Towards a Smart, Self-scaling Cooperative Web Cache

January 10-12, 2011

Tomas Cerny[1], Petr Praus[2], Slavka Jaromerska[2],
Lubos Matl[1], and Michael J. Donahoo[2]

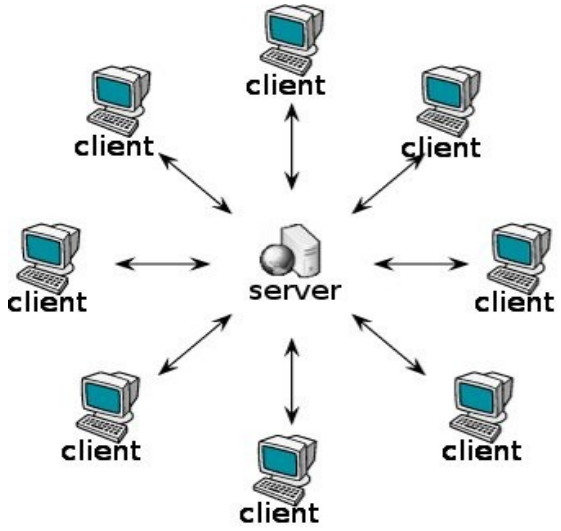
[1] Department of Computer Science and Engineering,
Czech Technical University

[2] Department of Computer Science,
Baylor University

SofSem 2012 – Spindleruv Mlyn, Czech Rep.

Current status

- HTTP protocol
- Client-server architecture
  - one provider
  - many clients



- Disadvantages
  - bottleneck? Server
    - more users => slower response
  - different distance for clients from the server
    - slower download for remote users

Existing solutions

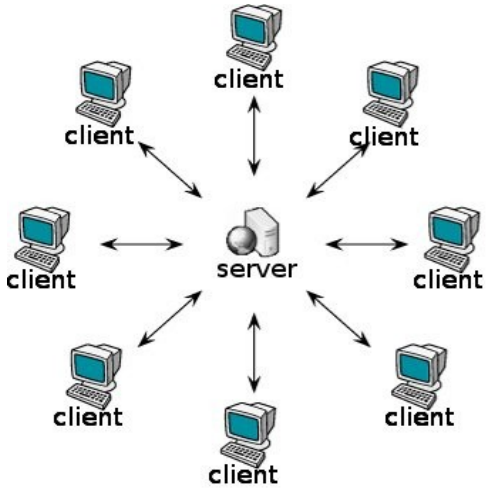
- More servers
  - dividing the load by using a proxy or DNS
  - these servers should be distributed through the world
    - expensive
- Services of third part providers (CDN)
  - large numbers of powerful hardware
  - Akamai, EdgeCast
    - very expensive
- Cloud services
  - expensive, responsibility
- CWC
  - no additional hardware

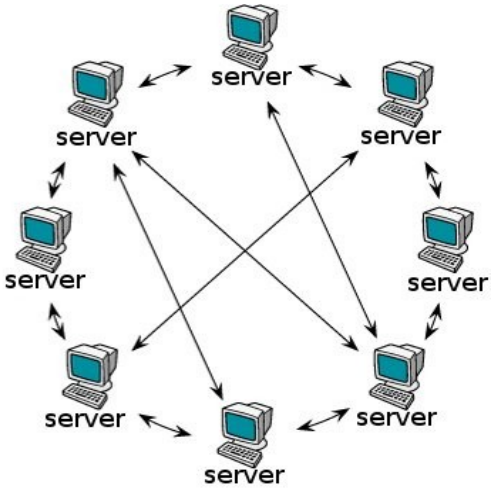
Goals of CWC

- scalability
- no aditional hardware
- no need for CDN services
- acceleration on the client side

