

WWW Server Optimizations

John-Patrick Wowra

Email: JohnPatrickWowra@web.de

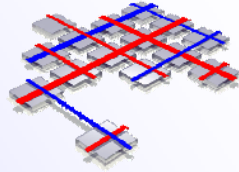
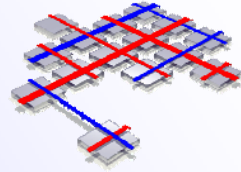
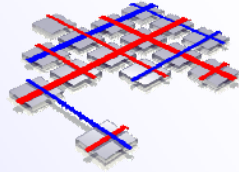


Table of Contents

- Introduction
- Segment Based Proxy Caching for Distributed Cooperative Media Content Servers
- Performance Issues in WWW Servers
- Conclusion

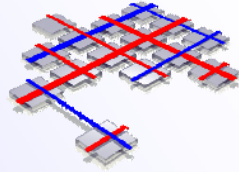


Introduction



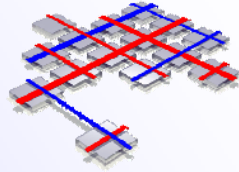
Introduction

- Problem:
 - Extreme Growth of live media services
 - Audio & Video Streaming
 - Limited Capacity of network
 - Significant increases in:
 - Latency
 - Network congestion for Internet Applications



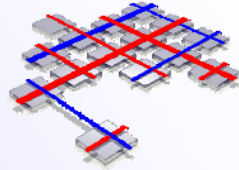
Organization

- Global View:
 - Segment Based Proxy Caching for Distributed Cooperative Media Content Servers
 - Simulation on:
 - Proxy Servers
 - Distributed Cooperative Web Servers (DCWS)
- Detailed View:
 - Performance Issues in WWW Servers
 - Evaluation techniques for improving OS and network protocol software support
 - New Socket Functions
 - Per-byte optimizations
 - Per connection optimizations



First Part

Segment Based Proxy Caching for Distributed Cooperative Media Content Servers



Segment Based Proxy Caching

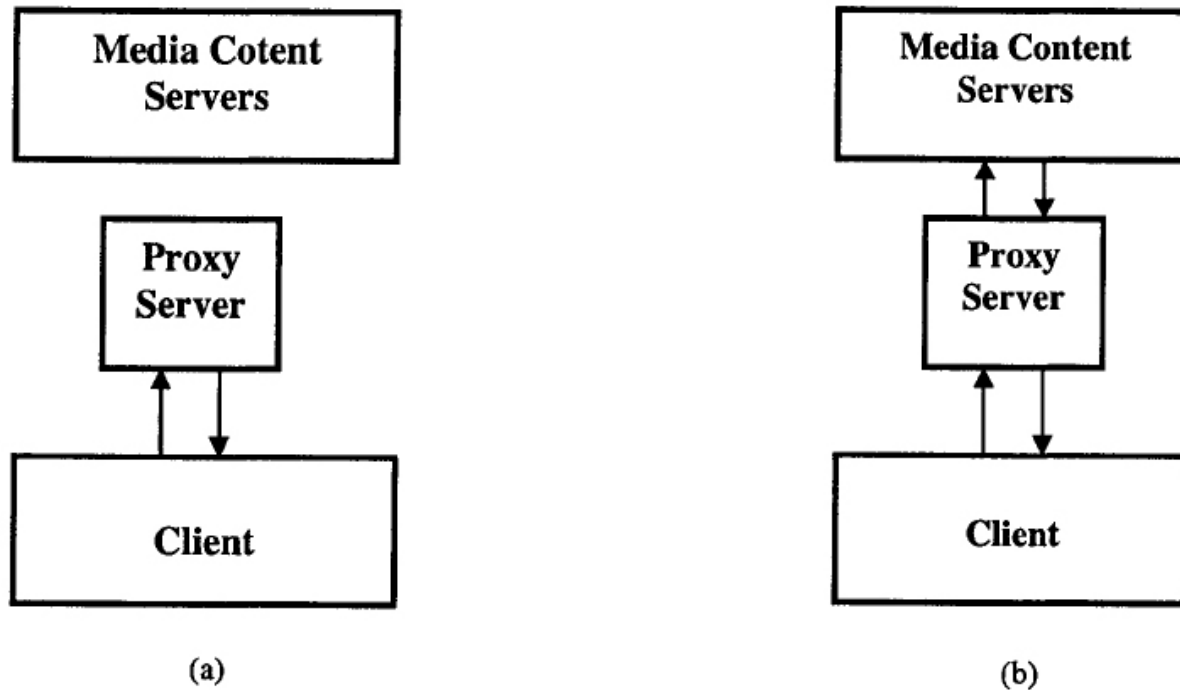
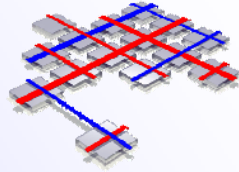
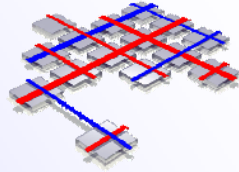


Figure 1. Flow-diagrams for proxy-based services. (a) The proxy server contains the document, the document is directly sent to the client. (b) The proxy needs to contact the content servers to obtain the document.



