



Porting Portals to OFED

Ron Brightwell
Scalable System Software
Sandia National Laboratories
Albuquerque, NM, USA

Open Fabrics Alliance International Workshop
Monterey, CA

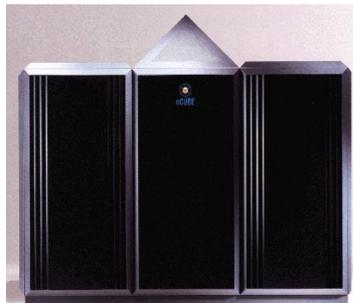
April 4, 2011

*Sandia is a Multiprogram Laboratory Operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy Under Contract DE-ACO4-94AL85000.*

Sandia Massively Parallel Systems

2004

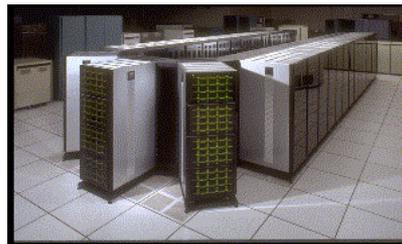
1990



nCUBE2

- Sandia's first large MPP
- Achieved Gflops performance on applications

1993



Paragon

- Tens of users
- First periods processing MPP
- World record performance
- Routine 3D simulations
- SUNMOS lightweight kernel

1997



ASCI Red

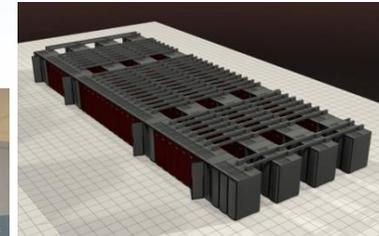
- Production MPP
- Hundreds of users
- Red & Black partitions
- Improved interconnect
- High-fidelity coupled 3-D physics
- Puma/Cougar lightweight kernel

1999



Cplant

- Commodity-based supercomputer
- Hundreds of users
- Enhanced simulation capacity
- Linux-based OS licensed for commercialization
- ~2000 nodes



Red Storm

- Prototype Cray XT
- Custom interconnect
- Purpose built RAS
- Highly balanced and scalable
- Catamount lightweight kernel
- Currently 38,400 cores (quad & dual)

Portals Network Programming Interface

- Network API developed by Sandia, U. New Mexico, Intel
- Previous generations of Portals deployed on several production massively parallel systems
 - 1993: 1800-node Intel Paragon (SUNMOS)
 - 1997: 10,000-node Intel ASCI Red (Puma/Cougar)
 - 1999: 1800-node Cplant cluster (Linux)
 - 2005: 10,000-node Cray Sandia Red Storm (Catamount)
 - 2009: 18,688-node Cray XT5 – ORNL Jaguar (Linux)
- Focused on providing
 - Lightweight “connectionless” model for MPP systems
 - Low latency
 - High bandwidth
 - Independent progress
 - Overlap of computation and communication
 - Scalable buffering semantics
- Supports MPI, Cray SHMEM, ARMCI, GASNet, Lustre, etc.

What Makes Portals Different?

- One-sided communication with optional matching
- Provides elementary building blocks for supporting higher-level protocols well
- Allows structures to be placed in user-space, kernel-space, or NIC-space
- Allows for zero-copy, OS-bypass, and application-bypass implementations
- Scalable buffering of MPI unexpected messages
- Supports multiple protocols within a process
- Runtime system independent
- Well-defined failure semantics

Portals 4.0:

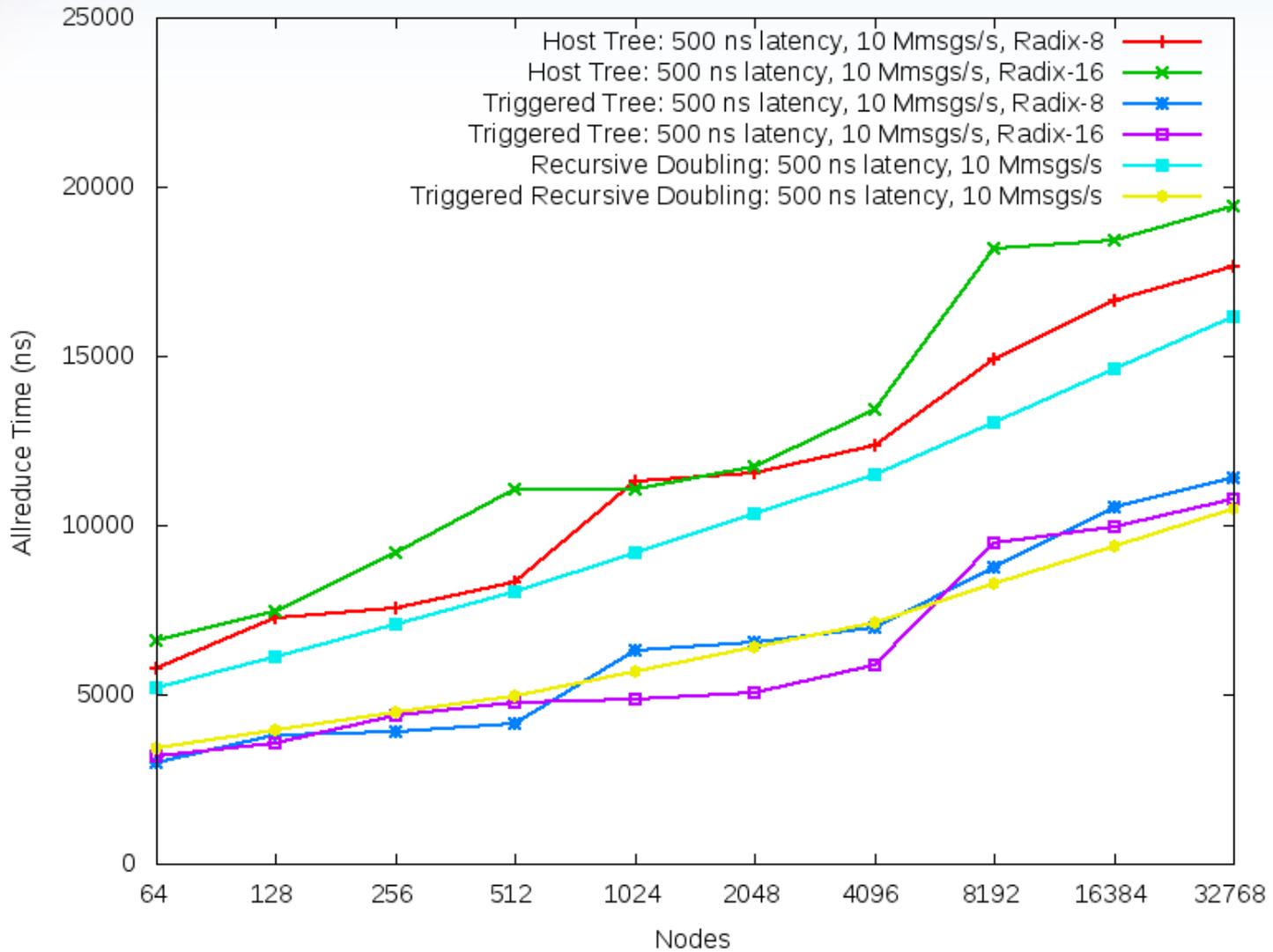
Applying Lessons Learned from Cray SeaStar

- High message rate
 - Atomic search and post for MPI receives required round-trip across PCI
 - Eliminate round-trip by having Portals manage unexpected messages
- Flow control
 - Encourages well-behaved applications
 - Fail fast
 - Identify application scalability issues early
 - Resource exhaustion caused unrecoverable failure
 - Recovery doesn't have to be fast
 - Resource exhaustion will disable Portal
 - Subsequent messages will fail with event notification at initiator
 - Applies back pressure from network
 - Performance for scalable applications
 - Correctness for non-scalable applications

