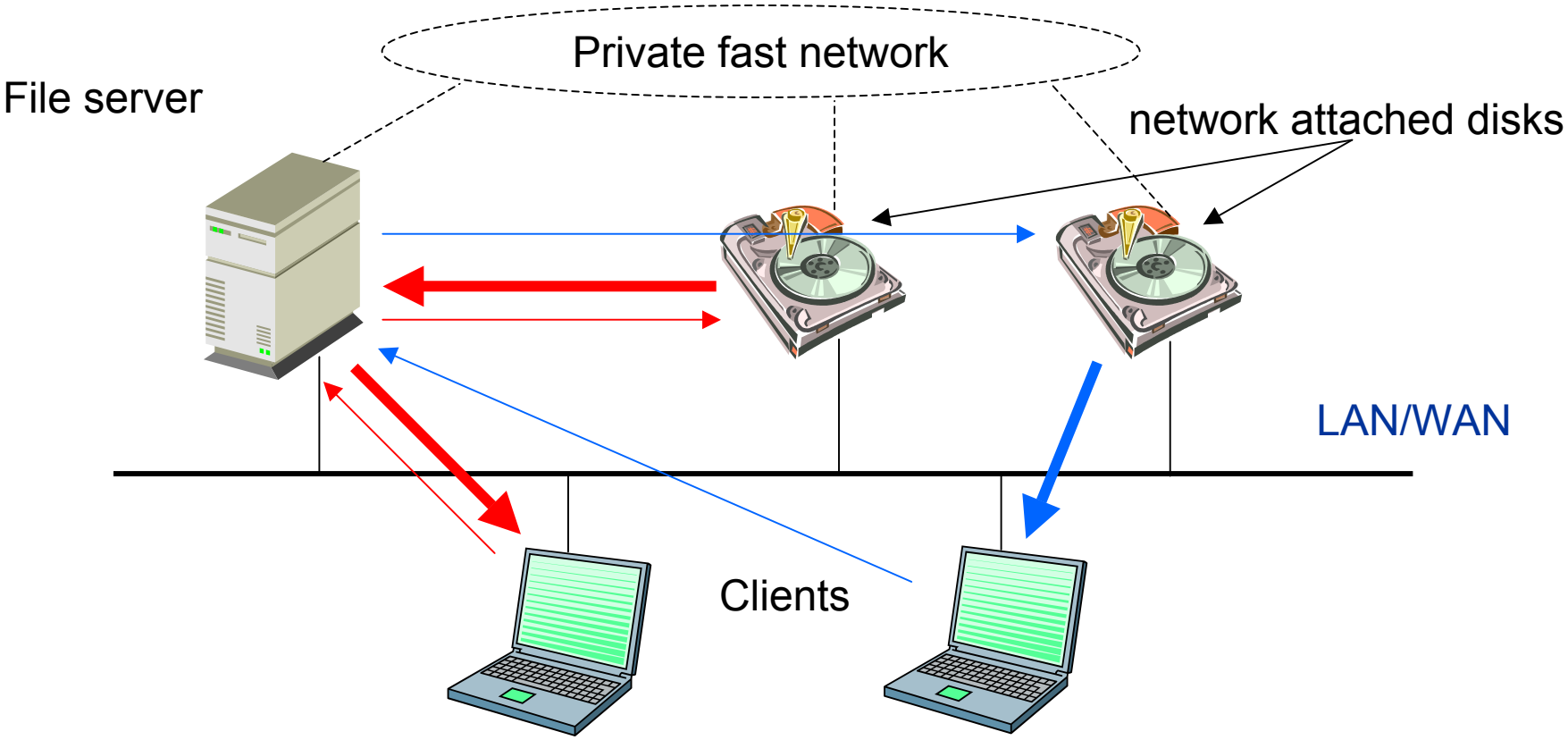
SPIRAL: A Client-Transparent Third-Party Transfer Scheme for Network Attached Disks

Xiaonan Ma
A. L. Narasimha Reddy

Motivation

- Problems with traditional networked storage systems
  - large volumes of data handled by server
    - multiple data copies
  - reduced buffer cache effectiveness
  - situation is even worse when disks are attached to the network

Data path with/without third-party transfer



Current approaches

- Exploit network connectivity of disks
  - Third-party transfer using Network-Attached Disk (NAD)
- Removing file manager bottleneck
  - Data operations (reads and writes) go straight to disk
  - Less common operations (namespace and access control) go to file manager
  - Clients given tokens from file manager dictating access rights

