VIDAS: OBJECT-BASED VIRTUALIZED DATA SHARING FOR HIGH PERFORMANCE STORAGE I/O

Pablo Llopis, Javier Garcia Blas, Florin Isaila, Jesus Carretero

Computer Architecture and Technology Group (ARCOS)
University Carlos III of Madrid, Spain

ScienceCloud New York City, June 2013

Agenda

1. Motivation
2. Goals and common problems
3. Common storage I/O virtualization solutions
4. VIDAS Design and Implementation
5. Evaluation
6. Conclusion

Motivation

- HPC in virtualized clouds gaining popularity
- Increasing amount of data stresses importance of I/O performance
- I/O overhead in virtualized environments is still significant
- Agreement that POSIX not suitable for high performance
- Virtualized environments trade off protection for performance
- Lack of I/O coordination mechanisms across domains

Goals and contributions

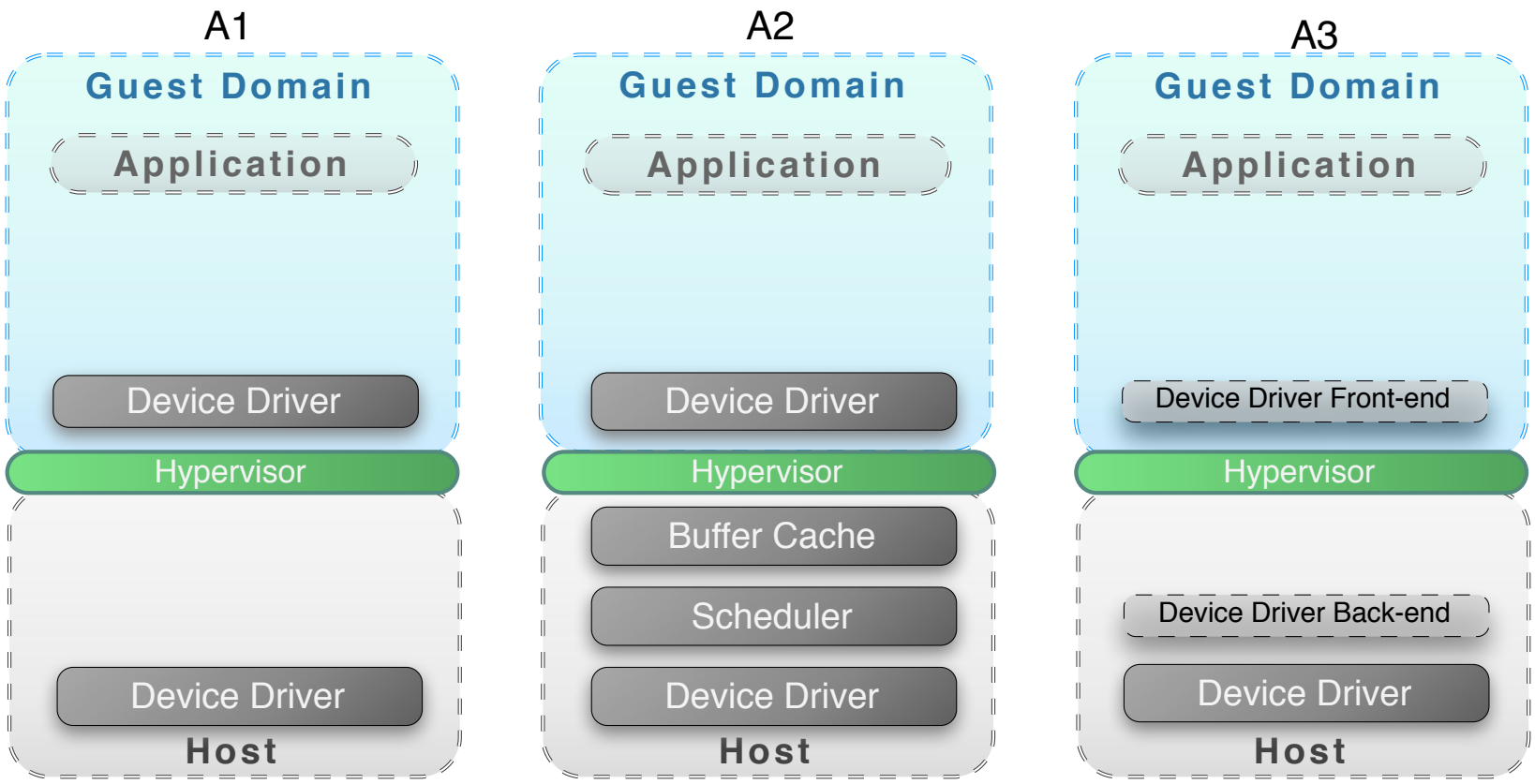
Propose new node-level abstractions and mechanisms in virtualized environments which:

- Enable building efficient virtualized data sharing
- Coordinate I/O across domains
- Provide shared access spaces across node-local domains
- Relax POSIX consistency
- Allow flexible data write and data read policies
- Expose data locality

Agenda

1. Motivation
2. Goals and common problems
3. Common storage I/O virtualization solutions
   3.1. Block-level solution
   3.2. Filesystem-level solution
4. VIDAS Design and Implementation
5. Evaluation
6. Conclusion

