Programming-Model Centric Debugging for OpenMP/OMPss

Kevin Pouget
Jean-François Méhaut, Miguel Santana

Université Grenoble Alpes / LIG, STMicroelectronics, France
Nano2017-DEMA project

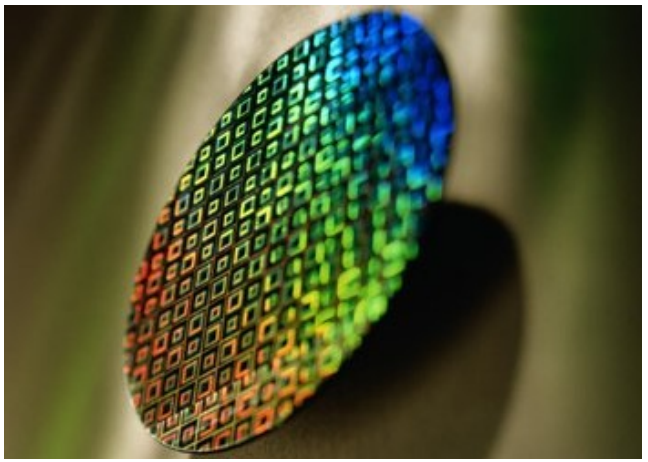
NANO2017/DEMA Workshop, Chamonix
December 14th, 2015

Introduction: Embedded Systems and MPSoC

Compiler Optimization and Runtime SystEms

Convergence of Embedded and HPC

- HPC in embedded systems
  - high-def. multimedia
  - augmented reality
  - video games on smartphones
- Embedded systems in HPC
  - dedicated hardware accelerators
  - energy efficiency
    - ★ e.g. Mont-Blanc projects

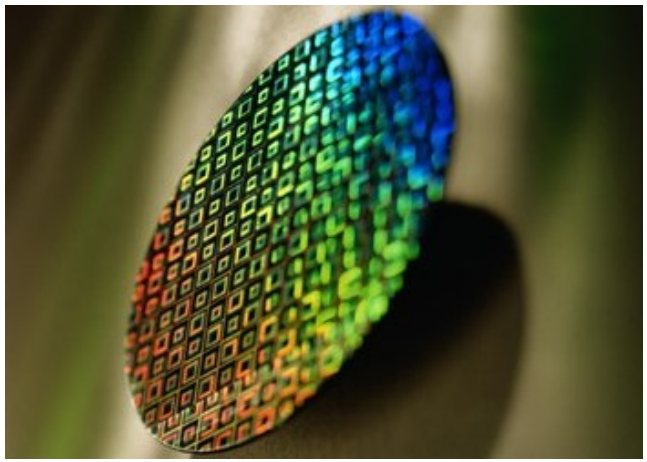


informatics mathematics

Introduction: Embedded Systems and MPSoC

⇒ important demand for:

- Powerful parallel architectures
- High-level development methodologies
- Efficient verification & validation tools

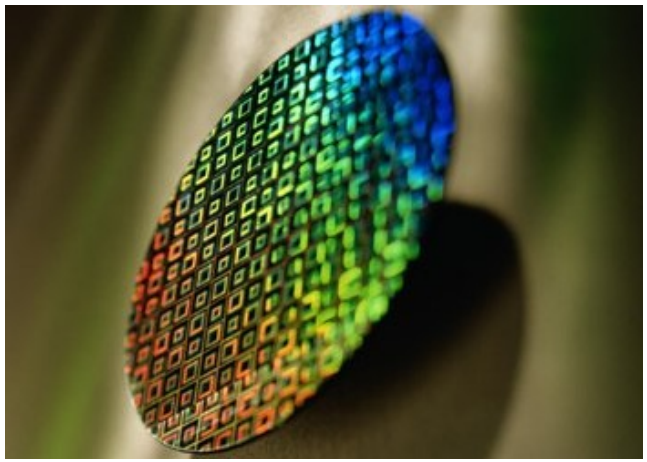


Introduction: Embedded Systems and MPSoC

Compiler Optimization and Runtime SystEms

⇒ important demand for:

- Powerful parallel architectures
  - Shared+distrib mem cores+accelerators
- High-level development methodologies
- Efficient verification & validation tools

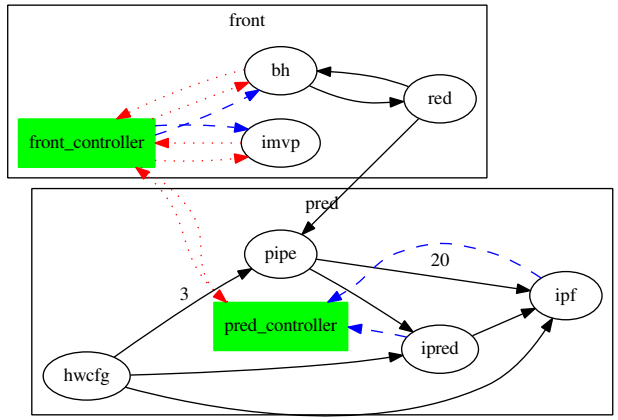


Introduction: Embedded Systems and MPSoC

Compiler Optimization and Runtime SystEms

⇒ important demand for:

- Powerful parallel architectures
  - Shared+distrib mem cores+accelerators
- High-level development methodologies
  - Programming models & environments
- Efficient verification & validation tools

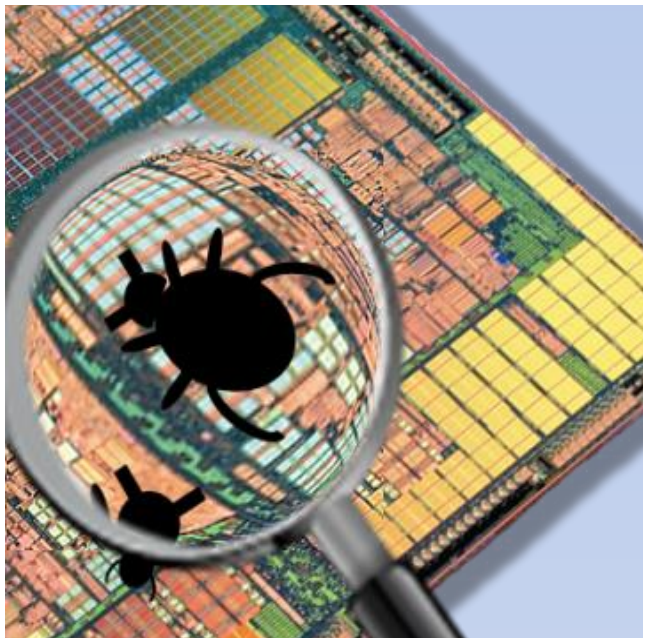


Introduction: Embedded Systems and MPSoC

Compiler Optimization and Runtime SystEms

⇒ important demand for:

- Powerful parallel architectures
  - Shared+distrib mem cores+accelerators
- High-level development methodologies
  - Programming models & environments
- Efficient verification & validation tools
  - Our research effort



informatics mathematics

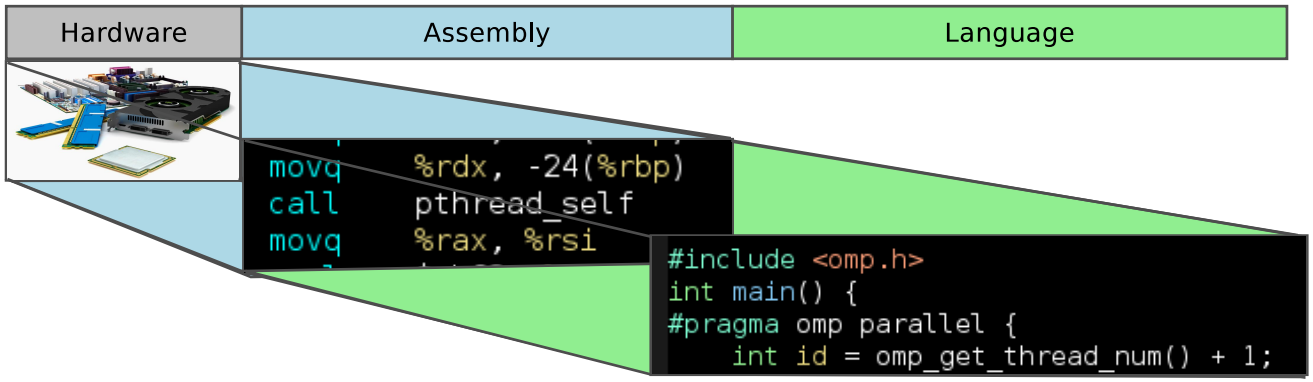
Agenda

Compiler Optimization and Runtime SystEms

Agenda

Compiler Optimization and Runtime SystEms

Verification and Validation: Debugging

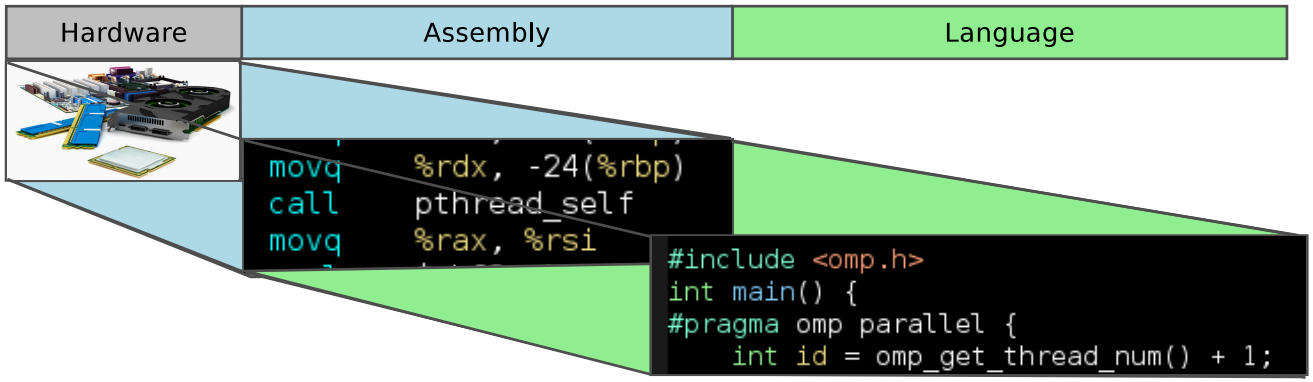


Source-Level Interactive Debugging (e.g. GDB)

- Developers mental representation VS. actual execution
- Understand the different steps of the execution

