OpenMP Tool Interfaces in OpenMP 5.0

Joachim Protze (protze@itc.rwth-aachen.de)

i12 High Performance Computing
RWTH AACHEN UNIVERSITY

Motivation

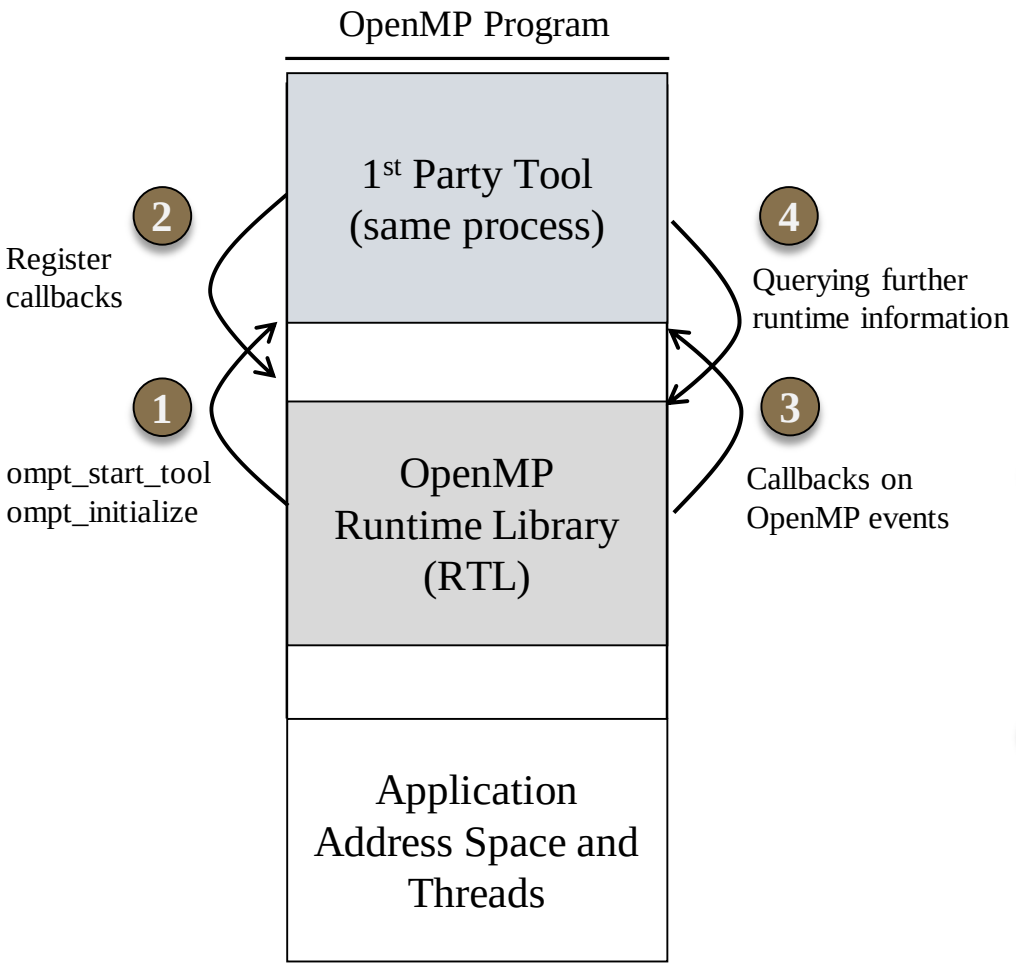
- Motivation
  - Tool support essential for program development
    - Users expect tool support, especially in today's complicated systems
    - Question of productivity
  - Users want portable tools across systems/implementations

- Currently not possible in OpenMP due to missing interfaces
  - Tools must be OpenMP implementation specific
  - Only alternative: show underlying system information only

Two OpenMP Tools Interfaces

- OpenMP Tools Interface (OMPT)
  - 1st party interface as integral part of the runtime system
  - Combination of passive state tracking and active callbacks
  - Mechanism to allow “clean” stack traces
- OpenMP Debug Interface (OMPD)
  - 3rd party interface for external debuggers
  - Separate address space, but closely coupled with runtime
  - Follows OMPT concepts and definitions
- Goal: inclusion into OpenMP 5.0 (reached with TR6)