Problems

- Modifications to client OS/applications
  - OK for small closed environments
  - Resource consuming
- Porting file-system and application-level processing into NADs
  - Cost, complexity
  - Not compatible with block level protocols

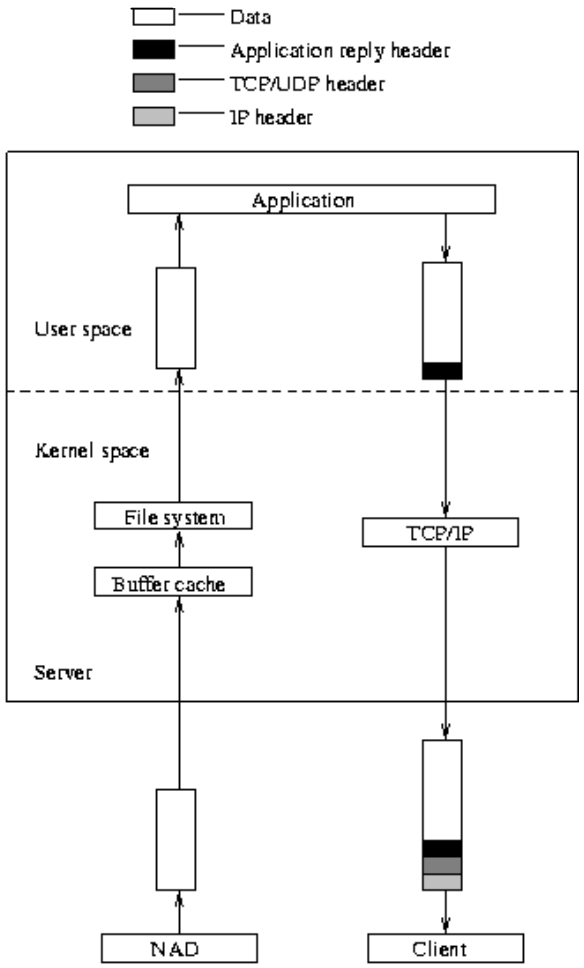
SPIRAL

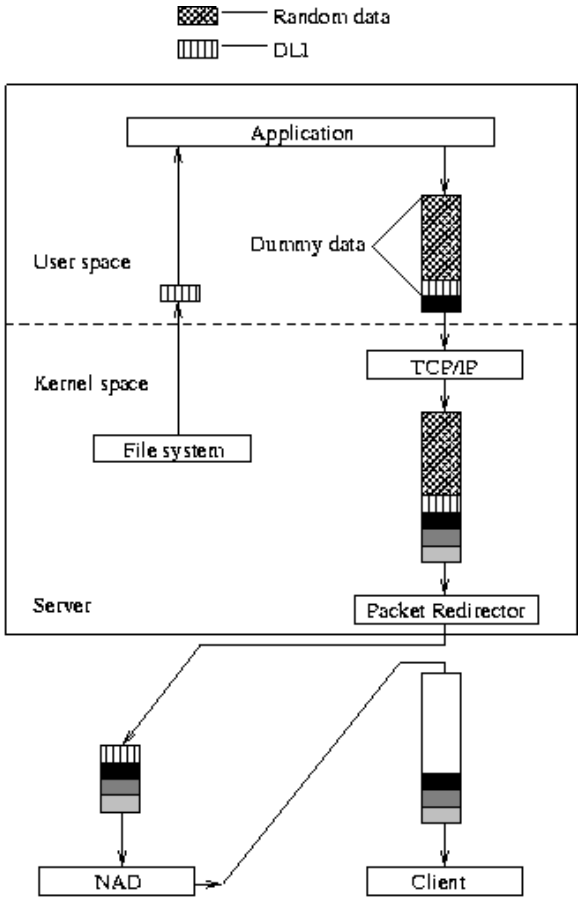
- Characteristics:
  - client-transparent
  - based on existing block interface
  - simple software layer on disk
  - few modifications on the server
  - supports both UDP and TCP