How to achieve this

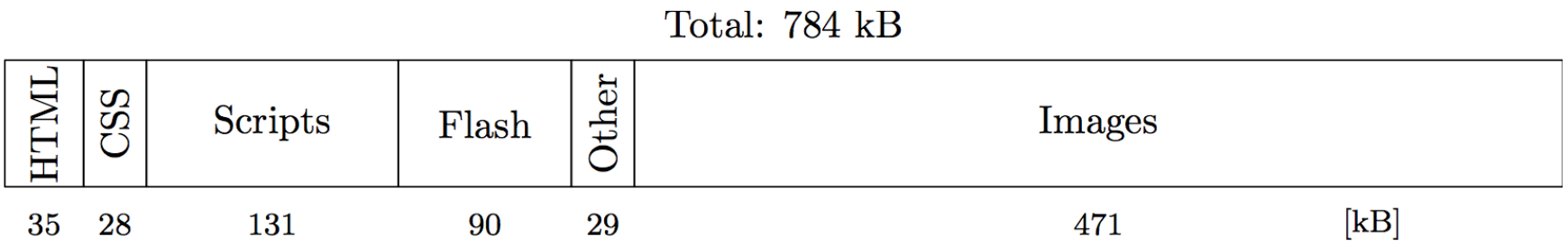
- users become a provider
- use a very small amount of user performance
- main advantage
  - more users = more providers (natural scalability)





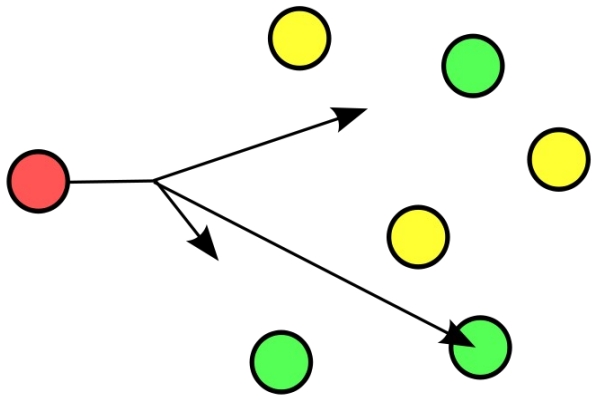
How it works

- Web page consists of two types of content:
  - Static (CSS, Images, JavaScript) - about 90% of the site
  - Dynamic (HTML)
- Each client has own cache
- Static content can be stored at the cache
- Client provides data from cache to others

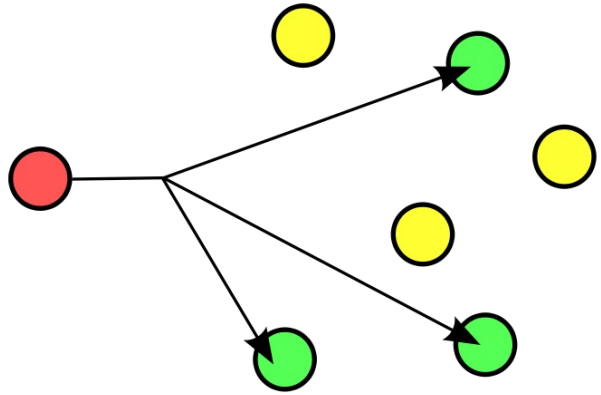


Technologies

Pastry
- defines P2P network
- scalable, decentralized
- provides object location and routing



