# OpenSolaris iSCSI Extensions for RDMA (iSER)

Peter Dunlap

Staff Engineer, Sun Microsystems

# Agenda



- iSER Overview
  - Protocol
  - Hardware Requirements
- Datamover Architecture
- OpenSolaris iSER project
  - Initiator
  - Target
- Performance

# ISER Overview



- RFC 5046
- Accelerate iSCSI data transfers by replacing TCP/IP transport with RDMA transfers
  - Initiator and Target negotiate iSER support during the login phase using TCP/IP
  - If both Initiator and Target agree to use iSER the connection is switched to "iSER-assisted" mode
  - Subsequent communication for that session takes place using RDMA operations

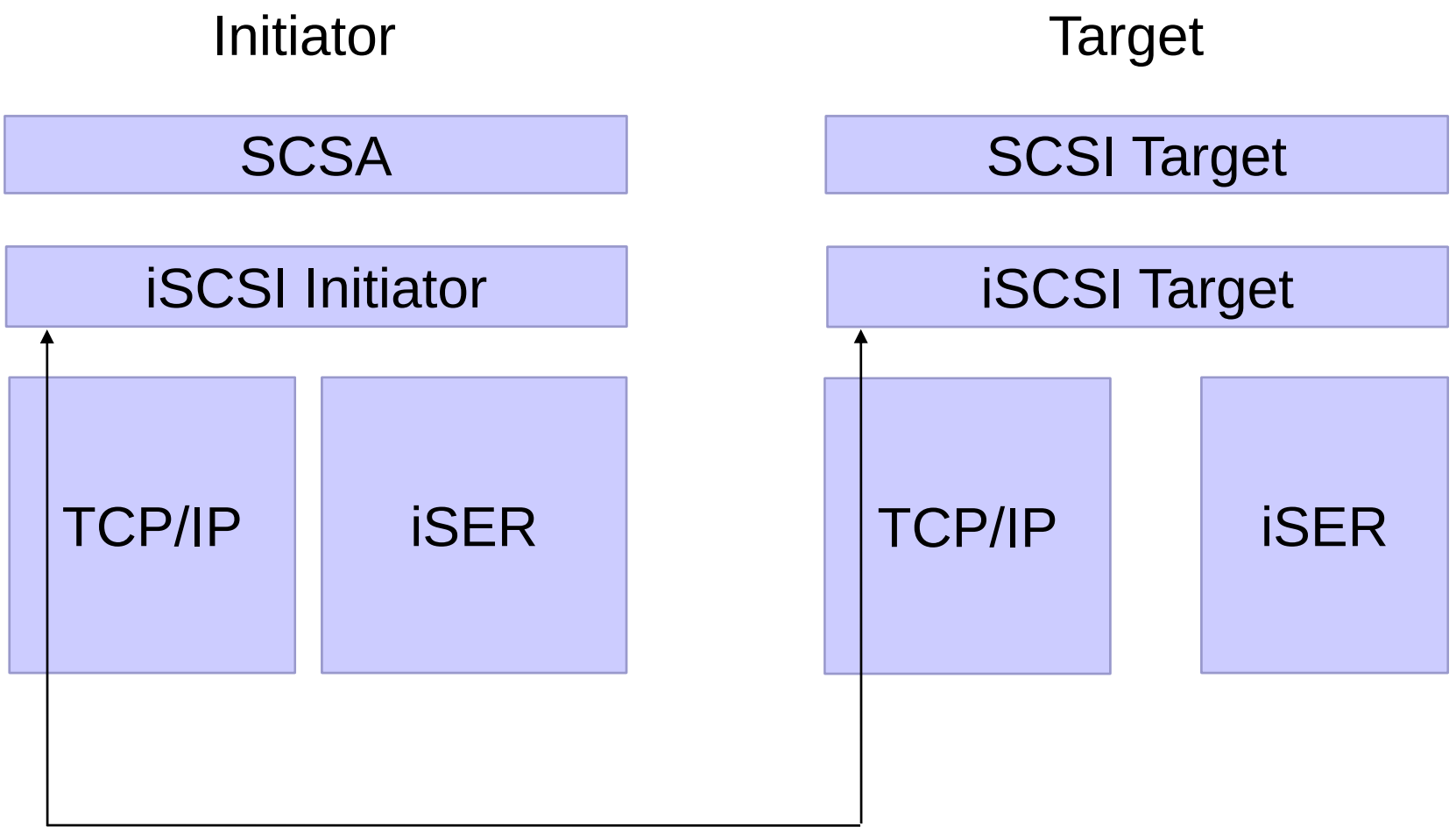
# What is RDMA?