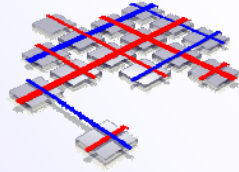
Segment Based Proxy Caching

- Latency between proxy & client small in comparison to the latency between proxy & content server
- Proxy Server must contain enough initial blocks to avoid a delay
- Not yet cached media can be pre-fetched
- Storage space needed to store a whole media file too large
- Not a realistic solution for performance improvement



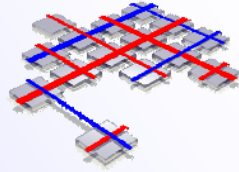
Segment Based Proxy Caching

- Idea:
 - Divide the media files into segments
 - Perform a weighted caching on segments



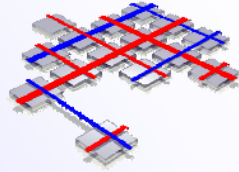
Segment Based Proxy Caching

- Division of media files
 - Simple segmentation method is used
 - Divides media file into multiple equal sized blocks of transmission units
 - Proxy groups multiple blocks to a segment
 - The size of a segment is sensitive to the distance to the media file
 - Number of blocks in segment n is equal to $2^{(n+1)}$
 - Segment (n) has twice the size of segment $(n-1)$



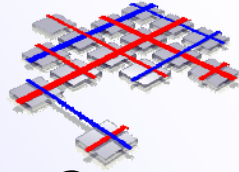
Segment Based Proxy Caching

- Cache admission policy/ control
 - Idea: Fill the cache with most frequently used segments from media objects.
 - Different criteria are applied to the prefix part of an object; e.g. the segment number
- Cache replacement policy
 - Least recently used segments are removed from the stack
 - Factors like caching weight of a segment/ distance etc. are used to decide this



Distributed Web Server

- Goal of Distributed Cooperative Web Servers (DCWS):
 - Find application level techniques for distributing web content.
 - Hyperlinks, stored within the documents are manipulated
 - Several conditions and factors shall be considered to determine, which documents should be migrated
 - An algorithm is developed to decide, which documents to migrate



Algorithm for Document Migration

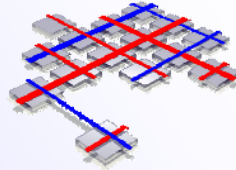
Algorithm *Document selection for migration*

Input: Given a local document graph of a home server, and a threshold T of load

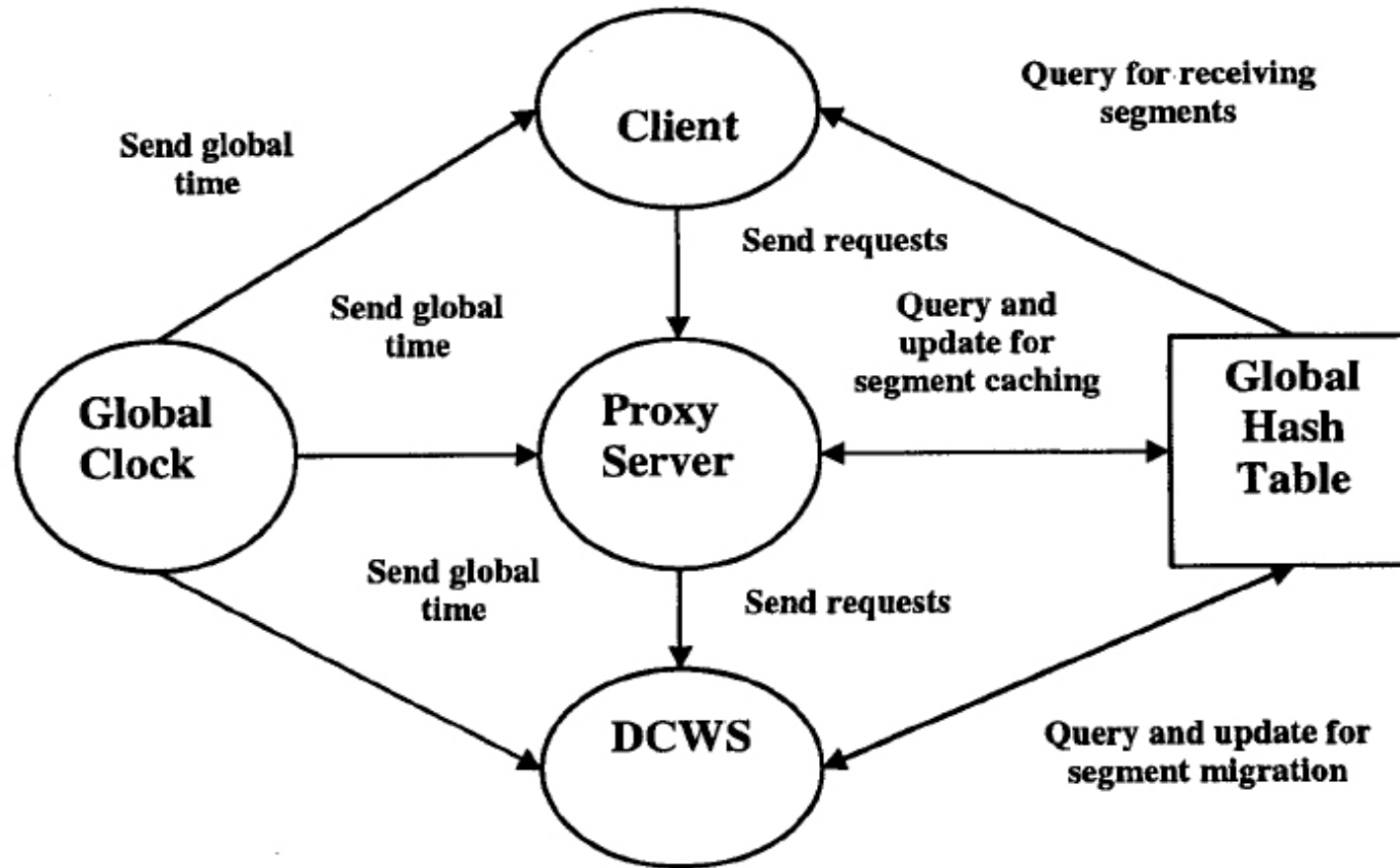
Output: This algorithm selects a document to be migrated to a cooperative server.

