Distributed File System Support for Virtual Machines in Grid Computing

Ming Zhao, Jian Zhang, Renato Figueiredo

Advanced Computing and Information Systems
Electrical and Computer Engineering
University of Florida

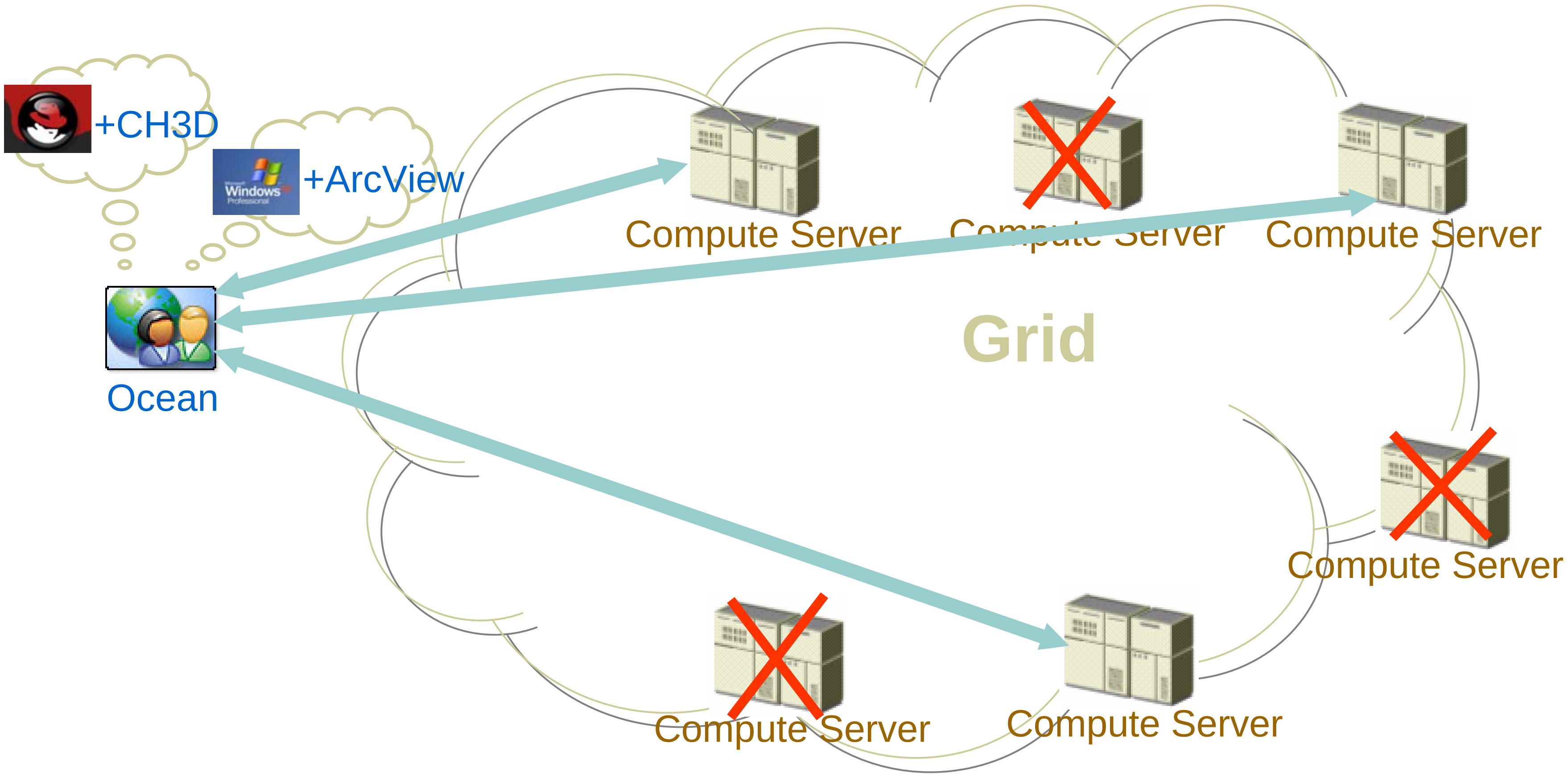
Overview

- Goal: Support virtual machines (VMs) as execution environments for Grid computing
- Challenge: Efficient and on-demand transfer of large VM state in Grids
- Contribution: User-level extensions to distributed file system, optimized for VM state transfer, supporting unmodified VM technology

