Empowering Domain Experts through Automatic Code Transformation of HPC Kernels

Jannek Squar

2023-05-25

Universität Hamburg
jannek.squar@uni-hamburg.de

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

HPC Environment

- Today's HPC architecture:
  - Distributed many-core systems
  - Accelerators
  - High-speed interconnection
  - Separated IO servers for parallel file system
- Popular tools
  - Parallelisation on shared memory: OpenMP
  - Parallelisation on shared memory: MPI
  - Self-describing scientific data format: netCDF

"The questions don't change, the answers do" – *Daniel Reed*

Solving the Problem

Domain experts' situation:

- More complex computer science skills required [AC16]
- HPC skills underrepresented [MHS+20]
- Support by application software engineers sometimes possible
  - about 80 000 €/yr

Solution approach provided by CATO [SJB+20]

- Partitioning relevant memory
- Optimise netCDF usage: chunking, parallel IO, compression
- `https://github.com/JSquar/cato`

LLVM Infrastructure

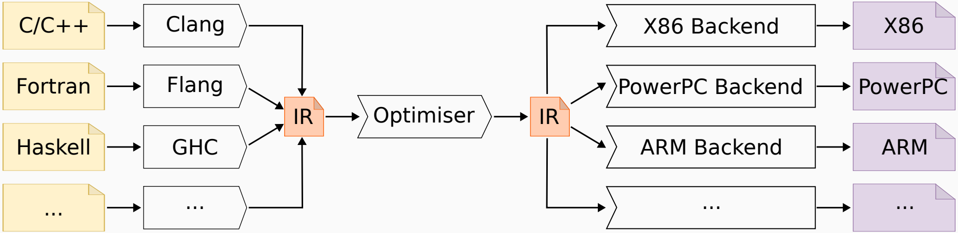


Figure 1: Modular LLVM infrastructure (based on [Lat]).

CATO Pass

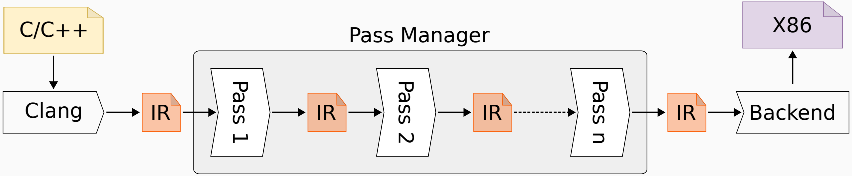


Figure 2: Insertion of CATO Pass (based on [Sam15]).

- `EP_ModuleOptimizerEarly`
- Load pre-compiled C++ replacement code as LLVM module
- Find relevant code instructions (OpenMP kernels)
- Perform modification

Memory Distribution

Memory Access Pattern

- 7+6 Dwarves [Col04, ABC+06]
- Nature is (usually) local $\rightarrow$ Stencil

$$
\begin{aligned}
f(x,y) =& C + a_0 \cdot f(x,y) \\
& + a_1 \cdot f(x-1,y) + a_2 \cdot f(x+1,y) \\
& + a_3 \cdot f(x,y-1) + a_4 \cdot f(x,y+1)
\end{aligned}
$$

