

An Environment for Interactive Parallel Programming with MPI and OpenMP

Christian Terboven

<terboven@itc.rwth-aachen.de>

<terboven@pm.de>



RWTHAACHEN
UNIVERSITY

Jonas Hahnfeld

<hahnjo@hahnjo.de>



Motivation and Project Goals

- Increasingly more students and fields of study are interested in parallel programming and data analysis
 - Learning success and motivation increase when effects of parallelism are experienced directly
 - Challenge: technical barriers
 - Access to Linux systems via the command line, using a batch system, using many new dev tools, ...

```
(c747764) login18-1.hpc.itc.rwth-aachen.de ~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE MODEL
c18n          up  3h-00:00:00    17  drain ncn[000,0415,0583]
c18n          up  3h-00:00:00    116  mix  ncn[009,0036,0839,0078,0808,0096-0997,0103,0111,0144,0225,0229,0234,0248-0249,0251-0253,0256,0280,
0297,0309,0325,0327,0329,0350,0358,0400,0417,0421,0435-0437,0448,0456-0460,0467,0474-0477,0501,0506-0512,0533,0537,0576-0577,0579-0582,0584,
0613,0648-0654,0677,0688,0727-0728,0833,0861,0872-0882,0886,0888,0899-0900,0934,0941-0943,0945,0949-0950,0959-0960,1000,1009-1011,1019-1020]
nrm1[001-003,039,079,147-148,150-152,283,288]
c18n          up  3h-00:00:00    437  alloc ncn[001-0008,0038,0073-0079,0087,0100,0102,0107-0110,0112-0143,0145-0152,0182,0217-0224,0226-0228,
0238-0239,0242-0244,0258,0254-0255,0261,0284,0286-0287,0289-0296,0300-0308,0310-0324,0326,0329-0354,0356-0359,0361-0368,0401-0414,0416,0418,
0420,0422-0434,0438-0439,0441-0455,0461,0463-0464,0473-0499,0504,0531,0532,0534-0535,0539-0540,0542-0543,0545,0548,0550,0573,0655-0656,0724-
0726,0730-0800,0803-0804,0806-0809,0883-0885,0887,0889-0898,0901-0904,0920-0933,0935-0936,0938-0940,0946-0948,0951-0958,0975-0977,0999-
1001,1004,1012-1016,1021-1032],nrm[004-005,009-030,040,073,149,185-202,204,206-207]
c18n          up  3h-00:00:00    607  idle ncn[010-0021,0023-0035,0037,0040-0069,0071-0072,0081-0084,0086,0089-0092,0094-0095,0098-0099,0101,
0104-0106,0153-0154,0183-0216,0257-0260,0262-0283,0285,0298-0299,0369-0399,0465-0470,0500-0502,0513-0539,0536,0538,0541,0544,0546-0547,0549,
0551-0572,0574-0575,0585-0612,0616,0617-0620,0624-0647,0657-0676,0678-0687,0689-0723,0729-0792,0801-0832,0834-0808,0878-0871,0905-0927,0937,0944,
0976],nrm[006-000,041-072,074-078,080-142,144-146,153-184,205]
c18n_low      up  3h-00:00:00    17  drain ncn[002,005,0039,0078,0614-0616,0621-0623,0602,0805,1005-1008],nrm1[043]
c18n_low      up  3h-00:00:00    116  mix  ncn[009,0036,0839,0078,0808,0096-0997,0103,0111,0144,0225,0229,0234,0248-0249,0251-0253,0256,0280,
0297,0309,0325,0327-0328,0350,0358,0400,0417,0421,0435-0437,0448,0456-0460,0467,0474-0477,0501,0506-0512,0533,0537,0576-0577,0579-0582,0584,
0613,0648-0654,0677,0688,0727-0728,0833,0861,0872-0882,0886,0888,0899-0900,0934,0941-0943,0945,0949-0950,0959-0960,1000,1009-1011,1019-1020]
nrm1[001-003,039,079,147-148,150-152,283,288]
c18n_low      up  3h-00:00:00    437  alloc ncn[001-0008,0038,0073-0079,0087,0100,0102,0107-0110,0112-0143,0145-0152,0182,0217-0224,0226-0228,
0238-0239,0242-0244,0258,0254-0255,0261,0284,0286-0287,0289-0296,0300-0308,0310-0324,0326,0329-0354,0356-0359,0361-0368,0401-0414,0416,0418,
0420,0422-0434,0438-0439,0441-0455,0461,0463-0464,0473-0499,0504,0531,0532,0534-0535,0539-0540,0542-0543,0545,0548,0550,0573,0655-0656,0724-
0726,0730-0800,0803-0804,0806-0809,0883-0885,0887,0889-0898,0901-0904,0920-0933,0935-0936,0938-0940,0946-0948,0951-0958,0975-0977,0999-
