iSER Storage Target for Object-based Storage Devices

Dennis Dalessandro
Ananth Devulapalli
Pete Wyckoff (speaker)

Ohio Supercomputer Center

{dennis,ananth,pw}@osc.edu

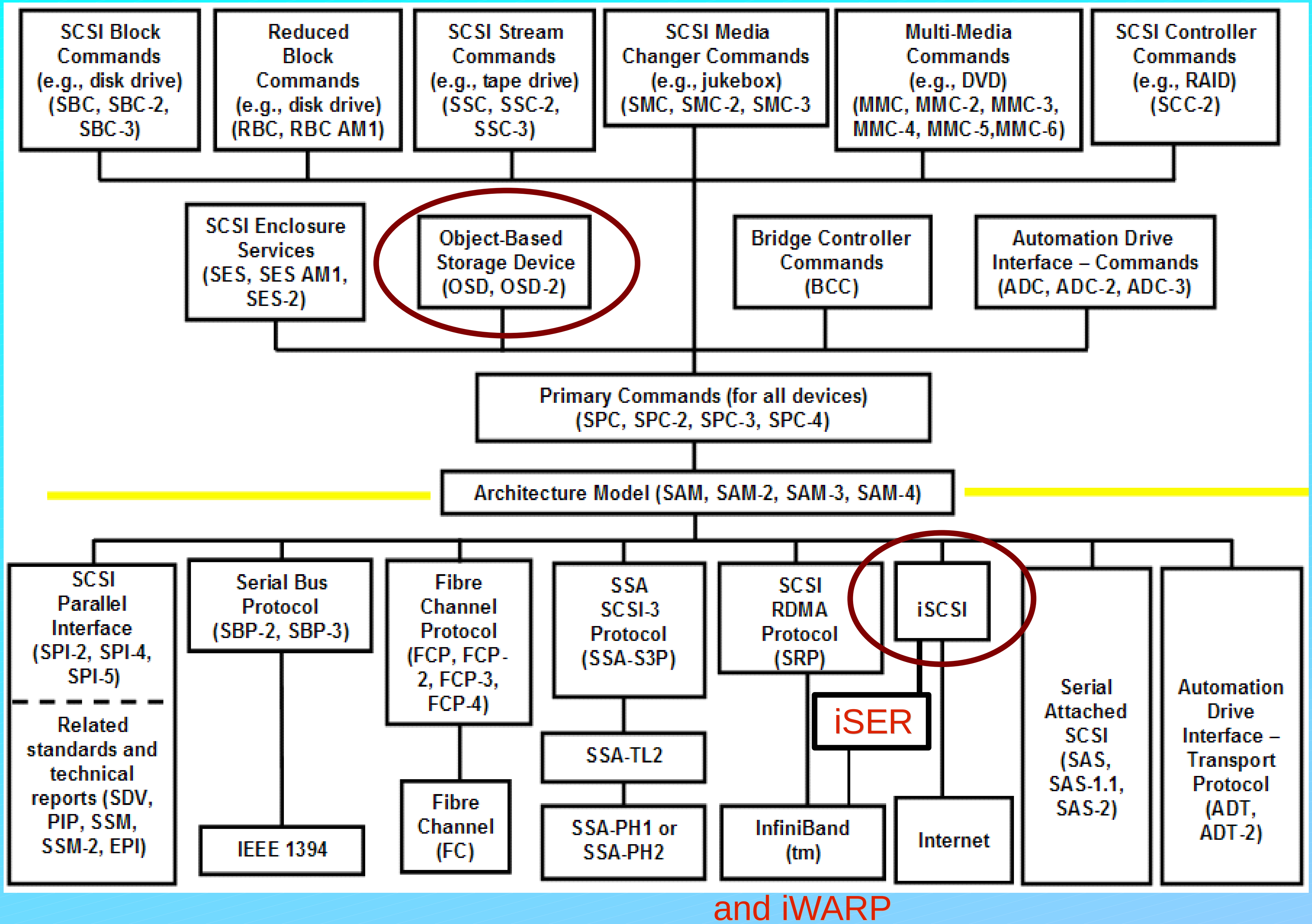
SNAPI '07
24 September 2007

Work supported by the National Science Foundation, #0621484

Storage Interconnects

- Locally Attached
  - parallel SCSI
  - SATA
- Fibre Channel
  - dominates market
- iSCSI
  - much smaller market, but growing
- iSER, SRP
  - RDMA-specific SCSI transports
- AoE
  - minimalist ethernet-based storage network
- Networks
  - 1 Gb/s Ethernet
  - 10 Gb/s Ethernet
  - iWARP
  - InfiniBand
  - Myrinet

SCSI Architecture



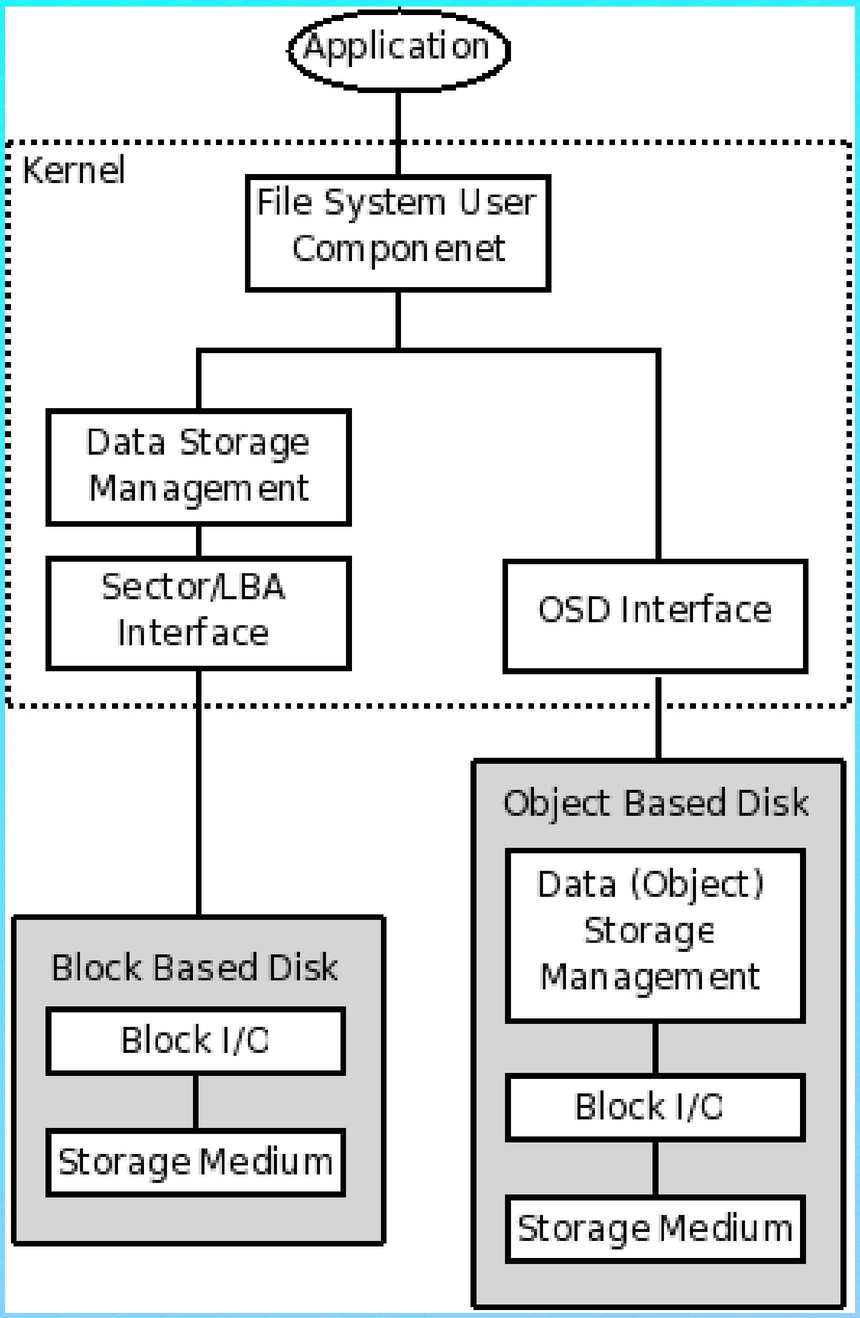
RDMA

- Two major aspects
  - Protocol Offload
    - NIC handles network processing
    - Removes biggest burden from host CPU
  - Zero Copy (with or without OS bypass)
    - Data is moved directly between network and user buffers
- TCP Offload Engine (TOE)
  - TCP/IP stack processing offloaded
  - CPU still moves data from buffers to user memory
- Remote Direct Memory Access (RDMA)
  - TCP/IP offloaded (in iWARP, or replaced in IB)
  - Data goes directly to and from user buffers
- Popular in High-Performance Computing
- Starting to get attention for other applications

