The Future of MPI RMA

Joseph Schuchart <schuchart@icl.utk.edu>
JLESC’13, December 14, 2021
MPI Remote Memory Access

The Good:
- Fast random access to peer memory
- Zero-copy, no software protocols

The Bad:
- Static MPI Windows inflexible, dynamic windows limited
- Atomic operations are inefficient

The Ugly:
- Multi-threaded flushes not multi-threaded
- Operation ordering requires blocking

MPI Remote Memory Access

The Good:
- Fast random access to peer memory
- Zero-copy, no software protocols

The Bad:
- Static MPI Windows inflexible, **dynamic windows** limited
- **Atomic operations** are inefficient
MPI Remote Memory Access

The Good:
- Fast random access to peer memory
- Zero-copy, no software protocols

The Bad:
- Static MPI Windows inflexible, **dynamic windows** limited
- **Atomic operations** are inefficient

The Ugly:
- **Multi-threaded flushes** not multi-threaded
- **Operation ordering** requires blocking


The Good:
- Fast random access to peer memory
- Zero-copy, no software protocols

The Bad:
- Static MPI Windows inflexible, **dynamic windows** limited
- **Atomic operations** are inefficient

The Ugly:
- **Multi-threaded flushes** not multi-threaded
- **Operation ordering** requires blocking

MPI Remote Memory Access

The Good:
- Fast random access to peer memory
- Zero-copy, no software protocols

The Bad:
- Static MPI Windows inflexible, **dynamic windows** limited
- **Atomic operations** are inefficient

The Ugly:
- **Multi-threaded flushes** not multi-threaded
- **Operation ordering** requires blocking

Proposal: MPI Memory Handles

No good implementation of dynamic windows!
Proposal: MPI Memory Handles

No good implementation of dynamic windows!

![Graph showing latency vs transfer size for different MPI implementations.](image-url)
Proposal: MPI Memory Handles

No good implementation of dynamic windows!

![Graph showing latency versus transfer size for different MPI implementations]

- Open MPI (allocate)
- Open MPI (dynamic)
- MVAPICH (allocate)
- MVAPICH (dynamic)
- MPICH (allocate)
- MPICH (dynamic)

![Bar chart comparing put + flush (sleep) latency for different MPI implementations]
Proposal: MPI Memory Handles

- Dynamically register memory for RMA
- Provide life-time guarantees for attached memory
Proposal: MPI Memory Handles

- Dynamically register memory for RMA
- Provide life-time guarantees for attached memory

<table>
<thead>
<tr>
<th>Transfer Size [B]</th>
<th>Latency [us]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
</tr>
<tr>
<td>128</td>
<td>101</td>
</tr>
<tr>
<td>1024</td>
<td>102</td>
</tr>
<tr>
<td>8192</td>
<td>103</td>
</tr>
<tr>
<td>65536</td>
<td>10</td>
</tr>
<tr>
<td>524288</td>
<td>100</td>
</tr>
<tr>
<td>4194304</td>
<td>101</td>
</tr>
</tbody>
</table>

Process A

```c
char memhandle[MPI_MAX_MEMORY_HANDLE_SIZE]
MPI_Send(memhandle, size, B)
MPI_Recv(memhandle, ...
```

Process B

```c
MPI_Send(signal, A)
MPI_Recv(signal, B)
```

Diagram showing memory handle usage and operations.
Multi-threaded Flushes

- Multiple threads may flush independent operations
- MPI RMA operates at **process-scope**
Multi-threaded Flushes

- Multiple threads may flush independent operations
- MPI RMA operates at **process-scope**
- Proposal: Request **thread-scope synchronization**
  - Info key "mpi_win_scope" set to "thread"
  - MPI_Win_get_info to query support
  - Process-scope ⊇ Thread-scope
Multi-threaded Flushes

- Multiple threads may flush independent operations
- MPI RMA operates at **process-scope**
- Proposal: Request **thread-scope synchronization**
  - Info key “mpi_win_scope” set to “thread”
  - `MPI_Win_get_info` to query support
  - Process-scope ⊇ Thread-scope

![Graph showing latency versus number of threads for different flush scenarios]
Multi-threaded Flushes

- Multiple threads may flush independent operations
- MPI RMA operates at **process-scope**
- Proposal: Request **thread-scope synchronization**
  - Info key "mpi_win_scope" set to "thread"
  - MPI_Win_get_info to query support
  - Process-scope ⊇ Thread-scope
Operation Ordering

Ordering

- Only guaranteed for accumulate on same memory
- Otherwise requires **flush**
- **Example**: Ordered put operations

```
Put
Target Process
Flush
Rput
Test
Test
```
Operation Ordering

Ordering

- Only guaranteed for accumulate on same memory
- Otherwise requires flush
- Example: Ordered put operations

Proposal: Operation Ordering Info Key

- Knowing which operations to order ahead of time
- Set info key “mpi_win_order” to “true”
- Order all operations issued on that window
- Check support using MPI_Win_get_info

---

Transfer Size [B] | Latency [us]
---|---
10 | 1
100 | 100
101 | 101
102 | 102
103 | 103

Open MPI (flush) | Open MPI (no sync) | Open MPI (ordered)
Window Duplication

Currently:

- Window handles tied to **distinct resources**
- Info keys set on construction
- Subject to change using `MPI_Win_set_info`

```c
/* Duplicate window handle with different info values. Both window handles are used to access the same memory. */

int MPIX_Win_dup_with_info(
    MPI_Win parentwin,
    MPI_Info info,
    MPI_Win *newwin);
```
Window Duplication

Currently:
- Window handles tied to distinct resources
- Info keys set on construction
- Subject to change using `MPI_Win_set_info`

Mix & match info keys
- Duplicate window handle with different info values
- Access same resources

```c
/* Duplicate window handle with different info values. Both window handles are used to access the same memory. */
int MPIX_Win_dup_with_info(
    MPI_Win parentwin,
    MPI_Info info,
    MPI_Win *newwin);
```
Other Proposals

- Signal **intended atomic use** and query hardware support
- Target notification\(^1\)
- Remote-completing request-based operations (e.g., `MPI_Rrput`?)

Conclusions

MPI RMA

- Plenty potential for efficient communication
- Rethink parts of design
- Additions for usability & performance

Conclusions

MPI RMA

- Plenty potential for efficient communication
- Rethink parts of design
- Additions for usability & performance


The MPI RMA-WG is looking for feedback and input for redesigning MPI RMA!

mpiwg-rma@lists.mpi-forum.org
Thank you!

schuchart@icl.utk.edu

https://github.com/mpiwg-rma/rma-issues/issues/