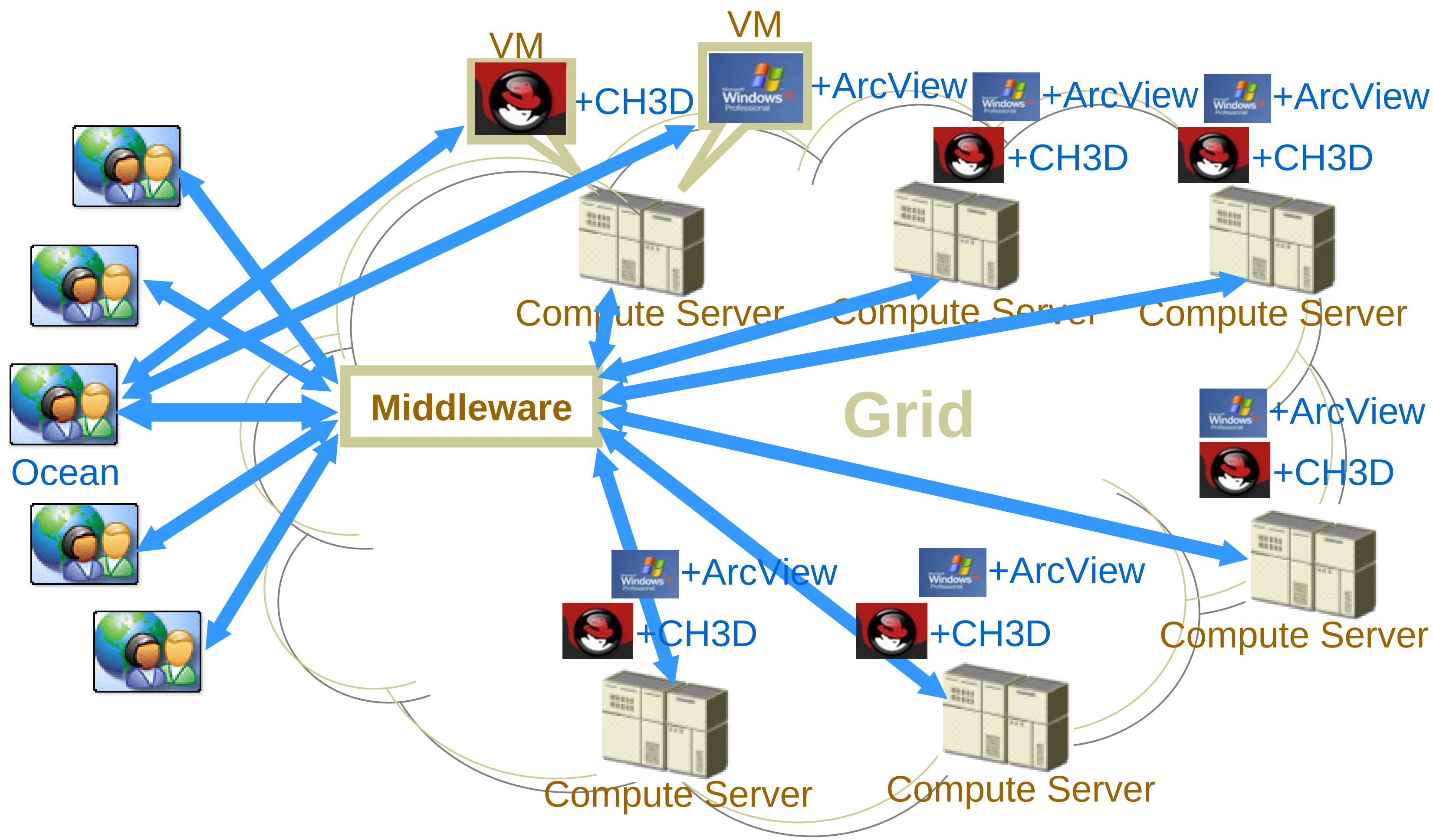
Grid Computing on Virtual Machines

- Fundamental goal of Grid computing:
  - Seamlessly multiplexing distributed computational resources of providers among users
- Key challenge to Grid middleware:
  - The provisioning of execution environments that have flexible, customizable configurations and allow for secure execution of untrusted code from Grid users
- Virtual Machines for Grid Computing [*ICDCS'03*]
  - Resource security and user isolation
  - Flexible customization and legacy support
  - Site independent deployment and migration

