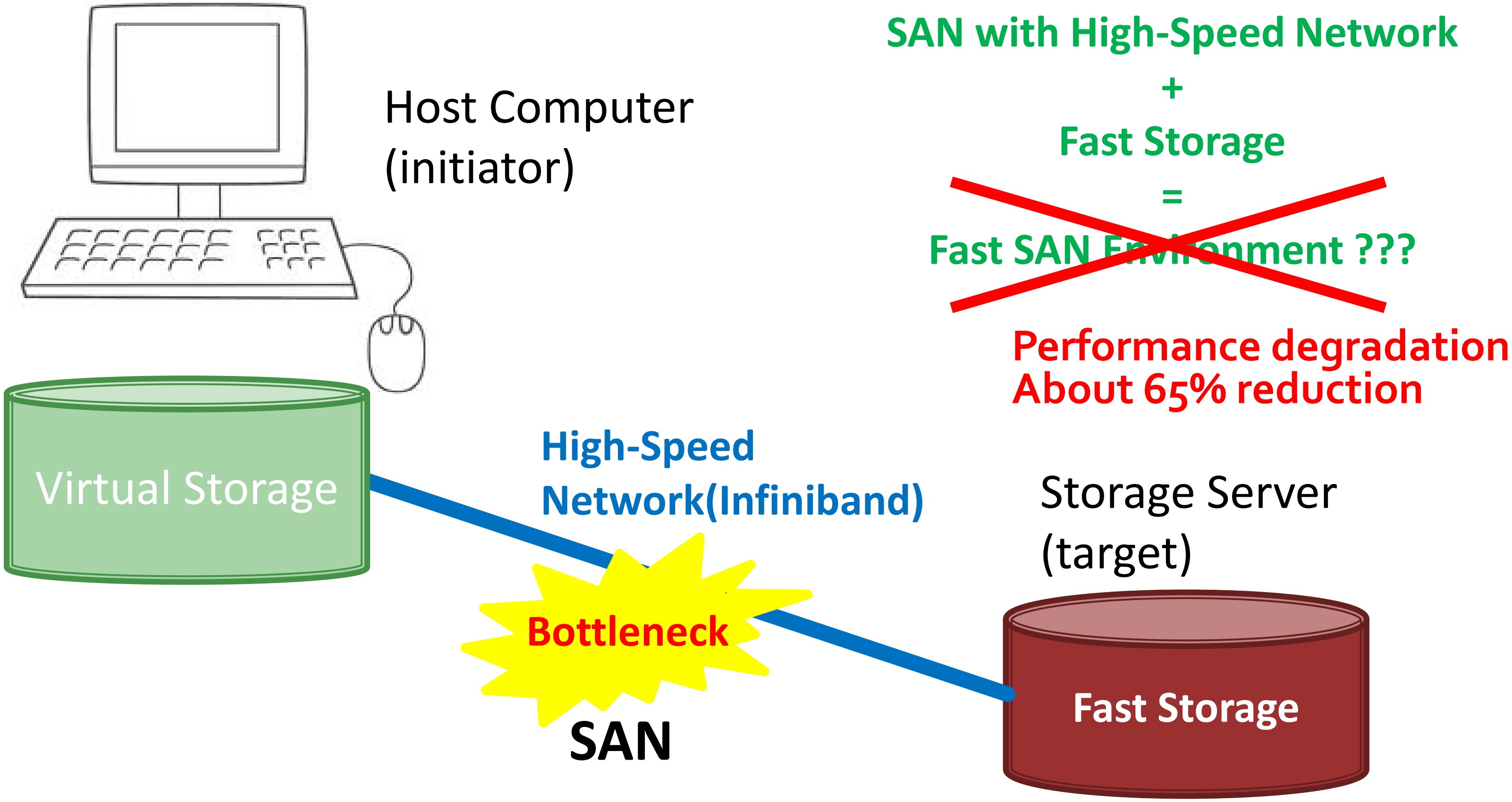
SAN Optimization for High Performance Storage with RDMA Data Transfer

Jae Woo Choi, Dong In Shin, Young Jin Yu,
Hyunsang Eom, Heon Young Yeom

Seoul National Univ. TAEJIN INFOTECH

Motivation



SAN with High-Speed Network

+

Fast Storage

~~=~~

~~Fast SAN Environment ???~~

Performance degradation
About 65% reduction

Contribution

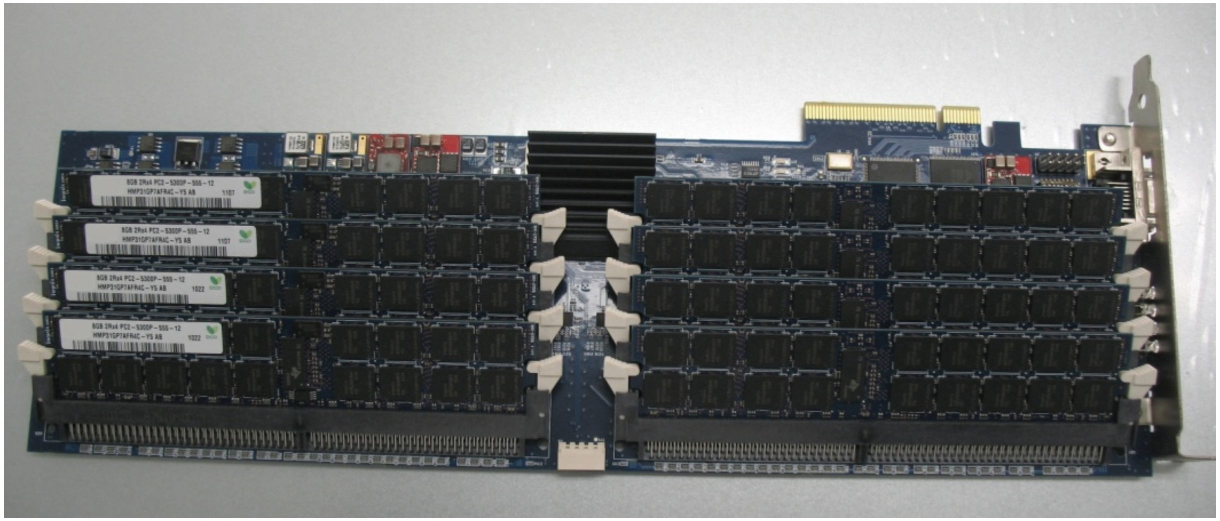
- Found performance degradation in existing SAN solution with a fast storage

- Proposed three optimizations for Fast SAN solution
  - Mitigate software overheads in SAN I/O path
  - Increase parallelism on Target side
  - Temporal merge for RDMA data transfer