Scribe
- extends Pastry
- groups
  - anycast
  - multicast
  - *manycast*



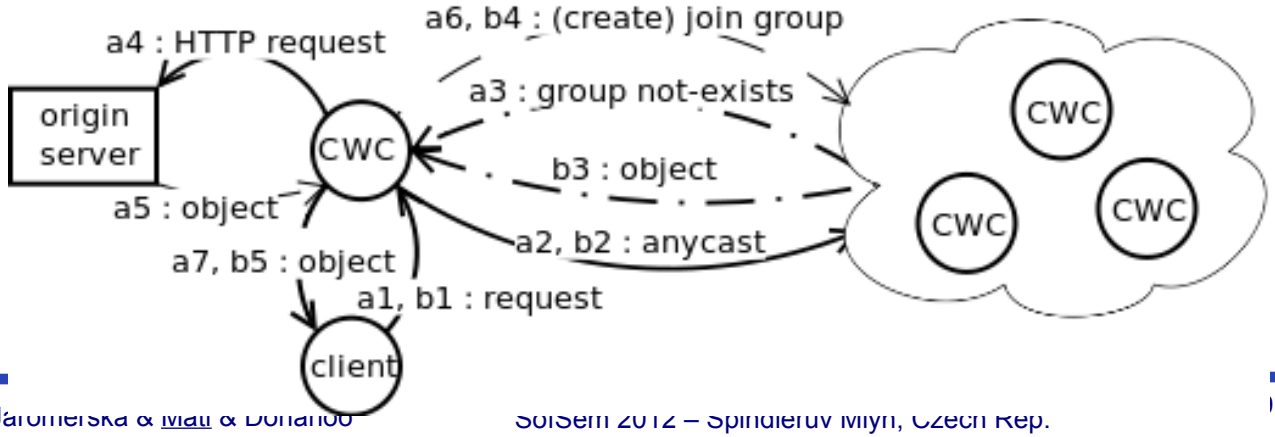
Scribe

- Extends Pastry with groups
- Pastry nodes can join to groups and share the same information

- Anycast is message which receive one node from the group
  - this node is not selected completely at random
  - used metrics
    - *bandwidth, delay (RTT)*

Simple method

- file URL=> hash
  - URL is unique
- Scenario (for downloading file)
  1. client sends Anycast to group hash(file-url)
     - if Anycast returns with status *failed* group doesn´t exist and client downloads file from server
     - else client recieves file from someone in group
  2. client stores file to its local cache
  3. and joins to group - if group doesn´t exist it will be created



Improved method

- additional group is created for entire web page
- each node provides files list of web page

- site download algorithm:
  - first request for static file:
    1. download website file list
       - send anycast to group with hash(web-site-url)
    2. if group doesn't exist client must download whole web site from server
    3. start downloading all files from file list
       - prioritize files that were requested
    4. join to web-page group and to groups with all downloaded files

Improved method

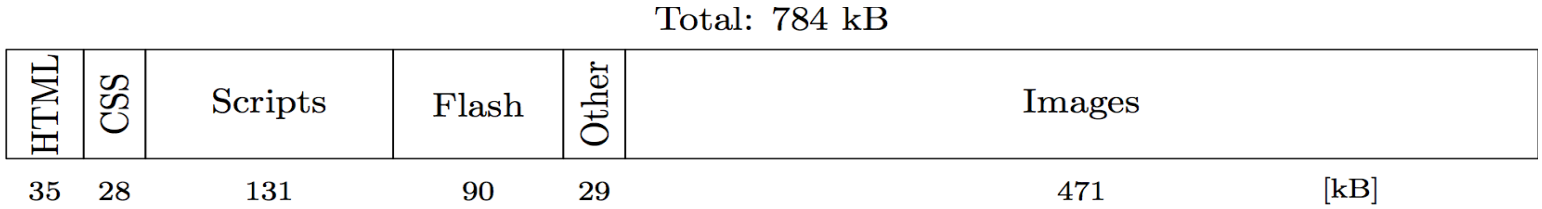
- Advantages
  - some files are downloaded before browser ask for files
  - quick discovery for bunch of files may leads to download all files from web server without asking CWC (web-page group doesn't exists probably means that groups with files also)
  - files list can contain additional information (e.g. for acceleration of download)

Communication with web browser

- HTTP Proxy
  - advantages
    - simply
    - universal
    - easy for evaluation
  - disadvantages
    - additional information only in HTTP headers
- Browser plugin
  - advantages
    - more information from browser
    - can use browser cache
  - disadvantages
    - browser dependency
    - unsuitable for evaluation