Portals 4.0 (cont'd)

- Hardware implementation
 - Designed for intelligent or programmable NICs
 - Arbitrary list insertion is bad
 - Remove unneeded symmetry on initiator and target objects
- New functionality for one-sided operations
 - Eliminate matching information
 - Smaller network header
 - Minimize processing at target
 - Scalable event delivery
 - Lightweight counting events
 - Triggered operations
 - Chain sequence of data movement operations
 - Build asynchronous collective operations
 - Mitigate OS noise effects

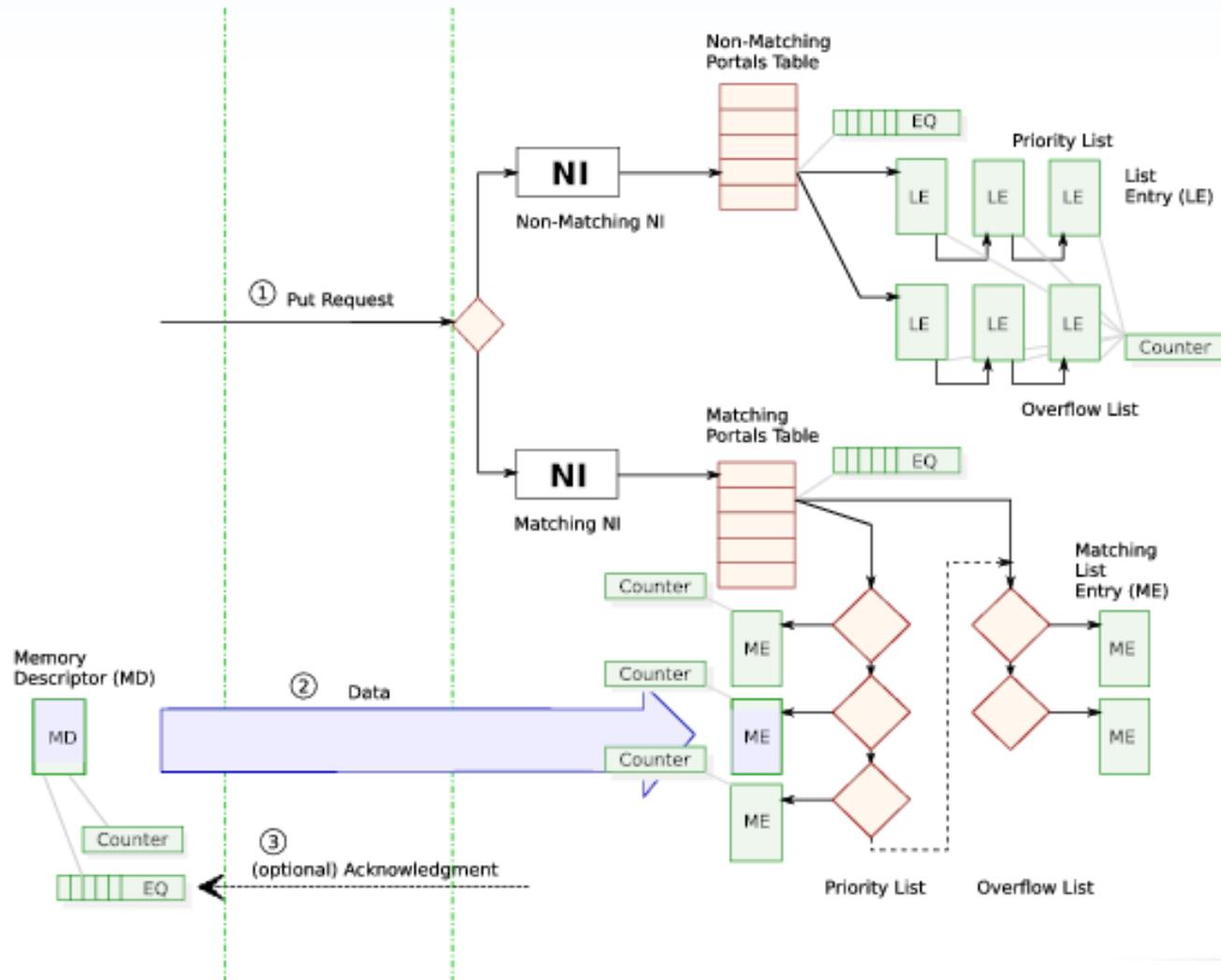
Triggered Operations



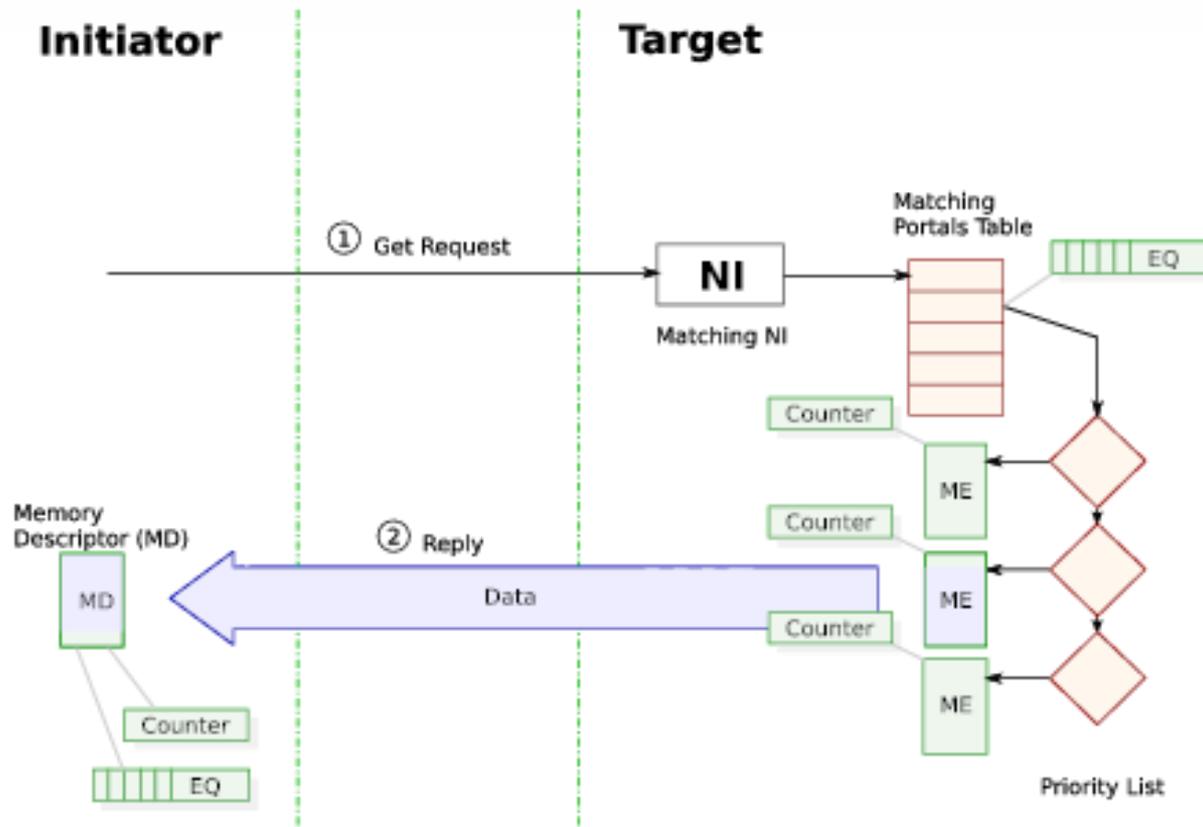
