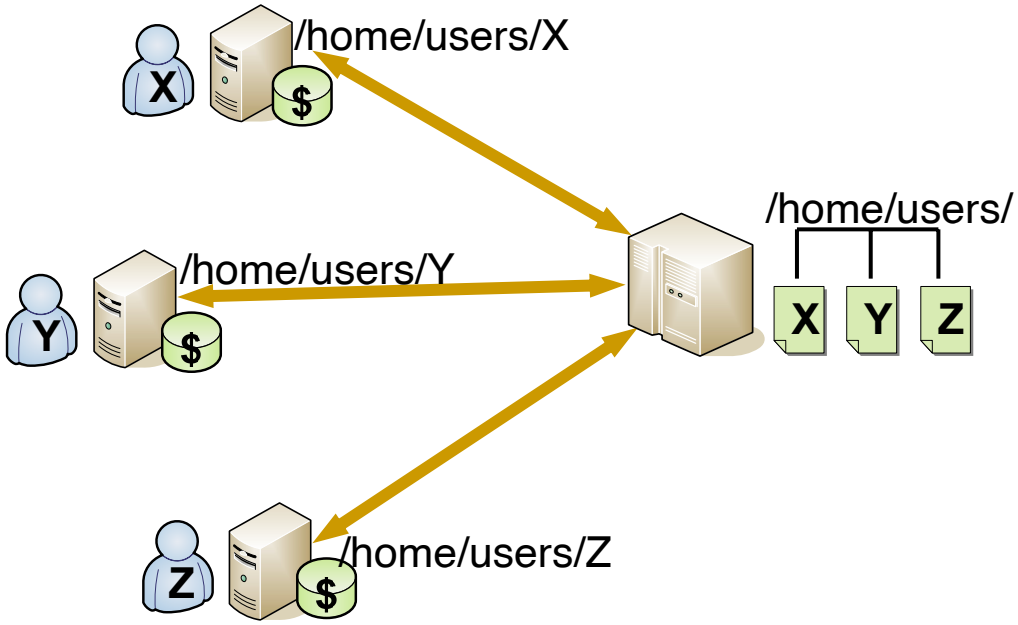
Application-Tailored Cache Consistency for Wide-Area File Systems

Ming Zhao, Renato Figueiredo

Advanced Computing and Information Systems (ACIS)
Electrical and Computer Engineering
University of Florida
{ming, renato}@acis.ufl.edu

Motivation

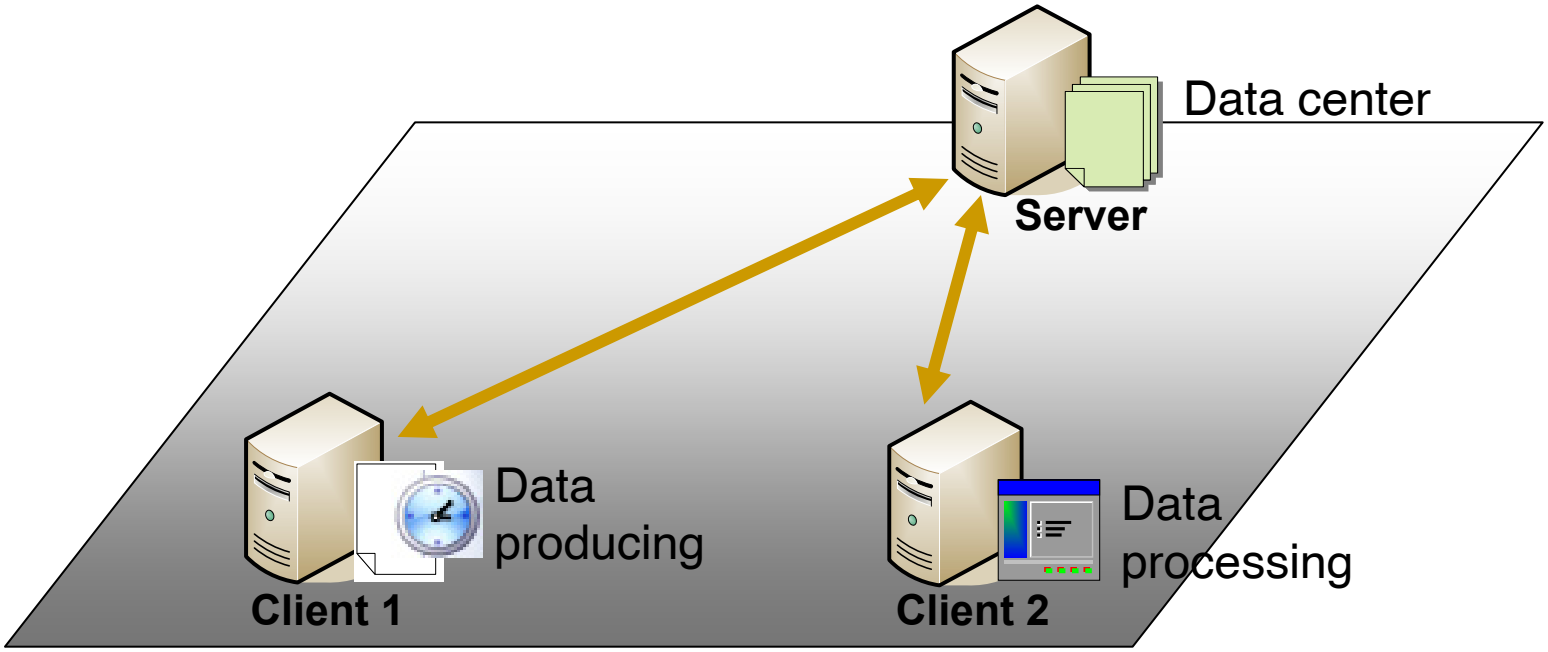
- Traditional distributed file system (DFS)