Verification and Validation: Debugging

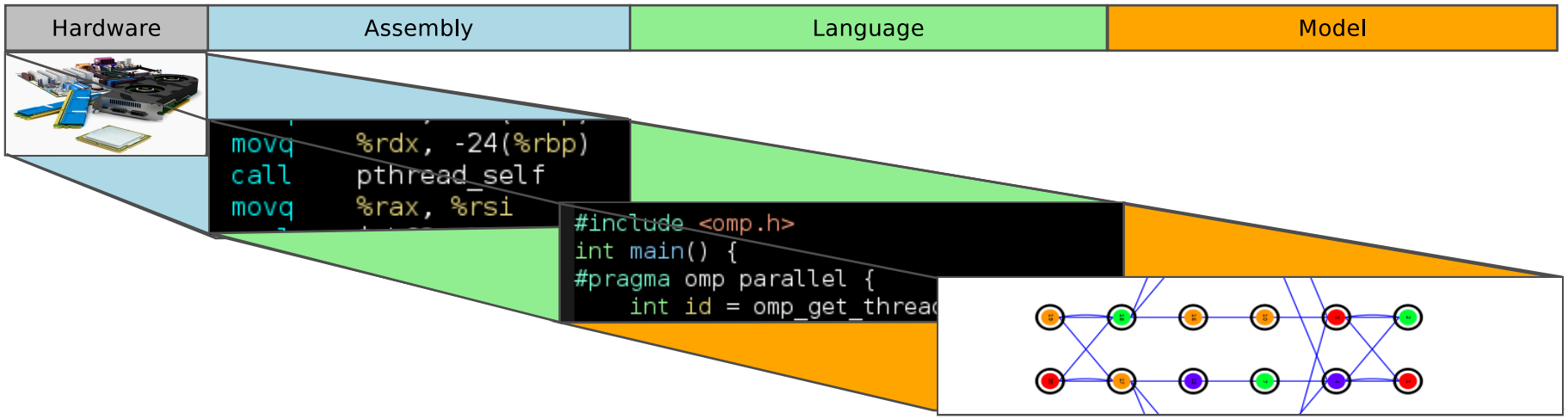
Compiler Optimization and Runtime SystEms



What about programming models?

Verification and Validation: Debugging

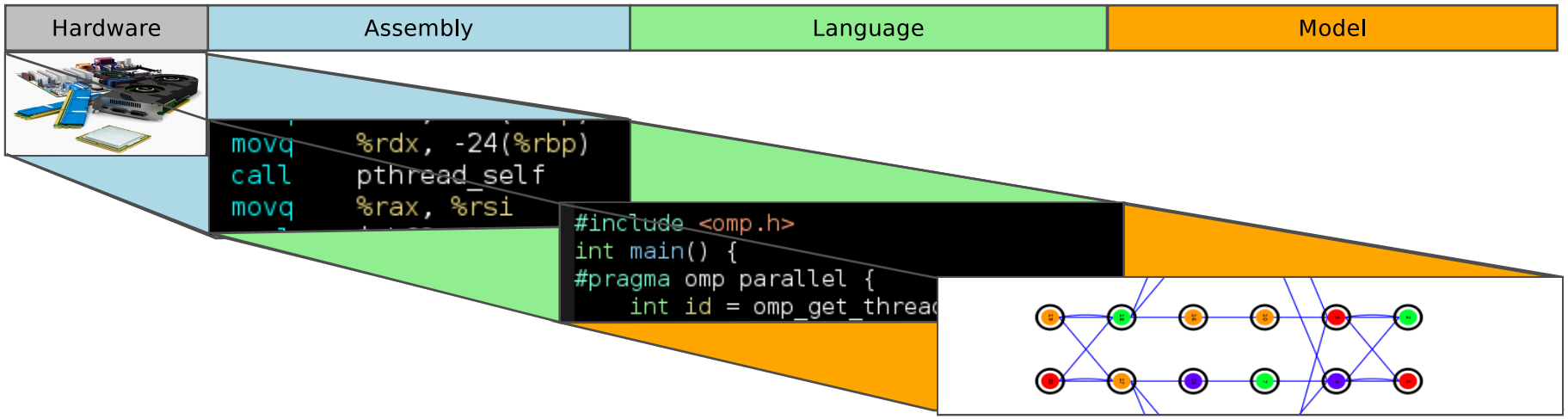
Compiler Optimization and Runtime SystEms



What about programming models?

Verification and Validation: Debugging

Compiler Optimization and Runtime SystEms

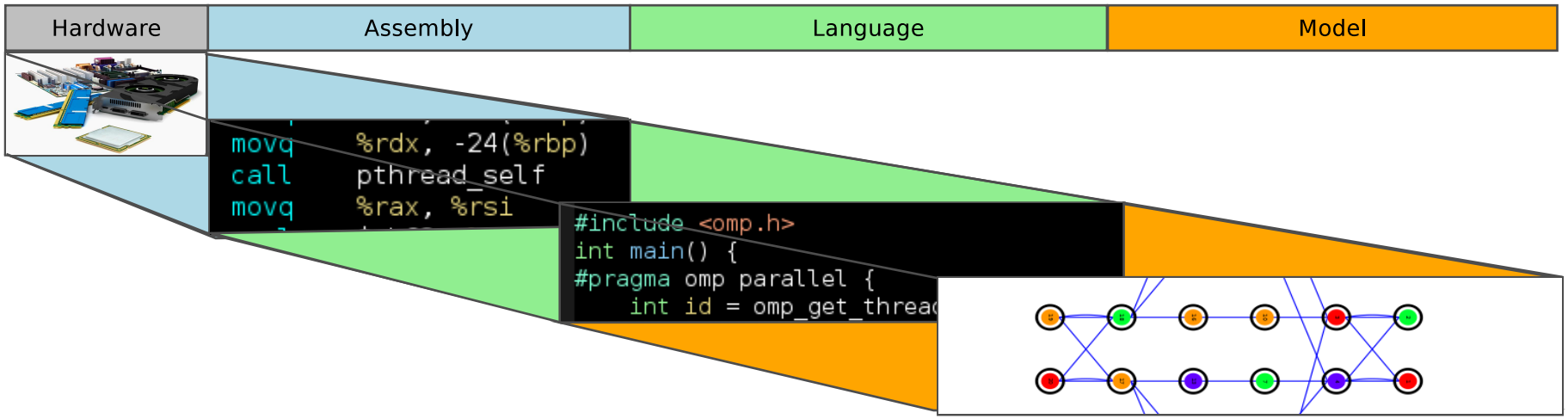


What about programming models?

Source-level Interactive Debuggers operate at language-level.

Verification and Validation: Debugging

Compiler Optimization and Runtime SystEms



What about programming models?

Source-level Interactive Debuggers operate at language-level.

They have no knowledge about
high-level abstract machines!

Research Context

Compiler Optimization and Runtime SystEms

Objective

Provide developers with means to
better understand the state of the high-level applications and control more easily their execution,
suitable for various models and environments.

Agenda

Compiler Optimization and Runtime SystEms

Idea: **Integrate programming model concepts in interactive debugging**

Programming Model Centric Debugging

Compiler Optimization and Runtime SystEms

1. Provide a Structural Representation
   - Draw application architecture diagrams
   - Represent the relationship between the entities
2. Monitor Dynamic Behaviors
   - Monitor the collaboration between the tasks
   - Detect communication, synchronization events
3. Interact with the Abstract Machine
   - Control the execution of the entities
   - Support interactions with *real* machine

Programming Model Centric Debugging

Compiler Optimization and Runtime SystEms

1. Provide a Structural Representation
   - Draw application architecture diagrams
   - Represent the relationship between the entities
2. Monitor Dynamic Behaviors
   - Monitor the collaboration between the tasks
   - Detect communication, synchronization events
3. Interact with the Abstract Machine
   - Control the execution of the entities
   - Support interactions with *real* machine

Programming Model Centric Debugging

Compiler Optimization and Runtime SystEms

1. Provide a Structural Representation
   - Draw application architecture diagrams
   - Represent the relationship between the entities
2. Monitor Dynamic Behaviors
   - Monitor the collaboration between the tasks
   - Detect communication, synchronization events
3. Interact with the Abstract Machine
   - Control the execution of the entities
   - Support interactions with *real* machine

Programming Model Centric Debugging

Compiler Optimization and Runtime SystEms

1. Provide a Structural Representation
   - Draw application architecture diagrams
   - Represent the relationship between the entities
2. Monitor Dynamic Behaviors
   - Monitor the collaboration between the tasks
   - Detect communication, synchronization events
3. Interact with the Abstract Machine
   - Control the execution of the entities
   - Support interactions with *real* machine

Let's apply to OpenMP!

Agenda

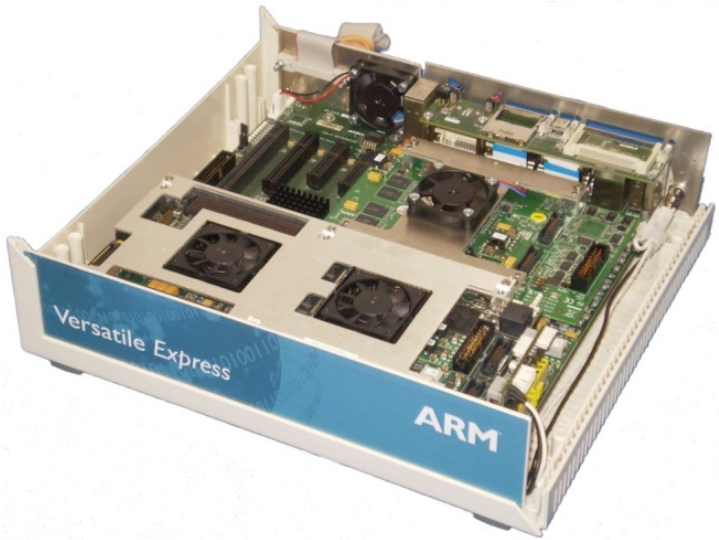
Compiler Optimization and Runtime SystEms

Nano2017/Dema project (with STMicroelectronics)

Compiler Optimization and Runtime SystEms

Debugging Embedded and Multicore Applications

ARM Juno



- heterogenous archi.
- ARM big.LITTLE + Mali GPU

OpenMP Parallel Programming

- fork/join multithreading
- tasks with dependencies
- industrial standard for HPC

mcGDB debugger

- Python extension of GDB
- support for dataflow, components, ...
- developed in partnership with STMicro

OpenMP and GDB

$\Rightarrow$ No high-level vision of the application by GDB

```
(gdb) list
17  // <==== current thread is here
18  #pragma omp critical
19  {
20     printf("@%d Inside critical zone", id);
21  }
(gdb) next
@4 Inside critical zone
@2 Inside critical zone
20  printf("@%d Inside critical zone", id);
(gdb) # I wanted to be the first :‘-(
```

OpenMP and GDB

$\Rightarrow$ No high-level vision of the application by GDB

```
(gdb) list
17  // <==== current thread is here
18  #pragma omp critical
19  {
20     printf("@%d Inside critical zone", id);
21  }
(gdb) next
@4 Inside critical zone
@2 Inside critical zone
20  printf("@%d Inside critical zone", id);
(gdb) # I wanted to be the first :‘-(
```