Put Operation

Initiator

Target



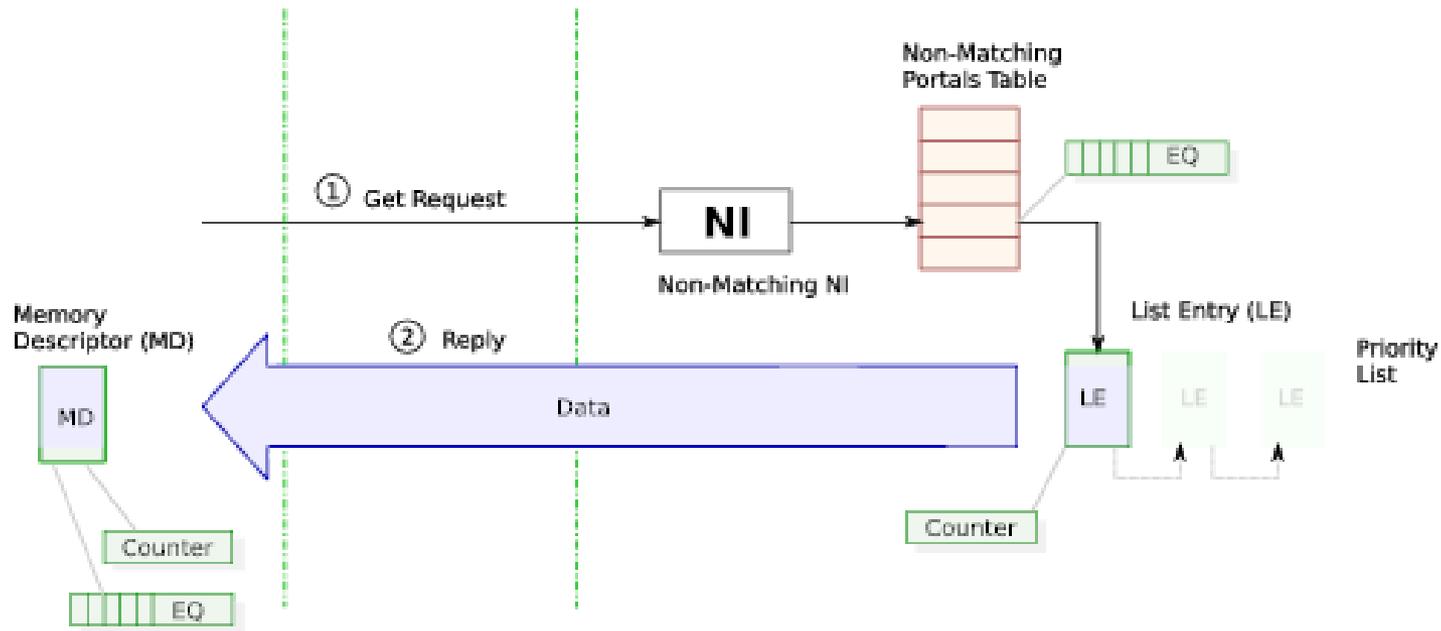
Matched Get Operation



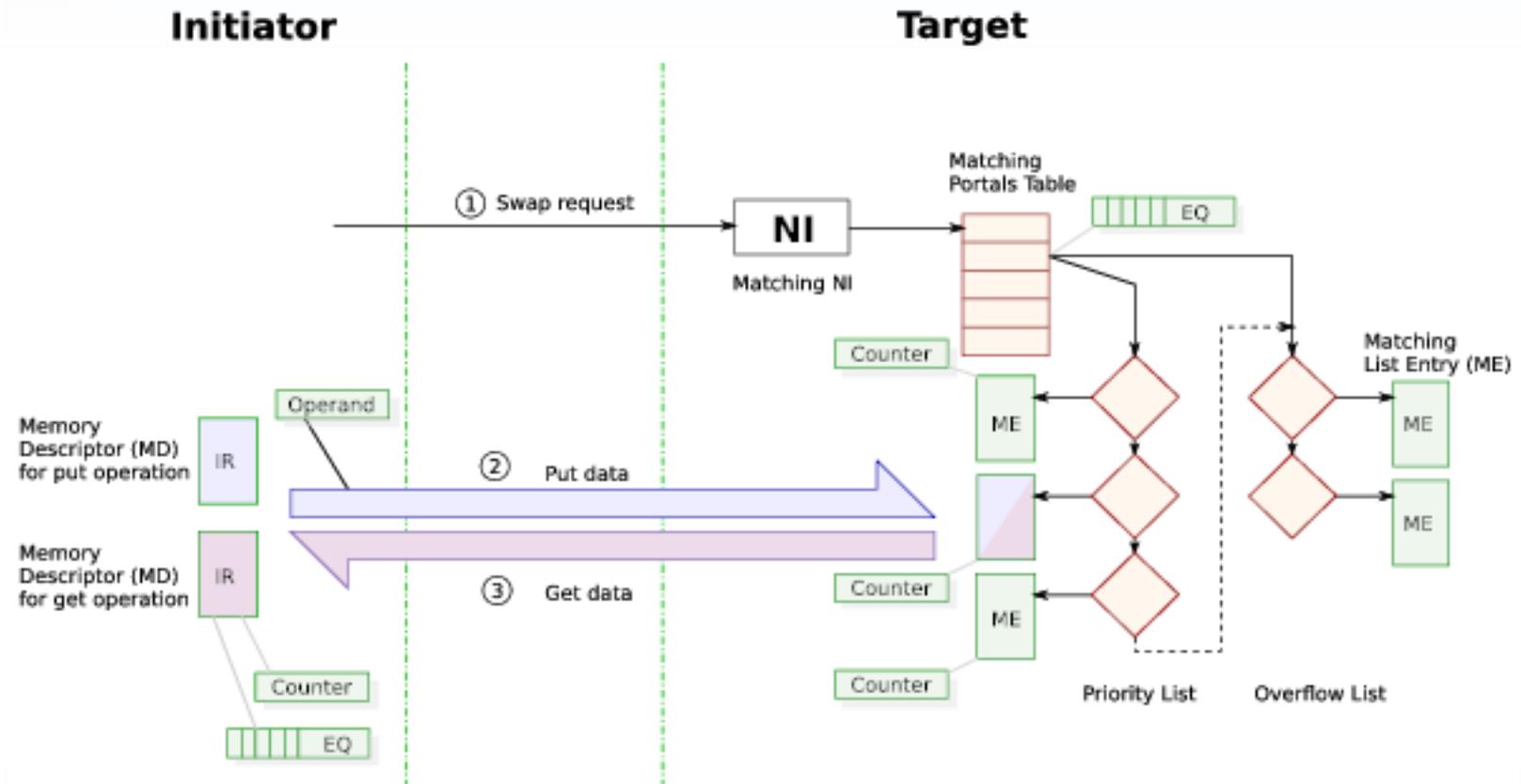
Unmatched Get Operation

Initiator

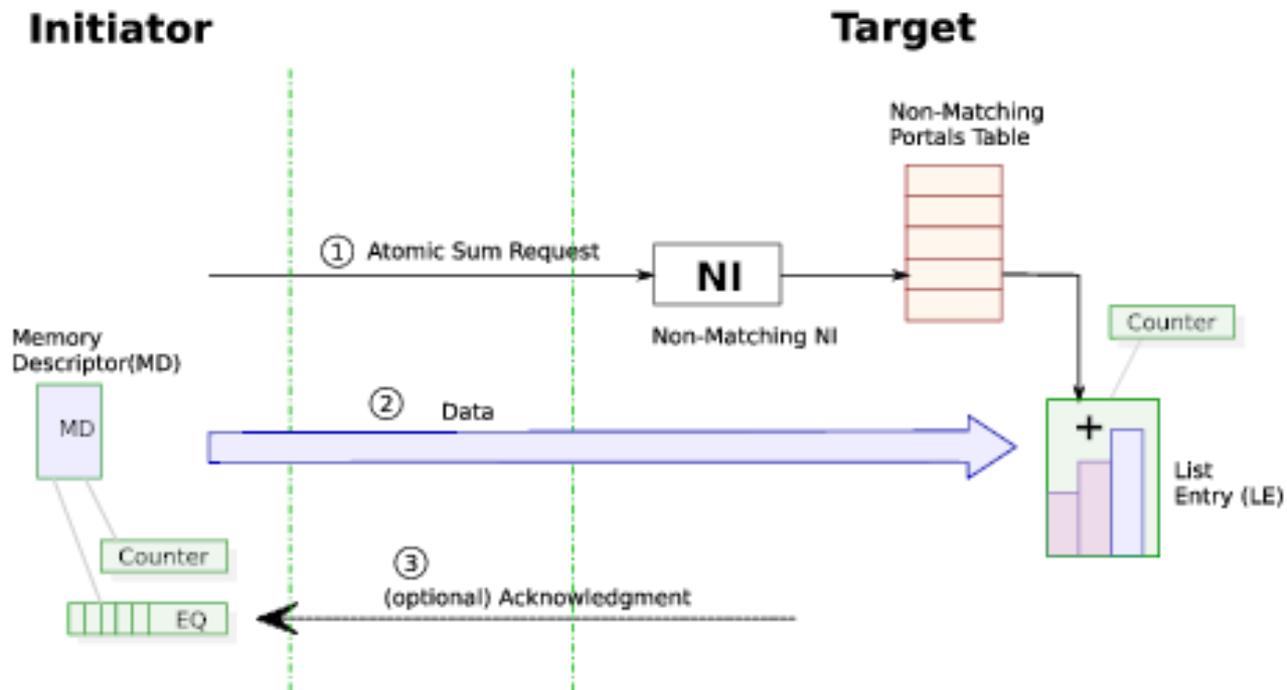
Target



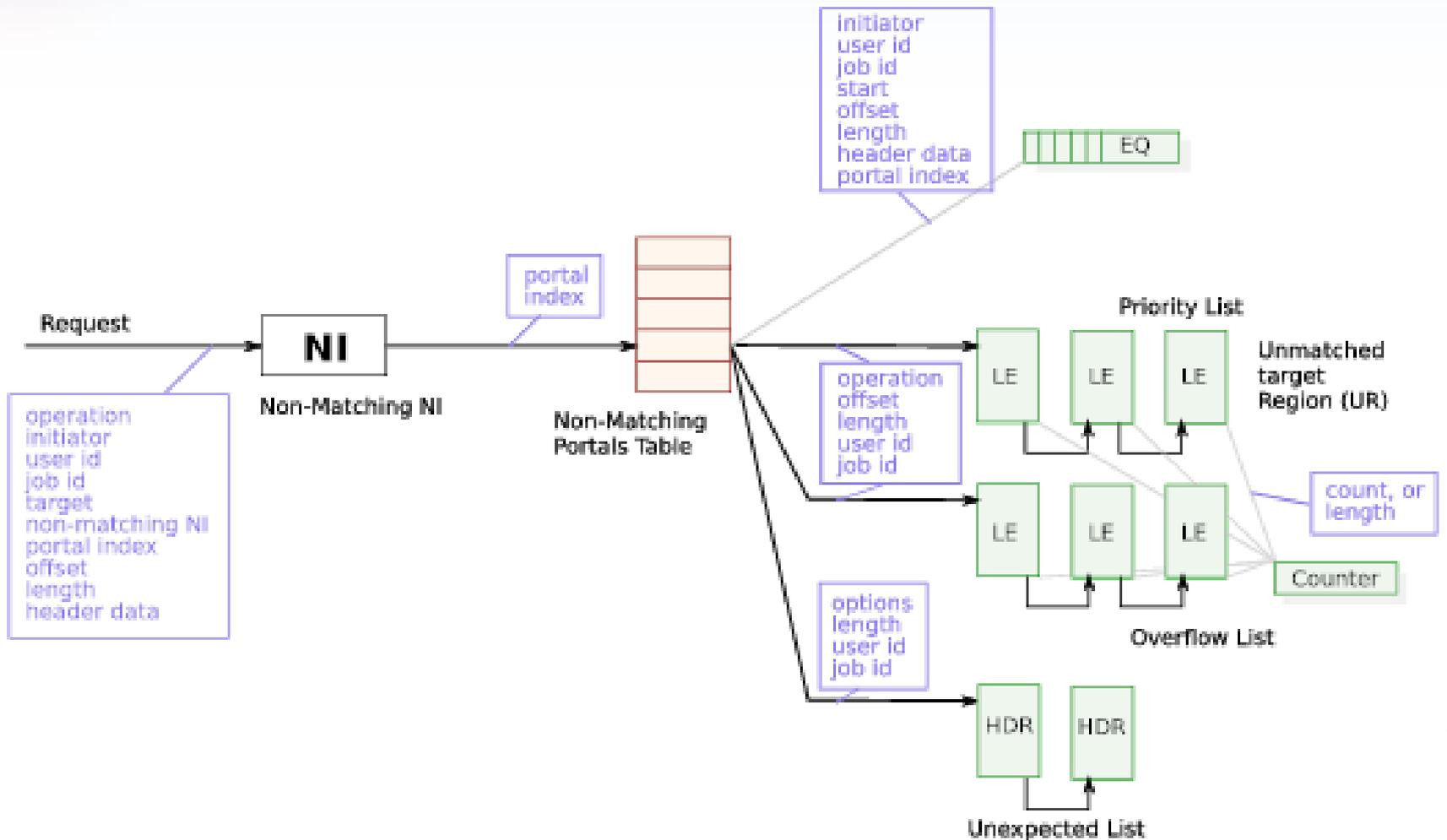
Atomic Operation (Swap)