Key ideas

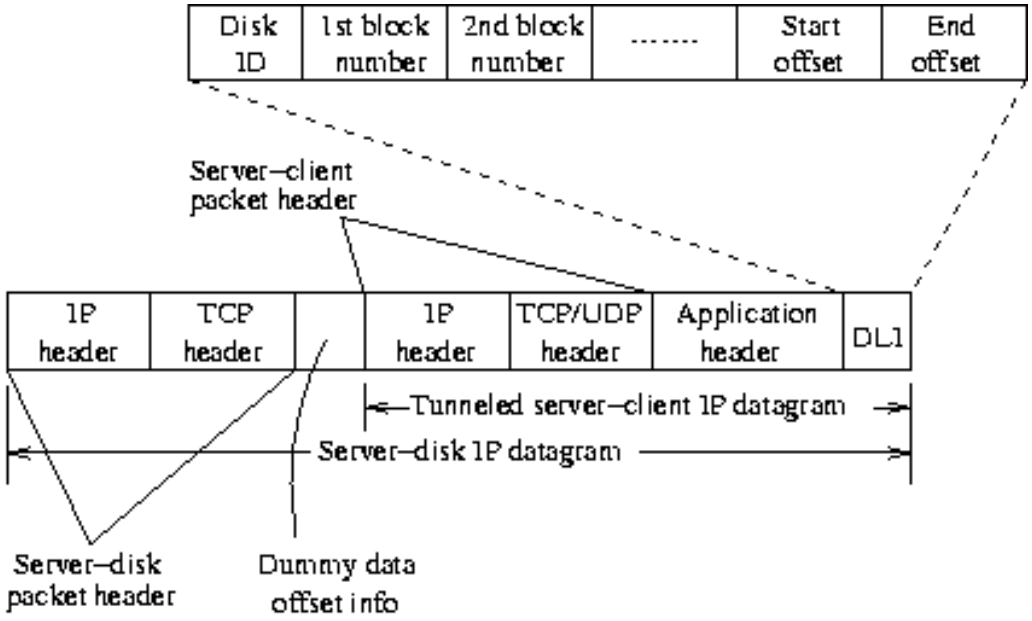
- new file system interface — tp_read
  - tp_read resolves DLI
  - application sends "dummy data" out
- datalink-layer packet redirector
  - intercepts packets containing dummy data
  - shrinks and redirects these packets

Data flow (traditional system vs. SPIRAL)





Typical structure of redirected packets



SPIRAL benefits

- reduced data traffic between server and disks
- reduced memory usage on the server
  - requests for metadata and small files served directly by the server
  - third-party transfer for large data transfers
- reduced CPU load on the server
  - network processing

Why datalink layer redirection?

- Advantages of datalink layer interception:
  - no transport-level state information on disks
  - disk only performs application-independent operations
  - TCP options
- TCP handoff may incur high overhead
  - P-HTTP, NFS over TCP
  - data striped over multiple disks

Challenges

- How to implement packet redirector with minimum performance impact?
- How to distinguish packets that need redirection?
- How to handle IP fragmentation and TCP retransmission?
- File system integrity and consistency?
- Multi-disk redirection?
- Security?

Packet filtering

- Two level scheme
  - filtering based on port number
    - IP fragments
  - filtering based on information provided by application
    - logical unit (e.g., an NFS reply)
      - UDP, TCP with framing support
    - byte range (e.g., 8000th ~ 12000th bytes)
      - HTTP

TCP issues

- Retransmission
  - retransmitted packets may not contain enough info
  - maintain history of previous redirections
- Inbound filtering
  - free saved redirection entries
  - detect connection termination/abortion

File system integrity and consistency

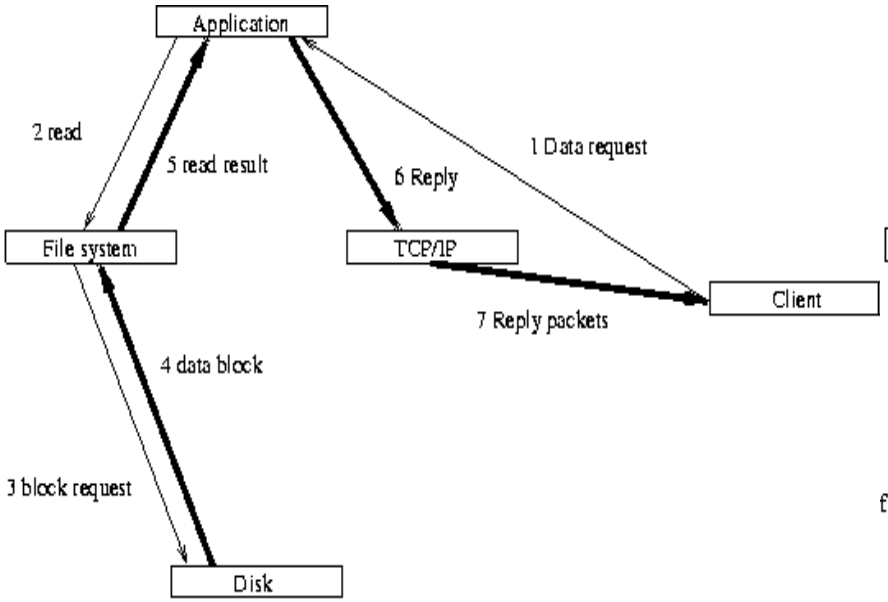
- server may have dirty buffers
- delay between DLI resolving and redirection
- solution — notification messages
  - disk marks data blocks as "pending"
  - sequence number (same block, different data)
    - writes happened in between
  - reference counter (same block, different DLIs)
    - pending block released when counter = 0

