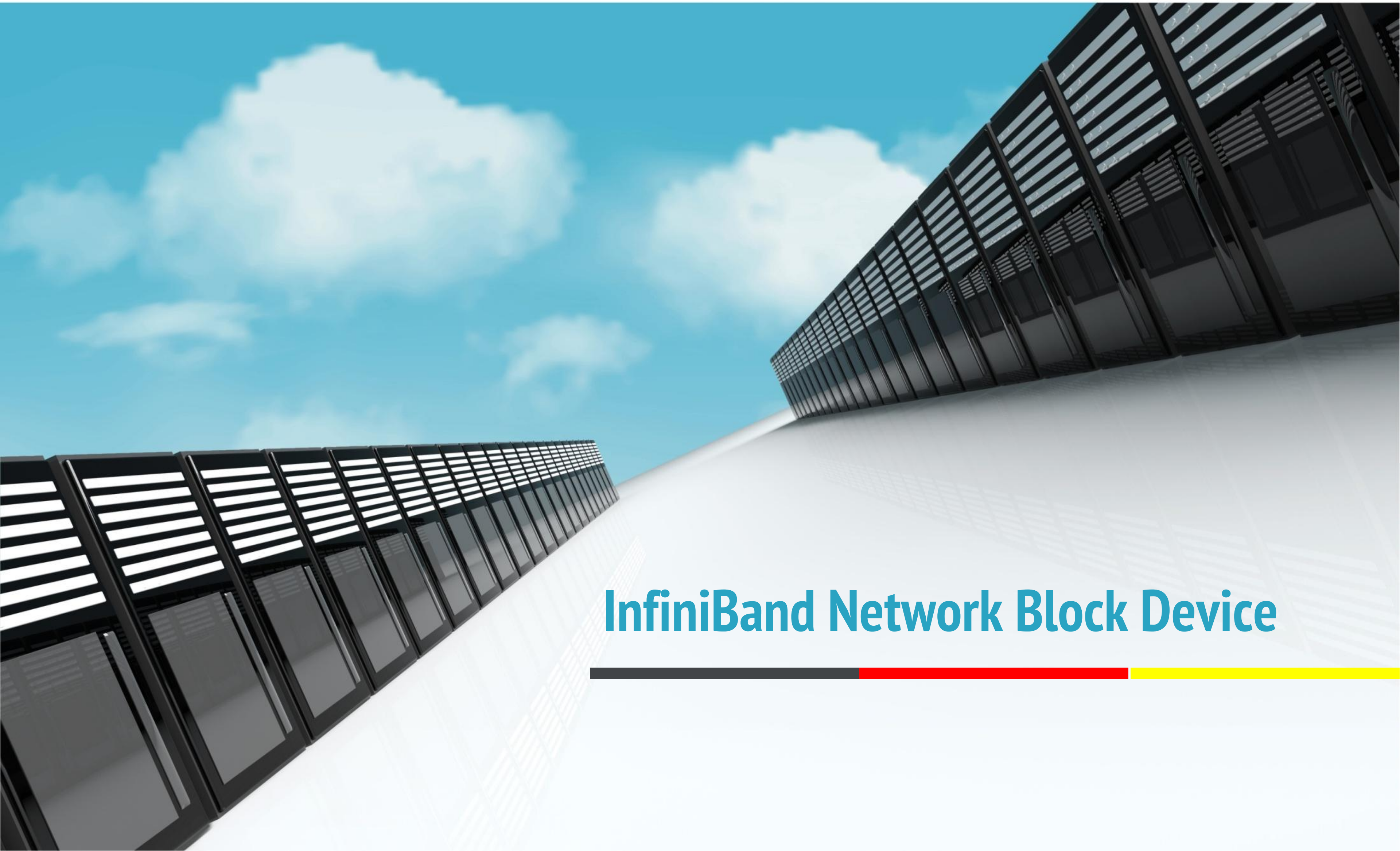
PROFITBRICKS
The IaaS-Company.



InfiniBand Network Block Device

Overview

- IBNBD: InfiniBand Network Block device
- Transfer block IO using InfiniBand RDMA
- Map a remote block device and access it locally
- Client side
    - registers as a block device, i.e. `/dev/ibnbd0`
    - transfers block requests to the remote side
- Server side
    - Receives RDMA buffers and convert them to BIOs
    - Submit BIOs down to the underlying block device
    - Send IO responses back to the client

Motivation

- ProfitBricks GmbH is an IaaS provider
- Our data centers:
  - compute nodes with customer VMs
  - storage servers with the HDDs/SSDs
  - InfiniBand network
- SRP/SCST for transfer of customer IOs from the VM on a compute node to the physical device on the storage server.
- Problems:
  - SCSI IO Timeouts
  - SCSI Aborts
  - Overhead of intermediate protocol

Goals

- Simplify operation
  - regular tasks (i.e. mapping / unmapping)
  - maintenance (i.e. server crash)
- Thin implementation
  - plain Block IO - no intermediate SCSI layer
  - better maintainability
  - integration into a software defined storage solution
- Performance
  - optimize for io latency