- Implemented the new SAN solution as a prototype

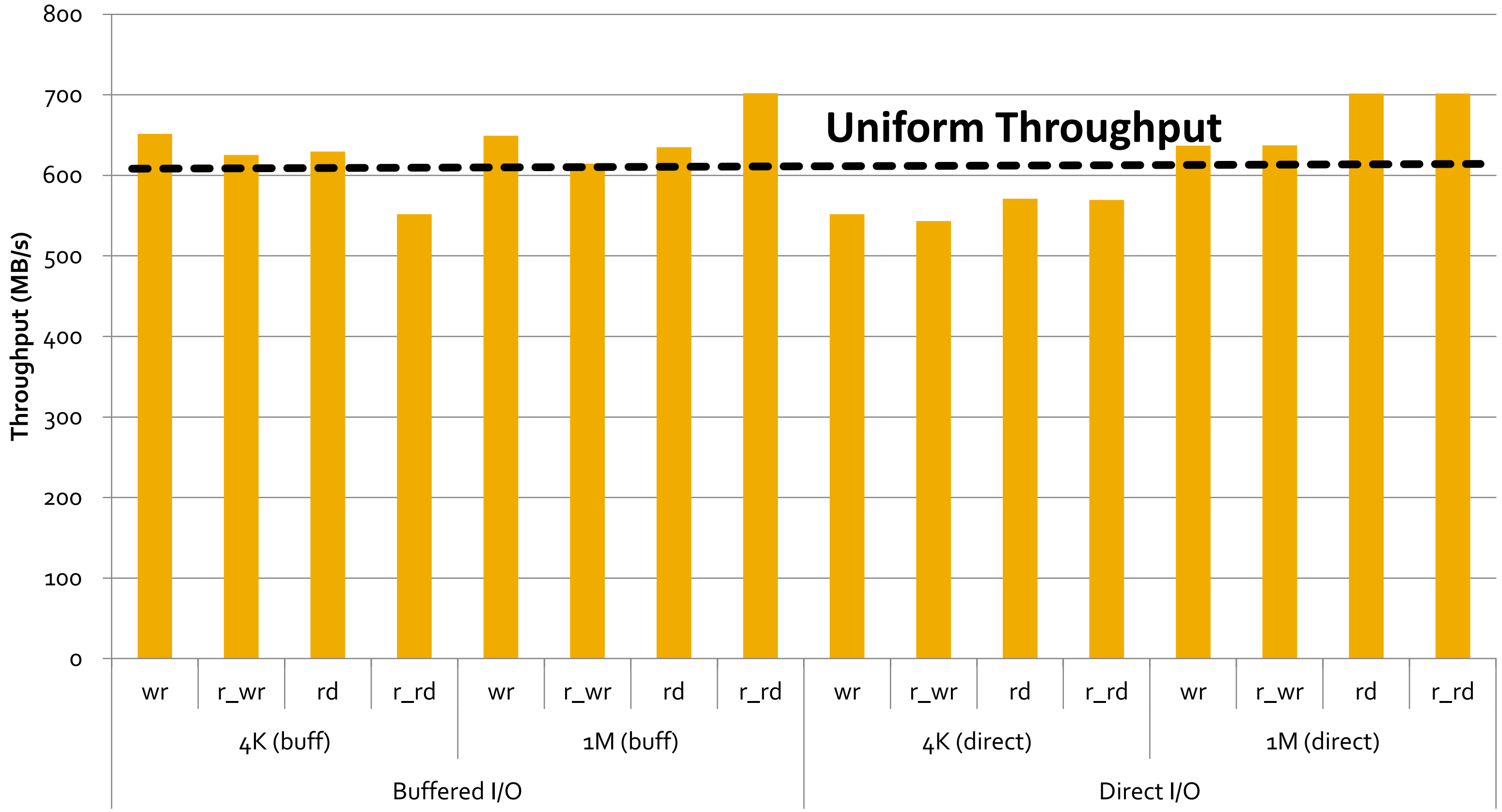
Fast Storage

- DRAM-SSD (provided by TAEJIN Infotech)
  - 7 usecs for reading/writing a 4KB page
  - Peak device throughput: 700 MB/s
  - DDR2 64 GB, PCI-Express type



DRAM SSD Performance Test

- FIO micro benchmark, 16 threads



SCST

- Generic SCSI Target Subsystem for Linux
    - Open Program for implementing SAN environment
    - Support Ethernet, FC, Infiniband and so on.
    - Use SRP(SCSI RDMA Protocol) for Infiniband

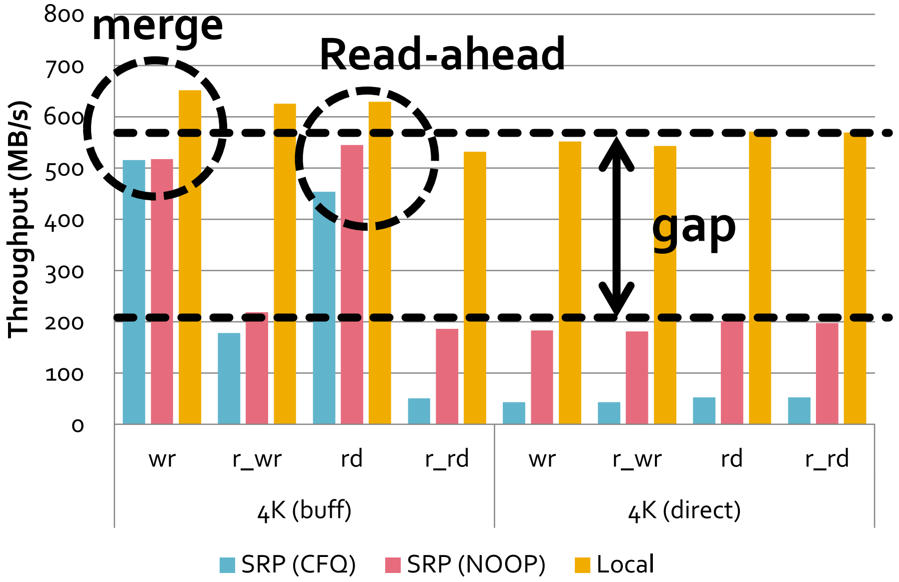
Evaluation Environment

SPEC	TARGET	INITATOR
CPU	Intel Xeon E5630 (8 core)	Intel Xeon E5630 (8 core)
Memory	16GB	8GB
INFINIBAND CARD	MHQH19B-XTC 1port (40Gb/s)	MHQH19B-XTC 1port (40Gb/s)