OMPT Architecture



1 Runtime initialization

- OpenMP runtime searches for tool
- Runtime finished initialization
- Runtime allows the tool to initialize

2 Tool initialization

- Tool looks up OMPT functions with look-up
- Tool registers callbacks for OpenMP events

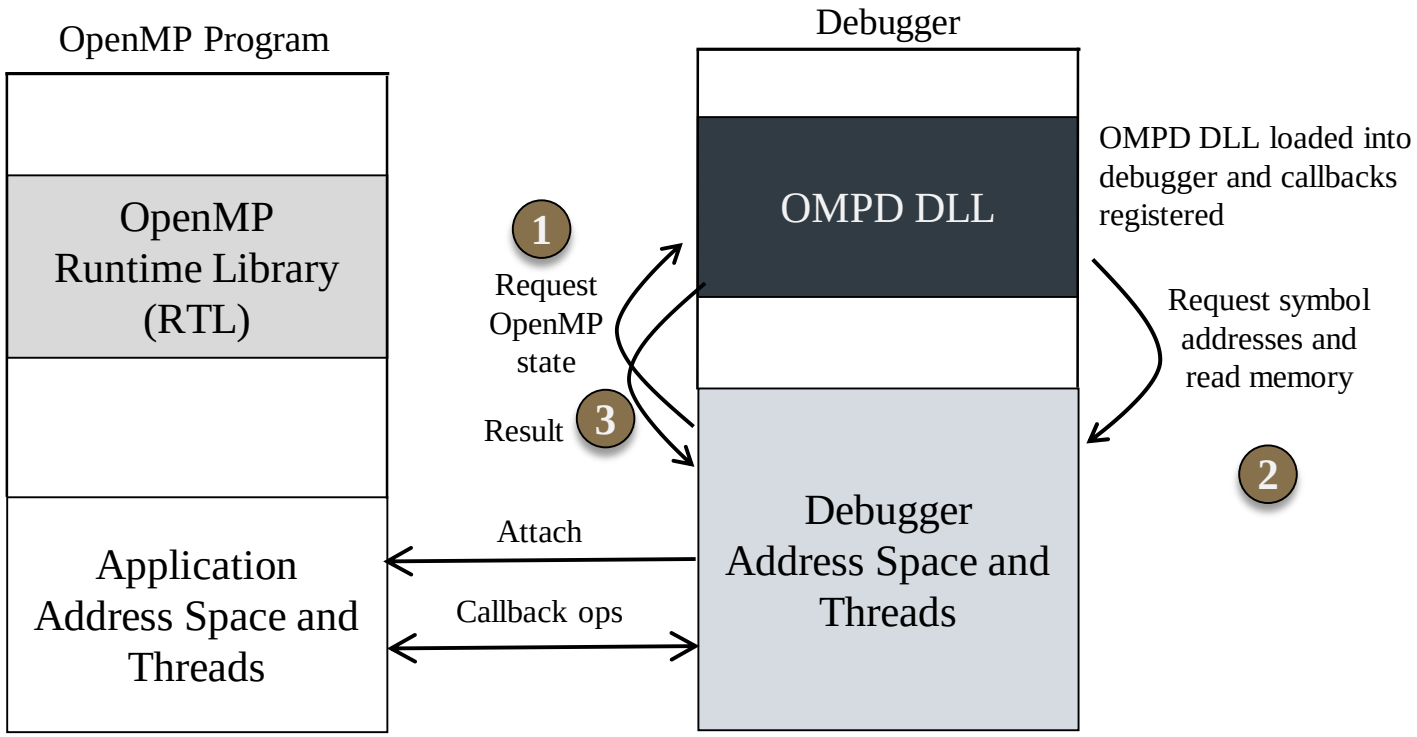
3 Callback functions

- Callbacks on OpenMP events
- E.g., entering/exiting parallel, target, worksharing, or synchronization regions

4 Querying further information

- OMPT functions to
  - investigate thread state
  - identify OpenMP runtime frames