- Remote Direct Memory Access
  - Local memory is mapped into a remote systems address space allowing the remote system to read and write local memory buffers
    - Remote systems only have access to the memory the local system has mapped
  - RDMA Primitives
    - Send
    - RDMA Read
    - RDMA Write

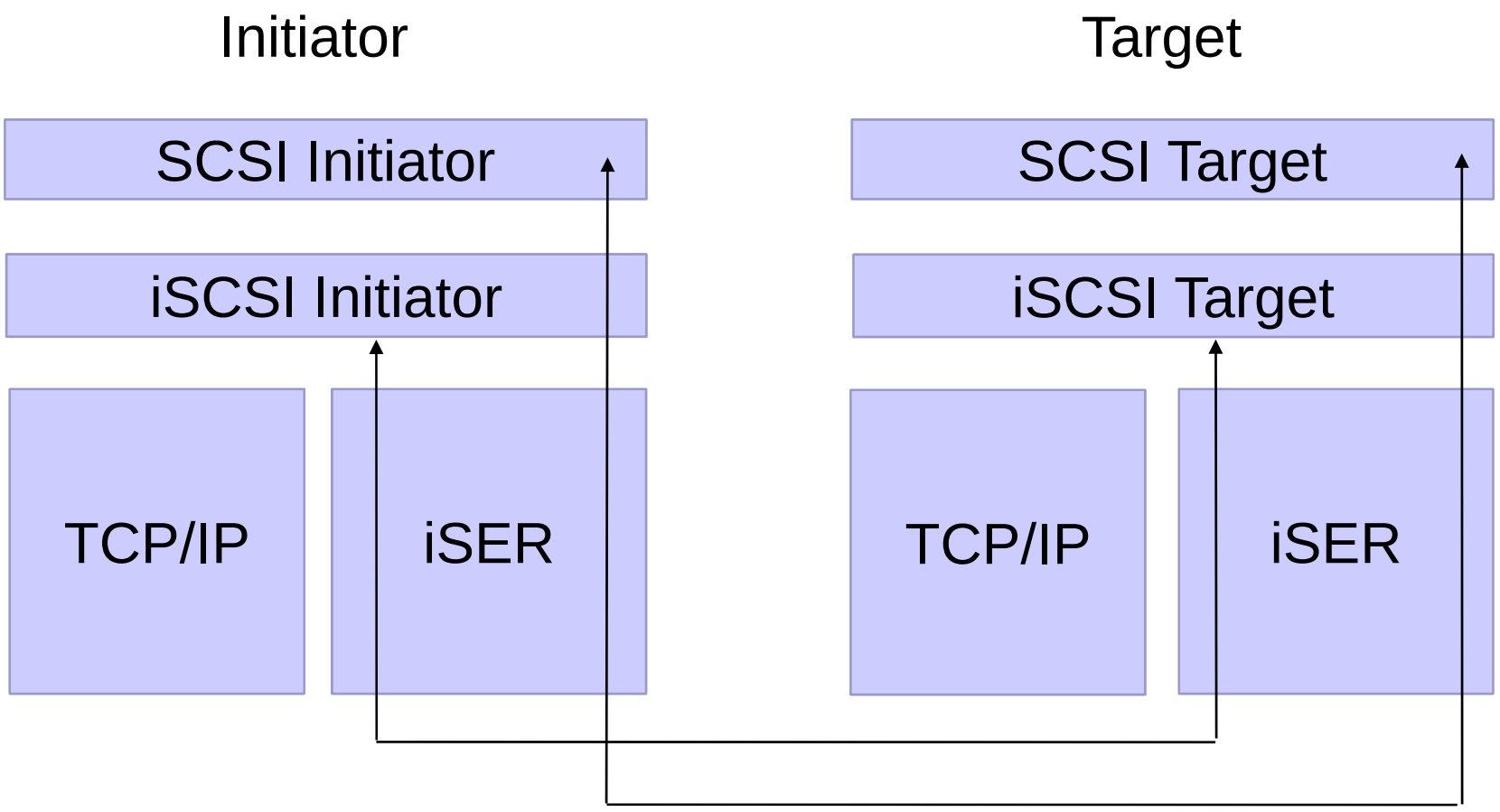
# iSER Login Phase





# iSER Full Feature Phase





# ISER Overview (continued)



- ISER works with any RDMA capable protocol (RcaP)
  - iWarp
    - Effectively requires RDMA capable NIC (RNIC)
      - Intelligent NIC with TCP/IP offload engine that understands the iWarp protocol
  - Infiniband
    - RDMA is designed into Infiniband networks
    - Host Channel Adapter (HCA)
    - Target Channel Adapter (TCA)

# ISER Overview (continued)



- Leverages existing iSCSI infrastructure
  - Device discovery
  - Naming
  - Management
  - SRP lacks naming and management functionality
- iSER allows the use of iSCSI mechanism to discovery and manage Infiniband storage targets