Evaluation - settings

- Testing page
  - Google statistics
  - evaluation Alexa top 500 pages



- Environment

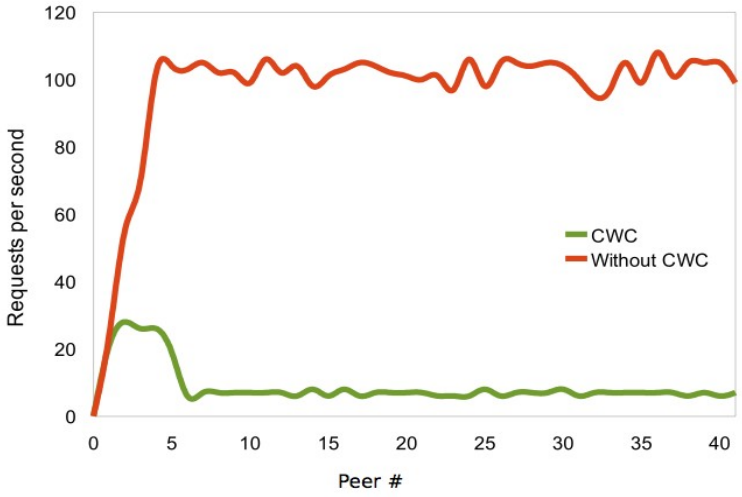
Type	Download [Mbps]	Upload
Server	100	100
USA	3	0.6
Sweden	22.2	4.5
Japan	18.0	7.0
Prague, CZ	9.3	2.2
UPC ISP, CZ	10.5	1.3

