SeMiNAS: A Secure Middleware for Wide-Area Network-Attached Storage

Ming Chen
Erez Zadok
{mchen, ezk}@cs.stonybrook.edu
Stony Brook University

Arun O. Vasudevan
aov@nutanix.com
NUTANIX

Kelong Wang
kelong@dssd.com
EMC² DSSD

Outline

- **Background & Motivation**
- Design
- Implementation
- Evaluation
- Conclusions

Cloud Computing

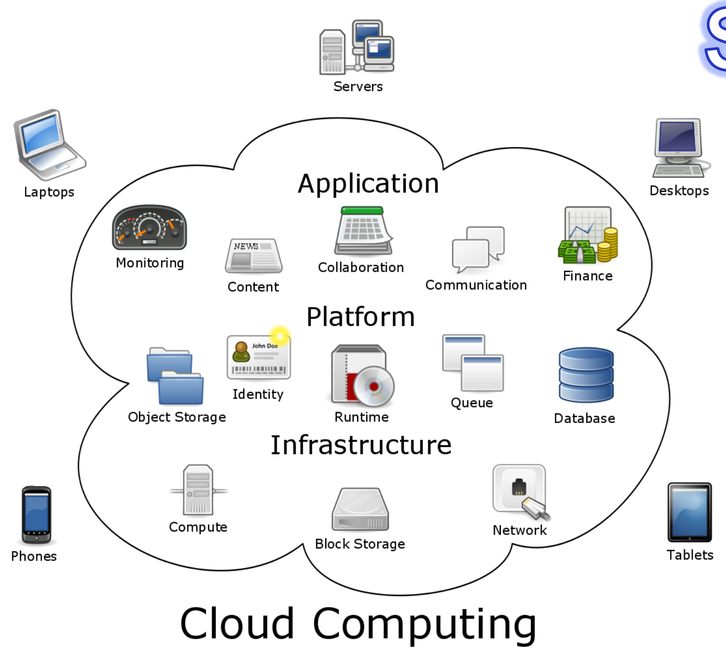
Accessibility

Scalability

Agility

Productivity

Availability



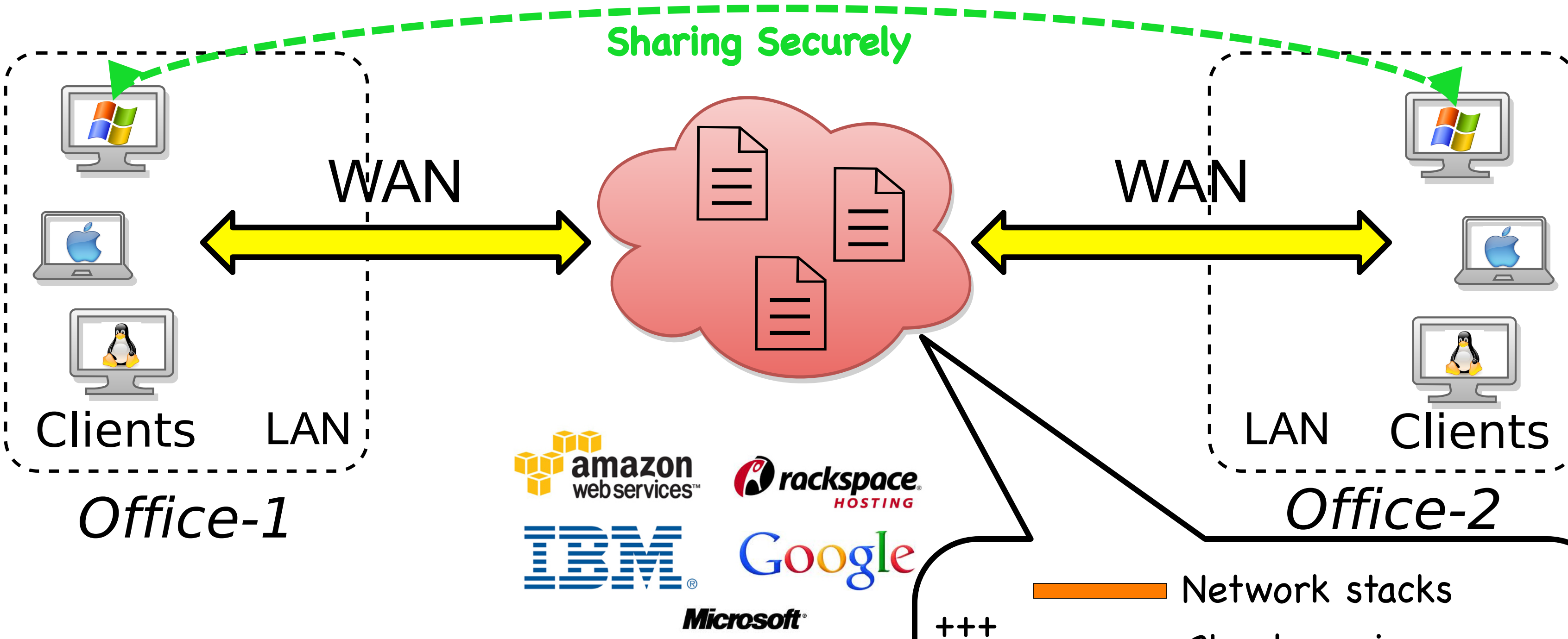
Economy

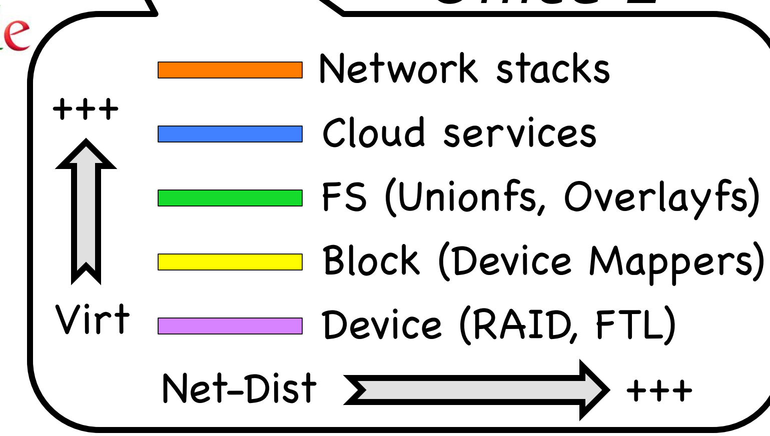
Flexibility

Security Concerns of Cloud

- Raised by cloud nature
  - Opaque & intangible
  - Multi-tenant
  - Large exploit surface
  - Complexity (buggy)
- Intensified by high-profile incidents
  - Silent data corruption
  - Leak of intimate photos of celebrities
  - Leak of user accounts and credentials

Securing Cloud Storage





New challenges:

