



Finding Memory errors in MPI applications

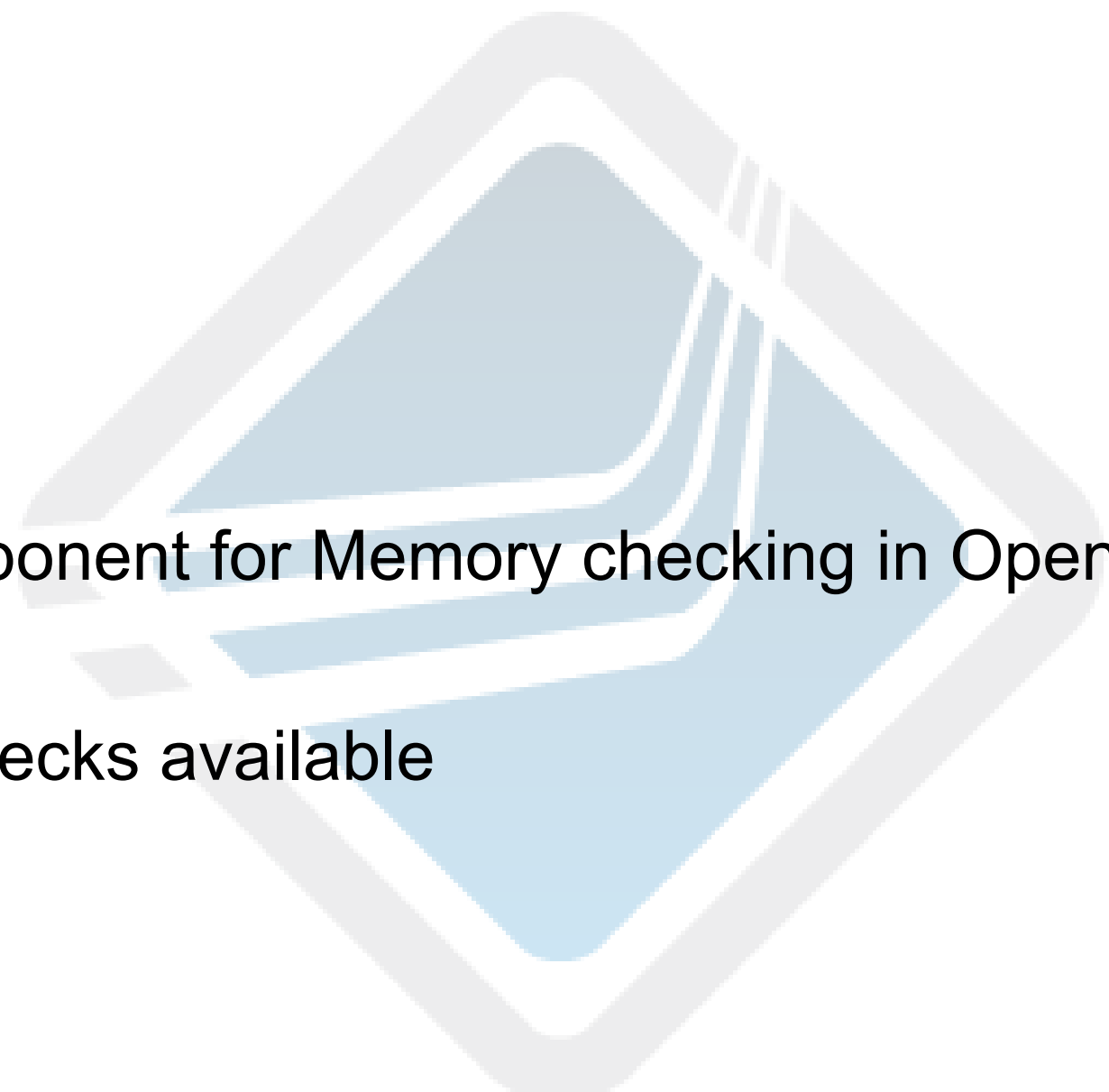
Rainer Keller – HLRS

Shiqing Fan – HLRS

Michael Resch – HLRS

Cisco Booth Talk, SC2010, New Orleans

Overview

- Introduction to
 - § MPI 2.2
 - § Open MPI
 - § Valgrind
 - Memchecker Component for Memory checking in Open MPI
 - MPI application Checks available
 - Conclusion
- 

Introduction to MPI-2.2

- MPI is the standard for efficient, scalable parallelization paradigm and has been shown to work on PFlops machines (IBM BlueGene, Cray XEs, Linux)
- The current official standard version is MPI-2.2.
- E.g.: Usage of buffers, that are to be send immediately (non-blocking):
 - § Old: **may not be** read or written to by the application.
 - § New: **may be** read from by the application.
- This affects the usage of the memchecker tool, as we will see.