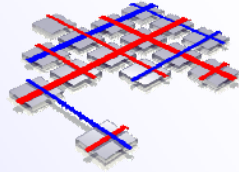
- (i) Let the candidate document set C be a set of all documents in the graph.
- (ii) Remove all the well-known entry points from C . If C is empty, return **nil**.
- (iii) Remove documents from C if their load is less than the threshold value T . If C is empty, reset it to the previous set and repeat this step with reduced value of T until C becomes nonempty.
- (iv) Select documents pointed to by the minimum number of link-from documents that do not reside on the home server.
- (v) If two or more documents are selected in step (iv), pick those that point to the minimum number of link-to documents.

End Algorithm



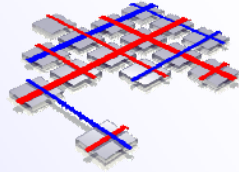
Simulation Setup



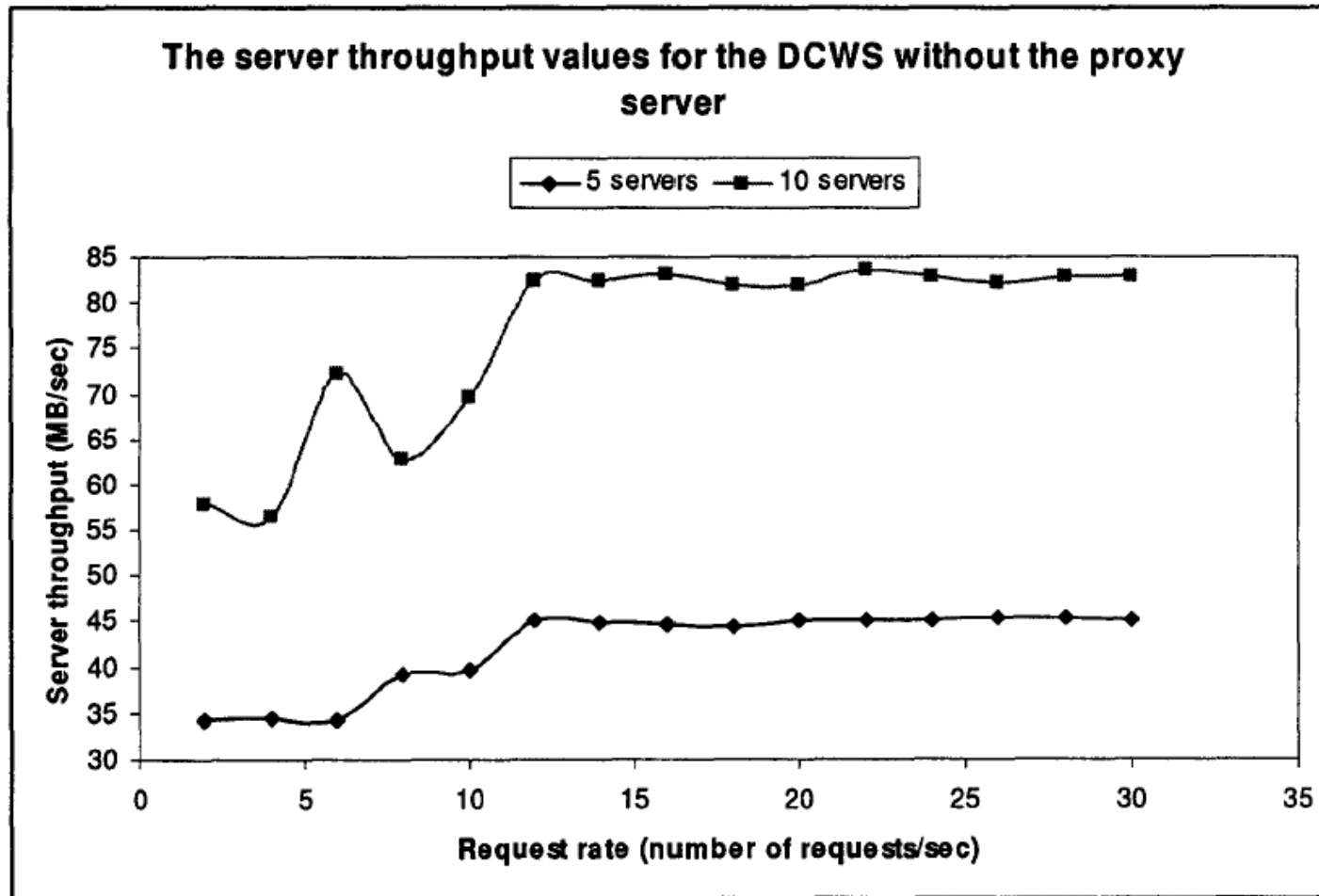


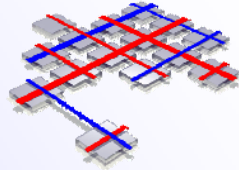
Segment Based Proxy Caching

Test Results

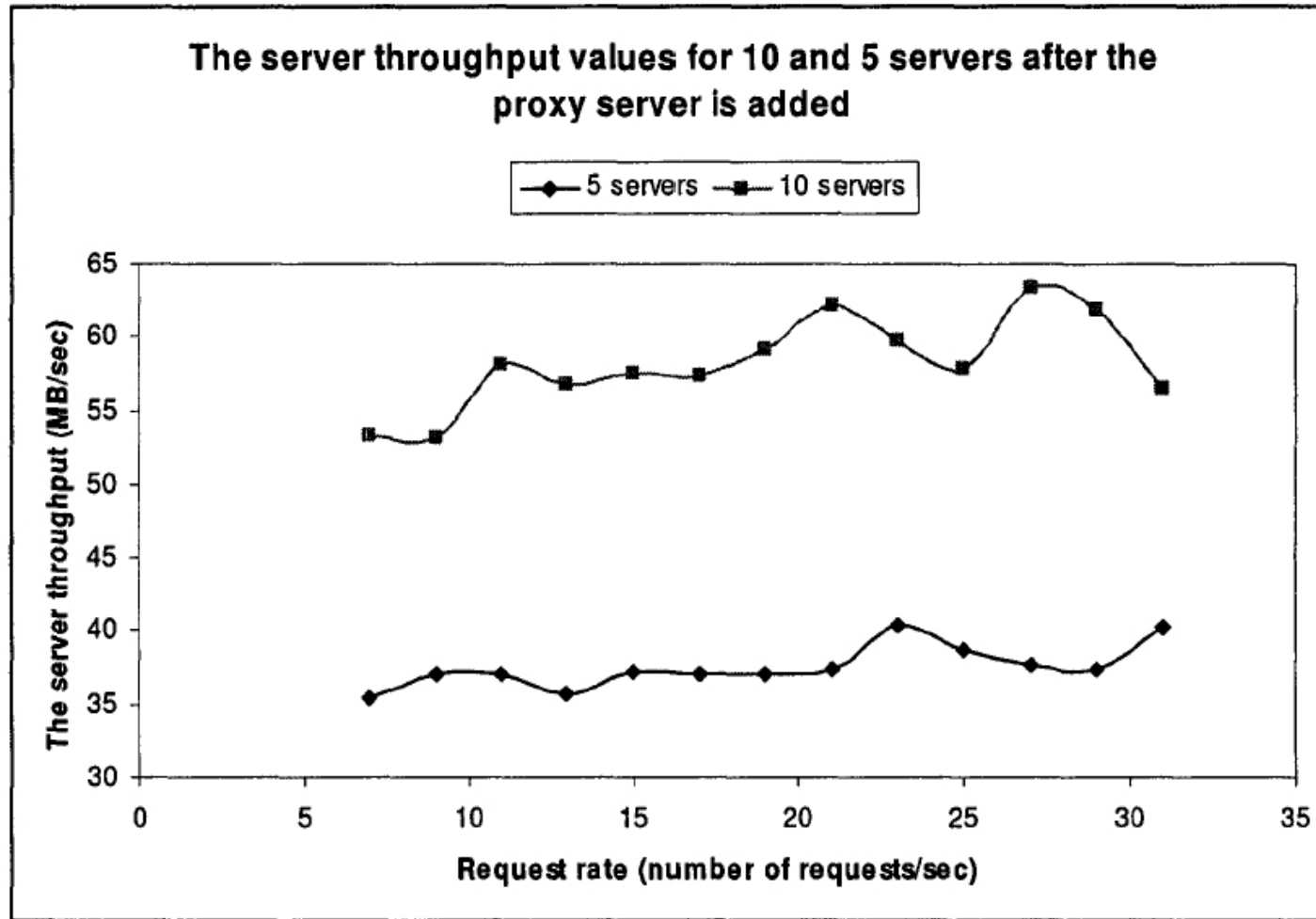


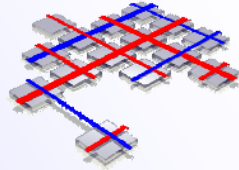
Test Results (1)



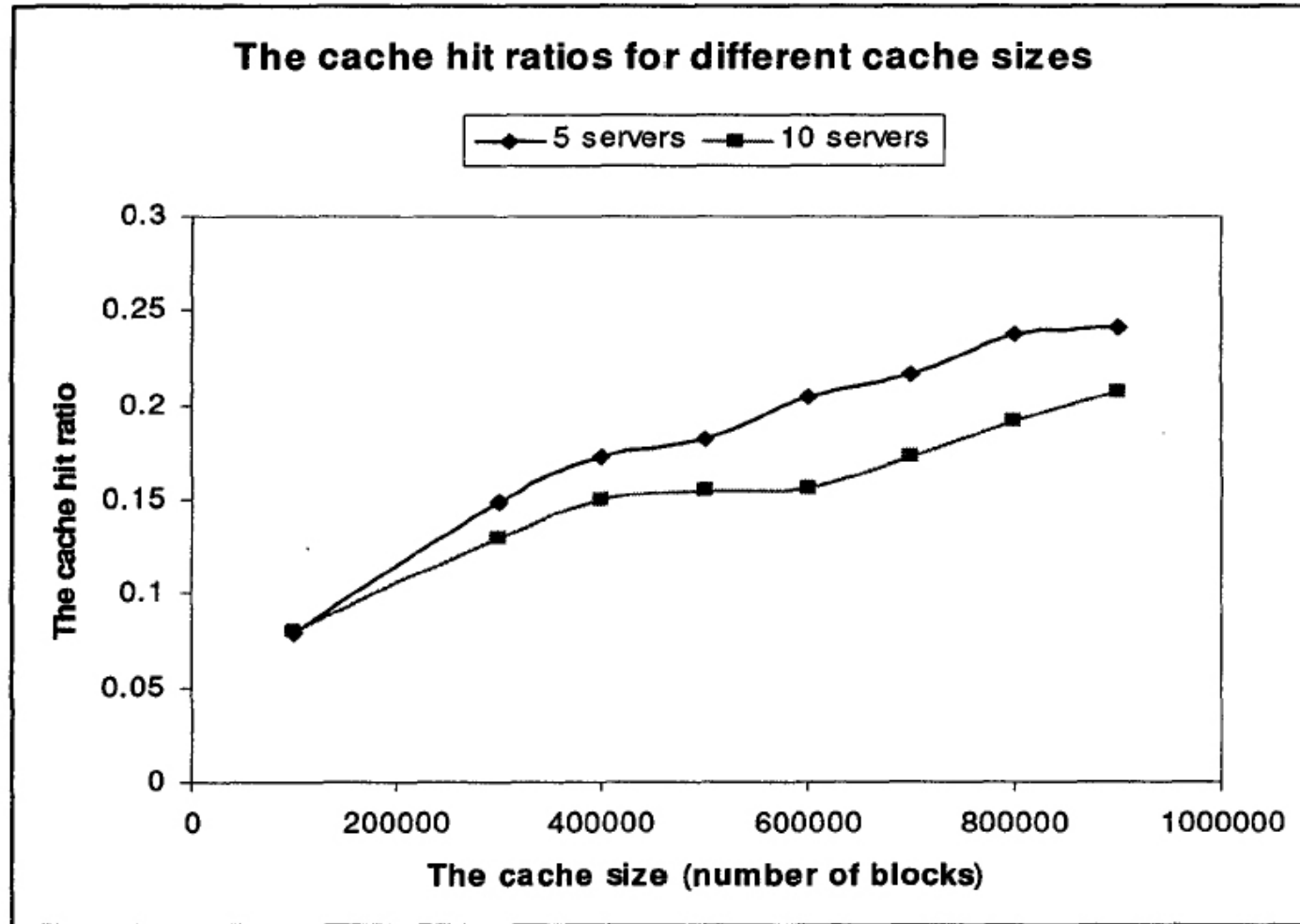


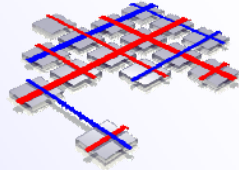
Test Results (2)