Virtualized block device drivers

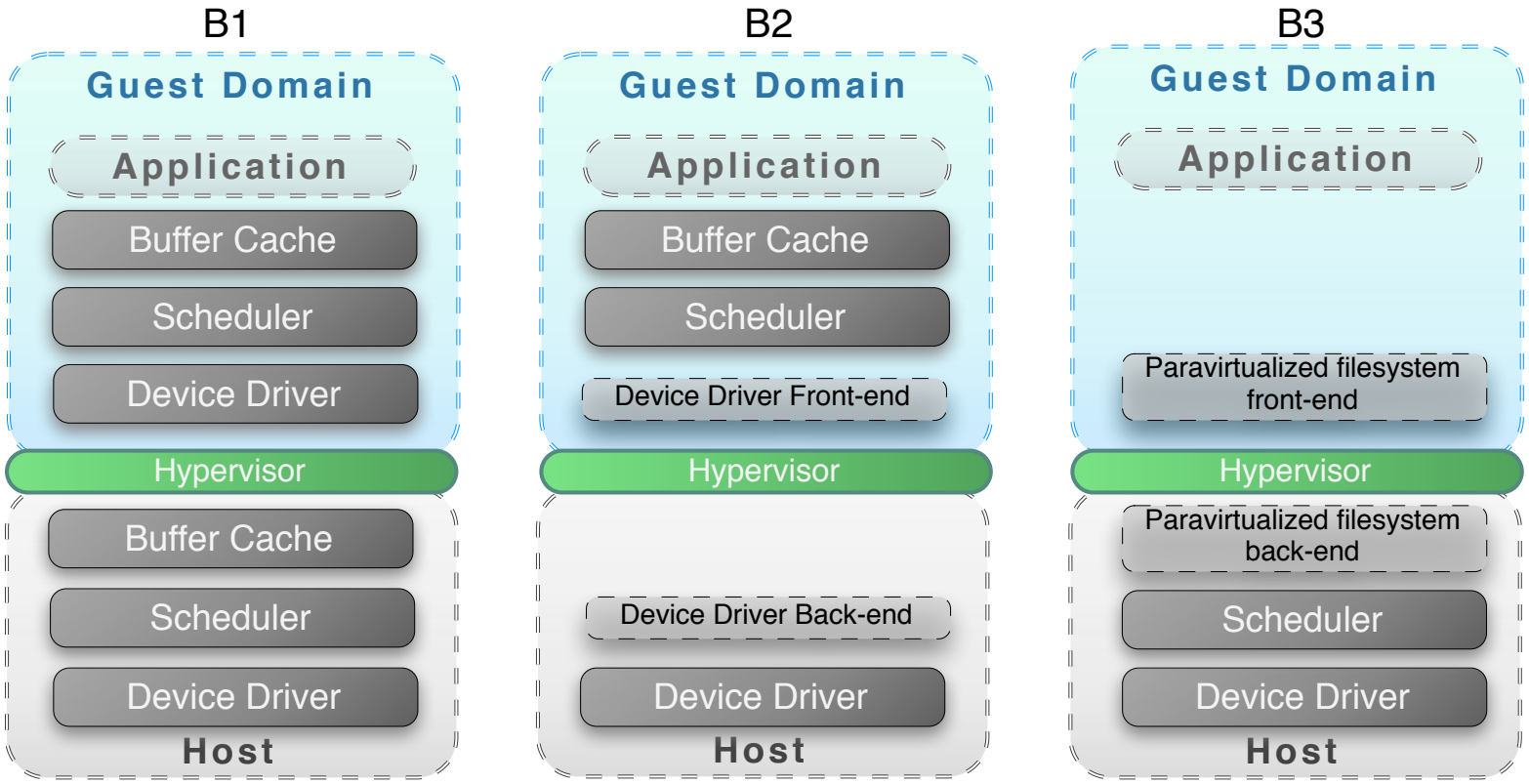


A1 virtualized block-level device on top of a physical device driver

A2 virtualized block-level device on top of filesystem

A3 paravirtualized device drivers

Virtualized filesystems

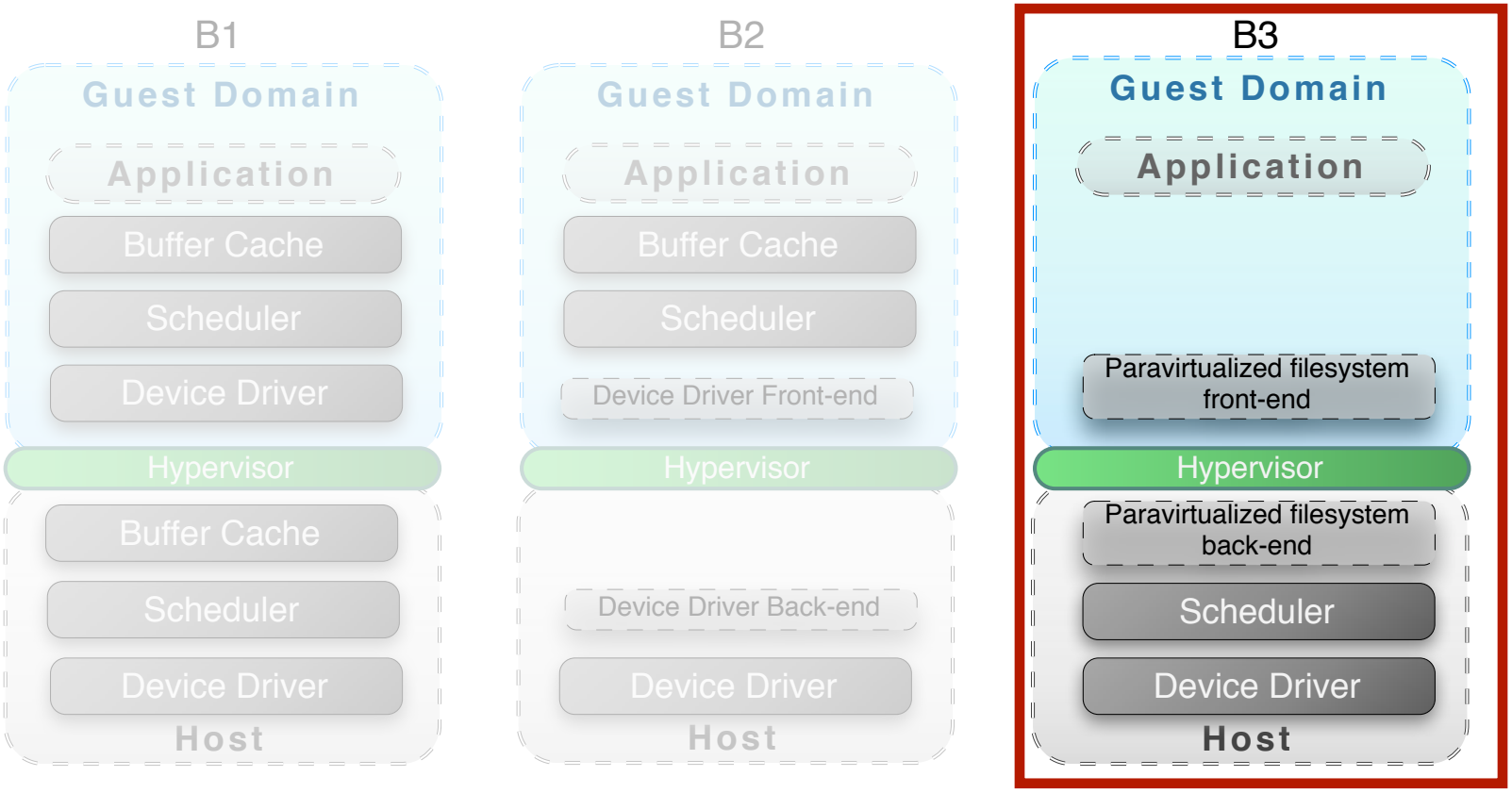


B1 virtualized filesystem within a file (stacked filesystems)

B2 paravirtualized filesystem

B3 sophisticated paravirtualized filesystem, inter-domain coordination

Virtualized filesystems



B1 virtualized filesystem within a file (stacked filesystems)