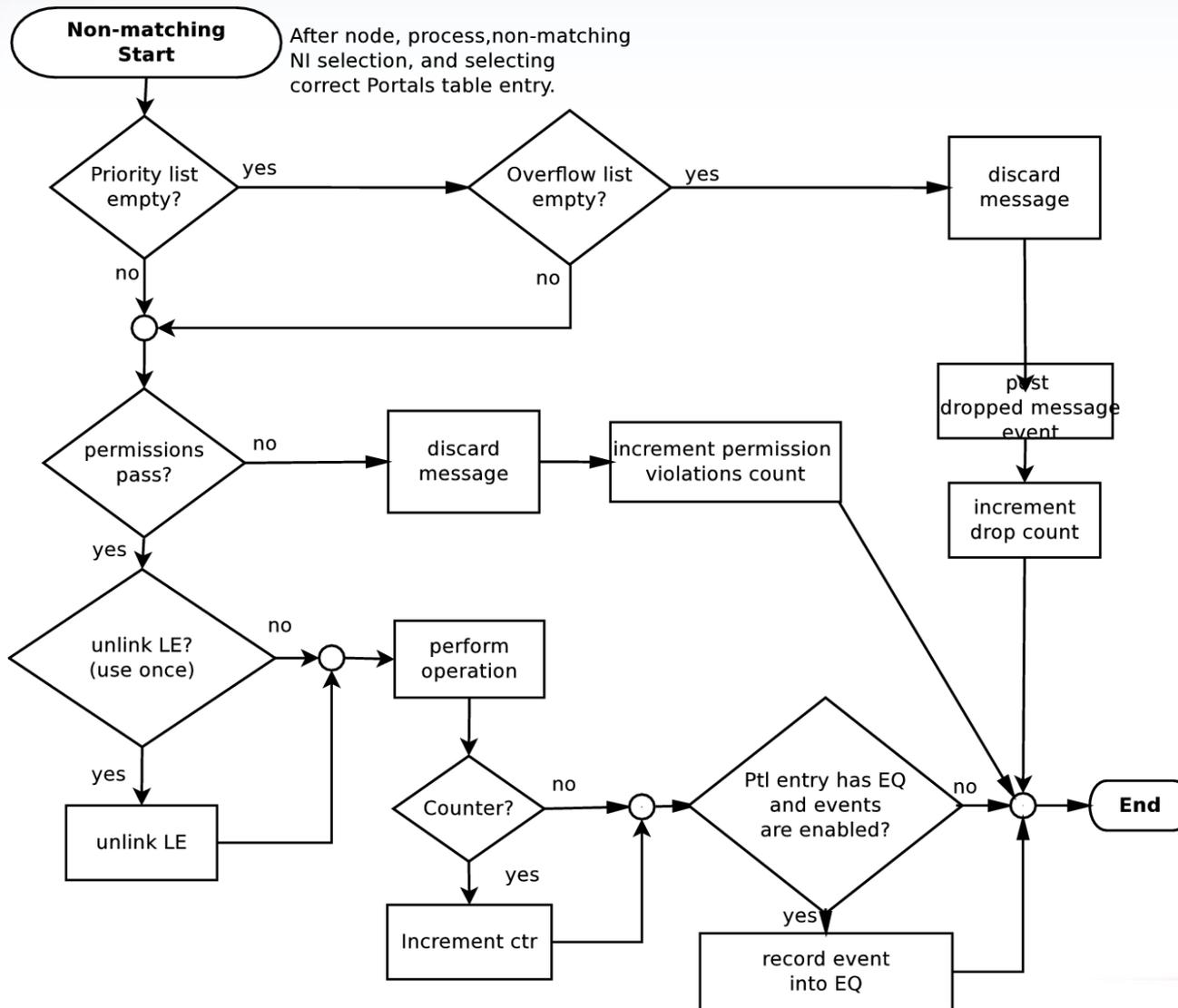
Atomic Operation (Sum)