OMPD Architecture



Request types

1

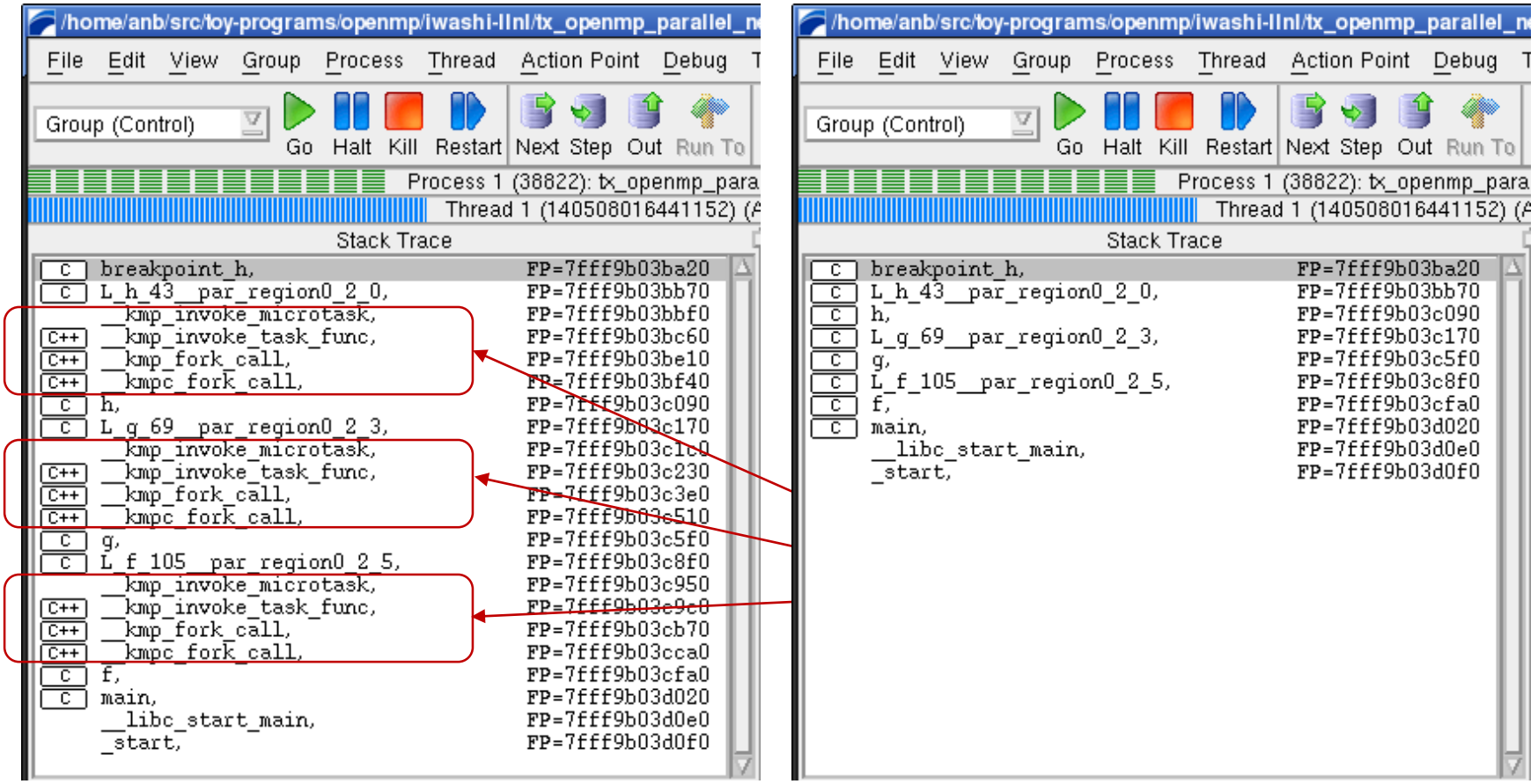
- Handles for threads, parallel regions, tasks
- Parent / child relationships
- State of handles (wait, work, idle)

