On-the-Fly Data Race Detection in MPI One-Sided Communication

Presentation – Master Thesis

Simon Schwitanski (schwitanski@itc.rwth-aachen.de)
Joachim Protze (protze@itc.rwth-aachen.de)
Prof. Dr. Matthias S. Müller (mueller@itc.rwth-aachen.de)

SPPEXA Annual Plenary Meeting 2018
March 21, 2018
Motivation

- Traditional MPI communication model is two-sided
- Modern HPC cluster networking architectures: RDMA
  - Increasing popularity of one-sided communication models
- MPI Remote Memory Access (RMA)
  - Used in implementations of PGAS paradigms
- **Problem:** MPI RMA programming error-prone
- New kind of programming mistakes: **Data Races**
### Motivation

<table>
<thead>
<tr>
<th>Process A</th>
<th>Process B</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPI_Barrier</strong></td>
<td><strong>window location X</strong></td>
</tr>
<tr>
<td><strong>MPI_Win_lock(B)</strong></td>
<td><strong>MPI_Barrier</strong></td>
</tr>
<tr>
<td><code>buf = 42</code></td>
<td></td>
</tr>
<tr>
<td><strong>MPI_Put(&amp;buf, B, X)</strong></td>
<td><strong>print(X)</strong></td>
</tr>
<tr>
<td><strong>MPI_Win_unlock(B)</strong></td>
<td></td>
</tr>
<tr>
<td><strong>MPI_Barrier</strong></td>
<td><strong>MPI_Barrier</strong></td>
</tr>
</tbody>
</table>

- Data race between process A and B at window location X
- How can we detect such data races?
Thesis Goals

- Classification of data races in MPI RMA
- Design of an on-the-fly data race detection model
- Implementation in correctness checking framework MUST
- Evaluation with different test cases
Data Race Classes

MPI RMA Data Race

- Process-Local
  - Local access and API calls
  - API calls only
- Across Processes
  - Local access and API calls
  - API calls only

- Memory access tracking needed
- Tracking of API calls sufficient
### Race Detection – Formalization

- Formalize MPI RMA semantics
  - Hoefler et al. “Remote Memory Access Programming in MPI-3”
  - Track consistency order $\xrightarrow{co}$ (memory synchronization)
  - Track happens-before order $\xrightarrow{hb}$ (process synchronization)

<table>
<thead>
<tr>
<th>Process A (Origin)</th>
<th>Process B (Target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI_Barrier</td>
<td>window location $X$</td>
</tr>
<tr>
<td>MPI_Win_lock(B)</td>
<td>MPI_Barrier</td>
</tr>
<tr>
<td>buf = 42</td>
<td></td>
</tr>
<tr>
<td>MPI_Put(&amp;buf, B, X)</td>
<td>print($X$)</td>
</tr>
<tr>
<td>MPI_Win_unlock(B)</td>
<td></td>
</tr>
<tr>
<td>MPI_Barrier</td>
<td>MPI_Barrier</td>
</tr>
</tbody>
</table>
Race Detection – Concurrent Regions

- Local access: Origin operation / Remote access: Target operation
- Identify *concurrent regions* of origin and target operations
  - Cover *earliest* and *latest* point in time when operation can take place

<table>
<thead>
<tr>
<th>Process A (Origin)</th>
<th>Process B (Target)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>window location X</td>
</tr>
<tr>
<td>MPI_Barrier</td>
<td>MPI_Barrier</td>
</tr>
<tr>
<td>MPI_Win_lock(B)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>buf = 42</td>
<td></td>
</tr>
<tr>
<td>MPI_Put(&amp;buf, B, X)</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>print(X)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MPI_Win_unlock(B)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

On-the-Fly Data Race Detection in MPI One-Sided Communication
Simon Schwitanski | March 21, 2018
## Implementation

- **Tracking of MPI calls: MUST correctness checking framework**
  - Capture memory and process synchronization in MPI programs
  - Determine concurrent regions of MPI RMA operations

- **Actual data race detection: ThreadSanitizer**
  - Runs locally on each process
  - Captures local memory accesses
  - Provides annotation API

- **Idea: Annotate concurrent regions in ThreadSanitizer**
MUST Infrastructure

Process A → Tool Threads
Process B → Tool Threads
Process C → Tool Threads
Process D → Tool Threads

Application Threads

Tool Processes
Conclusion

• MPI RMA introduces one-sided communication model
• Data races in MPI RMA as new error class
• Main concepts of on-the-fly data race detection
  – Tracking memory consistency and happens-before order
  – Identification and annotation of concurrent regions of RMA operations
• Prototype implementation in MUST and ThreadSanitizer
  – Potential of false positives (benign races)
  – Detects representatives of defined data race classes