- Static setup, homogeneous configurations
  - Unable to integrate application-tailored features
- Usually supports only one consistency model
  - Network File System (NFS): mainly periodic revalidation polling
- But application-tailored DFS (cache consistency) is key
  - Performance, coherence, complexity, scalability …
  - Especially in WAN (high latency, limited bandwidth …)

Motivation

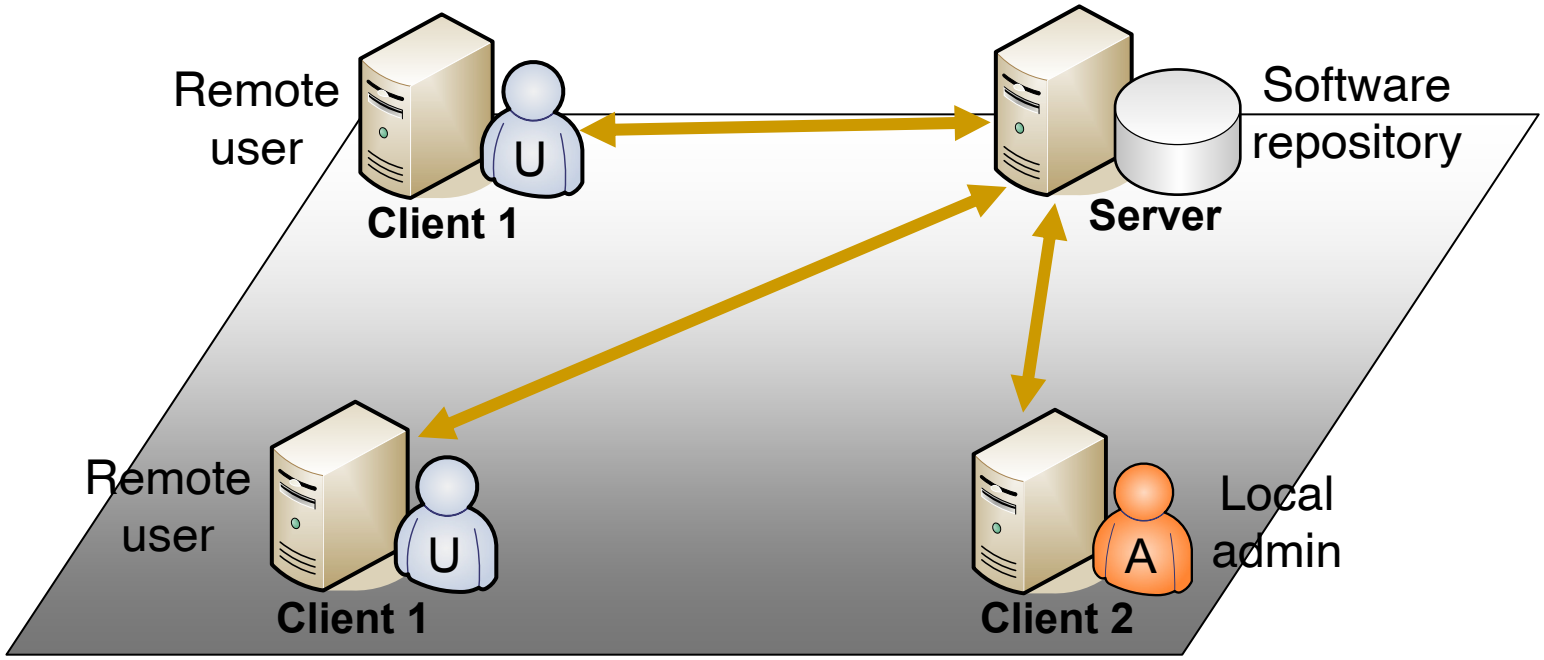
- Real-time scientific data processing
  - Timely processing of new data



[CLADE-2004]