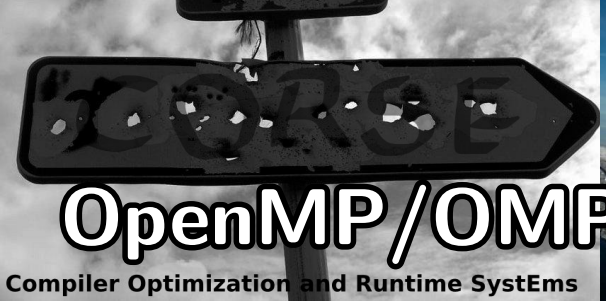
OpenMP/OMPss: Problems with GDB

Compiler Optimization and Runtime SystEms

OpenMP and GDB

$\Rightarrow$ No high-level vision of the application by GDB

```
(gdb) list
17  // <==== current thread is here
18  #pragma omp critical
19  {
20      printf("@%d Inside critical zone", id);
21  }
(gdb) next
@4 Inside critical zone
@2 Inside critical zone
20  printf("@%d Inside critical zone", id);
(gdb) # I wanted to be the first :‘-(
```



OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  ①// beginning of main function
  #pragma omp parallel {
    // beginning of parallel zone

    #pragma omp single {
      // execute single
    }//implicit barrier

    #pragma omp critical {
       // execute critical
    }
```

OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    ①②③④// beginning of parallel zone

    #pragma omp single {
      // execute single
    }//implicit barrier

    #pragma omp critical {
       // execute critical
    }
```

OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    ②③④// beginning of parallel zone

    #pragma omp single {
      ①// execute single
    }//implicit barrier

    #pragma omp critical {
       // execute critical
    }
```

OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    // beginning of parallel zone

    #pragma omp single {
      // execute single
    }①②③④//implicit barrier

    #pragma omp critical {
       // execute critical
    }
```

OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

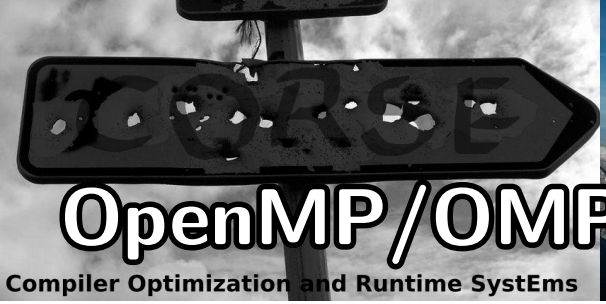
... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    // beginning of parallel zone

    #pragma omp single {
      // execute single
    }//implicit barrier

    #pragma omp critical ①③④ {
      ② // execute critical
    }
```



OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    // beginning of parallel zone

    #pragma omp single {
      // execute single
    }//implicit barrier

    #pragma omp critical ③④ {
       ①// execute critical
    }②
```

OpenMP/OMPss: mcGDB Execution Control

Compiler Optimization and Runtime SystEms

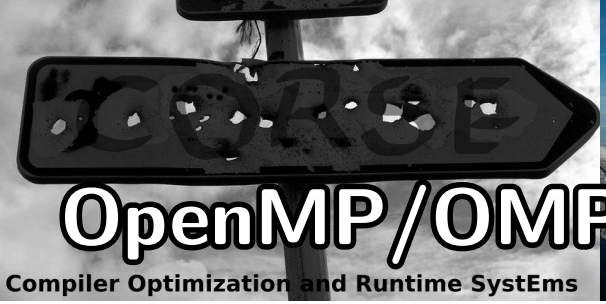
... control the execution of the entities ...

1. start
2. omp start
3. omp step
4. omp next barrier
5. omp critical next
6. omp critical next
7. omp critical next

```
int main() {
  // beginning of main function
  #pragma omp parallel {
    // beginning of parallel zone

    #pragma omp single {
      // execute single
    }//implicit barrier

    #pragma omp critical ④ {
       ③// execute critical
    }①②
```



OpenMP/OMPss: structural representation

Compiler Optimization and Runtime SystEms

... provide a structural representation
... provide details about entity state

1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

OpenMP/OMPss: structural representation

Compiler Optimization and Runtime SystEms

... provide a structural representation
... provide details about entity state

1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

OpenMP/OMPss: structural representation

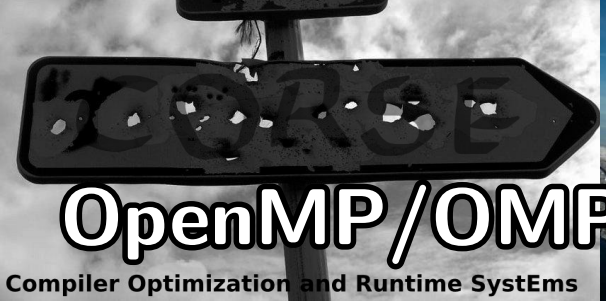
Compiler Optimization and Runtime SystEms

... provide a structural representation
... provide details about entity state

1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation


OpenMP/OMPss: mcGDB sequence diagram
Compiler Optimization and Runtime SystEms

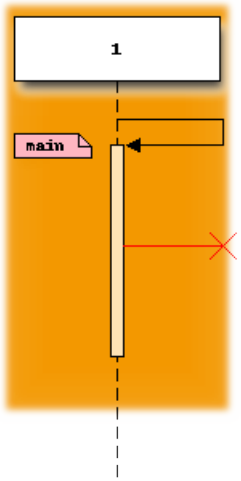
DEMO

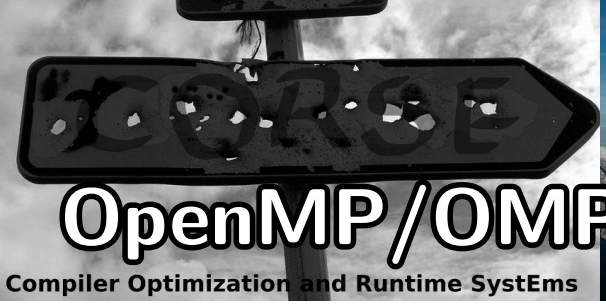


OpenMP/OMPss: mcGDB sequence diagram

Compiler Optimization and Runtime SystEms

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next

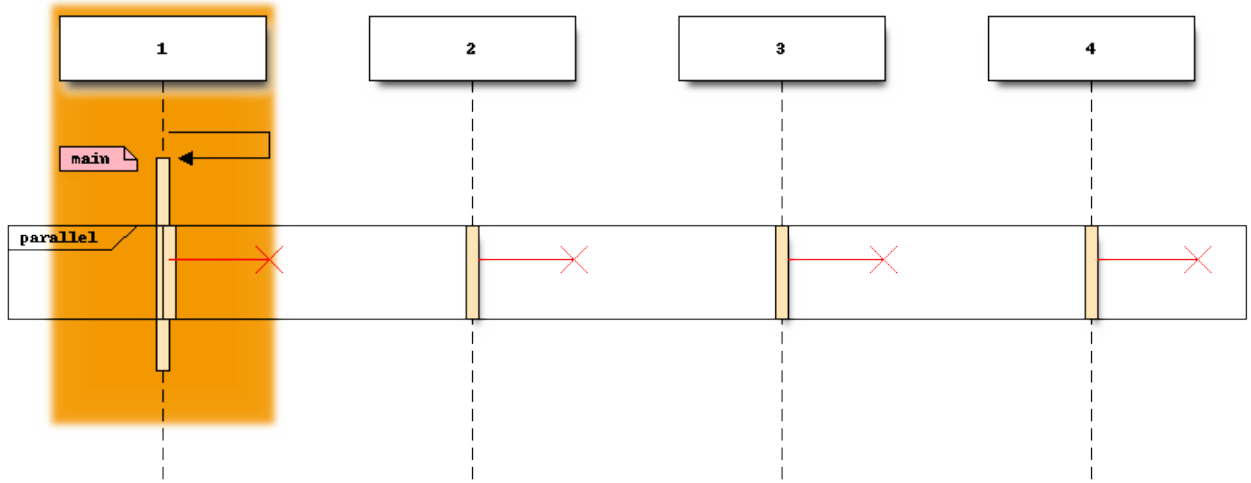




OpenMP/OMPss: mcGDB sequence diagram

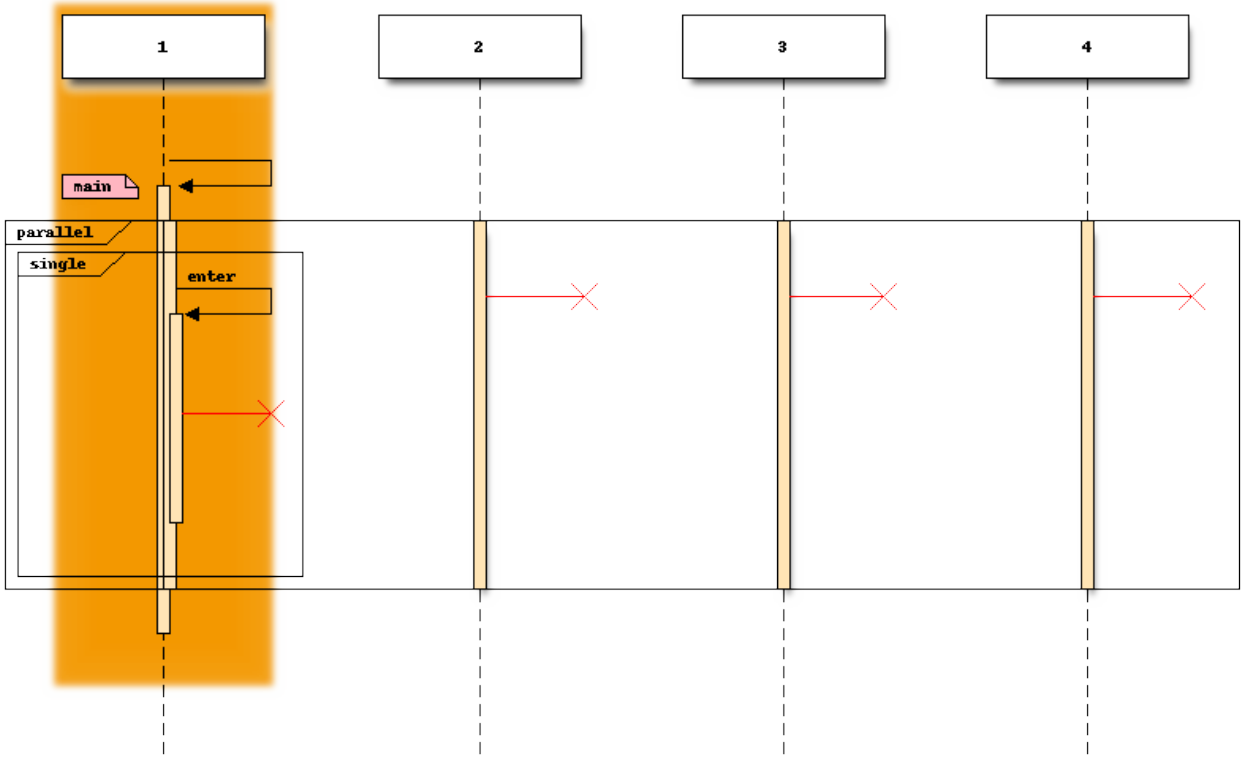
Compiler Optimization and Runtime SystEms