Callback functions

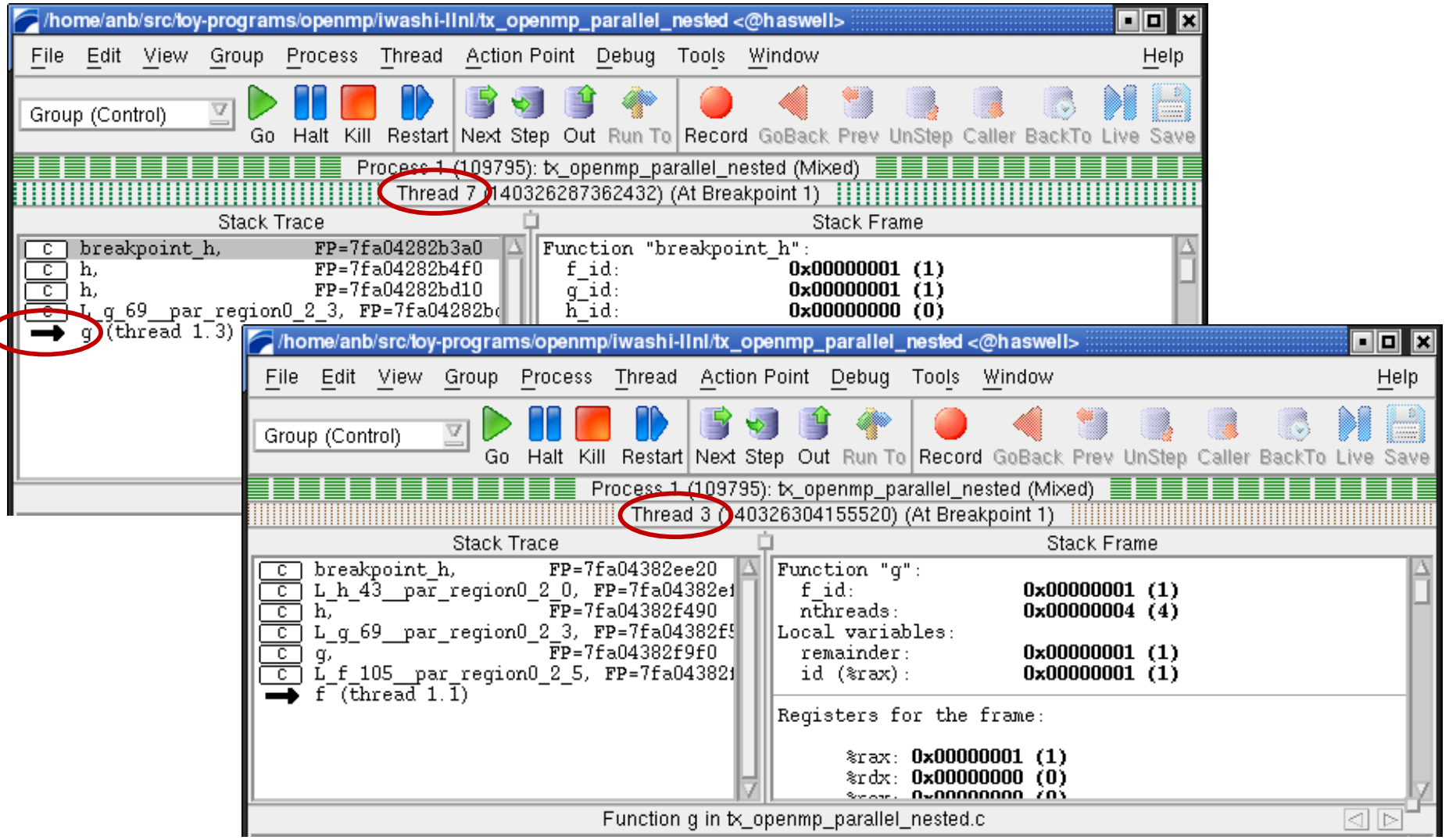
2

- Lookup symbols in the target process
- Read/write target address spaces
- Support for GPUs (coming soon)