OSD

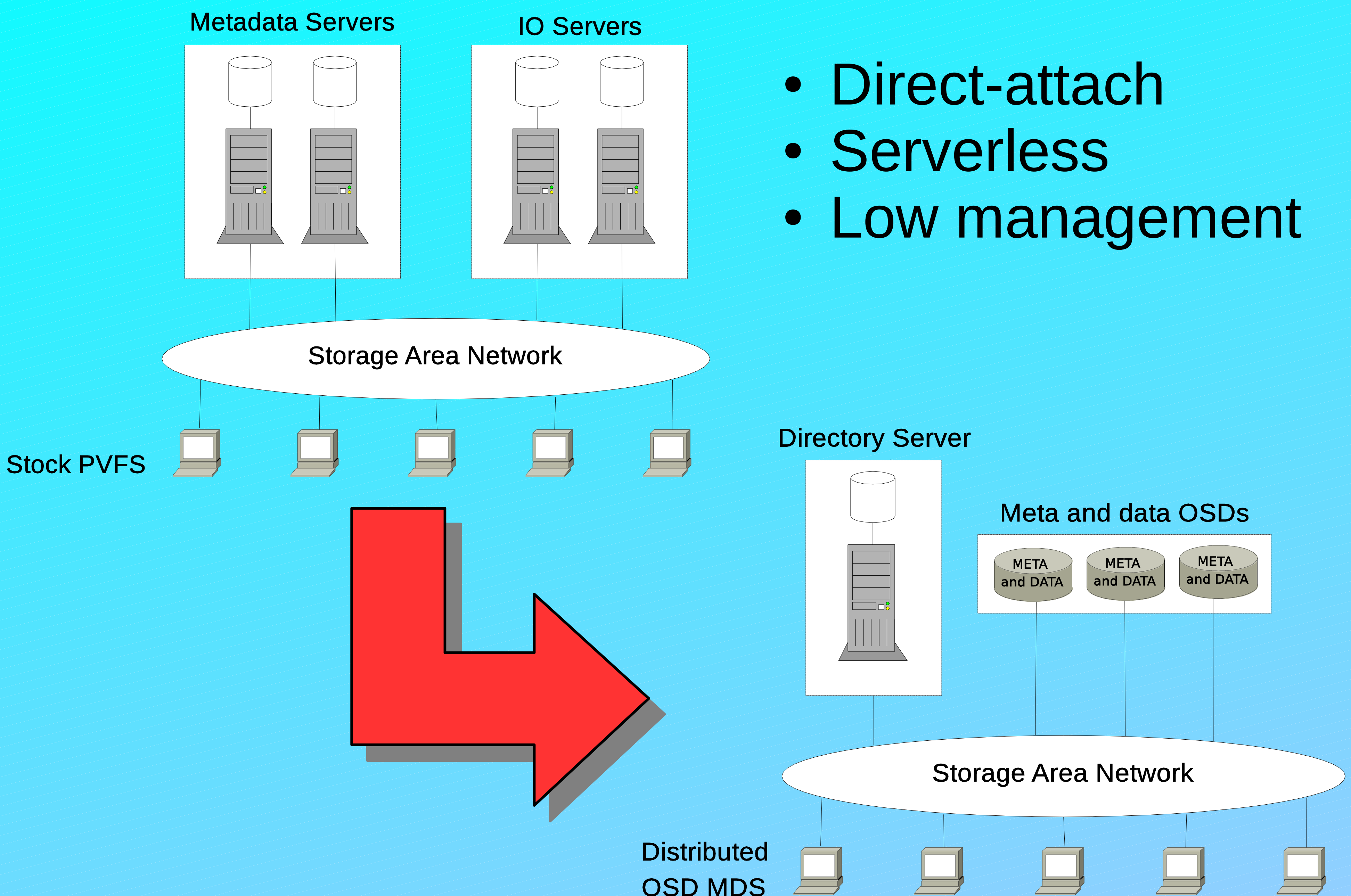
- T10 Specification
- Stores objects
- User attributes
- Strong security
- Pure target device