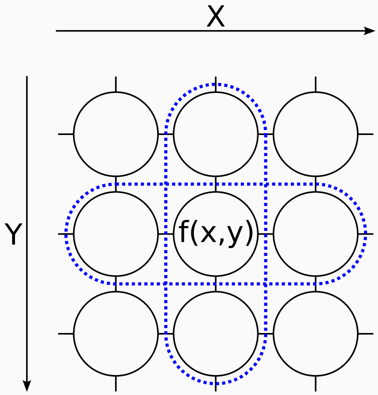


Figure 3: 5-point stencil

Stencil Distribution

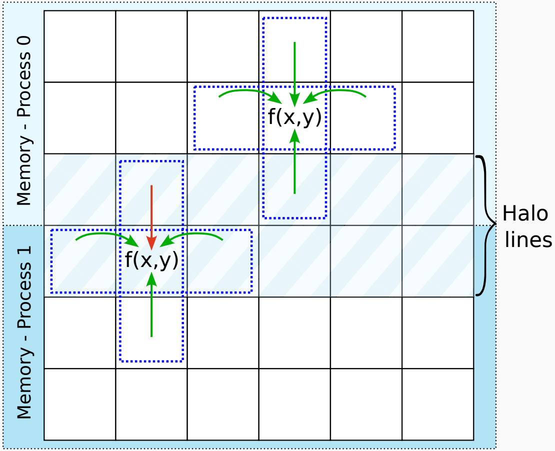


Figure 4: Green arrows: local memory. Red arrows: neighbouring process

Improved Memory Consumption - Runtime Data

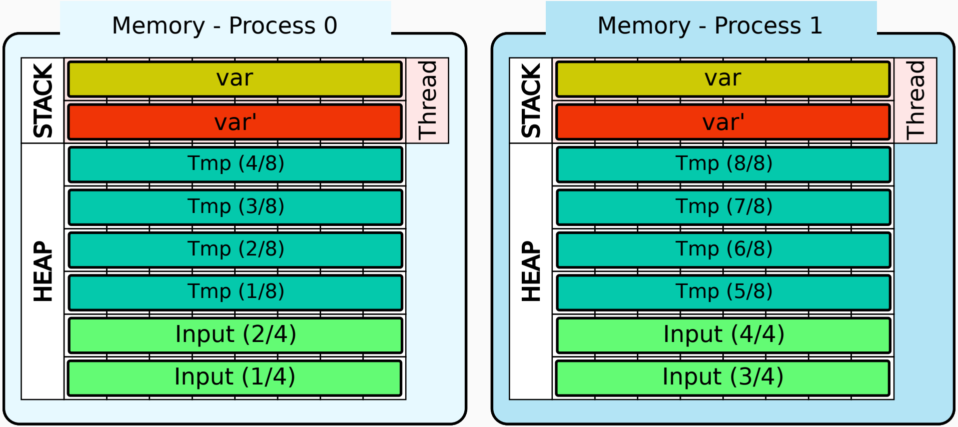


Figure 5: Distribution of initial data, focus on **runtime** data.

Improved Memory Consumption - Input Data

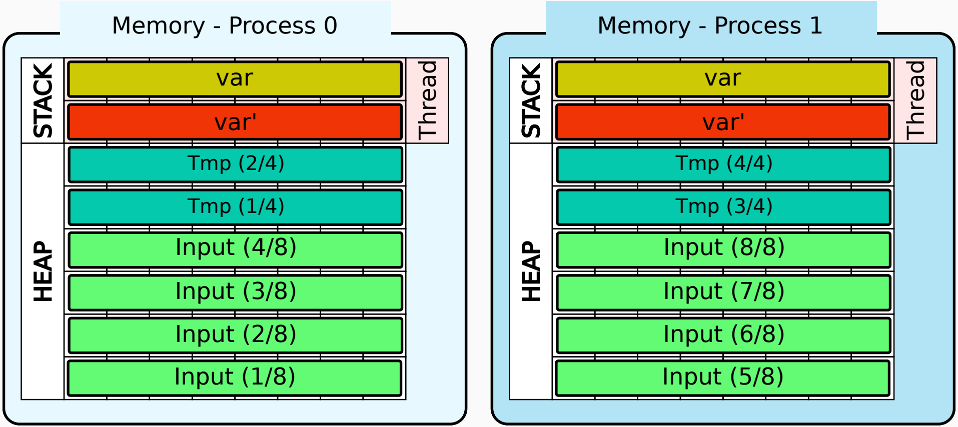


Figure 6: Distribution of initial data, focus on **input** data.