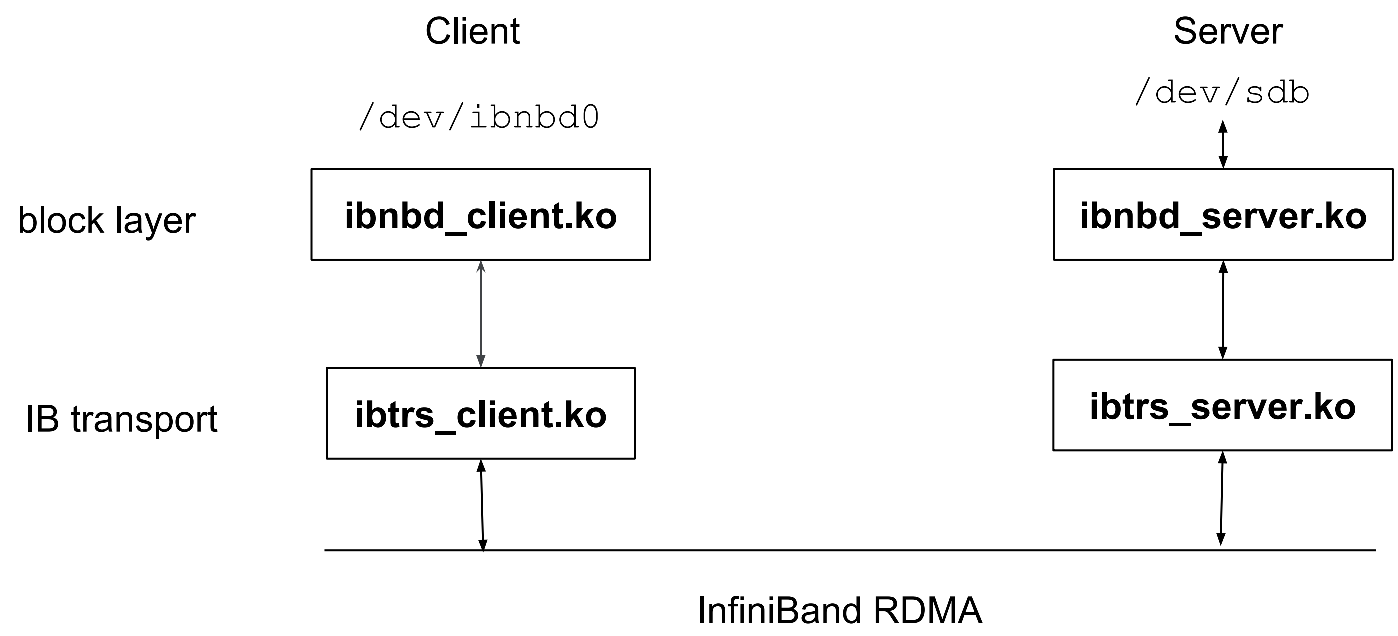
Design objective

- Eliminate SCSI as intermediate transport layer
- Rely on the IB service to reduce design complexity
  - Minimal error handling: take advantage of the RC mode of IB, which guarantees an RDMA operation to either succeed or fail.
  - simpler, robust and easier to maintain transport layer
  - No IO timeouts and retransmissions
- Minimize number of RDMA operations per IO to achieve lower latency
- Allow for an IO response to be processed on the CPU the IO was originally submitted on

Operation

- Mapping client side
  - Server address and device path on the server
  - `$echo "device=/dev/sdb server=gid:xxxx:xxx:xxxx" > /sys/kernel/ibnbd/map_device`
  - `/dev/ibnbd<x>` is created
- Export server side
  - no configuration is required
- Devices listed under /sys/kernel/ibnbd/devices/
- Session listed under /sys/kernel/ibtrs/sessions/
- Mapping options
  - Input mode (client side): Request or Multiqueue
  - IO mode (server side): block IO or file IO

Overall structure



- IBTRS (InfiniBand transport)
  - generic UAL for IB RDMA
  - can be reused by a different block device or any application utilizing request read/write RDMA semantics (i.e. replication solution)

Module functions

IBNBD is responsible for the delivery of block IO requests from client to storage server. Uses **IBTRS** as its IB rdma transport layer

- **Client** on compute node:
  - **ibnbd_client.ko** provides the mapped block devices (/dev/ibnbd<x>) and prepares IO for the transfer.
  - **ibtrs_client.ko** establishes connection to a server and executes rdma operations requested by ibnbd

- **Server** on storage side:
  - **ibtrs_server.ko** accepts connections from client, executes rdma transfers, hands over received data to ibnbd_server.
  - **ibnbd_server.ko** processes incoming IO requests and hands them over down to the underlying block device (i.e. an /dev/sdb device)

