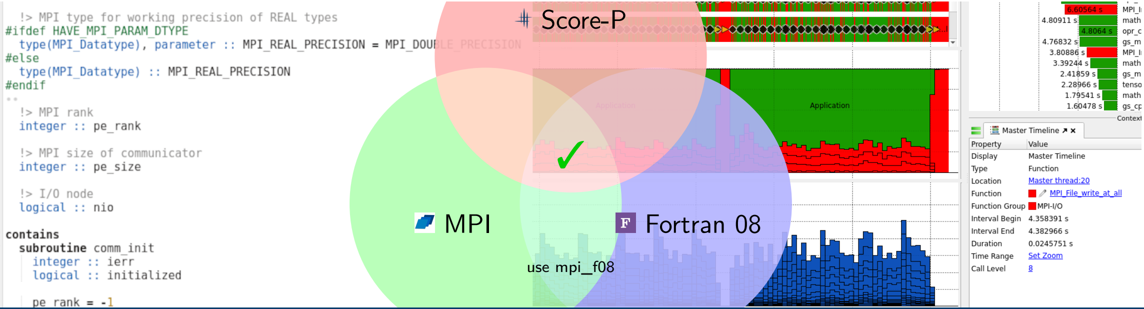


MPI F08 support in Score-P
15th IPTW

19.09.2024 | Gregor Corbin | JSC

TECHNISCHE UNIVERSITÄT DRESDEN

Score-P
Scalable performance measurement infrastructure for parallel codes

JÜLICH Forschungszentrum

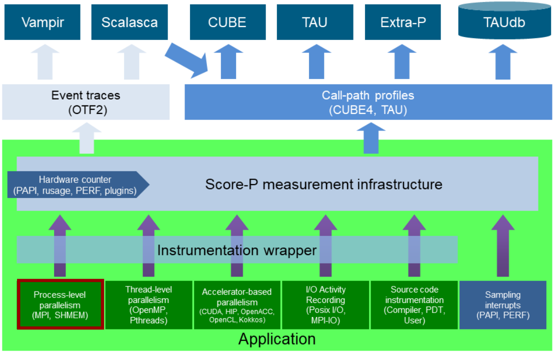
Agenda

- What is Score-P?
- Current (released) state of MPI Fortran support in Score-P
- New Fortran 2008 MPI wrappers in Score-P
- Wrapper generator
- Development status

Score-P

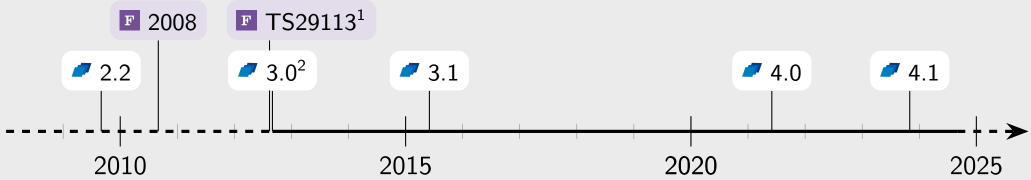
Instrumentation and measurement infrastructure

- Scalable instrumentation tool
- Support for multi-process, thread-parallel, and accelerator-based paradigms
- Support for additional metrics (I/O, HW counters, . . .)
- Measurement
  - Profile generation
  - Event trace recording
- Support for C, C++, Fortran, and Python



MPI and Fortran

Some history



- Fortran bindings → `use mpi` or `include 'mpif.h'`
- Fortran 2008 bindings → `use mpi_f08`

[1] TS29113 Further Interoperability of Fortran with C

[2] `use mpi_f08`

Why you should 'use mpi_f08'

- Recommended by the MPI standard[3]
- Fully compliant with the Fortran standard (if TS29113 is supported)
- Pass MPI handles in a type safe manner
- Pass noncontiguous buffers to nonblocking MPI routines (if `MPI_SUBARRAYS_SUPPORTED`)
- Use `ASYNCHRONOUS` to protect buffers of nonblocking operations
- Use large-count overloads
- `ierror` is optional

[3] Message Passing Interface Forum (Nov. 2023). MPI: A Message-Passing Interface Standard Version 4.1. URL: https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf.