B2 paravirtualized filesystem

B3 sophisticated paravirtualized filesystem, inter-domain coordination

Agenda

1. Motivation

2. Goals and common problems

3. Common storage I/O virtualization solutions

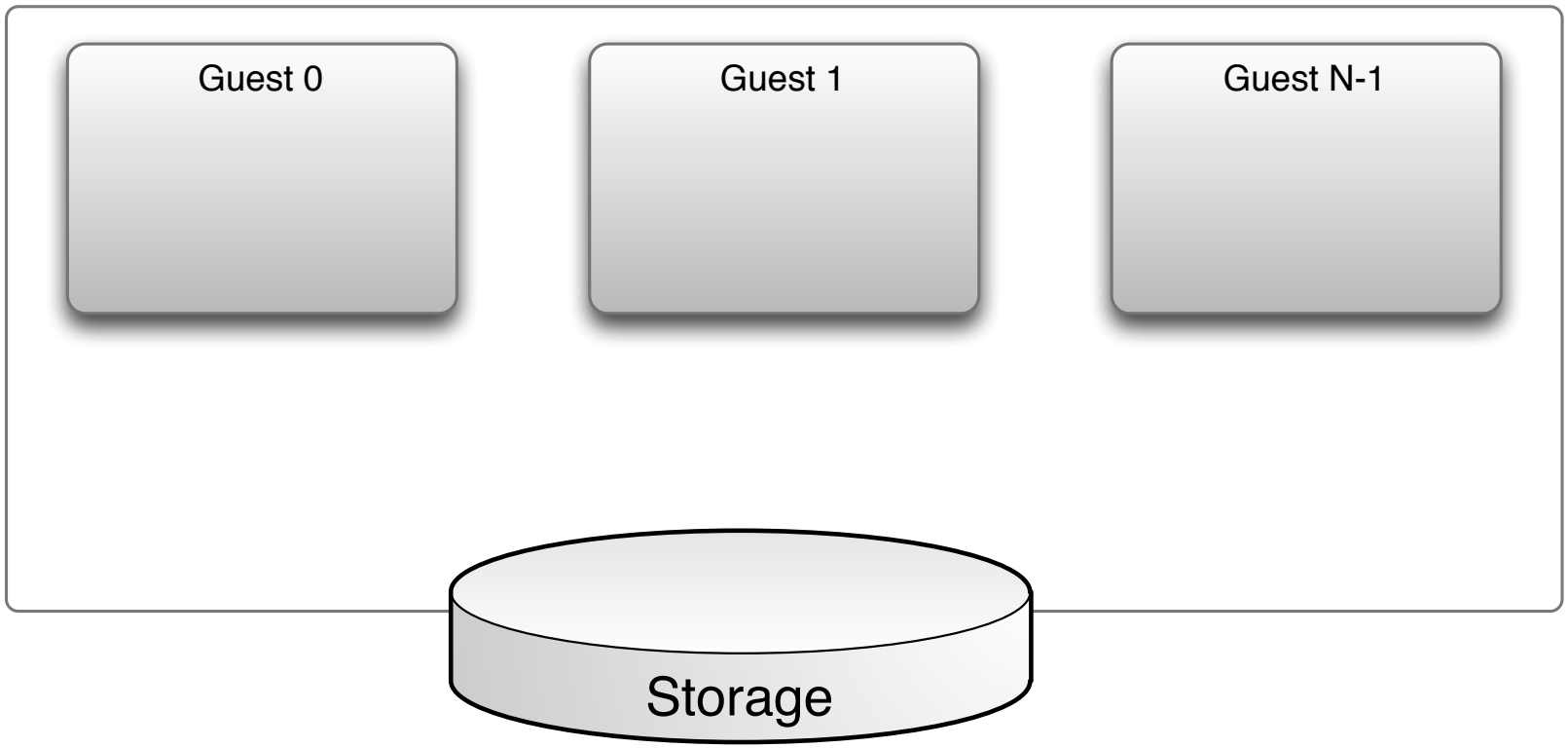
4. VIDAS Design and Implementation

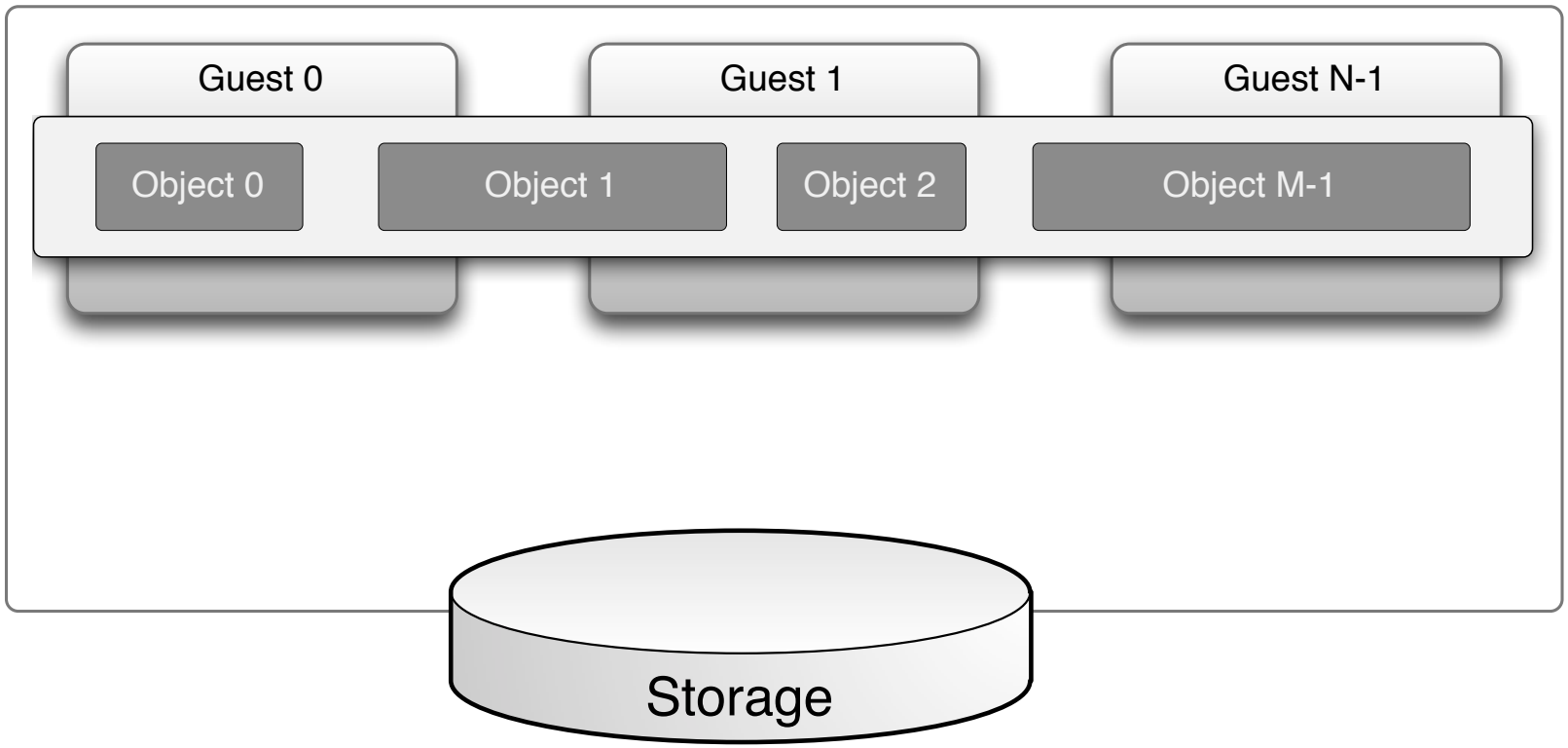
   4.1. Abstractions

   4.2. Design

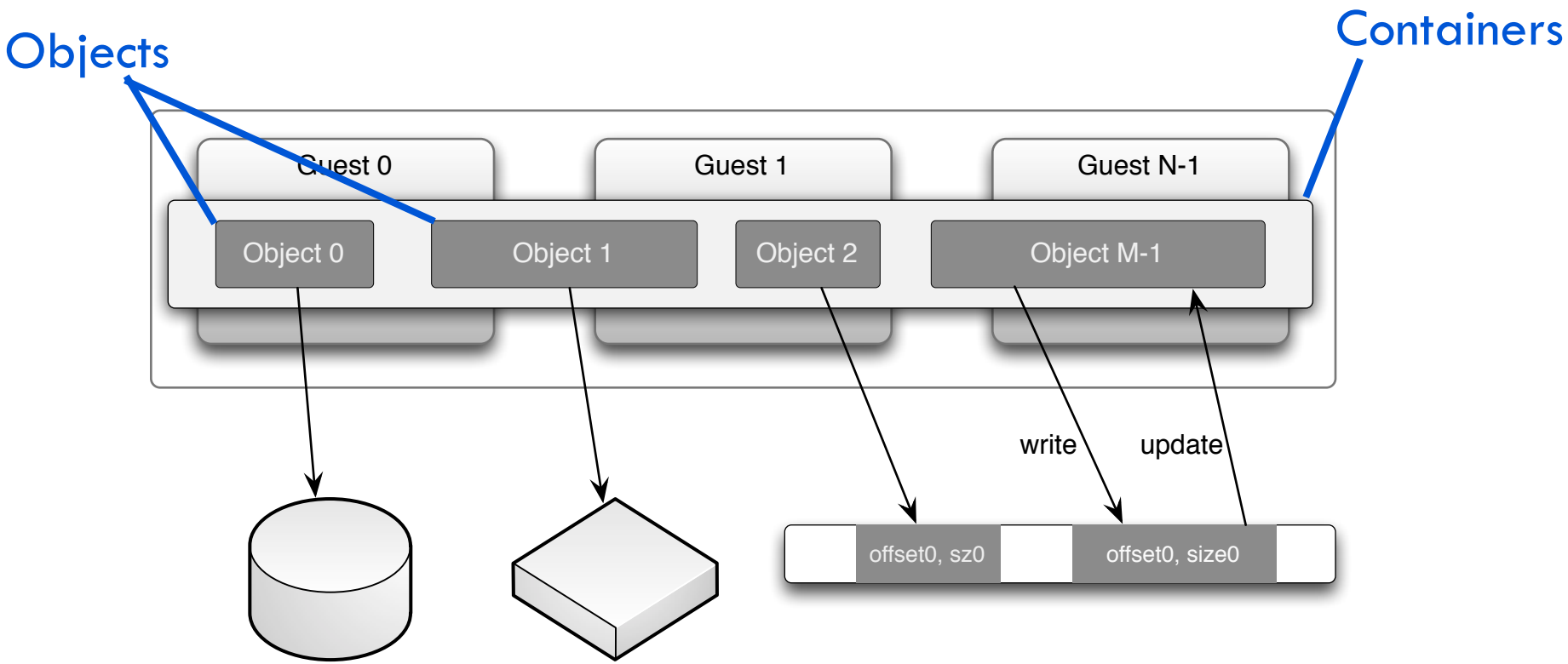
   4.3. Implementation

5. Evaluation


Guest 0
Guest 1
Guest N-1
Storage

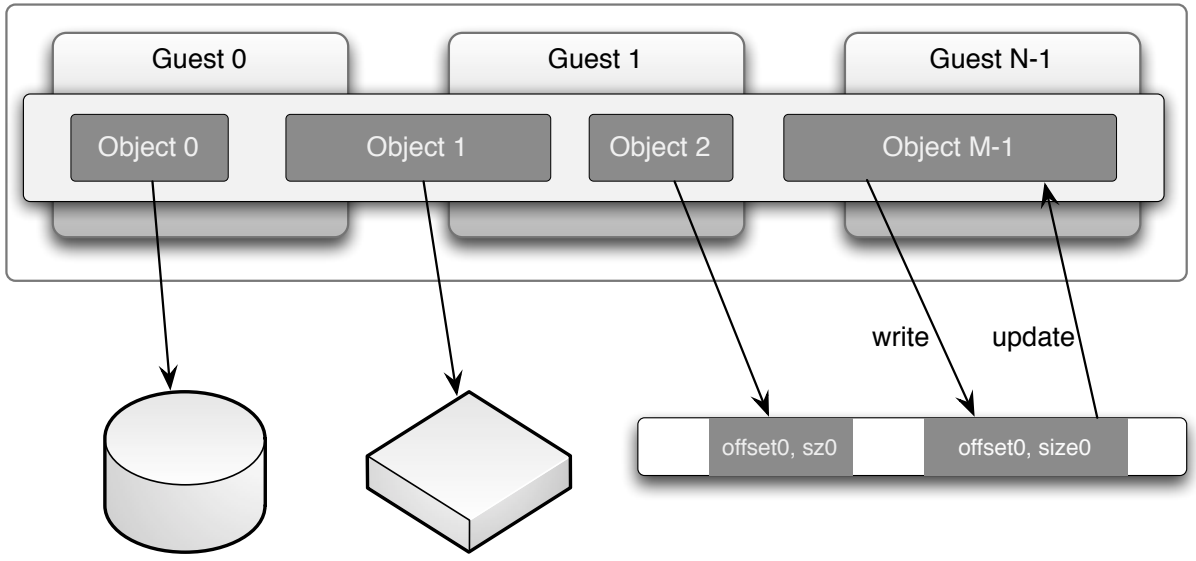

Guest 0
Guest 1
Guest N-1
Object 0
Object 1
Object 2
Object M-1
Storage

Abstractions



Containers

Containers are **access control domains** which allow to share objects among a set of guests.