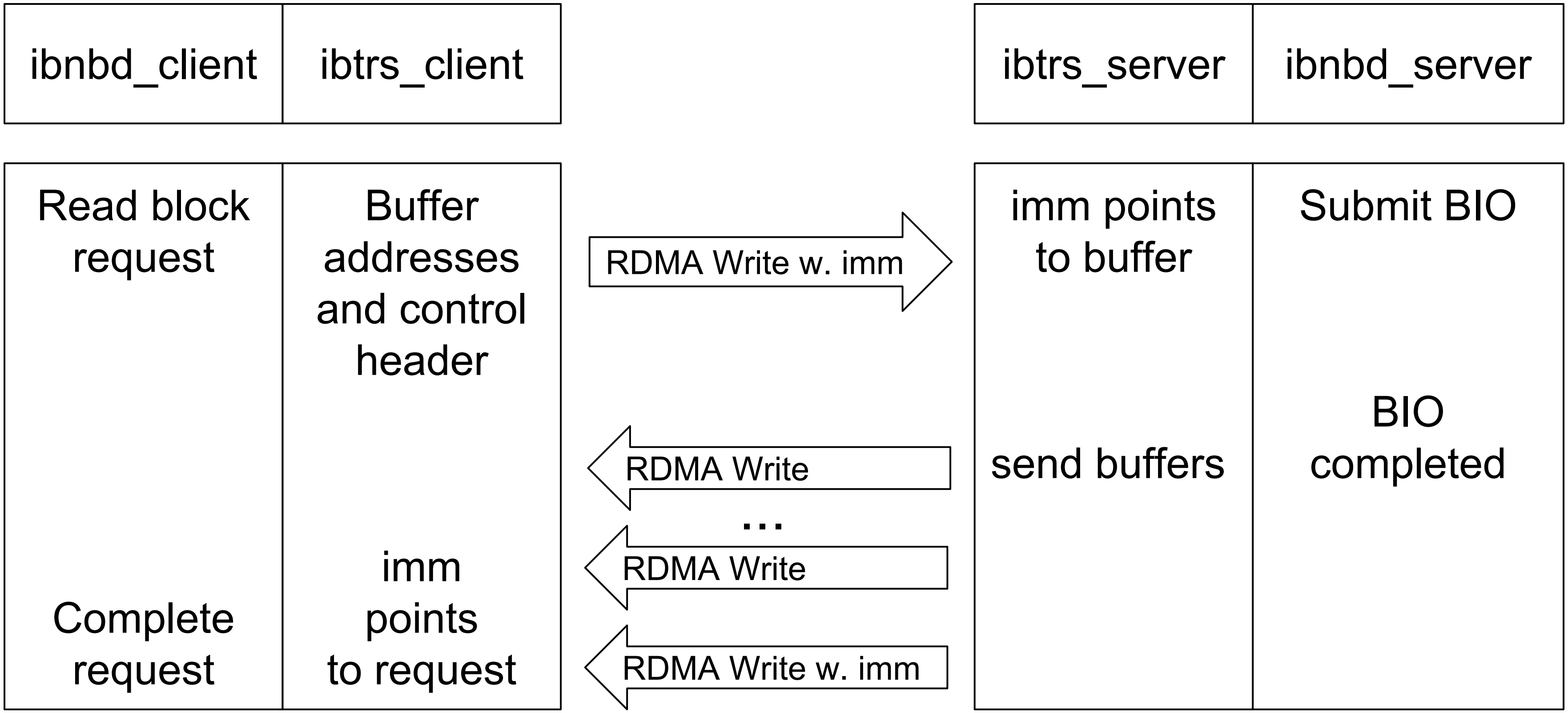
Memory management, immediate field

- Client-side server memory management
- Server reserves queue_depth chunks each max_io_size big
- Client is managing this memory
- Allows to reduce number of RDMA operations per IO
- Tradeoff between memory consumption vs. latency
- client uses 32 bit imm field to tell server where transferred data can be found
- server uses imm field to tell client which outstanding IO is completed

Transfer procedure

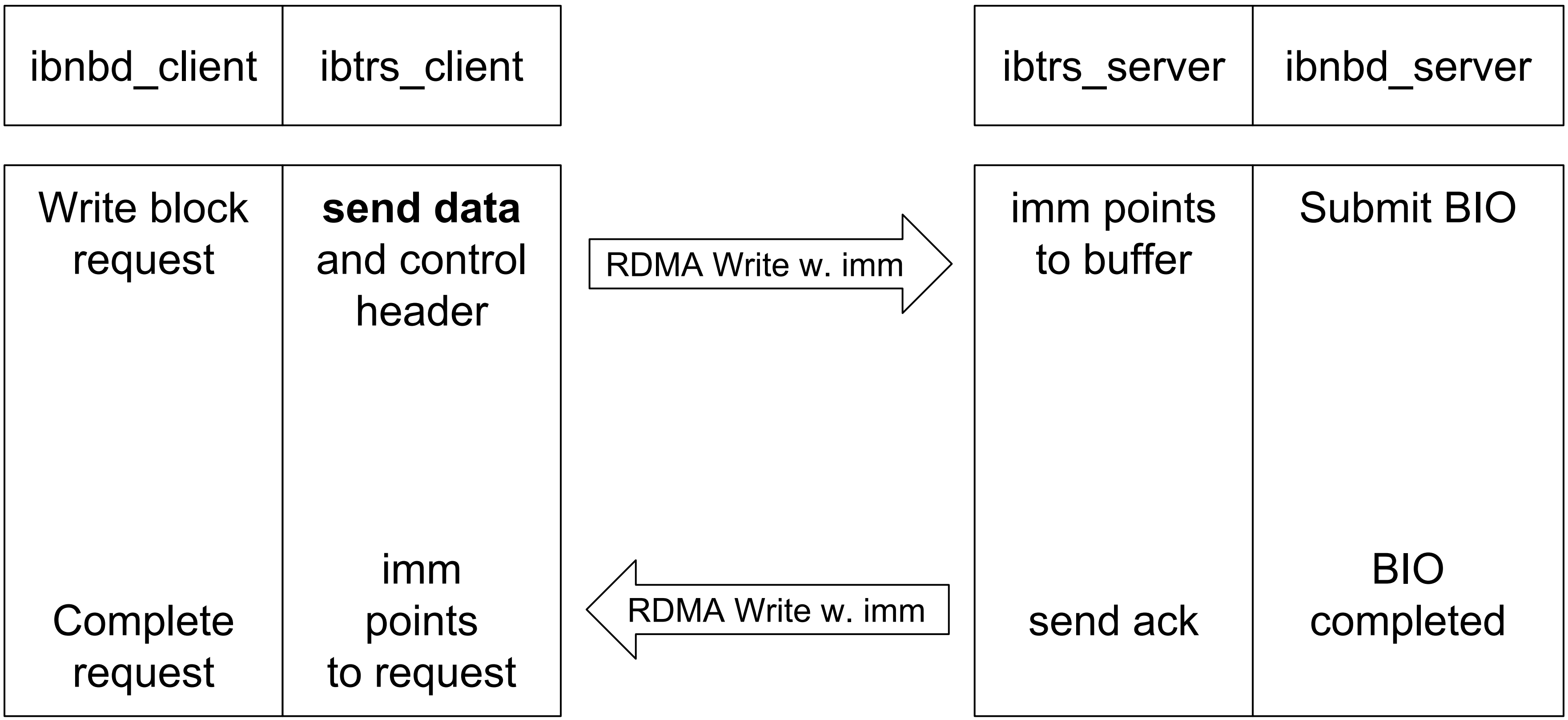
1. **ibnbd_client**
   - converts incoming block request into an sg list with a header
2. **ibtrs_client**
   - transfers data (write IO) or control (read IO) in a single rdma write
   - uses 32 bit imm field to tell the server where the data can be found
3. **ibtrs_server**
   - notifies ibnbd_server about an incoming IO request
4. **ibnbd_server**
   - generates BIO and submits it to underlying device
   - acknowledges the RDMA operation, when BIO comes back
5. **ibtrs_server** sends confirmation (write IO) or data (read IO) back to client
6. **ibtrs_client** notifies ibnbd_client about a completed RDMA operation
7. **ibnbd_client** completes the original block request