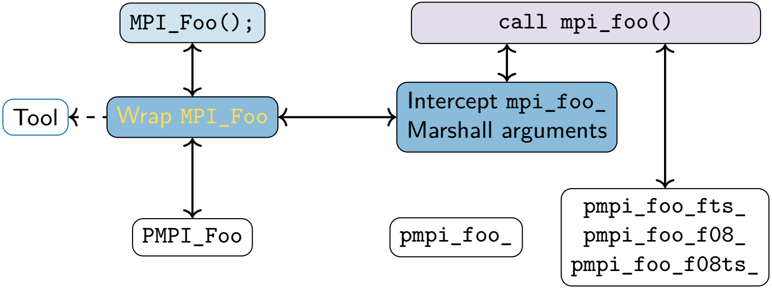
This might have happened to you

```fortran
program awesome_f08_program
! Use nice features from the Fortran 2008 bindings
use :: mpi_f08


call MPI_Init()

...
```

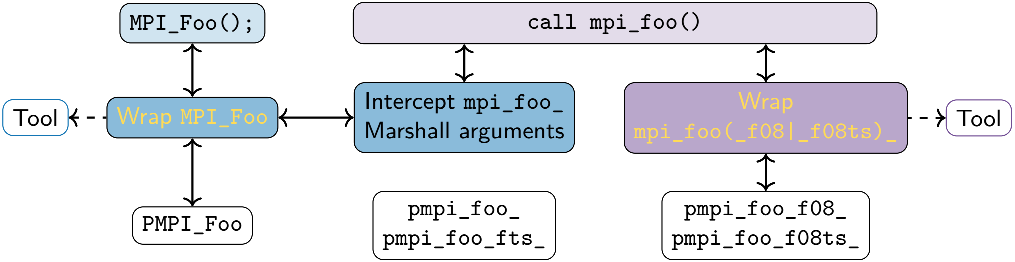
```
$ scorep mpifort awesome_f08_program.F90
$ mpirun -n 128 ./a.out
[Score-P] 'USE mpi_f08' not supported yet,
          consider switching to 'USE mpi'.
```

PMPI interface and wrappers

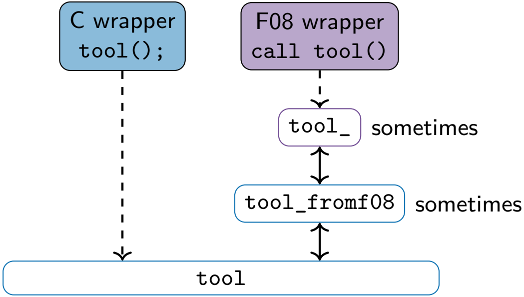
Current (released) state in Score-P



The new F08 wrappers



Fortran 2008 interface to Score-P



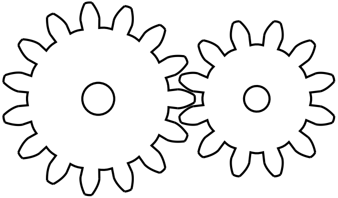
Fortran to C issues[4]

	Problem	Supported with	
		Fortran	Fortran 2008
1	Logical	✗	✓
2	Error return	✓	✓
3	Fortran only routines	✗	✓
4	Callbacks	✗	✓
5	Attributes	✗	✓
6	Array descriptors	✗	✓
7	MPI handles	✓	✓
8	MPI constants	✓	✓
9	Character strings	✓	✓
10	Status object	✓	✓
11	Array indices	✓	✓
12	Info strings	✗	✓