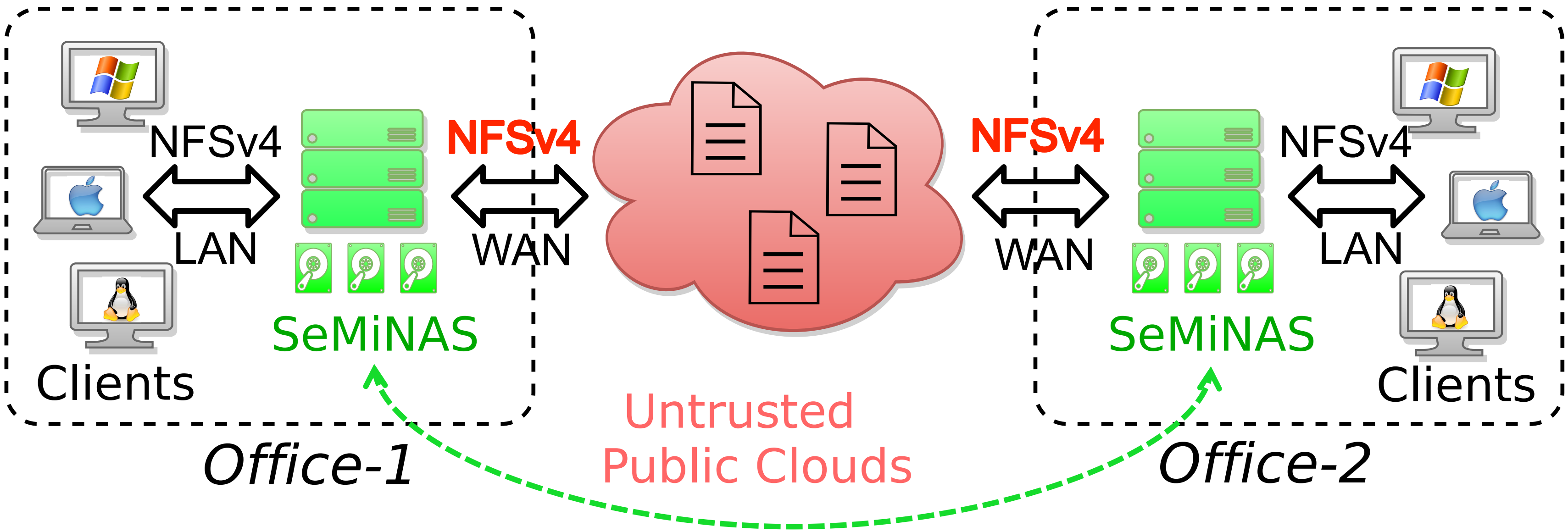
1. Cost-efficiency despite high latency
2. Heterogeneous clients & clouds
3. Complex storage stack

Outline

- ☑ Background & Motivation
- ➢ **Design**
- ☐ Implementation
- ☐ Evaluation
- ☐ Conclusions

Stony Brook University

SeMiNAS Architecture



Benefits of a middleware:

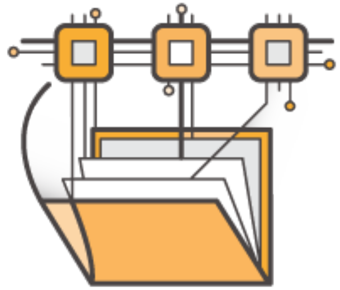
1. Easy management (a few proxies vs. many clients)
2. Simple key distribution without trusted third parties
3. Fit well with WAN caching and firewalls

Stony Brook University

Why Use NFSv4?

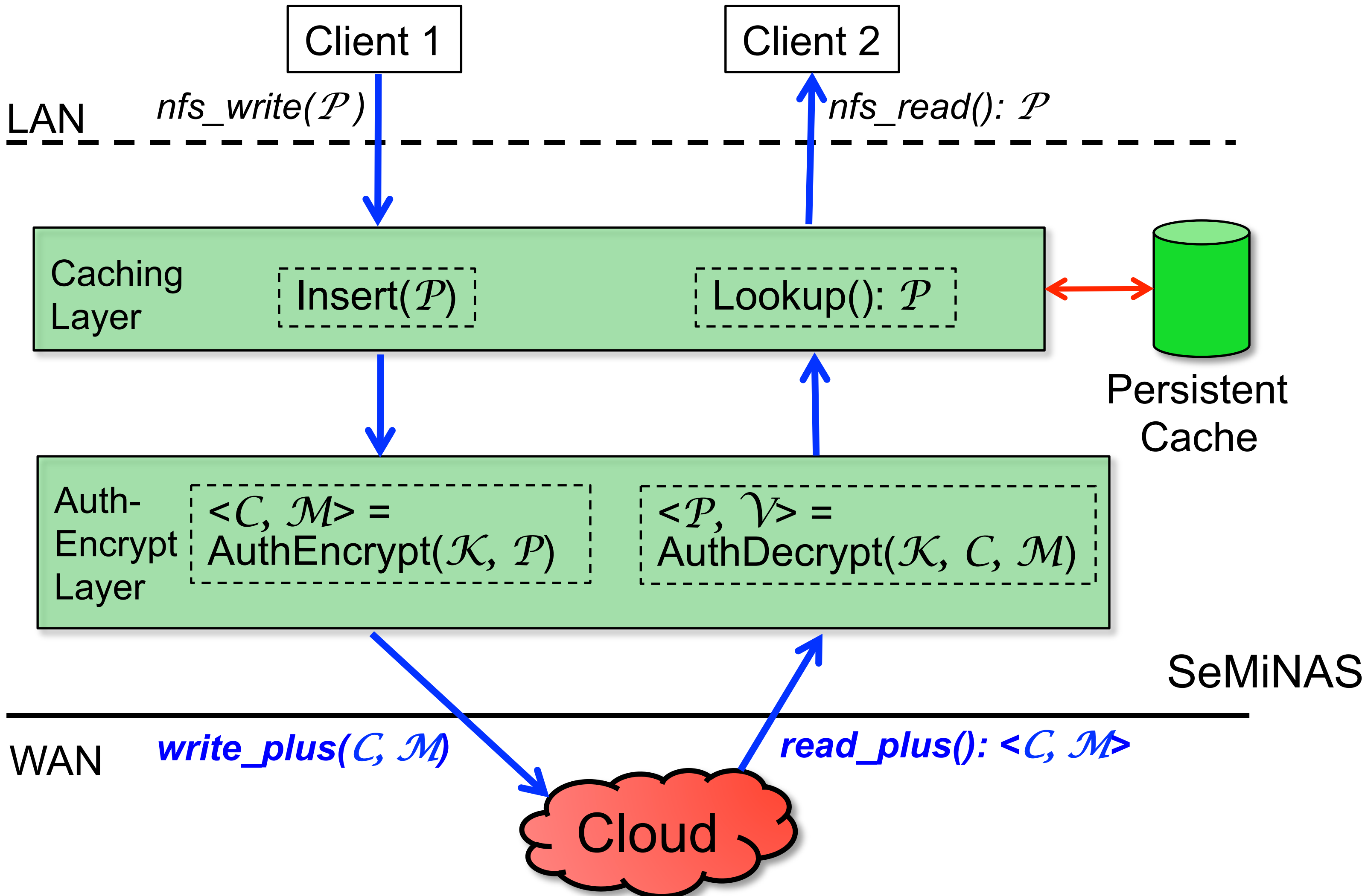
- Advantages over vendor-specific key-value stores
  - Open, pervasive, and standard
    - POSIX-compliant and cross-platform interoperability
    - Suffering less from data or vendor lock-in
  - Optimized for WAN
    - Compound procedures
    - Delegations
  - Richer semantics
    - Simplify application development
    - More optimizations: server-side copying, ADB
- Advantages over older versions
  - Easier administration with a single port
  - More scalable with pNFS
  - More secure with RPCSEC_GSS, ACL, and Labeled NFS