Transfer procedure: read



- Same procedure as used by iSER or SRP: server initiates transfer
- Fast memory registration feature is used to reduce number of transfers

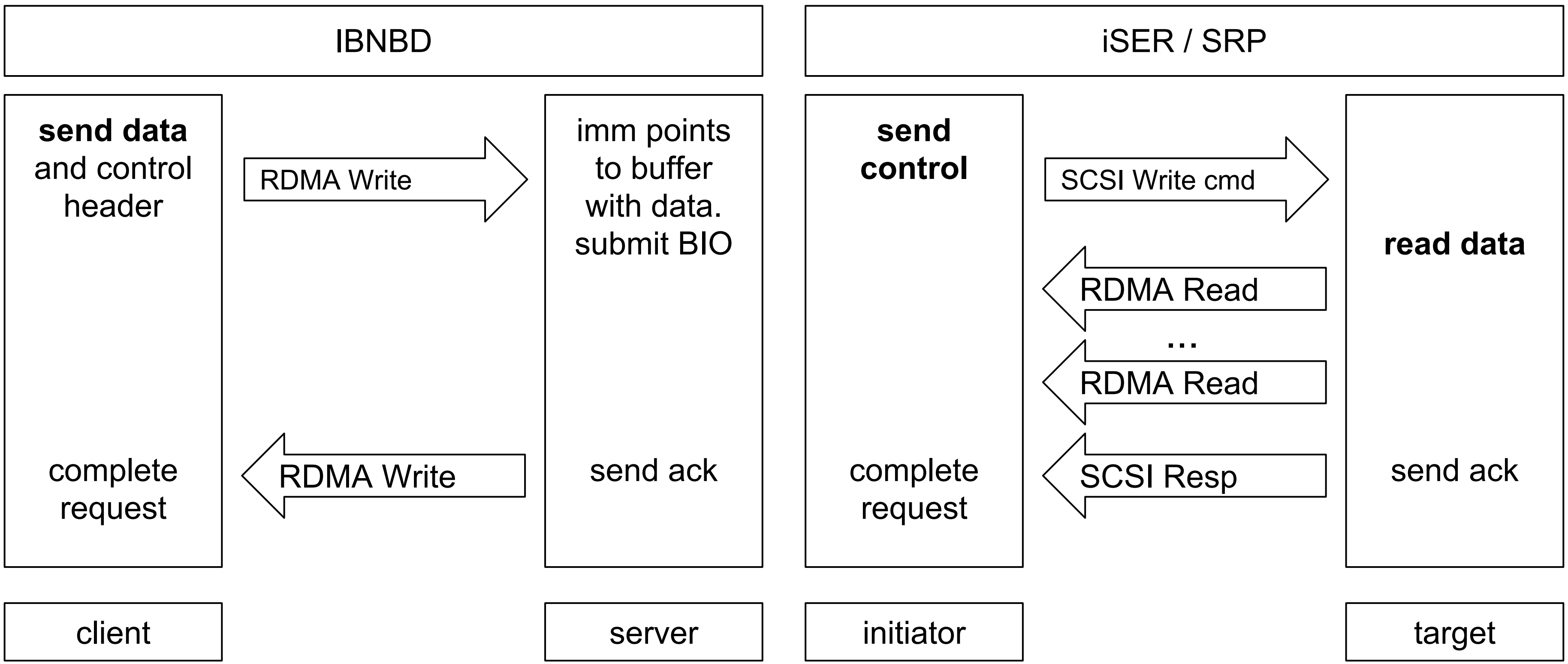
Transfer procedure: write



- Different to iSER or SRP: Client initiates the transfer into a server buffer
- Only two RDMA operations

PROFITBRICKS
The IaaS-Company.

Transfer procedure: write, IBNBD vs iSER/SRP



Connection management

- “Session” is connecting a client with a server.
- Consists of as many IB connections as CPUs on client.
- Each IB connection: separate cq_vector (and IRQ).
- Affinity of each IRQ is set to a separate CPU.
- Server sends IO response on the same connection he got the request on.
- Interrupt on client is generated on the same cpu where the IO was originally submitted.
- Reduce data access across different NUMA nodes

PROFITBRICKS
The IaaS-Company.

Queue Depth and MQ support

- Inflight on client side is limited by the number of buffers reserved on the server side
- All the ibnbd devices mapped from the same server share the same remote buffers
- Fair sharing by making use of the shared tags feature
- MQ: As many hardware queues as CPUs - each IB connection belonging to a session does in fact function as a separate hardware queue.