1. Add initialisation code
2. Identify and modify `malloc` calls
3. Replace OpenMP microtask
   - `@__kmpc_fork_call`
4. Modify memory accesses:
   - Inside and outside of microtask
   - Private/shared stack variables
   - Heap variables
5. Add finalisation code

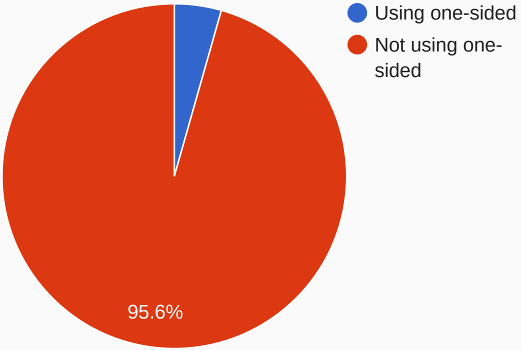


Figure 7: Usage one-sided MPI (C/C++)

IR Representation


```
%3 = tail call noalias dereferenceable_or_null(16) i8* @malloc(i64 noundef 16) #5
%4 = bitcast i32** %1 to i8**
store i8* %3, i8** %4, align 8, !tbaa !4
call void (%struct.ident_t*, i32, void (i32*, i32*, ...)*, ...) @__kmpc_fork_call(%struct.ident_t* nonnull @1, i32 1, void (i32*, i32*, ...)* bitcast (void (i32*, i32*, i32**)* @.omp_outlined. to void (i32*, i32*, ...)*), i32** nonnull %1)
```

Pass Workflow

```
RuntimeHandler r(M);
r.replace_omp_functions();
auto microtasks = find_microtasks(M);
replace_fork_calls(M, r, microtasks);
replace_memory_allocations(M, r);
replace_sequential_shared_memory_accesses(M, r);
replace_microtask_shared_memory_accesses(M, r, microtasks);
replace_parallel_for(M, r, microtasks);
replace_reductions(M, r, microtasks);
replace_criticals(M, r, microtasks);
replace_memory_deallocations(M, r);
r.adjust_netcdf_regions();
```

netCDF

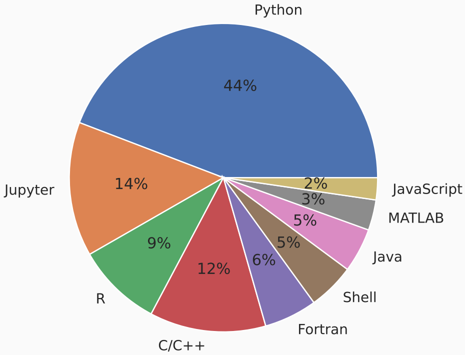


Figure 8: netCDF language distribution

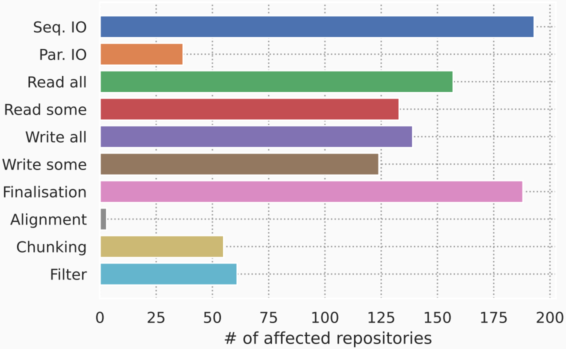


Figure 9: Used netCDF functions (C/C++)