Amazon EFS



Stony Brook University

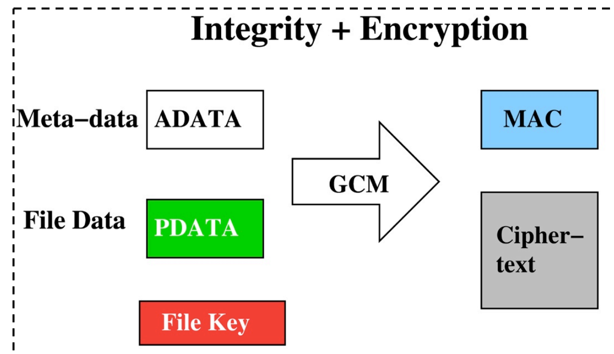
SeMiNAS Data Path



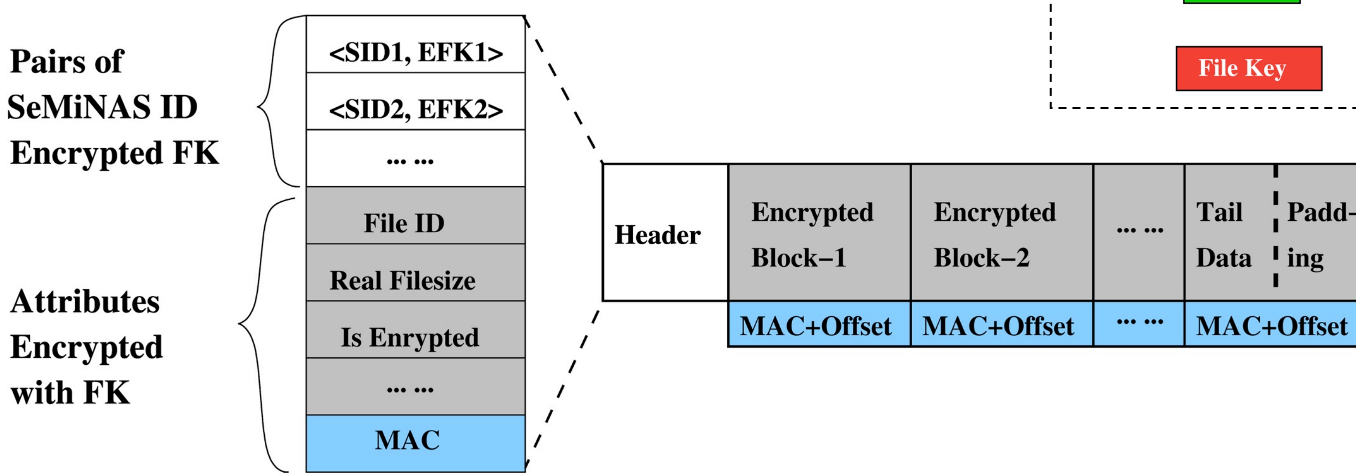
Stony Brook University

Meta-Data Management

- Each SeMiNAS proxy has `<SID, PubKey, PriKey>`
  - Each proxy knows public keys of all proxies
  - Distributed via a secret channel or manually
- Each file has a unique symmetric file key
  - Encrypted by master key pairs
  - Encrypt each block with GCM:

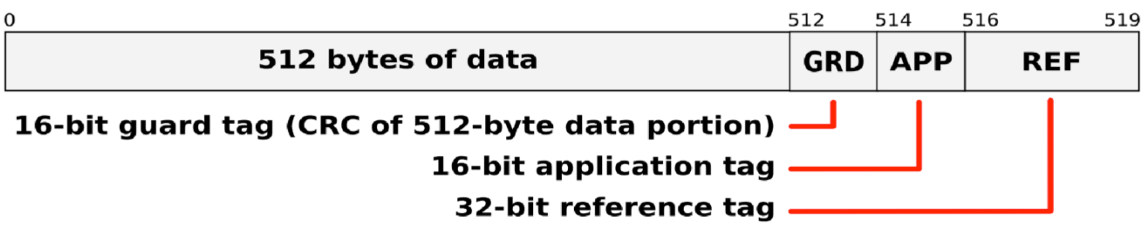


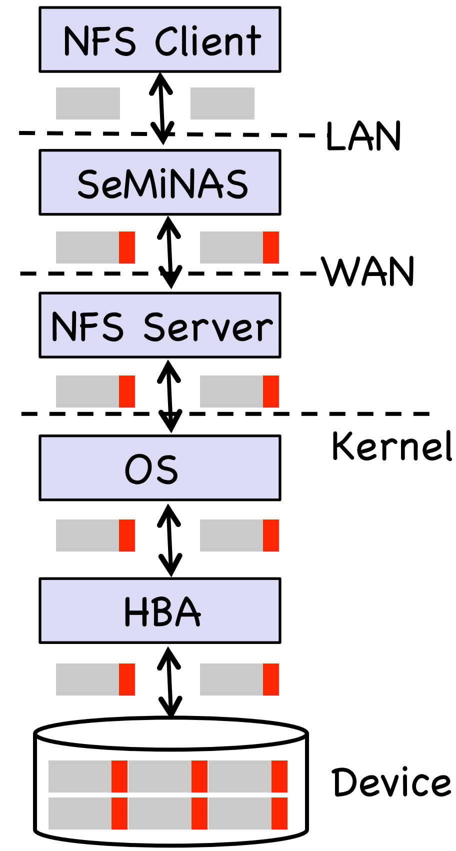
- File layout:



NFSv4-Based Optimizations (1)

- NFS Data-Integrity eXtensions

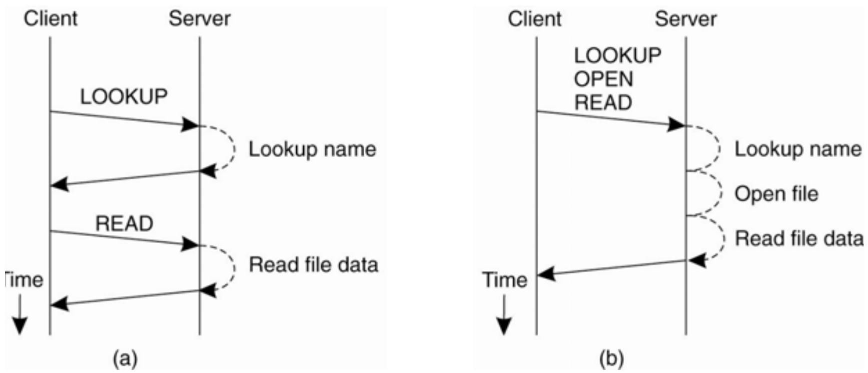




Alternatives	Drawbacks
Concatenate a block and its MAC as a separate file.	Break close-to-open consistency
Uses a separate file for all MACs of a file.	Add extra I/O and disk seeks
Map a block to a larger block in cloud (16➔20KB).	Waste space for small block sizes

NFSv4-Based Optimizations (2)