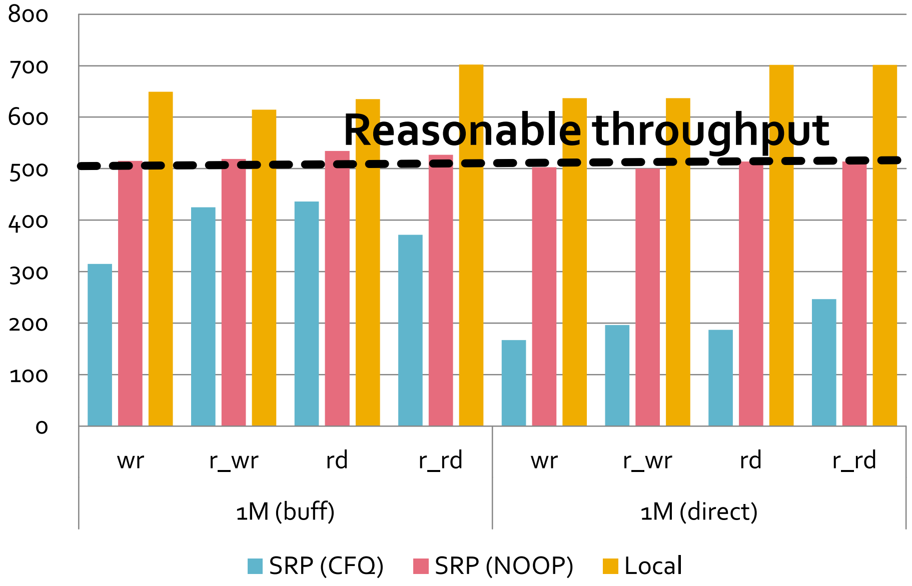
- Device :DRAM SSD(64GB)
- Workload size : 16 thread x 3GB (48GB)
- Request size : 4K/1M
- I/O type: Buffered/Direct, Sequential/Random, Read/Write
- Benchmark Tool : FIO micro benchmark

Evaluation (SCST)