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



OpenMP/OMPss: mcGDB sequence diagram

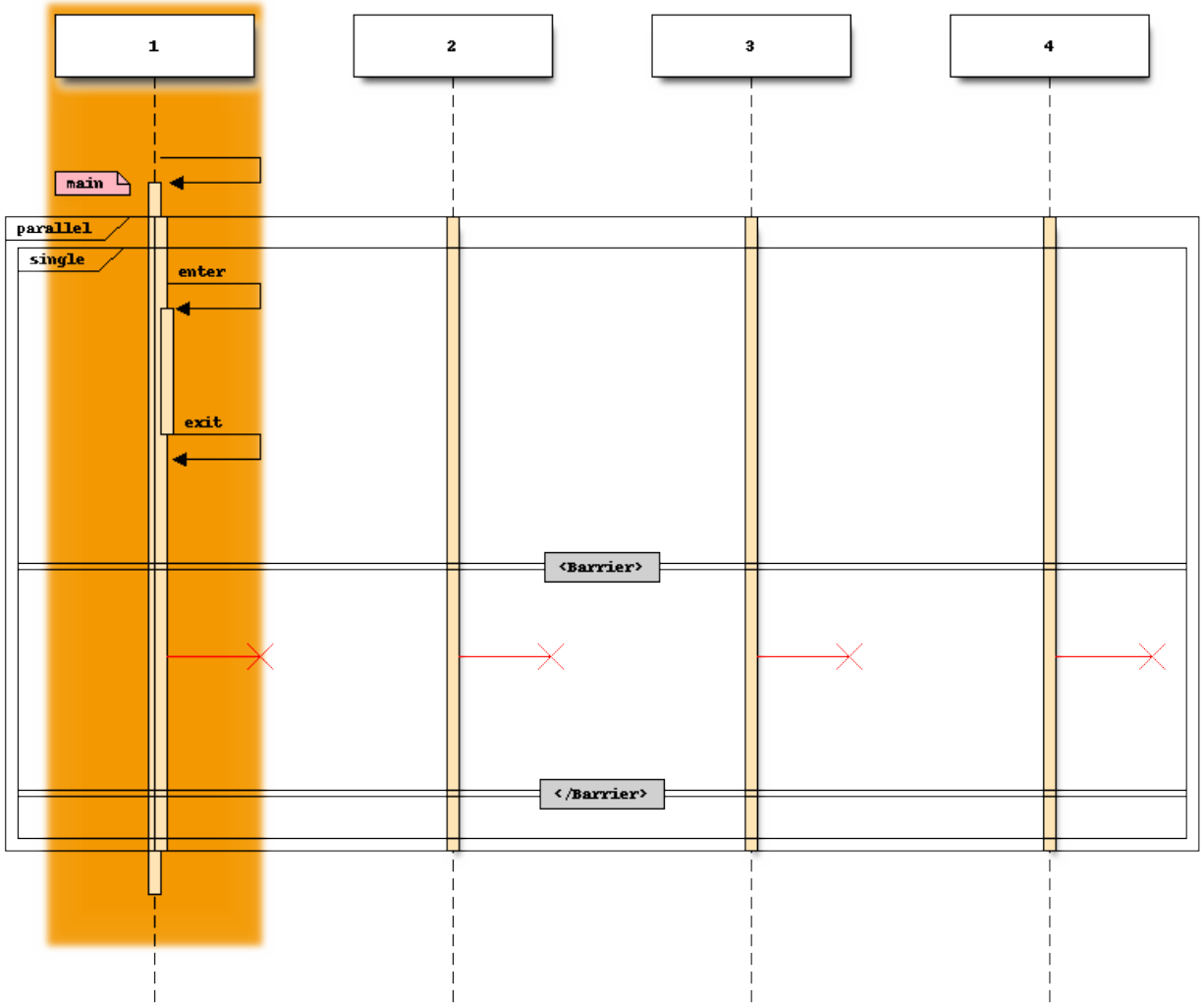
1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



OpenMP/OMPss: mcGDB sequence diagram

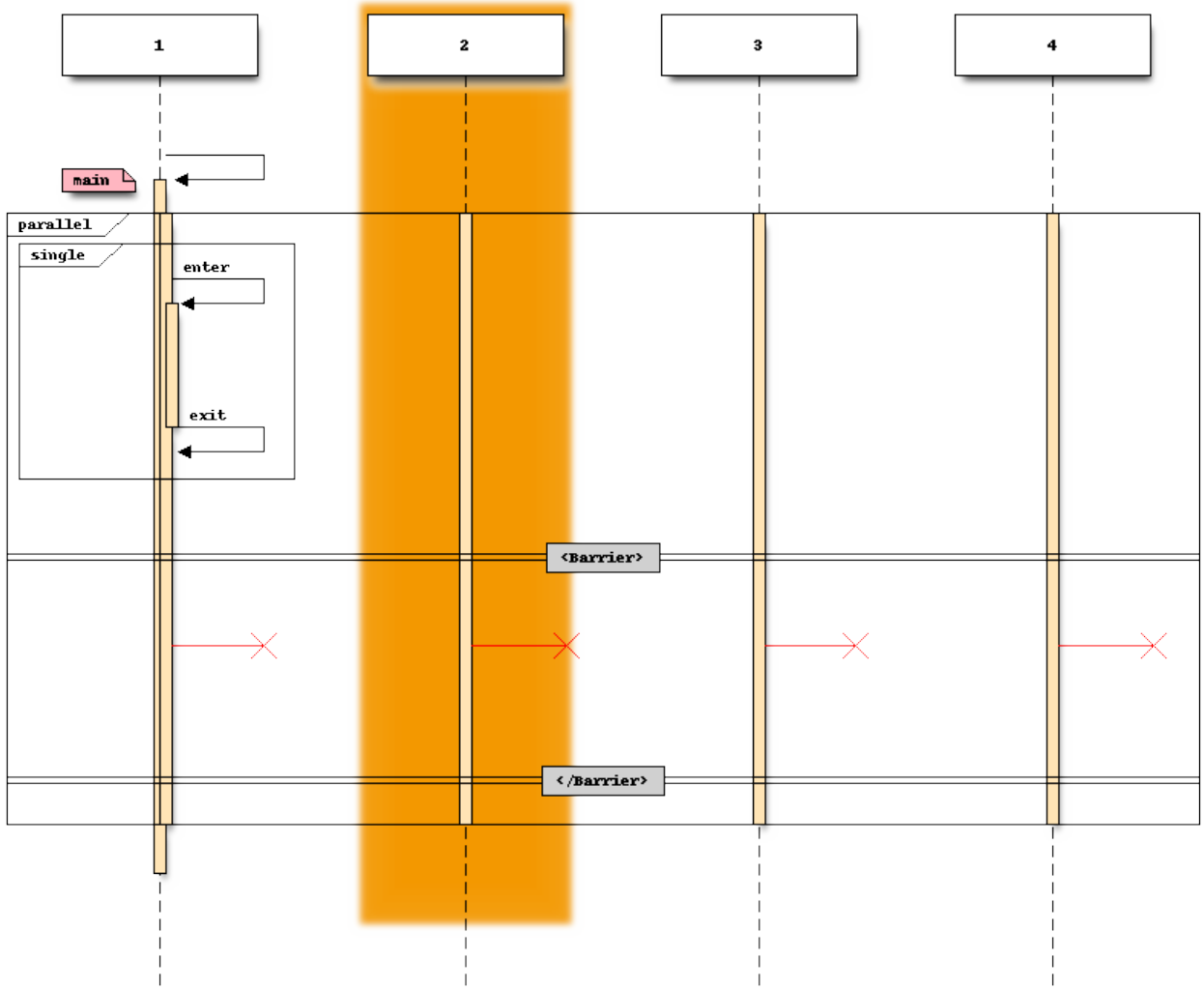
Compiler Optimization and Runtime SystEms