- SCSI Features
  - Bidirectional
  - Extended CDBs

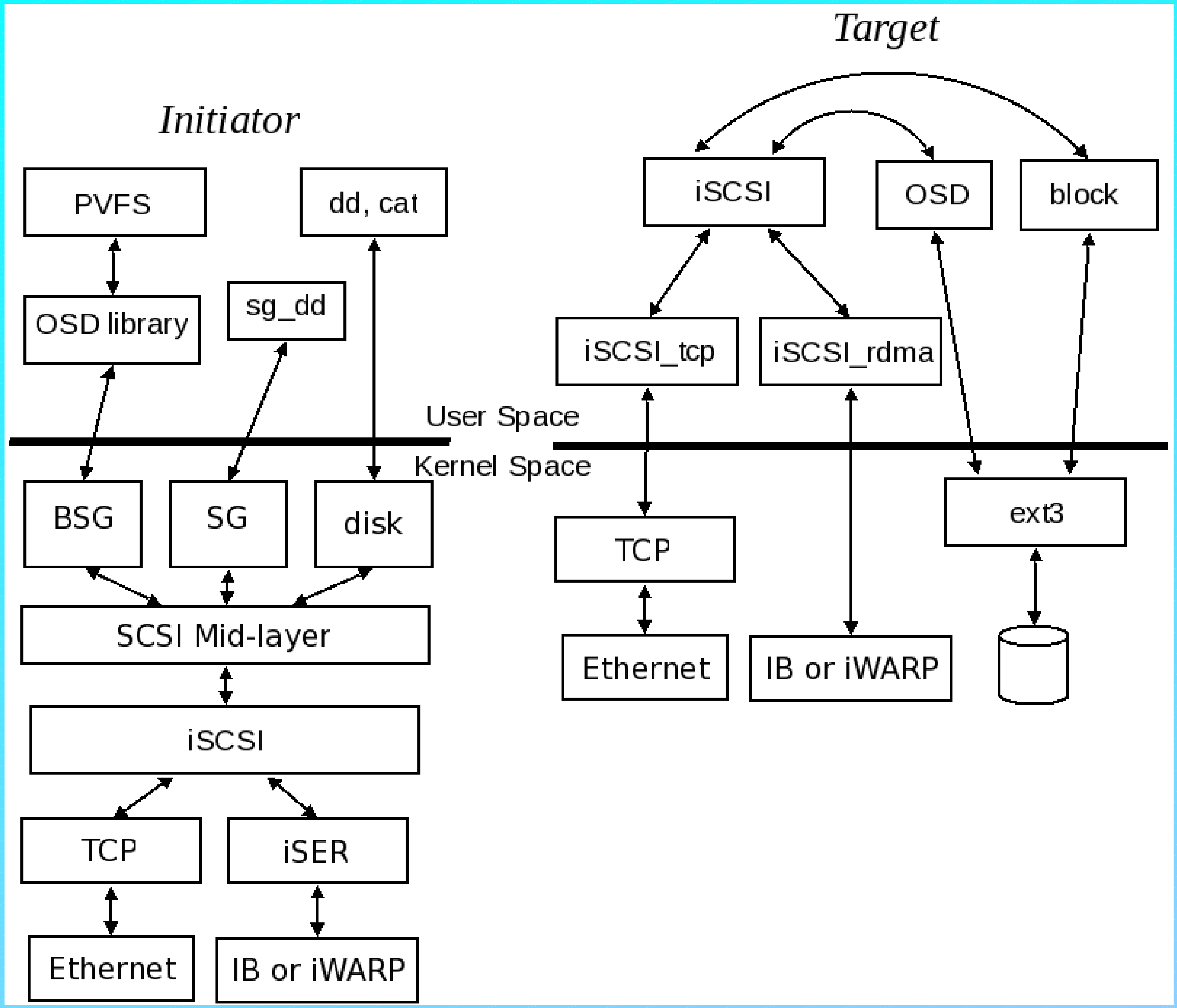


Parallel File System Design

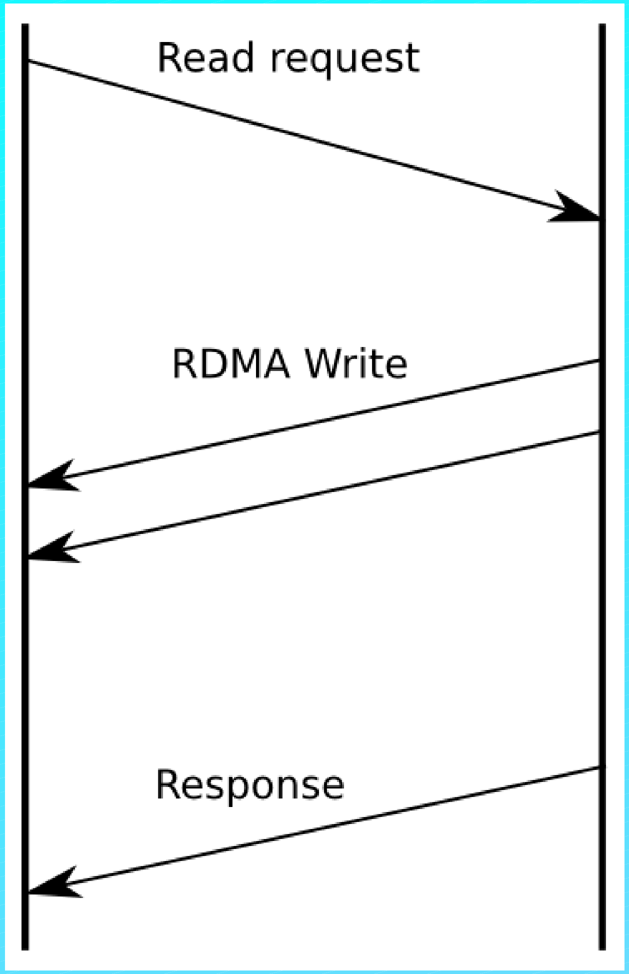
- Direct-attach
- Serverless
- Low management

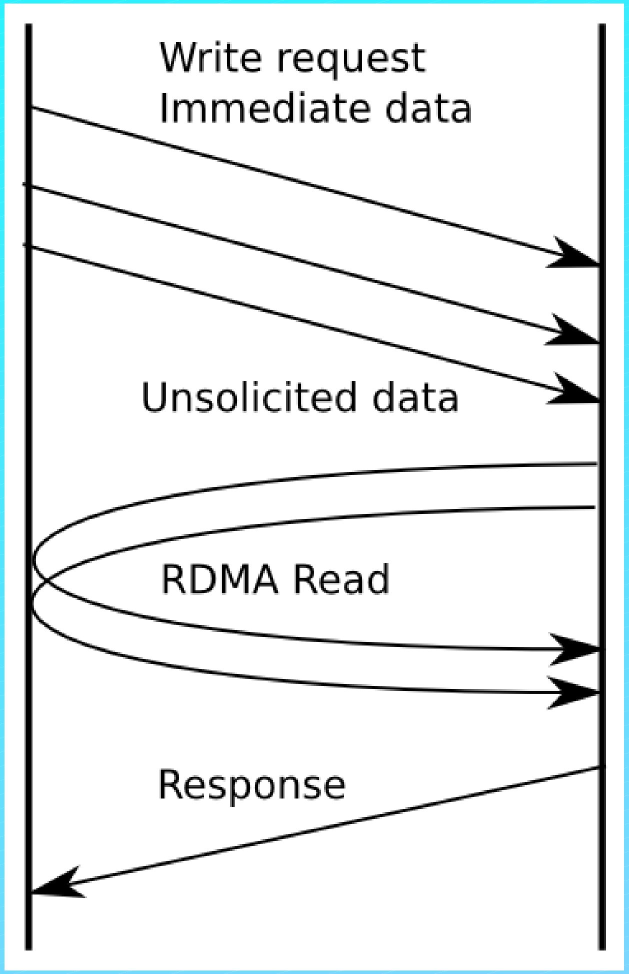


Architectural Overview



Data Flow





- Target initiates all data transfers
- Except for immediate, unsolicited data in write

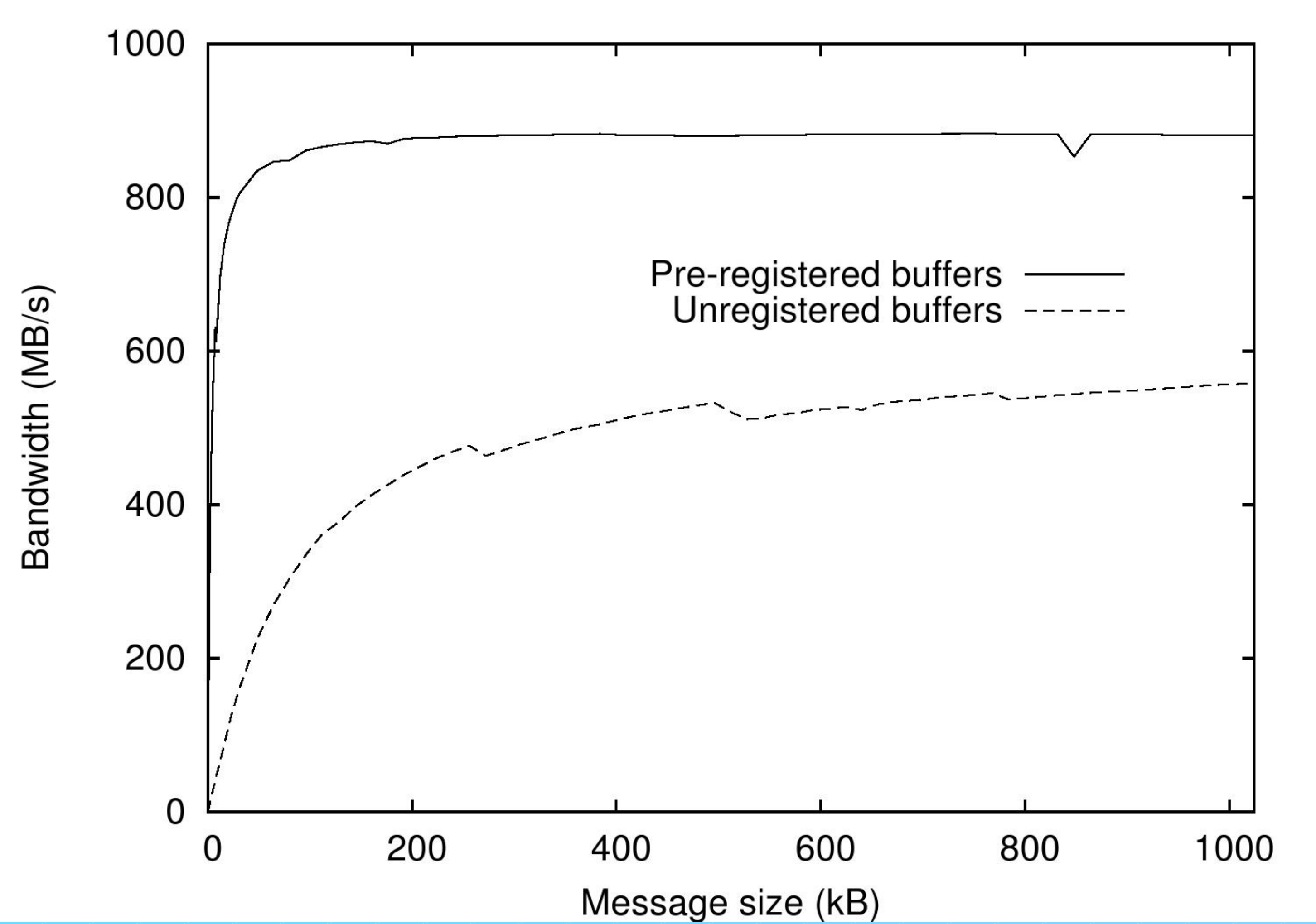
iSER Design and Implementation

- Memory registration
- Event management
- Data completion semantics
- Padding

- Modifications to existing stgt project by FUJITA Tomonori and others
- 18 separate patches for easier review
  - infrastructure additions
  - virtualization of aspects of iSCSI core
  - more parameters to negotiate
  - entire RDMA transport layer