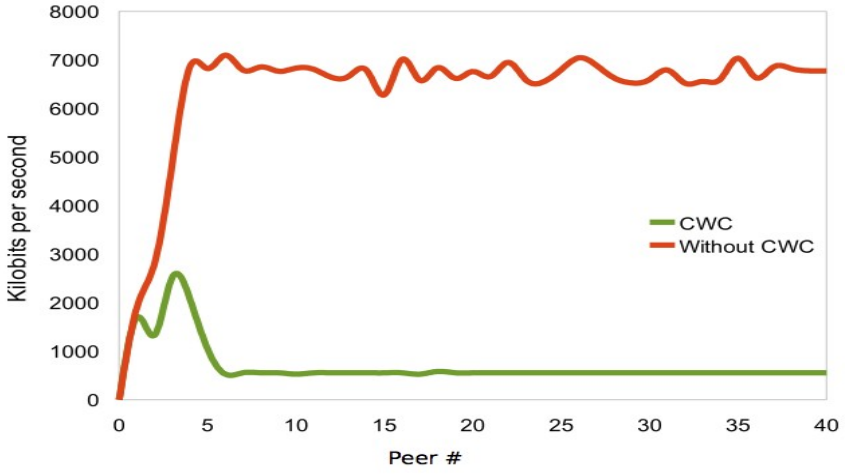
Evaluation - server

- without CWC
  - more nodes => greater load on the server = slower download

Number of clients	1	20	30	40
Download time [ms]	6311	6860	6939	7297

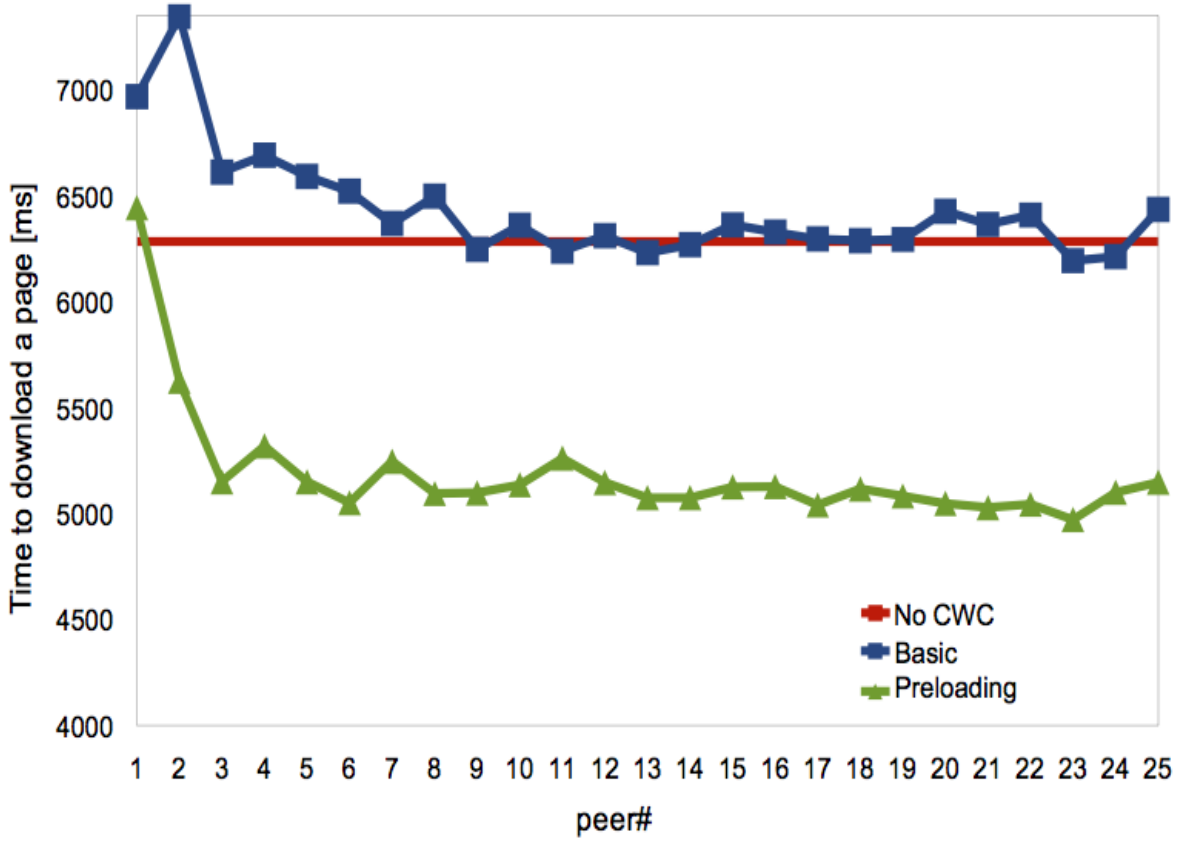
- with CWC
  - more nodes => more providers





Evaluation – simple vs. improved

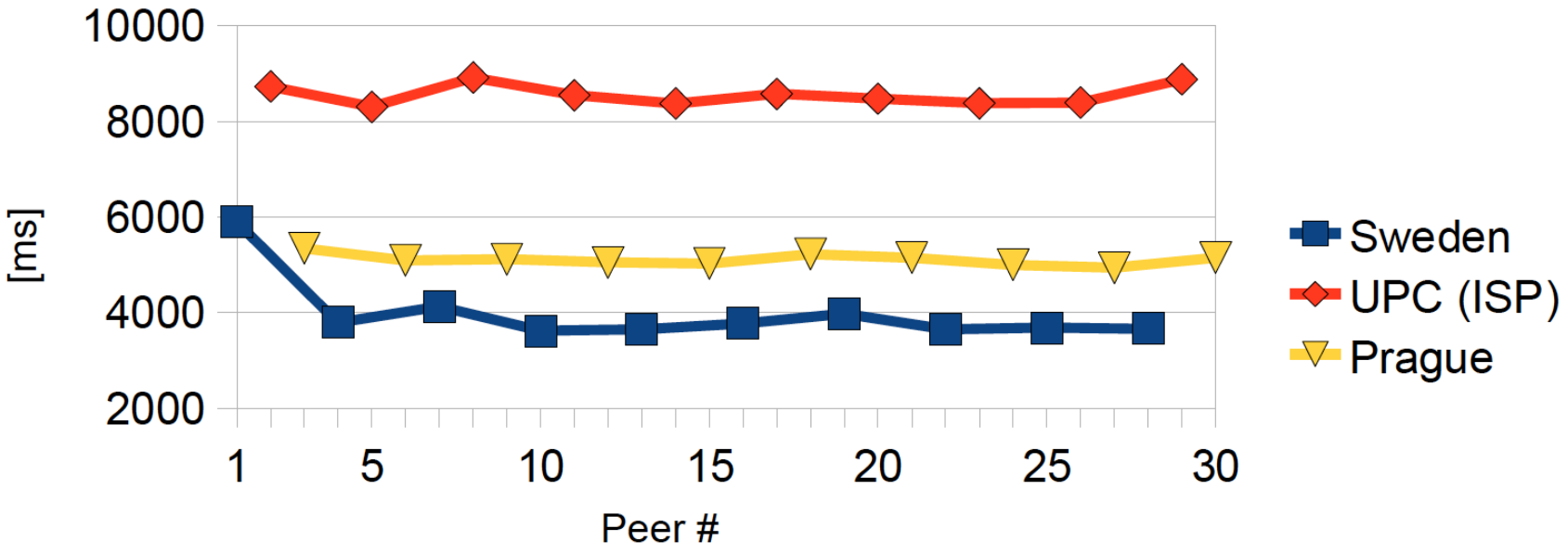
- Simple
  - 6.3 sec
- Improved
  - 5.2 sec