Universidad Carlos III de Madrid
www.uc3m.es

Objects: POSIX differences

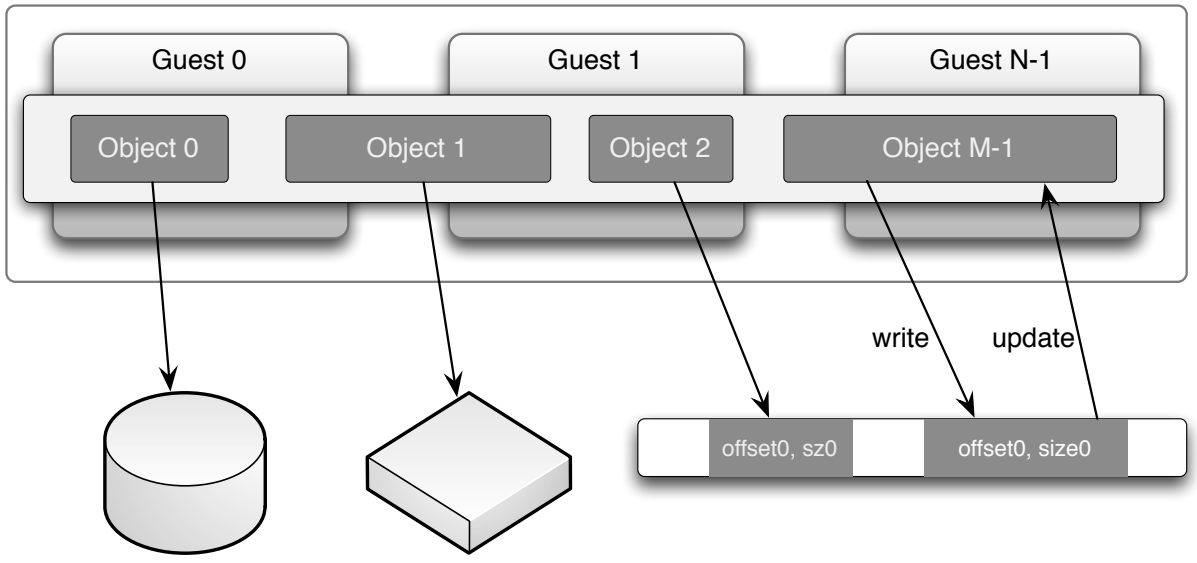
ARCOS

Strong **consistency not enforced**, but optional.

Data writes and updates guided by configurable policy: **write-through** or **write-back**

Provide **locality awareness**

Objects are uniquely associated to an external storage resource through its name



API Overview

Container operations

int container create(char *name, int domain ids[])

int container destroy(char* name)
int container attach(char *name)
int container leave(char* name)

Object data operations

int obj write(obj handle t o, char *buf, size t o↲set, size t sz)

int obj read(obj handle t o, char *buf, size t o↲set, size t sz)

int obj flush(obj handle t o)
int obj update(obj handle t o)

Object metadata operations

obj handle t object create(char* ext storage rsc, size t o↲set, size t size, char* cname)

obj handle t object join(char* ext storage rsc, size t o↲set, size t size, char* cname)

int object get locality(char* ext storage rsc, obj handle t *objects[])

int object leave(obj handle t o)
int object destroy(obj handle t o)
int object getattr(obj handle t o, char *name,

void *value, size t size)
int object setattr(obj handle t o, char *name,

void *value, size t size)

Object synchronization operations

int object wait(obj handle t o, char **bufp)

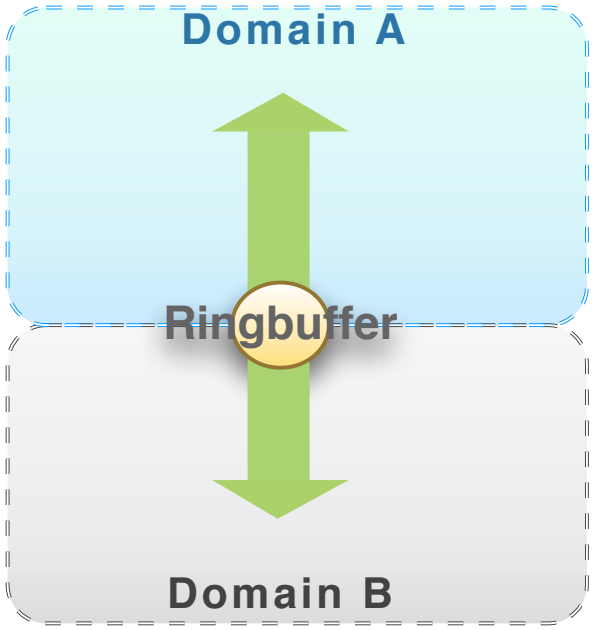
int object notify(obj handle t o, char *buf)

Agenda

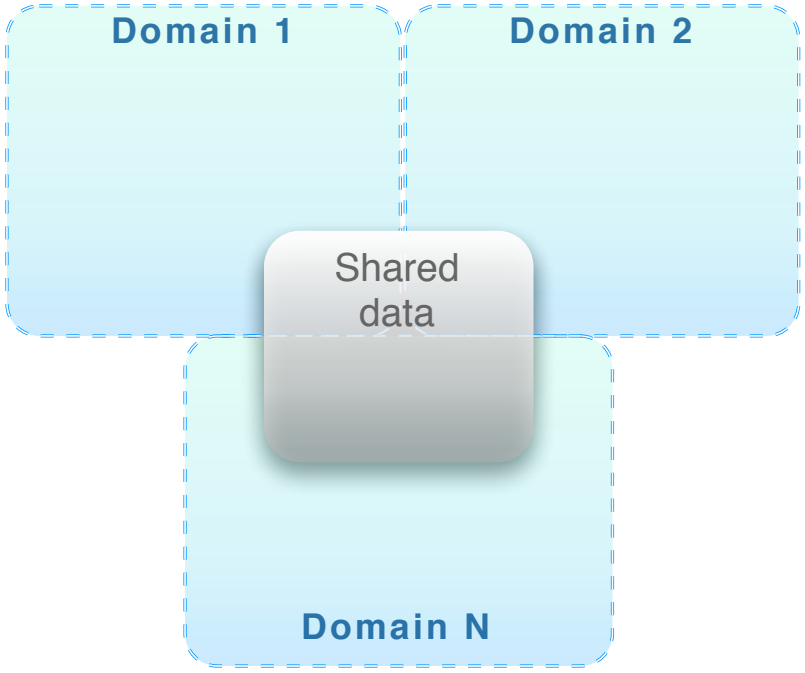
1. Motivation
2. Goals and common problems
3. Common storage I/O virtualization solutions
4. VIDAS Design
5. VIDAS Implementation
   5.1. Inter-domain communication mechanisms
   5.2. Implementation
6. Using VIDAS
7. Evaluation

Inter-domain data communication techniques

Ring buffers



Shared memory

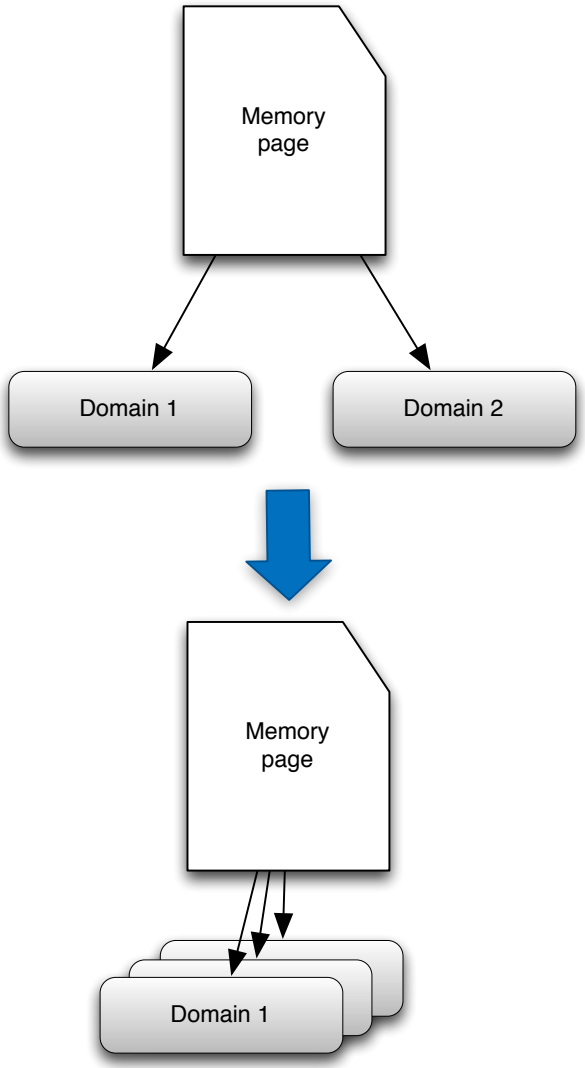


Inter-domain communication

- A high-efficient inter-domain data transferring system for virtual machines. (2009) pp. 385–390 Presented at the Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, New York, NY, USA: ACM.
- A study of Inter-domain Communication Mechanisms on Xen-based Hosting Platforms. (2009).
- Diakthaté F., Perache, M., Namyst, R., & Jourdren, H. (2009). *Efficient Shared Memory Message Passing for Inter-VM Communications*. *Euro-Par 2008 Workshop*
- *Huang, W.*, Koop, M. J., Gao, Q., & Panda, D. K. (2007). Virtual machine aware communication libraries for high performance computing (p. 9). Presented at the SC '07
- Inter-domain socket communications supporting high performance and full binary compatibility on Xen (pp. 11–20). Presented at the Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, New York, NY, USA: ACM
- Wei Huang, Koop, M. J., & Panda, D. K. (2008). Efficient one-copy MPI shared memory communication in Virtual Machines (pp. 107–115). Presented at the 2008 IEEE International Conference on Cluster Computing (CLUSTER), IEEE.

All of these optimize inter-vm data sharing and communication, but none of them work around the limitation of being able to share pages with only 2 domains.