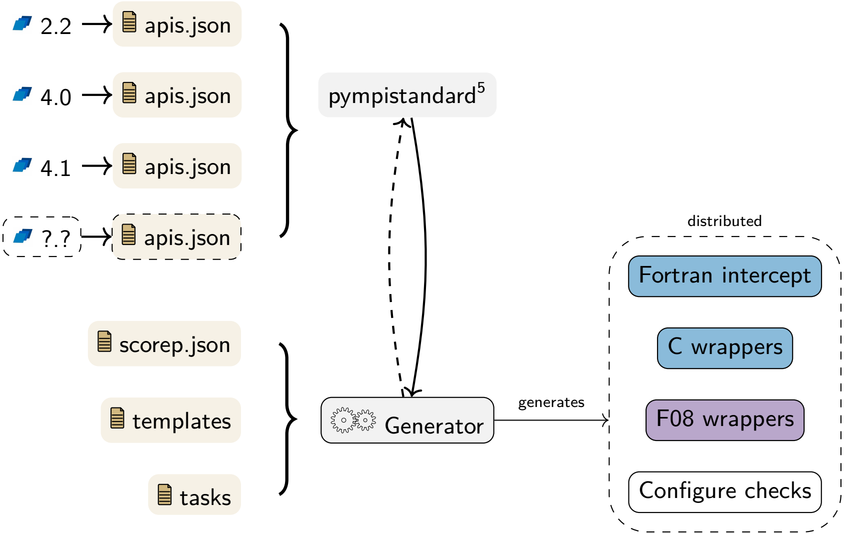
[4] Rasmussen, Schulz, and Mohror 2016, Allowing MPI tools builders to forget about Fortran.

Some numbers

- The current MPI standard (4.1) has over 1100 pages
- Number of procedures: 491
  - C wrappers: 433
  - Fortran wrappers: 408
  - F08 wrappers: 393
- Lines of wrapper code:
  - C: 55000
  - F08: 46000

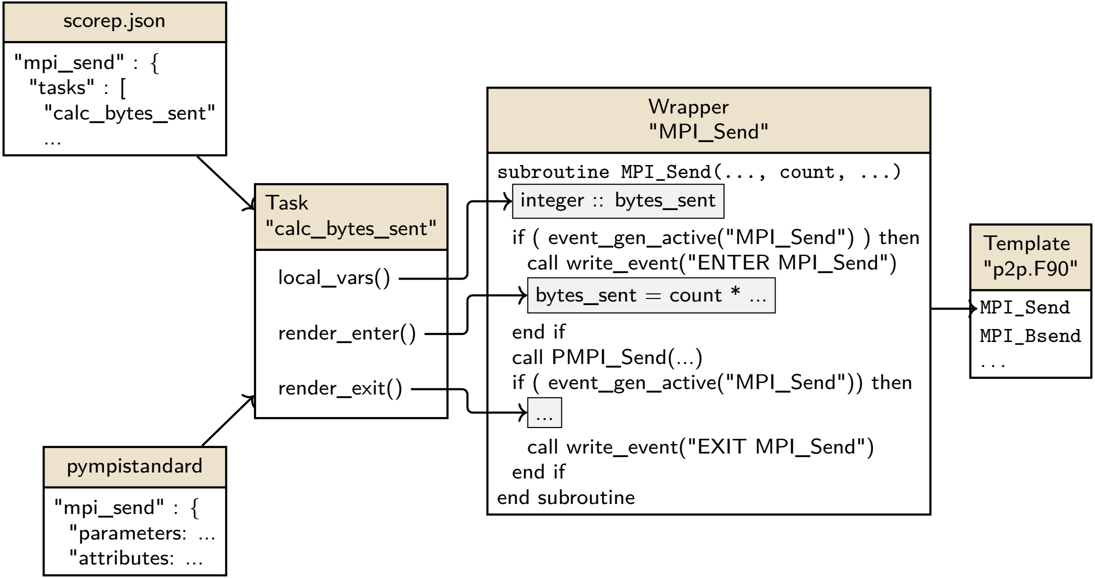


Wrapper Generator



[5] Martin Ruefenacht (2024). pympistandard. URL: https://github.com/mpi-forum/pympistandard.

Wrapper rendering



Development status

Features

- all MPI 4.0 procedures wrapped
- same features as the C wrappers
- support for large-count procedures
- support for OpenMPI, MPICH

Tested with

- MPICH test suite (`https://github.com/pmodels/mpich`)
- Neko[6] (`https://github.com/ExtremeFLOW/neko`)
- EPIC[7] (`https://github.com/EPIC-model/epic`)

[6] Jansson et al. 2024.

[7] Frey, Dritschel, and Böing 2022.

Try it

Available as development version.[8]

Feedback welcome!