1001,1004,1012-1016,1021-1032],nrm[004-005,009-030,040,073,149,185-202,204,206-207]
c18n_low      up  3h-00:00:00    607  idle ncn[010-0021,0023-0035,0037,0040-0069,0071-0072,0081-0084,0086,0089-0092,0094-0095,0098-0099,0101,
0104-0106,0153-0154,0183-0216,0257-0260,0262-0283,0285,0298-0299,0369-0399,0465-0470,0500-0502,0513-0539,0536,0538,0541,0544,0546-0547,0549,
0551-0572,0574-0575,0585-0612,0616,0617-0620,0624-0647,0657-0676,0678-0687,0689-0723,0729-0792,0801-0832,0834-0808,0878-0871,0905-0927,0937,0944,
0976],nrm[006-000,041-072,074-078,080-142,144-146,153-184,205]
c18g          up  3h-00:00:00    1  drain ncg27
c18g          up  3h-00:00:00    24  mix  ncg[07-06,11-20,30-35,41,46,48],nrm[02-03,05]
c18g          up  3h-00:00:00    29  alloc ncn[01-06,09-10,21,26,28-29,36-40,42-45,47],nrm[01,04,06]
c18g_low      up  3h-00:00:00    1  drain ncg27
c18g_low      up  3h-00:00:00    24  mix  ncg[07-06,11-20,30-35,41,46,48],nrm[02-03,05]
c18g_low      up  3h-00:00:00    29  alloc ncn[01-06,09-10,21,26,28-29,36-40,42-45,47],nrm[01,04,06]
c16n          up  3h-00:00:00    1  drain lms34
c16n          up  3h-00:00:00    1  down lms100
c16n          up  3h-00:00:00    17  drain lnm[001,177,221-232,314,460,479]
c16n          up  3h-00:00:00    352  alloc lnm[002-097,099,101-103,105-140,142-176,178-189,191-220,233-234,236-238,242-286,289-313,315-320,449,
468-478,480,577-578,582-593,595-600]
c16n          up  3h-00:00:00    237  idle lnm[008,104,141,190,235,239-241,287-288,320-448,481-576,579-581],lnm[01-08]
c16n_low      up  3h-00:00:00    1  drain lms34
c16n_low      up  3h-00:00:00    1  down lms100
c16n_low      up  3h-00:00:00    17  drain lnm[001,177,221-232,314,460,479]
c16n_low      up  3h-00:00:00    252  alloc lnm[002-097,099,101-103,105-140,142-176,178-189,191-220,233-234,236-238,242-286,289-313,315-320,449,
468-478,480,577-578,582-593,595-600]
c16n_low      up  3h-00:00:00    237  idle lnm[008,104,141,190,235,239-241,287-288,320-448,481-576,579-581],lnm[01-08]
c16s          up  3h-00:00:00    1  drain lms100
c16s          up  3h-00:00:00    7  idle lms[02-08]
c16s          up  3h-00:00:00    1  drain lms100
c16s          up  3h-00:00:00    7  idle lms[02-08]
c16g          up  3h-00:00:00    1  drain lmg04
c16g          up  3h-00:00:00    3  mix  lmg[01-02,07]
c16g          up  3h-00:00:00    5  alloc  lmg[03,05-06,08-09]
c16g_low      up  3h-00:00:00    1  drain lmg04
c16g_low      up  3h-00:00:00    3  mix  lmg[01-02,07]
c16g_low      up  3h-00:00:00    5  alloc  lmg[03,05-06,08-09]
d0p2          up  3h-00:00:00    2  mix  m28-[01-02]
d0s2_low      up  3h-00:00:00    2  mix  m28-[01-02]
lh            up  3h-00:00:00    5  drain linuxhdc[188,167],lnh[091,099,100]
lh            up  3h-00:00:00    116  mix  ncn[100-1001],lnm[hdc[001-004,013,015,025,043,045-040,050,055,070-071,080-009,091-093,104-107,109,
112-136,139,151,154,156-161,160],lnm[hdc[001,005-005,001-020,030-032],lnh[002-007,013-022,028,031,040-049,133-141,147,173],lnhm[005-006,008-010,020-024,032,034,037,041]
lh            up  3h-00:00:00    25  alloc  linuxhdc[006,010-019,022,024,044,067,069,332-335,337-339,340-341],linuxhdc[002,006],lnhm[001,000-012,023-027,029-030,032,037,040-047,050-067,069-070,081-090,092,102,104-105,142-146],lnhm[030-031]
lh            up  3h-00:00:00    193  idle  linuxhdc[007-012,014,016-017,020-021,023,026-042,055-066,072-086,094-103,113-131,142-150,155],lnm
x[hdc[007-009,011-029,032,033-036],lnh[020-029,058,009,093-096,100-101,103,107,132,140-160],lnhg[01-08],lnhm[004-009,010-020,025,033,03,03,03,038-040,042-045]
(login18-1)
```

