peer's point of view. However, there is second party – a strained server. We represent the server load by two variables, the number of hits and downloaded data over the duration of the test. Forty peers participated in the test where every second, one peer started downloading the page. To compare results we also performed this test without CWC with all requests going straight to the server. Fig. 3 shows data downloaded from the server during the test. The green curve suggests that once the data is stored in the cooperative web cache the server

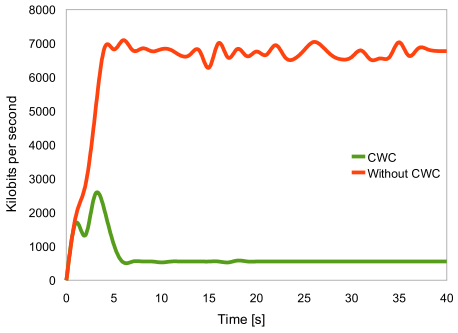

Fig. 3. Server load - data downloaded from the server

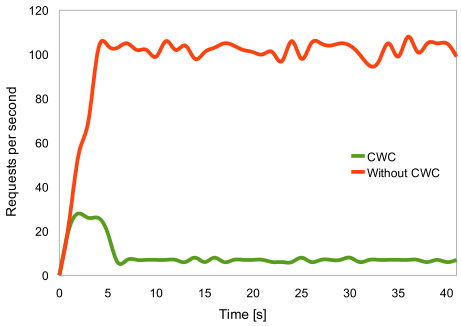

Fig. 4. Server load - number of requests made to the server

load declines by 90%. This figure coincides with the ratio of dynamic content on a web page discussed above. Fig. 4 showing the number of requests to the server supports this idea. It should be noted the amount of downloaded data during the first five seconds roughly corresponds to the size of the page and its resources. This results from the fact the first peer has already downloaded some resources when further peers placed their queries and was already able to offer the resources.

3) Server delay dependency: Motivation for this test is discovering the boundary of server distance with which it is advantageous (performance-wise) to use CWC. For each bandwidth setup from Table I we compared download times of a client downloading directly from the server and a CWC peer downloading from a network of 10 CWC peers. Fig. 5 shows that for Prague-like bandwidth it is useful to download page cached in a CWC network from server-to-peer distances greater than 170 milliseconds. The same applies to UPC ISP and USA bandwidths with RTT 250 ms and 490 ms respectively. Fig. 6 shows RTT values from Alexa TOP500 mentioned above. The three horizontal lines represent efficiency boundaries for different setups. Access to websites above the lines could be improved by using a CWC network. The percentage of improvable websites by Prague, UPC ISP and USA bandwidth setups is 34%, 22% and 6.4% respectively.

4) Resilience to abrupt departure of peers: We simulated departure of 5% and 10% peers from a network of 50 peers with all static sources in their caches. To mimic the worst

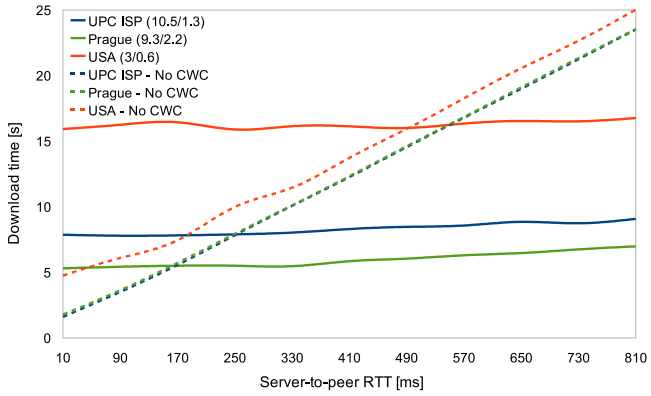

Fig. 5. Server delay - dependency between peer-to-server RTT and speed

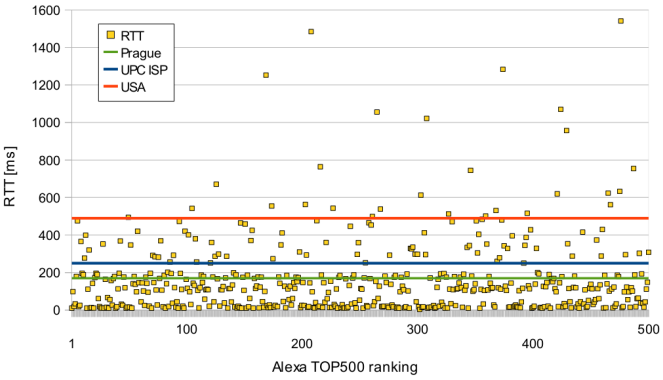

Fig. 6. Websites with improvable RTT values by different CWC setups

possible case, the nodes left at the very same moment. We cut off the peers harshly by preventing operating system's TCP queue to close the link correctly (no TCP FIN packet has been sent) to simulate abrupt disconnection event. An instant after the departure a new peer joined the CWC network and downloaded the page. We measured the time of downloading. Table II shows the average time of ten measurements and its standard deviation. The referential download times were measured in a healthy CWC network with the same number of peers - for 5% and 10% departure 48 and 45 peers respectively.