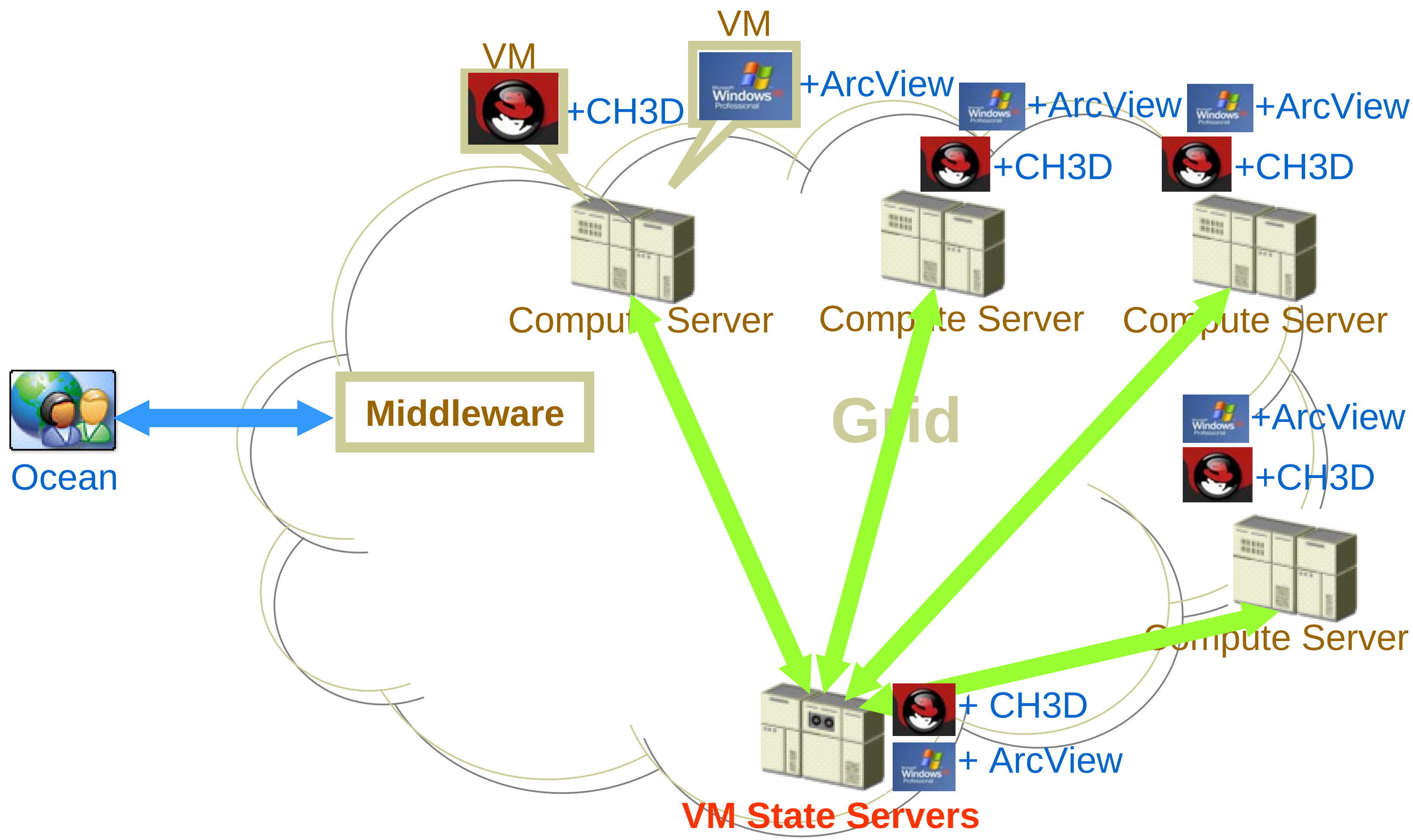
Application-Centric Solution



Application-Centric Solution



Challenge: VM State Transfer



Dynamic, efficient transfer of large VM state is important

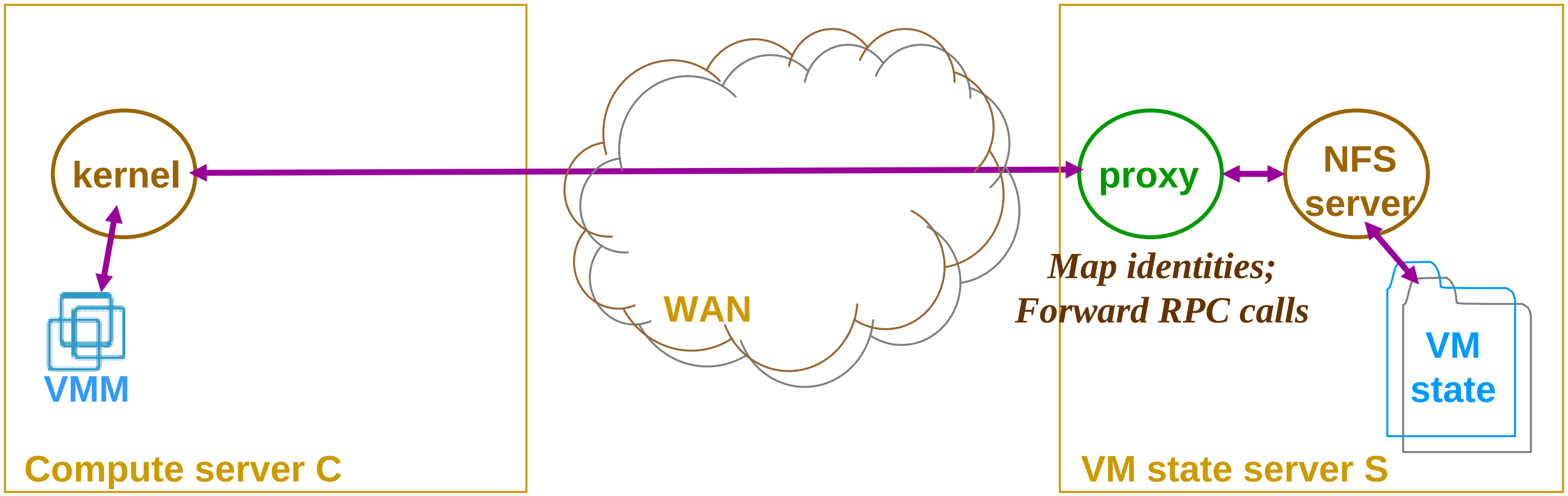
Outline

- Background
- Approach
- Performance
- Summary

VM State

- Defines hardware/software state of a VM
  - Includes disk state and memory state
  - Stored in file systems by VMMs (VMWare, UML, Xen)
- How to transfer?
  - VM memory state (order of hundreds of MBytes)
    - Entire state needed to resume a VM
    - Full file transfer desirable
  - VM disk state (order of several GBytes)
    - State only partially needed by VM
      - "Reboot + run SpecSEIS" accesses <5% of 1.3GB disk state
    - Partial file transfer desirable
      - Full file transfers: long start-up latencies, unnecessary storage
- Both supported by GVFS (Grid Virtual File System)

Grid Virtual File System (GVFS)



- Logical user accounts [*HCW'01*] and Virtual file system [*HPDC'01*]
  - Shadow account + file account, managed by middleware
  - NFS call forwarding via middle tier user-level proxy
  - User identities mapped by proxy
- Extended to improve support for VM state transfer