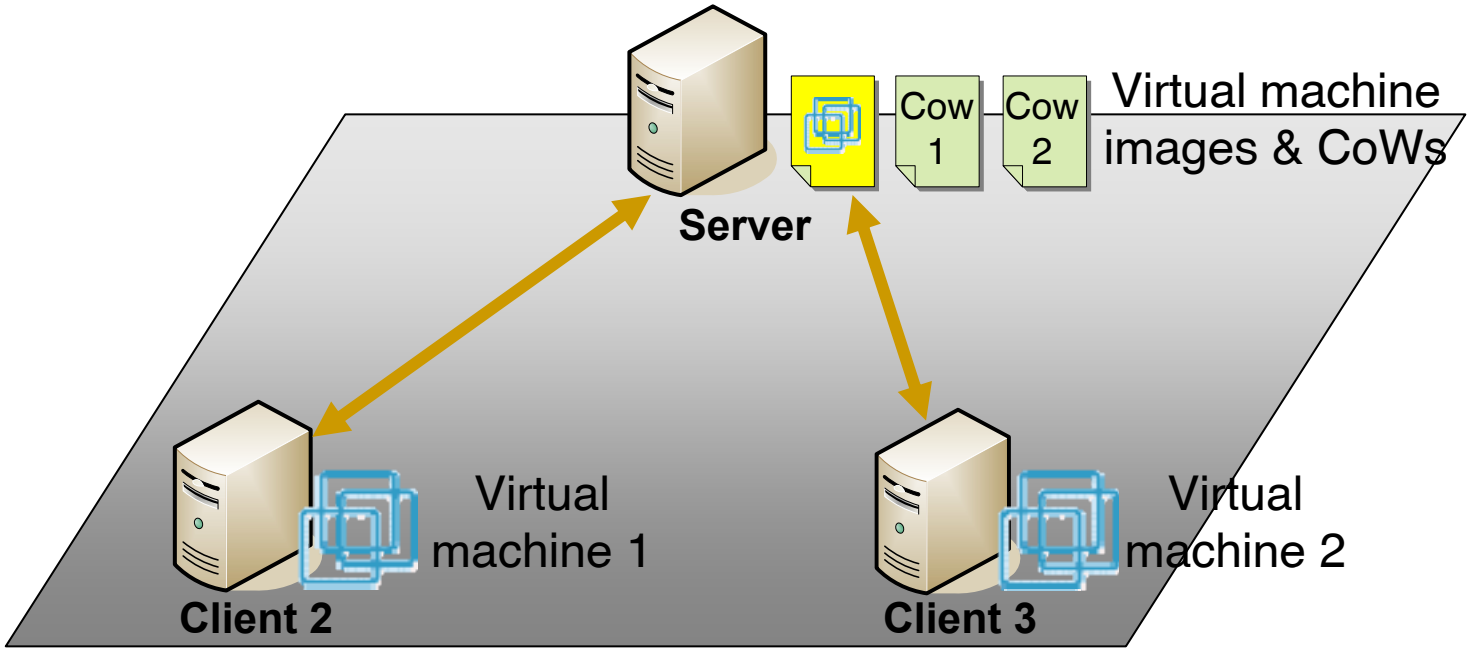
Motivation

- Wide-area software repository
  - Effective caching of read-only data



Motivation

- Virtual machine based grid computing
  - Aggressive read caching of shared image and write caching of independent Copy-on-Write (CoW) state



[Cluster Computing, Vol 9, 2006]

Overview

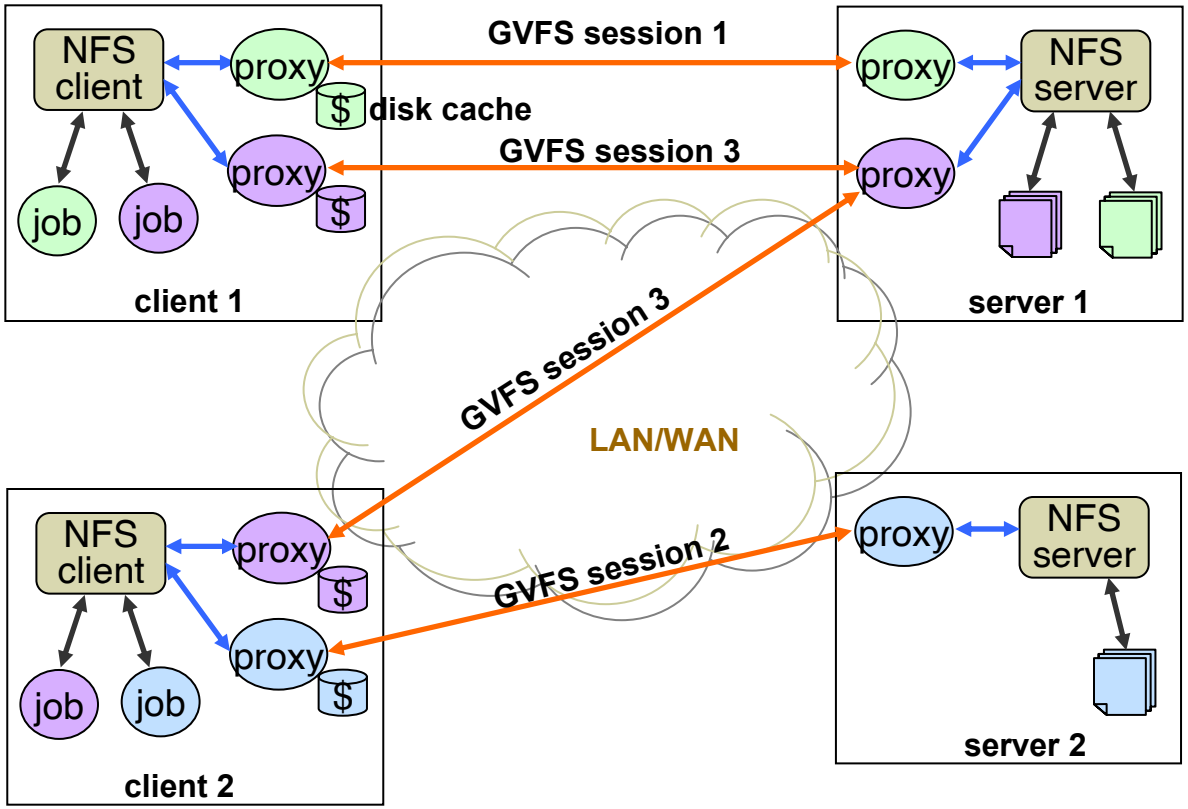
- Goal:
  - Application-tailored cache consistency for wide-area FS
- Challenges:
  - How to customize DFS with different cache consistency?
  - What consistency models to provide for different applications?
- Contribution:
  - User-level DFS virtualization (GVFS) supporting per-session, per-application customization
  - Cache consistency models tailored to application needs (relaxed and strong)