- I/O Scheduler policy
  - CFQ -> NOOP



Small Size



Large Size

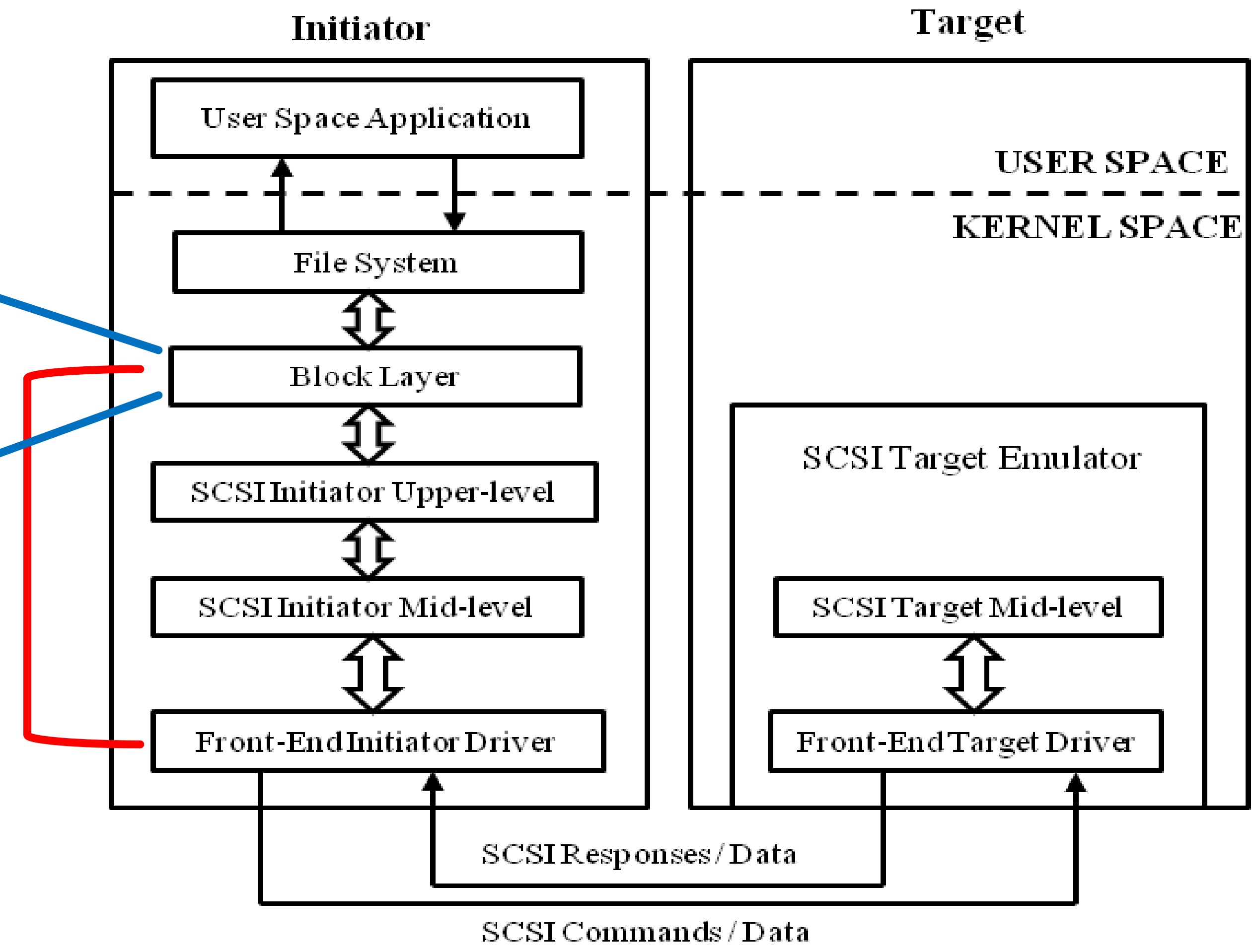
Traditional SAN I/O Path in the Linux

Elevator for request merge

Plug-Unplug mechanism

Cause some delays

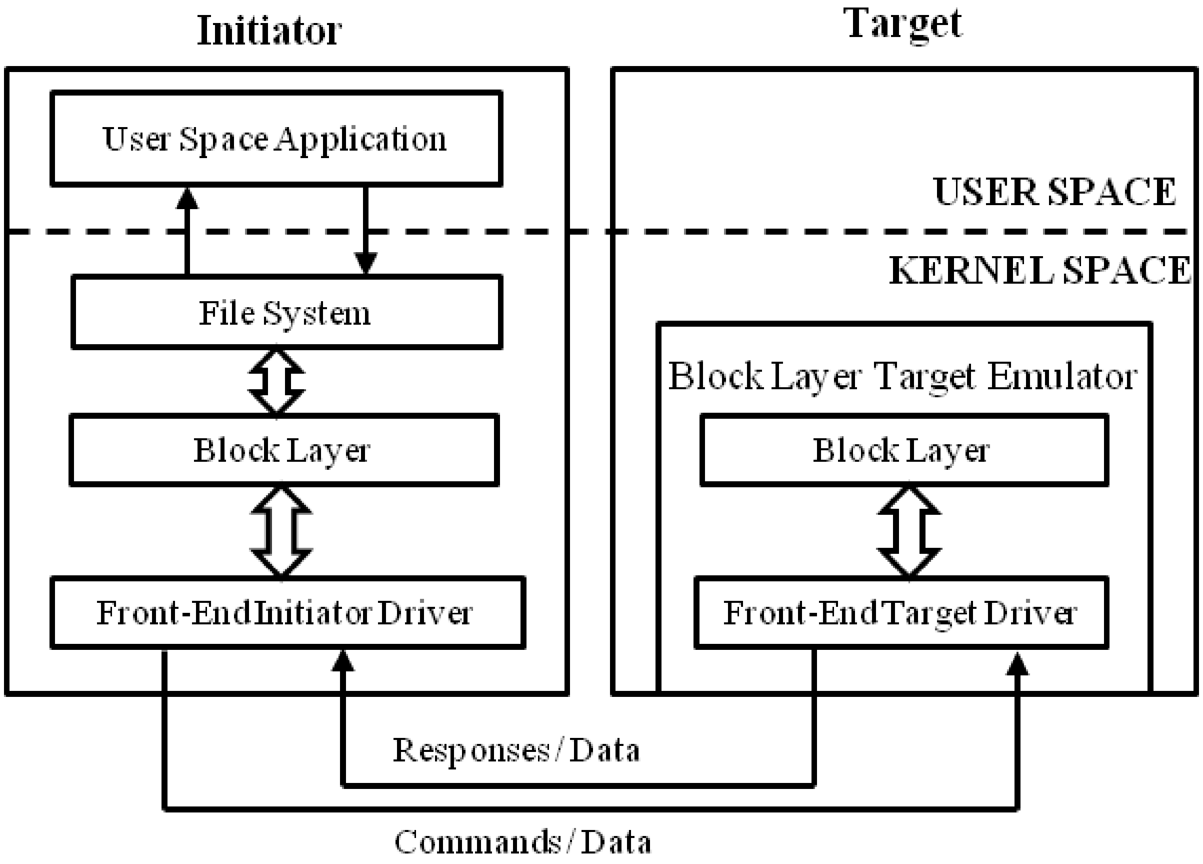
Too long



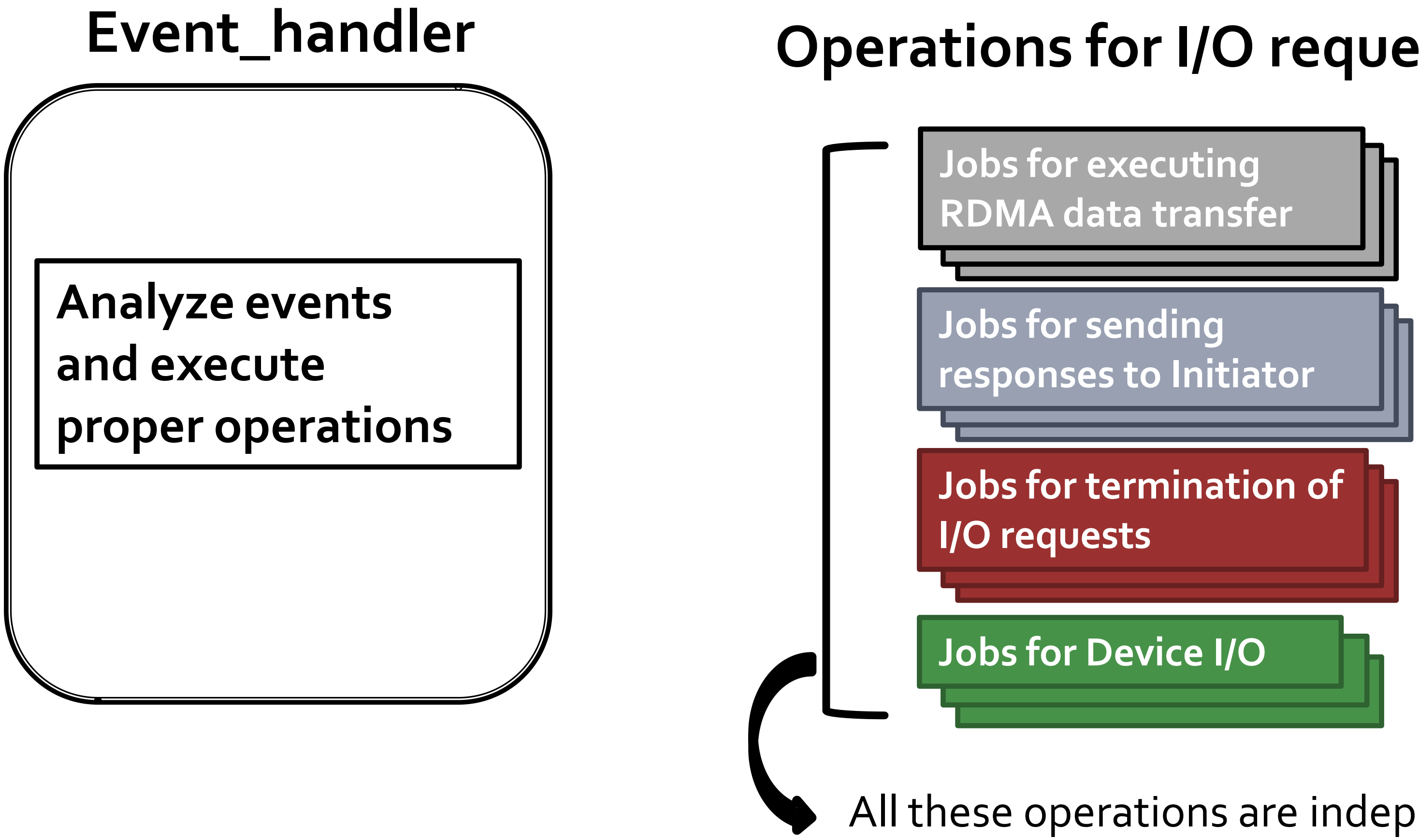
Optimization 1

- Remove software overheads in I/O path
  - Bypass SCSI layer
  - Discard existing I/O scheduler
    - Remove elevator merge and plug-unplug
    - Maintain wait-queue based on *bio* structure
    - Very simple & fast I/O scheduler
- BRP(Block RDMA Protocol)
  - Commands are also based on *bio* structure, not scsi command