versus

The screenshot shows a Jupyter Notebook interface with a terminal window at the top displaying the output of the `sinfo` command. Below the terminal, the notebook contains several code cells. Cell [5] shows `cluster.verify_MPI("matrix.c")` with the output "no MPI problems found". Cell [6] shows `p = cluster.compile("matrix.c")` with the output "compiled". Cell [7] shows `nodes = [1,2,4], cores = [2,4,8,16], result = cluster.run_job(p, nodes, cores)`. At the bottom, a 3D surface plot shows Speedup on the vertical axis (ranging from 0 to 60) against Nodes and Cores on the horizontal axes (Nodes from 1 to 40, Cores from 2 to 16). The plot shows a blue surface representing the speedup achieved for different combinations of nodes and cores.

Interactive C++ with OpenMP and MPI

Interactive C++ and OpenMP / 1

- OpenMP: de-facto standard for shared memory parallelization with threads and tasks
- MPI: de-facto standard for distributed memory message-passing parallelization

```
In [1]: #include <iostream>
```

```
In [2]: std::cout << "Test" << std::endl;
```

Test

```
In [3]: #include <omp.h>
#pragma clang load("/usr/lib/libomp.so")
```

```
In [4]: int r = 0;
```

```
In [5]: r = 0;
#pragma omp parallel reduction(+:r)
{
    r += omp_get_num_threads();
}
r
```

```
Out[5]: 16
```

- Interactive C++

- Enabling OpenMP

- OpenMP example: 4 threads
 - Reduction on variable `r`

Interactive C++ and OpenMP / 2

- Interactive demonstration of and experiment with key concepts of parallel programming

```
In [19]: #pragma omp parallel
#pragma omp single
{
    std::cout << "Hello from single" << std::endl;

    int a;
    #pragma omp task depend(out: a)
    {
        std::cout << "Hello from task 1" << std::endl;
    }
    #pragma omp task depend(in: a)
    {
        std::cout << "Hello from task 2" << std::endl;
    }
    #pragma omp taskwait
    std::cout << "Hello after taskwait" << std::endl;
}
```

```
Hello from single
Hello from task 1
Hello from task 2
Hello after taskwait
```

- OpenMP example: tasking
 - Two tasks
 - Ordered via depend clauses
 - Synchronized via taskwait

Interactive MPI

```
In [1]: #include <stdio.h>
#include <mpi.h>
```

```
In [2]: void exchange_info() {
        int rank, size;
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        int other = 1 - rank;
        int data = rank;
        MPI_Send(&data, 1, MPI_INT, other, 0, MPI_COMM_WORLD);
        MPI_Recv(&data, 1, MPI_INT, other, 0, MPI_COMM_WORLD, M
        PI_STATUS_IGNORE);

        printf("rank %d received %d from rank %d\n", rank, dat
        a, other);
    }
```

```
In [3]: %%executable mpi.x -- -lmpi
MPI_Init(NULL, NULL);

exchange_info();

MPI_Finalize();
```

Writing executable to mpi.x

```
In [4]: !mpiexec -np 2 ./mpi.x
```

```
rank 0 received 1 from rank 1
rank 1 received 0 from rank 0
```

- Enabling MPI

- Building an MPI program

- Executing an MPI program



Interactive Correctness Checking

- MUST / ThreadSanitizer: Correctness Checking of MPI / multi-threaded parallel programs
- Did you notice the problem in the code on the previous slide? It could potentially deadlock!