Outline

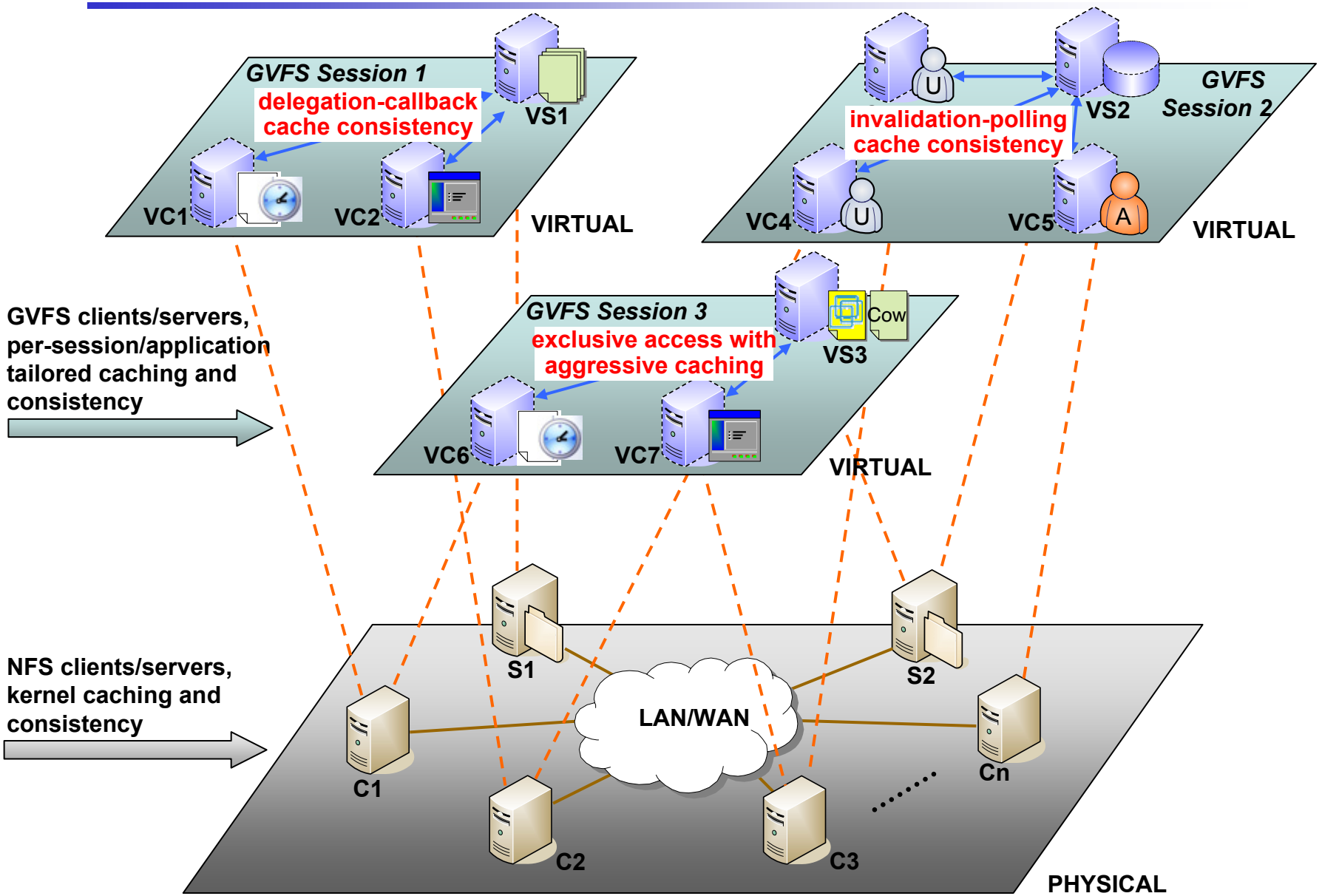
- Background
  - Grid virtual file system (GVFS)
- Cache consistency models
  - Invalidation-polling based consistency
  - Delegation-callback based consistency
- Experimental evaluation
- Summary

Grid Virtual File System (GVFS)

- DFS virtualization through user-level NFS proxies
- Enhancements for Grid (disk caching, secure tunneling)
- Dynamic, independent, application-tailored sessions managed by Grid middleware **[HPDC-2005]**