Access via Spatial Dimensions

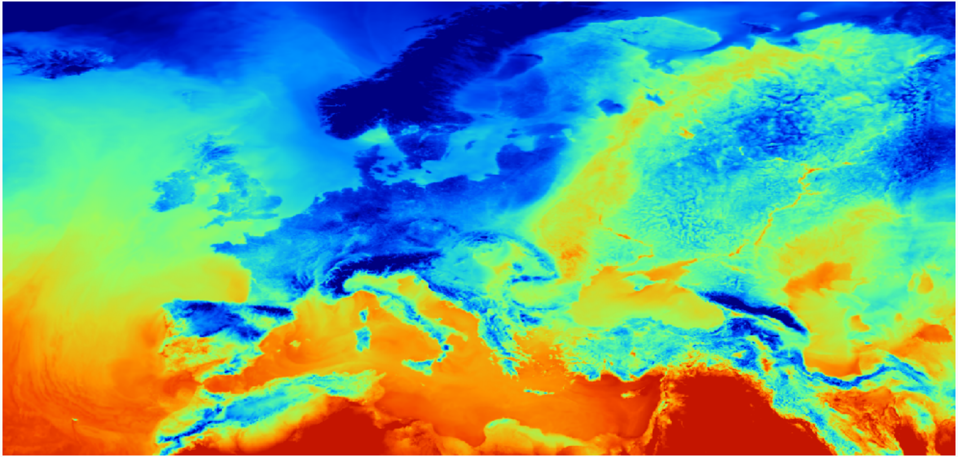


Figure 10: DWD ICON EU-nested 2m temperature from 2023-05-18-00

Access via Spatial Dimensions

- Contiguous memory layout
- Chunked data:
  - Time: slow unlimited netCDF dimension
  - Spatial dimensions: fast-varying
  - `2t:_ChunkSizes = 1, 1, 657, 1377`

$\Rightarrow$ Good read-ahead ✓

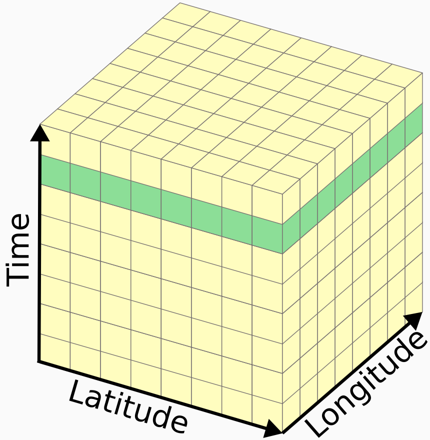


Figure 11: Convenient access pattern

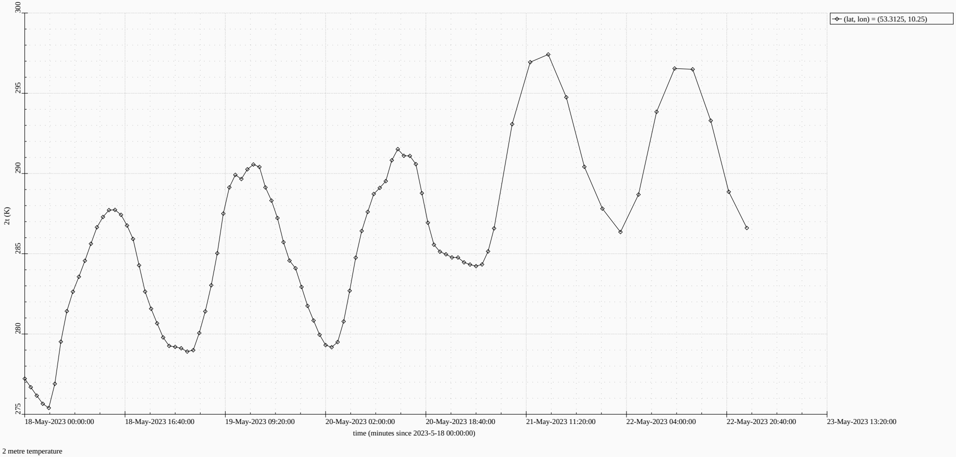


Figure 12: DWD ICON EU-nested 2m temperature time series from 2023-05-18-00 (HH)

Access via Time Dimension

- Contiguous memory layout
- Chunked data:
  - Time: slow unlimited netCDF dimension
  - Spatial dimensions: fast-varying
  - `2t:_ChunkSizes = 1, 1, 657, 1377`