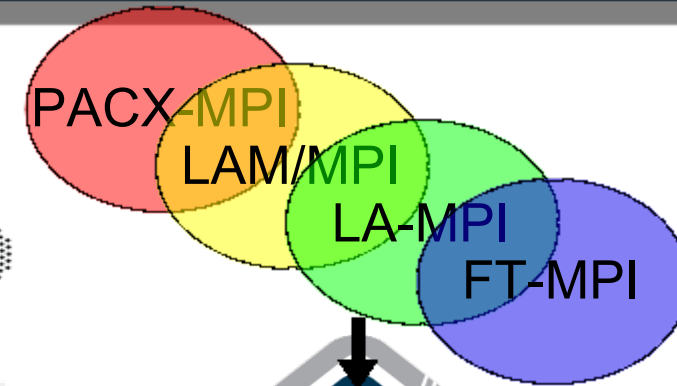
About Open MPI

- Features of Open MPI:

- § Full MPI-2.1 implementation,
- § Fast, reliable and extensible,
- § Production-grade code quality as a base for research.

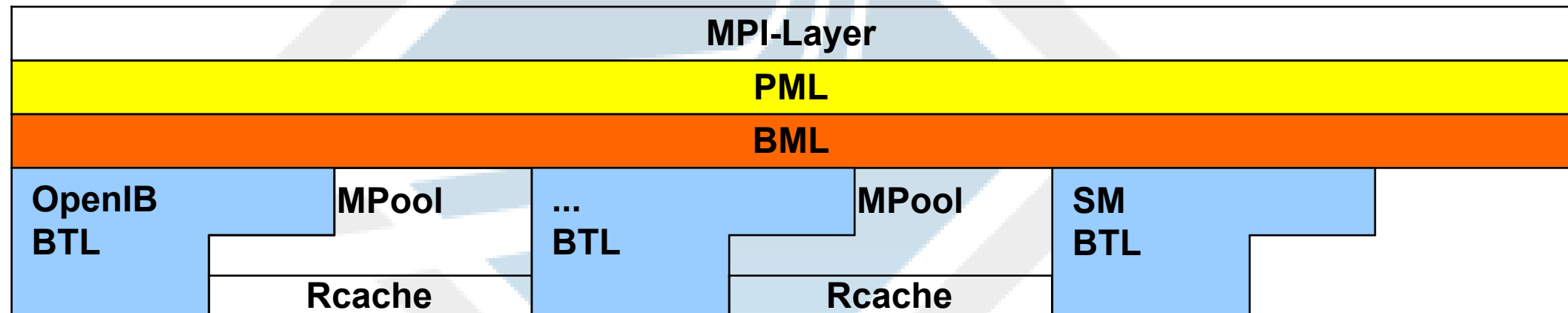
- Current status:

- § Stable: v1.4.3 since Oct. 5th.
- § Feature: v1.5 since Oct. 10th.



Open MPI Architecture

- The Modular Component Architecture (MCA -- think plugin) allows:
 - § Dynamically load available modules and check for hardware
 - § Select best modules and unload others (e.g. if hw not available)
 - § Fast indirect calls into each component.



- Very versatile setup for varying installations (ship one RPM)
- Allows easy integration of new functionality

Introduction into Valgrind

- An Open-Source Debugging & Profiling tool
- Works with dynamically & statically linked applications
- Emulates CPU:
i.e. executes instructions on a synthetic x86/Opteron/Power
- It's easily configurable to ease debugging & profiling through *tools*:
 - § Cachegrind: A memory & cache profiler
 - § Helgrind: Find Races in multithreaded programs
 - § Callgrind: A Cache & Call-tree profiler
 - § **Memcheck**: Every memory access is being checked...



Introduction into Valgrind



- Memcheck tool scans for:
 - § Use of uninitialized memory
 - § Malloc Errors:
 - Usage of free'd memory
 - Double free
 - Reading/writing past malloc'd memory
 - Lost memory pointers
 - Mismatched malloc/new & free/delete
 - § Stack write errors
 - § Overlapping arguments to system functions like `memcpy`.
- Why not use this functionality for MPI checking purposes?