- Compound Procedures



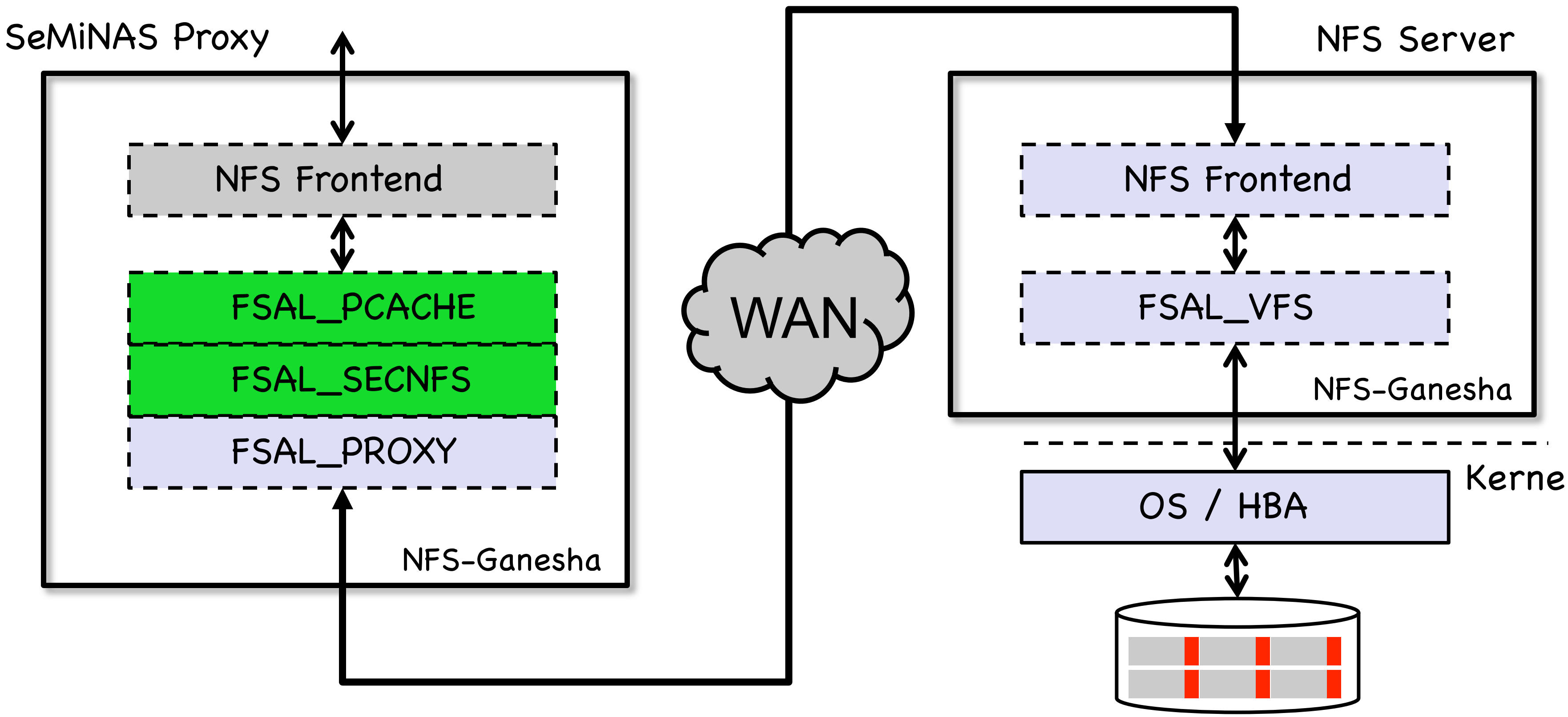
- SeMiNAS Compounds
    1. Write header after creating a file
    2. Read header after opening a file
    3. Update header before closing a dirty file
    4. Read header when getting attributes
    5. Get attributes after writing to a file

Outline

- ☑ Background & Motivation
- ☑ Design
- ➢ **Implementation**
- ☐ Evaluation
- ☐ Conclusions

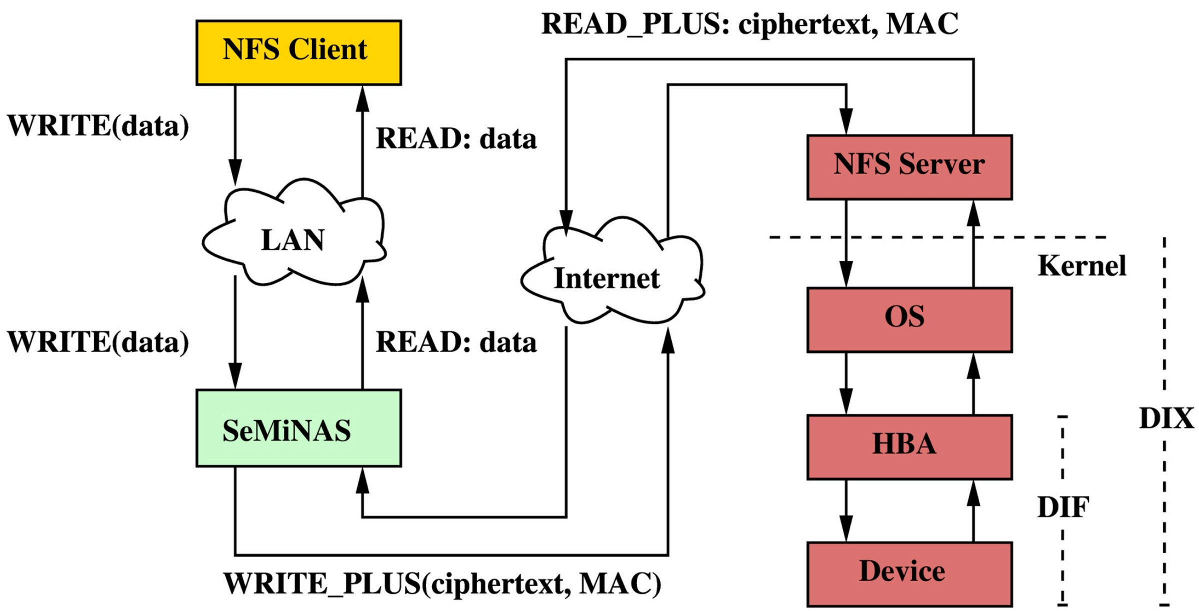
SeMiNAS Implementation

- NFS-Ganesha: a user-land NFS server
  - File System Abstraction Layer (FSAL) back-ends
  - FSAL_VFS, FSAL_PROXY, and stackable FSALs



Extending DIX to NFS

- Data Integrity eXtensions (DIX) in NFS
  - READ_PLUS
  - WRITE_PLUS



Implementation Details

- Details
  - Added caching and security layers in NFS-Ganesha
  - Added support of multiple stackable layers
  - Extended DIX further to NFS
  - Cryptographic C++ library: `cryptopp`
  - Pass all applicable `xfstests` cases
- Development efforts
  - 25 man-months of 3 graduate students over 3 years
  - Added 13,000 lines of C/C++ code to NFS-Ganesha
  - Fixed 11 NFS-Ganesha and 4 kernel bugs

Outline

- ☑ Background & Motivation
- ☑ Design
- ☑ Implementation
- ➢ **Evaluation**
- ☐ Conclusions

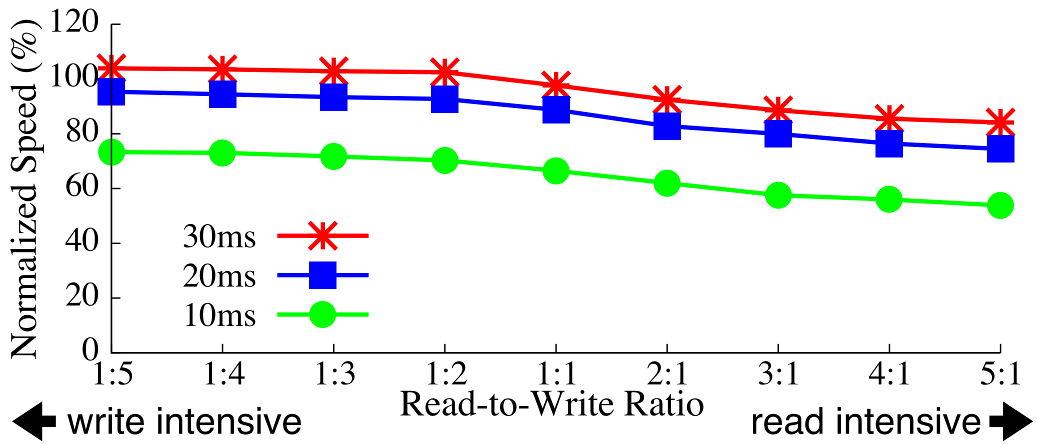
Setup & Workloads

- Experimental setup
  - Five NFS clients: 1G RAM; 6-core CPU; 10GbE NIC
  - SeMiNAS proxy: 64G RAM; 6-core CPU;10GbE NIC for LAN; 1GbE NIC for WAN; 200GB SSD for cache
  - Server: 64G RAM; 6-core CPU; 1GbE NIC; 20GB virtual SCSI DIX disk backed by RAM
- Workloads

Micro-Workloads	Filebench Workloads
Random file read/write	NFS Server
File creation	Web Proxy
File deletion	Mail Server