# FC vs. iSCSI



- FC is currently a dominant SAN technology
  - High bandwidth, low CPU utilization
  - Hardware handles most of the data transfers
    - This is so taken for granted that it is not even called "offload"
      - But it is...
      - One interrupt per I/O, host CPU is not in the data path
  - Expensive hardware

# FC vs. iSCSI



- iSCSI runs on inexpensive commodity hardware
- Capable of running just as fast as FC
  - Um, there are caveats though
    - Large CPU and memory cost for a software-only iSCSI implementation or....
    - Use a TCP/IP offload engine that implements iSCSI
      - Expensive
      - Vendor specific drivers and software stacks
      - Sounds like FC
  - Software-only iSCSI is an excellent fit for environments with plenty of spare CPU capacity

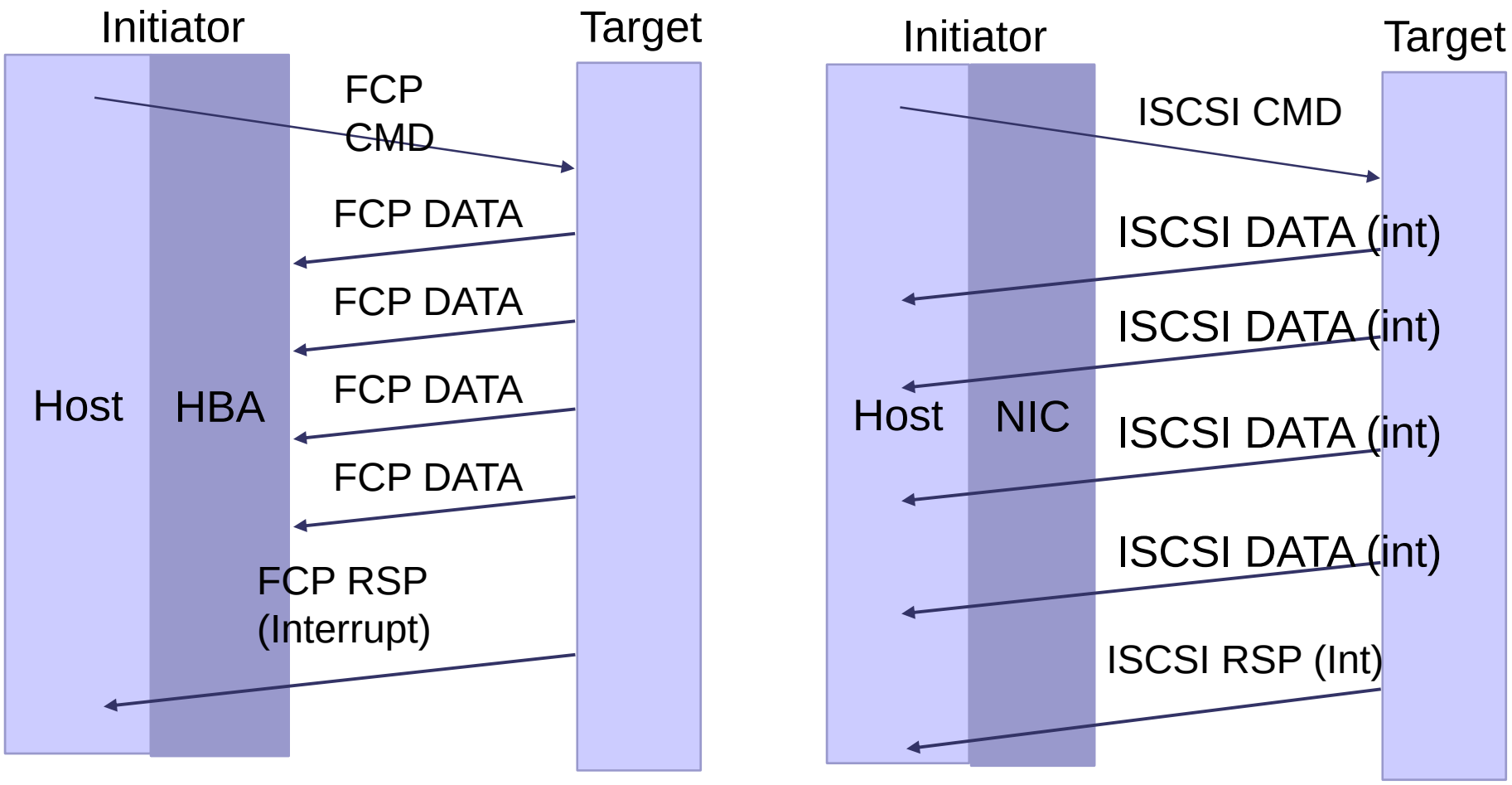
# FC vs. iSCSI



- At the wire level the FC and iSCSI protocols are not significantly different
  - Command Phase
  - Data Phase
  - Status Phase
- iSER uses an existing mechanism (RDMA) to implement the CPU intensive portions of iSCSI in hardware
  - Yes, here we are with the specialized hardware again