Memory Registration

- Required for direct-access network protocols
- Act of registration is very slow: 30 to 100 µs



- Static registration makes sense for server

Event Management

- iSCSI target uses file descriptor polling
- One fd per connection
- Readable = incoming PDU
- Writeable = socket buffer space to send more
- Remember TX state using poll bits

- RDMA uses one fd for CQ notifications
- No concept of writeable

- Maintain separate list of ready-to-TX conns
- Non-zero counter drives progress engine
- Difficult to sequence state machine properly

Data Completion

- SCSI Read operation
  - Initiator: issue Send request for a READ
  - Target: receive Send request
  - Target: issue RDMA Writes
  - Target: issue Send response
  - Initiator: receive Send response
  - Initiator: are RDMA Writes finished?
- RDMA Write operations are not ordered with respect to the response
- Add state:
  - Target: wait for RDMA Writes to finish
  - Target: issue Send response
  - ...

Padding

- Messages (PDUs) consist of multiple segments
- Request
  - Header (48 bytes)
  - Add'l header 1 (200 bytes)
  - Add'l header 2 (8 bytes)
  - Header digest (4 bytes)
  - Data segment (7800 bytes)
  - Data digest (4 bytes)
- Data-out
  - Header (48 bytes)
  - Header digest (4 bytes)
  - Data segment (1 byte?)
  - Data digest (4 bytes)
- iSCSI says segments must be four-byte aligned
- iSER is quiet about padding
- So, pad between segments, but not data
- Avoids significant complexity on initiator

Experiments

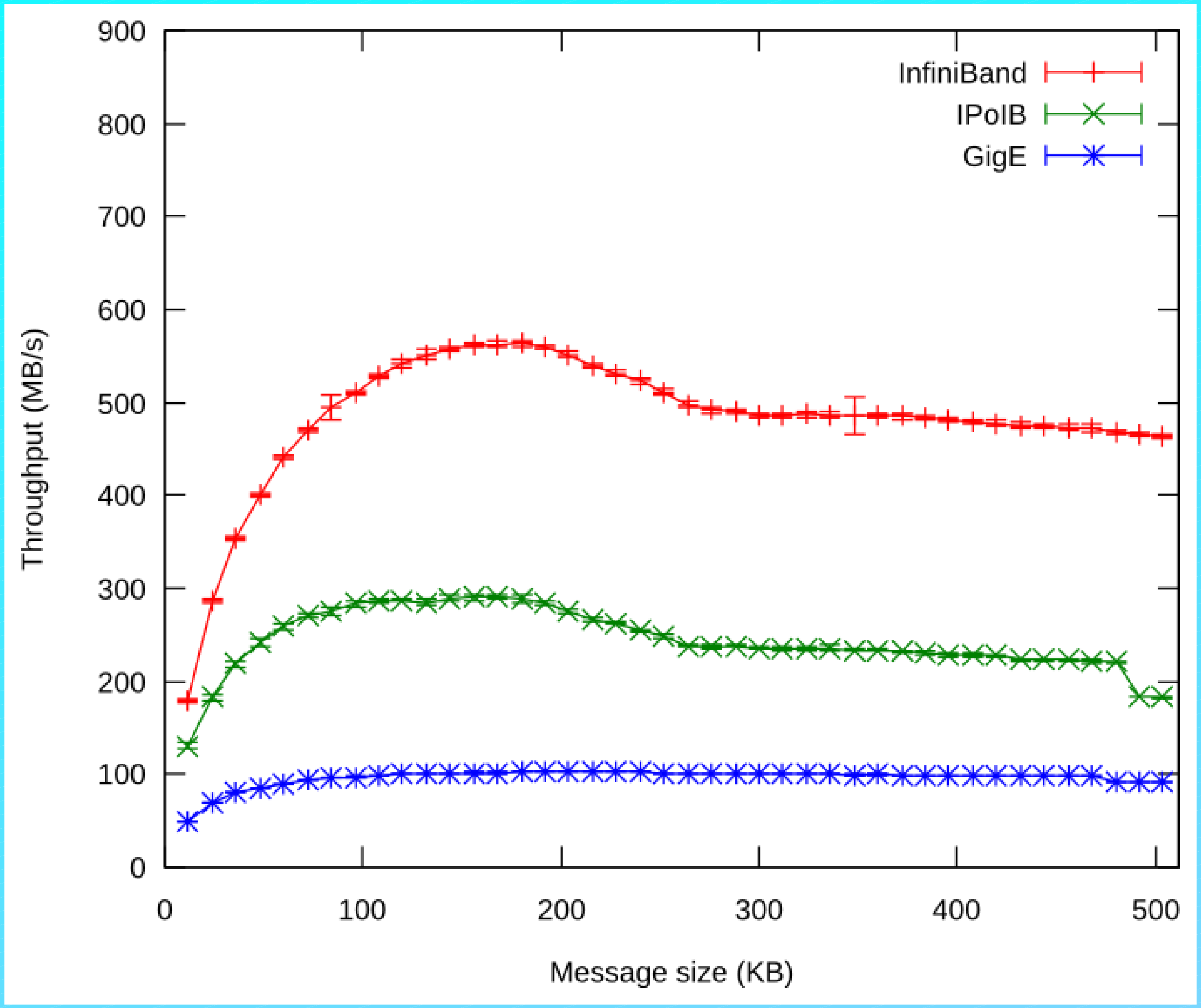
- Tyan S2891 motherboard
- Dual 2.4 GHz Opteron
- 2 GB Memory
- 80 GB SATA
- Mellanox 4X SDR, switch
- Linux 2.6.22-rc5
  - plus bidirectional patches
  - plus little OSD bits
  - plus AHS for TCP and iSER
- Linux 2.6.23-rc6
  - stock for block experiments
- OpenFabrics libmthca, libibverbs, librdmacm

Latency

OSD command	TCP	IPoIB	IB
Ping	86.94 ± 3.87	36.42 ± 3.63	33.27 ± 3.53
Create	265.26 ± 9.81	220.11 ± 3.59	206.76 ± 3.05
Remove	257.36 ± 17.61	215.36 ± 11.02	201.05 ± 14.74
Getattr	143.89 ± 2.74	85.51 ± 1.58	65.41 ± 0.63
Setattr	238.54 ± 53.55	201.27 ± 3.18	175.14 ± 2.65

