Compiler Attributes of MPI Functions

TECHNISCHE UNIVERSITÄT DARMSTADT

Tim Jammer **Adrian Schmidt** **Christian Bischof**

NHR4 CES NHR for Computational Engineering Science

SC Scientific Computing

tim.jammer@tu-darmstadt.de

https://github.com/AdrSchm/mpi-attributes-pass

Introduction

TECHNISCHE UNIVERSITÄT DARMSTADT

- C and C++ programs use libraries via header files
  - Oftentimes only contain declarations
- Missing information
  - Especially about memory access behavior
  - Important for pointer parameters

⇒ Use compiler attributes annotated to declarations to provide hints

Motivation: Example Optimization

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int var = 42;
2  int out = foo(&var);
3  var = var + 1337;
```

Without information how `foo` uses the pointer:

```
1  mov  dword ptr [rsp + 4], 42
2  lea  rdi, [rsp + 4]
3  call foo
4  mov  eax, dword ptr [rsp + 4]
5  add  eax, 1337
```

With information that `foo` only **reads**:

```
1  mov  dword ptr [rsp + 4], 42
2  lea  rdi, [rsp + 4]
3  call foo
4
5  mov eax, 1379
```

LLVM Attributes

TECHNISCHE UNIVERSITÄT DARMSTADT

- `memory`
  - Specifies how memory is accessed by a function
  - Parameters `argmem`, `inaccessiblemem`
  - Mostly used as `memory(argmem: readwrite, inaccessiblemem: readwrite)`
- `readonly`, `writeonly`, `readnone`
  - Annotated to pointer parameters
  - Allow more precise description of memory access behavior
- `nocapture`
  - Annotated to pointer parameters
  - Capturing $\approx$ Storing the pointer somewhere
- `nofree`
  - Function and parameter attribute

Blocking Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

- ✓ Buffer only read (or written for receive)
- ✓ Buffer not used after call returns

```
; Function Attrs: nofree memory(argmem: readwrite, inaccessiblemem: readwrite)
declare i32 @MPI_Send(
            ptr nocapture readonly %buf,
            i32 %count, ptr %datatype, i32 %dest, i32 %tag, ptr %comm) #2
```

- ? `memory` attribute specifies that no accessible memory is used
- ? May not seem correct because of `MPI_Buffer_attach`
  - ⇒ Application using this buffer is undefined behavior

Example

MPI_Buffer_attach

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int message = 1234;
2  int buffer_size = (MPI_BSEND_OVERHEAD + sizeof(int));
3  char *buffer = malloc(buffer_size);
4  MPI_Buffer_attach(buffer, buffer_size);
5
6  MPI_Bsend(&message, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
7  // Using the buffer here is undefined behavior
8
9  MPI_Buffer_detach(&buffer, &buffer_size);
10 free(buffer);
```

Nonblocking Communication

TECHNISCHE UNIVERSITÄT DARMSTADT

- ✓ Message buffer only read (or written for receive)
- ? Buffer not used after call returns

```
; Function Attrs: nofree memory(argmem: readwrite, inaccessiblemem: readwrite)
declare i32 @MPI_Isend(
            ptr nocapture readonly %buf,
            i32 %count, ptr %datatype, i32 %dest, i32 %tag, ptr %comm,
            ptr nocapture writeonly %request) #2
```

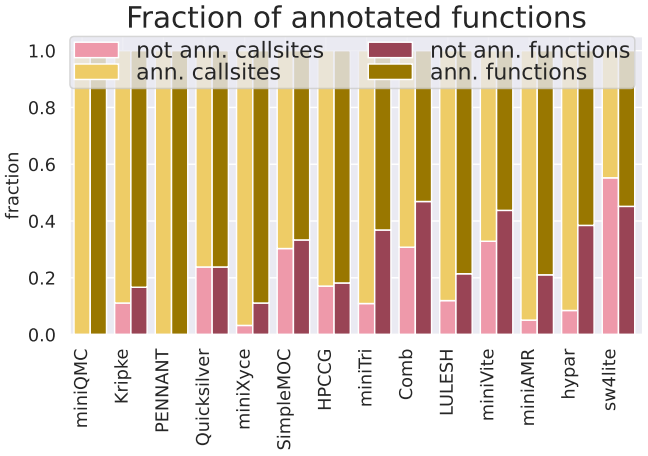
⇒ `nocapture` reflects semantics of the standard:

⇒ Access before completion (`MPI_Wait`) is undefined behavior

Evaluation

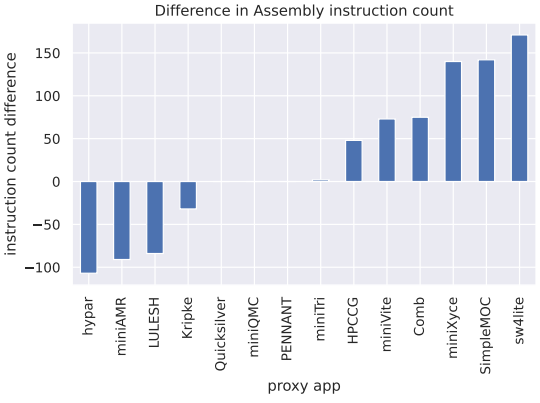
TECHNISCHE UNIVERSITÄT DARMSTADT

- Tested with Exascale Proxy Applications
- Annotate 95 MPI functions
- Annotated most used functions
- Not annotated: Creation of datatypes or communicators



NHR4CES

Evaluation



- E.g. LULESH: removal of instructions
- E.g. SimpleMOC: addition of instructions
  - Amount of executed instructions decreases

LULESH

Removed Memory Accesses

Left: Without attributes

```
1  mov   rsi, rsp
2  mov   rdi, rbx
3  call  MPI_Comm_size@PLT
4  lea   rsi, [rsp + 4]
5  mov   rdi, rbx
6  call  MPI_Comm_rank@PLT
7  ; [...]   Load size:
8  mov   eax, dword ptr [rsp]
9  add   eax, 10
10 ; [...]   Load rank:
11 mov   edx, dword ptr [rsp + 4]
12 call  ParseCommandLineOptions
13 mov   esi, dword ptr [rsp + 4]
14 mov   eax, dword ptr [rsp + 40]
15 or    eax, esi
16 ; [...]   re-load size from memory:
17 mov   esi, dword ptr [rsp]
```

Right: With Attributes

```
1  lea    rsi, [rsp + 76]
2  mov    rdi, rbx
3  call   MPI_Comm_size@PLT
4  lea    rsi, [rsp + 72]
5  mov    rdi, rbx
6  call   MPI_Comm_rank@PLT
7  ; [...]   Load size:
8  movsxd rbx, dword ptr [rsp + 76]
9  lea    eax, [rbx + 10]
10 ; [...]   Load rank:
11 mov    ebp, dword ptr [rsp + 72]
12 mov    edx, ebp
13 call   ParseCommandLineOptions
14 mov    eax, dword ptr [rsp + 32]
15 or     eax, ebp
16 ; [...]   reuse of size from register:
17 mov    esi, ebx
```

TECHNISCHE UNIVERSITÄT DARMSTADT

LULESH

Resolved Function Pointer

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  MPI_Wait(&domain.recvRequest[pmsg], &status);
2  for (Index_t fi = 0; fi < xferFields; ++fi) {
3    Domain_member dest = fieldData[fi];
4    for (Index_t i = 0; i < opCount; ++i) {
5      (domain.*dest)(dx*dy*(dz - 1) + i)
6                  = srcAddr[i];
7    }
8    srcAddr += opCount;
9  }
```

- Outer loop is unrolled
- `fieldData` contains function pointers
  - Function calls in inner loop(s) can theoretically be resolved at compile time
- Adding attributes allows to resolve first pointer statically
  - Removes 22 instructions
  - ... only 7 of which are actually executed
  - Instructions are needed for potentially virtual calls

SimpleMOC
Condition Hoisting

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  MPI_Comm_rank(MPI_COMM_WORLD, &mype);
2  // read input
3  if(mype == 0)
4      print_input_summary(input);
5  // inizialization
6  for(int i = 0; i < num_iters; i++) {
7      // calculation
8      if(mype == 0)
9          printf("keff = %f\n", keff);
10 }
11 // cleanup
12 if(mype == 0)
13     border_print();
14     center_print("RESULTS SUMMARY", 79);
15     // more output
16 }
```