⇒ ~~Good read-ahead~~

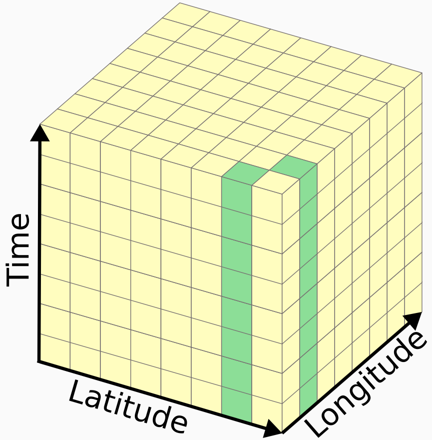


Figure 13: Inconvenient access pattern

Improved Chunking

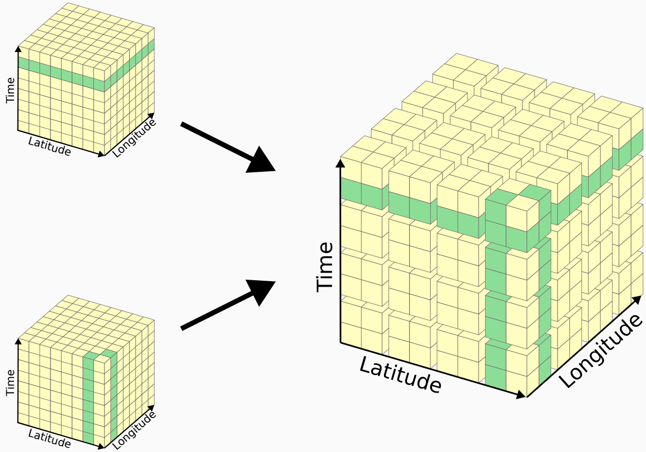


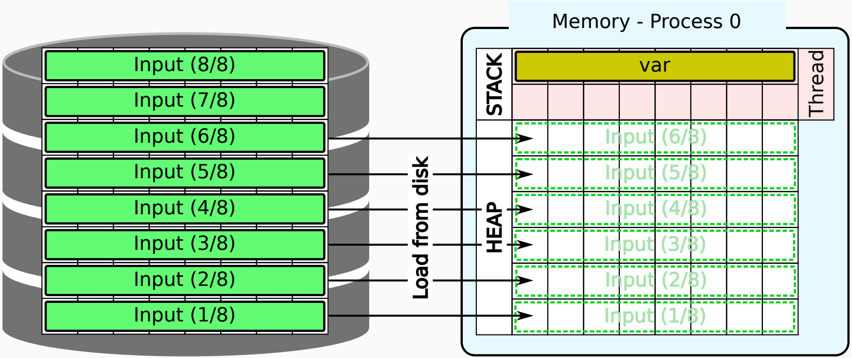
Figure 14: Combination of chunk orientations

Chunking:

- Find `nc_def_var` calls
- Add `nc_def_var_chunking`
- Add `nc_set_alignment` [BCK+15] before file initialisation

Combining parallel IO and compression makes chunking mandatory

Reading More Data



Original netCDF Workflow

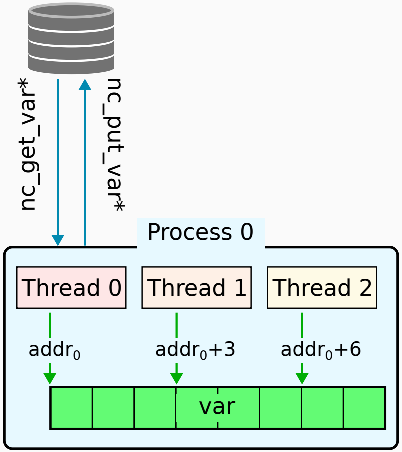


Figure 15: Access data on shared heap memory.