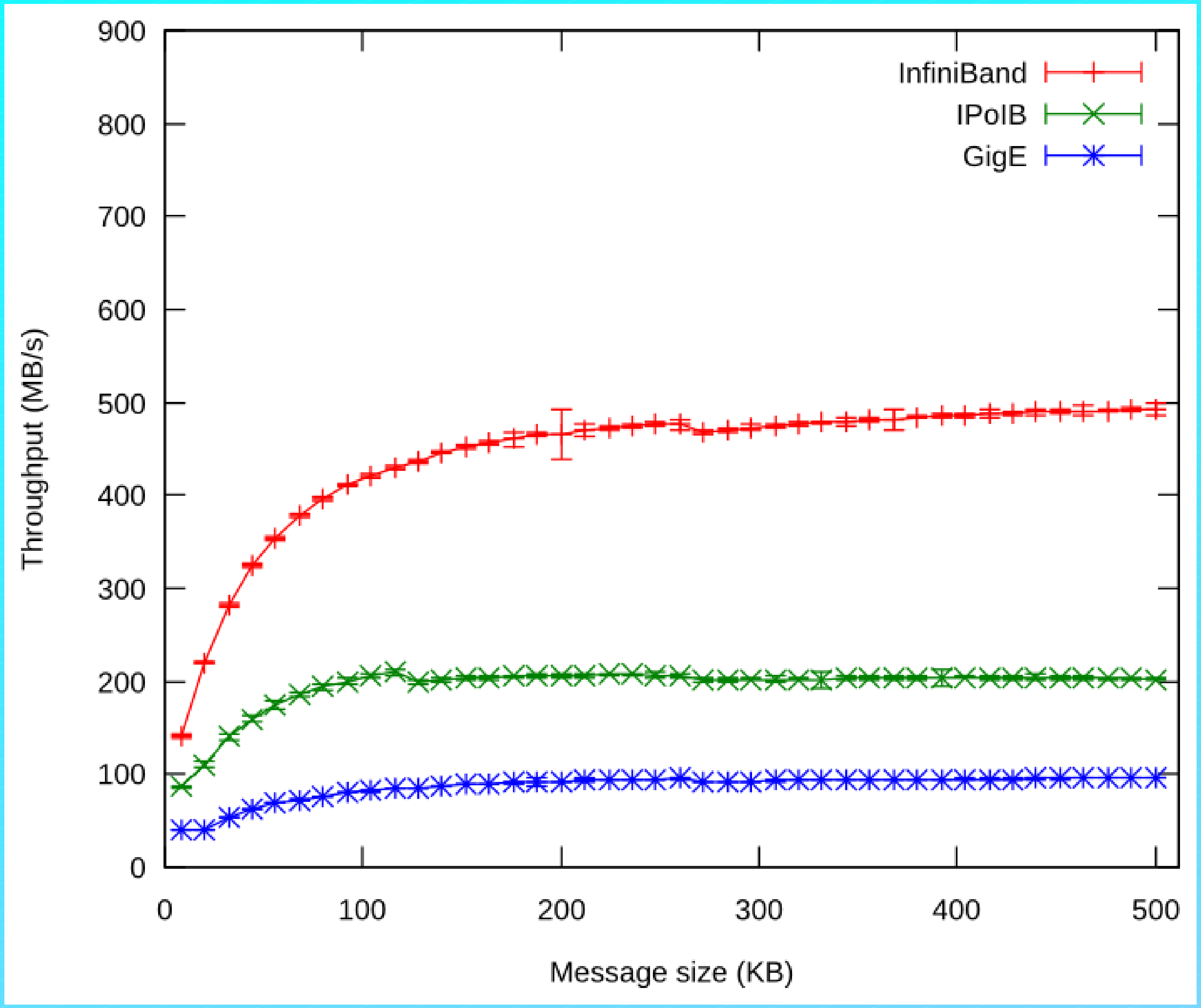
- Units in microseconds
- Differences arise from network latencies
  - IB 7 us
  - IPoIB 16 us
  - TCP 40 us
- No data transfers, except getattr
  - iSER does extra round-trip
  - no phase collapse

Single-client Read Throughput



- Only one command outstanding
- Gradual drop-off from cache effects in target

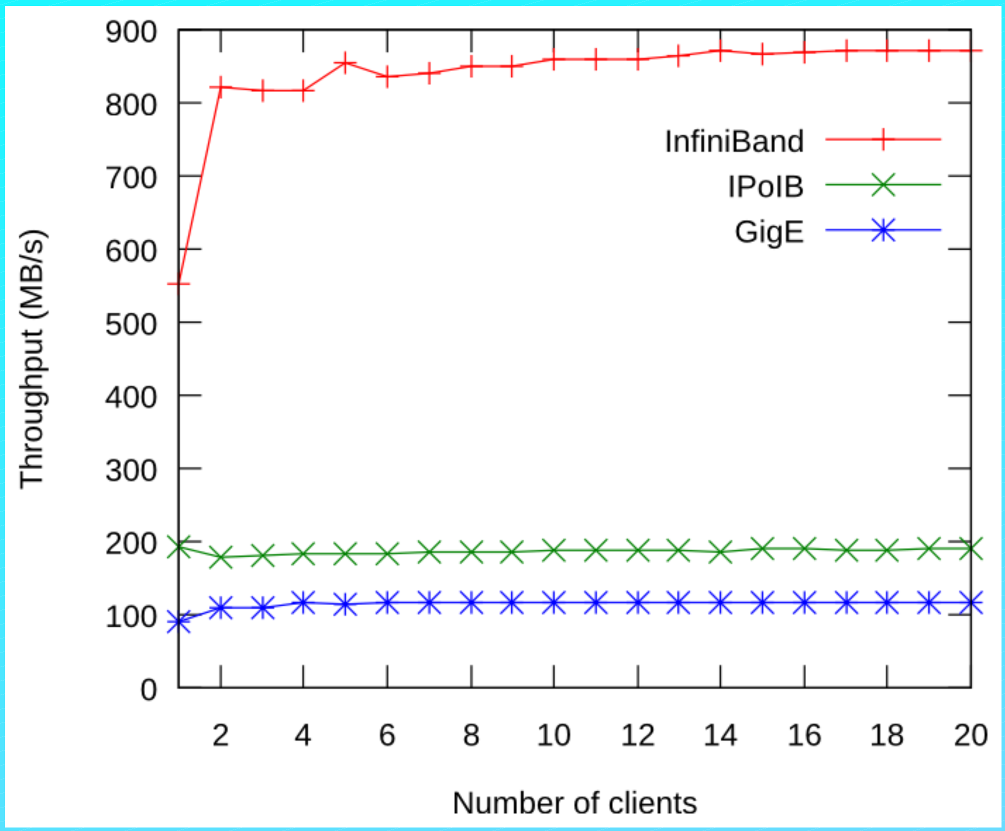
Single-client Write Throughput



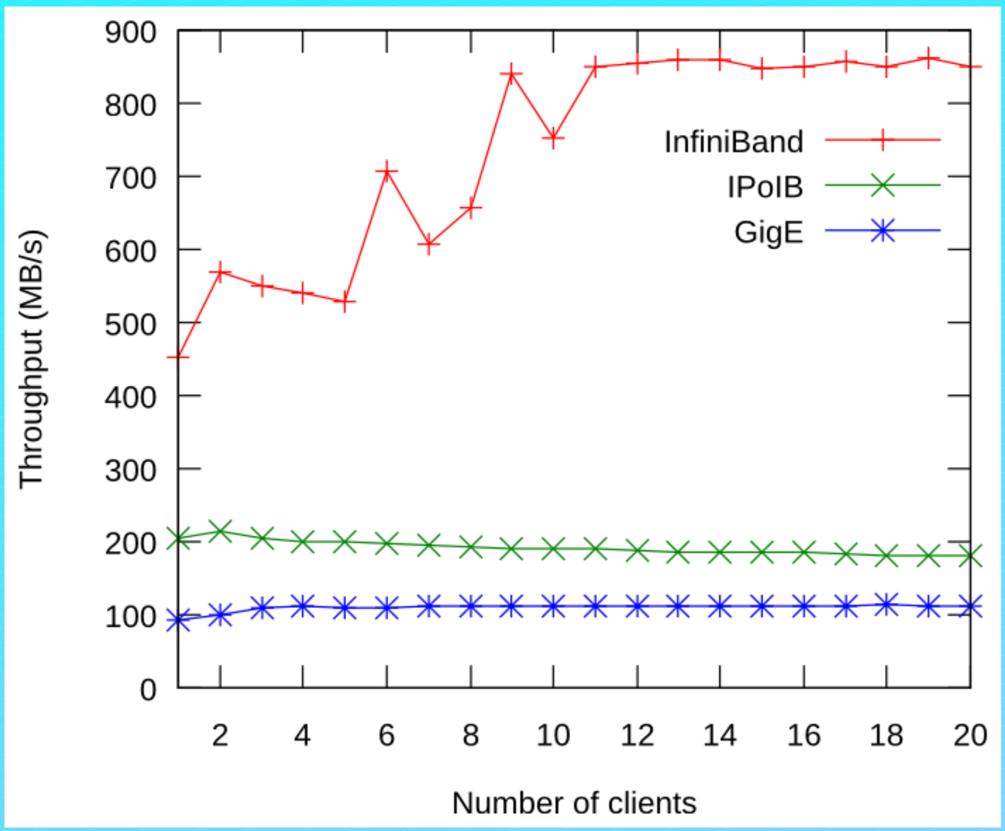
- Generally writes are slower
- Extra time for RDMA Read vs Write?