# FC vs. iSCSI Reads

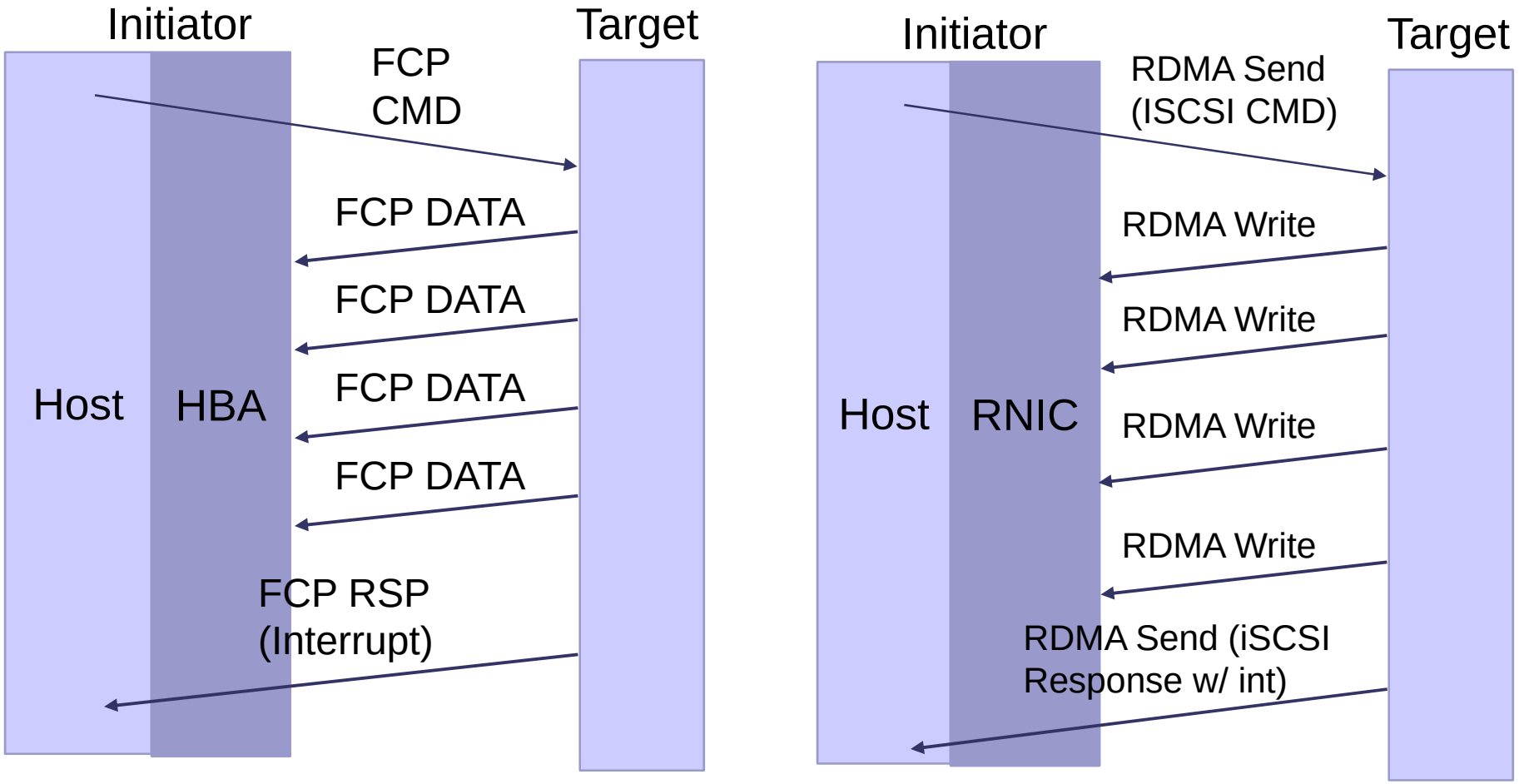





Most of the transfer on the FC host is handled within the HBA. The iSCSI host manages the entire transfer and (among other things) must handle several additional interrupts.