Error handling

- No IO timeouts and no IO retransmissions
- Heartbeats to detect unresponsive peers (i.e. kernel crash)
    - RDMA might succeed even if CPU on remote is halted
- Reconnecting after an IB error
    - Client keeps the devices and tries to reconnect
    - Server closes all devices and destroys session
- APM Support
    - Server is connected with two IB ports to two different switches
    - transparent failover in case of cable or IB switch failure

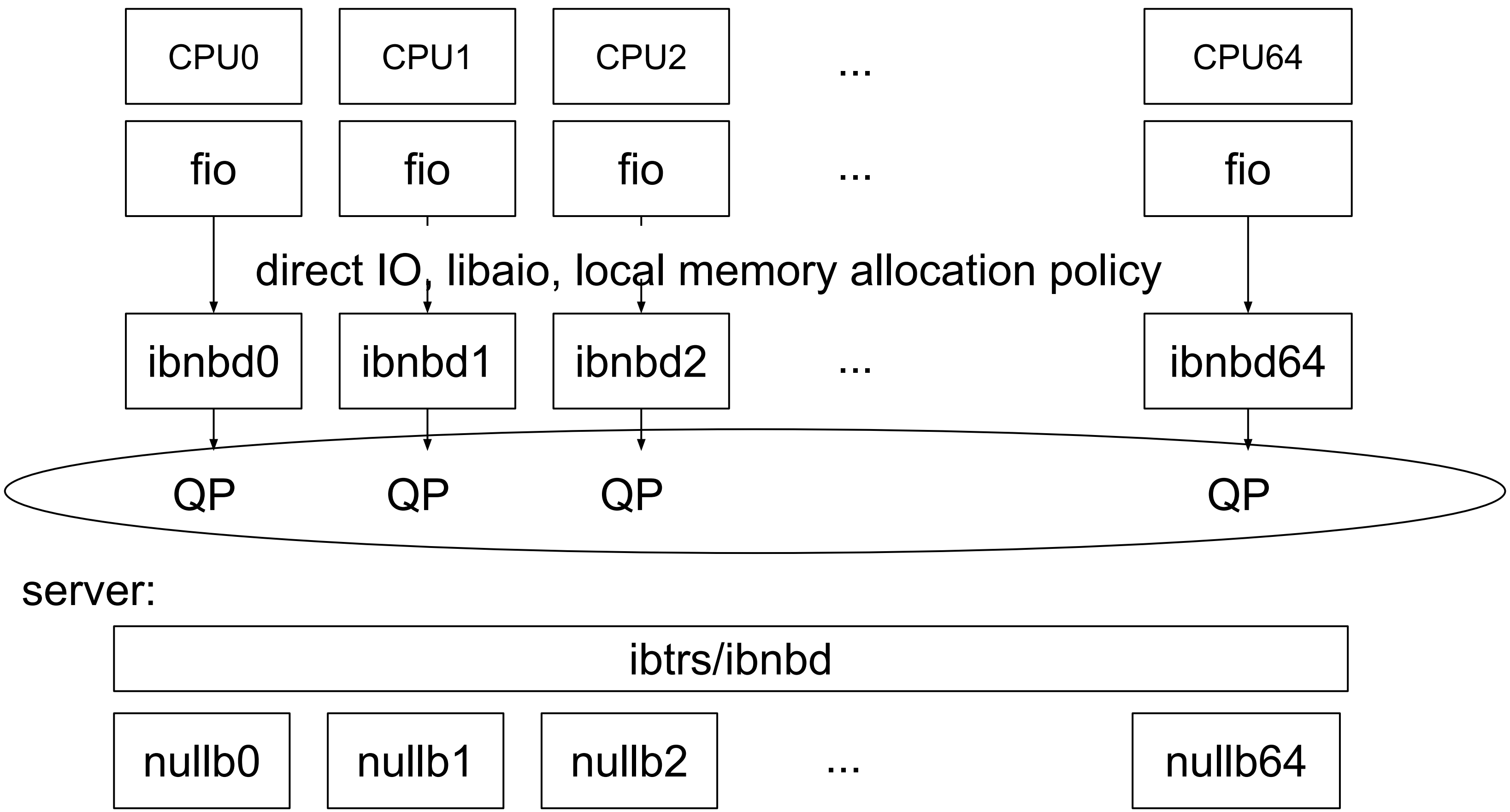
Outlook: Reliable Multicast

- Reliable multicast
- IBTRS API: Join several established sessions into one "multicast" session
- Submit IO once - it will be confirmed after the IO is delivered to all servers in the group
- Useful for replication (i.e. mirror)
- Reduce load on the IB link connecting a compute node with the IB switch