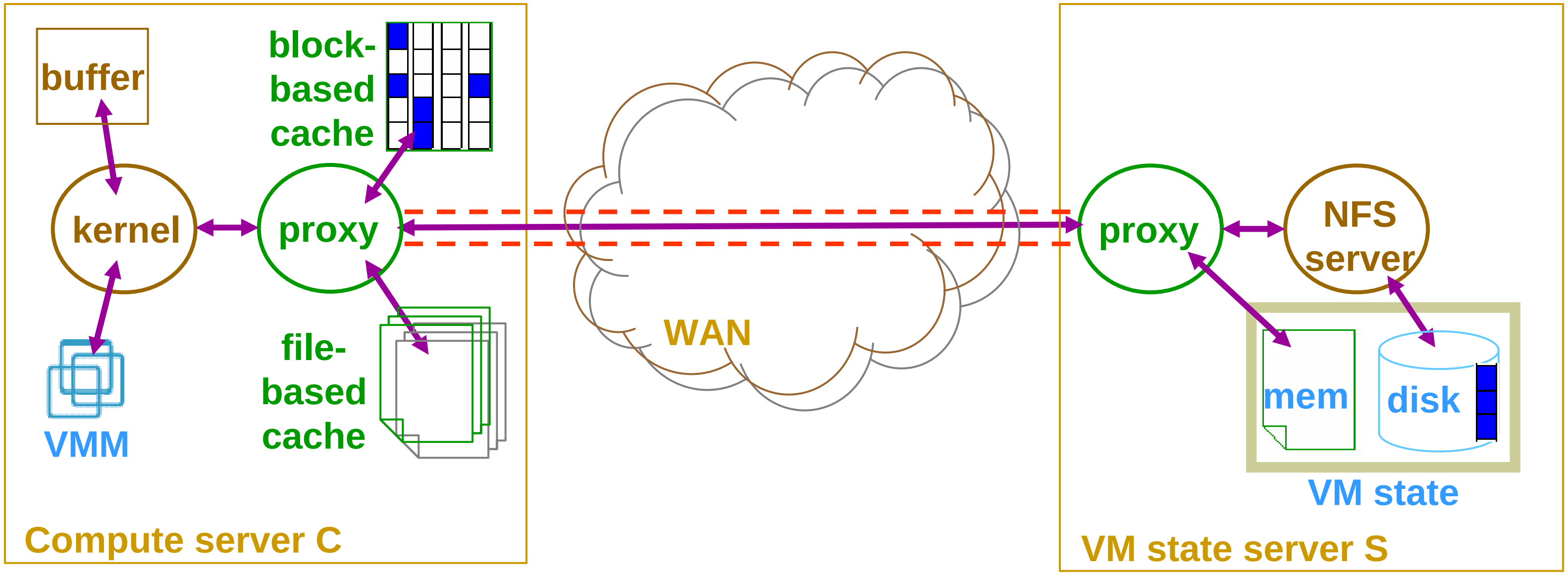
Efficient Access to VM Disk State

- User-level disk caching
  - Block-based cache, 2nd level to kernel buffer
  - Write-back policy for write operations
- Per-application caching policy
  - Cache size, write policy, etc.
  - Enables file-based disk caching by meta-data handling
- Middleware-driven consistency models
  - O/S signals for write-back/flushing of cache contents
  - Independent tasks, high-throughput computing
- On-demand partial state transfer

Efficient Access to VM Memory State

- Meta-data generated by middleware can be used to accelerate data transfers
  - Stored in meta-data file, transparent to application
  - Captures application-specific data characteristics
  - Defines actions to be processed by GVFS proxy
- Meta-data handling for VM memory state
  - Needed in entirety to resume; highly compressible
  - Actions: “compress”, “remote copy”, “decompress”, and “read locally”
- On-demand full state transfer and caching

User-level Extensions



- Client-side proxy disk caching
- Application-specific meta-data handling
- Encrypted file system channels and cross-domain authentication [*TR-ACIS-03-001*]

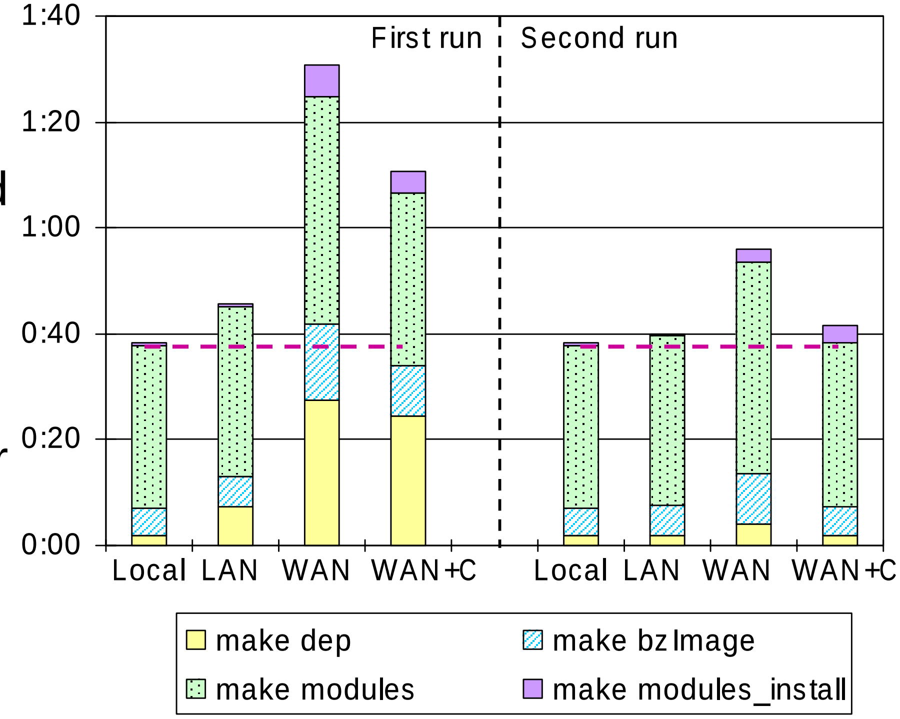
- User-level, leveraging ubiquitous NFS deployments; not application-specific but can be application-aware