Architecture

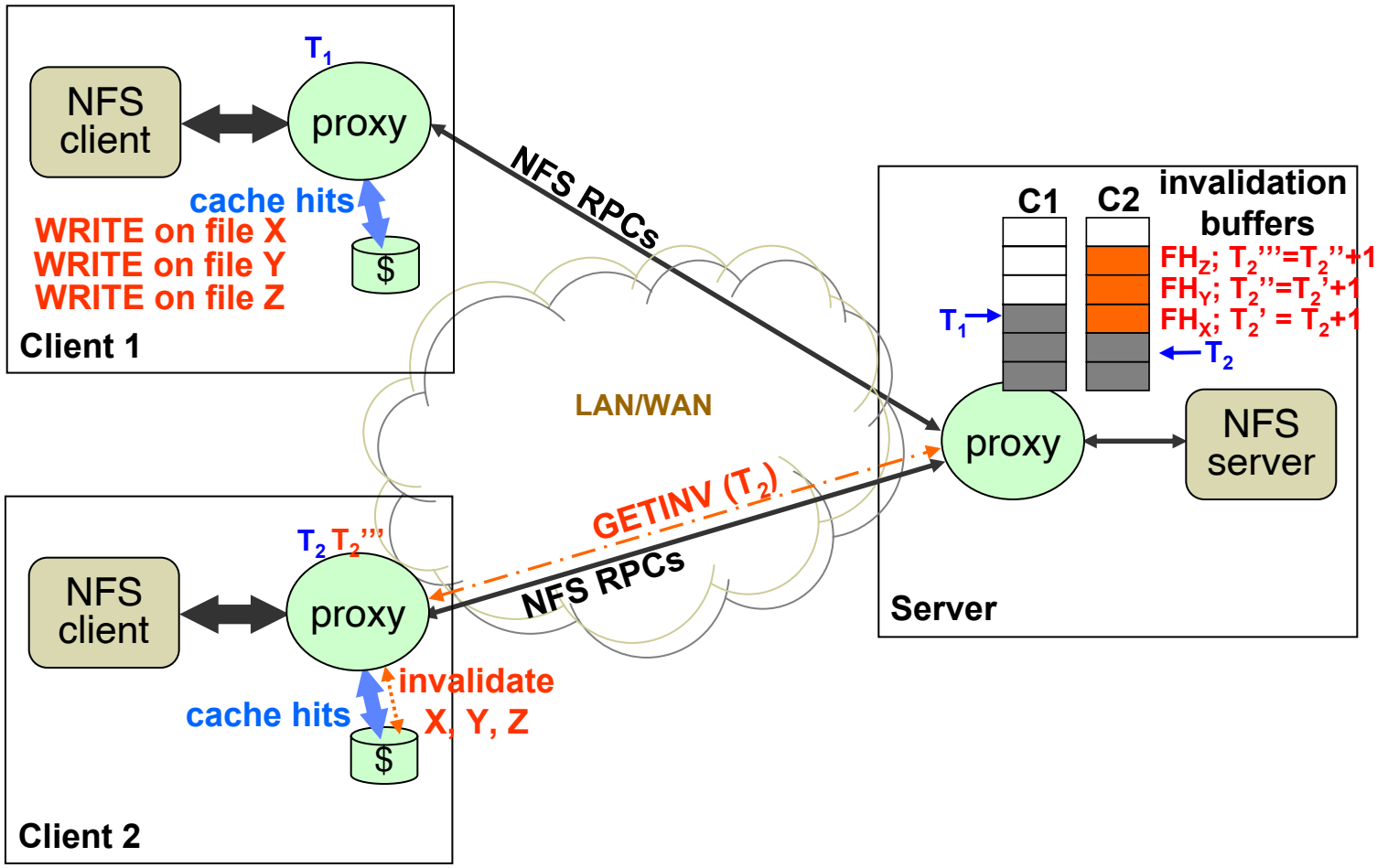


Invalidation-Polling Cache Consistency

- Relaxed consistency with significant performance improvement

- Protocol
  - Proxy-server tracks modifications
    - Per-client invalidation buffer stores information of files potentially modified by other clients
    - Each buffer entry stores the file's NFS handle with an access time stamp
  - Proxy-client polls for invalidations
    - Uses a new inter-proxy procedure call: GETINV
    - Polls proxy-server's invalidation buffer periodically
    - Invalidates cached attributes of the concerned files

Invalidation-Polling Cache Consistency



Invalidation-Polling Cache Consistency

- Strength
  - Substantially reduces the number of consistency calls by batching the necessary invalidations
  - Significantly improves application performance and reduces network and server load
  - Suitable for applications that can tolerate modest inconsistency (adjustable polling period)

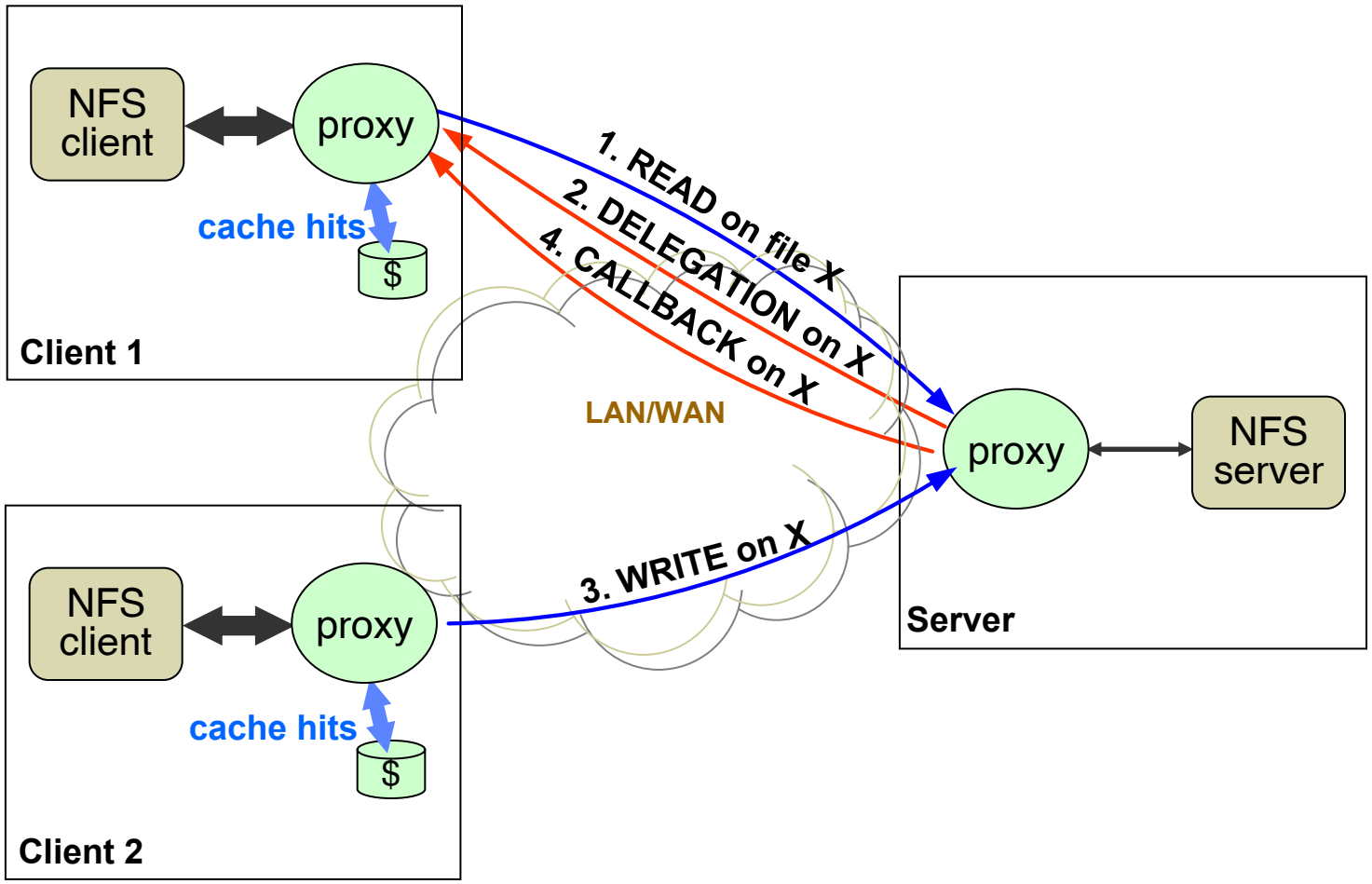
- Failure handling
  - Safe soft states (invalidation buffers, timestamps)
  - Server recovery by bootstrapping clients with new timestamps (forcing invalidating all cached attributes)
  - Client recovery by invalidating all cached attributes and acquiring the latest timestamp