OMPD Use Case: OpenMP Stack Filtering



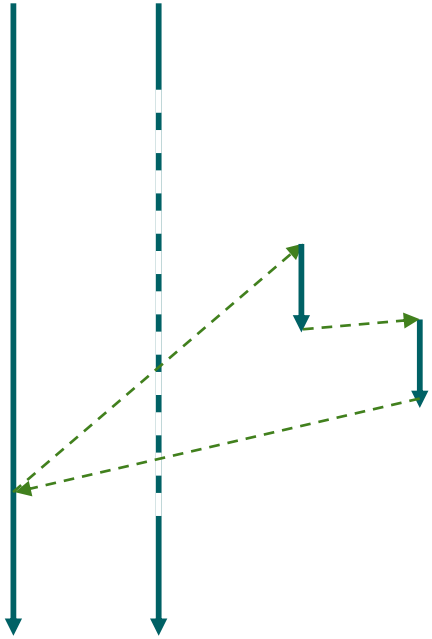
OMPD Use Cases: Linking master/worker threads



OMPD Use Case: Stepping into OpenMP Regions

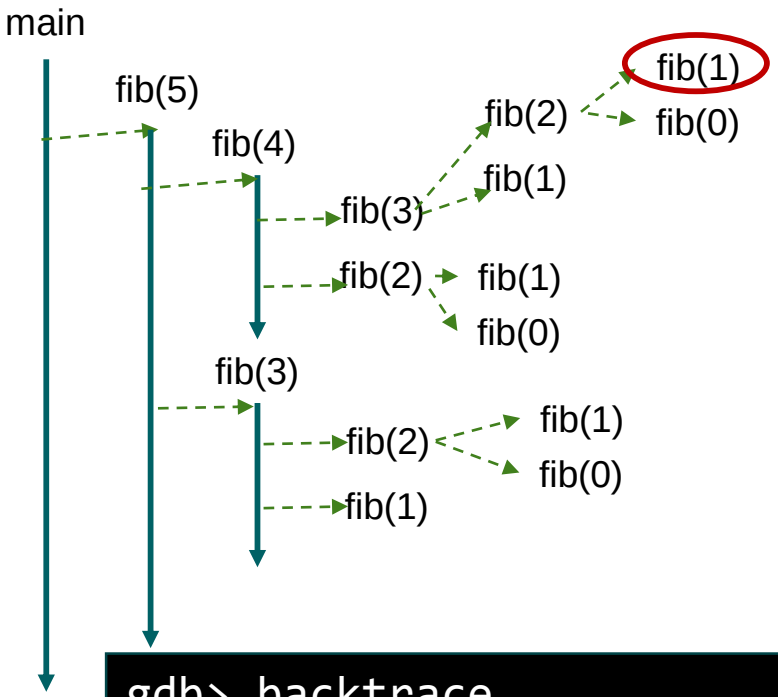
- OMPD provides information on where task execution starts:
  - This will enable to single step into task execution:
    - Program execution is paused before task scheduling point (task/taskwait/barrier)
    - Click on single-step could jump to start of next executed task

```
#pragma omp parallel
 {
  #pragma omp master
   {
    a++;
    #pragma omp task shared(a,b)
     { a++; b++; }
    #pragma omp task shared(b,c)
     { b++; c++; }
    c++;
    #pragma omp taskwait
   }
  sleep(20); // emulate load
 }
```



Stack trace views: serial code

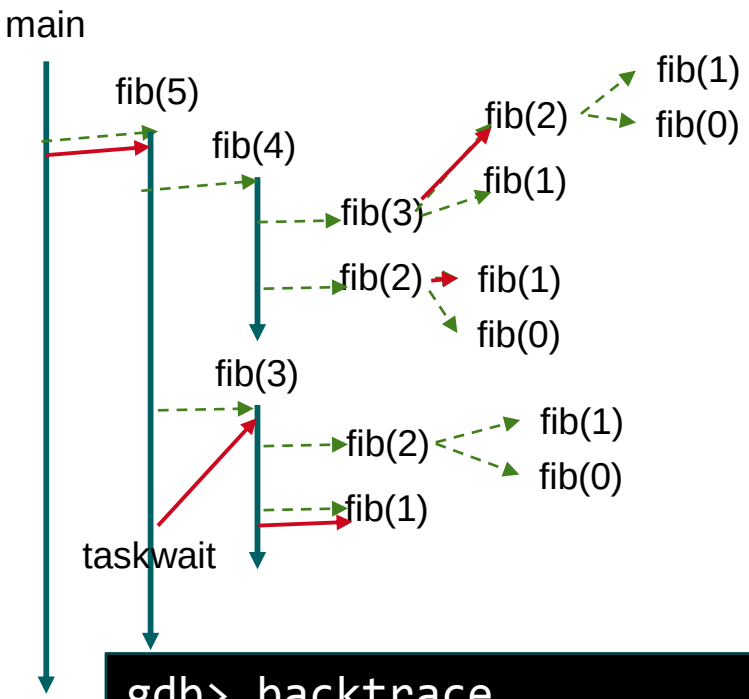
```
int fib(int i){
  if i<=1 return 1;
  int a,b;

    a = fib(i-1);

    b = fib(i-2);

  return a+b;
}
int main(){
  int result;

    result = fib(5);
  return result;
}
```




```
gdb> backtrace
#0 fib(1)
#1 fib(2)
#2 fib(3)
#3 fib(4)
#4 fib(5)
#5 main()
```