Xen/Linux grant reference extension



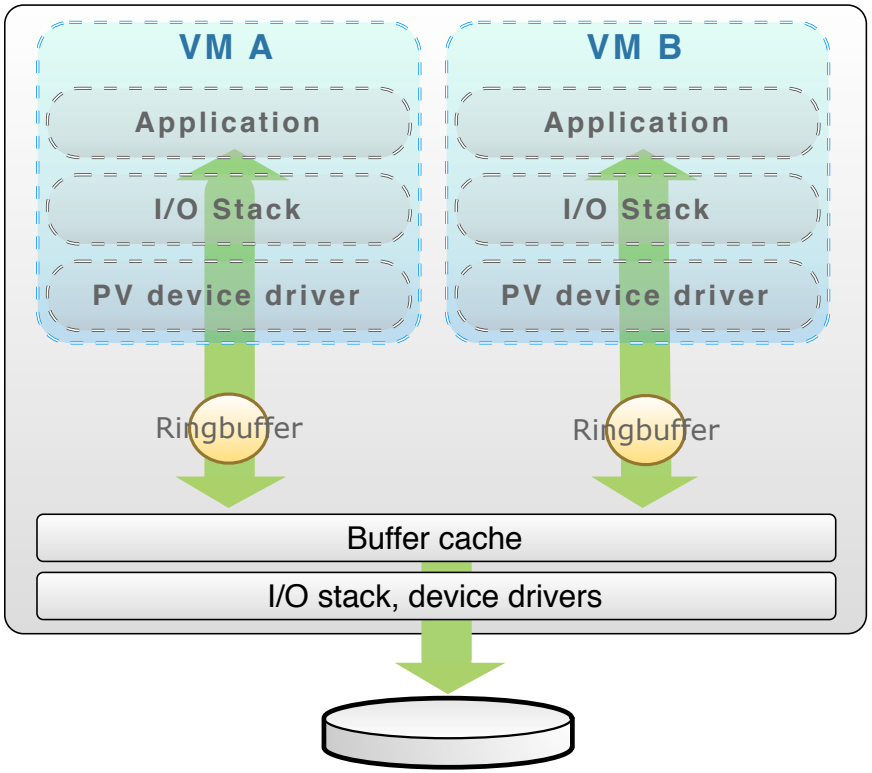
Hypervisors usually provide a mechanism to share a page with only 2 domains domains at once.

In our work we extended Linux/Xen to provide user-space applications with the ability to share a page with any number of domains.

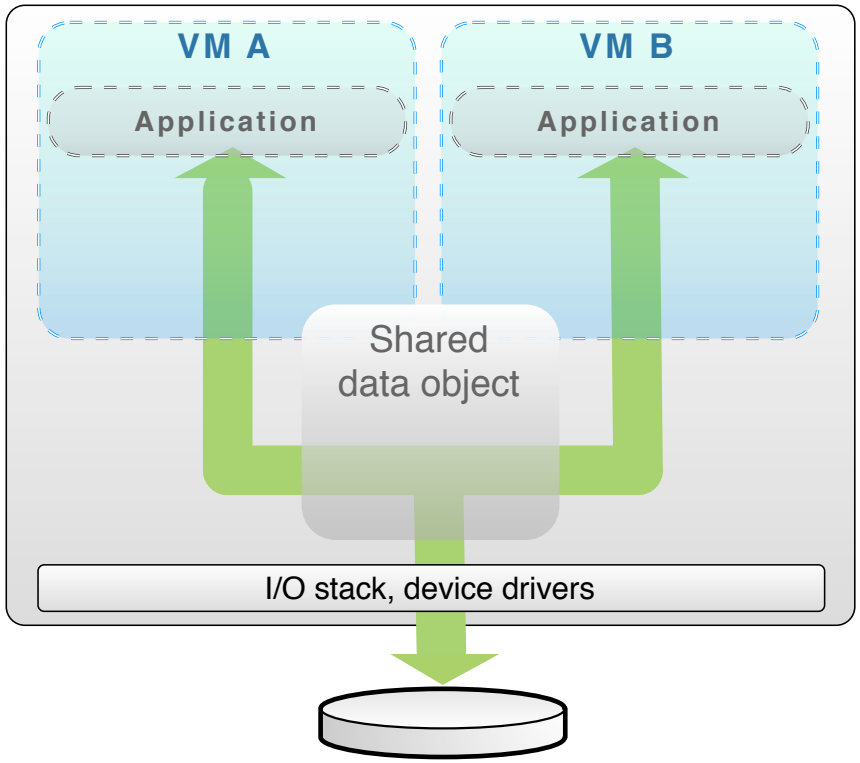
Paravirtualized data sharing

While most solutions use mainly ringbuffers for paravirtualization, VIDAS combines ringbuffers and multi-domain shared memory