Multi-client Throughputs

Read



Write



- 200 kB message size, increasing clients
- MPI used for synchronization, timing

Block Experiments

- Replace OSD back-end with block back-end
- Different (and more usual) SCSI command set
- Latency, 16-byte Write or Read operations:

	Write	Read
GigE	113 μs $\pm$ 15 μs	112 μs $\pm$ 14 μs
IPoIB	64 μs $\pm$ 1 μs	62 μs $\pm$ 1 μs
iSER	46 μs $\pm$ 1 μs	56 μs $\pm$ 1 μs

- Higher than OSD latencies due to bs_sync
- Notice 10 µs read penalty for iSER
  - no phase collapse for small response data

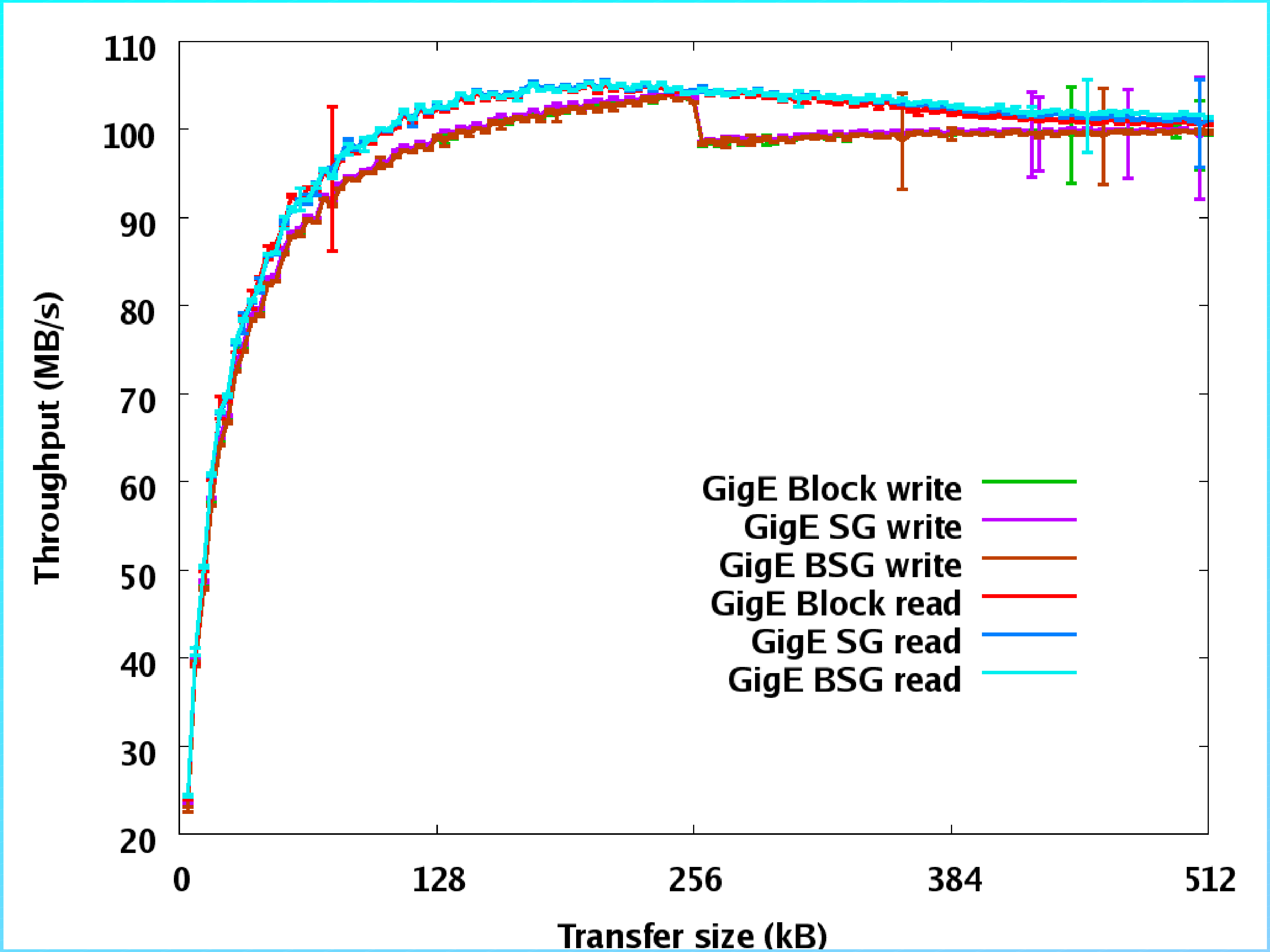
Initiator Interface Effects

- Initiator matters at high speed

- Three different ways to issue commands
  - Block: read and write to /dev/sdb
  - SG: ioctl(SG_IO) to /dev/sgN
  - BSG: ioctl(SG_IO) to /dev/bsg/sdb
- Actually more, and variations.

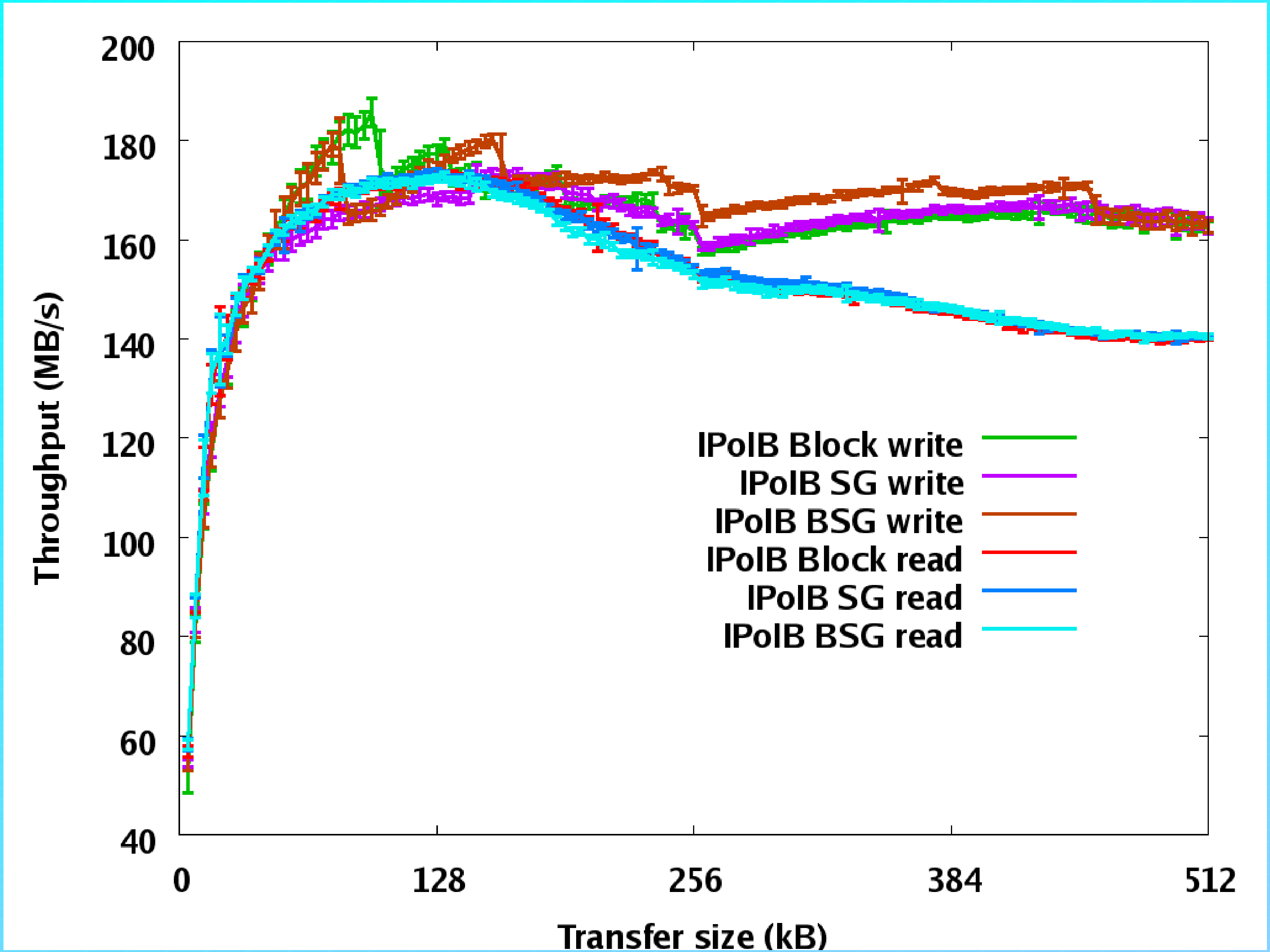
- Same setup for each of GigE, IPoIB, iSER
- Single command outstanding
- Read/write same block, stays in RAM