- Many print function calls, only executed by one process

- x Without attributes: Condition evaluated every time
  - Not necessary since `mype` does not change

SimpleMOC
Condition Hoisting

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  MPI_Comm_rank(MPI_COMM_WORLD, &mype);
2  // read input
3  if(mype == 0)
4      print_input_summary(input);
5  // inizialization
6  for(int i = 0; i < num_iters; i++) {
7      // calculation
8      if(mype == 0)
9          printf("keff = %f\n", keff);
10 }
11 // cleanup
12 if(mype == 0)
13     border_print();
14     center_print("RESULTS SUMMARY", 79);
15     // more output
16 }
```

- Many print function calls, only executed by one process

× Without attributes: Condition evaluated every time
  - Not necessary since `mype` does not change

✓ With attributes: Hoists condition evaluation
  - Duplicates code region
  - Reduced amount of instructions executed
  - Relevant attribute: `nocapture`

miniAMR
Reduction

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  do {
2    //  calculation
3    for (i = 0; i < num_pes; i++) {
4      n1 += from[i];
5    }
6    MPI_Allreduce(&n1, &n, 1,
7     MPI_INT, MPI_SUM, MPI_COMM_WORLD);
8  } while (n && k != n);
```

- Many unnecessary memory accesses
- Assumes that n1 can be read at any time

```
1  .LBB7_59:
2   add esi, dword ptr [rcx + 4*rdi]
3   mov dword ptr [rsp + 12], esi
4   add esi, dword ptr [rcx + 4*rdi + 4]
5   mov dword ptr [rsp + 12], esi
6   add esi, dword ptr [rcx + 4*rdi + 8]
7   mov dword ptr [rsp + 12], esi
8   add esi, dword ptr [rcx + 4*rdi + 12]
9   mov dword ptr [rsp + 12], esi
10  add rdi, 4
11  cmp rax, rdi
12  jne .LBB7_59
```

TECHNISCHE UNIVERSITÄT DARMSTADT

- Relevant attribute: `nocapture`

✓ Only writes result to memory once

✓ Attributes would still allow unrolling, but compiler decided not to unroll

```
1  .LBB7_53:
2    add esi, dword ptr [rcx + 4*rdx]
3    inc rdx
4    cmp rax, rdx
5    jne .LBB7_53
6  .LBB7_54:
7    mov dword ptr [rsp + 156], esi
```

Performance

- No measurable difference
- Optimizations occur where MPI is used
  - Communication separated from computational kernel
  - Only a small part of the run time

Performance

TECHNISCHE UNIVERSITÄT DARMSTADT

- No measurable difference
- Optimizations occur where MPI is used
  - Communication separated from computational kernel
  - Only a small part of the run time
- Synthetic examples can show general effectiveness
  - Performance improvement of 16% (of 2.7 seconds) when using code hoisting
  - Performance improvement of 11% when removing unnecessary memory accesses during reduction

```
1  for (int n = 0; n < num_iters; ++n) {
2      for (int i = 0; i < ARRAY_SIZE; ++i) {
3          array[i] = array[i] + (n * 42.1337);
4      }
5      if (n % ITER_TO_PRINT == 0 && rank == 0) {
6          printf("Iter %d : %f\n", n, array[0]);
7      }
8  }
```

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Annotating attributes influences code generation
- Several additional optimizations possible
- Different attributes have different impacts
- No measurable performance benefit in analyzed applications
  - Depends on specific application
- Usage generally sensible
  - ✓ Improves code
  - ✓ Does not introduce additional overhead

`https://github.com/AdrSchm/mpi-attributes-pass`