MigrOS: Transparent Live-Migration Support for Containerised RDMA Applications

Maksym Planeta[1] Jan Bierbaum[1] Leo Sahaya Daphne Antony[2] Torsten Hoefler[3] Hermann Härtig[1]

[1]TU Dresden [2]AMOLF [3] ETH Zürich

2021 USENIX Annual Technical Conference

2021-07-14

Motivation

- Containers are ubiquitous

Motivation

- Containers are ubiquitous
- Fast networks (40G to 400G NICs) have become widespread

Motivation

- Containers are ubiquitous
- Fast networks (40G to 400G NICs) have become widespread
- Traditional network stacks are unsustainable

Motivation

- Containers are ubiquitous
- Fast networks (40G to 400G NICs) have become widespread
- Traditional network stacks are unsustainable
- RDMA networks access device directly, breaking isolation

Motivation

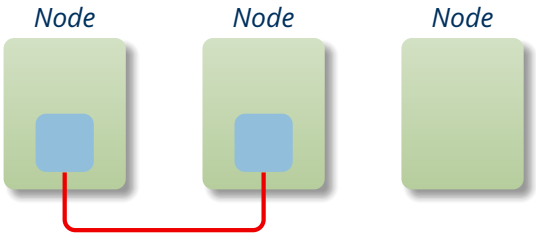
- Containers are ubiquitous
- Fast networks (40G to 400G NICs) have become widespread
- Traditional network stacks are unsustainable
- RDMA networks access device directly, breaking isolation
- Direct device access complicates live migration

Motivation

- Containers are ubiquitous
- Fast networks (40G to 400G NICs) have become widespread
- Traditional network stacks are unsustainable
- RDMA networks access device directly, breaking isolation
- Direct device access complicates live migration
- Goal: live migration with no application modifications and no performance overhead