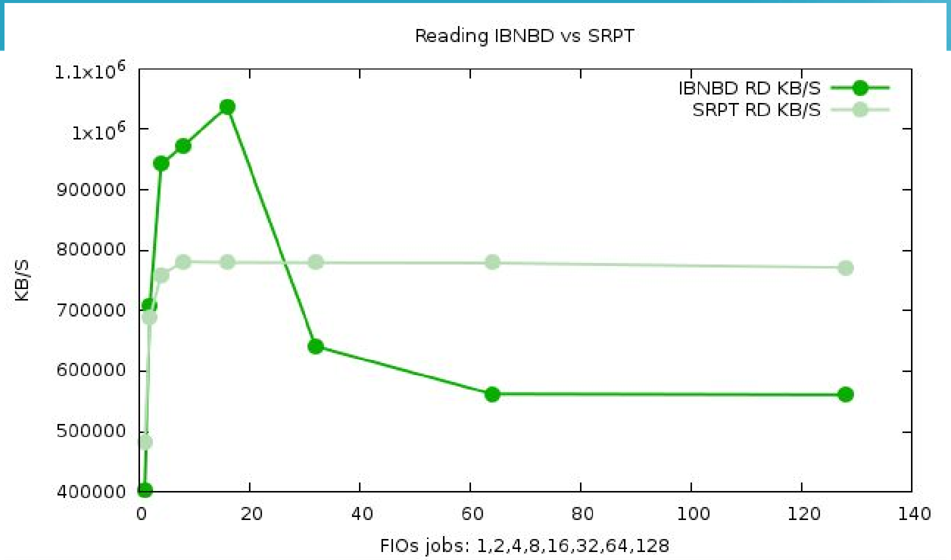
Performance: Measurement setup

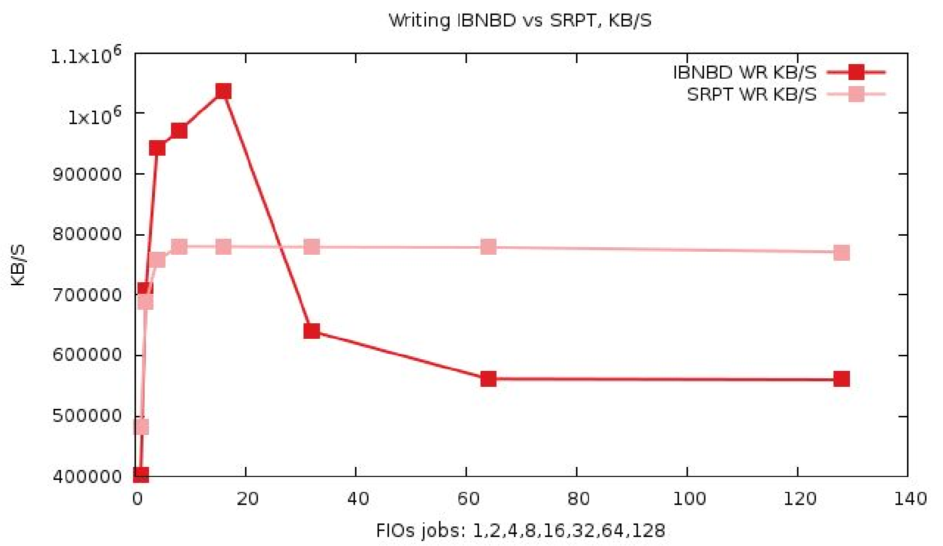
Mimic VMs running on different CPUs and accessing their devices.

client:



