MigrOS: Transparent Live-Migration Support for Containerised RDMA Applications

Maksym Planeta\textsuperscript{1} Jan Bierbaum\textsuperscript{1} Leo Sahaya Daphne Antony\textsuperscript{2} Torsten Hoefler\textsuperscript{3} Hermann Härtig\textsuperscript{1}

\textsuperscript{1}TU Dresden \quad \textsuperscript{2}AMOLF \quad \textsuperscript{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

Maksym Planeta: “MigrOS”
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

- Checkpoint
- State transfer
- Restart

– Network recon/uniFB01guration

Node

Node

Node
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

- Zero-copy
- OS-bypass

Higher network performance
Lower CPU overhead

Goal: Take back control

Maksym Planeta: “MigrOS”
RDMA Networks

- Zero-copy
RDMA Networks

- Zero-copy
- OS-bypass

Maksym Planeta: “MigrOS”
RDMA Networks

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

Maksym Planeta: “MigrOS”
RDMA Networks

- Zero-copy
- OS-bypass
  - ✓ Higher network performance
  - ✓ Lower CPU overhead
  - ✗ OS loses control

Maksym Planeta: “MigrOS”
RDMA Networks

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

Diagram:

```
+-------------------+     +-------------------+     +-------------------+     +-------------------+
|       Application  |     |        Kernel     |     |       Application  |     |        Kernel     |
|                   |     |                   |     |                   |     |                   |
|                  +-----+                  +-----+                  +-----+                  +-----+
|                  |     |                  |     |                  |     |                  |     |
|                  |     |                  |     |                  |     |                  |     |
|                  +-----+                  +-----+                  +-----+                  +-----+
|                   |     |                   |     |                   |     |                   |
|                   |     |                   |     |                   |     |                   |
|                   +-----+                  +-----+                  +-----+                  +-----+
|       Application  |     |        Kernel     |     |       Application  |     |        Kernel     |
|                   |     |                   |     |                   |     |                   |
|                  +-----+                  +-----+                  +-----+                  +-----+
|                  |     |                  |     |                  |     |                  |     |
|                  |     |                  |     |                  |     |                  |     |
|                  +-----+                  +-----+                  +-----+                  +-----+
|       NIC        |     |        NIC       |     |       NIC        |     |        NIC       |
+-------------------+     +-------------------+     +-------------------+     +-------------------+
```

Maksym Planeta: “MigrOS”
Consistent Checkpointing

- Send, receive, and completion queues

Application

- Completion queue

Send queue

NIC

- Receive queue

Send queue

Maksym Planeta: “MigrOS”
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

Application

Connection

NIC

Completion queue

Send queue

Receive queue
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

QP state machine diagram:

- QP1
  - R
  - Init
  - Send Q
  - Receive Q

Maksym Planeta: “MigrOS”
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

Maksym Planeta: “MigrOS”
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

\[ N_1 \quad R \]

\[ N_0 \quad R \]

No lost state updates
Pause/Resume Protocol

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

\[ N_0 \quad R \quad S \]
\[ N_1 \]

\[ t \]
Pause/Resume Protocol

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

Maksym Planeta: “MigrOS”
Pause/Resume Protocol

• QPs in RTS state
• Changes:
  - \textit{Stopped} $S$ state
  - \textit{Stopped} nack
  - \textit{Paused} $P$ state
Pause/Resume Protocol

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

\[ N_0 \rightarrow R \rightarrow S \rightarrow D \]
\[ N_1 \rightarrow R \rightarrow \text{send} \rightarrow P \rightarrow \text{nack} \]

Maksym Planeta: “MigrOS” 7/13
Pause/Resume Protocol

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

\[N_0 \rightarrow N_1 \rightarrow N_2 \rightarrow \ldots \]

\(t\)
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* state
  - *Stopped* nack
  - *Paused* state
  - *Resume* message
- No lost state updates
MigrOS Architecture