Live Migration

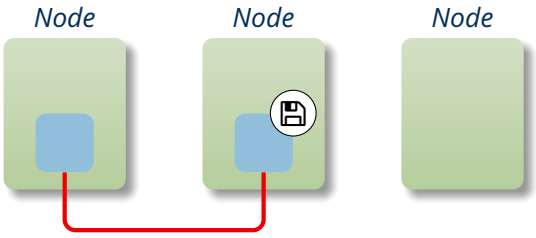
Consists of



Live Migration

Consists of

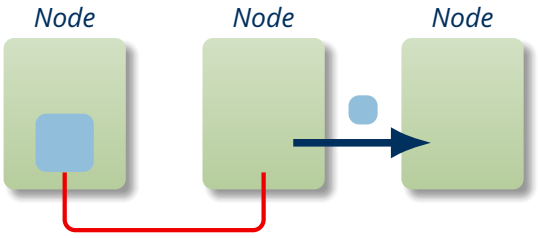
- Checkpoint



Live Migration

Consists of

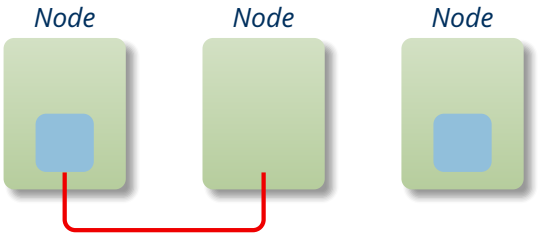
- Checkpoint
- State transfer



Live Migration

Consists of

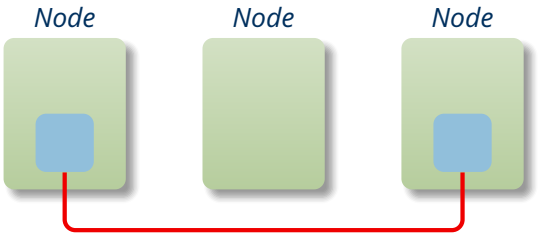
- Checkpoint
- State transfer
- Restart



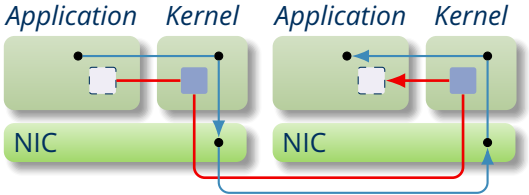
Live Migration

Consists of

- Checkpoint
- State transfer
- Restart
  - Network reconfiguration

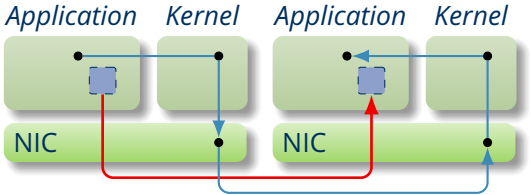


RDMA Networks



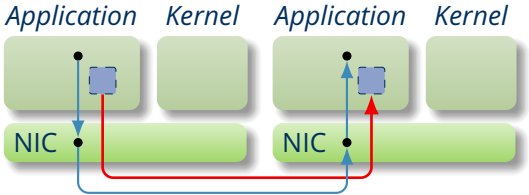
RDMA Networks

- Zero-copy



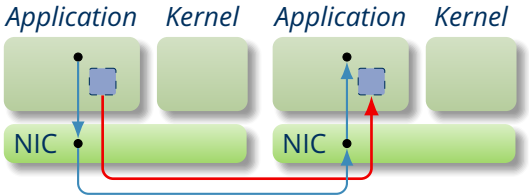
RDMA Networks

- Zero-copy
- OS-bypass



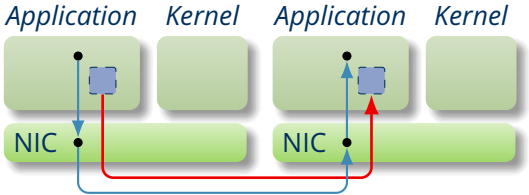
RDMA Networks

- Zero-copy
- OS-bypass
  - ✔ Higher network performance
  - ✔ Lower CPU overhead



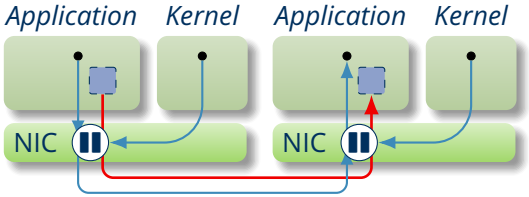
RDMA Networks

- Zero-copy
- OS-bypass
  - ✔ Higher network performance
  - ✔ Lower CPU overhead
  - ✖ OS loses control



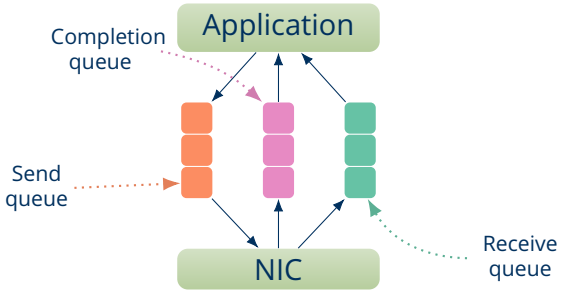
RDMA Networks

- Zero-copy
- OS-bypass
  - ✔ Higher network performance
  - ✔ Lower CPU overhead
  - ✖ OS loses control
- Goal: Take back control



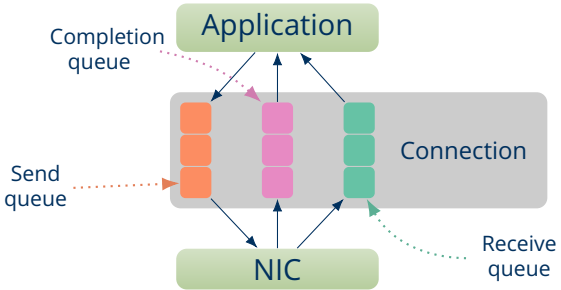
Consistent Checkpointing

- Send, receive, and completion queues



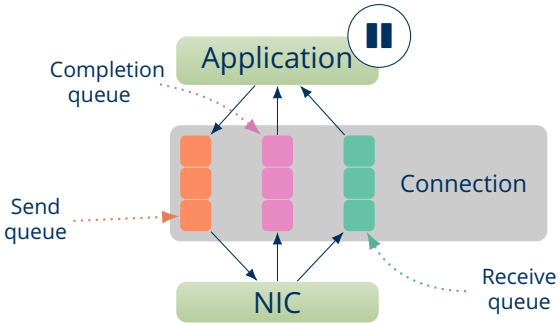
Consistent Checkpointing

- Send, receive, and completion queues
- Shared connection state



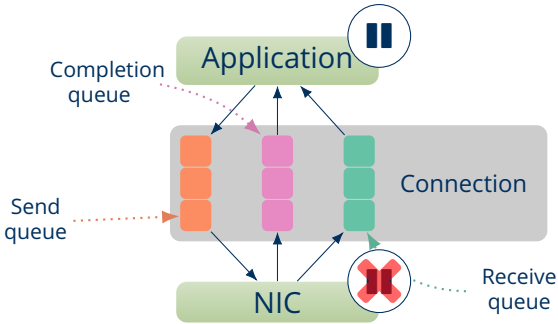
Consistent Checkpointing

- Send, receive, and completion queues
- Shared connection state
- OS can stop the application