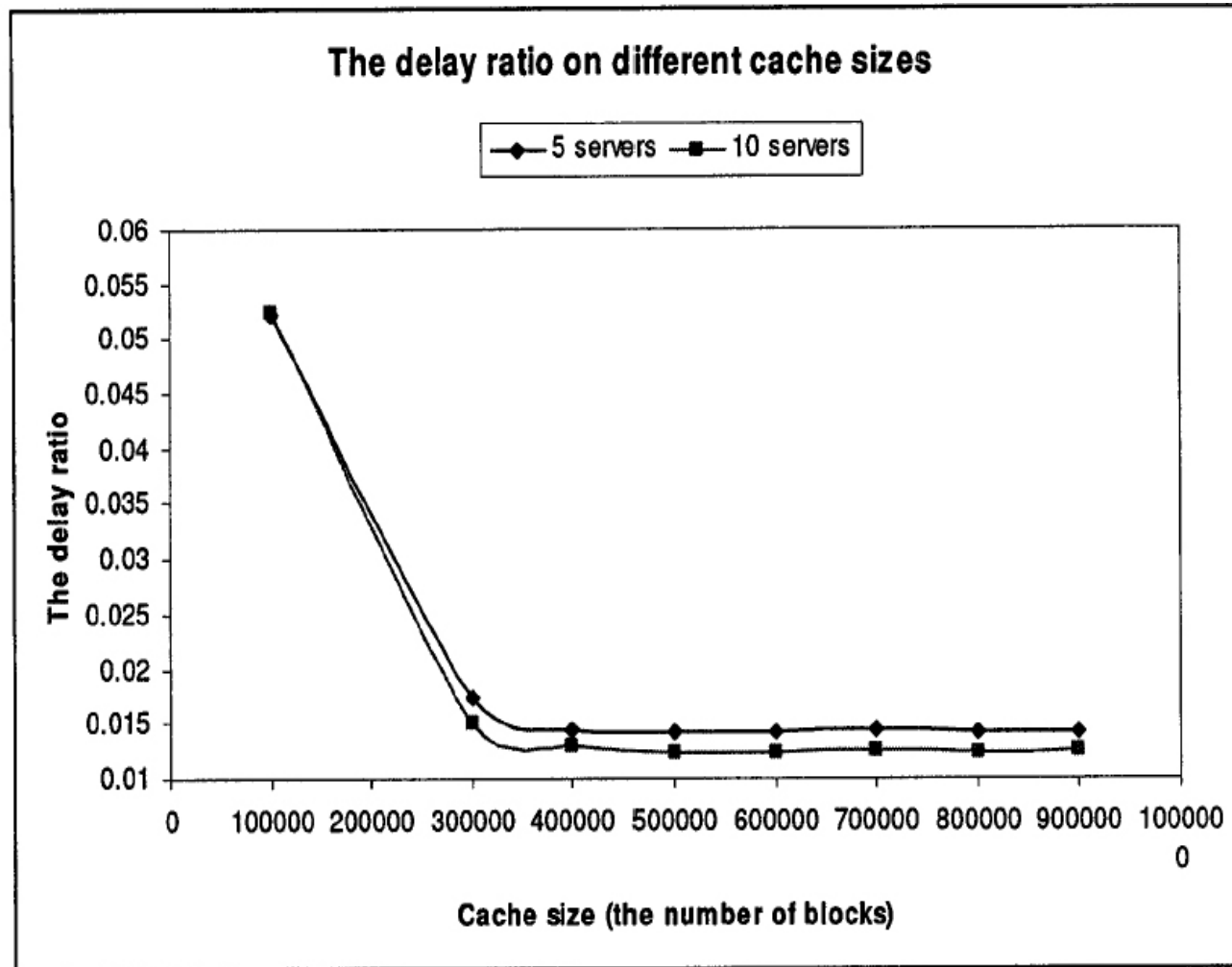


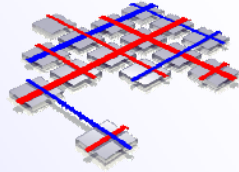
Test Results(3)





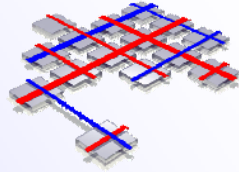
Test Results (4)