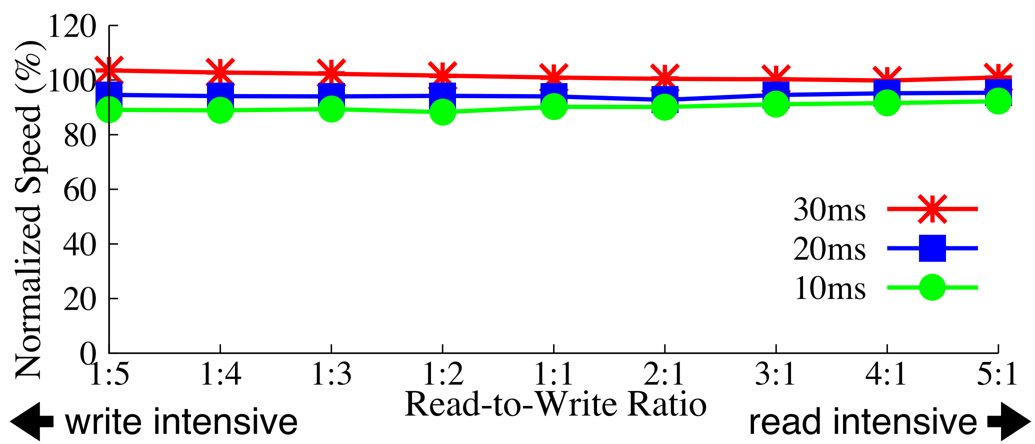
Different R/W Ratios

-46%➔+4%



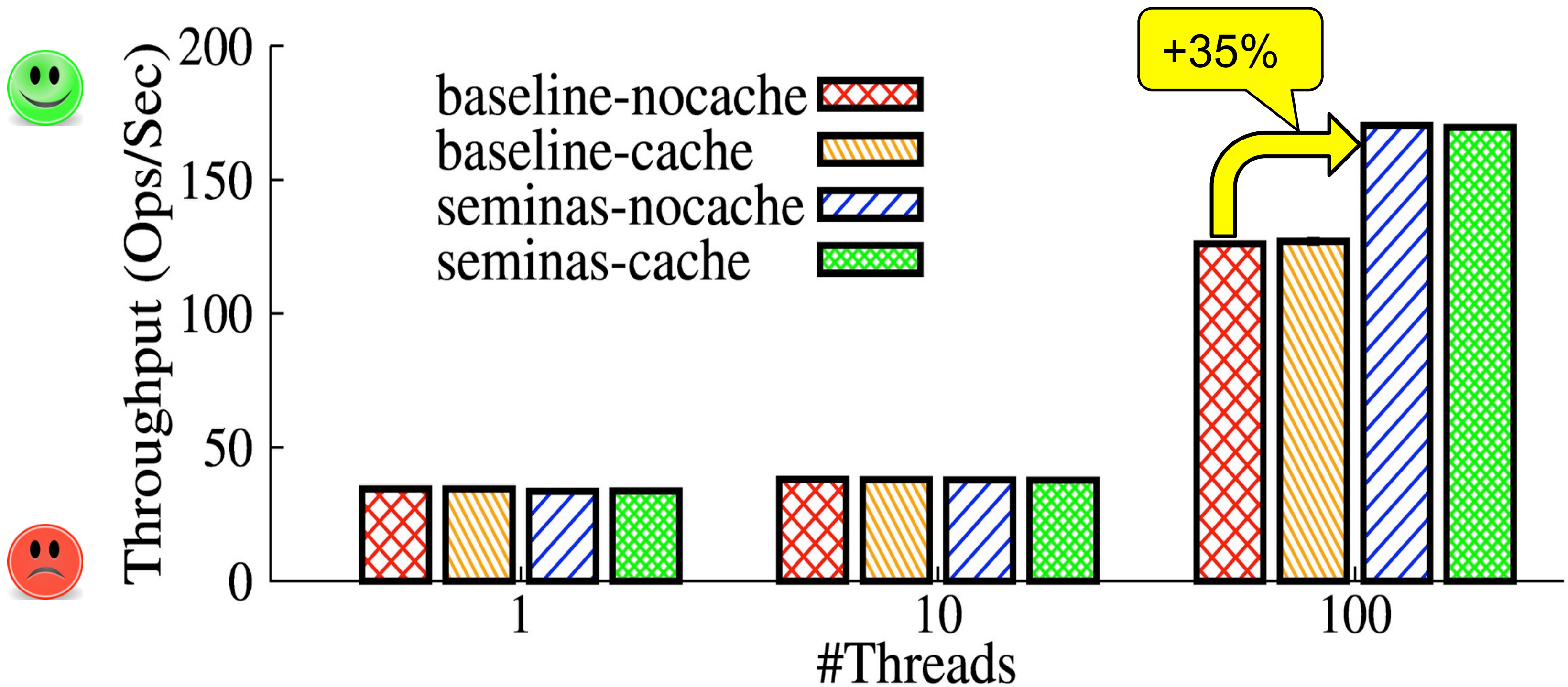
(a) Persistent Cache (FSAL_PCACHE) Off

-8%➔+4%



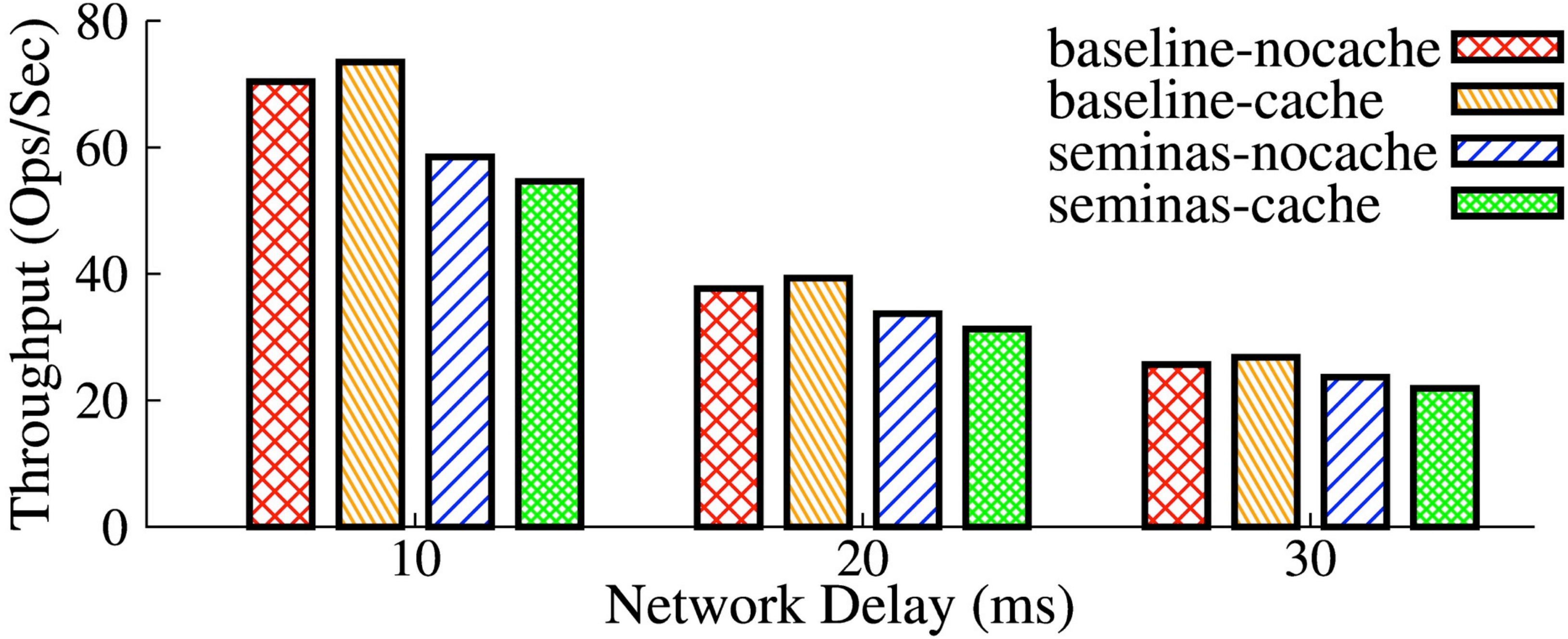
(b) Persistent Cache (FSAL_PCACHE) On

File-Creation Workload



- SeMiNAS makes file creation faster
  - TCP Nagle Algorithm
  - Multiple threads sharing one TCP connection
  - SeMiNAS write extra file headers