- Score-P mailing list: support@score-p.org
- Direct: g.corbin@fz-juelich.de

Questions?

[8] https://perftools.pages.jsc.fz-juelich.de/cicd/scorep/branches/MR358/latest.tar.gz

References I

- EPIC: Elliptical parcel-in-cell (2024). URL: https://github.com/EPIC-model/epic.
- Frey, Matthias, David Dritschel, and Steven Böing (2022). "EPIC: The Elliptical Parcel-In-Cell method". In: Journal of Computational Physics: X 14, p. 100109. ISSN: 2590-0552. DOI: https://doi.org/10.1016/j.jcpx.2022.100109. URL: https://www.sciencedirect.com/science/article/pii/S2590055222000051.
- Jansson, Niclas et al. (2024). In: Computer and Fluids 275. DOI: '10.1016/j.compfluid.2024.106243'. URL: https://www.neko.cfd.
- Message Passing Interface Forum (Nov. 2023). MPI: A Message-Passing Interface Standard Version 4.1. URL: https://www.mpi-forum.org/docs/mpi-4.1/mpi41-report.pdf.
- MPICH developers (2024). MPICH repository. URL: https://github.com/pmodels/mpich.
- Neko Extreme Flow (2024). URL: https://github.com/ExtremeFLOW/neko.

References II

Rasmussen, Soren, Martin Schulz, and Kathryn Mohror (2016). “Allowing MPI tools builders to forget about Fortran”. In: Proceedings of the 23rd European MPI Users’ Group Meeting. EuroMPI ’16. Edinburgh, United Kingdom: Association for Computing Machinery, pp. 208–211. ISBN: 9781450342346. DOI: 10.1145/2966884.2966889. URL: https://doi.org/10.1145/2966884.2966889.

Ruefenacht, Martin (2024). pympistandard. URL: https://github.com/mpi-forum/pympistandard.

Bonus slides

Listing 1: Fortran 2008 example with `use mpi_f08`

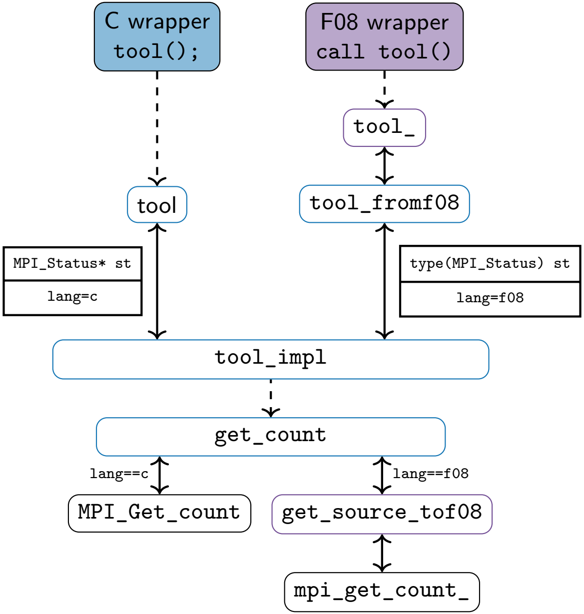
```fortran
subroutine comm_boundaries(comm, nx, ny, field, south, north)
    use :: mpi_f08

    integer :: nx, ny, south, north
    ! asynchronous attribute
    integer, asynchronous :: field(ny, nx)
    ! handle types
    type(MPI_Comm) :: comm
    type(MPI_Request) :: req(2)

    ! no copy to contiguous buffer needed
    call MPI_Isendrecv(field(ny - 1, :), nx, MPI_INTEGER, south, 0, &
                      &field(1, :), nx, MPI_INTEGER, north, 0, &
                      &comm, req(1)) ! no ierror

    call MPI_Isendrecv(field(2, :), nx, MPI_INTEGER, north, 0, &
                       &field(ny, :), nx, MPI_INTEGER, south, 0, &
                       &comm, req(2)) ! no ierror

    call MPI_Waitall(2, req, MPI_STATUSES_IGNORE)
    ! no MPI_F_sync_reg(field) needed
end subroutine
```