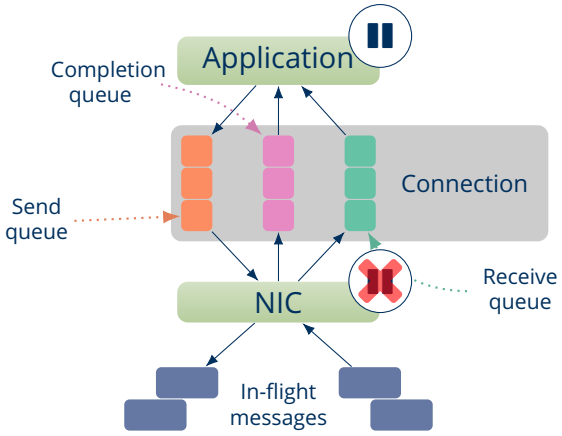
Consistent Checkpointing

- Send, receive, and completion queues
- Shared connection state
- OS can stop the application
- OS cannot stop the NIC



Consistent Checkpointing

- Send, receive, and completion queues
- Shared connection state
- OS can stop the application
- OS cannot stop the NIC
- Lost updates

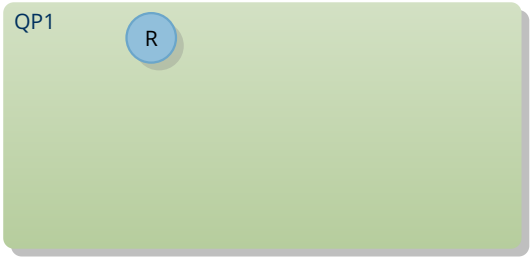


What is a Queue Pair (QP)?

- QP represents connection

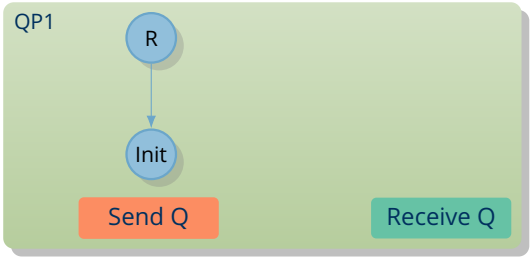
What is a Queue Pair (QP)?

- QP represents connection
- QP state machine
  - Reset



What is a Queue Pair (QP)?

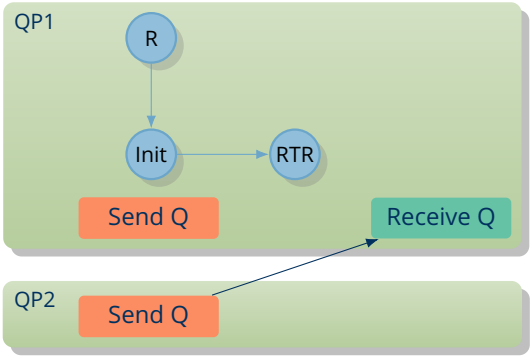
- QP represents connection
- QP state machine
  - Reset
  - Init



What is a Queue Pair (QP)?

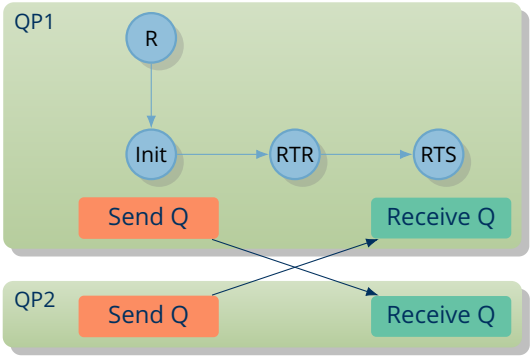
- QP represents connection
- QP state machine
  - Reset
  - Init
  - Ready-to-Receive

- Fixed source



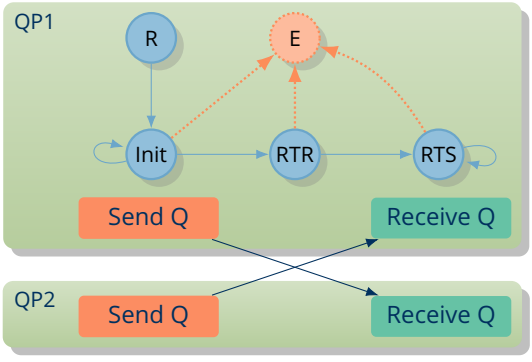
What is a Queue Pair (QP)?

