S. Di Girolamo, P. Jolivet, K. D. Underwood, T. Hoefler

Exploiting Offload Enabled Network Interfaces
Lossy Networks
Ethernet

1980’s

Lossless Networks
RDMA

2000’s

Device Programming
Offload

2020’s
Lossy Networks
Ethernet

1980’s

Lossless Networks
RDMA

2000’s

Device Programming
Offload

2020’s

Mellanox Technologies
Intel
Portals
OpenFabrics Alliance
Lossy Networks
Ethernet

Lossless Networks
RDMA

Device Programming
Offload

1980’s
2000’s
2020’s

How to program QsNet?
Lossy Networks  
Ethernet  
1980’s

Lossless Networks  
RDMA  
2000’s

Device Programming  
Offload  
2020’s

How to program QsNet?

How to offload in Portals 4?
Lossy Networks
Ethernet

Lossless Networks
RDMA

Device Programming
Offload

1980’s

2000’s

2020’s

How to program QsNet?

How to offload in Portals 4?

How to offload in libfabric?
We need an abstraction!
Communications (non-blocking)

Computations

Dependencies

Computations:

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>=</td>
</tr>
</tbody>
</table>

Dependencies:

- L0: recv a from P1
- L1: b = compute f(buff, a)
- L2: send b to P1
- L0 and CPU -> L1
- L1 -> L2

Offload Engine

Offload:

- recv
- comp
- send
Performance Model

P0

P1

P0{
    L0: recv m1 from P1;
    L1: send m2 to P1;
}

P1{
    L0: recv m1 from P1;
    L1: send m2 to P1;
    L0 -> L1
}

Performance Model

CPU  
0  0

P0
OE

(s-1)G

P1
OE
CPU

0  0

P0{
  L0: recv m1 from P1;
  L1: send m2 to P1;
}

P1{
  L0: recv m1 from P1;
  L1: send m2 to P1;
  L0 -> L1
}

Performance Model

P0

P1

P0{
    L0: recv m1 from P1;
    L1: send m2 to P1;
}

P1{
    L0: recv m1 from P1;
    L1: send m2 to P1;
    L0 -> L1
}

Performance Model

P0{  
  L0: recv m1 from P1;  
  L1: send m2 to P1;  
}

P1{  
  L0: recv m1 from P1;  
  L1: send m2 to P1;  
  L0 -> L1  
}

Offloading Collectives

A collective operation is fully offloaded if:

1. No synchronization is required in order to start the collective operation.
2. Once a collective operation is started, no further CPU intervention is required in order to progress or complete it.

Definition. A **schedule** is a local dependency graph describing a partial ordered set of operations.

Definition. A **collective communication** involving \( n \) nodes can be modeled as a set of schedules \( S = S_1, \ldots, S_n \) where each node \( i \) participates in the collective executing its own schedule \( S_i \).
Offloading Collectives

A collective operation is fully offloaded if:

1. No synchronization is required in order to start the collective operation.
2. Once a collective operation is started, no further CPU intervention is required in order to progress or complete it.

Definition. A **schedule** is a local dependency graph describing a partial ordered set of operations.

Definition. A **collective communication** involving \( n \) nodes can be modeled as a set of schedules \( S = S_1, \ldots, S_n \) where each node \( i \) participates in the collective executing its own schedule \( S_i \).
Asynchronous algorithms, with their ability to tolerate memory latency, form an important class of algorithms for modern computer architectures.

Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes
- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes
- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule

Collective call
Solo Collectives

- **Synchronized collectives lead to the synchronization of the participating nodes**

- **A solo collective starts its execution as soon as one node (the initiator) starts its own schedule**
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes

- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes
- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes

- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes

- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives

- Synchronized collectives lead to the synchronization of the participating nodes

- A solo collective starts its execution as soon as one node (the initiator) starts its own schedule
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- **Root-Activation**: the initiator is always the root of the collective
- **Non-Root-Activation**: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
Solo Collectives: Activation

- Root-Activation: the initiator is always the root of the collective
- Non-Root-Activation: the initiator can be any participating node
A Case Study: Portals 4

- Based on the one-sided communication model
- Matching/Non-Matching semantics can be adopted

A Case Study: Portals 4

- Based on the one-sided communication model
- Matching/Non-Matching semantics can be adopted

A Case Study: Portals 4

Communication primitives
- Put/Get operations are natively supported by Portals 4
- One-sided + matching semantic

Atomic operations
- Operands are the data specified by the MD at the initiator and by the ME at the target
- Available operators: min, max, sum, prod, swap, and, or, …

Counters
- Associated with MDs or MEs
- Count specific events (e.g., operation completion)

Triggered operations
- Put/Get/Atomic associated with a counter
- Executed when the associated counter reaches the specified threshold
A Case Study: Portals 4

Counters
- Associated with MDs or MEs
- Count specific events (e.g., operation completion)

Triggered operations
- Put/Get/Atomic associated with a counter
- Executed when the associated counter reaches the specified threshold
A Case Study: Portals 4

Counters
- Associated with MDs or MEs
- Count specific events (e.g., operation completion)

Triggered operations
- Put/Get/Atomic associated with a counter
- Executed when the associated counter reaches the specified threshold
A Case Study: Portals 4

Counters
- Associated with MDs or MEs
- Count specific events (e.g., operation completion)

Triggered operations
- Put/Get/Atomic associated with a counter
- Executed when the associated counter reaches the specified threshold
Experimental results

**Curie, a Tier-0 system**
- 5,040 nodes
- 2 eight-core Intel Sandy Bridge processors
- Full fat-tree Infiniband QDR

**OMPI**: Open MPI 1.8.4
**OMPI/P4**: Open MPI 1.8.4 + Portals 4 backend

**FFLIB**: proof of concept library

One process per computing node

More about FFLIB at:
http://spcl.inf.ethz.ch/Research/Parallel_Programming/FFlib/
Experimental results

Curie, a Tier-0 system
5,040 nodes
2 eight-core Intel Sandy Bridge processors
Full fat-tree Infiniband QDR
OMPI: Open MPI 1.8.4
OMPI/P4: Open MPI 1.8.4 + Portals 4 backend
FFLIB: proof of concept library
One process per computing node

More about FFLIB at:
http://spcl.inf.ethz.ch/Research/Parallel_Programming/FFlib/
Simulations

- Why? To study offloaded collectives at large scale
- How? Extending the LogGOPSIm to simulate Portals 4 functionalities

### Simulations

**Broadcast**

- **P4-SW**
- **P4-HW**

**Allreduce**

- **P4-SW**
- **P4-HW**

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>o</th>
<th>g</th>
<th>G</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>P4-SW</td>
<td>5μs</td>
<td>6μs</td>
<td>6μs</td>
<td>0.4ns</td>
<td>0.9ns</td>
</tr>
<tr>
<td>P4-HW</td>
<td>2.7μs</td>
<td>1.2μs</td>
<td>0.5μs</td>
<td>0.4ns</td>
<td>0.3ns [4]</td>
</tr>
</tbody>
</table>


Abstract Machine Model

Offloading Collectives

Solo Collectives

Mapping to Portals 4

Results

Co-Authors

P. Jolivet
K. D. Underwood
T. Hoefler