# FC vs. iSER Reads

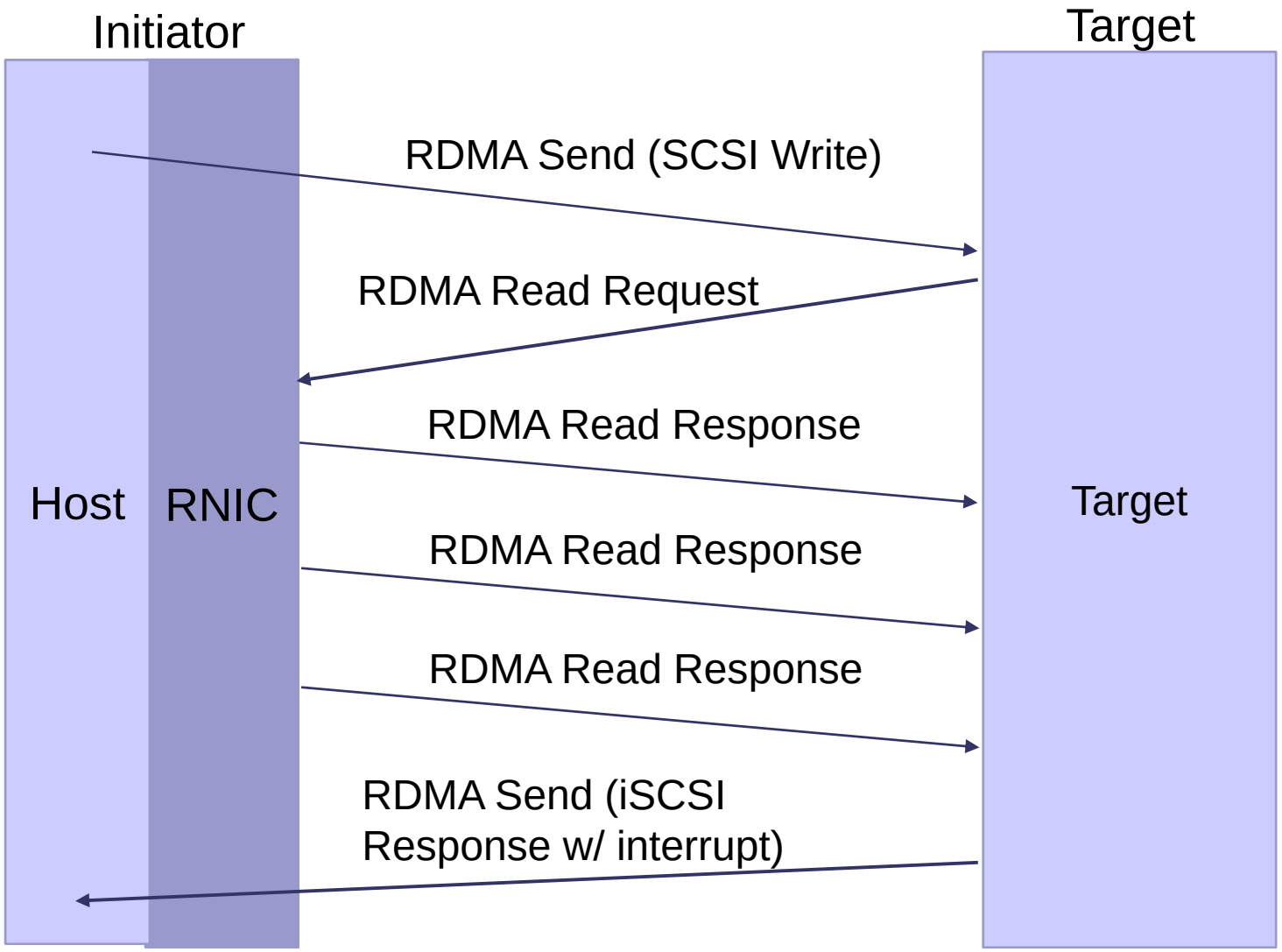





RDMA offloads the data processing from the host. As an added bonus the iSCSI command and responses are sent using “RDMA Send” which also bypasses the host TCP/IP stack.

# iSER Writes





# RNIC TOE vs. iSCSI TOE



- So if we need specialized hardware either way, why is an RNIC TOE better than an iSCSI TOE?
  - RDMA allows TOEs to support iSCSI with less onboard memory for IP fragment reassembly
    - Reduced component cost for the NIC
  - RDMA is useful outside of iSER so an RNIC TOE is less specialized than a non-RNIC iSCSI TOE
    - SDP, NFS, MPI
    - "Less specialized" is code for "cheaper", at least in theory

# Infiniband



- RDMA one of the architectural foundations of Infiniband and is available on all Infiniband networks
  - All HCAs provide RDMA services and can be used for iSER
  - HCAs are inexpensive compared to 10Gb Ethernet
    - At least for the time being
    - HCAs are also available in 20Gb/s and 40Gb/s
  - Consolidate multiple protocols on a single cable or redundant pair of cables

# OpenSolaris iSER



- Open source project
  - Design discussions and code reviews all take place on iser-dev@opensolaris.org
  - http://www.opensolaris.org/os/project/iser/
    - Design documents
    - Source code tarballs
    - Binary packages
    - Subscribe to iser-dev@opensolaris.org and participate