Multi-disk redirection

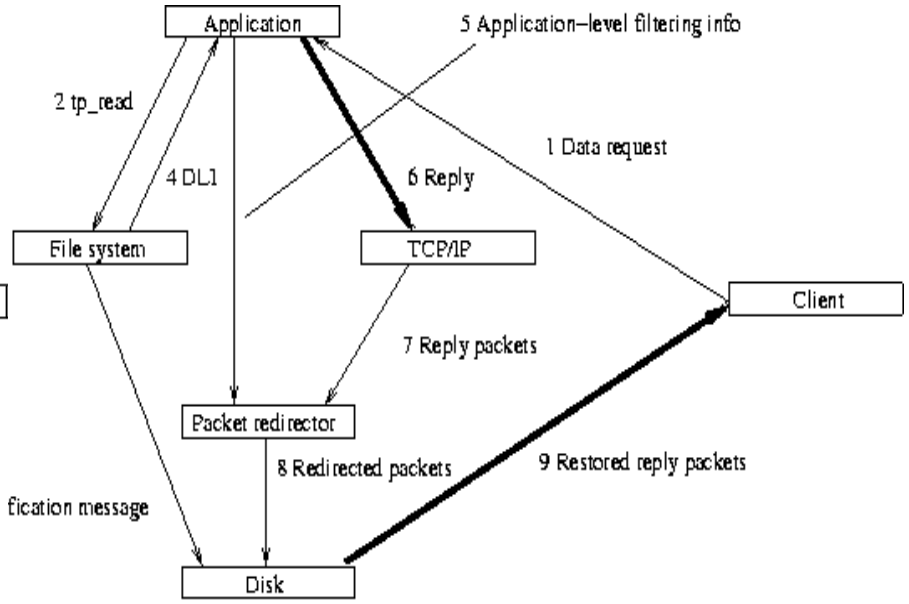
- a single packet may contain data from different disks
  - persistent HTTP connection
  - data striped over multiple disks
- solutions
  - circulate the redirected packets among disks
  - using IP fragments
- normally only a small percentage of all packets redirected

Data transfer procedures

Traditional system



SPIRAL



Limitation of SPIRAL

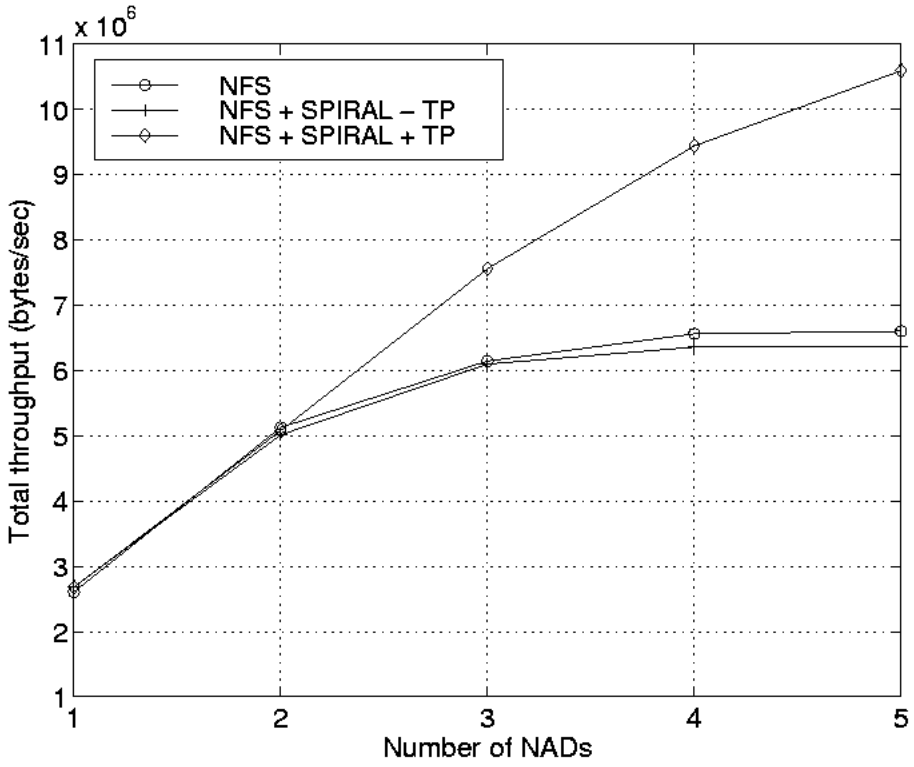
- only for outgoing data transfer
- requires applications to build reply without real data
  - dynamic web pages
- more work needs to be done to support encryption
  - SSL, IPsec