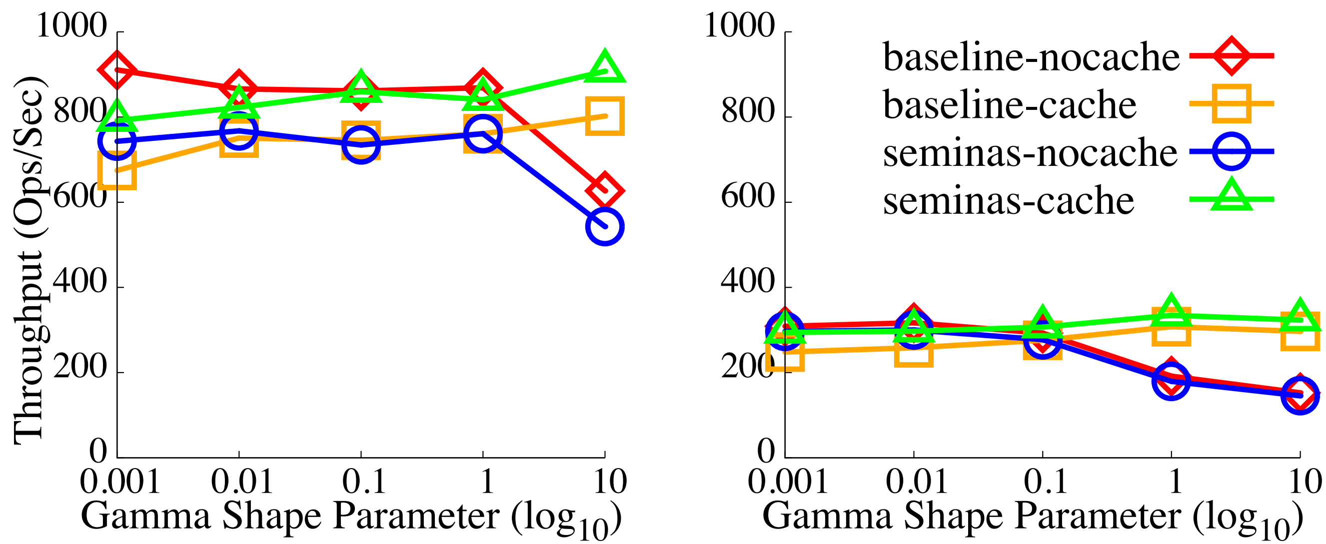
Filebench NFS-Server Workload



- SeMiNAS performance penalty
  - 8−17% without cache
  - 18−26% with cache
  - Decreases as network delay increases

Stony Brook University

Filebench Web-Proxy Workload



(a) 10ms Network Delay (b) 30ms Network Delay

- SeMiNAS makes web-proxy
  - 4−6% **slower** without cache
  - 9−19% **faster** with cache (because of TCP Nagle)

Outline

- ☑ Background & Motivation
- ☑ Design
- ☑ Implementation
- ☑ Evaluation
- ➢ **Conclusions**

Conclusions

- We proposed SeMiNAS to secure cloud storage
- We designed SeMiNAS to
  - Be a middleware
  - Take advantages of NFSv4 compounds, and
  - Data Integrity eXtensions
- We implemented SeMiNAS based on
  - Add security stackable file-systems layers
  - Extend DIX to NFS
- We evaluated SeMiNAS:
  - small performance penalty less than 26%
  - performance boost by up to 19%

Limitations & Future Work

- Limitations
  - Not safe against replay attacks
  - Does not handle side-channel attacks
- Future work
  - Efficiently detect replay attacks
    - Avoid using expensive Merkle trees
    - Synchronize file versions among proxies
  - File- and directory-name encryption
  - Transactional Compounds

https://github.com/sbu-fsl/txn-compound

SeMiNAS: A Secure Middleware for Wide-Area Network-Attached Storage

Q&A

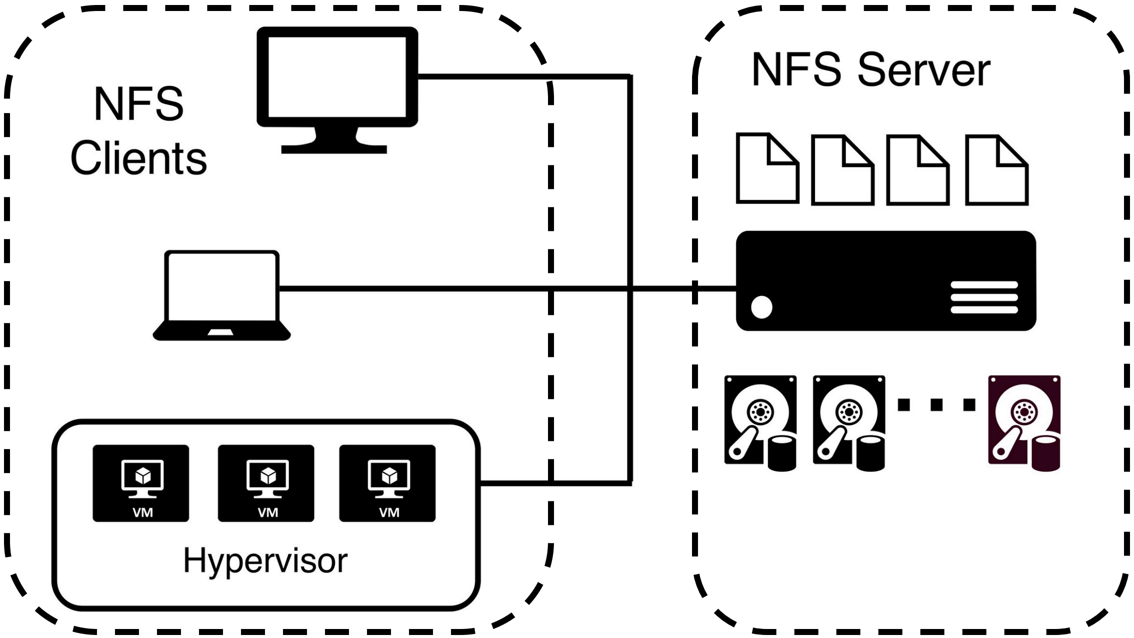
Ming Chen
Erez Zadok
{mchen, ezk}@cs.stonybrook.edu
Stony Brook University

Arun O. Vasudevan
aov@nutanix.com
NUTANIX

Kelong Wang
kelong@dssd.com
EMC² DSSD

Network File System (NFS)