Original scalability problem



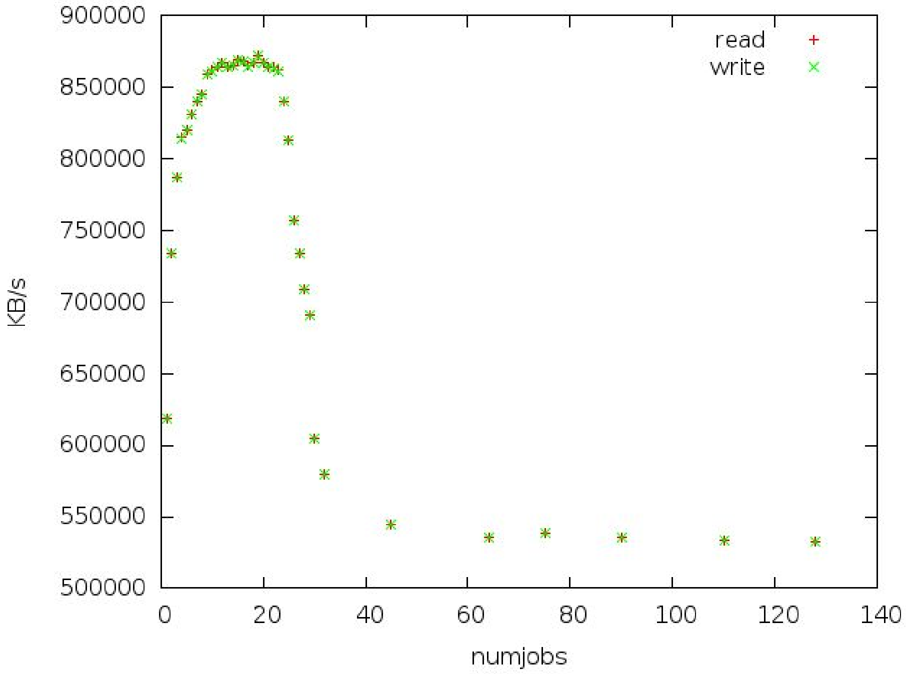


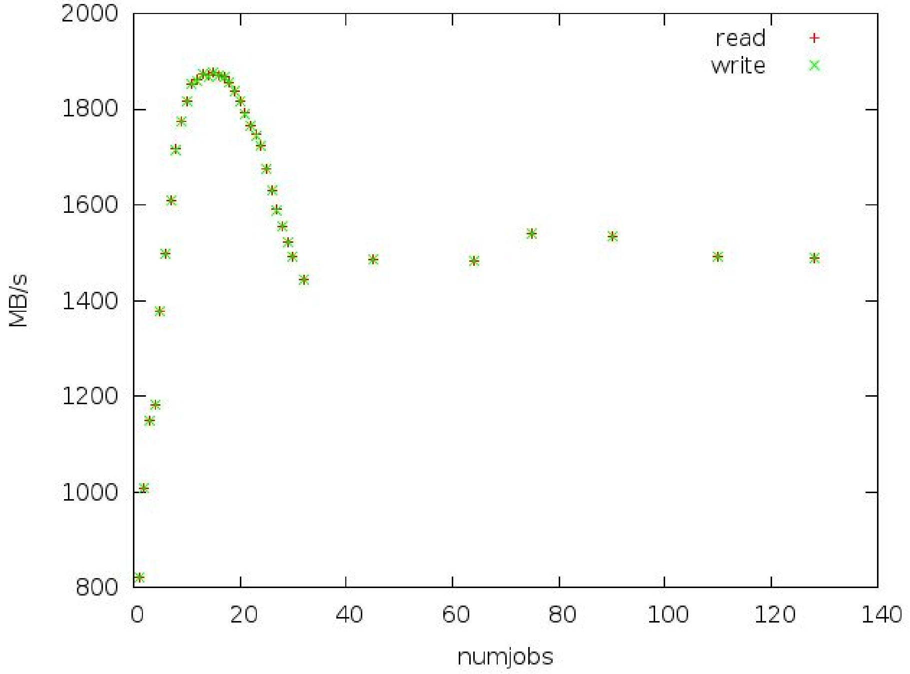
PROFITBRICKS
The IaaS-Company.

IOMMU

```
+   97.59%    0.00%          8          fio  [.] io_submit
+   97.58%    0.00%          0          fio  [k] sys_io_submit
+   97.54%    0.01%        397          fio  [k] do_io_submit
+   97.48%    0.01%        397          fio  [k] aio_run_iocb
+   97.07%    0.03%       2059          fio  [k] blkdev_direct_IO
+   97.04%    0.00%        172          fio  [k] __blockdev_direct_IO
+   96.99%    0.06%       3520          fio  [k] do_blockdev_direct_IO
+   95.11%    0.00%        282          fio  [k] submit_bio
+   95.09%    0.00%        168          fio  [k] generic_make_request
+   93.47%    0.04%       2577          fio  [k] map_sg
-   92.60%   92.60%    5786351          fio  [k] _raw_spin_lock_irqsave
      - _raw_spin_lock_irqsave
         + 50.39% map_sg
         + 49.38% unmap_sg
+   48.86%    0.00%        124          fio  [k] blkdev_write_iter
+   48.85%    0.01%        518          fio  [k] __generic_file_write_iter
+   48.82%    0.00%        269          fio  [k] generic_file_direct_write
```

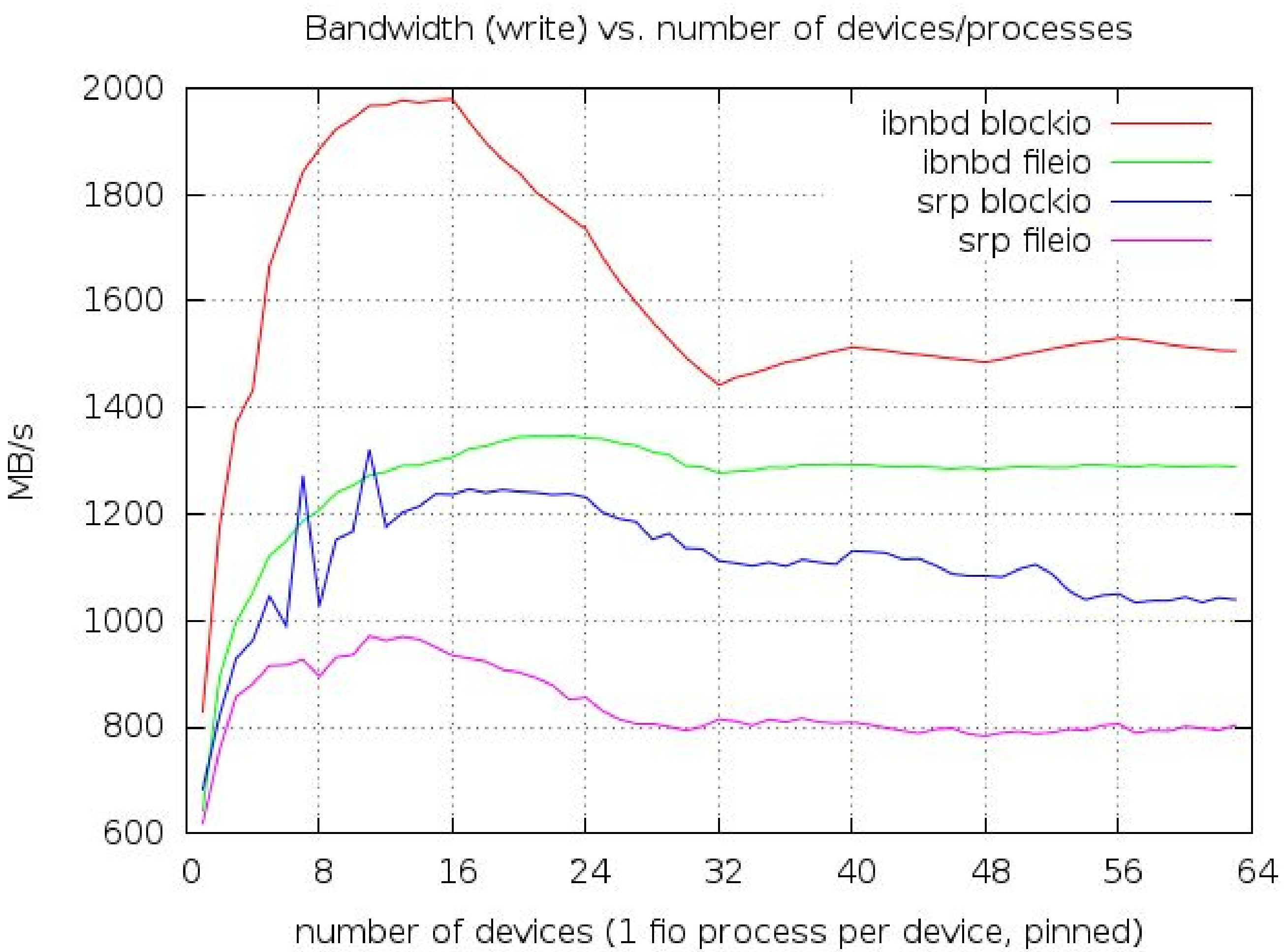
IOMMU vs no IOMMU





PROFITBRICKS
The IaaS-Company.