Experiments: Application Execution

- Benchmarks:
  - Linux kernel compilation (development environment)
  - SPECseis96 (compute- and I/O-intensive)
  - LaTeX (interactive environment)
- Scenarios:
  - Local: VM state stored on a local disk
  - Remote (GVFS mounted):
    - LAN server
    - WAN server
    - WAN+Cache

Experiments: Application Execution

- Linux kernel compilation:
  - "Warm" proxy disk cache allows WAN+Cache outperforms WAN >30%; overhead <9% compared to Local
- SPECseis96:
  - Write-back effectively reduces write latency and avoids transfer of temporary data; WAN+Cache is 33% faster than WAN
- LaTeX:
  - Interactive sessions: small start-up overhead; response times comparable to Local when data partially reused across sessions

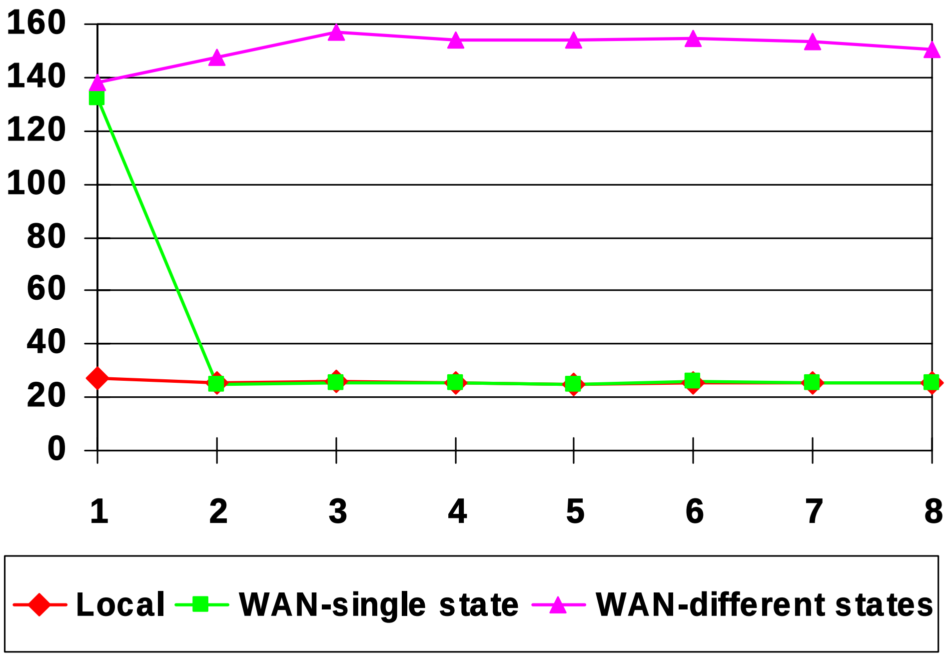


Execution times (hours:minutes) of Linux kernel compilation

Experiments on VM Cloning

- VM cloning:
  - Creates a run-time VM instance from stored VM state
  - File copy (memory state), symbolic link (disk state)
- Scenarios:
  - Local: VM state stored on a local disk
  - WAN: VM state GVFS-mounted
    - single state-sequential (locality)
    - different states-sequential (no locality)
    - single state-parallel (scalability)

VMWare VM Cloning



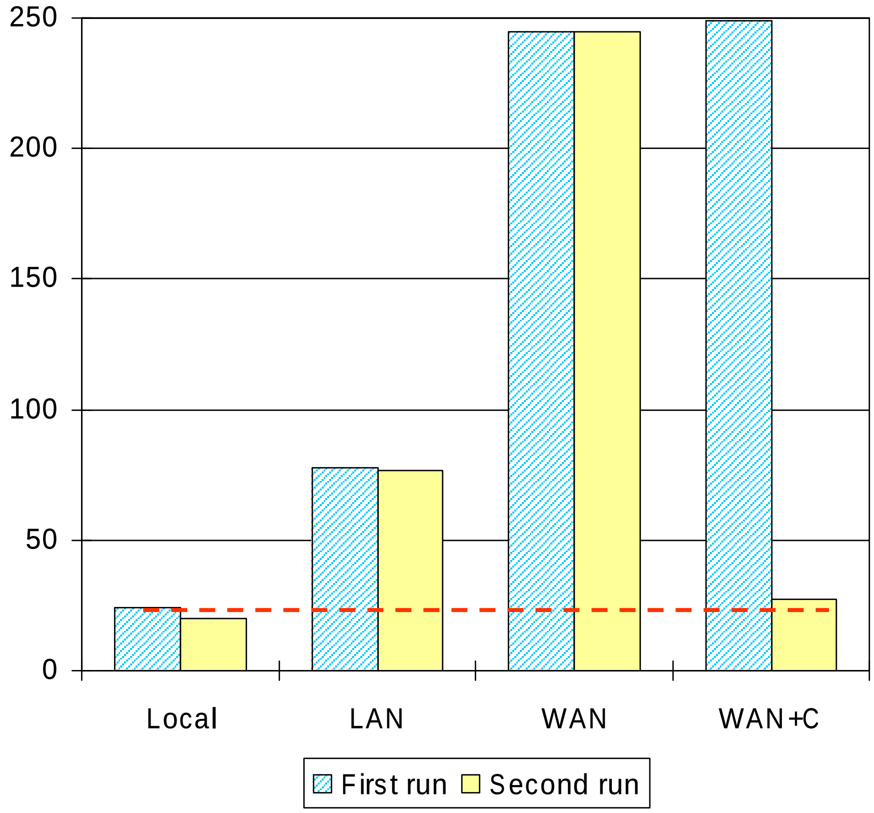
Sequential VM cloning times (seconds)