GigE



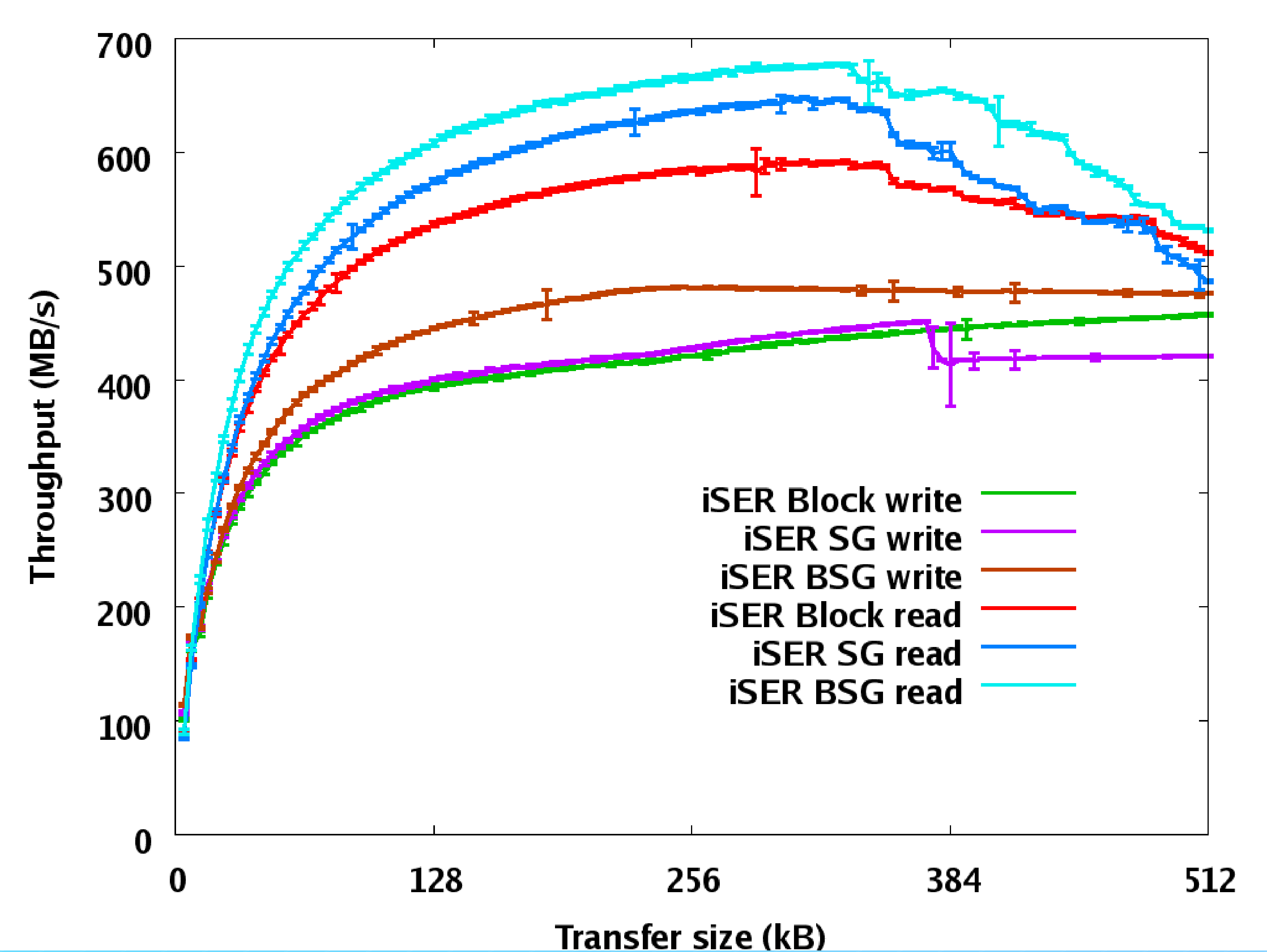
Reads faster than writes: one or two fewer round-trips

IPoIB



Writes faster than reads(!)

iSER



Reads faster than writes as one would expect.

Interface effect visible. Cache inval on write. Cache copy on read.

Timing Analysis

SCSI Read, 350 kB

	Time	Bandwidth
Total	564 μs	635 MB/s
Initiator	71 μs	
pread	94 μs	3810 MB/s
RDMA write	387 μs	930 MB/s
Ack	12 μs	

SCSI Read, 500 kB

	Time	Bandwidth
Total	945 μs	540 MB/s
Initiator	65 μs	
pread	315 μs	1625 MB/s
RDMA write	550 μs	930 MB/s
Ack	15 μs	