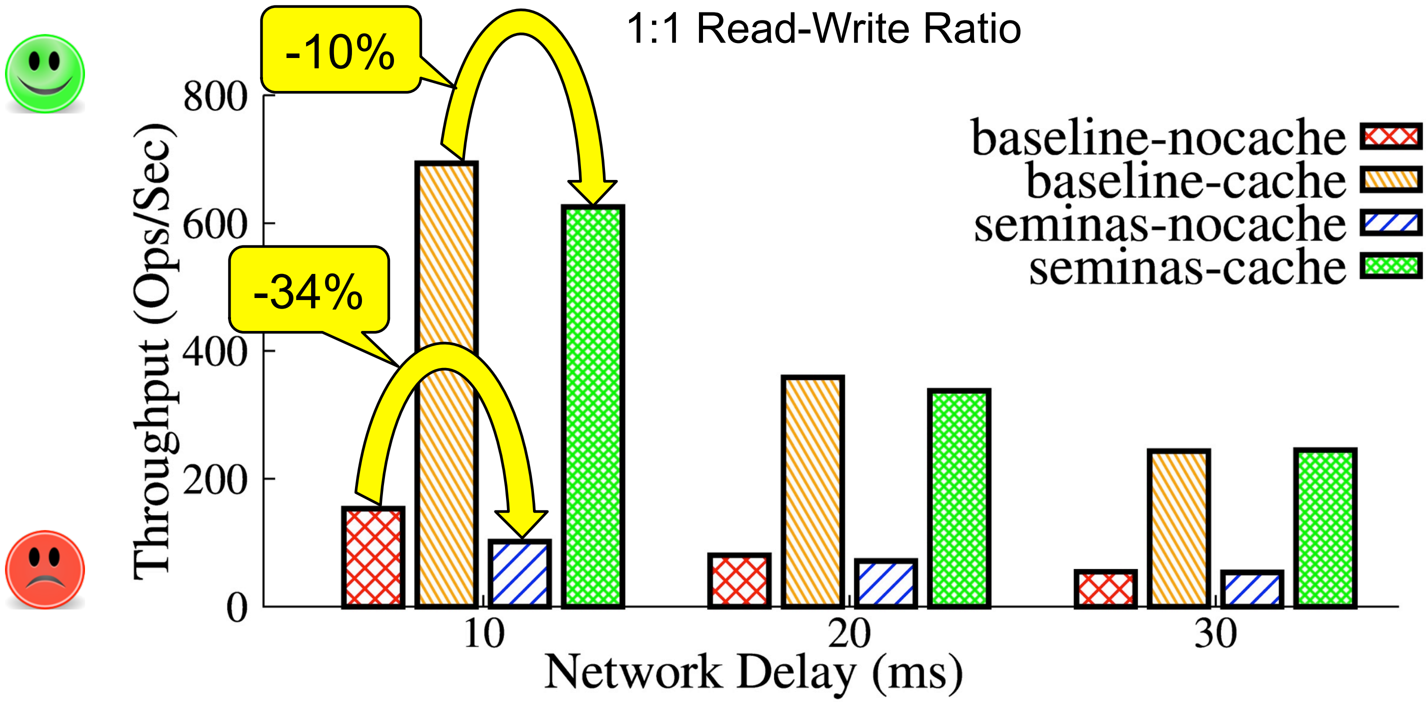
- An IETF standardized storage protocol
- Provides transparent remote file access
- Shares files over networks

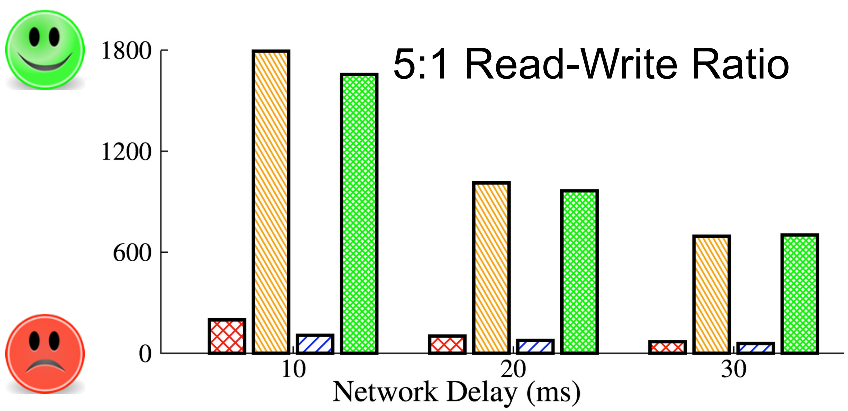


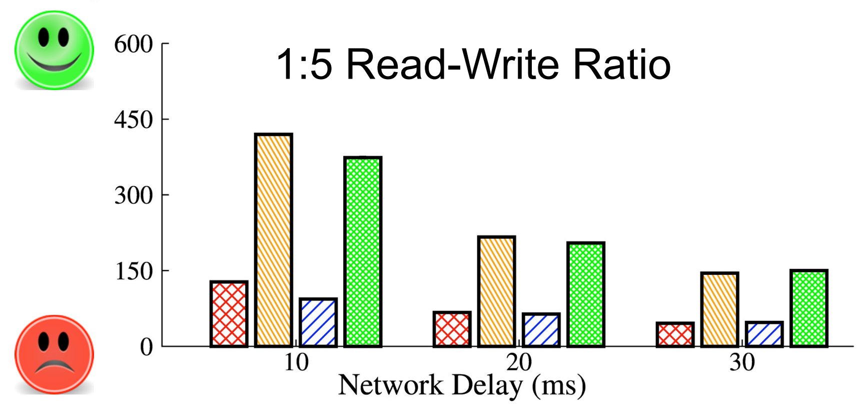
Methodology

- Benchmaster
  - Automate multiple runs of experiments
  - Launch workloads concurrently on clients
  - Periodically collect system statistics
- Workloads
  - Data-intensive workloads
  - Metadata-intensive workloads
  - Delegation workloads
  - Filebench macro-workloads