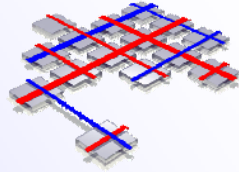
Conclusions

- Network performance improves
 - (at least in the simulation)
- Further research is needed to
 - Optimize values, like number of cooperative servers
 - Observe the behavior when the servers have more than one serving thread working concurrently in the system



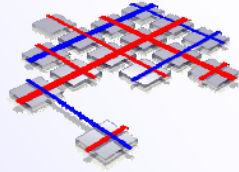
Second Part

Performance Issues in WWW Servers



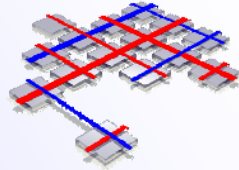
Introduction

- Different techniques for improving OS and network protocol software support for high performance WWW servers are evaluated:
 - New Socket functions
 - Per-byte optimizations
 - Per-connection optimizations



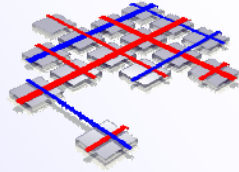
New Socket Functions

- Microsoft > Windows NT
 - Acceptex()
 - Transmitfile()
- HP > HP-UX
 - send_file()



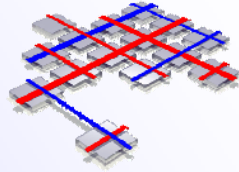
HTTP Transaction

Step #	Web Server Operation	User-level Optimization	Operating System Optimization
1.	<code>accept()</code> connection		<code>acceptex()</code>
2.	<code>getsockname()</code> for peer		<code>acceptex()</code>
3.	<code>read()</code> request		<code>acceptex()</code>
4.	<code>setsockopt()</code> Nagle off		inherit socket option
5.	<code>gettimeofday()</code>	fast clock reads	
6.	HTTP request parsing	URI cache	
7.	<code>stat()</code> for requested file	cache <code>stat()</code> info	
8.	<code>open()</code> requested file	cache <code>mmap()</code> 'ed files	file descriptor cache
9.	<code>read()</code> file into server	cache <code>mmap()</code> 'ed files	<code>send_file()</code>
10.	<code>write()</code> HTTP header	<code>writev()</code>	<code>send_file()</code> header option
11.	<code>write()</code> file data	<code>writev()</code>	<code>send_file()</code>
12.	<code>close()</code> file	cache <code>mmap()</code> 'ed files	file descriptor cache
13.	<code>close()</code> socket		<code>send_file()</code> close option
14.	<code>write()</code> log entry	buffered write	



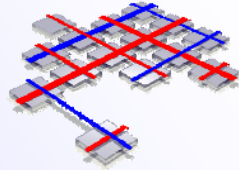
Per-byte Optimizations

- Data touching operations are expensive
- Unix filesystems force data to be copied when it is moved from one subsystem to another
- An approximate zero-copy integrated I/O architecture is tested to analyse the performance impact



Per Connection Optimizations

- TCP connection was not designed for client-server traffic
- Exchange of more packets than necessary
- How can connection overhead be reduced without violating the TCP protocol?

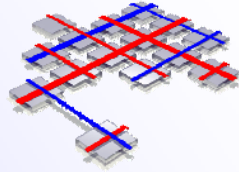


Per Connection Optimizations

1. Client: SYN 0:0(0)
2. Server: SYN 0:0(0) ACK 1
3. Client: ACK 1
4. Client: 1:61(60) ACK 1
5. Server: 1:1159(1158) ACK 61
6. Server: **FIN 1159:1159(0) ACK 61**
7. Client: ACK 1160
8. Client: FIN 61:61(0) ACK 1160
9. Server: ACK 62

Original TCP packet exchange.

- Sequence of exchanged TCP packets in a HTTP transaction
- 6th packet only carries a FIN bit

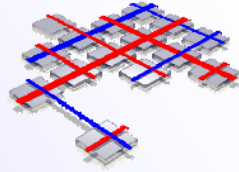


Per Connection Optimizations

- The FIN bit is carried by the 5th packet

1. Client: SYN 0:0(0)
2. Server: SYN 0:0(0) ACK 1
3. Client: ACK 1
4. Client: 1:61(60) ACK 1
5. **Server: FIN 1:1159(1158) ACK 61**
6. Client: ACK 1160
7. Client: FIN 61:61(0) ACK 1160
8. Server: ACK 62

Piggybacking the FIN.

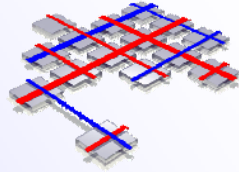


Per Connection Optimizations

1. Client: SYN 0:0(0)
2. Server: SYN 0:0(0) ACK 1
3. Client: ACK 1
4. Client: 1:61(60) ACK 1
5. Server: FIN 1:1159(1158) ACK 61
6. **Client: FIN 61:61(0) ACK 1160**
7. Server: ACK 62

Delaying ACK of FIN.