Modified netCDF Workflow

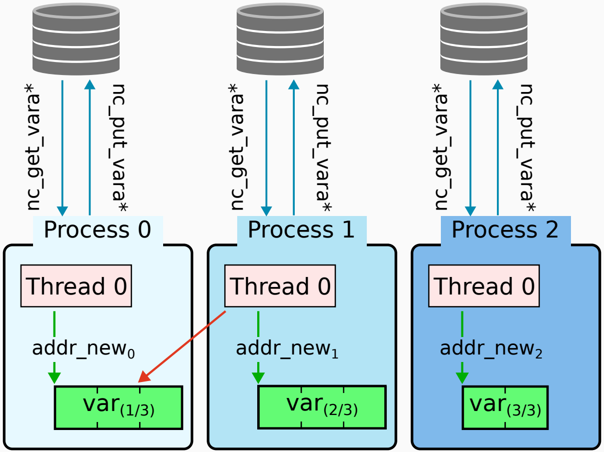


Figure 16: Access data on shared heap memory using Lustre FS.

CATO's Procedure - Parallel IO

- `nc_open` → `nc_open_par`
- `nc_create` → `nc_create_par`
- `nc_get_var*` → `nc_get_vara*`
- `nc_put_var*` → `nc_put_vara*`

Example: nc_open


```
std::optional<std::size_t> env_alignment = parse_env_size_t("CATO_NC_ALIGNMENT");
if(env_alignment.has_value()){
    alignment = env_alignment.value();
    err = nc_set_alignment(0,alignment);
}
err += nc_create_par(path, cmode, MPI_COMM_WORLD, MPI_INFO_NULL, ncidp);
check_error_code(err, "io_create_par (netCDF backend)");
```

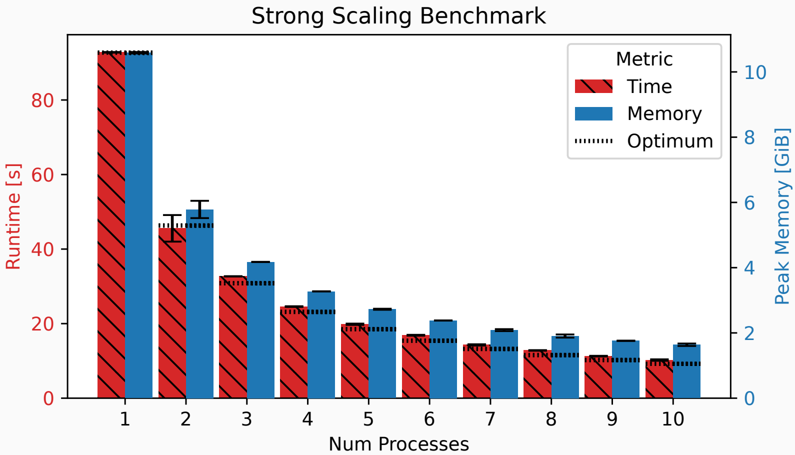
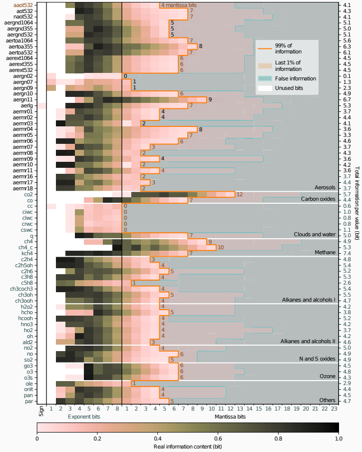


Figure 17: Benchmark using one process per node, memory peak consumption measured per node (lower is better)

Benefits of Compression

- CAMS data [IAAP+19]
- Lossless/lossy compression
- Analysis from Klöwer et al. [KRD+21, Fig. 2]



CATO’s Procedure - Filter

- Add `nc_def_var_quantize`
- Add `nc_def_var_deflate`
- Add `nc_def_var_filter`

Providing Feedback

Feedback

- Modified high-level code
  - Decompiler (RetDec [Kř23], llvm2c [sta22], LLVM-BCE [Jul22])
  - Annotated CFG (LLVis)
- Sanity Checks
  - Error checking
  - File not found
  - File not closed
- User Training
  - General introduction
  - References to official ressources
  - Tips for usage
  - References to code examples
- Optional user control via environment variables