- QP represents connection
- QP state machine
  - Reset
  - Init
  - Ready-to-Receive
  - Ready-to-Send
- Fixed source
- Fixed destination



What is a Queue Pair (QP)?

- QP represents connection
- QP state machine
  - Reset
  - Init
  - Ready-to-Receive
  - Ready-to-Send
  - Error
- Fixed source
- Fixed destination



Pause/Resume Protocol

- QPs in RTS state

Pause/Resume Protocol

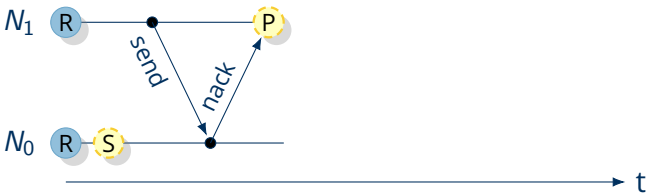
- QPs in RTS state
- Changes:
  - *Stopped* S state

Pause/Resume Protocol

- QPs in RTS state
- Changes:
  - *Stopped* (S) state
  - *Stopped* nack

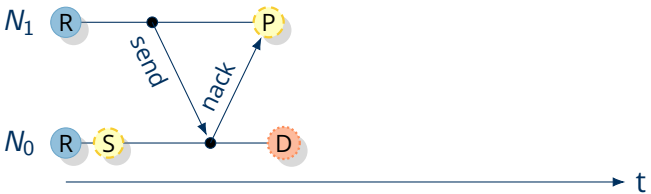
Pause/Resume Protocol

- QPs in RTS state
- Changes:
  - *Stopped* S state
  - *Stopped* nack
  - *Paused* P state



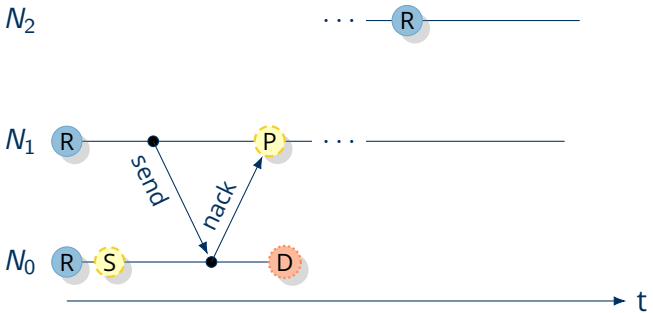
Pause/Resume Protocol

- QPs in RTS state
- Changes:
  - *Stopped* S state
  - *Stopped* nack
  - *Paused* P state



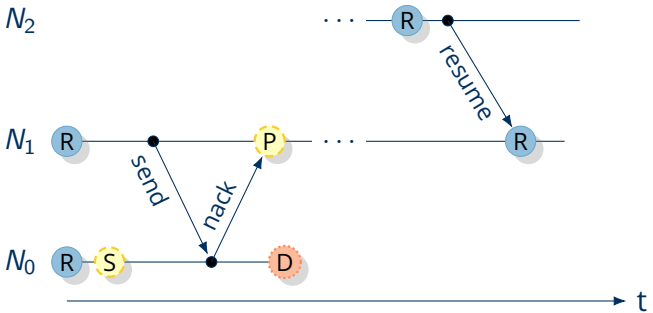
Pause/Resume Protocol

- QPs in RTS state
- Changes:
  - *Stopped* (S) state
  - *Stopped* nack
  - *Paused* (P) state



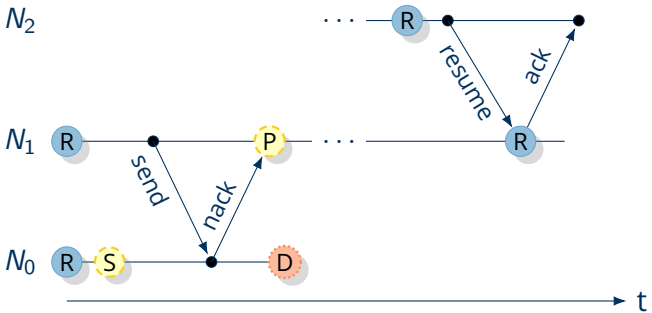
Pause/Resume Protocol

- QPs in RTS state
- Changes:
  - *Stopped* (S) state
  - *Stopped* nack
  - *Paused* (P) state
  - *Resume* message
- No lost state updates