Point-to-point



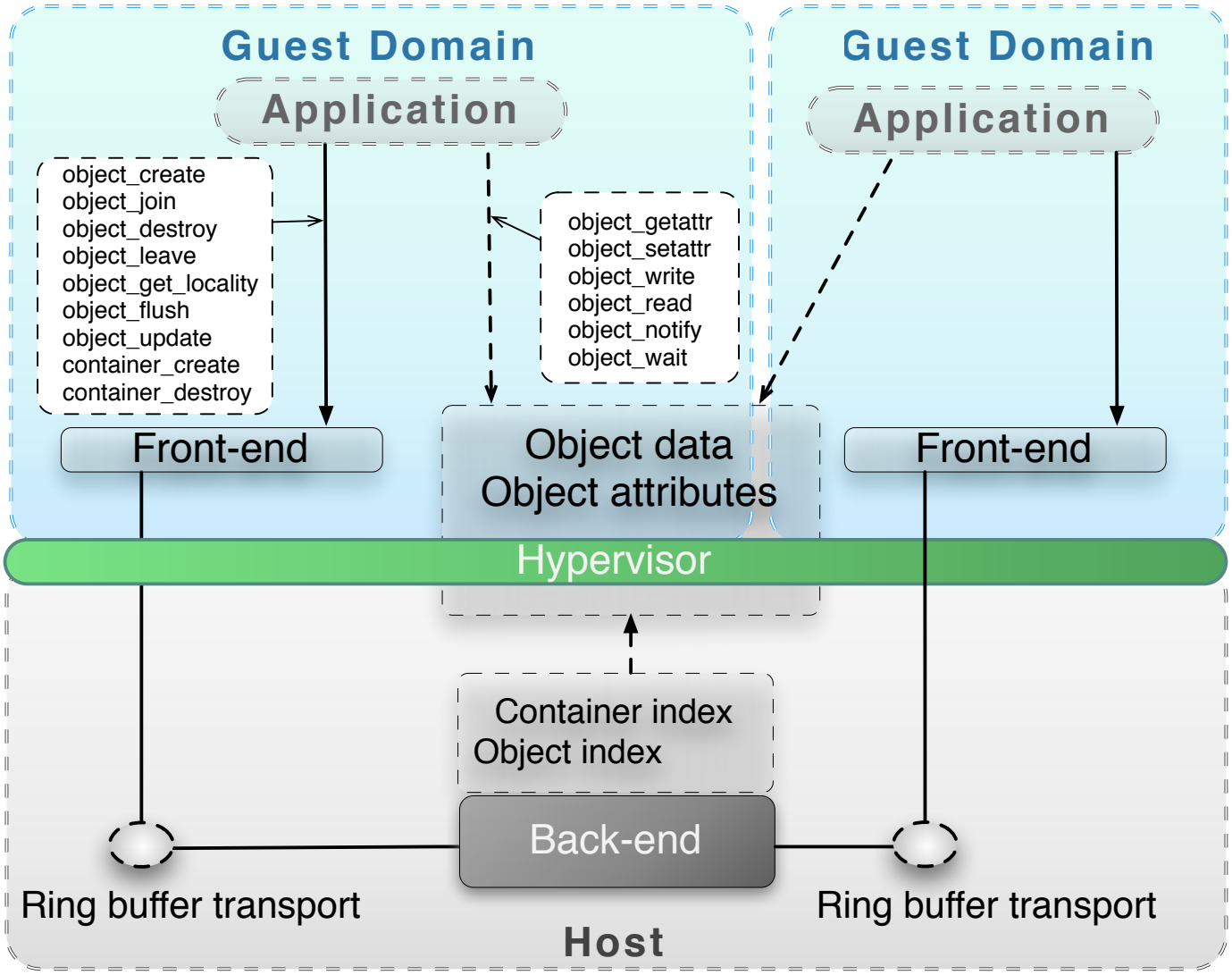
Collective



Universidad Carlos III de Madrid
www.uc3m.es

Implementation

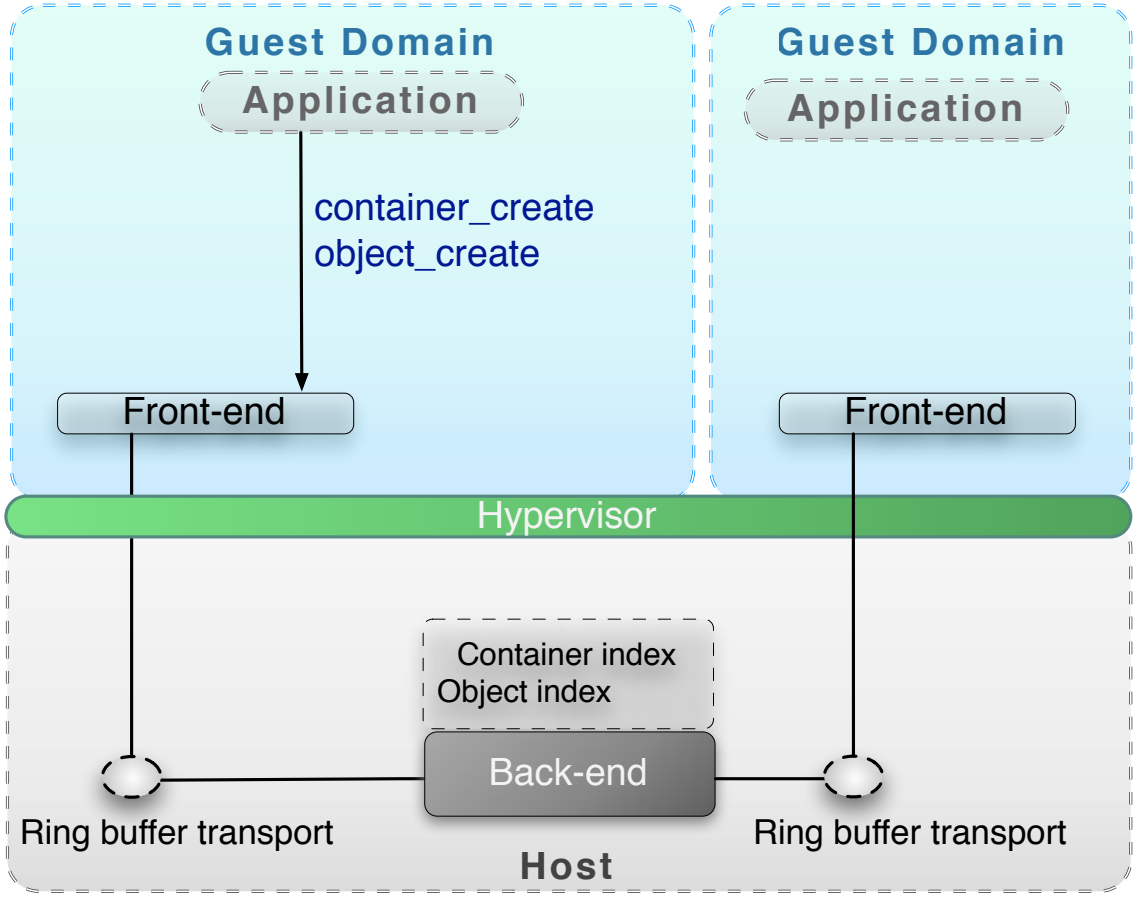
ARCOS



Universidad Carlos III de Madrid
www.uc3m.es

Implementation

ARCOS

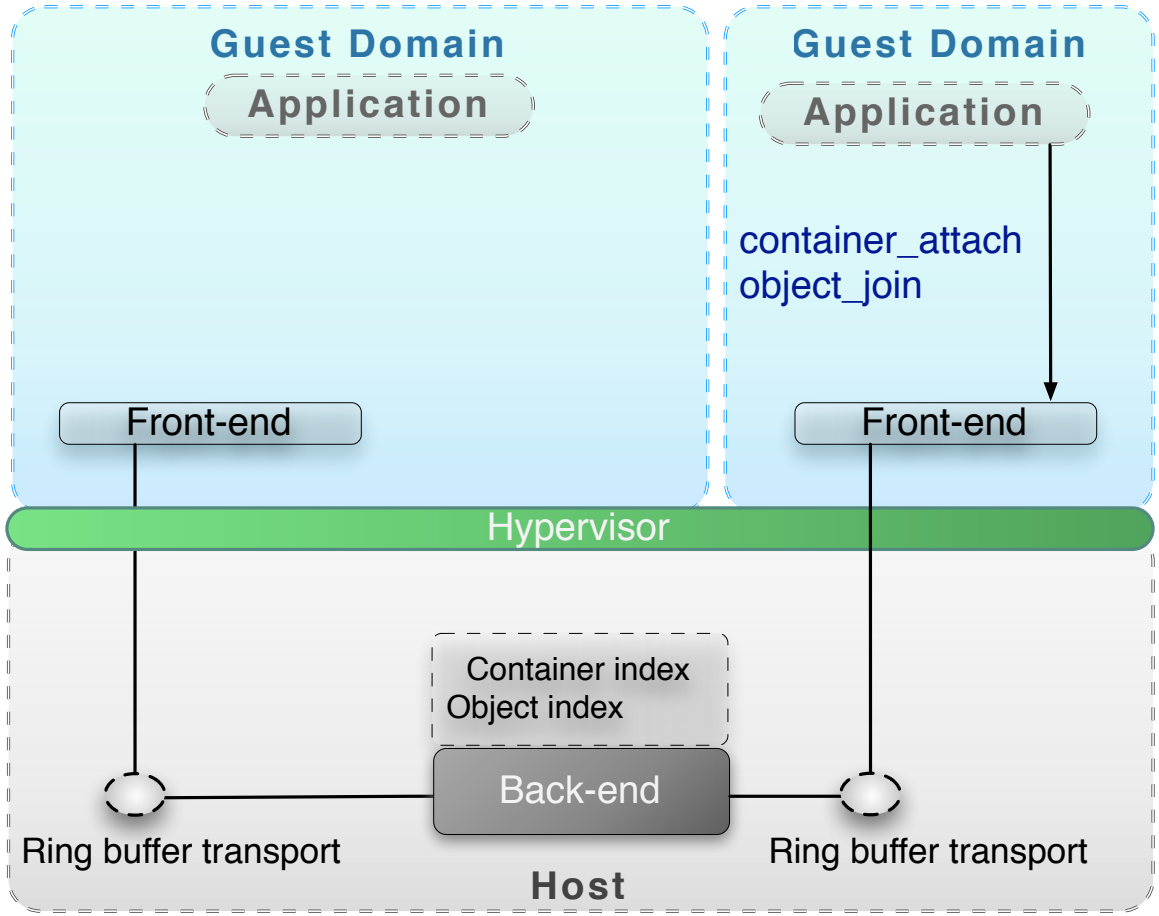


A container creator specifies which domains are allowed to access the objects within the container

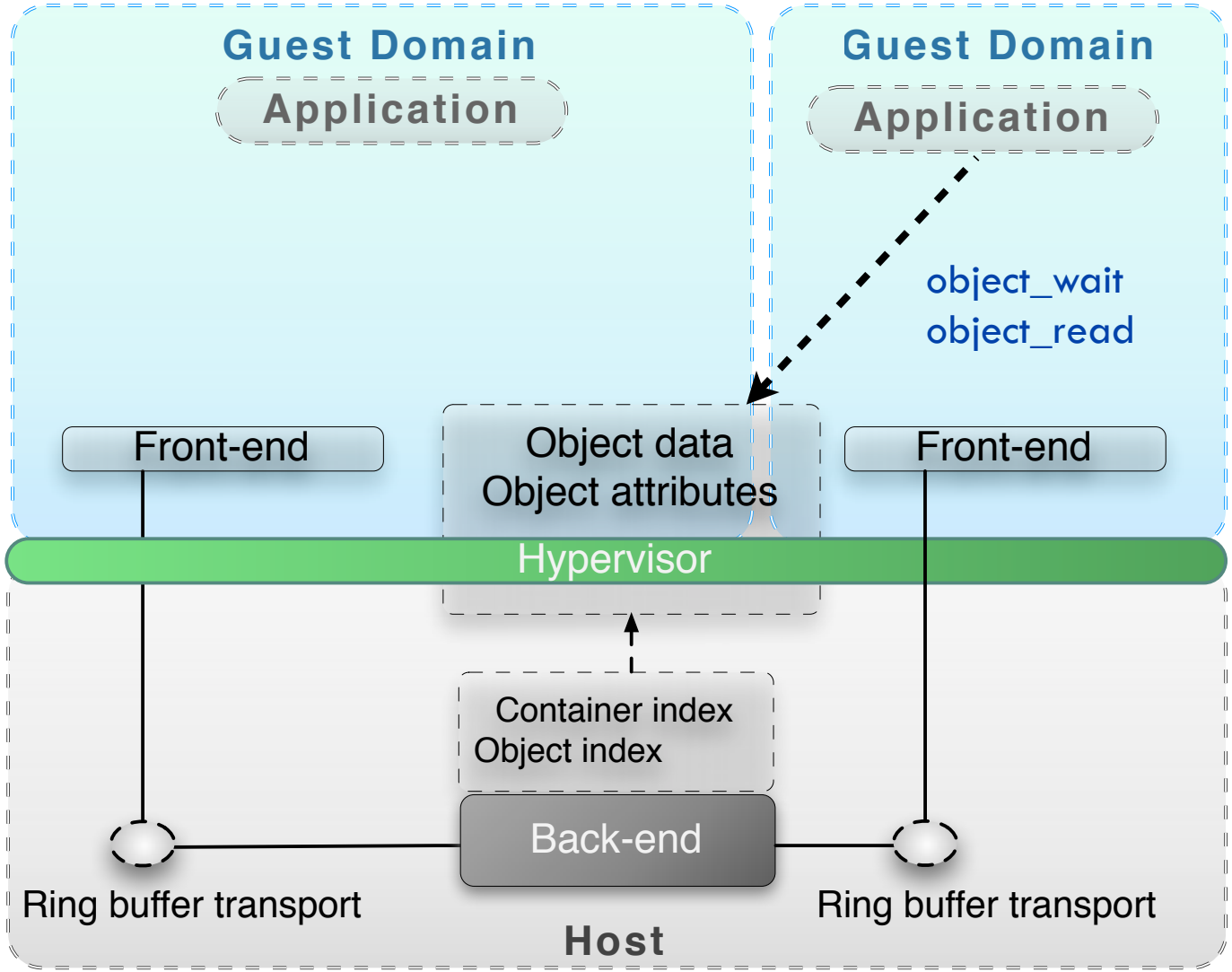
Universidad Carlos III de Madrid
www.uc3m.es