delay - 114 ms

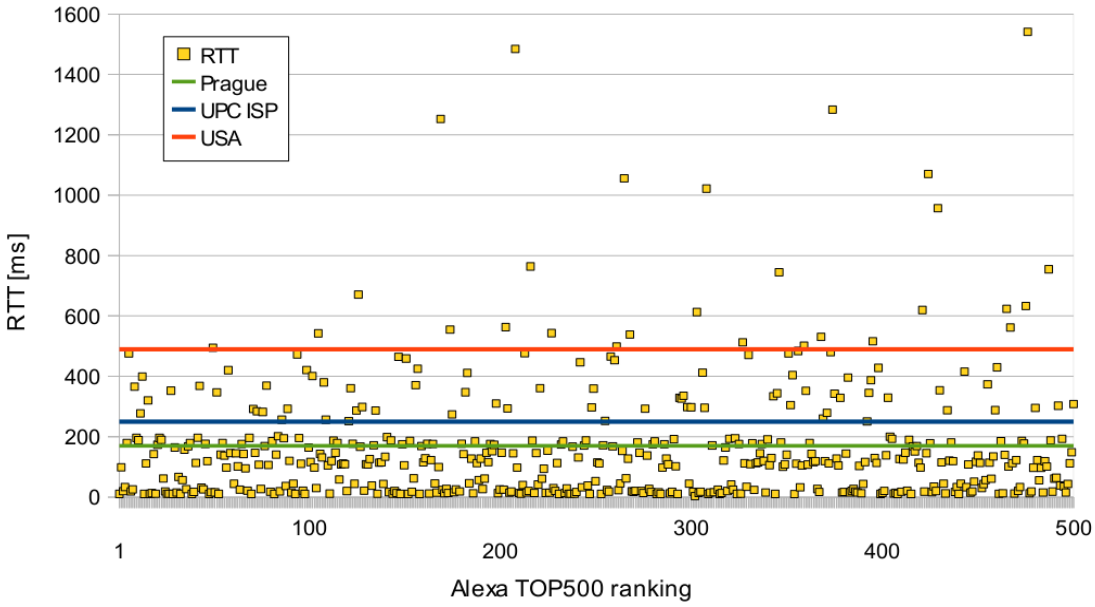
Evaluation - heterogeneous network

Location	Download	Upload	Latency [ms]				Web page load time
	[Mbps]	[Mbps]	Sweden	Prague UPC	Prague	Server	[ms]
Sweden	22200	4500	20	40	30	114	4303
Prague UPC	10500	1300	40	60	50	134	9497
Prague	9300	2200	30	50	40	124	6311



Conclusion

- CWC reduces server load
- faster page download on the client side
  - number of pages downloaded faster with CWC
    - Prague 34%
    - UPC ISP 22%
    - USA 6.4%



Future work

- issues to solve
  - security
  - data freshness
- accelerate download time
  - improve selection of the node with anycast
  - manycast and multicast (one file - more provider nodes)
  - connect the server to the peer to peer networks
- more general use => ELISA
  - Streaming media
  - SWWS

Thank you for your attention
Questions?