Pause/Resume Protocol

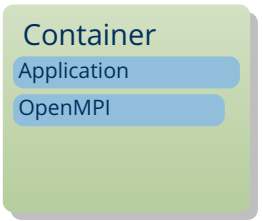
- QPs in RTS state
- Changes:
  - *Stopped* S state
  - *Stopped* nack
  - *Paused* P state
  - *Resume* message
- No lost state updates



MigrOS Architecture

Unmodifed guest:

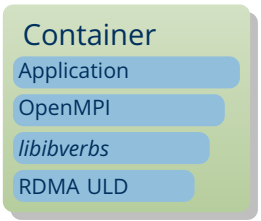
- Application



MigrOS Architecture

Unmodifed guest:

- Application
- `libibverbs`
- User-level driver



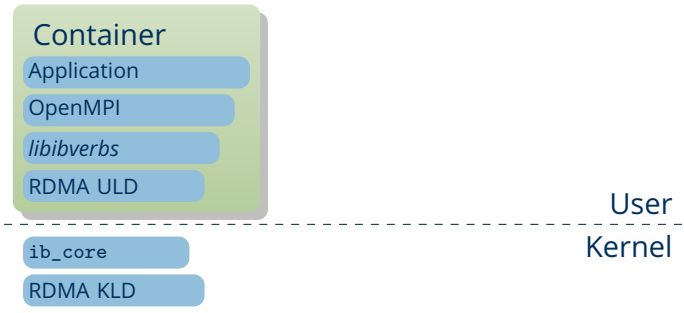
MigrOS Architecture

Unmodifed guest:

- Application
- `libibverbs`
- User-level driver

Modified host:

- Kernel-level driver



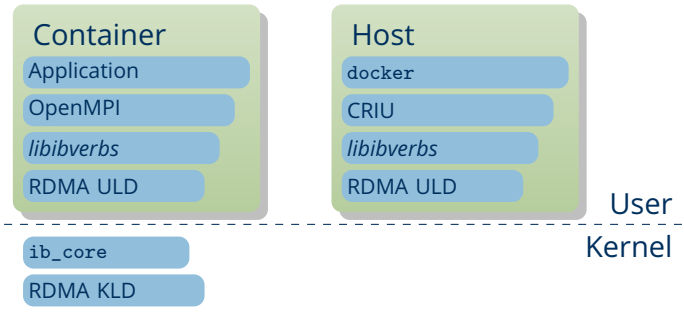
MigrOS Architecture

Unmodifed guest:

- Application
- `libibverbs`
- User-level driver

Modified host:

- Kernel-level driver
- User-level driver
- `libibverbs`
- CRIU



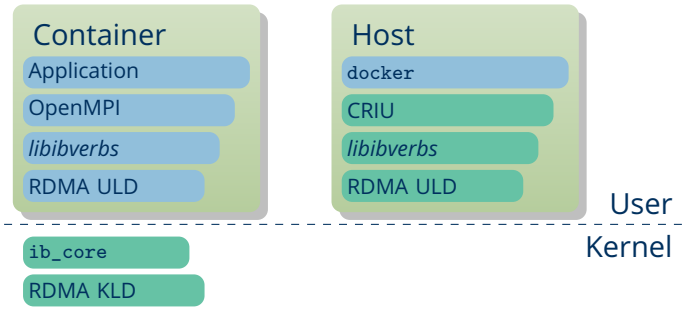
MigrOS Architecture

Unmodifed guest:

- Application
- `libibverbs`
- User-level driver

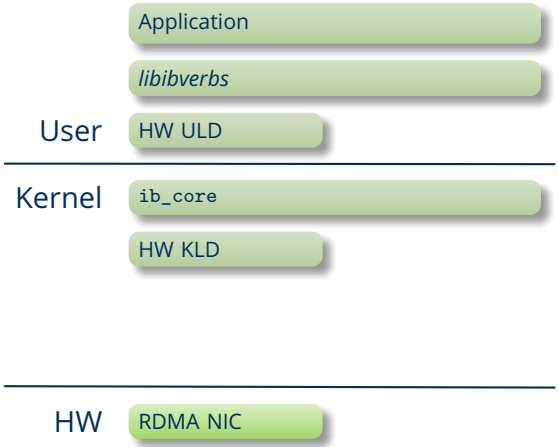
Modified host:

- Kernel-level driver
- User-level driver
- `libibverbs`
- CRIU



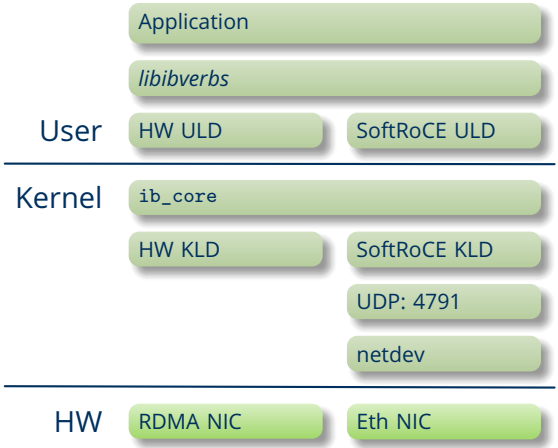
Implementation

- RoCEv2 – InfiniBand protocol over UPD port



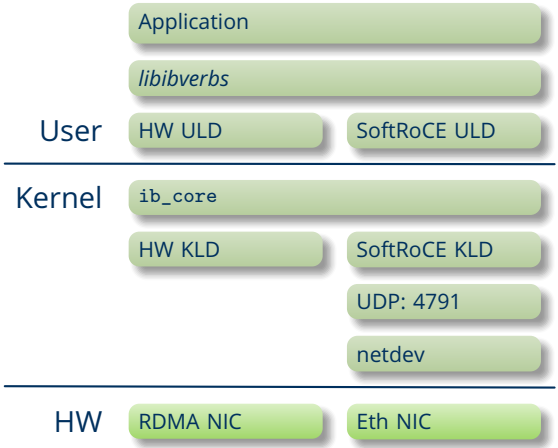
Implementation

- RoCEv2 – InfiniBand protocol over UPD port
- SoftRoCE – software RoCEv2 implementation



Implementation

- RoCEv2 – InfiniBand protocol over UPD port
- SoftRoCE – software RoCEv2 implementation
- ✖ Performance overhead

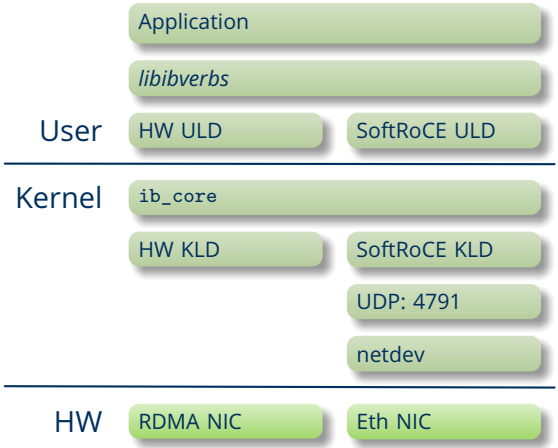


Implementation

- RoCEv2 – InfiniBand protocol over UPD port
- SoftRoCE – software RoCEv2 implementation

✖ Performance overhead

✔ Easy to change the protocol



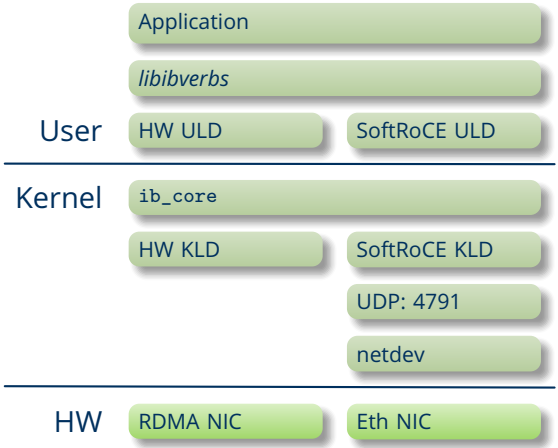
Implementation

- RoCEv2 – InfiniBand protocol over UPD port
- SoftRoCE – software RoCEv2 implementation