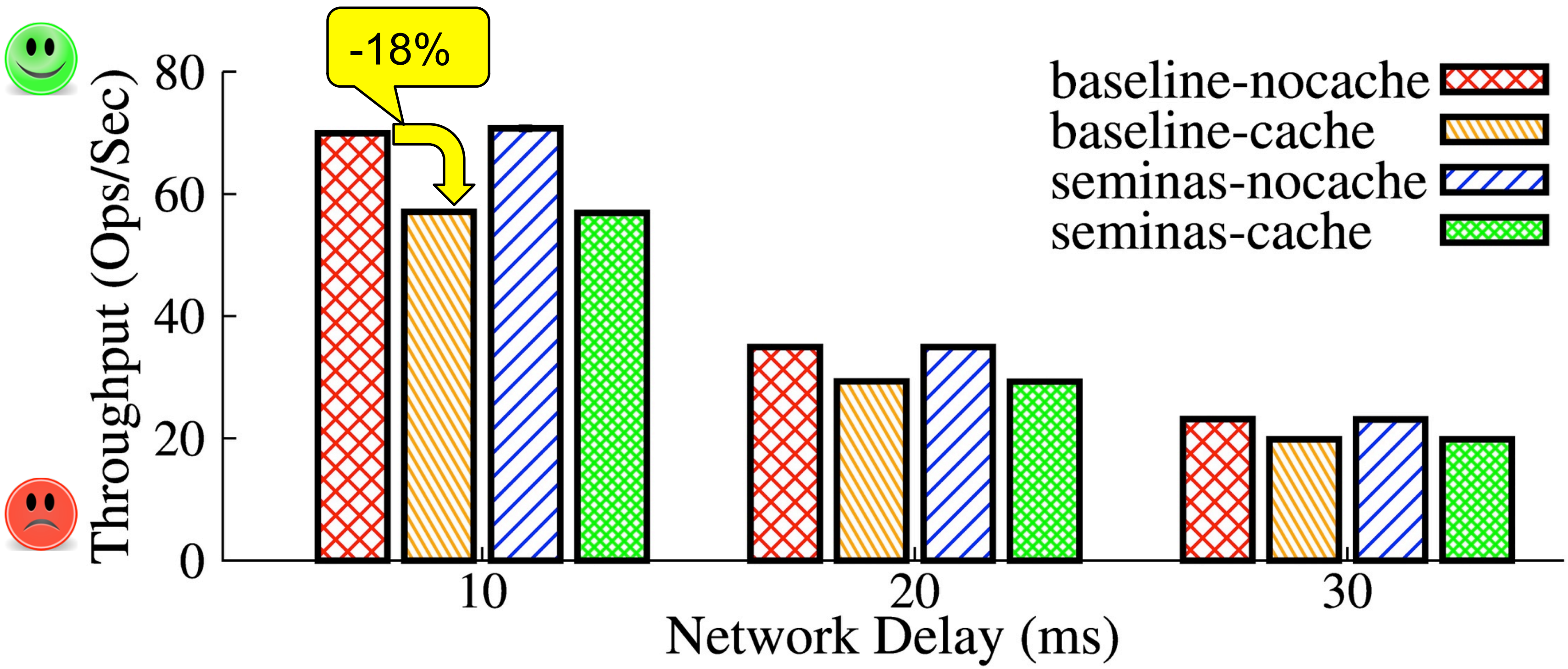
Random Read/Write





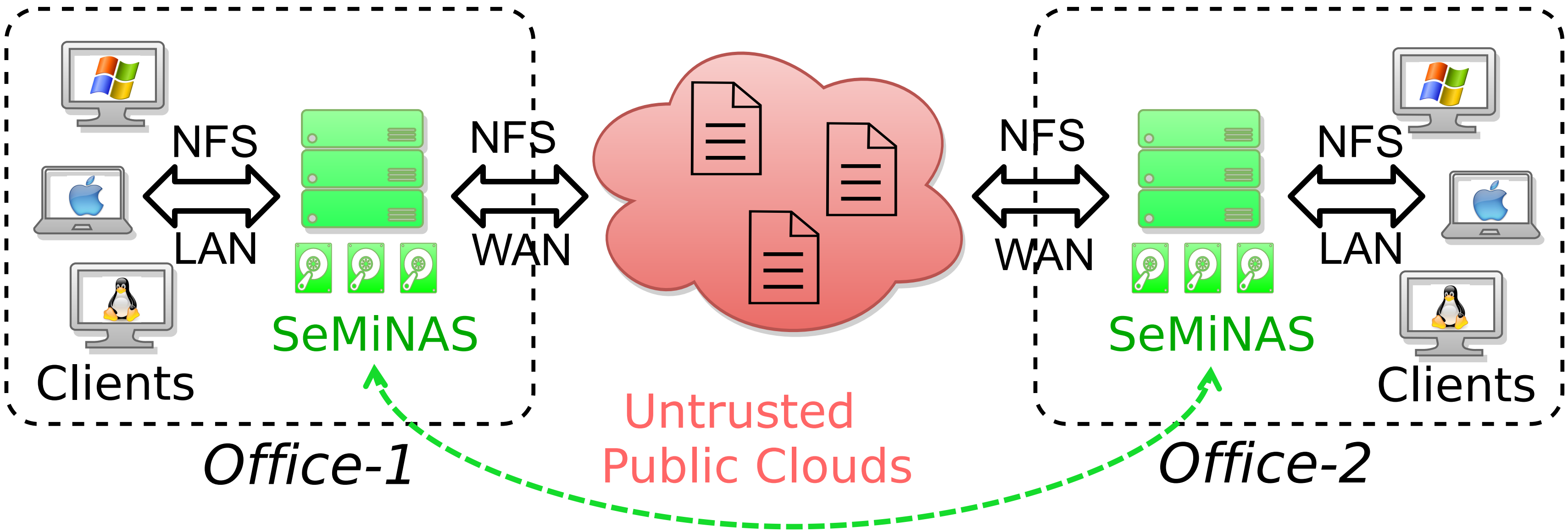


File-Deletion Workload



- Caching makes file deletion slower
  - Introduce extra network round-trip
  - Remove cache upon `unlink()`
- However, SeMiNAS does not make file deletion slower

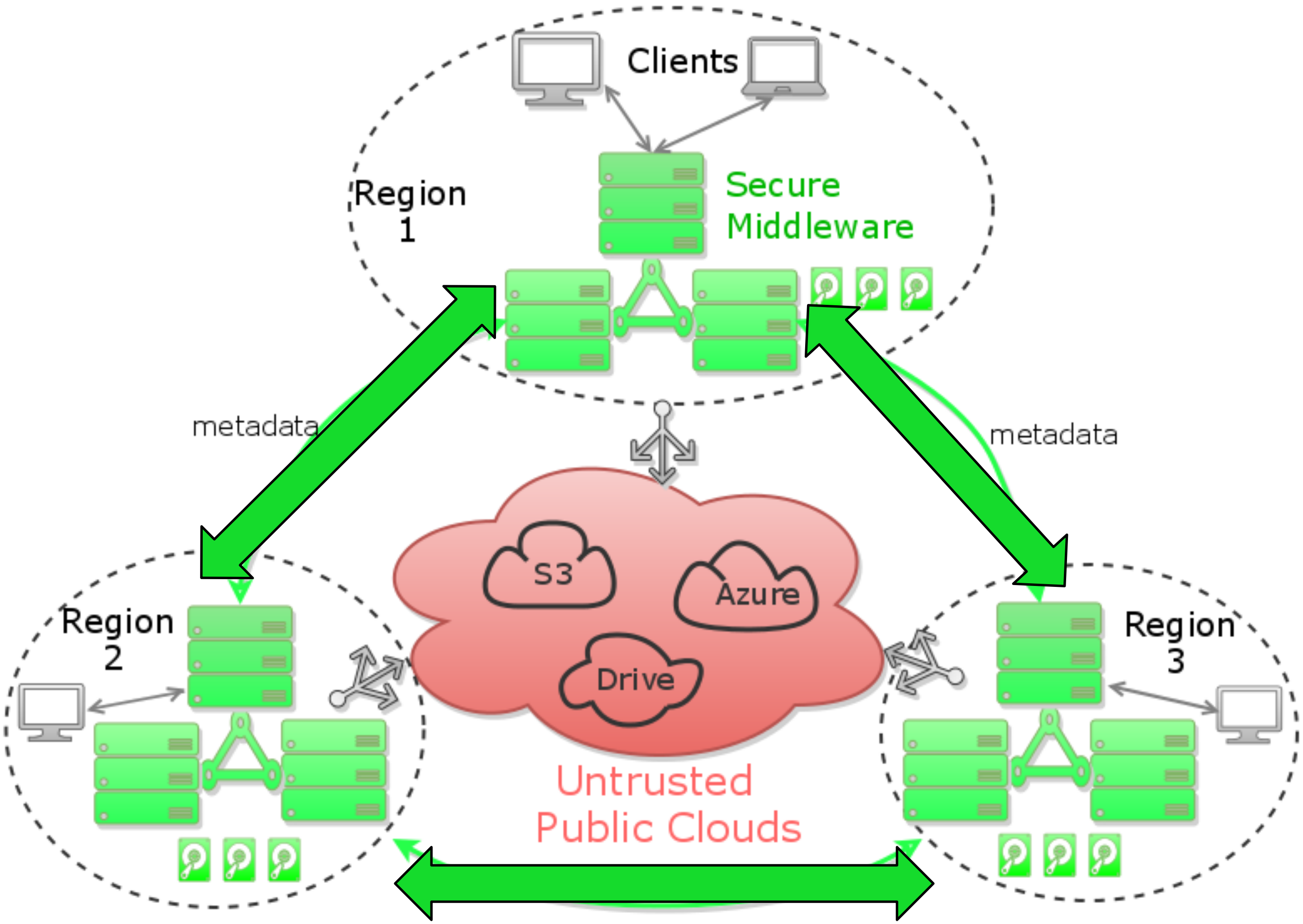
SeMiNAS



- Goal: Securely and efficiently store and share files in cloud for geo-distributed organizations.
- Approach: take advantages of new opportunities in NFSv4 and Data Integrity eXtensions (DIX).

Stony Brook University

Kurma Architecture



Kurma Components