Prototype

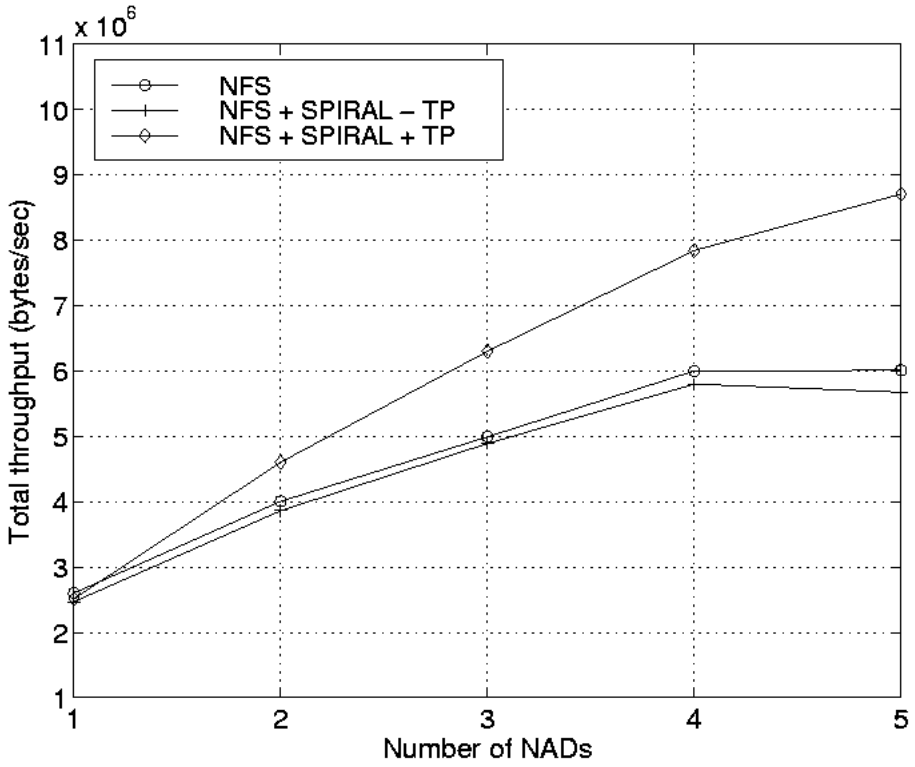
- Testbed
  - Server – 500MHz CPU, 128M memory
  - NADs – 166MHz CPU, 64M memory, Linux network block device (NBD)
  - Clients – 233MHz CPU, 64M memory
  - Dedicated 100Mbps Ethernet switch
- Linux loadable kernel modules for SPIRAL
- Minor modifications to kernel NFS server and Apache web server