Future Work

Memory Distribution:

- Fix bugs
- Re-introduce OpenMP
- Improve memory management
- Use MPI shared memory
- (Pattern recognition)

Parallel IO

- Feature completeness (e.g. multidimensional chunking)

Summary

- HPC is a volatile working environment
- Potential Domain scientist's struggles:
  - Confusing amount of technologies
  - Usually no trivial trying out
- Cato provides toolbox
  - Focus on natural science
  - Rapid trying out
  - 100% user space compatible
  - But: Inferior than handmade solution

Contact: `jannek.squar@uni-hamburg.de`

References

[ABC+06] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick, The Landscape of Parallel Computing Research: A View from Berkeley, Tech. report, December 2006.

[AC16] Ankit Agrawal and Alok Choudhary, Perspective: Materials informatics and big data: Realization of the “fourth paradigm” of science in materials science, APL Materials **4** (2016), no. 5, 053208.

[BCK+15] Christopher Bartz, Konstantinos Chasapis, Michael Kuhn, Petra Nerge, and Thomas Ludwig, A Best Practice Analysis of HDF5 and NetCDF-4 Using Lustre, High Performance Computing - 30th International Conference, ISC High Performance 2015, Frankfurt, Germany, July 12-16, 2015, Proceedings (Julian M. Kunkel and Thomas Ludwig, eds.), Lecture Notes in Computer Science, vol. 9137, Springer, 2015, pp. 274–281.

[Col04] Phillip Colella, Defining software requirements for scientific computing, presentation, 2004.

[IAAP+19] Antje Inness, Melanie Ades, Anna Agustí-Panareda, Jérôme Barré, Anna Benedictow, Anne-Marlene Blechschmidt, Juan Jose Dominguez, Richard Engelen, Henk Eskes, Johannes Flemming, Vincent Huijnen, Luke Jones, Zak Kipling, Sebastien Massart, Mark Parrington, Vincent-Henri Peuch, Miha Razinger, Samuel Remy, Michael Schulz, and Martin Suttie, The CAMS reanalysis of atmospheric composition, Atmospheric Chemistry and Physics **19** (2019), no. 6, 3515–3556.

[Jul22] JuliaHubOSS, Resurrected LLVM "C Backend", with improvements , online, September 2022, last accessed 2023-02-28.

[KRD+21] Milan Klöwer, Miha Razinger, Juan J. Dominguez, Peter D. Düben, and Tim N. Palmer, Compressing atmospheric data into its real information content, Nature Computational Science **1** (2021), no. 11, 713–724.

[Kř23] Jakub Křoustek, Retdec, Online, May 2023, Last accessed: 2023-05-07.

[Lat] Chris Lattner, The architecture of open source applications, Online, Last accessed: 2023-05-12.

References

[MHS+20] Glen MacLachlan, Jason Hurlburt, Marco Suarez, Kai Leung Wong, William Burke, Terrence Lewis, Andrew Gallo, Jaroslav Flidr, Raoul Gabiam, Janis Nicholas, and Brian Ensorgauging, Building a shared resource HPC center across university schools and institutes: A case study, CoRR **abs/2003.13629** (2020).

[Sam15] Adrian Sampson, LLVM for Grad Students, Online, August 2015, Last accessed: 2023-05-12.

[SJB+20] Jannek Squar, Tim Jammer, Michael Blesel, Michael Kuhn, and Thomas Ludwig, Compiler Assisted Source Transformation of OpenMP Kernels, 2020 19th International Symposium on Parallel and Distributed Computing (ISPDC), IEEE, July 2020, pp. 44–51.

[sta22] staticafi, llvm2c, Online, 2022, Last accessed: -2023-05-24.