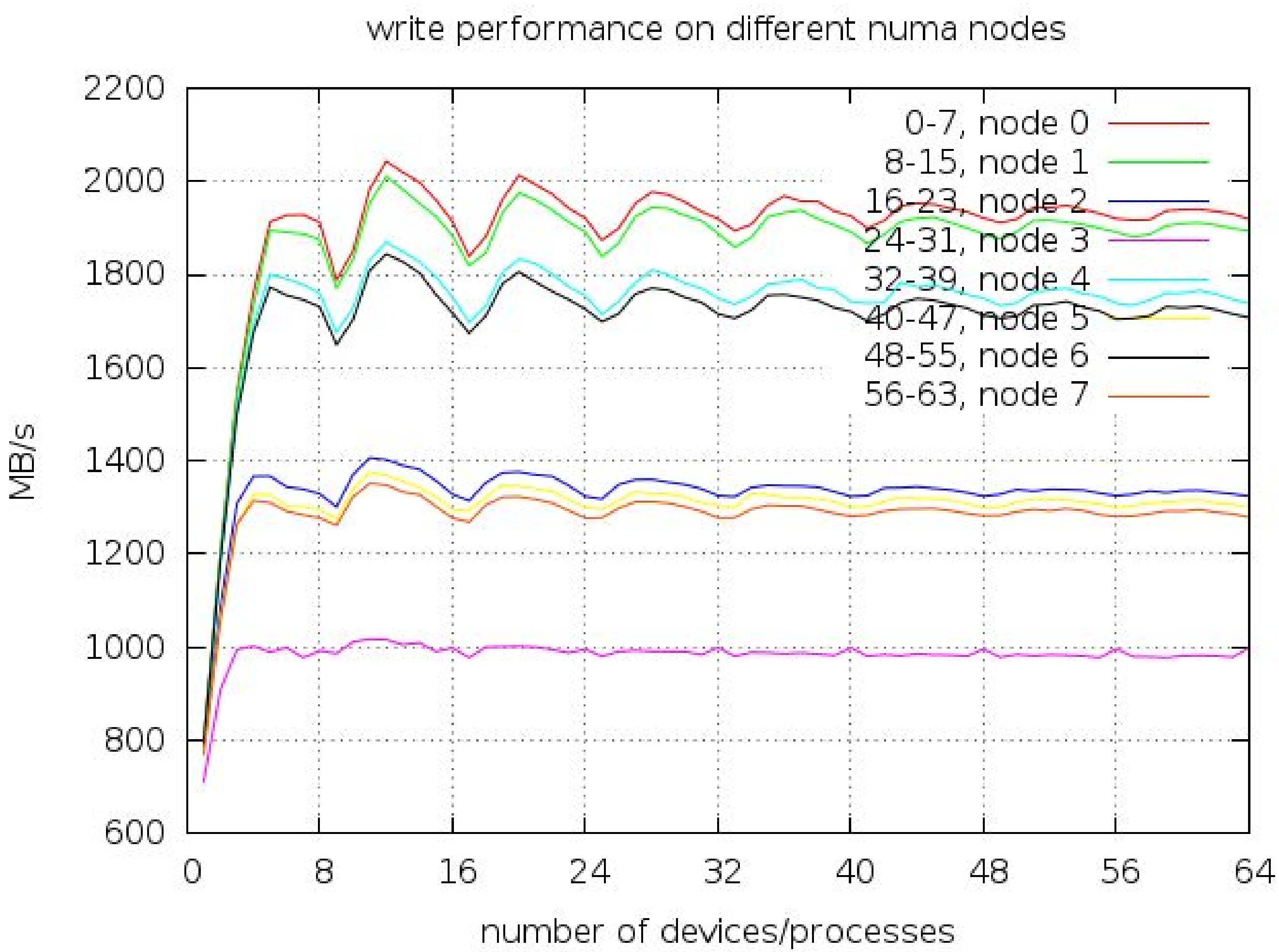
IBNBD vs SRP, block io vs, fileio, NUMA effects



PROFITBRICKS
The IaaS-Company.

NUMA effects

```
numa-ctl --hardware
node     0    1    2    3    4    5    6    7
   0:   10   16   16   22   16   22   16   22
```



HCA is on NUMA 0

Summary: Major characteristics of the driver

- High throughput and low latency due to:
  - Only two rdma messages per IO
  - Simplified client side server memory management
  - Eliminated SCSI sublayer
- Simple configuration and handling
  - Server side is completely passive: volumes do not need to be explicitly exported
  - Only IB port GID and device path needed on client side to map a block device
  - A device can be remapped automatically i.e. after storage reboot
- Pinning of IO-related processing to the CPU of the producer

Existing Solutions

- SRP/SCST
  - SCSI RDMA Protocol
- ISER
  - iSCSI extension for RDMA
  - target executes RDMA operations
- accelio/nbdx
  - server side in user space
  - obsolete in favor of NVMEoF
- NVMEoF
  - transports NVME commands
  - target initiates RDMA transfers

PROFITBRICKS
The IaaS-Company.