Optimization 1



Operations on Target side



Event_handler

Analyze events and execute proper operations

Operations for I/O request

Jobs for executing RDMA data transfer

Jobs for sending responses to Initiator

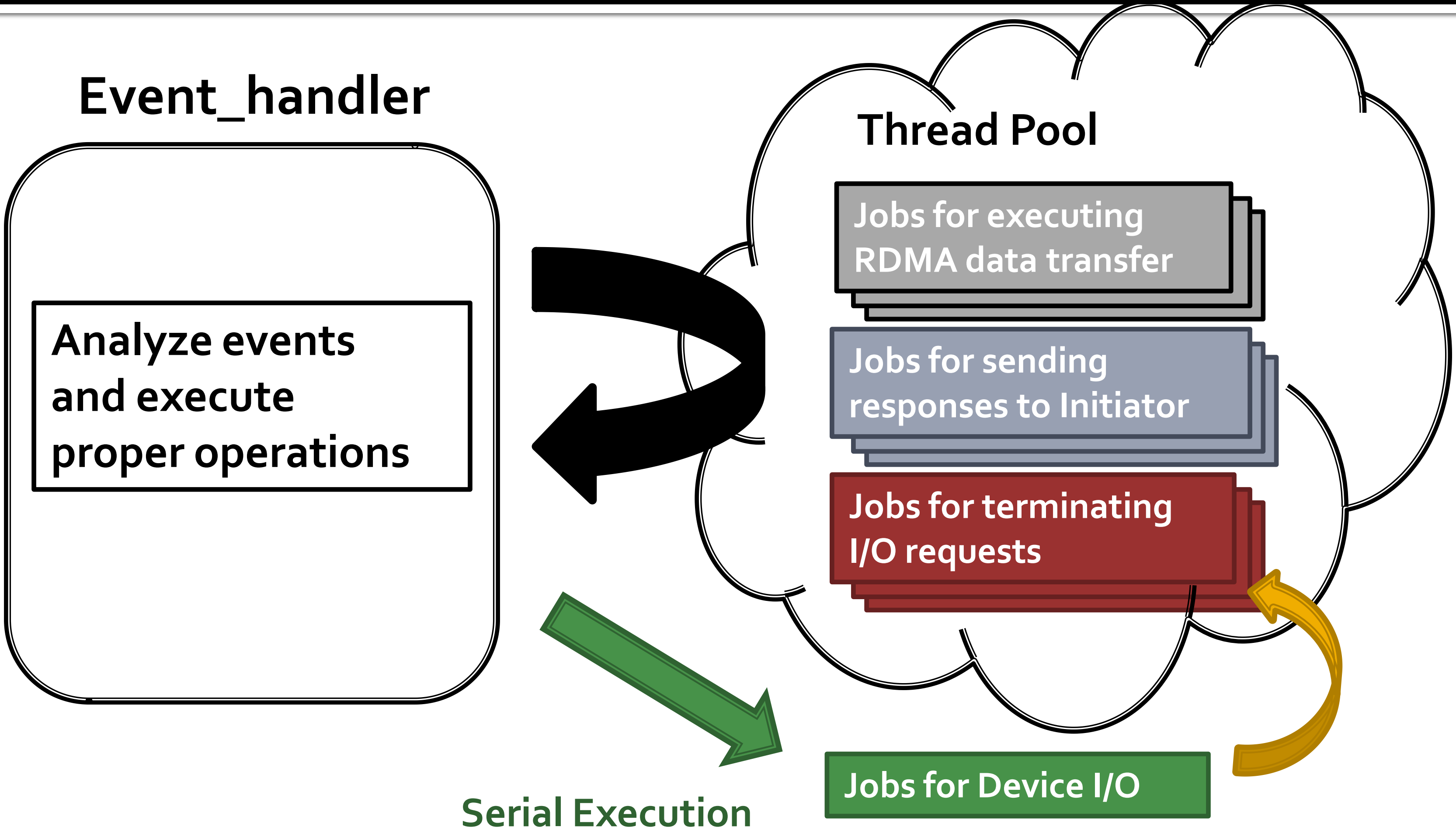
Jobs for termination of I/O requests

Jobs for Device I/O

All these operations are independent each other

can be processed in parallel

Operations on Target side



Event_handler

Analyze events and execute proper operations

Thread Pool

Jobs for executing RDMA data transfer

Jobs for sending responses to Initiator

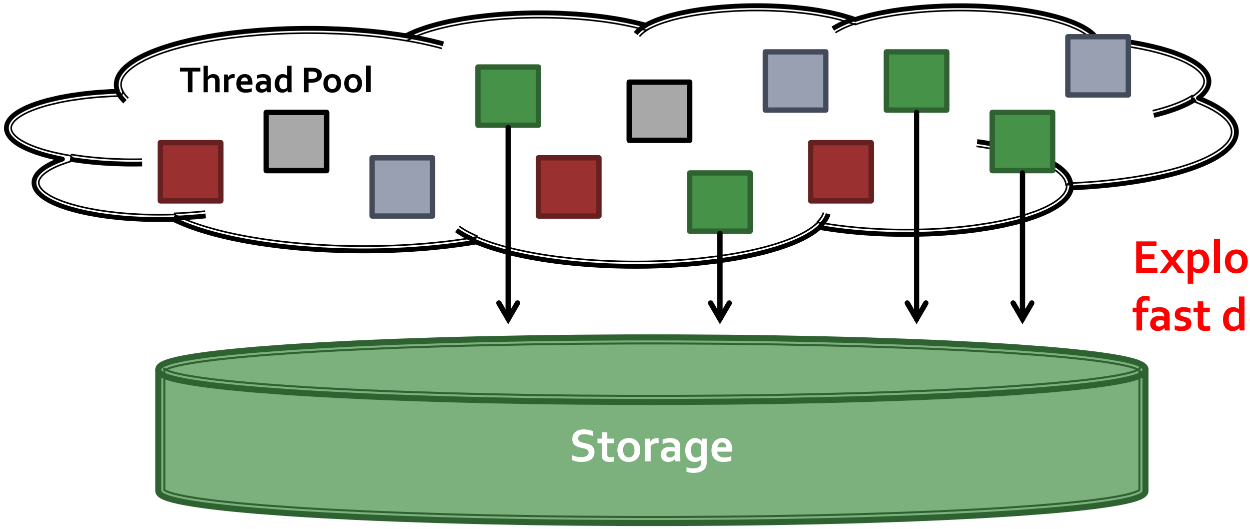
Jobs for terminating I/O requests

Jobs for Device I/O

Serial Execution

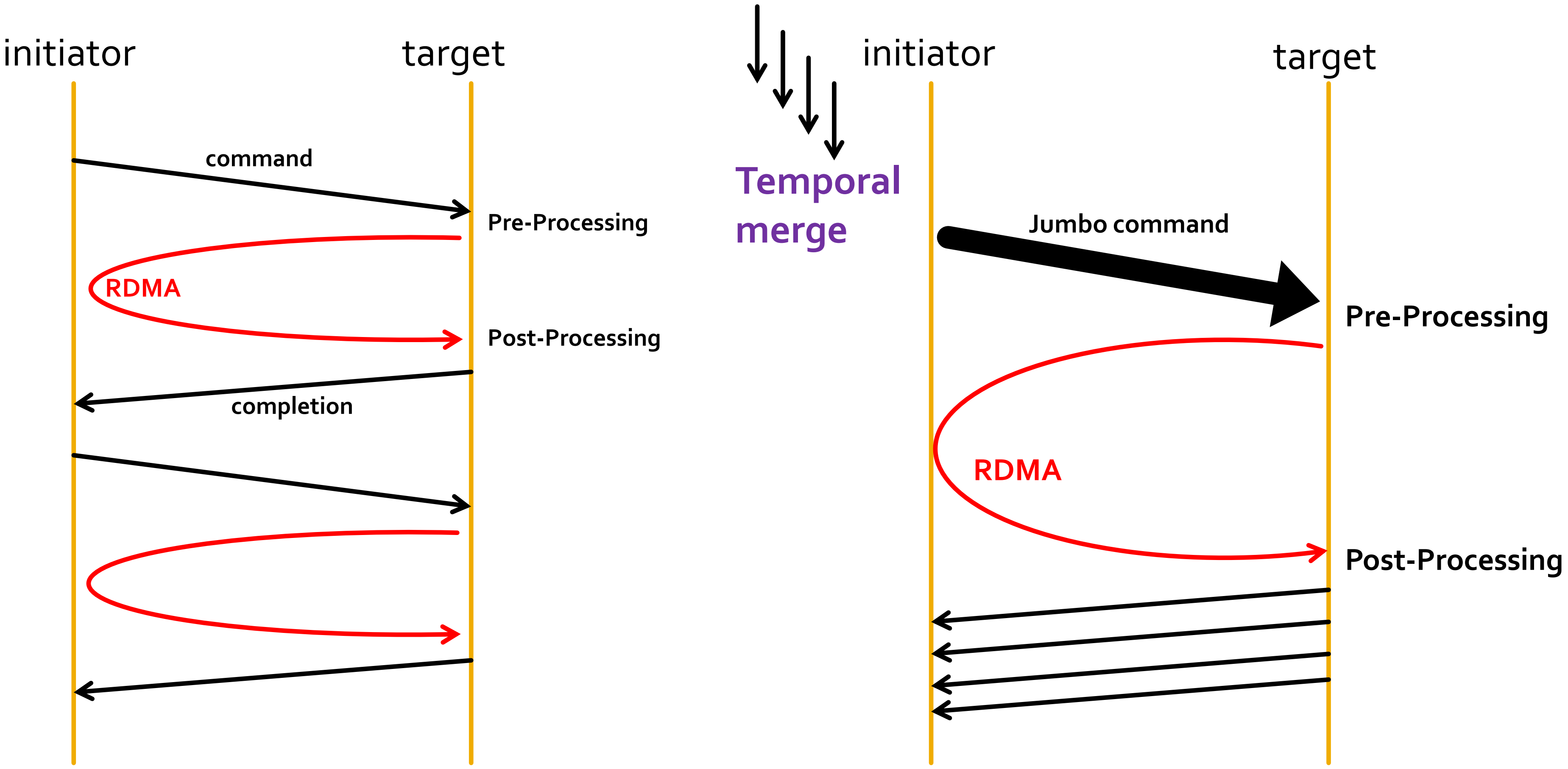
Optimization 2

- Increase Parallelism on the Target side
  - All the procedures for I/O requests are processed in thread-pool
    - Induce Multiple device I/O



Exploit high bandwidth of fast device

RDMA Data Transfer



initiator

target

command

Pre-Processing

RDMA

Post-Processing

completion

Temporal merge

initiator

target

Jumbo command

Pre-Processing

RDMA

Post-Processing

Optimization 3

- RDMA data transfer with temporal merge
  - Merge small sized data regardless of its spatial continuance
  - Enabled at the only intensive-I/O situation

Evaluation

- BRP-1
  - Remove software overhead in I/O path
- BRP-2
  - BRP-1 + Increase Parallelism
- BRP-3
  - BRP2 + Temporal Merge at the intensive I/O situation
  - Just BRP means BRP-3