SCSI Write, 400 kB

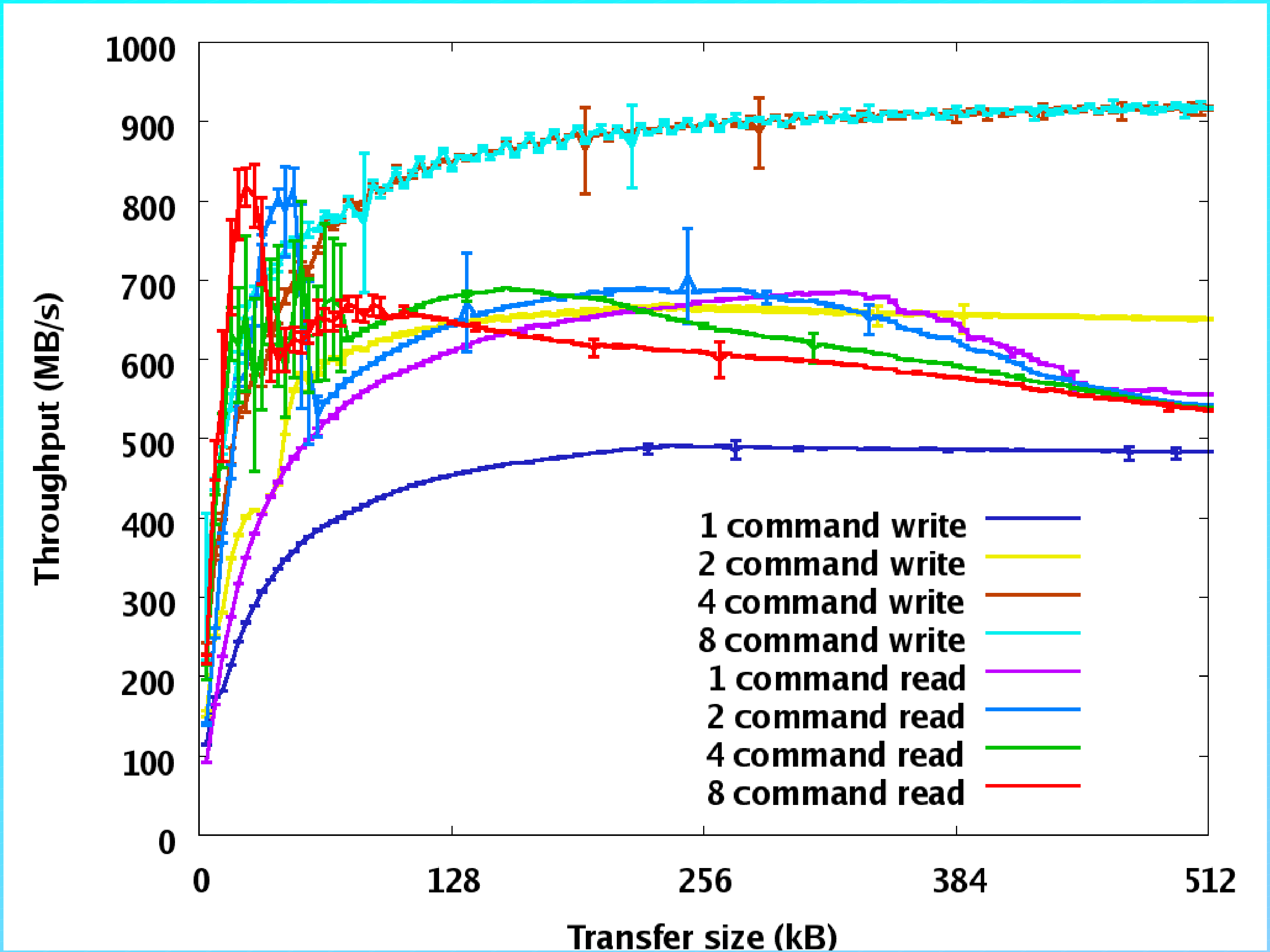
	Time	Bandwidth
Total	1020 μs	500 MB/s
Initiator	75 μs	
pwrite	492 μs	1040 MB/s
RDMA read	440 μs	1100 MB/s
Ack	12 μs	

Multiple-command Performance

- Block drivers issue multiple SCSI commands
- Current iSCSI maximum 128
- More, smaller transfers for pipelining

- Look at BSG performance
- Single client again

iSER Multiple Commands



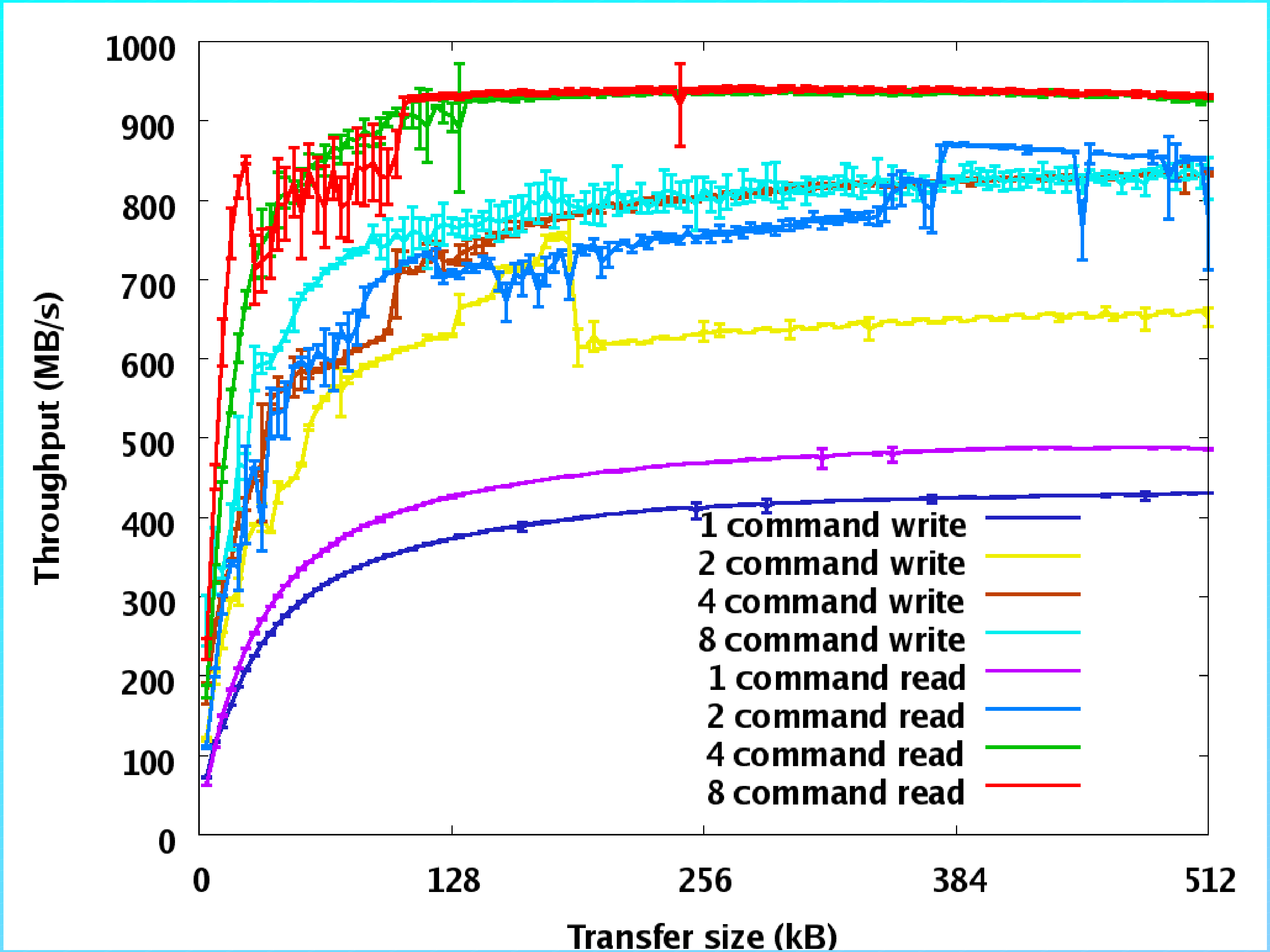
Great overlap possibility for write.
Little overlap for read, side effect of RX vs TX ordering.

Threading on the Target

- Use worker threads for IO
  - 1 SCSI thread
  - 4 IO threads
  - 1 IO completion thread
- Default configuration of tgt ("bs_sync")

- Inter-thread communication somewhat expensive
- Cost amortized when multiple commands present

iSER Threaded Target



Reads proceed at line rate.
Writes limited by slower RDMA Read operation, target queuing.

Related Work

- Voltaire
  - Good initiator work
  - Proprietary iSER target
- Other IB transports
  - SRP for point-to-point in DDN et al.
  - Custom protocols for PVFS, Lustre
- Sun
  - Initiator and target iSCSI work
- Intel folks
  - Allocate host CPU to iSCSI stack processing
  - CRC-32c is expensive for TCP
  - IB and iWARP transport layer provides checksum

Future Work

- iWARP
  - Current linux initiator depends on FMR
  - Opportunity to use iWARP STAG invalidate
  - Zero-based VA issues
- Memory requirements and flow control
  - Space for 128 outstanding commands per conn
  - Plus RDMA static buffers to reply to those
  - Only need a few for overlap
  - Linux initiator does not support MaxOutstUnexPDU
- SRP
  - Alternate RDMA transport for SCSI

Pull code from *git://git.osc.edu/tgt*
Browse source at *http://git.osc.edu/?p=tgt.git*
Mail issues to *pw@osc.edu*