In [5]: `!mustrun -np 2 ./mpi.x`

```
[MUST] MUST configuration ... centralized checks with fall-  
back application crash handling (very slow)  
[MUST] Information: overwriting old intermediate data in d  
irectory "/rwthfs/rz/cluster/home/jh366276/IkapP/must_tem  
p"!  
[MUST] Generating P^nMPI configuration ... success  
[MUST] Search for linked P^nMPI ... not found ... using LD_  
PRELOAD to load P^nMPI ... success  
[MUST] Executing application:  
rank 0 received 1 from rank 1  
rank 1 received 0 from rank 0  
=====MUST=====  
ERROR: MUST detected a deadlock, detailed information is av  
ailable in the MUST output file. You should either investig  
ate details with a debugger or abort, the operation of MUST  
will stop from now.  
=====  
[MUST] Execution finished, inspect "/rwthfs/rz/cluster/home  
/jh366276/IkapP/MUST_Output.html"!
```

After execution, see results in [MUST_Output.html \(./MUST_Output.html\)](#)

- Executing an MPI program with MUST

- Explanation of possible error

- Detailed report w/ graphical explanation



Implementation

Cling & xeus-cling: C++ in Jupyter

- Cling: interactive C++ interpreter
 - Developed by CERN & Fermilab for the ROOT data analysis framework
 - Uses Clang to parse Abstract Syntax Tree (AST)
 - Invokes LLVM's just-in-time (JIT) compilation & execution
- xeus-cling: C++ kernel for Jupyter
 - Sends code from notebook cells to Cling
 - Cling parses code and hands to JIT
 - Result is transferred back to user



More information: <https://root.cern/cling/> (logo: CC BY 4.0)
<https://xeus-cling.readthedocs.io/>

Enabling OpenMP

- Clang supports OpenMP via `-fopenmp`
 - Add to arguments in `kernel.json`
 - Install LLVM OpenMP runtime next to Cling / `xeus-cling`
- Load OpenMP runtime in the notebook:
 - `#pragma cling load("libomp.so")`
 - `#include <omp.h>`

```
In [4]: int r = 0;
```

```
In [5]: r = 0;
#pragma omp parallel reduction(+:r)
{
    r += omp_get_num_threads();
}
r
```

```
Out[5]: 16
```

More information: <https://openmp.llvm.org/>

Support for [f]printf in xeus-cling / 1

- Muscle memory for many HPC developers:
 - Use [f]printf instead of C++ streams (std::cout, std::cerr)
 - But output not redirected with xeus-cling ([jupyter-xeus/xeus-cling#112](https://github.com/jupyter-xeus/xeus-cling/issues/112))

```
In [1]: printf("Hello FOSDEM!");
```

- Redirection worked fine with std::cout and std::cerr
 - Reason: Can replace buffer of C++ streams

Support for [f]printf in xeus-cling / 2

- Proposed solution: [jupyter-xeus/xeus-cling#315](https://github.com/jupyter-xeus/xeus-cling/pull/315)
 - Inject custom implementation of [f]printf
 - Pass formatted string to std::cout, std::cerr, redirected to user
 - Merged 76 minutes after putting up the PR (issue had been open for more than 2 years)

```
In [1]: printf("Hello FOSDEM!");  
Hello FOSDEM!
```

Support for MPI Programs – inside JIT?

- MPI for distributed parallelism
 - Requires starting multiple processes
 - Naive approach: `fork()`
- Difficult for multiple reasons:
 - Communication with MPI library
 - Process discovery
 - Multiple `MPI_Init` + `MPI_Finalize`
 - Keep processes alive across cells?
 - Integration with Cling & Jupyter
 - Interpret new code in all processes?
 - Limited to one machine, no “distributed”

Support for MPI Programs – via Executables!