TABLE II
RESILIENCE TO ABRUPT DEPARTURE OF PEERS

Network	Time [ms]	Deviation [ms]
5% DEPARTURE		
Healthy	8263	95
After departure	8756	556
10% DEPARTURE		
Healthy	8339	168
After departure	8999	667

IV. Related work

Similar proposal to CWC is Squirrel [8]. It also tries to engage clients in content distribution and also builds on Pastry. However, it aims only for corporate LAN networks and lacks anycast and multicast capabilities. Scribe [3], [4] which implements these capabilities on top of Pastry [2] allows us to implement efficient invalidation/update mechanism (multicast) and better scaling of popular files using anycast. Squirrel node, hosting a popular file, is bound to be eventually overloaded because all traffic for that given object is routed through a "home node". This also applies to the "directory mode" where the home node keeps reference to a certain number of it's latest clients and redirects new clients' requests to them. This approach just holds off the inevitable because even if the home node will not became overloaded with requests, it will be at some point overloaded with redirect responses. Dalesa [9] also aims to provide web content caching for LAN networks but they have working prototype software. Kache [10] is very inspiring work in terms of lookup performance, its authors present a method reducing the number of necessary hops between nodes to just one hop. They do this by using $O(\sqrt{n})$ space on each node (where n is the number of nodes) and a probabilistic approach to content retrieval – a node has a certain low probability it will not be able to fulfill a request for the content it previously advertised. Finally, Survey for Web caching [11] (section 4.1.2) contains some early (1999) thoughts how a distributed caching of web content might look like but in the spirit of the time it mainly focuses on exchanging content between institutional and national level caches.

V. Summary and Future Work

In this paper we suggest that Internet web services take advantage of peer-to-peer overlay network that is formed by users with similar interest in the given time frame. We have presented simulations which are relatively close to the real world conditions and prove our concept is feasible in two regards – consumer peers may experience improved download times for relatively common cases and service providers attain large savings and are more apt to survive peak loads. We are aware that CWC concept is missing some key properties. Namely security, old content invalidation or more effective object discovery strategies. These are subject to further research.

There are security considerations unique to P2P networks because nodes are much more powerful. They issue their own node IDs, act as routers and relay messages and issues with trust arise. Uniform random distribution of node IDs is fundamental operational presumption of Pastry and all similar networks. If an attacker can *choose node ID*, she can surround a victim node, isolating it from the rest of the network, partition the network into smaller pieces or become a root key holder. This issue could be solved by centralizing node ID issuing into the hands of trusted CA. Miguel Castro et al. described some possible solutions in [12].

There are a several possible approaches to checking freshness of a cached content. Each has its benefits and downsides but it always boils down to a compromise between hitting the server or risk having an outdated file. All have in common that *once a single peer in a group notices an outdated file it notifies the whole group with multicast message* containing hash of an updated content. They also have in common that such message might be forged by an attacker attempting to erase the content from the network to overwhelm the server. The network can employ various tactics to mitigate such threats (see FreeNet, [13]).

Objects are grouped around their source web page and it is likely that a client, who has one such object, will have more. This inspires us to consider implementing another object discovery strategy based on grouping objects in packages by their source web page. Client could then ask for certain chunks of these packages. It might be considered to apply a mechanism for deciding whether it is meaningful for a peer to use the CWC network or direct service provider. This could be implemented based on Fig. 5 which shows the limits for certain peer bandwidths and server delays. Server delay can be readily measured while downloading the dynamic content of the page. Neighboring peers' bandwidth availability could be established by a statistic of their past behavior.

Future work also includes better simulation environment particularly in terms of scale and improving emulation fidelity. We were able to simulate at most about a fifty nodes with our hardware. More extensive environment (e.g. PlanetLab) would better reflect viability of our proposal in practical terms.

References

[1] B. Cohen, "Incentives Build Robustness in BitTorrent," 2003.

[2] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, ser. Middleware '01. London, UK: Springer-Verlag, 2001, pp. 329–350.

[3] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 8, pp. 1489–1499, 2002.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scalable Application-Level Anycast for Highly Dynamic Groups." Springer-Verlag, 2003, pp. 47–57.

[5] "Alexa top sites," http://www.alexa.com/topsites.

[6] "Speed matters, 2010 internet speeds report," http://www.speedmatters.org/2010report.

[7] S. Hemminger *et al.*, "Network emulation with NetEm," in *Linux Conf Au*, 2005.

[8] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A decentralized peer-to-peer web cache," in *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. ACM, 2002, pp. 213–222.

[9] "Dalesa: The Peer-to-Peer Web Cache," http://www.dalesa.lk/.

[10] P. Linga, I. Gupta, and K. Birman, "Kache: Peer-to-peer web caching using kelips," *In submission*, 2004.

[11] J. Wang, "A survey of web caching schemes for the internet," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 5, pp. 36–46, 1999.

[12] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, "Secure routing for structured peer-to-peer overlay networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 299–314, 2002.

[13] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Designing Privacy Enhancing Technologies*. Springer, 2001, pp. 46–66.