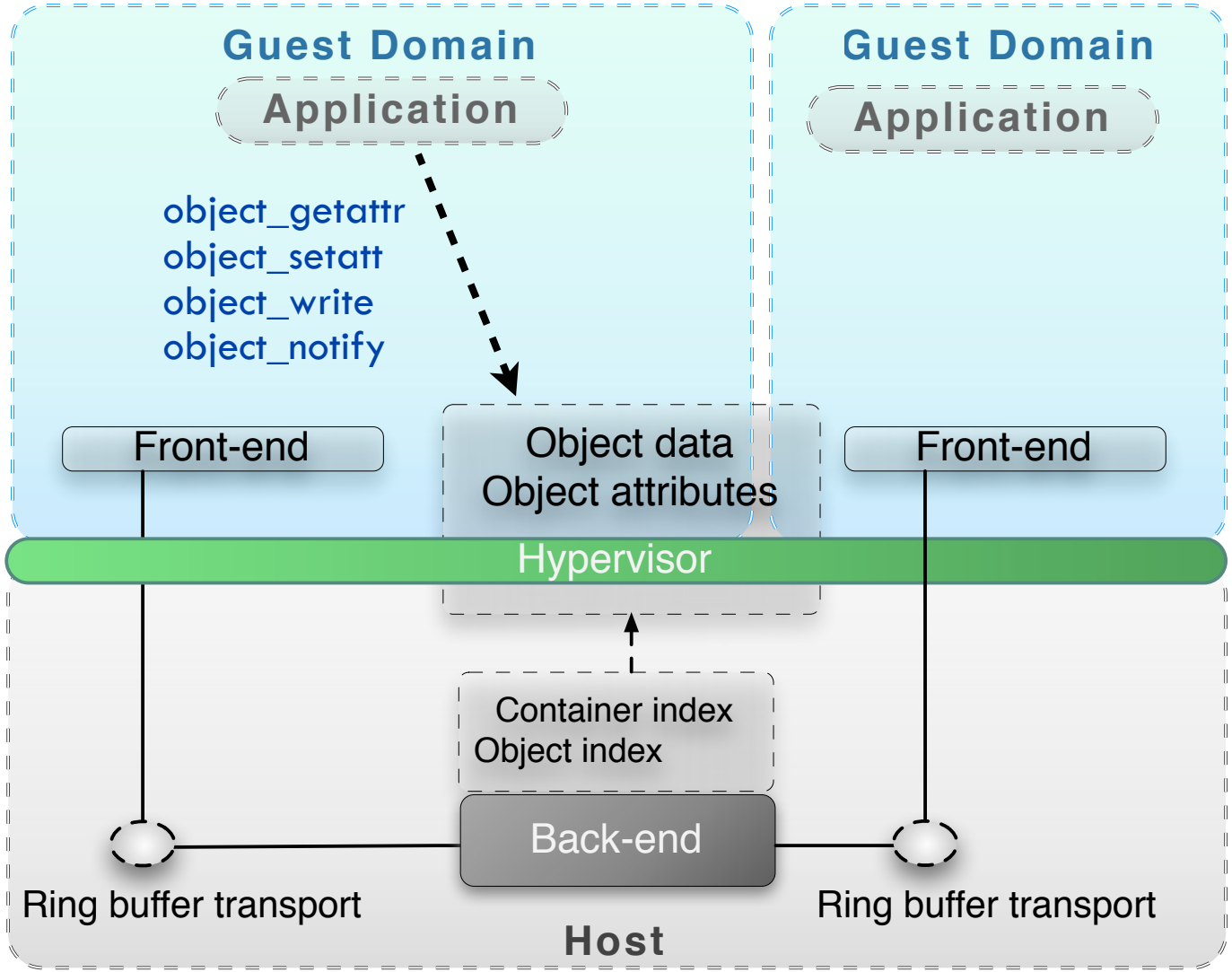
Implementation

ARCOS



Another domain can then access the objects in a container by joining in


Guest Domain
Application
Guest Domain
Application
object_wait
object_read
Front-end
Object data
Object attributes
Front-end
Hypervisor
Container index
Object index
Back-end
Ring buffer transport
Ring buffer transport
Host


Guest Domain
Application
Guest Domain
Application
object_getattr
object_setatt
object_write
object_notify
Front-end
Object data
Object attributes
Front-end
Hypervisor
Container index
Object index
Back-end
Ring buffer transport
Ring buffer transport
Host

Agenda

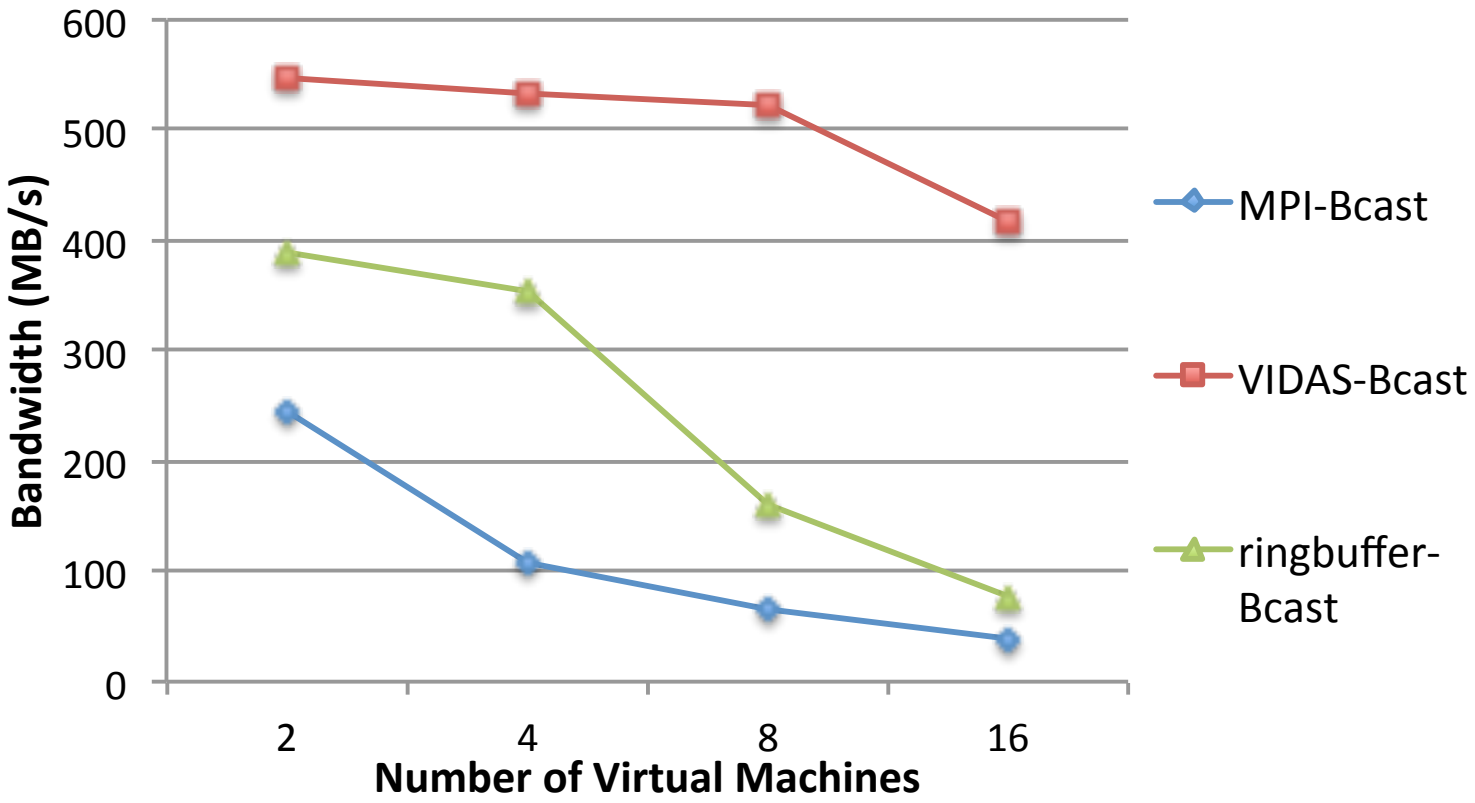
1. Motivation
2. Goals and common problems
3. Common storage I/O virtualization solutions
4. VIDAS Design and Implementation
5. Evaluation
6. Conclusion