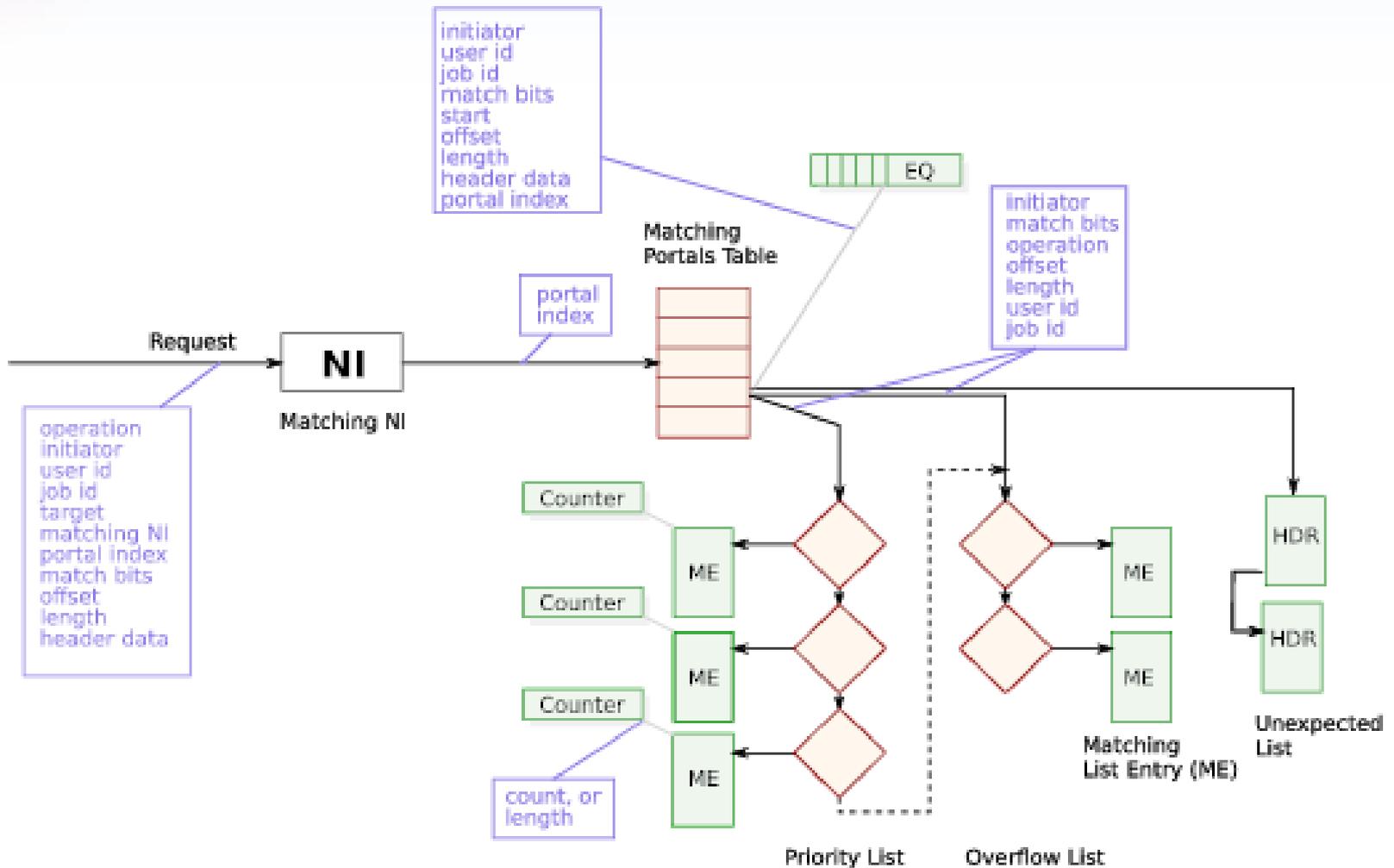
Non-Matching Address Translation



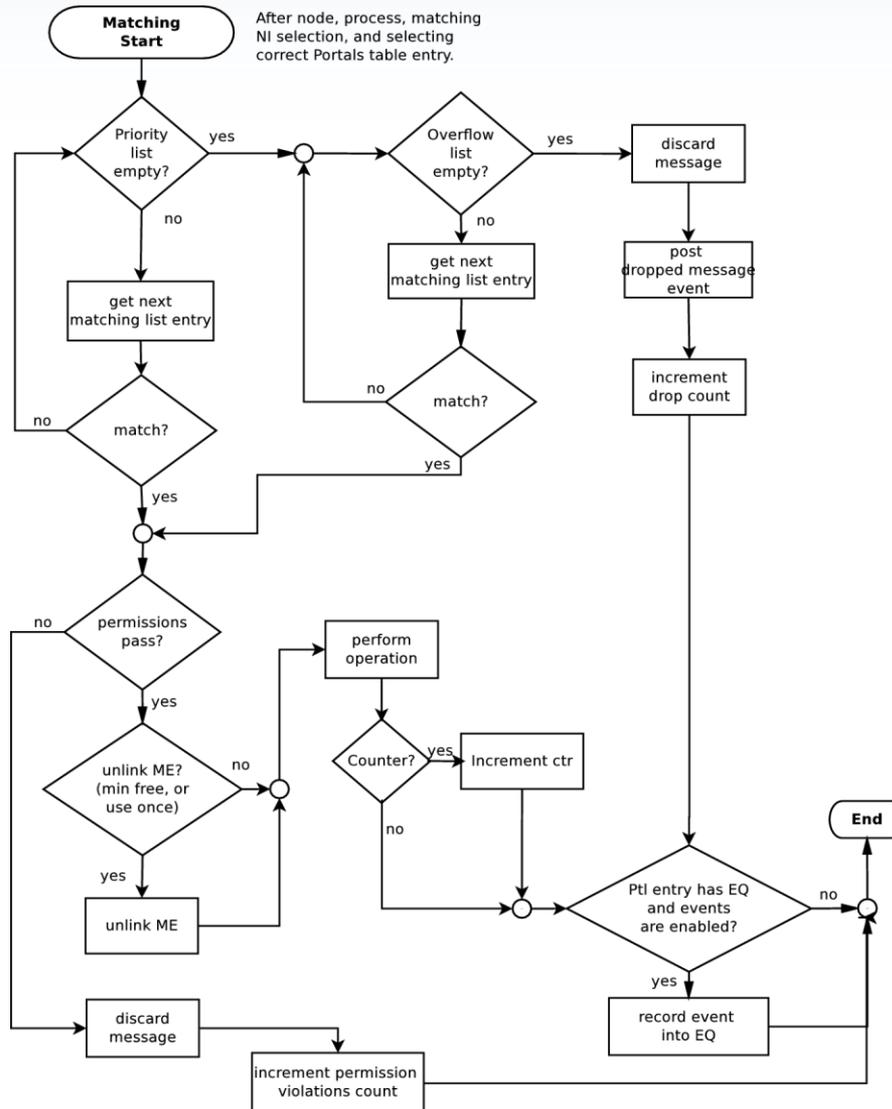
Non-Matching Address Translation



Matching Address Translation

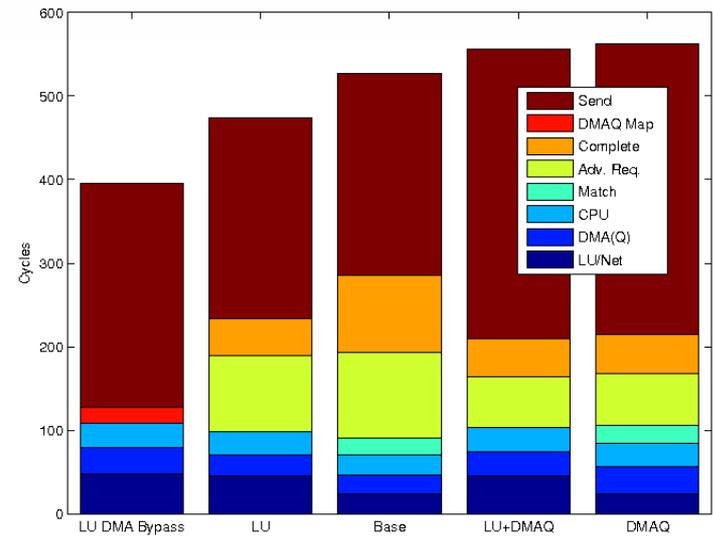


Matching Address Translation



NIC Architecture Co-design

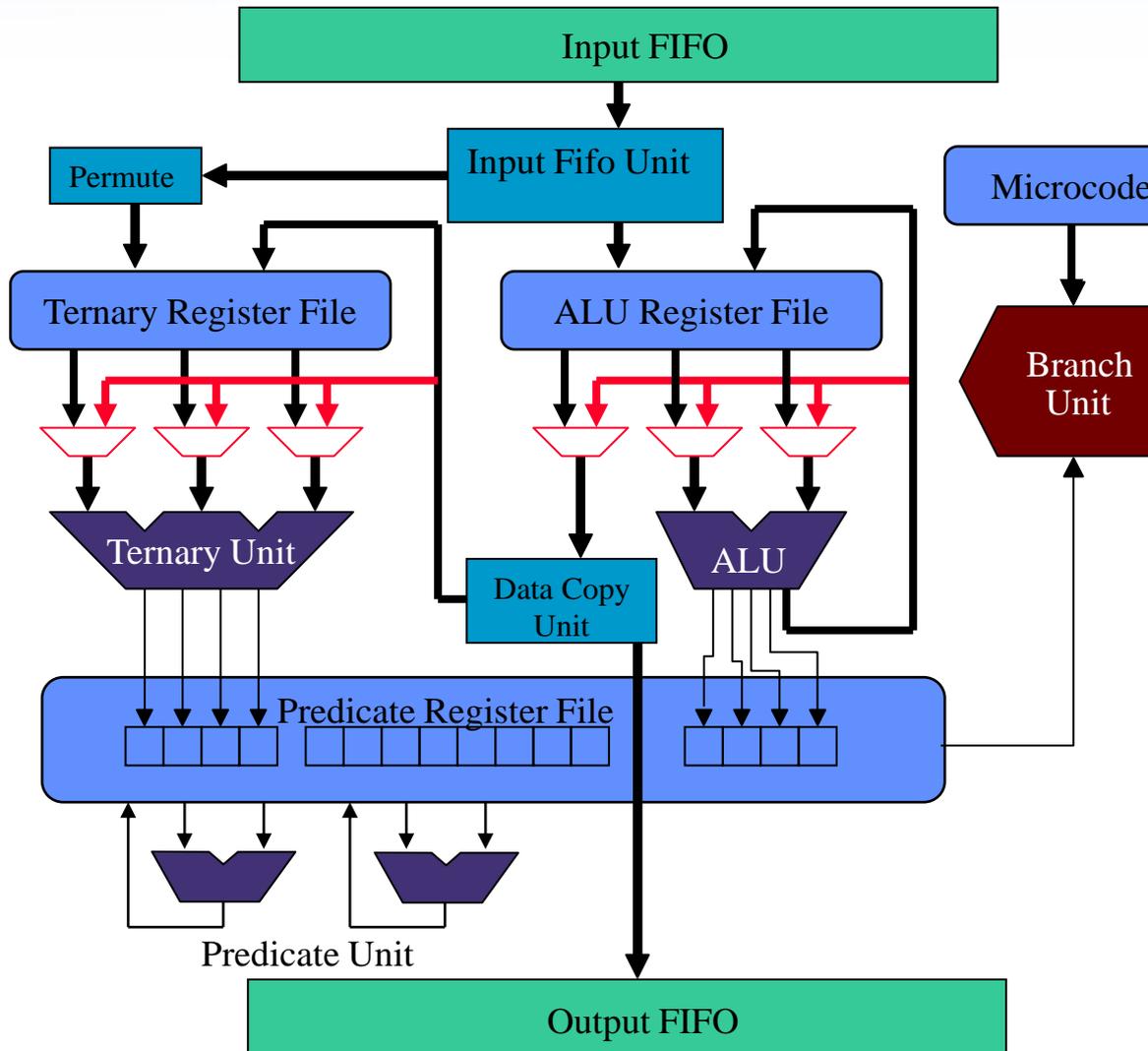
- Prevailing architectural constraints have driven many applications to highly bursty communication patterns
- In a power constrained world this trend will be unsustainable due to inefficient use of the system interconnect
- Design Goal: Produce a NIC architecture that enables overlap through high message rates and independent progress
- Using simulation, NIC hardware & software and host driver software were simultaneously profiled for various architecture choices
- Trade-offs:
 - Which architectural features provide performance advantages
 - What software bottlenecks need to be moved to hardware
 - Which functions can be left to run on NIC CPU or in the host driver
- Next step: rework applications (or portions) to take advantage of the new features and provide feedback for more architectural improvements