✖ Performance overhead

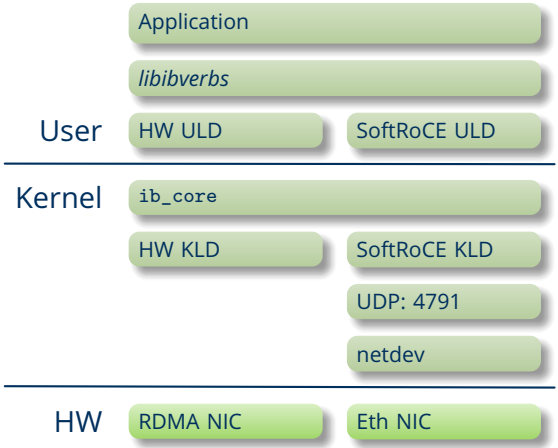
✔ Easy to change the protocol

✔ Feature-full implementation



Implementation

- RoCEv2 – InfiniBand protocol over UPD port
- SoftRoCE – software RoCEv2 implementation
- ✖ Performance overhead
- ✔ Easy to change the protocol
- ✔ Feature-full implementation
- Generalisable for other protocols



Evaluation

A few changes to the RDMA protocol enable transparent live-migration

Evaluation

A few changes to the RDMA protocol enable transparent live-migration

- Changes are small and backwards compatible

Evaluation

A few changes to the RDMA protocol enable transparent live-migration

- Changes are small and backwards compatible
- Performance of normal operation is not affected

Evaluation

A few changes to the RDMA protocol enable transparent live-migration

- Changes are small and backwards compatible
- Performance of normal operation is not affected
- Faster than software-level interception

Evaluation

A few changes to the RDMA protocol enable transparent live-migration

- Changes are small and backwards compatible
- Performance of normal operation is not affected
- Faster than software-level interception
- Practicality

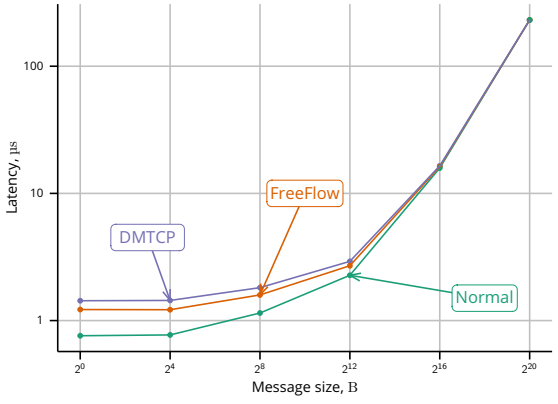
Evaluation

A few changes to the RDMA protocol enable transparent live-migration

- Changes are small and backwards compatible
- **Performance of normal operation is not affected**
- Faster than software-level interception
- **Practicality**

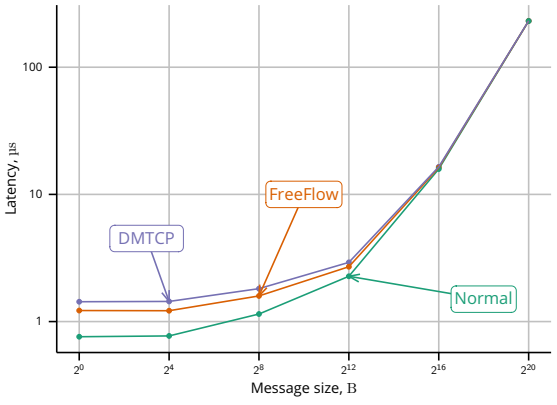
Evaluation: Motivation

- ConnectX-3 40GbE



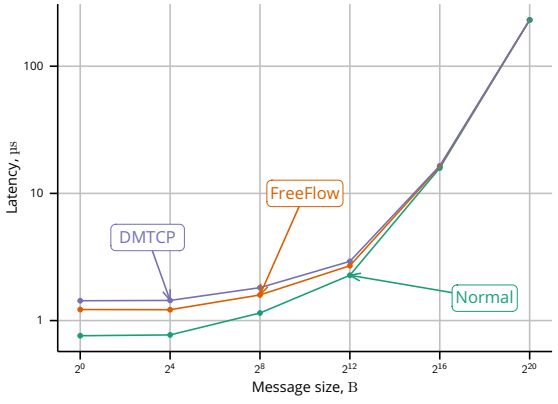
Evaluation: Motivation

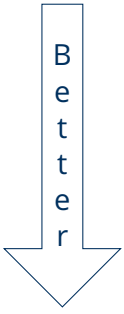
- ConnectX-3 40GbE
- MigrOS: No overhead, normal operation
- FreeFlow: Software RDMA switch
- DMTCP: Checkpoint/restore library