NFS results

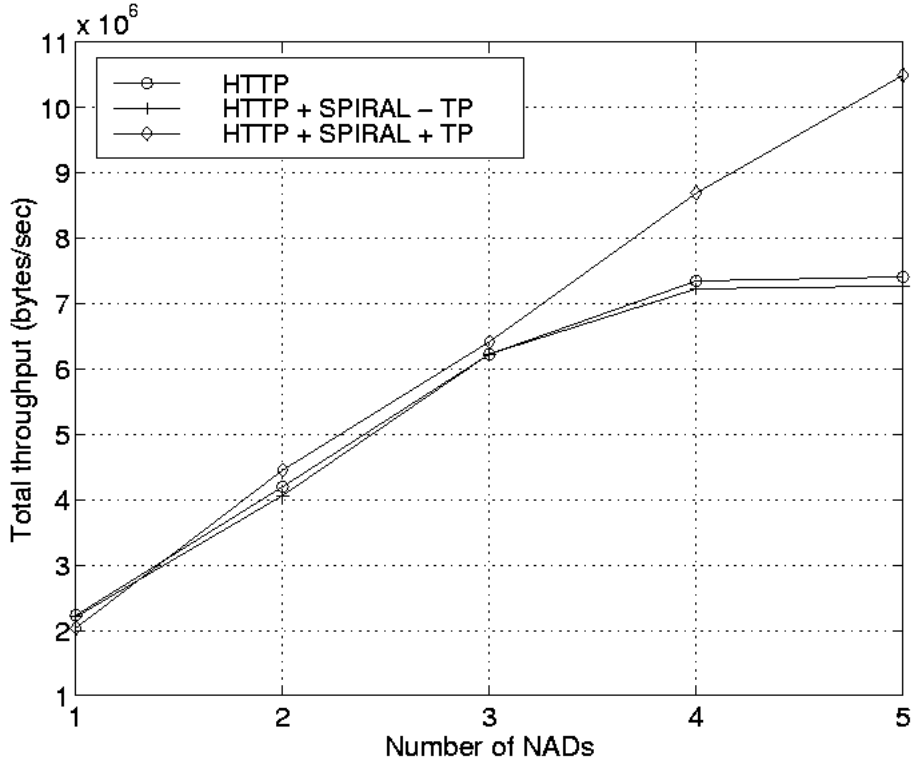
NFS UDP



NFS TCP



HTTP results



Remaining CPU power

- Dhrystone benchmark
- 628,930 dhrystones during idle time

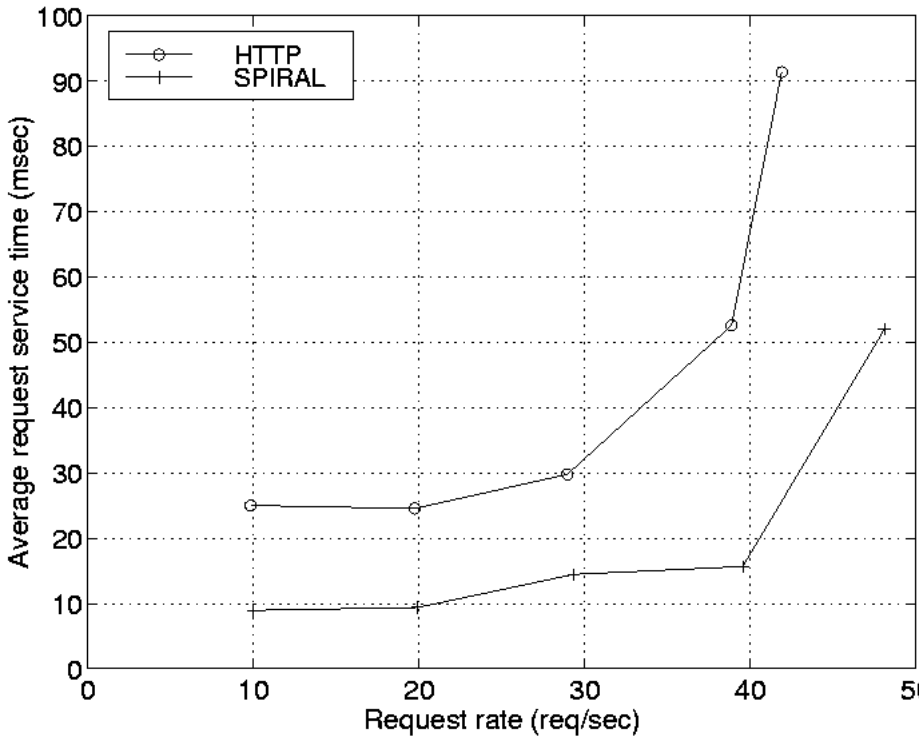
Type	Normal	SPIRAL without TP	SPIRAL
NFS UDP	70,109	59,784	375,657
NFS TCP	46,285	42,612	226,295
HTTP	62,985	50,959	263,504