	Total time when caches are cold	Total time when caches are warm
WAN-single state-sequential	1056 seconds	200 seconds
WAN-single state-parallel	150.3 seconds	32 seconds

Sequential cloning time vs. parallel cloning time for eight VMs

- GVFS greatly reduces VM cloning time, and achieves speed close to the local disk setup if temporal locality exists across clone requests
- Good performance also achieved when VMs cloned in parallel in a cluster

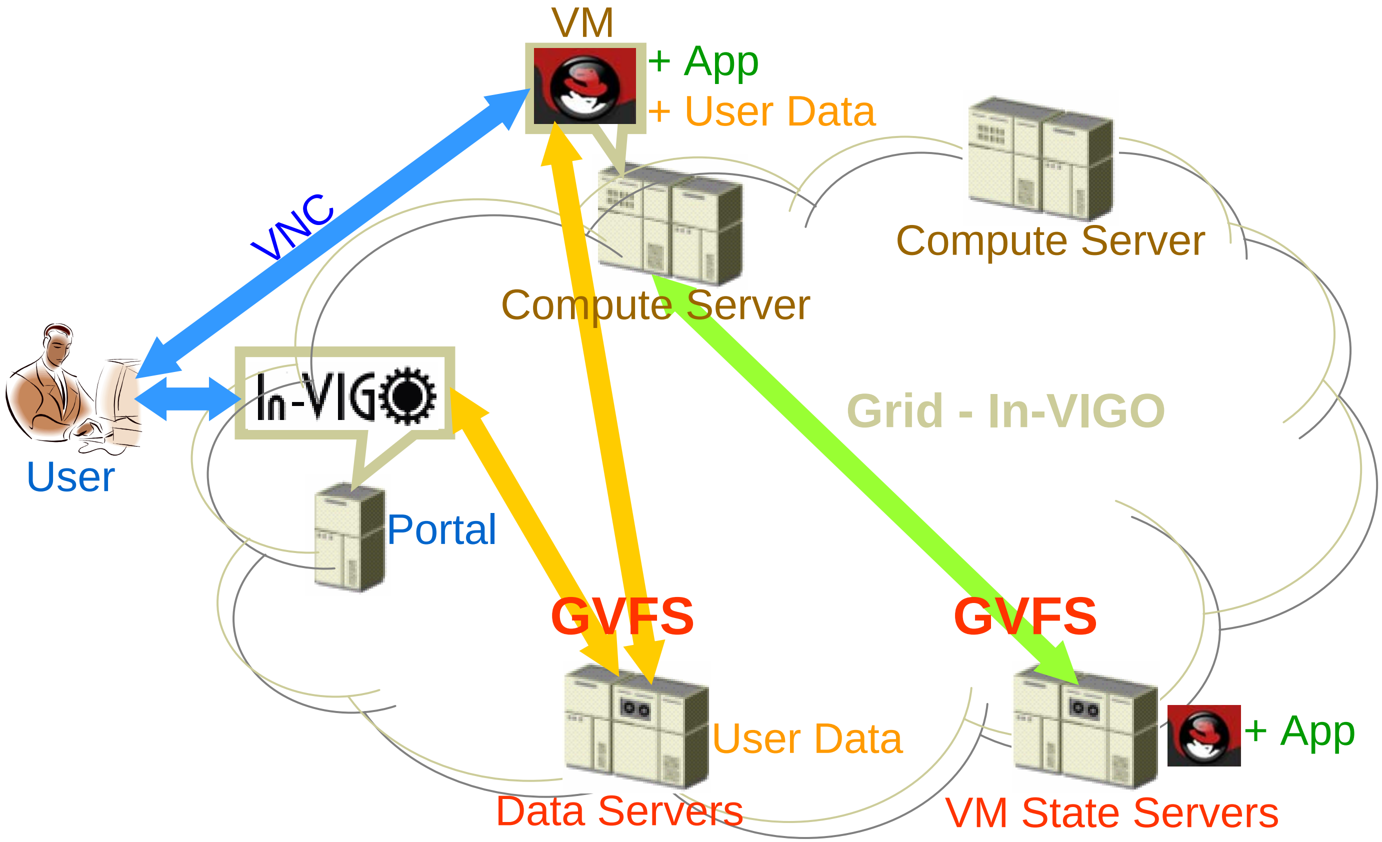
User-Mode Linux VM Startup



Booting times of UML (seconds)

- The initial overhead of WAN / WAN+C is >900% compared to Local
- Kernel memory buffer does not reduce overhead significantly
- "Warm" proxy disk cache brings the overhead down to <8 seconds; WAN+C outperforms WAN by >800%

The In-VIGO Approach



In-VIGO [*FGCS'04*]: Virtualization middleware for computational Grids

On-demand virtual resources:

- Machines (VMs)
- Data (GVFS)
- Accounts
- Interface (VNC)

Related work

- GridFTP, GASS
  - High performance transfer of large files
  - GVFS is a file system supporting block-based transfer for unmodified applications
- Condor, Kangaroo
  - Support on-demand remote data access, intermediates
  - GVFS supports statically linked applications
- Legion, SFS
  - Based on the de-facto NFS distributed file system
  - GVFS: dynamically, per-user/-application, middleware support, user-level extensions
- Collective, Internet Suspend/Resume
  - VM technology + distributed file system
  - GVFS: problem-solving environment, optimization for VM state transfer at file system level, support for unmodified VMMs

Summary

- Problem: VM-based Grid computing poses challenges on VM state management for Grid middleware
- Solution: GVFS supports efficient VM state transfer at the user-level
- Evidence: Experiments on both application execution and VM cloning show good performance