- Further reduction possible
- ACK is delayed

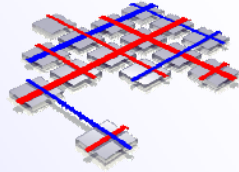


Per Connection Optimizations

1. Client: SYN 0:0(0)
2. Server: SYN 0:0(0) ACK 1
3. **Client: 1:61(60) ACK 1**
4. Server: FIN 1:1159(1158) ACK 61
5. Client: FIN 61:61(0) ACK 1160
6. Server: ACK 62

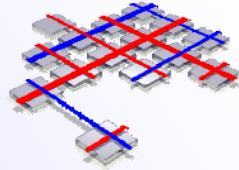
Delaying ACK of SYN-ACK.

- ACK of server's SYN-ACK was removed
- Another 5% performance increase



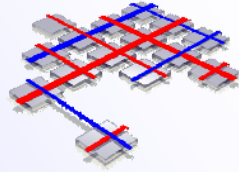
Experimental Setup and Testbed

- Experimental Setup and Testbed
 - 4 IBM 43P RS/6000 Workstations, with one machine as a server
 - Server got three 100 mb/s Ethernet interfaces
 - Each client is connected point full duplex with the server



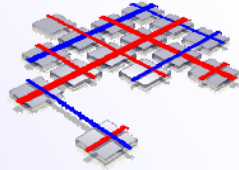
Client Workload Generator Software

- WebStone and SURGE were used as workload generators
- WebStone is used as a microbenchmark
 - Concurrent requests for the same file
- SURGE is used as a macrobenchmark
 - Concurrent request for a range of files



Performance Issues in WWW Servers

Test Results

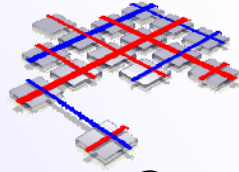


Evaluation of Socket Functions

TABLE III
THROUGHPUT IN HTTP OPERATIONS/S

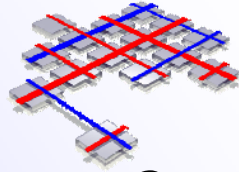
File Size (KB)	Flash Poll	Flash Poll AE	Diff (%)
1	1140.11	1151.89	1.03
2	1059.26	1072.58	1.26
4	904.15	913.93	1.08
8	722.22	727.99	0.80
16	501.92	504.59	0.53
64	187.72	188.49	0.41
256	54.36	54.23	-0.24
1024	13.55	13.56	0.07

The `acceptex()` system call makes only a small difference of 1% in small transfers



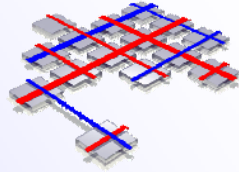
Evaluation of Per-Byte Optimizations

- A `send_file()` implementation combined with an integrated I/O system which does not copy data provides an increase of throughput by up to 85%
- Offloading the checksum to the network device can improve the performance by up to 9%



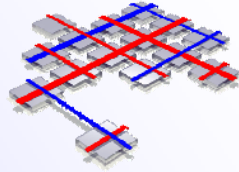
Evaluation of Per Connection Optimization

- `Send_file()` provides semantic support to enable piggybacking the FIN on the last data segment
- This eliminates one packet
- Throughput for small transfers increases by 7%
- Delaying acknowledgements for the FIN and SYN-ACK packets can eliminate two more packets, increasing performance by additional 11 %
- Total number of packets in small HTTP exchanges was reduced from nine to six, reducing network utilization by up to 20%



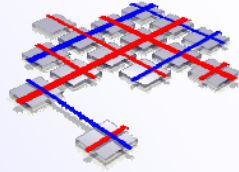
Conclusions

- Only little or no increase using `acceptex()`
- Reducing packet exchanges should help other TCP based applications
- `send_file()` combined with an integrated I/O system provides better performance
- `send_file()` is a general function that can also be used by other network servers like NFS, FTP or SMB

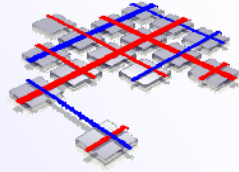


Future Work

- Evaluate mechanisms with HTTP 1.1 workloads
 - Per connection optimization will most likely be less significant
 - Per-byte optimization should be more effective than with HTTP 1.0

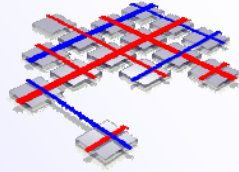


General Conclusion



Conclusion

By rational analysis and good ideas the performance of the World Wide Web can be improved which is quite necessary because more and more people are causing network congestion in the World Wide Web



Wake Up,
it's Weekend!