Evaluation: Motivation

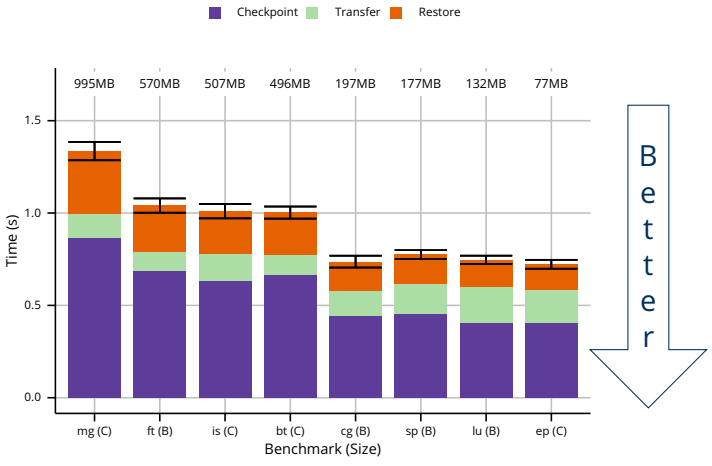
- ConnectX-3 40GbE
- MigrOS: No overhead, normal operation
- FreeFlow: Software RDMA switch
- DMTCP: Checkpoint/restore library
- Constant per message latency increase





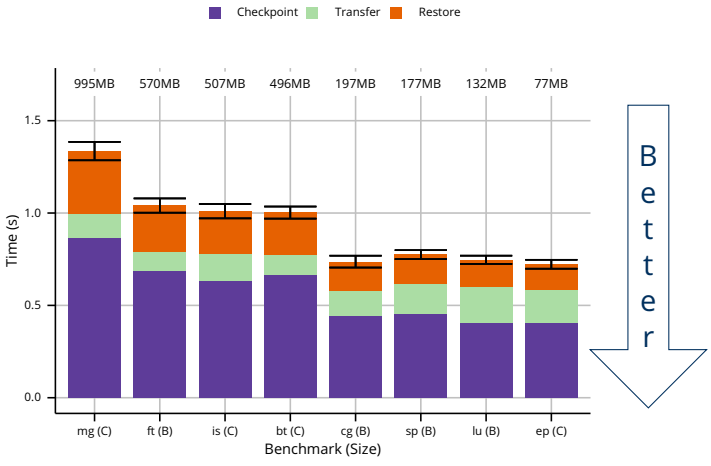
Evaluation: Practicality

- SoftRoCE



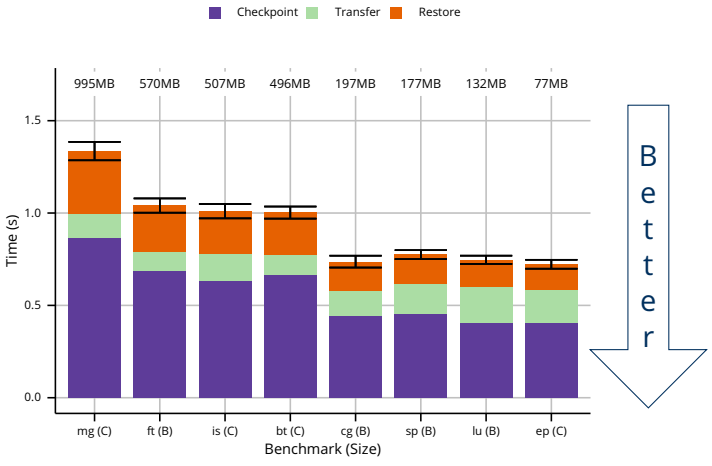
Evaluation: Practicality

- SoftRoCE
- NAS Parallel Benchmarks



Evaluation: Practicality

- SoftRoCE
- NAS Parallel Benchmarks
- Checkpoint and transfer in parallel



Conclusion

- Added two new states and two new message types to RoCEv2
- Small changes to the software stack (CRIU, kernel, libibverbs)
- Integrate RDMA migration into existing container runtime and orchestration

Conclusion

- Added two new states and two new message types to RoCEv2
- Small changes to the software stack (CRIU, kernel, libibverbs)
- Integrate RDMA migration into existing container runtime and orchestration

Thank you!

Backup Slides