# OpenMP in LLVM

Johannes Doerfert <jdoerfert@anl.gov>

Argonne National Lab

EXASCALE COMPUTING PROJECT

## OpenMP 5.0/5.1 Features

Feature list (right) is available at `https://clang.llvm.org/docs/OpenMPSupport.html`. Currently integrating various 5.0/5.1 features and improvements including:

- `loop`
- `tile`
- `declare variant`
- `declare mapper`
- `target nowait`
- `metadirective`
- proper math function support on GPUs

Main open problem is the interaction of static linking and GPU offloading code.

Contributors include IBM, Intel, BNL, ANL, and others.

## GPU Offloading Support

Native math functions and intrinsics, e.g., CUDA shuffle, are available in target regions.

### NVIDIA Devices

functional, several performance issues identified (see the *TRegion* section)

### AMD Devices

actively worked on, device runtime almost complete, code generation is part of the *OpenMP-IR-Builder* development

### Intel Devices

in the planning stage

Contributors include AMD, ANL, and others.

## OpenMP in Fortran

The Fortran frontend will be Flang (aka F18). Work in progress with various moving parts. In the current design, Flang lowers Fortran to MLIR dialects. From there the *OpenMP-IR-Builder* will generate LLVM-IR.

### Flang

OpenMP parsing and some semantic analysis implemented

### OpenMP MLIR dialect

OpenMP dialect started, very early stages

### Code Generation

OpenMP code generation via the *OpenMP-IR-Builder*

Contributors include NVIDIA, ARM, LANL, ORNL, ANL, and others.

## OpenMP 5.0 Implementation Details

The following table provides a quick overview over various OpenMP 5.0 features and their implementation status. Please contact *openmp-dev* at *lists.llvm.org* for more information or if you want to help with the implementation.

| Category | Feature | Status | Reviews |
|---|---|---|---|
| loop extension | support != in the canonical loop form | done | D54441 |
| loop extension | #pragma omp loop (directive) | worked on | |
| loop extension | collapse imperfectly nested loop | done | |
| loop extension | collapse non-rectangular nested loop | done | |
| loop extension | C++ range-base for loop | done | |
| loop extension | clause: if for SIMD directives | done | |
| loop extension | inclusive scan extension (matching C++17 PSTL) | unclaimed | |
| memory mangagement | memory allocators | done | r341687,r357929 |
| memory mangagement | allocate directive and allocate clause | done | r355614,r335952 |
| OMPD | OMPD interfaces | not upstream | **https://github.com/OpenMPToolsInterface/LLVM-openmp/tree/ompd-tests** |
| OMPT | OMPT interfaces | mostly done | |
| thread affinity extension | thread affinity extension | done | |
| task extension | taskloop reduction | done | |
| task extension | task affinity | not upstream | |
| task extension | clause: depend on the taskwait construct | worked on | |
| task extension | depend objects and detachable tasks | worked on | |
| task extension | mutexinoutset dependence-type for tasks | done | D53380,D57576 |
| task extension | combined taskloop constructs | done | |
| task extension | master taskloop | done | |
| task extension | parallel master taskloop | done | |
| task extension | master taskloop simd | done | |
| task extension | parallel master taskloop simd | done | |
| SIMD extension | atomic and simd constructs inside SIMD code | done | |
| SIMD extension | SIMD nontemporal | done | |
| device extension | infer target functions from initializers | worked on | |
| device extension | infer target variables from initializers | worked on | |
| device extension | OMP_TARGET_OFFLOAD environment variable | done | D50522 |
| device extension | support full ‘defaultmap’ functionality | done | D69204 |
| device extension | device specific functions | done | |
| device extension | clause: device_type | done | |
| device extension | clause: in_reduction | worked on | r308768 |
| device extension | omp_get_device_num() | worked on | D54342 |
| device extension | structure mapping of references | unclaimed | |
| device extension | nested target declare | done | D51378 |
| device extension | implicitly map ‘this’ (this[:1]) | done | D55982 |
| device extension | allow access to the reference count (omp_target_is_present) | worked on | |
| device extension | requires directive (unified shared memory) | done | |
| device extension | clause: unified_address, unified_shared_memory | done | D52625,D52359 |
| device extension | clause: reverse_offload | unclaimed parts | D52780 |
| device extension | clause: atomic_default_mem_order | unclaimed parts | D53513 |
| device extension | clause: dynamic_allocators | unclaimed parts | D53079 |
| device extension | user-defined mappers | worked on | D56326,D58638,D58523,D58074,D60972,D59474 |
| device extension | mapping lambda expression | done | D51107 |
| device extension | clause: use_device_addr for target data | worked on | |
| device extension | map(replicate) or map(local) when requires unified_shared_me | worked on | D55719,D55892 |
| device extension | teams construct on the host device | worked on | Clang part is done, r371553. |
| device extension | support non-contiguous array sections for target update | worked on | |
| atomic extension | hints for the atomic construct | worked on | D51233 |
| base language | C11 support | unclaimed | |
| base language | C++11/14/17 support | worked on | |
| base language | lambda support | done | |
| misc extension | array shaping | unclaimed | |
| misc extension | library shutdown (omp_pause_resource[_all]) | unclaimed parts | D55078 |
| misc extension | metadirectives | worked on | |
| misc extension | conditional modifier for lastprivate clause | worked on | |
| misc extension | user-defined function variants | worked on | D67294, D64095 |
| misc extensions | pointer/reference to pointer based array reductions | unclaimed | |
| misc extensions | prevent new type definitions in clauses | unclaimed | |