Delegation-Callback Cache Consistency

- Stronger consistency with also performance improvement

- Protocol
  - Per-file delegation
    - Read-delegation: allows use of cached data w/o revalidation
    - Write-delegation: allows delay of writes
    - Speculates file open and close by tracking data access
    - Disable native kernel NFS' attributes caching
  - Recall of delegation
    - Proxy-server to proxy-client procedure call: CALLBACK
    - Recall of read-delegation: invalidates cached attributes
    - Recall of write-delegation: also forces write-back

Callback-Delegation Cache Consistency



Callback-Delegation Cache Consistency

- Strength
  - Ensures strong consistency
  - Reduces the number of consistency calls and improves the performance when writes are infrequent
  - Suitable for applications that rely on strict consistency

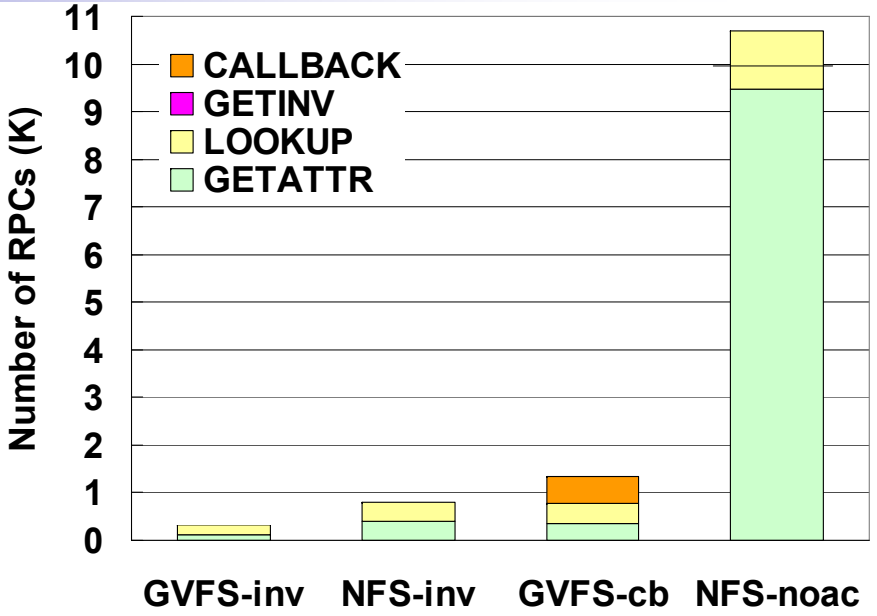
- Failure handling
  - Server recovery by proxy-server's special callbacks
    - Proxy-client with read delegation revalidates cached files
    - Proxy-client with write delegation returns dirty file list
  - Client recovery by reclaiming delegations
    - Invalidates all cached attributes
    - Writes back a single block for each dirty file