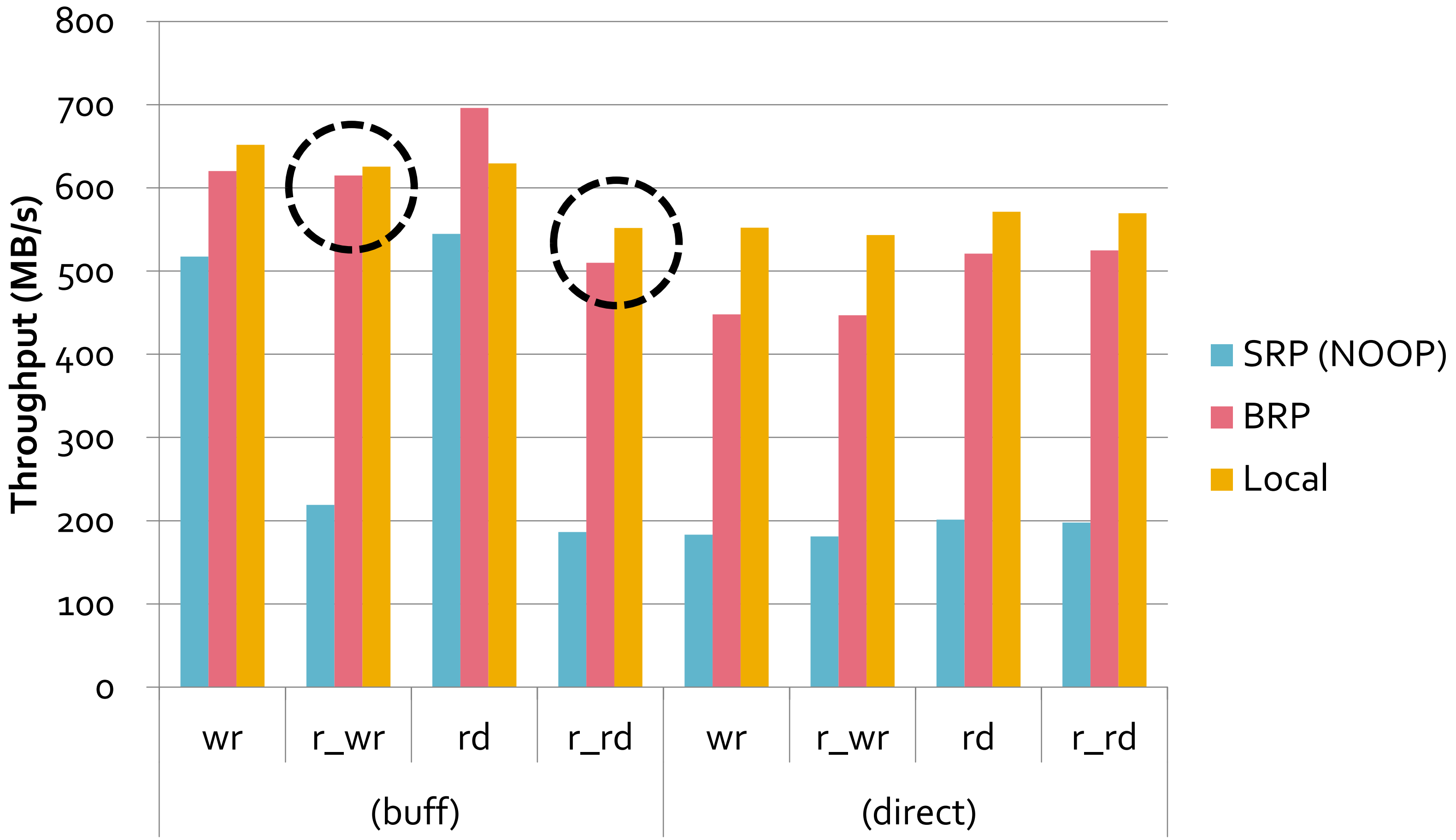
Evaluation

- Latency comparison
  - Direct I/O, 4KB
  - dd test

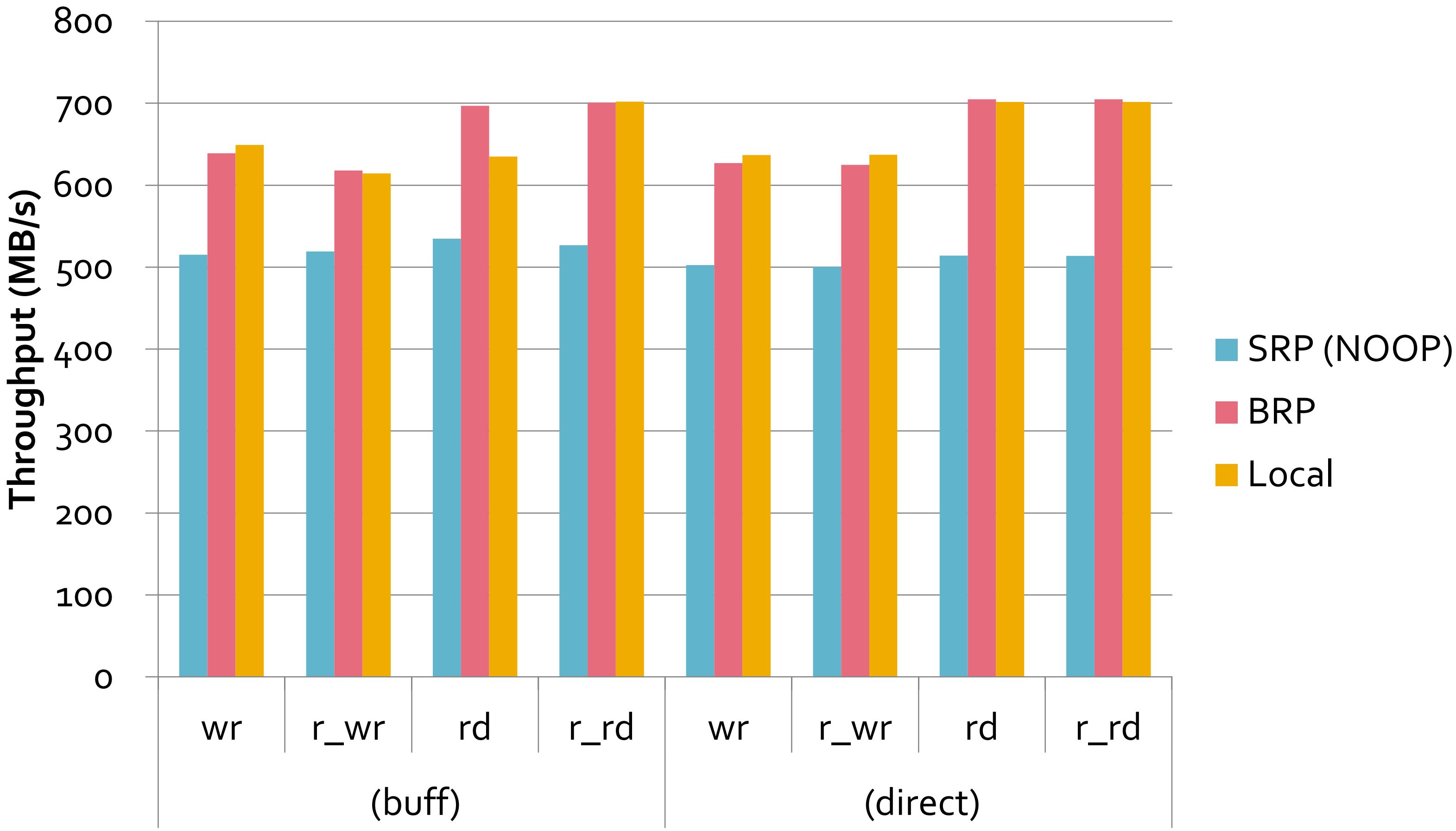
I/O Type	SRP(usec)	BRP(usec)	Latency Reduction
Read	63 (51)	43 (31)	-31.7 (-39.2) %
Write	75 (62)	54 (41)	-28 (-33.8)%

( ) : the value excepting device I/O latency
read-12usec, write-13usec

Evaluation (small size: 4KB)