Memory usage experiment

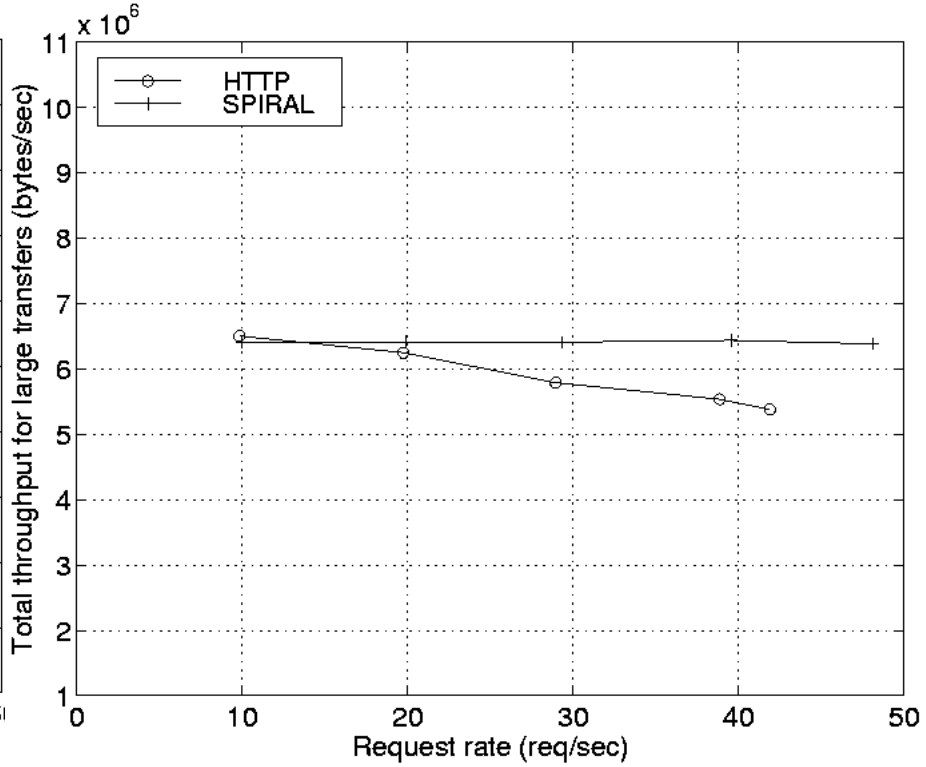
- lager data transfers no longer flush buffer cache
- one client repeatedly retrieve 250 64KB files
  - cache warmed up before experiment
  - average time for retrieving one cached file 8.8ms
- other clients retrieve large files simultaneously