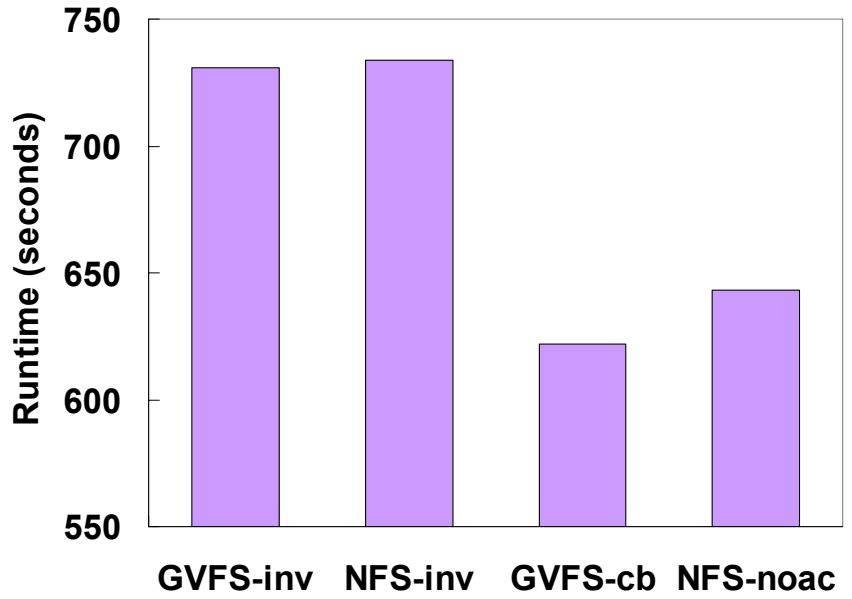
Experimental Setup

- GVFS sessions
  - Based on virtualization of NFS v3
  - Deployed on VMware-based virtual machines
  - Setup with NIST Net emulated WAN links
- Benchmarks
  - Micro benchmarks
  - Real scientific applications

Shared File-based Locks

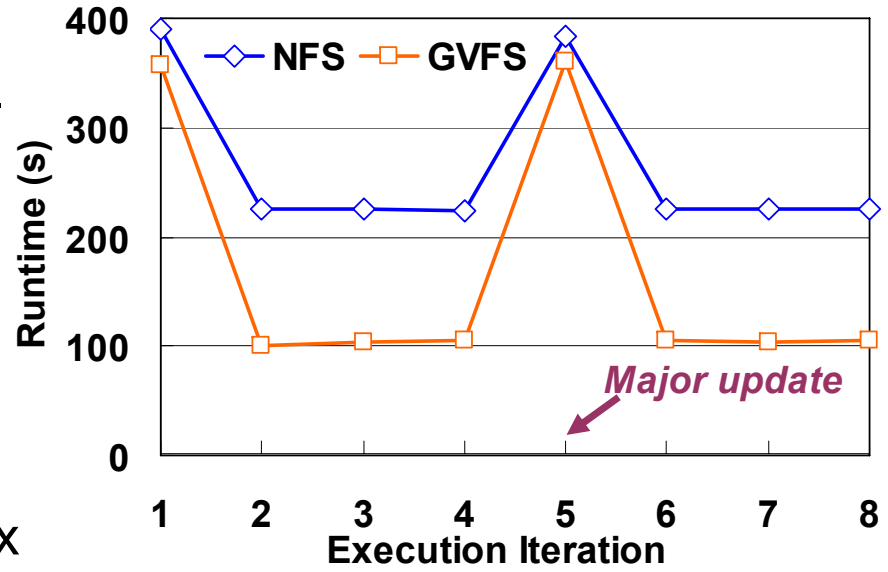
- Setup:
  - Six distributed clients compete for a file-based lock
- Cases:
  - *NFS-inv* (typical NFS setup) **vs.** *GVFS-inv (*invalidation-polling)
  - *NFS-noac* (no attributes cache) **vs.** *GVFS-cb* (delegation-callback)
- Observation:
  - GVFS substantially reduces the number of consistency calls and improves the performance