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



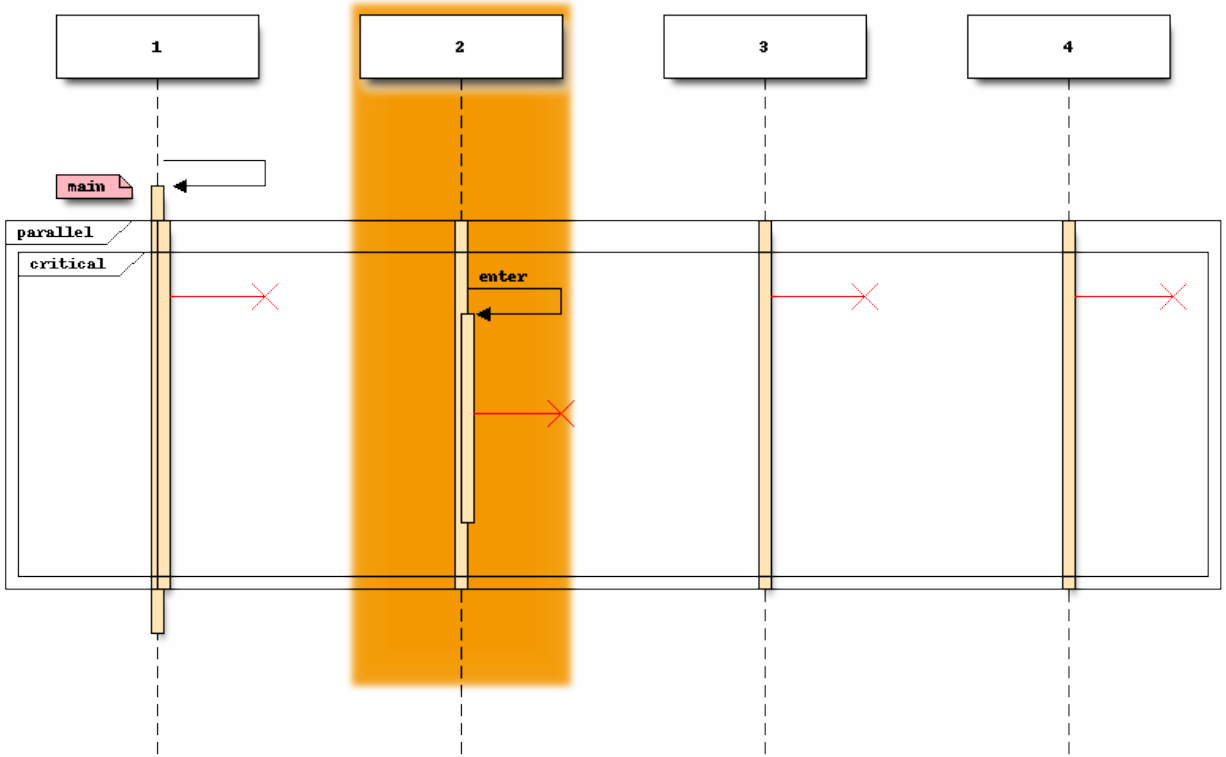
OpenMP/OMPss: mcGDB sequence diagram

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



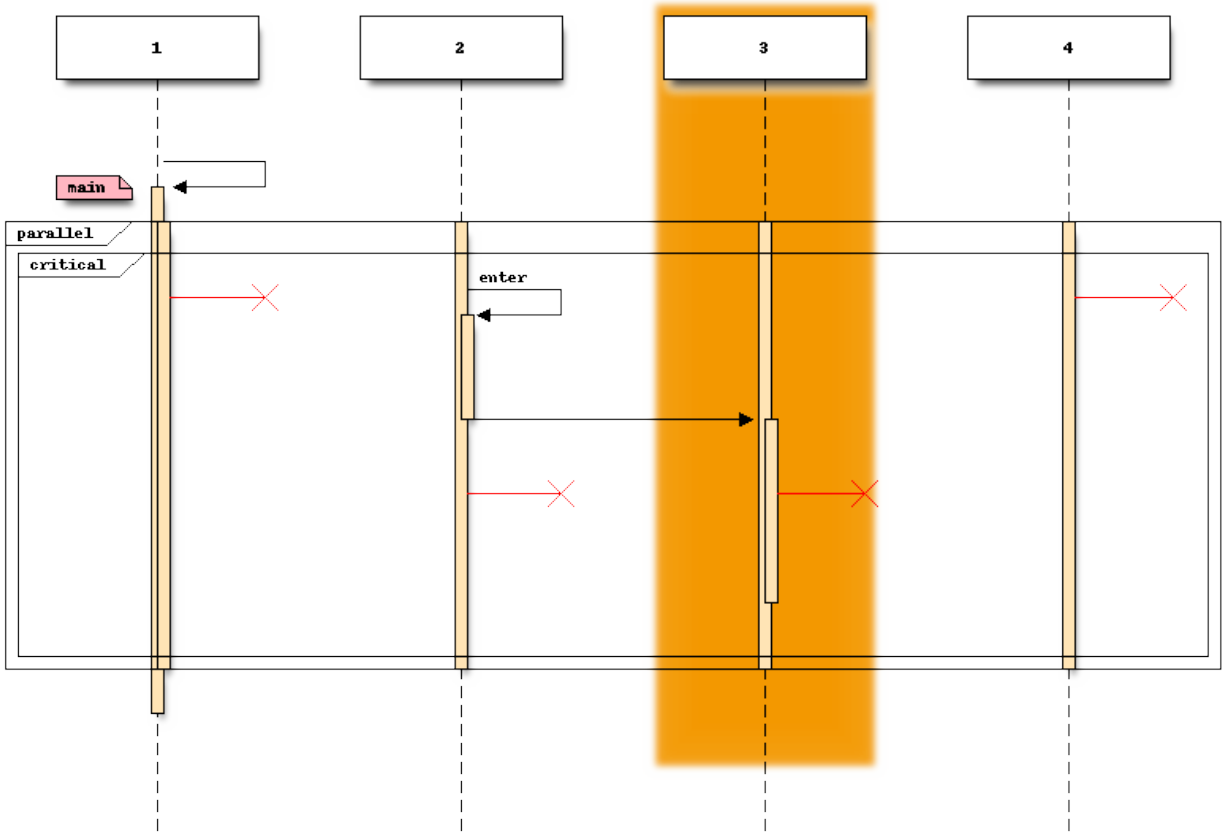
OpenMP/OMPss: mcGDB sequence diagram

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



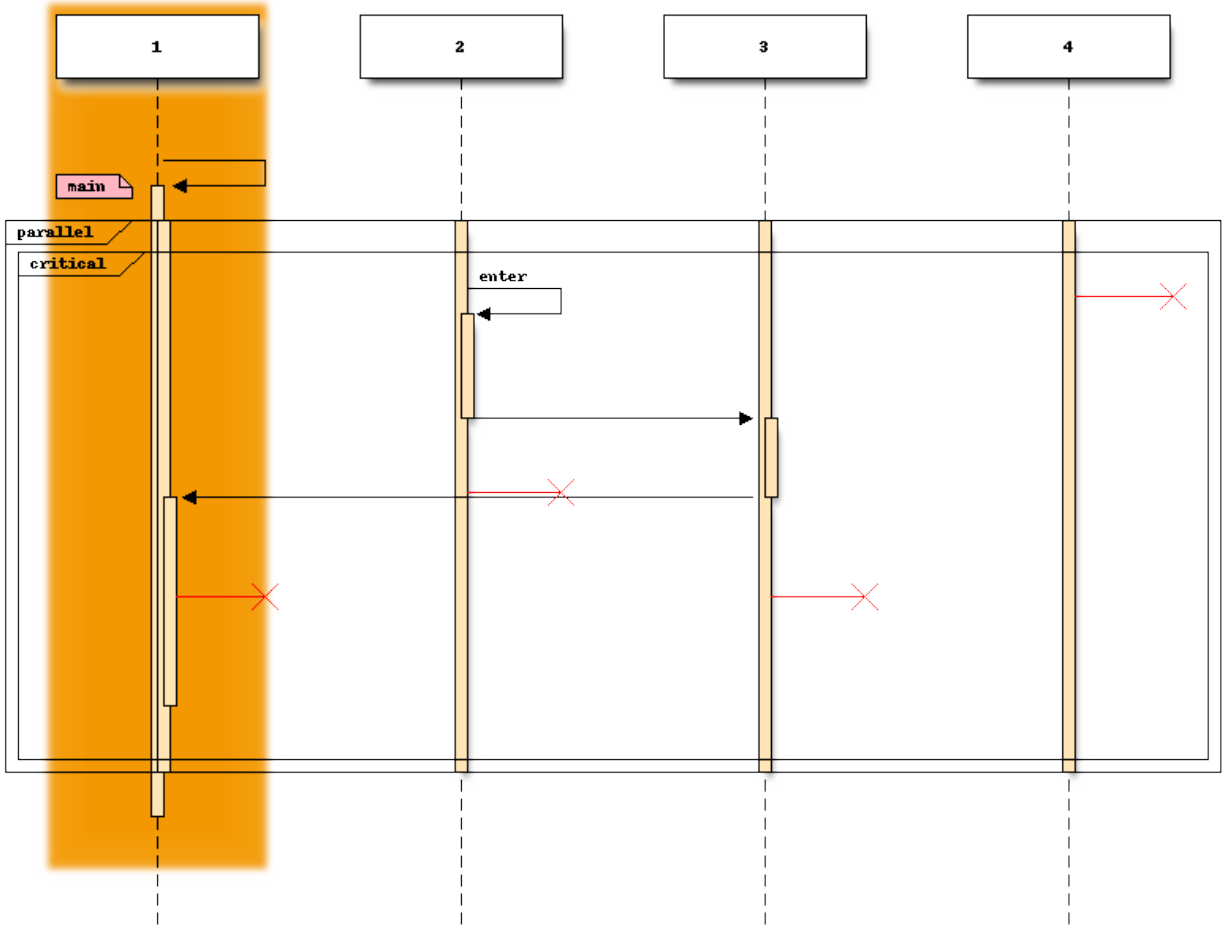
OpenMP/OMPss: mcGDB sequence diagram

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



OpenMP/OMPss: mcGDB sequence diagram

1. start
2. omp start
3. omp step
4. omp next barrier
5. thread 2
6. omp critical next
7. omp critical next
8. omp critical next



OpenMP/OMPss: structural representation

Compiler Optimization and Runtime SystEms

... provide a structural representation
... provide details about entity state

1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

OpenMP 4.0 task dependencies

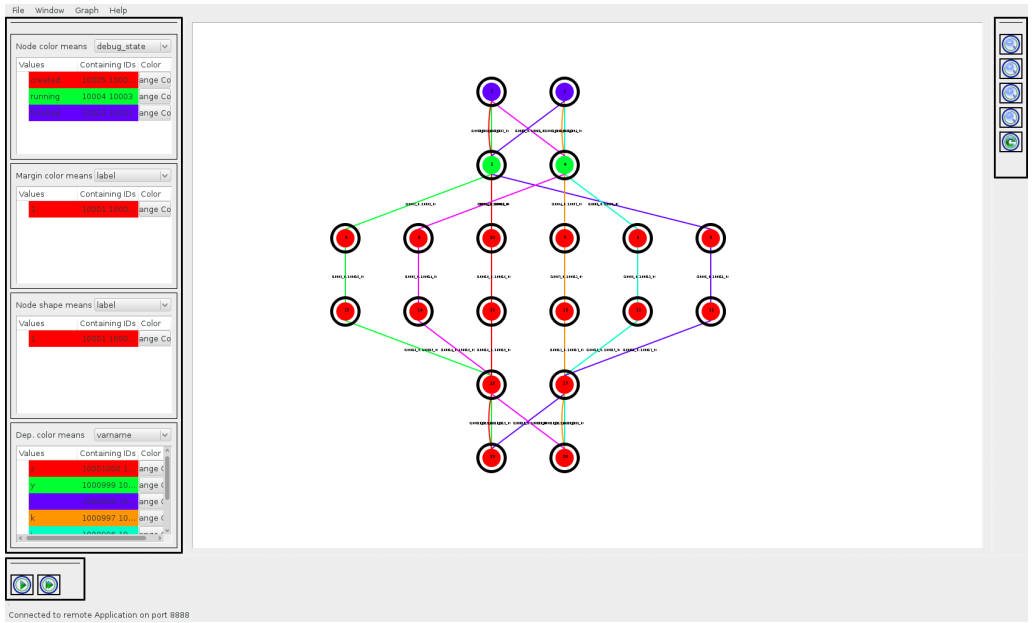
```
#pragma omp task depend(in: i,j) depend(out:i,j)
update_1(&i, &j);

#pragma omp task depend(in: i)   depend(out:i)
update_2a(&i);

#pragma omp task depend(in: j)   depend(out:j)
update_2b(&j);
```

(HLRS Stuttgart) Temanejo ...

✓ is a great visualization tool for task debugging,
✗ and does not support source-level debugging.



(HLRS Stuttgart) Temanejo ...

✓ is a great visualization tool for task debugging,

✗ and does not support source-level debugging.

mcGDB ...

✗ has no visualization engine,

✓ but provides source debugging at language (gdb) and model level.