Unmodified guest:
- Application

Diagram:
- Container
  - Application
  - OpenMPI
MigrOS Architecture

Unmodified guest:
• Application
• libibverbs
• User-level driver

Container
- Application
- OpenMPI
- libibverbs
- RDMA ULD
MigrOS Architecture

Unmodified guest:
• Application
• libibverbs
• User-level driver

Modified host:
• Kernel-level driver

Diagram:
- Container:
  - Application
  - OpenMPI
  - libibverbs
  - RDMA ULD
- ib_core
- RDMA KLD
MigrOS Architecture

Unmodified guest:
- Application
- libibverbs
- User-level driver

Modified host:
- Kernel-level driver
- User-level driver
- libibverbs
- CRIU
MigrOS Architecture

**Unmodified guest:**
- Application
- libibverbs
- User-level driver

**Modified host:**
- Kernel-level driver
- User-level driver
- libibverbs
- CRIU

---

**Container**
- Application
- OpenMPI
- libibverbs
- RDMA ULD

**Host**
- docker
- CRIU
- libibverbs
- RDMA ULD

**User Kernel**
- ib_core
- RDMA KLD
Implementation

• RoCEv2 – InfiniBand protocol over UDP port

<table>
<thead>
<tr>
<th>User</th>
<th>Hardware ULD</th>
<th>Kernel</th>
<th>Hardware KLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>libibverbs</td>
<td>ib_core</td>
<td>RDMA NIC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HW KLD</td>
<td></td>
</tr>
</tbody>
</table>

Maksym Planeta: “MigrOS”
# Implementation

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

<table>
<thead>
<tr>
<th>User</th>
<th>Application</th>
<th>libibverbs</th>
<th>Hardware Layer Driver (ULD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>HW ULD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SoftRoCE ULD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ib_core</th>
<th>HW KLD</th>
<th>SoftRoCE KLD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>UDP: 4791</td>
<td>netdev</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware (HW)</th>
<th>RDMA NIC</th>
<th>Eth NIC</th>
</tr>
</thead>
</table>

-Maksym Planeta: “MigrOS”
Implementation

• RoCEv2 – InfiniBand protocol over UDP 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

<table>
<thead>
<tr>
<th>User</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>libibverbs</td>
</tr>
<tr>
<td></td>
<td>HW ULD</td>
</tr>
<tr>
<td></td>
<td>SoftRoCE ULD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ib_core</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HW KLD</td>
</tr>
<tr>
<td></td>
<td>SoftRoCE KLD</td>
</tr>
<tr>
<td></td>
<td>UDP: 4791</td>
</tr>
<tr>
<td></td>
<td>netdev</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HW</th>
<th>RDMA NIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eth NIC</td>
</tr>
</tbody>
</table>

Maksym Planeta: “MigrOS”
### Implementation

- **RoCEv2** – InfiniBand protocol over UPD port
- **SoftRoCE** – software RoCEv2 implementation
  - ✗ Performance overhead
  - ✓ Easy to change the protocol
  - ✓ Feature-full implementation

<table>
<thead>
<tr>
<th>User</th>
<th>HW ULD</th>
<th>SoftRoCE ULD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>libibverbs</em></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ib_core</th>
<th>SoftRoCE KLD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HW KLD</td>
<td>UDP: 4791</td>
</tr>
<tr>
<td></td>
<td></td>
<td>netdev</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HW</th>
<th>RDMA NIC</th>
<th>Eth NIC</th>
</tr>
</thead>
</table>
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

Diagram:

- User
  - Application
  - libibverbs
  - HW ULD
  - SoftRoCE ULD

- Kernel
  - ib_core
  - HW KLD
  - SoftRoCE KLD
  - UDP: 4791
  - netdev

- HW
  - RDMA NIC
  - Eth NIC

Maksym Planeta: “MigrOS” 9/13
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

Maksym Planeta: “MigrOS”
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!