Evaluation

- CPU Intel Xeon 12 core
- RAM 64GB DDR3 synchronous RAM @ 1333Mhz
- HDD 1TB Toshiba MK1002TS
  - Linux LVM
  - ext4
- Hypervisor Xen 4.2
- Privileged domain Linux 3.5.7
- Guest domains Linux 3.5.7

- Synthetic Benchmarks: Broadcast, Independent read, Collective r/w

Bandwidth: In-memory communication

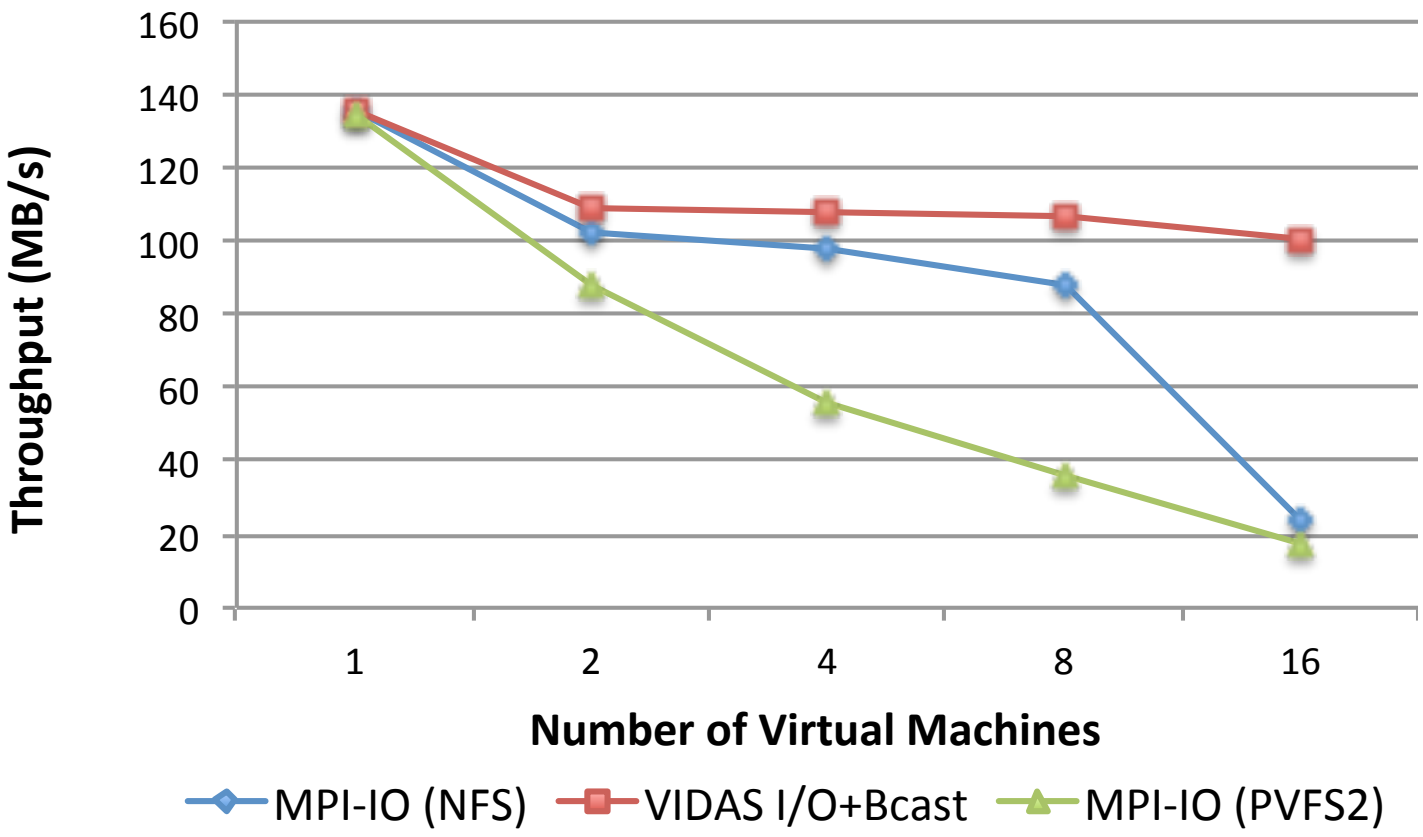


MPI-Bcast: share 128MB of data using MPI_Bcast

VIDAS-Bcast: share 128MB of data using VIDAS

ringbuffer-bcast: transfer 128MB of data to each VM using Xen ringbuffer transports

Independent file read (512MB)

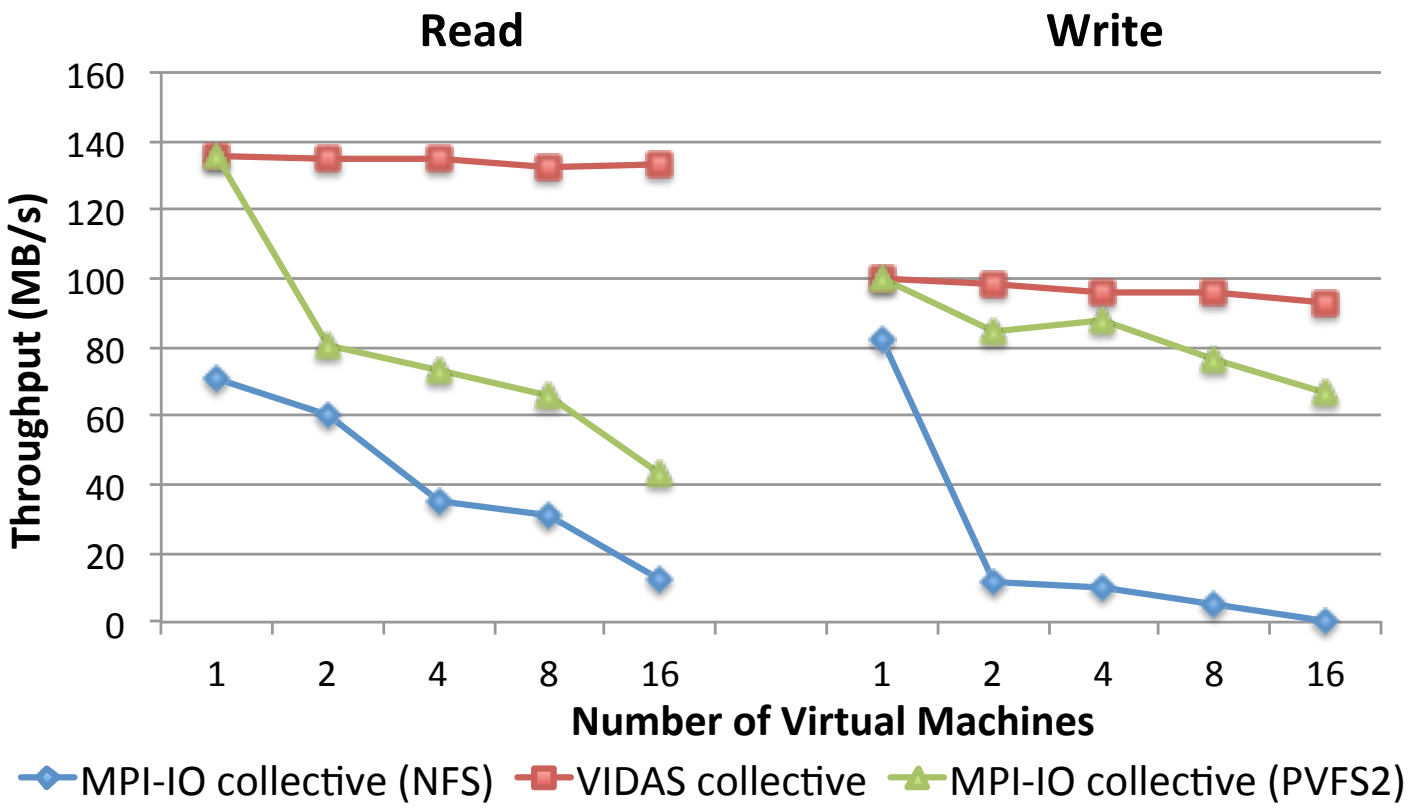


MPI-IO (NFS) Concurrent file read on a node-local NFS mount

MPI-IO (PVFS2) Concurrent file read on a node-local PVFS2 mount

VIDAS I/O+Bcast Load data into object, broadcast to all VMs

Collective I/O (Two-phase I/O)



Non-overlappingly interleaved strided vectors of 2MB blocks

Agenda

1. Motivation
2. Goals and common problems
3. Common storage I/O virtualization solutions
4. VIDAS Design and Implementation
5. Evaluation
6. Conclusion

Conclusion and Contributions

- I/O overhead in virtualized environments is still significant
  - Reduce memory copy operations
  - Reduce domain context switches
- Agreement that POSIX not suitable for high performance
  - Relax POSIX consistency
  - Control data write and update policies
  - Control data locality
- Virtualized environments trade off protection for performance
  - Created shared access spaces
  - Reduced memory copy operations and context switches
- Lack of I/O coordination mechanisms across domains
  - Coordinate storage I/O across domains
  - Proposed new multi-domain data sharing mechanisms

Questions?

Q?