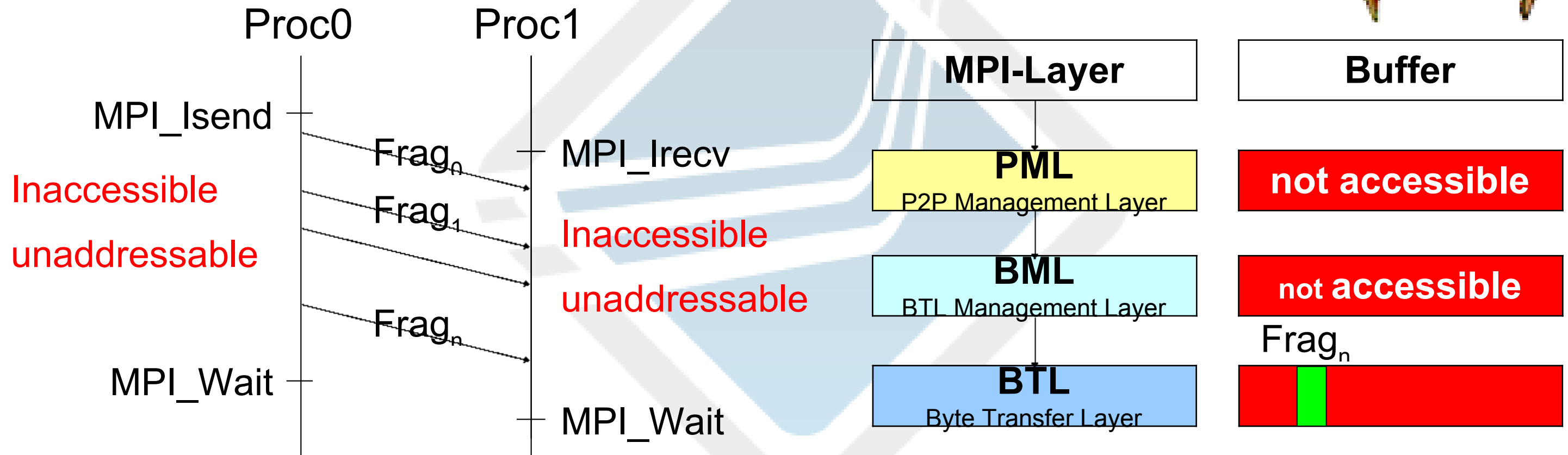
Open MPI valgrind extension

- Detect application's memory violation of MPI-standard:
 - § Application's usage of undefined data
 - § Application's memory access due to MPI-semantics
- Detect Non-blocking/One-sided communication errors:
 - § Functions in BTL layer for both communications
 - § Set memory accessibility independent of MPI operations
 - § i.e. only set accessibility for the fragment to be sent/received
- MPI object checking:
 - § Check definedness of MPI objects that passing to MPI API
 - § `MPI_Status`, `MPI_Comm`, `MPI_Request` and `MPI_Datatype`
 - § Could be disabled for better performance



Open MPI memchecker

- Non-blocking send/receive buffer error checking



Open MPI memchecker



- Access to buffer under control of MPI:

```
MPI_Irecv (buffer, SIZE, MPI_CHAR, ..., &request);
```

```
buffer[1] = 4711;
```

```
MPI_Wait (&request, &status);
```

- Side note: CRC-based methods do not reliably catch these cases.

- Memory that is outside receive buffer is overwritten :

```
buffer = malloc( SIZE * sizeof(MPI_CHAR) );
```

```
memset (buffer, SIZE * sizeof(MPI_CHAR), 0);
```

```
MPI_Recv(buffer, SIZE+1, MPI_CHAR, ..., &status);
```

- Side note: MPI-1, p21, rationale of overflow situations: “no memory that outside the receive buffer will ever be overwritten.”

Open MPI memchecker



- Usage of the Undefined Memory passed from Open MPI

```
MPI_Wait(&request, &status);  
if (status.MPI_ERROR != MPI_SUCCESS)
```

- Side note: This field should remain undefined.
 - § MPI-1, p22 (not needed for calls that return only one status)
 - § MPI-2, p24 (Clarification of status in single-completion calls).

- Write to buffer before accumulate is finished :

```
MPI_Accumulate(A, NROWS*NCOLS, MPI_INT, 1, 0, 1,  
              expose, MPI_SUM, win);
```

```
A[0][1] = 4711;
```

```
MPI_Win_fence(0, win);
```

Open MPI memchecker



- Non-blocking buffer accessed/modified before finished

```
MPI_Isend (buffer, SIZE, MPI_INT, ..., &request);
```

```
buffer[1] = 4711;
```

```
MPI_Wait (&req, &status);
```

- The standard does **now** allow **read** access:

```
MPI_Isend (buffer, SIZE, MPI_INT, ..., &request);
```

```
result[1] = buffer[1];
```

```
MPI_Wait (&request, &status);
```

- Historic side note:

§ MPI-1, p30, Rationale for restrictive access rules; “allows better performance on some systems”.

Open MPI memchecker extension

- To allow this checking (and more), valgrind extensions:

```
MPI_Isend (buffer, SIZE, MPI_INT, ..., &request);  
result[1] = buffer[1];  
MPI_Wait (&request, &status);
```



buffer[1]

Thank You

- Thank You very much!