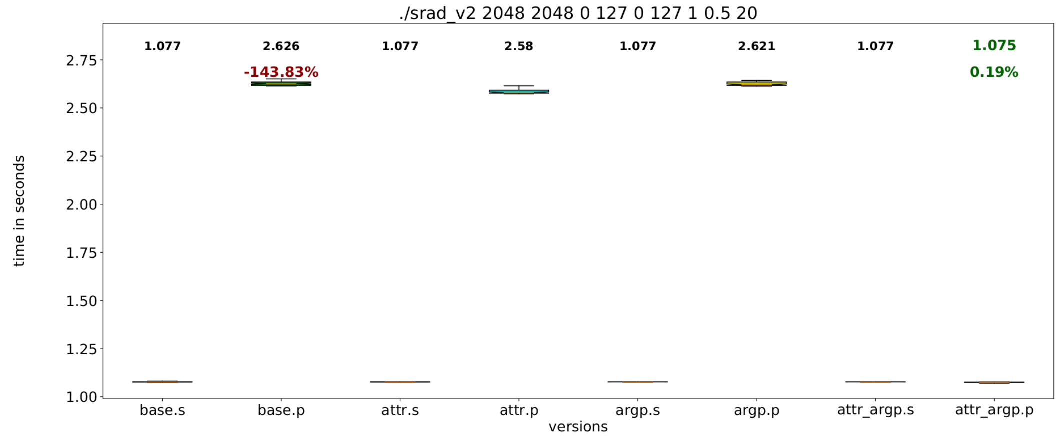
## OpenMP 5.1 Implementation Details

The following table provides a quick overview over various OpenMP 5.1 features and their implementation status, as defined in the technical report 8 (TR8). Please contact *openmp-dev* at *lists.llvm.org* for more information or if you want to help with the implementation.

| Category | Feature | Status | Reviews |
|---|---|---|---|
| misc extension | user-defined function variants with #ifdef protection | worked on | D71179 |
| loop extension | Loop tiling transformation | claimed | |

## Scalar Optimizations For Parallel Programs

Enable existing scalar optimizations, e.g., constant propagation, to deal with (OpenMP) parallel programs [1]. Mostly merged into LLVM as part of the *Attributor* framework [2].



## OpenMPOpt: Parallelism-Aware Optimizations

The *OpenMPOpt* pass augments existing “scalar” optimizations with an OpenMP (parallelism) aware, one. OpenMP runtime call deduplication, parallel region merging [1] (below), and more are under review.

```
#pragma omp parallel for
for (int j = 0; j < M; j++)
  work_0(j);
#pragma omp parallel for
for (int j = 0; j < M; j++)
  work_1(j);
```

```
#pragma omp parallel
{
  #pragma omp for
  for (int j = 0; j < M; j++)
    work_0(j);
  #pragma omp for
  for (int j = 0; j < M; j++)
    work_1(j);
}
```

## TRegions

GPU architecture agnostic interface that allows static program optimization. In the simplest case the left is normalized to the right resulting in up to 1.55× speedup [3].

```
#pragma omp target
for(int i = 0; i < N; i++)
  #pragma omp parallel for
  for (int j = 0; j < M; j++)
    work(i, j);
```

```
#pragma omp target parallel
for(int i = 0; i < N; i++)
  #pragma omp for
  for (int j = 0; j < M; j++)
    work(i, j);
```

## Acknowledgements & References

This poster shows the status of work done by various people across different Department of Energy organizations, academia, and industry.

Johannes Doerfert was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation’s exascale computing imperative.

[1] Johannes Doerfert and Hal Finkel. Compiler Optimizations For OpenMP. In *International Workshop on OpenMP*. Springer, 2018.

[2] J. Doerfert, H. Ueno, and S. Stipanovic: The Attributor: A Versatile Inter-procedural Fixpoint Iteration Framework. LLVM Developer Meeting 2019, `https://www.youtube.com/watch?v=HVvvCSSLiTw`, 2019.

[3] J. Doerfert, J. M. Diaz and H. Finkel. The TRegion Interface and Compiler Optimizations for OpenMP Target Regions. In *International Workshop on OpenMP*. Springer, 2019.