Memory experiment results

Average service time for small HTTP requests

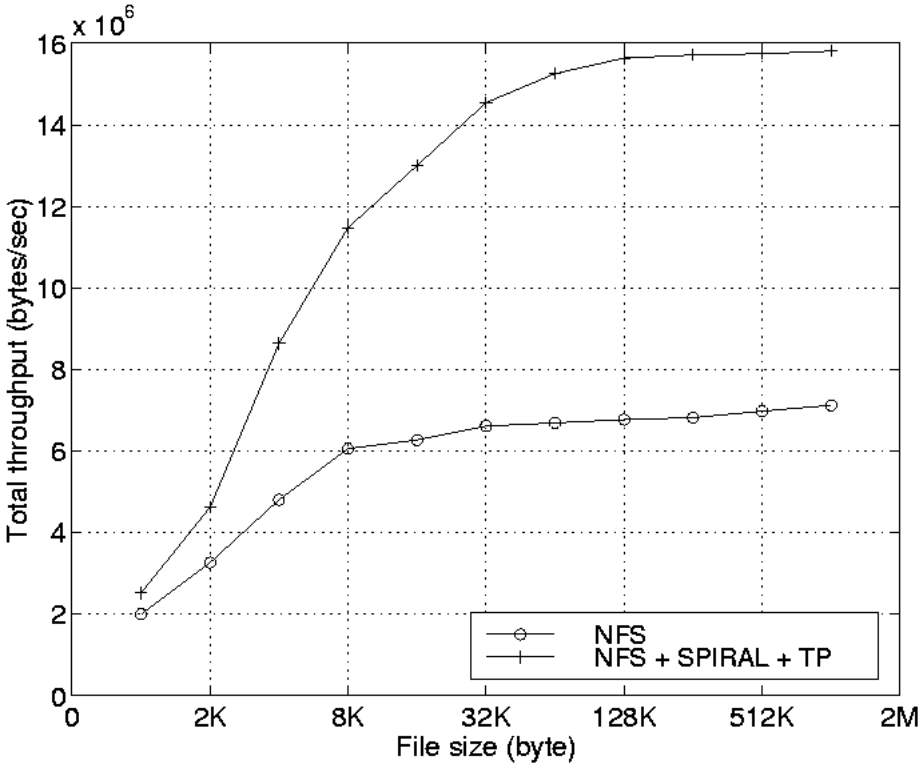


Total throughput for large HTTP transfers

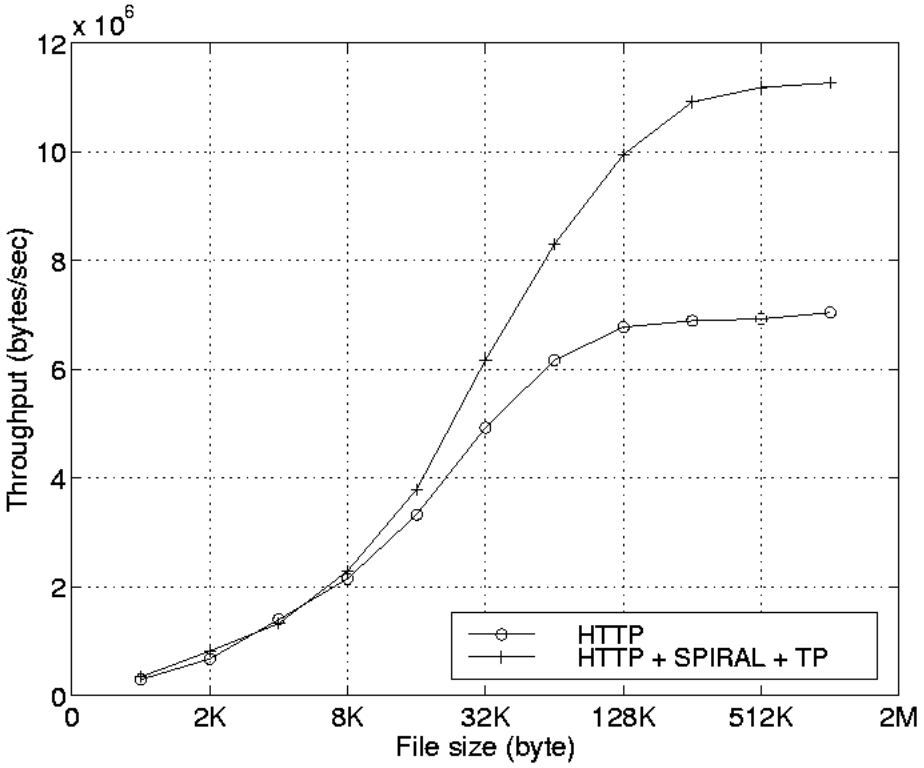


Different request sizes

NFS UDP



HTTP



WebBench

- Ziff Davis Media WebBench 4.1
  - 15 threads per client
- Standard workload tree
  - 6,160 files (HTML, GIF), around 61MB
- Third-party transfer only for files larger than 8KB

File size distribution among requests

File size (byte)	223	735	1522	2895	6040	11426	22132	41518	87360	541761	32M
Distribution w/o video (%)	20	8	12	20	14	16	7	0.8	0.1	0.1	
Distribution w/ video (%)	20	8	12	20	14	16	7	0.8	0.1	0.05	0.05

WebBench Results

	Server Memory (MB)	With HTTP 1.1	With video	Requests per sec	Throughput (MB/sec)	Data fetched from disk to server (MB)
Traditional	64	No	No	251	1.51	250
SPIRAL	64	No	No	252	1.51	17
Traditional	128	No	No	253	1.51	65
SPIRAL	128	No	No	255	1.55	17
Traditional	64	Yes	No	789	4.75	650
SPIRAL	64	Yes	No	844	5.09	17
Traditional	128	Yes	No	1020	6.01	65
SPIRAL	128	Yes	No	849	5.12	17
Traditional	64	Yes	Yes	248	5.66	1550
SPIRAL	64	Yes	Yes	396	6.93	22
Traditional	128	Yes	Yes	348	6.62	1100
SPIRAL	128	Yes	Yes	399	7.76	18

Generalization of SPIRAL

- SPIRAL can be generalized for use in clusters
  - server holding the connection may not be the best candidate to serve the request

Related work

- Network Attached Secure Disk (NASD)
- Derived Virtual Disk (DVD)
- Locality-aware Request Distribution (LARD)
- Imposed request routing (Slice)

Conclusions

- Client-transparent third-party transfer can be supported with "dumb" block-based NADs
- TCP can be supported efficiently with datalink-level packet interception
- Best suited for workloads where a large part of the server's load is due to data transfer and where read requests dominate
  - Online video servers or digital libraries