- Our approach: teach xeus-cling to dump executables
 - Define current cell as main()
 - Query current AST from Cling
 - Use clang::CodeGenerator to generate LLVM IR
 - Use LLVM backend to produce an object file
 - Link object file to an executable
- Implementation: around 200 lines of code interfacing the APIs

```
In [2]: %%executable mpi.x -- -lmpi
MPI_Init(NULL, NULL);
int rank;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
printf("Hello from rank %d\n", rank);
MPI_Finalize();
```

Writing executable to mpi.x

Support for MPI Programs – Launching Executables

- Launch executable with mpiexec: business as usual!

```
In [2]: %%executable mpi.x -- -lmpi
        MPI_Init(NULL, NULL);
        int rank;
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);
        printf("Hello from rank %d\n", rank);
        MPI_Finalize();
```

Writing executable to mpi.x

```
In [3]: !mpiexec -np 4 ./mpi.x
```

```
Hello from rank 1
Hello from rank 2
Hello from rank 3
Hello from rank 0
```

Interactive Correctness Analysis: ThreadSanitizer

- Instrumentation would be hard in JIT
 - Sanitizer output at process termination
 - Cannot instrument already generated code
- Feasible extension with support for dumping executables
 - Parse flags from the user
 - Enable sanitizer & debug information if requested

```
In [2]: %%executable tsan.x -- -fsanitize=thread
int a = 0;
auto t = std::thread([&]() { a++; });
a++;
t.join();
```

```
Enabling instrumentation for ThreadSanitizer
Writing executable to tsan.x
```

```
In [3]: !./tsan.x
```

```
=====
WARNING: ThreadSanitizer: data race (pid=623)
Write of size 4 at 0x7ffdc60d157c by main thread:
```


Interactive Correctness Analysis: ThreadSanitizer for OpenMP

- Extension to OpenMP:
 - “Archer” library part of the LLVM OpenMP runtime
 - Developed in collaboration between RWTH Aachen University and DoE labs
 - Makes OpenMP semantics available to ThreadSanitizer
- Avoid false-positives

```
In [2]: %%executable tsan-omp.x -- -fopenmp -fsanitize=thread
int r = 0;
#pragma omp parallel num_threads(4) reduction(+:r)
{
    r += omp_get_num_threads();
}
printf("r = %d\n", r);
```

```
Enabling instrumentation for ThreadSanitizer
Writing executable to tsan-omp.x
```

```
In [3]: !TSAN_OPTIONS='ignore_noninstrumented_modules=1' ./tsan-omp.x
r = 16
```

Interactive Correctness Analysis: MUST for MPI Programs

- MUST: runtime correctness analysis for MPI
 - Developed by TU Dresden and RWTH Aachen University
 - Checks for common classes of errors
- Usage: replace mpiexec/mpirun by mustrun




More information: <https://itc.rwth-aachen.de/must/>

```
In [4]: !mustrun -np 4 ./mpi.x

[MUST] MUST configuration ... centralized checks with fall-back application crash handling (very slow)
[MUST] Information: overwriting old intermediate data in directory "/home/jovyan/must_temp"!
[MUST] Weaver ... success
[MUST] Code generation ... success
[MUST] Build file generation ... success
[MUST] Configuring intermediate build ... success
[MUST] Building intermediate sources ... success
[MUST] Installing intermediate modules ... success
[MUST] Generating P^nMPI configuration ... success
[MUST] Search for linked P^nMPI ... not found ... using LD_PRELOAD to load P^nMPI ... success
[MUST] Executing application:
Hello from rank 1
Hello from rank 2
Hello from rank 3
Hello from rank 0
[MUST] Execution finished, inspect "/home/jovyan/MUST_Output.html"!
```

Summary

Agenda

- jupyter-based environment for interactive C/C++ OpenMP and MPI parallel programming
 - Developed with funding from IkapP project   
- In use for teaching
 - Kubernetes-based infrastructure at jupyter.rwth-aachen.de (only for members of RWTH)
 - Live illustrations in lectures on
 - High-Performance Computing
 - Parallel and Data-centric Programming
 - Corresponding exercises can be done classically or in the new environment
 - Environment was designed without a pandemic in mind, but proved to be an excellent tool
- All components are open source, all patches got accepted