Stack trace views: OpenMP tasks

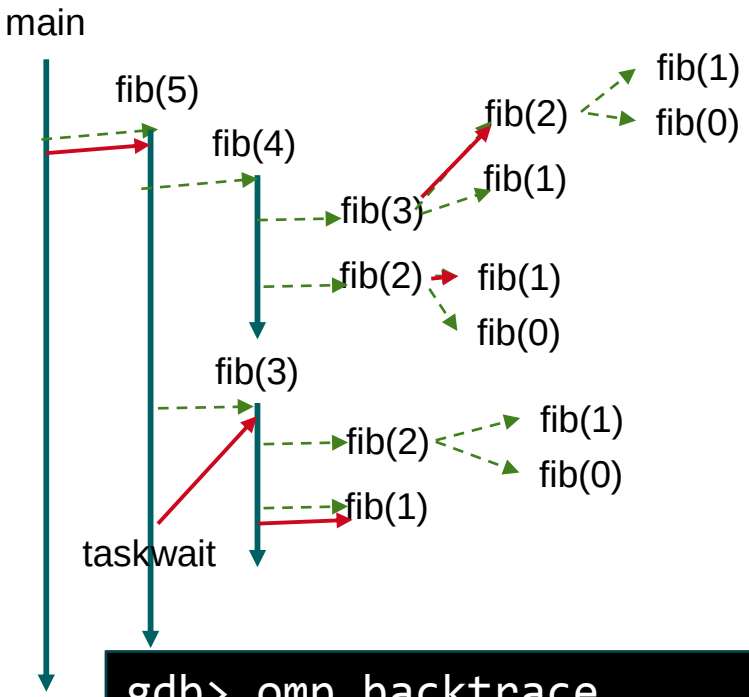
```
int fib(int i){
  if i<=1 return 1;
  int a,b;
  #pragma omp task shared(a)
    a = fib(i-1);
  #pragma omp task shared(b)
    b = fib(i-2);
  #pragma omp taskwait
  return a+b;
}
int main(){
  int result;
  #pragma omp parallel sections
    result = fib(5);
  return result;
}
```




```
gdb> backtrace
#0 fib(2)
#1 fib(5)
#2 main()
```

Stack trace views: OpenMP tasks

```
int fib(int i){
  if i<=1 return 1;
  int a,b;
  #pragma omp task shared(a)
    a = fib(i-1);
  #pragma omp task shared(b)
    b = fib(i-2);
  #pragma omp taskwait
  return a+b;
}
int main(){
  int result;
  #pragma omp parallel sections
    result = fib(5);
  return result;
}
```




```
gdb> omp_backtrace
#0 fib(1)@0
#1 fib(3)@1
#2 fib(4)@1
#3 fib(5)@0
#4 main()@0
```

Implementation Status

- Implementation of OMPT for host available in LLVM/OpenMP master
  - Patches applied just recently
  - Implementation matches OpenMP TR6 specification
- Implementation of OMPT is available in IBM OpenMP runtime
  - compatible with the „TR4“-patch in LLVM/OpenMP
- Implementation of OMPT for target is work in progress
- Implementation of OMPD for host/device is/will be available shortly:
  - https://github.com/OpenMPToolsInterface/
  - We have plans to upstream these efforts as well

Thank you for your attention.