Listing 2: Fortran example with `use mpi`

```fortran
subroutine comm_boundaries(comm, nx, ny, field, south, north)
    use :: mpi

    integer :: nx, ny, south, north
    integer :: field(ny, nx)
    ! no types
    integer :: comm
    integer :: req(2)

    ! scratch buffers
    integer, dimension(nx) :: sendbuf1, sendbuf2, recvbuf1, recvbuf2

    sendbuf1 = field(ny - 1, :) ! copy to contig. scratch buffer
    call MPI_Isendrecv(sendbuf1, nx, MPI_INTEGER, south, 0, &
                       &recvbuf1, nx, MPI_INTEGER, north, 0, &
                       &comm, req(1), ierror) ! ierror must be present
    field(1, :) = recvbuf1 ! copy from contig. scratch buffer

    sendbuf2 = field(2, :)
    call MPI_Isendrecv(sendbuf2, nx, MPI_INTEGER, north, 0, &
                       &recvbuf2, nx, MPI_INTEGER, south, 0, &
                       &comm, req(2), ierror ) ! ierror must be present
    field(ny, :) = recvbuf2

    call MPI_Waitall(2, req, MPI_STATUSES_IGNORE)

    ! prevent further access of field from moving before the wait
    call MPI_F_sync_reg(field)
end subroutine
```

Listing 3: Example wrapper in Fortran 2008

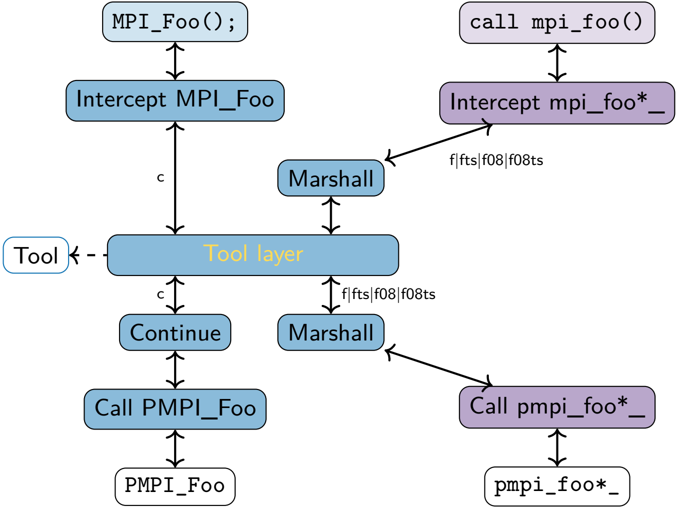
```fortran
subroutine mpi_send(buf, count, datatype, dest, tag, comm, ierror)
    ...
    integer, optional :: ierror

    ! Local variable declarations
    integer :: bytes, my_ierr

    if ( event_gen_active("MPI_Send") ) then
        call write_event("ENTER", "MPI_Send")
        ! Specific to MPI_Send:
        bytes=calc_bytes_sent(count, datatype)
        call write_event("SEND", bytes)
    end if

    ! <— Insert tool code here
    call PMPI_Send(buf, ...,  my_ierr)
    ! <— Insert tool code here

    if ( event_gen_active("MPI_Send") ) then
        ! <— Insert tool code here
        call write_event("EXIT", "MPI_Send")
    end if

    if (present(ierr)) ierr = my_ierr
end subroutine
```


C wrapper
tool();
F08 wrapper
call tool()
tool_
tool
tool_fromf08
MPI_Status* st
lang=c
type(MPI_Status) st
lang=f08
tool_impl
get_count
lang==c
lang==f08
MPI_Get_count
get_source_tof08
mpi_get_count_

Naming of MPI symbols in Fortran

- 2 bindings (Fortran, Fortran2008)
- 2 conventions for passing choice buffers (compiler extension, TS29113 type(*), dimension(..))
- normal/large-count
- 8 specific symbols for each MPI procedure
- variants for name mangling (one/two underscores, all caps/all lowercase)
- we could intercept all of them, but this is not practical: Implementations do not adhere to the standard: MPICH: mpi_isend_f08ts_large
- determine specific symbol names at configuration

WMPI architecture



Proposed tool-agnostic architecture