Future Work

- Efficient checkpointing and migration of VM instances
- Fine-grained consistency models by call-back and use of meta-data
- Dynamic profiling of application data access behavior
- Pre-fetching and high-bandwidth transfers using protocols such as GridFTP

Acknowledgments

- In-VIGO team at UFL
  - http://invigo.acis.ufl.edu
- Dr. Peter Dinda and Virtuoso team at NWU
  - http://virtuoso.cs.northwestern.edu
- NSF Middleware Initiative
  - http://www.nsf-middleware.org
- NSF Research Resources
- IBM Shared University Research
- VMWare

References

[*ICDCS’03*] *R. Figueiredo, P. A. Dinda, J. A. B. Fortes, “A Case for Grid Computing on Virtual Machines”, Proc. International Conference on Distributed Computing Systems (ICDCS), May 2003.*

[FGCS’04] S. Adabala, V. Chadha, P. Chawla, R. Figueiredo, J. Fortes, I. Krsul, A. Matsunaga, M. Tsugawa, J. Zhang, M. Zhao, L. Zhu, and X. Zhu. “From Virtualized Resources to Virtual Computing Grids: The In-VIGO System”, to appear, Future Generation Computing Systems (in press), 04/2004 .

[*HCW’01*] *N. Kapadia, R. Figueiredo and J. A. B. Fortes, “Enhancing the Scalability and Usability of Computational Grids via Logical User Accounts and Virtual File Systems”, Proceedings of HCW at IPDPS, April 2001.*

[*HPDC’01*] *R. Figueiredo, N. Kapadia and J. A. B. Fortes. "The PUNCH Virtual File System: Seamless Access to Decentralized Storage Services in a Computational Grid", Proceedings of HPDC, August 2001.*

[*TR-ACIS-03-001*] *R. Figueiredo, “VP/GFS: An Architecture for Virtual Private Grid File Systems”. In Technical Report TR-ACIS-03-001, ACIS Laboratory, Department of Electrical and Computer Engineering, University of Florida, 05/2003.*

In-VIGO prototype can be accessed from http://invigo.acis.ufl.edu; courtesy accounts available.

InVIGO
Processing....