# OpenSolaris iSER



- Based on Datamover Architecture RFC 5047
- Modify existing OpenSolaris iSCSI initiator to allow multiple transports
- Implement iSER target plugin for COMSTAR
  - **CO**mmon **M**ulti-protocol **S**CSI **TAR**get
  - http://www.opensolaris.org/os/project/comstar/
  - Peer transport to the existing FC support
  - Also provides COMSTAR iSCSI support

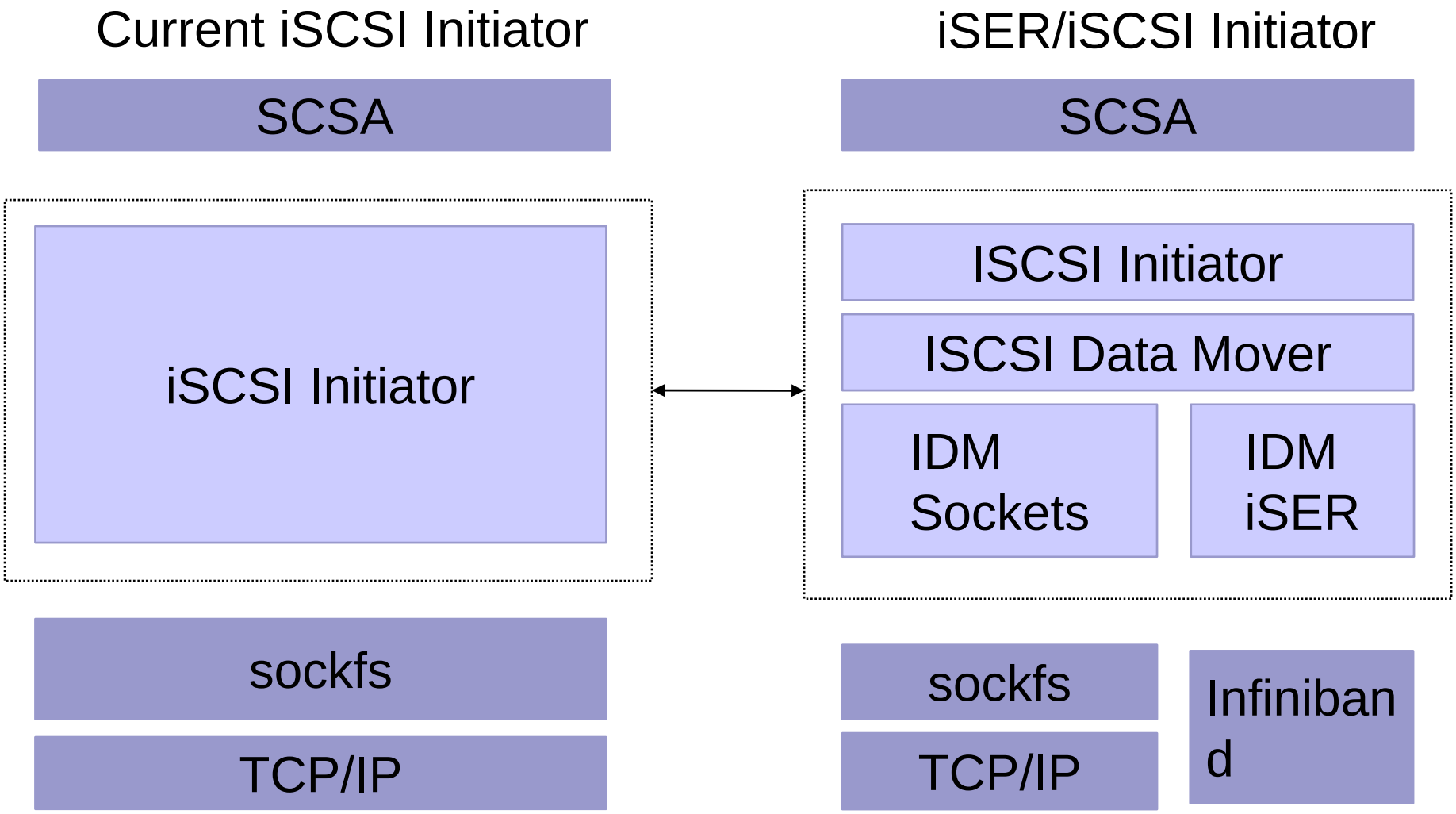
# iSCSI Data Mover Architecture



- RFC 5047 defines an abstraction layer called "Datamover Architecture" (or DA for short)
  - Separates data movement from the rest of the iSCSI protocol.
  - Organizes iSCSI protocol operations into DA primitives
  - RFC explicitly allows implementation flexibity
    - After all, how do you test conformance to an architectural RFC?