Network Interface Controller

- Power will be number one constraint for exascale systems
- Current systems waste energy
 - Using host cores to process messages is inefficient
 - Only move data when necessary
 - Move data to final destination
 - No intermediate copying due to network
- Specialized network hardware
 - Atomic operations
 - Match processing
- Addressing and address translation
 - Virtual address translation
 - Avoid registration cache
 - Logical node translation
 - Rank translation on a million nodes
- Hardware support for thread activation on message arrival

Match Unit Architecture

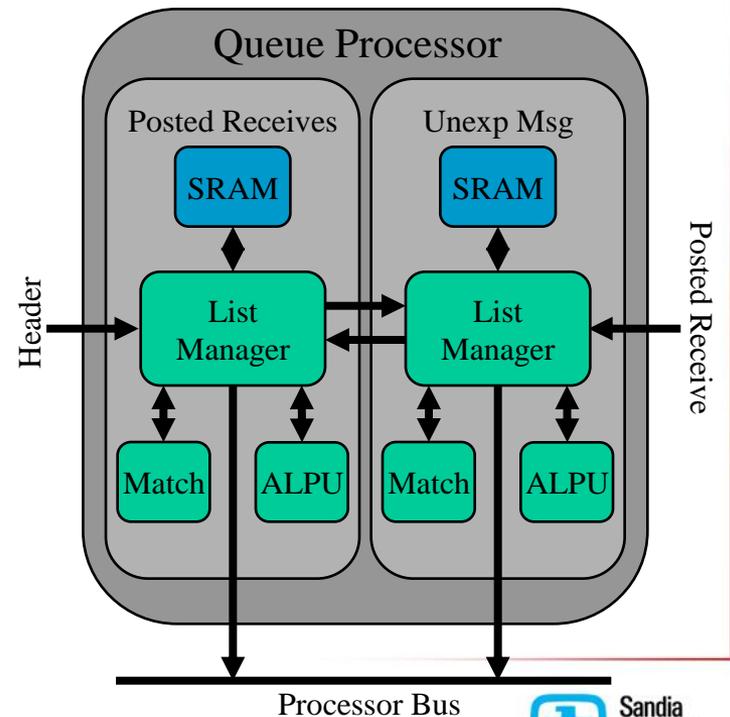


Architecture Drivers

- **High throughput**
 - 3 stage pipeline
- **Irregular data alignment**
 - SIMD operation
 - Permute units
- **Program Consistency**
 - Forwarding in data path
 - Read before write in register file

High Message Throughput Challenges

- 20M messages per second implies a new message every 50ns
- Significant constraints created by MPI semantics
- On-load approach
 - Roadblocks
 - Host general purpose processors are inefficient for list management
 - Caching (a cache miss is 70-120ns latency)
 - Microbenchmarks are cache friendly, real life is not
 - Benefits
 - Easier & cheaper
- Off-load approach
 - Roadblocks
 - Storage requirements on NIC
 - NIC embedded processor is worse at list management (than the host processor)
 - Benefits
 - Opportunity to create dedicated hardware
 - Macroscale pipelining



Portals on OFED

- Provides a high-performance reference implementation for experimentation
- Help identify issues with API, semantics, performance, etc.
- Independent analysis of the specification