OpenMP/OMPss: mcGDB + Temanejo

Compiler Optimization and Runtime SystEms

(HLRS Stuttgart) Temanejo ...

✓ is a great visualization tool for task debugging,

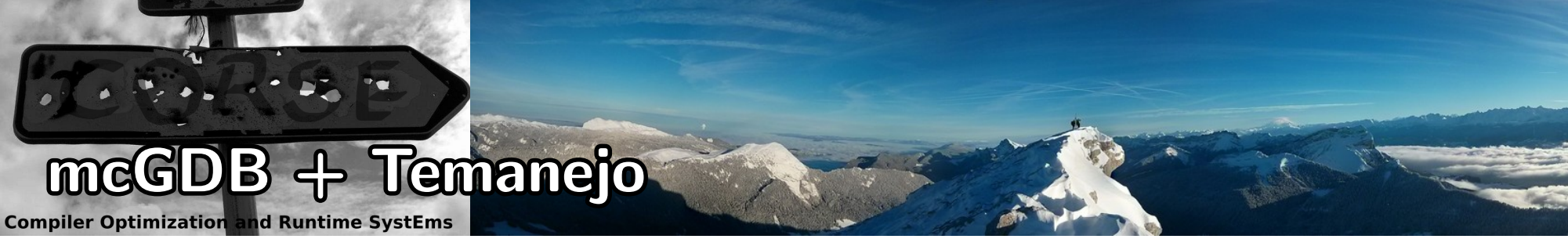
✗ and does not support source-level debugging.

mcGDB ...

✗ has no visualization engine,

✓ but provides source debugging at language (gdb) and model level.

So let's do both!



mcGDB – Temanejo cooperation:

Temanejo

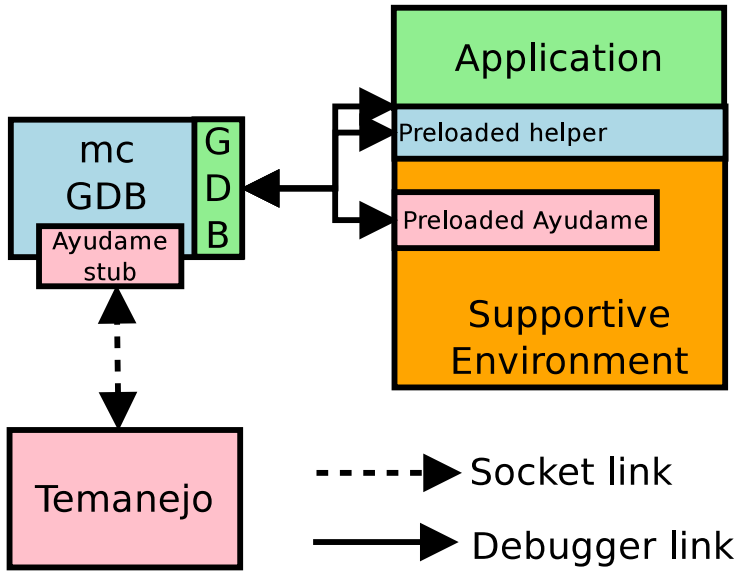
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

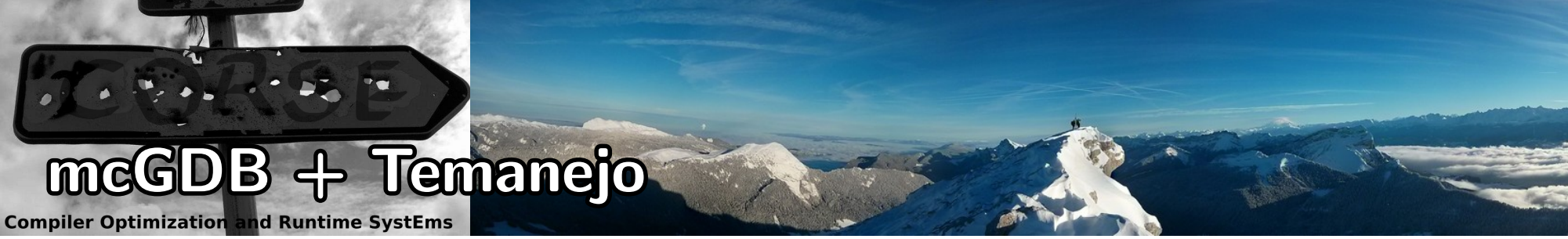
mcGDB

- task graph and exec. events capture,
- advanced model-level exec. control.

GDB

- language and assembly level
  execution control, memory inspection.





mcGDB – Temanejo cooperation:

Temanejo

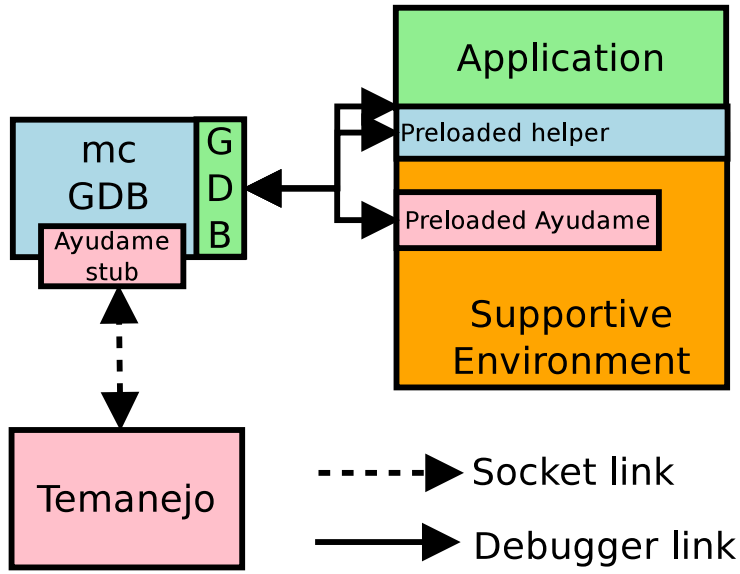
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

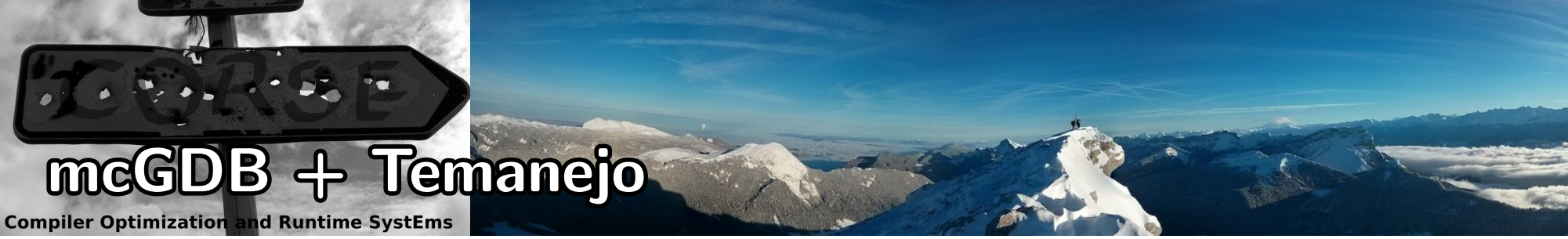
mcGDB

- task graph and exec. events capture,
- advanced model-level exec. control.

GDB

- language and assembly level execution control, memory inspection.





mcGDB – Temanejo cooperation:

Temanejo

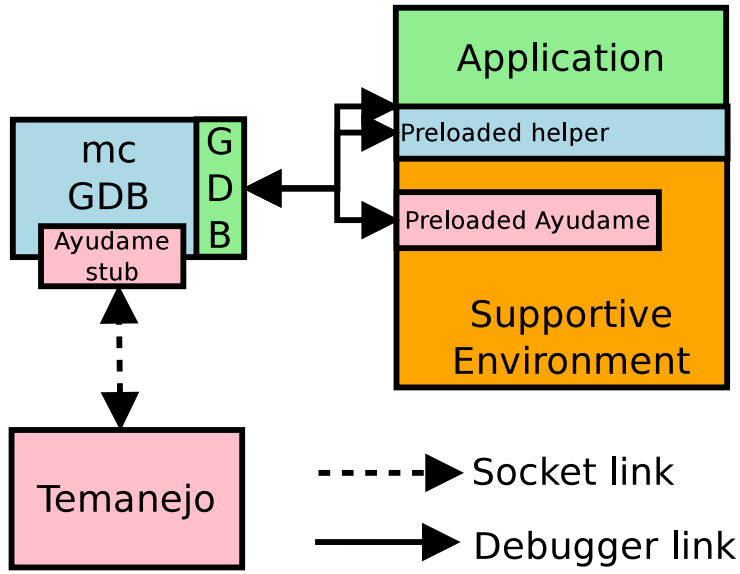
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

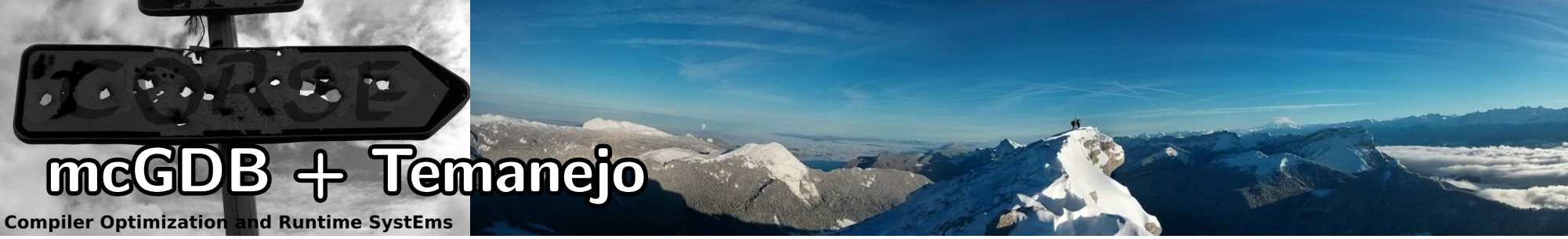
mcGDB

- task graph and exec. events capture,
- advanced model-level exec. control.