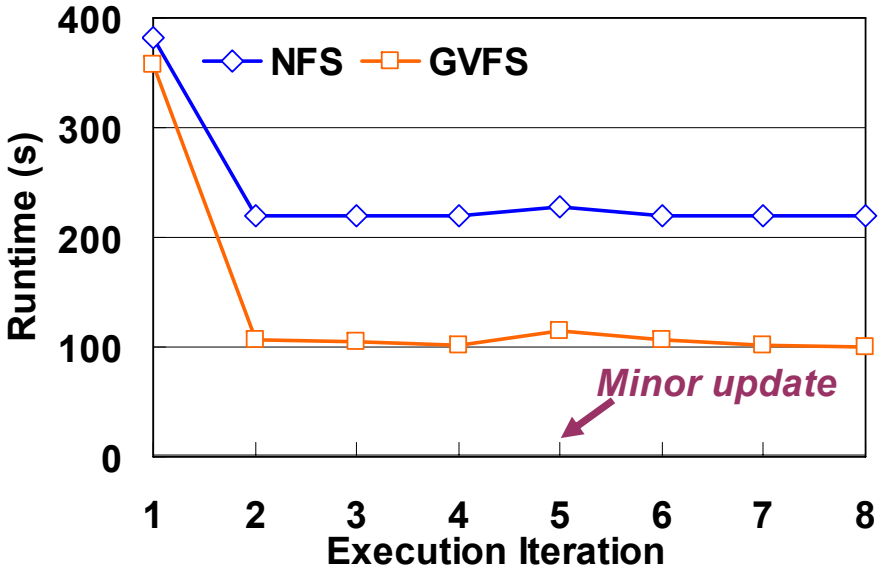




Software Repository

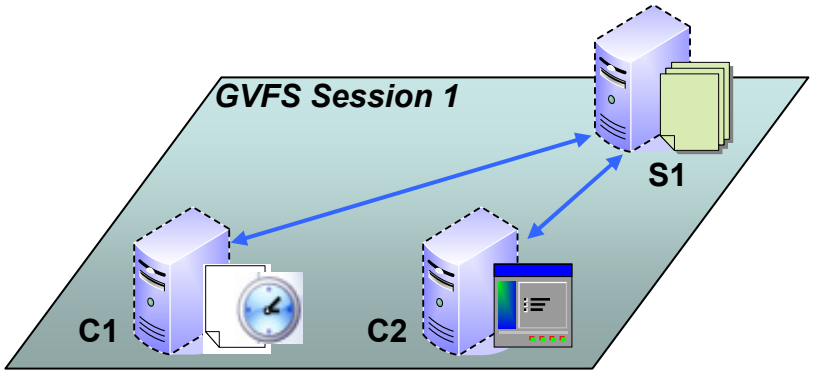
- Benchmark:
  - NanoMOS (MATLAB-based 2-D n-MOSFET simulator)
- Setup:
  - Software used by 6 users (C1-C6), updated by 1 administrator (C7):
  - *Major update*: entire MATLAB
    Minor update: only one MPI toolbox
  - *Typical NFS* **vs.** *GVFS invalidation-polling*
- Observation:
  - NFS: revalidates all the accessed files all the time
  - GVFS: only invalidates modified files and batches them in a few GETINVs

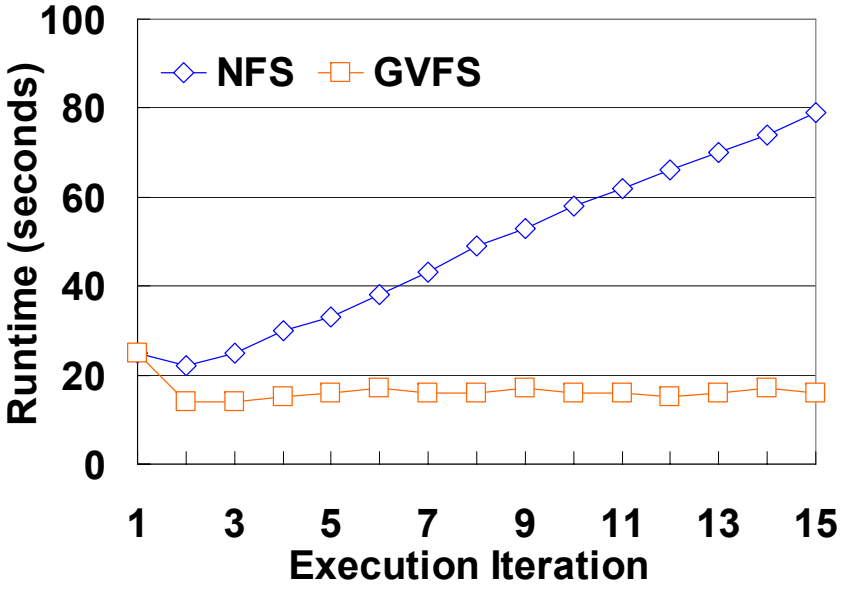




Scientific Data Processing

- Benchmark:
  - CH1D (coupled hydrodynamics simulation and post-processing)
- Scenario:
  - Data accumulated on-site (C1), processed off-site (C2)
  - 30 more files generated by C1 before each run on C2
  - *Regular NFS* **vs.** *GVFS delegation-callback*
- Observation:
  - As input dataset grows overhead caused by consistency checks:
    - Grows linearly in native NFS,
    - Stays constant in GVFS





Related Work

- Data access customizations based on system-call or library-based interception
  - UFO, BAD-FS, Condor
  - GVFS: better support for unmodified applications, OSs
- User-level DFS extensions based on RPC interception
  - Automounter, CFS, SFS, LegionFS
  - GVFS: support for dynamic application-tailored DFS sessions
- Kernel-level DFS improvements
  - AFS, CacheFS, NQ-NFS, Spritely NFS, NFS v4
  - GVFS: easy to deploy across Grid, support per-user/-application customizations

Summary

- Problem:
  - Application-tailored cache consistency for wide-area file systems
- Solution:
  - GVFS-based per-session, per-application DFS customization with relaxed or strong cache consistency
- Result:
  - Substantial reduction on consistency-related calls
  - Significant performance improvement for unmodified applications in wide-area/Grid environment

References

[1] J. Paladugula, M. Zhao, R. Figueiredo, “Support for Data-Intensive, Variable-Granularity Grid Applications via Distributed File System Virtualization - A Case Study of Light Scattering Spectroscopy”, CLADE-2004.

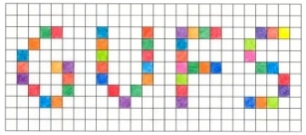
[2] M. Zhao, J. Zhang, R. Figueiredo, “Distributed File System Virtualization Techniques Supporting On-Demand Virtual Machine Environments for Grid Computing”, *Cluster Computing*, Vol 9/1, 2006.

[3] M. Zhao, V. Chadha, R. Figueiredo, “Supporting Application-Tailored Grid File System Sessions with WSRF-Based Services”, HPDC-2005.

[4] M. Zhao, J. Xu, R. Figueiredo, “Towards Autonomic Grid Data Management with Virtualized Distributed File Systems”, ICAC-2006.

In-VIGO: Virtualization middleware for computational Grids
http://www.acis.ufl.edu/invigo

GVFS: Virtualized distributed file system for Grid environment
http://www.acis.ufl.edu/~ming/gvfs



Acknowledgments

- In-VIGO team
  - http://invigo.acis.ufl.edu
- Justin Davis, Peter Sheng (Department of Civil & Coastal Engineering, University of Florida)
- NSF Middleware Initiative
- NSF Research Resources
- IBM Shared University Research

- **Questions?**