Portals over InfiniBand

Author: Bob Pearson
Date: April 3, 2011

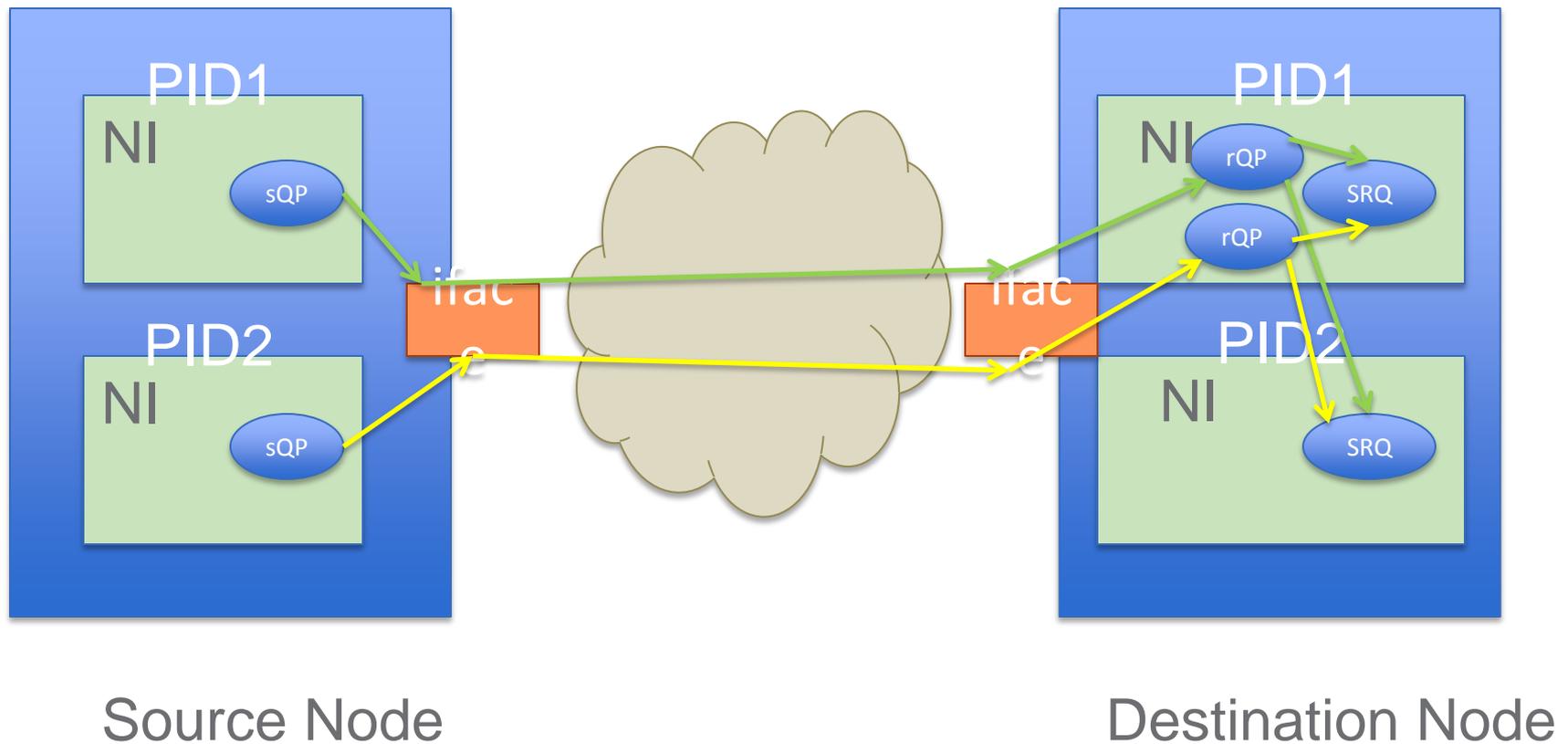
Implementation Details

- Connection Management
- Portals Transport
- Progress Thread

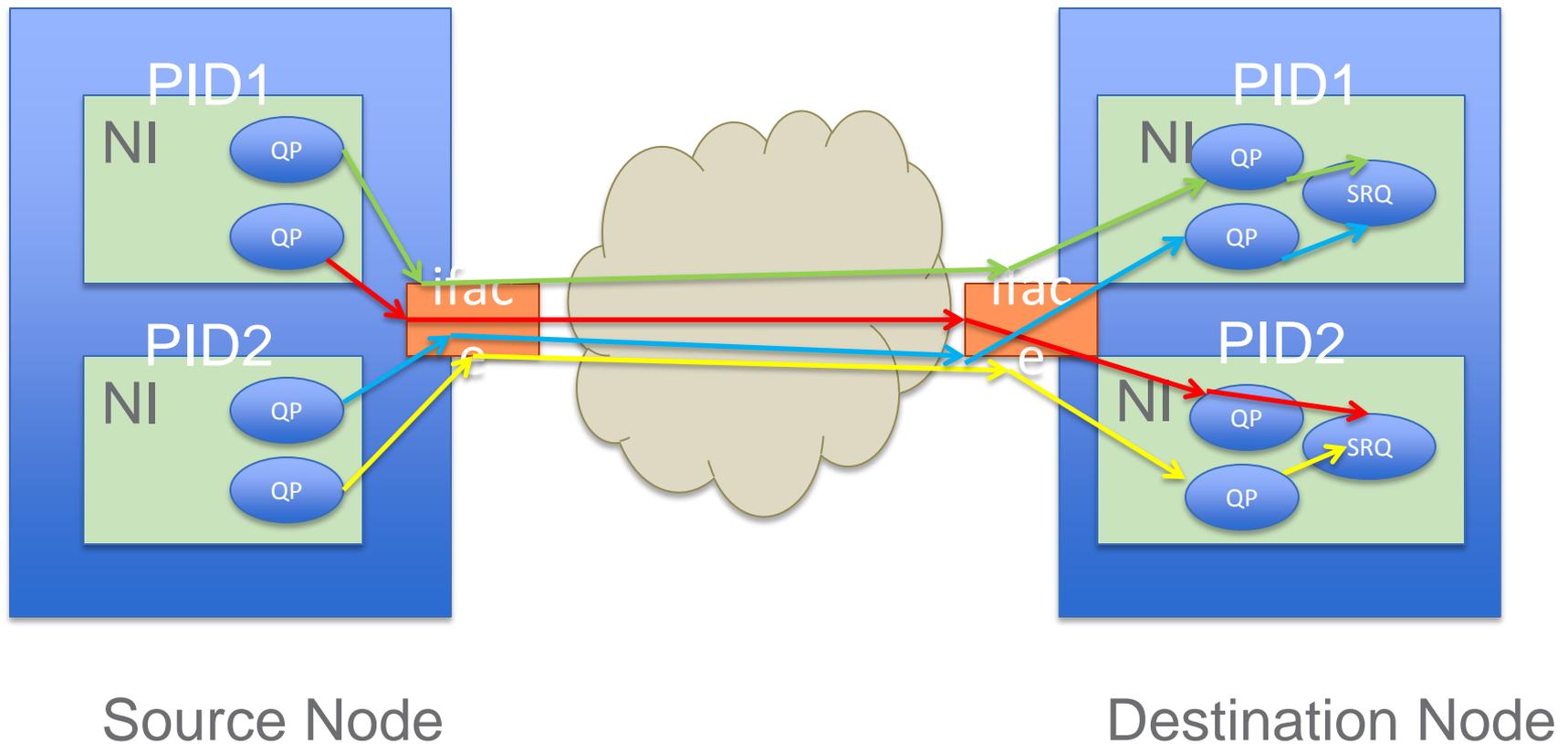
Connection Management

- rdma_cm based
 - One listening rdma_cm_id per PID
 - NID = IPV4 address of IB interface
 - PID = port in RMDA_PS_TCP port space
 - Extension to rdma_cm to support XRC QPs
 - rdma_create_qp(..., &init_attr);
 - Init_attr.xrc_domain new field, if not zero causes rdma_cm to create XRC type QP and ignore PD
 - rdma_cm internally calls ibv_modify_xrc_qp etc. for xrc QPs
 - Exchange XRC SRQ # in CM private data
 - Logical NIs => XRC QPs
 - Physical NIs => RC QPs

Logical NI

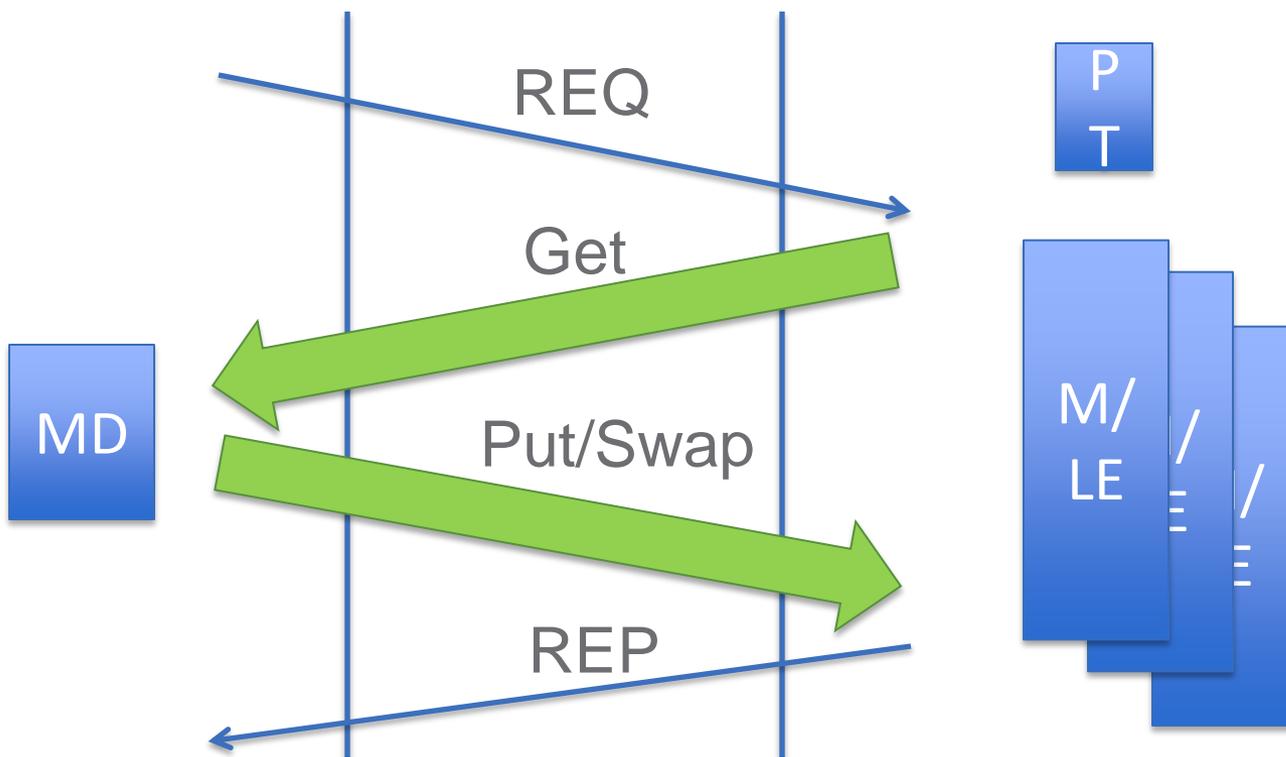


Physical NI



Portals Transport

- Transport similar to SRP
 - Requester sends message to target
 - Three data types for request => target data
 - Immediate (up to MAX INLINE DATA)
 - DMA (up to MAX INLINE SGE)
 - Indirect (up to MAX MESSAGE SIZE)
 - Two data types for target => requester data
 - DMA
 - Indirect
 - Target performs all RDMA operations and then generates reply message



Progress Thread

- One progress thread per PID
 - Handles rdma_cm events
 - CQ events
 - Async events
- Uses libev as event handler

The code

- Code is located at code.google.com/p/portals4