- OMPD: Provides a mechanism for enabling OpenMP specific debugging.
  - Set of APIs which the debugger can invoke to get more details regarding OpenMP specific aspects (including hierarchy) like:
    - Parallel regions, Tasks, OpenMP Threads, ICVs, etc
  - Also consists of a callback interface wherein the debugger is supposed to implement and provide a table of function pointers to OMPD
    - Symbol lookup (global, thread specific, load module specific), memory allocation/ deallocation, read/ write to memory etc
  - To be loaded as a library by the debugger /tool which provides basic debugging functionality like symbol lookup, memory reading functionalities, etc
  - Insulates the debugger (or the 3rd party tool) from the internals of the OpenMP runtime
  - Provides routines in the runtime where (internal) breakpoints can be placed by the debugger so that the user can get notified of various events like:
    - Parallel region begin, end
    - Task begin, end
    - Thead begin, end
  - Mechanism to view the backtrace skipping the runtime frames
- The LLVM OMPD implementation
  - By: RWTH-Aachen University (Joachim Protze and his students), Ignacio Laguna (LLNL), Rice University, etc. Contribution from: AMD, Perforce, etc
  - Currently on the ompd-tests branch (https://github.com/OpenMPToolsInterface/llvm-project/tree/ompd-tests)
    - Last rebase done a few months back (July)
    - A rebase from the LLVM 'main' branch in progress (in the form of a PR)
  - Currently, wrt the host support, mostly complete wrt TR9
  - Called libompd.so
  - Also includes a python based gdb plugin
    - Simple python based commands to utilize OMPD
    - Python 3.5 + required. Gdb also needs to be based off python 3.5
  - Works on the mechanism of being able to read the values of certain variables and data structures from libomp.so
  - OMP_DEBUG set to enabled
  - Debug build of either the application or libomp not required.
    - File/ Line number information might not be available.
  - OMP/OMPD implementation matched through the variable ompd_dll_locations.
- Compilers currently implementing OMPD
  - AOMP 11.7-1 and AOCC 2.3 from AMD
  - CCE 11.0 (?) from HPE
- Binary installation points:
  - Lib directory for the libompd.so
  - Plugin under lib/ompd directory
- Tests plan
  - Currently about 12 tests in the ompd-tests branch

    - Plan to expand to about 60+ tests before upstreaming to ‘main’
    - Enabled with ‘make check-ompd’ using the llvm lit framework
    - Many of the tests compare the values with the corresponding values obtained from OMPT
    - Gdb-plugin with gdb command files used for the other test scenarios
- Approximate plan for the patches
    - A very high level (compiled by Joachim) listed at: https://github.com/OpenMPToolsInterface/LLVM-openmp/issues/77
    - Upstream the runtime changes (about 400 LOC)
        - Changes fairly isolated with #ifdef OMPD_SUPPORT
        - kmp* changes are to call the runtime entry points at the appropriate locations.
        - Omp-tools.h.var (to add the API declarations)
        - Also a macro based mechanism to calculate the offsets and the sizes (runtime/src/ompd-specific.h and ompd-specific.cpp)
        - New items to be retrieved by OMPD needs to have a presence here
        - 150 extra symbols
    - Upstream the libompd changes (around 3400 LOC)
        - TargetValue.cpp and TargetValue.h (750 LOC)

          (This class encapsulates the access to target values by using OMPD callback functions. The member functions are designed to concatenate the operations that are needed to access values from structures

        - Some Misc code like ompd-state (to enumerate the thread states, etc)
        - Omp-debug.cpp (Where the definitions of most of the APIs reside) (around 1100 LOC)
        - ICV query code (around 1200 LOC)
    - Upstream the gdb-plugin changes (around 3000 LOC, but most of this is wrapper code and code for lookup of the various OMPD apis)
        - Would also include a README on the usage
    - Upstream the tests
    - Hoping to start with the first patch (the runtime one) in Jan 2021.
- What is missing wrt TR9?
    - Implementation for the retrieval for all the ICVs (21 out of 30 are implemented)
    - Some error codes being returned would need modifications wrt TR9
- Ongoing effort to refine all this underway currently : Tools Team Meetings
    - Better mechanism to provide most of the information through DWARF
    - Early stages of discussion

- Plugin commands:
    - Some ompd subcommands:
        - ompd bt -- Turn filter for "bt" output on or off. Specify "continued" option to trace worker threads back to master threads.
        - ompd icvs -- Display the values of the Internal Control Variables.
        - ompd init -- Find and initialize OMPD library
        - ompd parallel -- Display the details of the current and enclosing parallel regions.
        - ompd step -- Executes "step" and skips runtime frames as much as possible.
        - ompd threads -- Register information on threads of current context
- Sample Output:

(gdb)ompd parallel

```
Parallel Region: Nesting Level 2: Team Size: 4

=============================================

OMP Thread Nbr  Thread State                Wait Id  EntryAddr FuncName              File:Line

=================================================================================================

   0 (master) omp_state_work_parallel            0    0x401830 .omp_outlined.(void) const matrixmul.cpp    :87

   1              omp_state_work_parallel            0    0x401830 .omp_outlined.(void) const matrixmul.cpp    :87

   2              omp_state_work_parallel            0    0x401830 .omp_outlined.(void) const matrixmul.cpp    :87

   3              omp_state_work_serial              0    0x401830 .omp_outlined.(void) const matrixmul.cpp    :87

Parallel Region: Nesting Level 1: Team Size: 4

=============================================

OMP Thread Nbr  Thread State                Wait Id  EntryAddr FuncName              File:Line

=================================================================================================

   0 (master) omp_state_work_parallel            0    0x4026e0 .omp_outlined..26(void) const matrixmul.cpp    :132

   1              omp_state_overhead                 0    0x4026e0 .omp_outlined..26(void) const matrixmul.cpp    :132

   2              omp_state_overhead                 0    0x4026e0 .omp_outlined..26(void) const matrixmul.cpp    :132

   3               omp_state_overhead                0    0x4026e0 .omp_outlined..26(void) const matrixmul.cpp    :132

=================================================================================================
```

(gdb)ompd icvs

ICV Name	Scope	Value
debug-var	address_space	1
ompd-num-procs-var	address_space	64
stacksize-var	address_space	8388608
affinity-format-var	address_space	OMP: pid %P tid %i thread %n bound to OS proc set {%A}
ompd-implicit-var	task	1
ompd-final-var	task	0
default-device-var	thread	0
ompd-team-size-var	parallel	4
active-levels-var	parallel	2
tool-var	address_space	1
display-affinity-var	address_space	0
cancel-var	address_space	0
bind-var	task	0
thread-limit-var	address_space	2147483647
tool-libraries-var	address_space	
nthreads-var	thread	4
ompd-thread-num-var	thread	0
dyn-var	thread	0
levels-var	parallel	2
max-task-priority-var	address_space	0
max-active-levels-var	task	2147483647

(gdb)ompd threads

Thread 1 (1314391) is an OpenMP thread; state: omp_state_work_parallel

Thread 2 (1314400) is no OpenMP thread

Thread 3 (1314401) is an OpenMP thread; state: omp_state_overhead

Thread 4 (1314402) is an OpenMP thread; state: omp_state_overhead

Thread 5 (1314403) is an OpenMP thread; state: omp_state_overhead

Thread 6 (1314404) is an OpenMP thread; state: omp_state_work_parallel

Thread 7 (1314405) is an OpenMP thread; state: omp_state_work_parallel

Thread 8 (1314406) is an OpenMP thread; state: omp_state_work_serial

Thread 9 (1314407) is no OpenMP thread

(gdb)bt

#0 .omp_outlined._debug__.6 (.global_tid.=0x7ffff68c9e40, .bound_tid.=0x7ffff68c9e38) at nested_parallel.c:27

#1 0x000000000040138d in .omp_outlined..9 (.global_tid.=0x7ffff68c9e40, .bound_tid.=0x7ffff68c9e38) at nested_parallel.c:24

#2 0x00007ffff7f76c63 in __kmp_invoke_microtask () at
/home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/z_Linux_asm.S:1166

#3 0x00007ffff7ee1043 in __kmp_invoke_task_func(int) (gtid=3) at
/home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/kmp_runtime.cpp:7142

#4 0x00007ffff7edeada in __kmp_launch_thread(kmp_info_t*) (this_thr=0x438340) at
/home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/kmp_runtime.cpp:5816

#5 0x00007ffff7f50f63 in __kmp_launch_worker(void*) (thr=0x438340) at
/home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/z_Linux_util.cpp:589

#6 0x00007ffff7e4e609 in start_thread () at /lib/x86_64-linux-gnu/libpthread.so.0

#7 0x00007ffff7d75293 in clone () at /lib/x86_64-linux-gnu/libc.so.6

(gdb) ompd bt on continued

(gdb) bt

#0 @thread 4: .omp_outlined._debug__.6 (.global_tid.=0x7ffff68c9e40, .bound_tid.=0x7ffff68c9e38) at nested_parallel.c:27

#1 0x000000000040138d in @thread 4: .omp_outlined..9 "#pragma omp parallel" (.global_tid.=0x7ffff68c9e40,
.bound_tid.=0x7ffff68c9e38) at nested_parallel.c:24

switching to thread 1

#5 0x00007ffff7ec029d in @thread 1: __kmpc_fork_call (loc=0x402100, argc=0, microtask=0x401370 <.omp_outlined..9>)

    at /home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/kmp_csupport.cpp:307

#6 0x0000000000401318 in @thread 1: .omp_outlined._debug__.4 (.global_tid.=0x7fffffffd530, .bound_tid.=0x7fffffffd528) at nested_parallel.c:24

#7 0x00000000004013bd in @thread 1: .omp_outlined..10 "#pragma omp parallel" (.global_tid.=0x7fffffffd530, .bound_tid.=0x7fffffffd528) at nested_parallel.c:20

#10 0x00007ffff7ec029d in @thread 1: __kmpc_fork_call (loc=0x402118, argc=0, microtask=0x4013a0 <.omp_outlined..10>)

    at /home/jini/ompd_tools_if4/llvm-project/openmp/runtime/src/kmp_csupport.cpp:307

#11 0x00000000004012c1 in @thread 1: .omp_outlined._debug__.1 (.global_tid.=0x7fffffffd930, .bound_tid.=0x7fffffffd928) at nested_parallel.c:20

#12 0x00000000004013ed in @thread 1: .omp_outlined..11 "#pragma omp parallel" (.global_tid.=0x7fffffffd930, .bound_tid.=0x7fffffffd928) at nested_parallel.c:14

outermost generating task was reached