GDB

- language and assembly level execution control, memory inspection.





mcGDB – Temanejo cooperation:

Temanejo

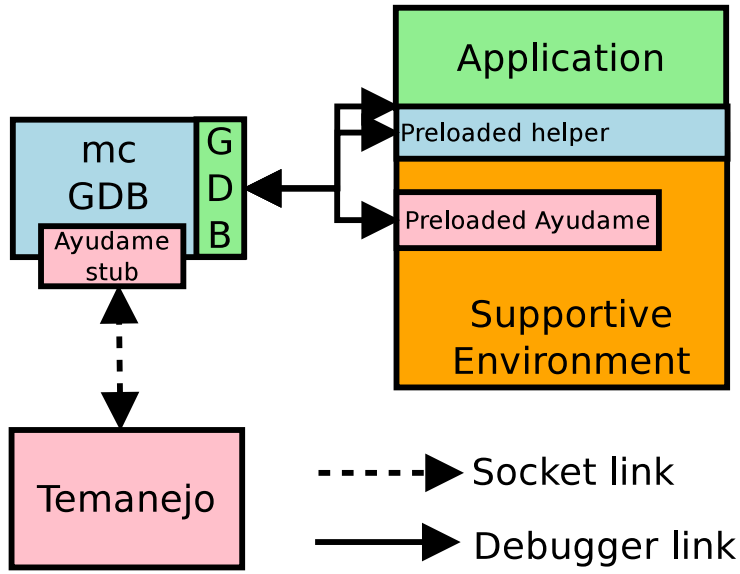
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

mcGDB

- task graph and exec. events capture,
- advanced model-level exec. control.

GDB

- language and assembly level execution control, memory inspection.

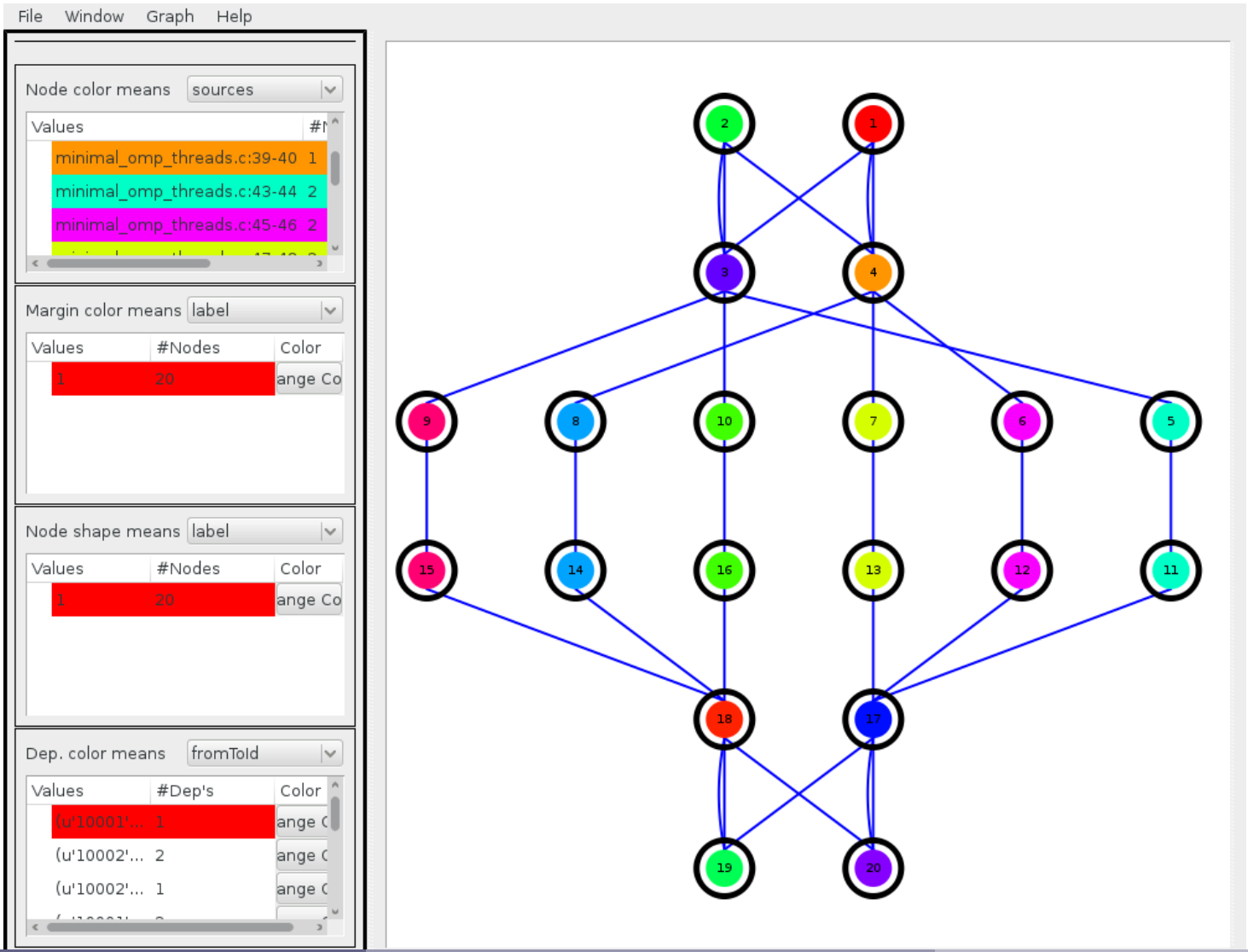



OpenMP/OMPss: structural representation
Compiler Optimization and Runtime SystEms

DEMO

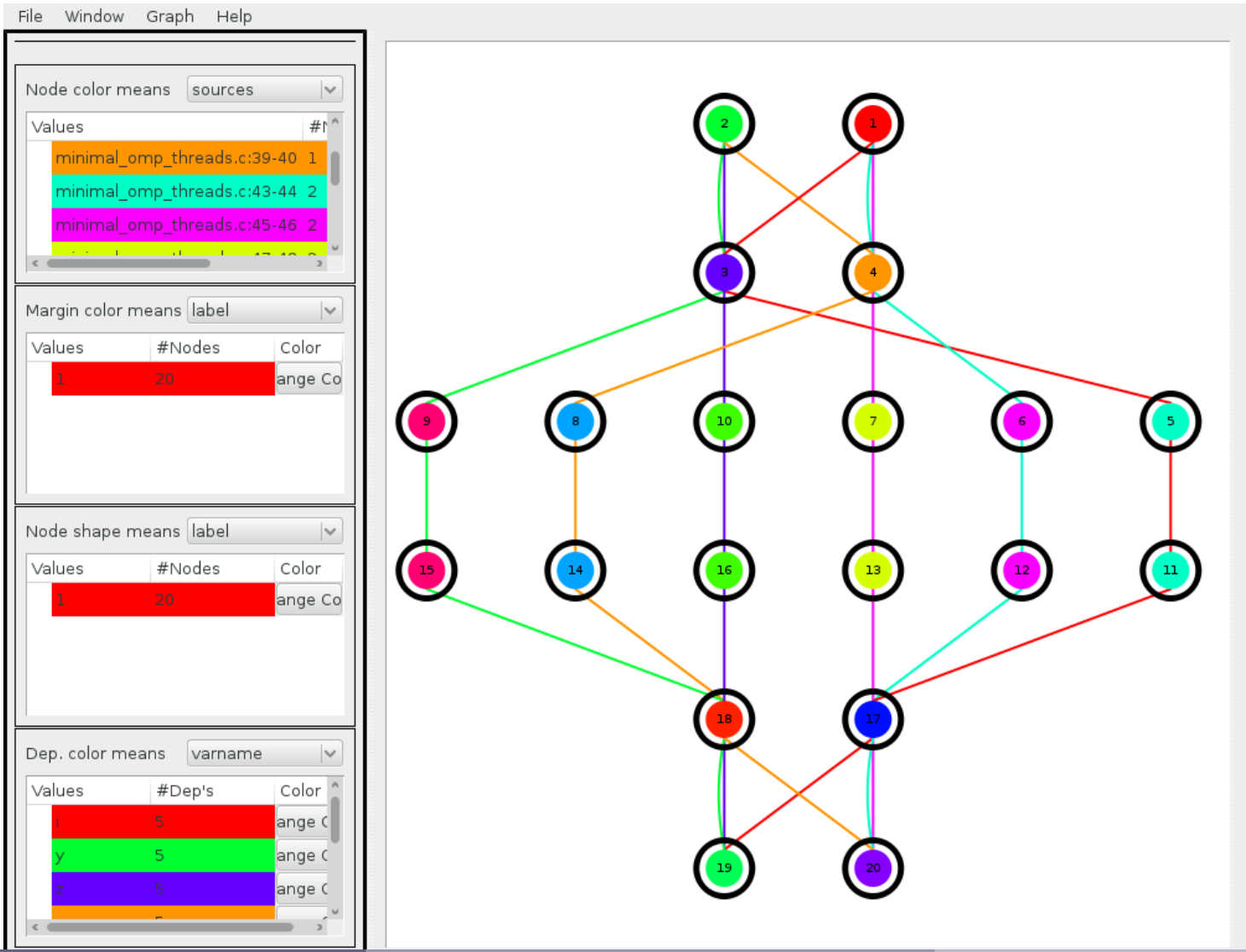
mcGDB + Temanejo: Node color means sources

Compiler Optimization and Runtime SystEms



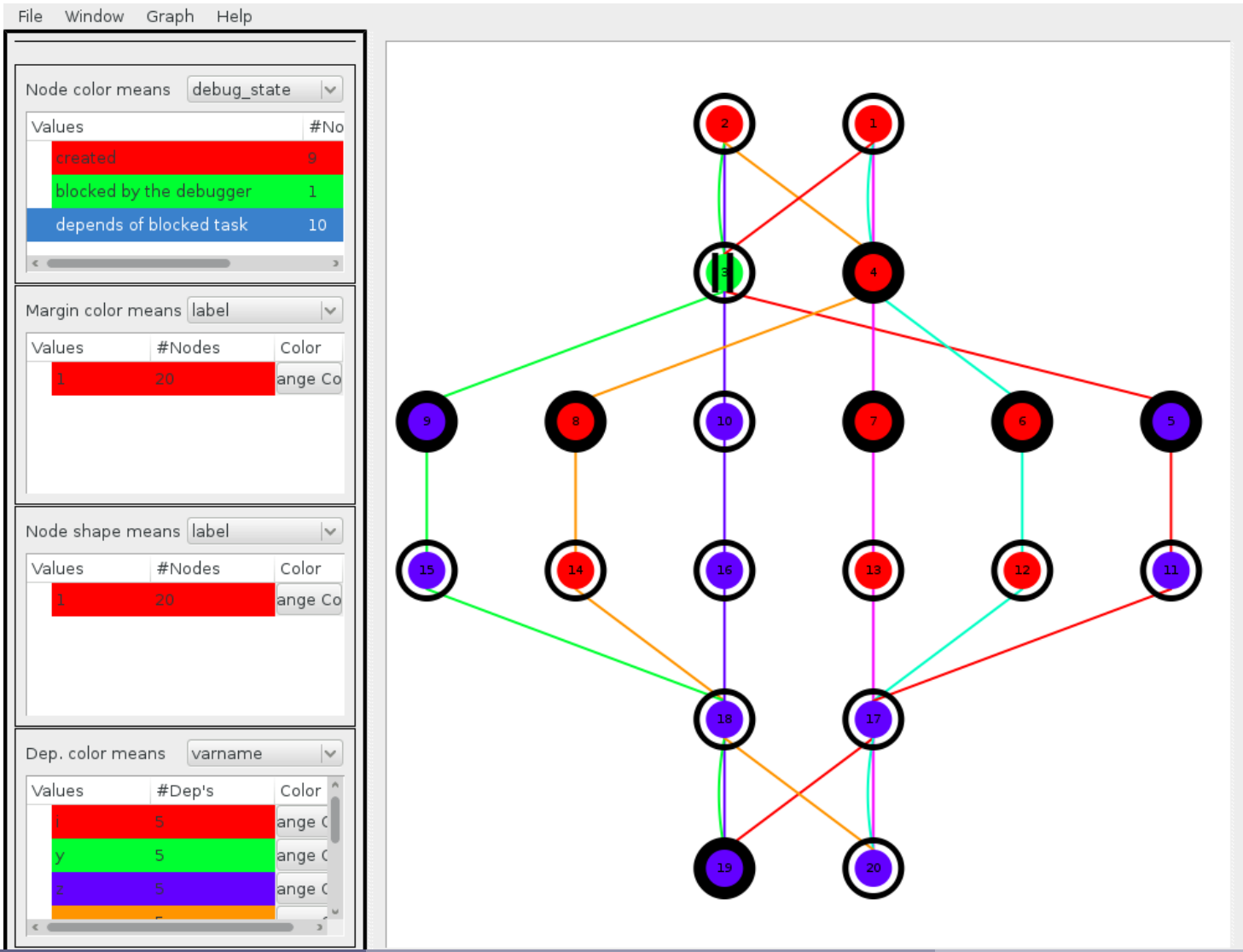
mcGDB + Temanejo: Dep. color means varname