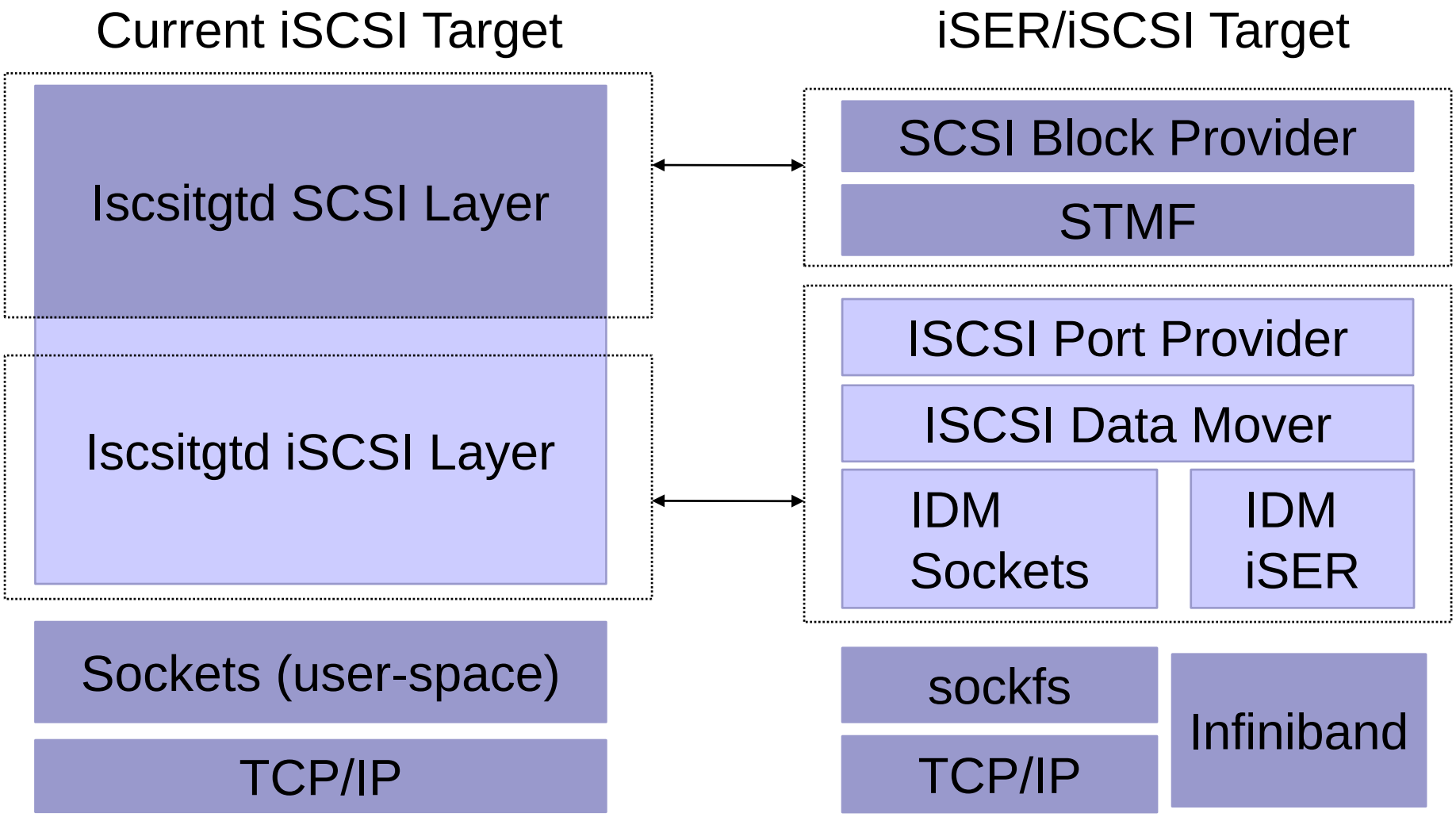
# OpenSolaris iSER/iSCSI Initiator





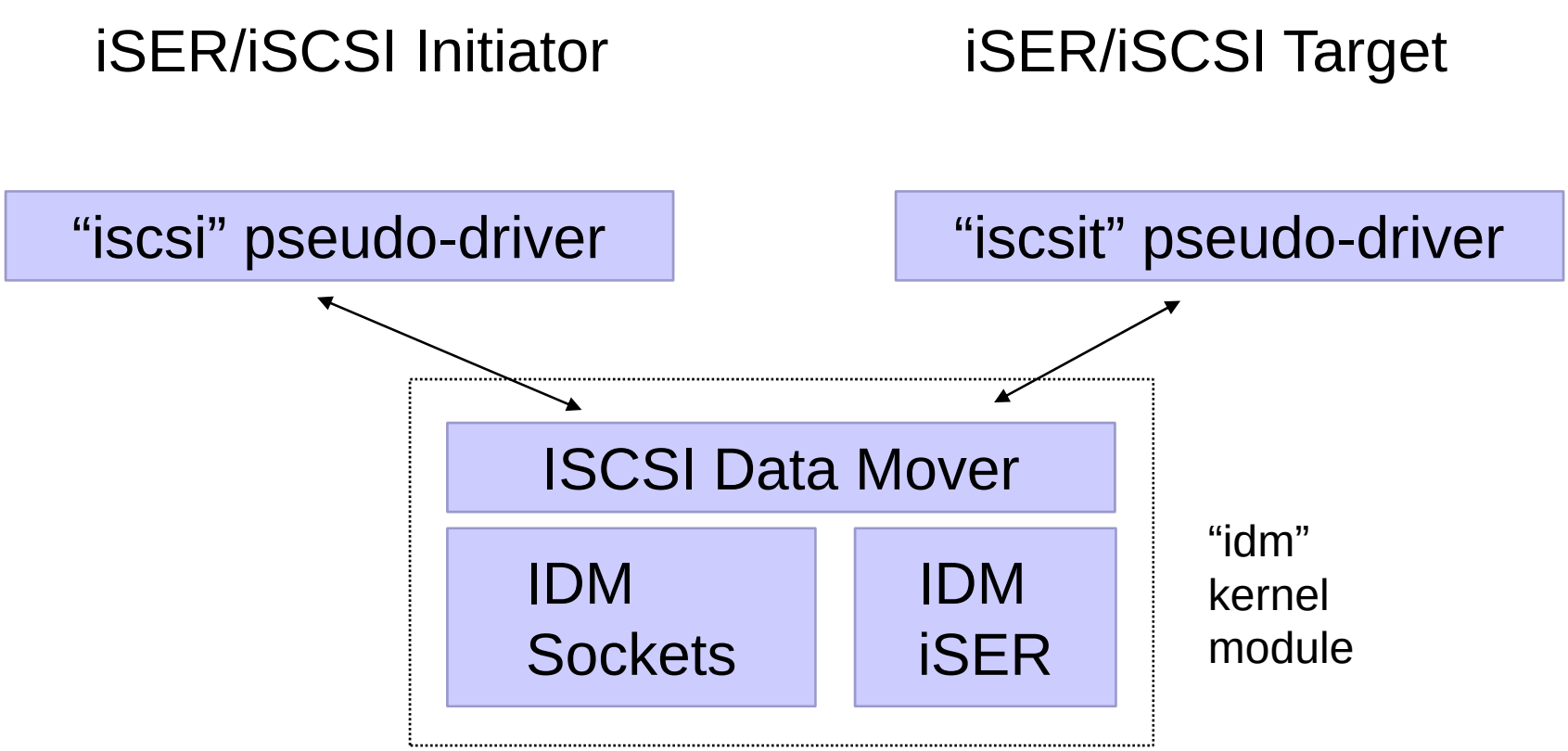
# OpenSolaris iSER/iSCSI Target





# ISCSI Data Mover is shared





# Sharing Initiator and Target Code



- This may not make sense for every iSER implementation but it made sense for us
  - The scope of project was initiator and target from the beginning
  - We didn't want to develop two separate iSER transport modules
    - The transport code only rarely cares whether it is moving initiator or target data
      - Primary differences are around the data transfer code
        - Initiator and targets have different tasks related to RDMA

# Sharing initiator and Target Code



- Socket transport code is also shared
  - Algorithm to read a received iSCSI PDU from the socket is identical
  - Initiator SCSI Write transfers and Target SCSI Read transfers use the same function to transmit a burst of SCSI data.
- MDB dcmds work for both initiator and target

# Sharing initiator and Target Code



- Any benefits to the end user?
  - Yes, but indirect
    - Few lines of code results in better test coverage
    - Optimization in the shared code improves both initiator and target performance

# Sharing Initiator and Target Code



- Downsides to code sharing
  - It's harder to evaluate the impact of any given change in the shared code
    - When tracking down a bug it's natural to consider the fix from the perspective of the environment in which it occured.
    - Similarly time constraints sometimes tempt developers to only test the environment on which are focused
    - Automated regression tests
    - Careful code reviews

# Performance



- Unfortunately at this time we have not had a chance to do detailed performance testing.
- The numbers below reflect a workload of 512k reads with 100% cache hits on a ZFS zvol backing store (one client)
- Best iSER number: 1390MB/s
  - 28% CPU utilization
- Best iSCSI number: 590MB/s
  - 85% CPU utilization
  - Writes are better... 850MB/s w/ 75% CPU util.