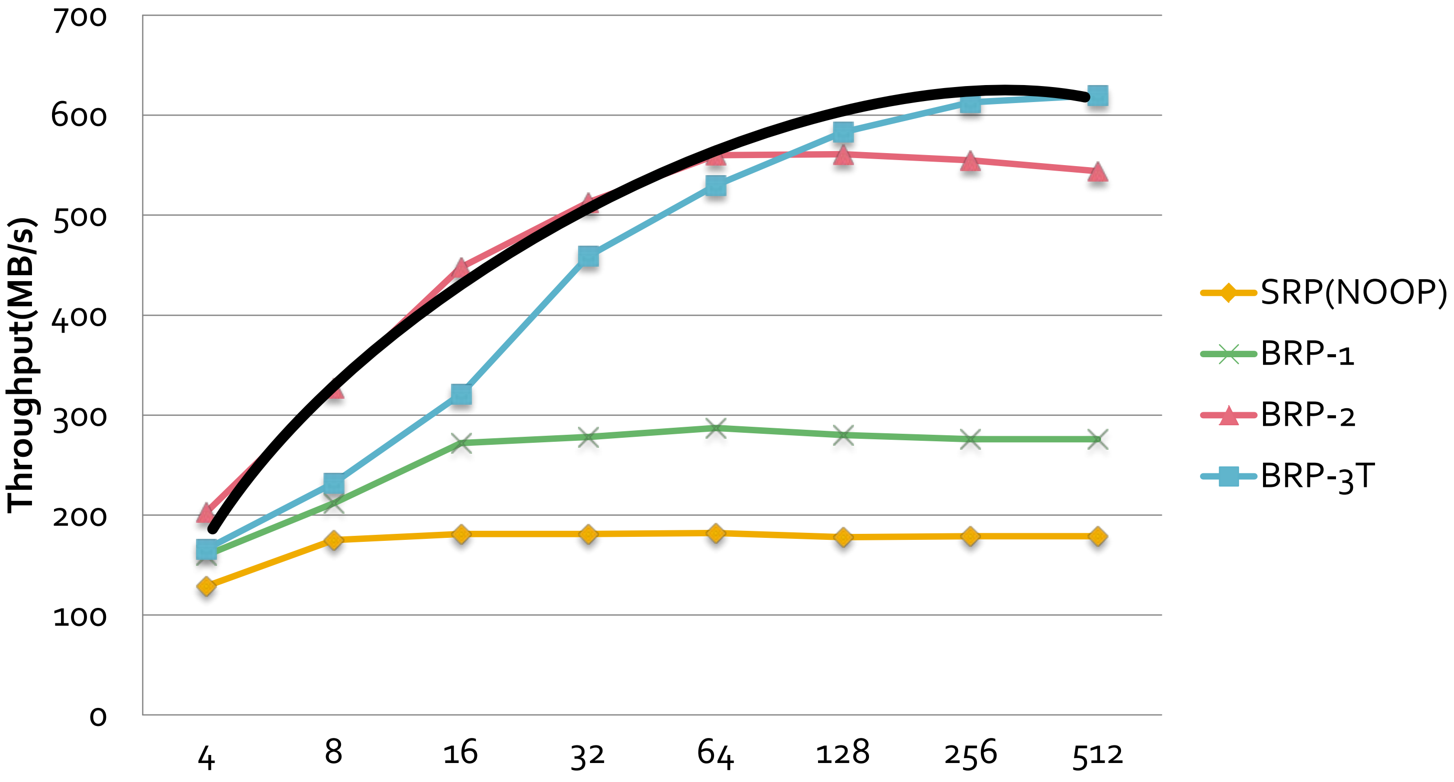


Evaluation (large size: 1MB)



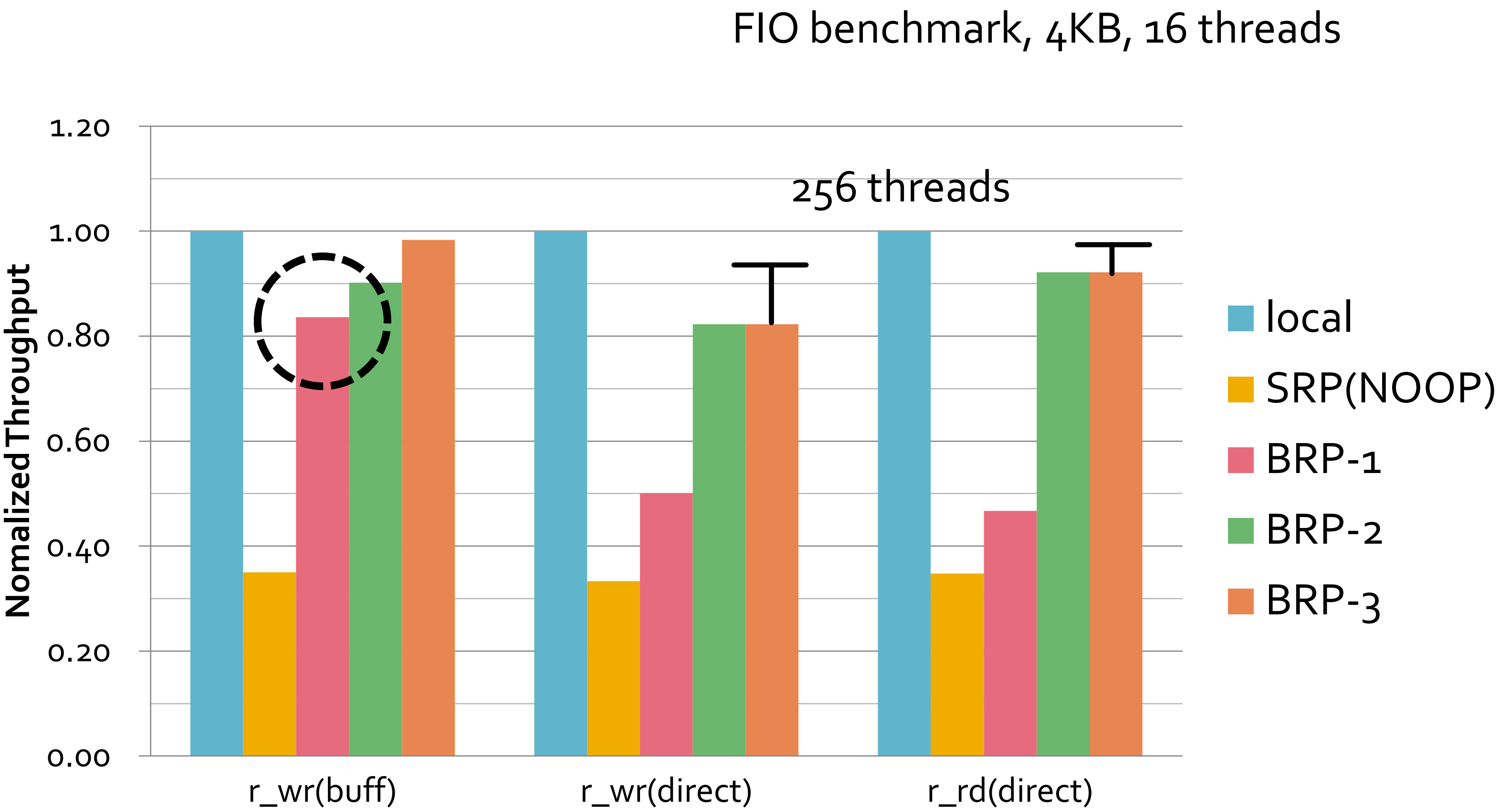
Evaluation

FIO benchmark, random write, 4KB, direct I/O,



BRP-3T: always executes temporal merge

Evaluation


FIO benchmark, 4KB, 16 threads
256 threads
Nomalized Throughput
1.20
1.00
0.80
0.60
0.40
0.20
0.00
r_wr(buff)
r_wr(direct)
r_rd(direct)
local
SRP(NOOP)
BRP-1
BRP-2
BRP-3

Conclusion

- SAN with high performance storage
- Propose new SAN solution
  - Remove Software overheads in I/O path
  - Increase parallelism on the Target side
  - Temporal merge for RDMA data transfer
- Implement the optimized SAN as a prototype

Thank you !

QnA?