Compiler Optimization and Runtime SystEms



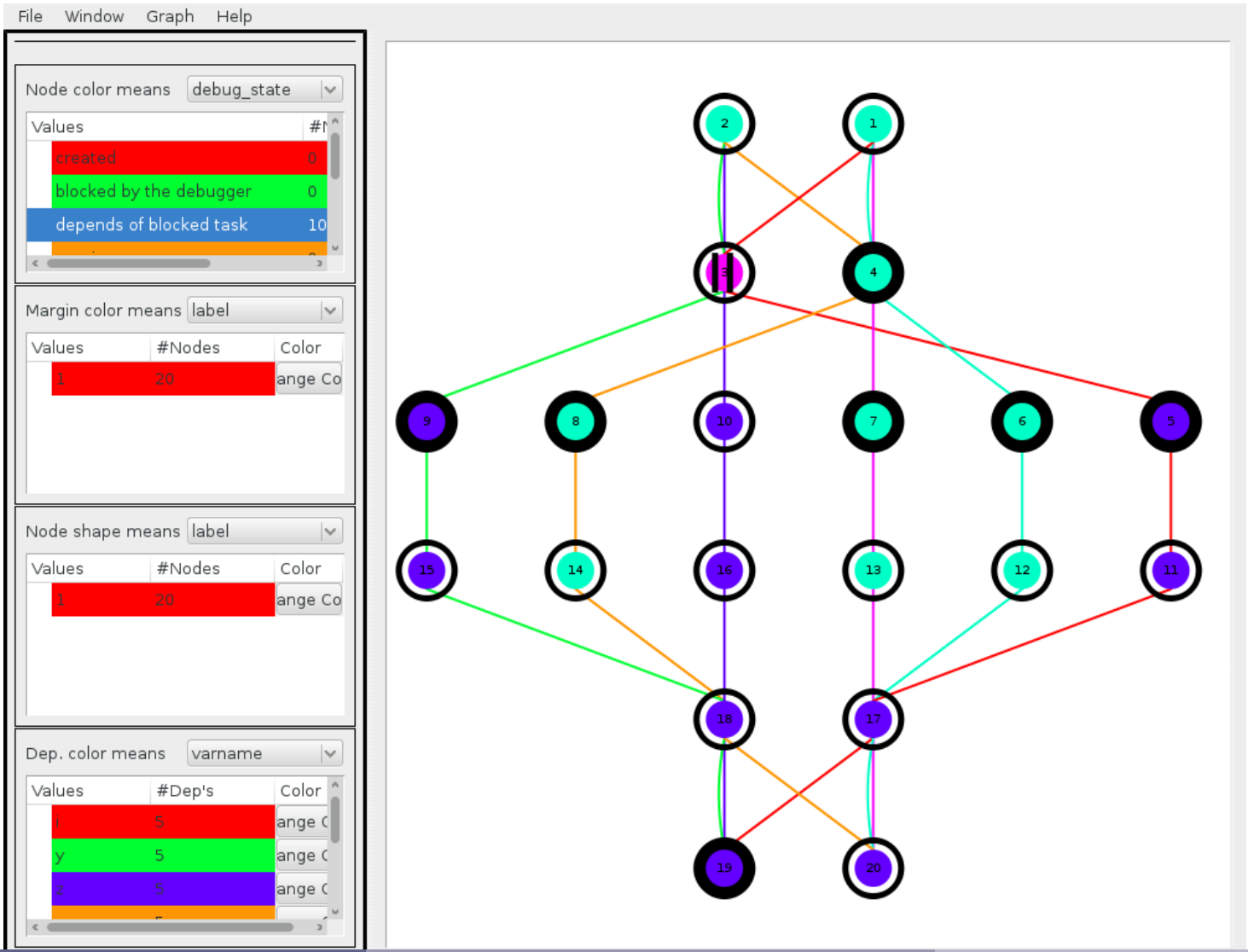
mcGDB + Temanejo: Task 3 blocked.

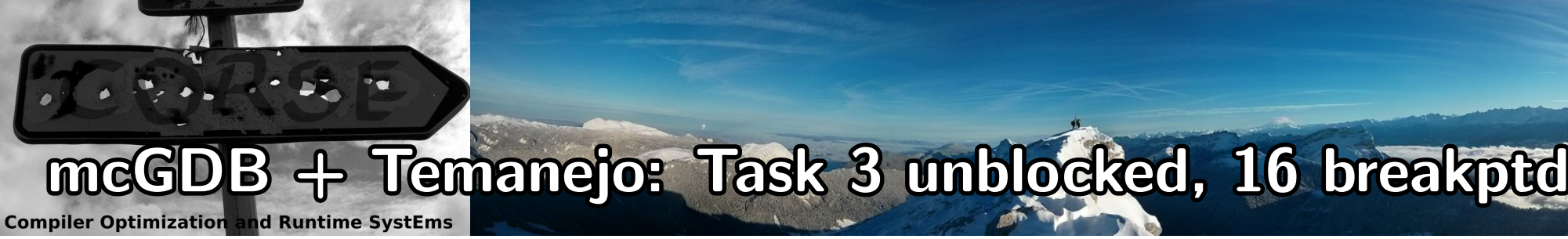
Compiler Optimization and Runtime SystEms



mcGDB + Temanejo: Execution continued til locked

Compiler Optimization and Runtime SystEms





File Window Graph Help

Node color means debug_state

Values	#N
finished	9
actively blocked by the debu...	0
created (unblocked)	11

Margin color means label

Values	#Nodes	Color
1	20	ange Co

Node shape means label

Values	#Nodes	Color
1	20	ange Co

Dep. color means varname

Values	#Dep's	Color
i	5	ange C
y	5	ange C
z	5	ange C

File Window Graph Help

Node color means executed_by

Values	#N
Worker #1	7
Worker #2	7
Worker #3	2

Margin color means label

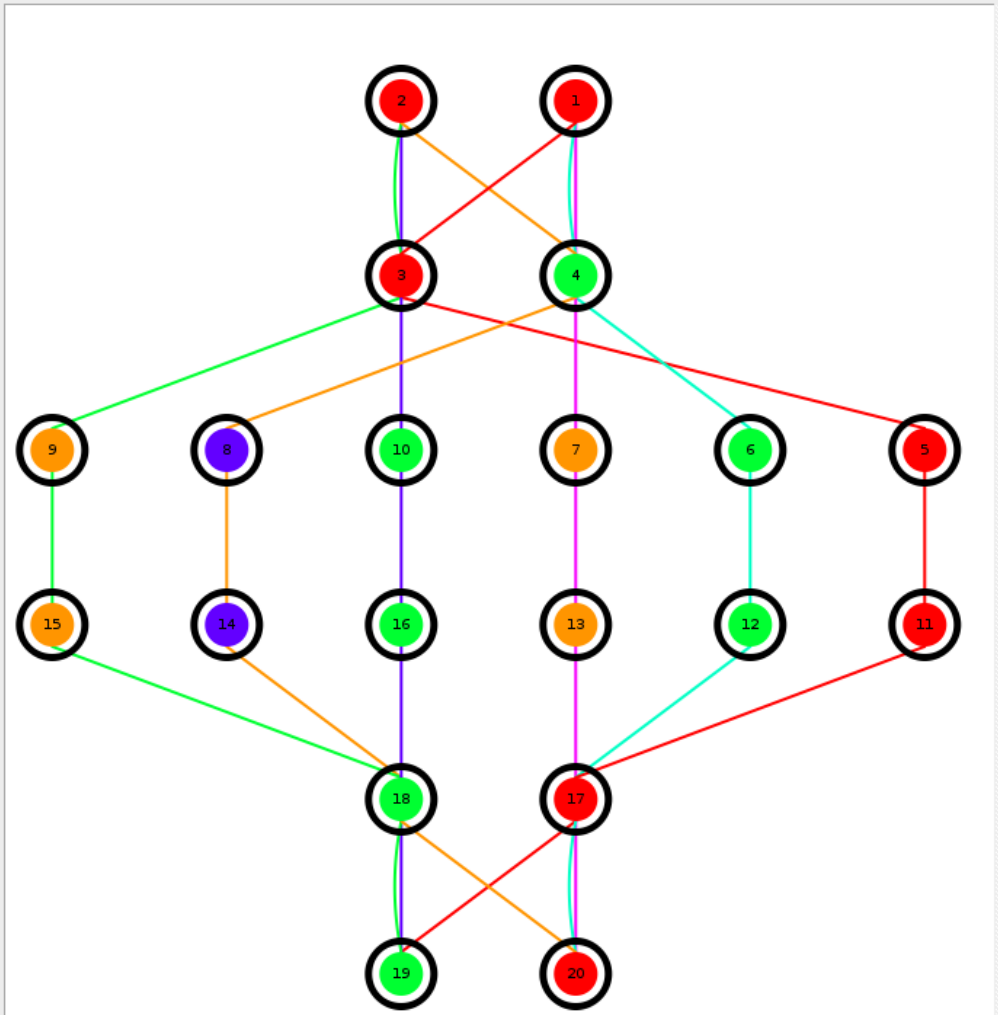
Values	#Nodes	Color
1	20	ange Co

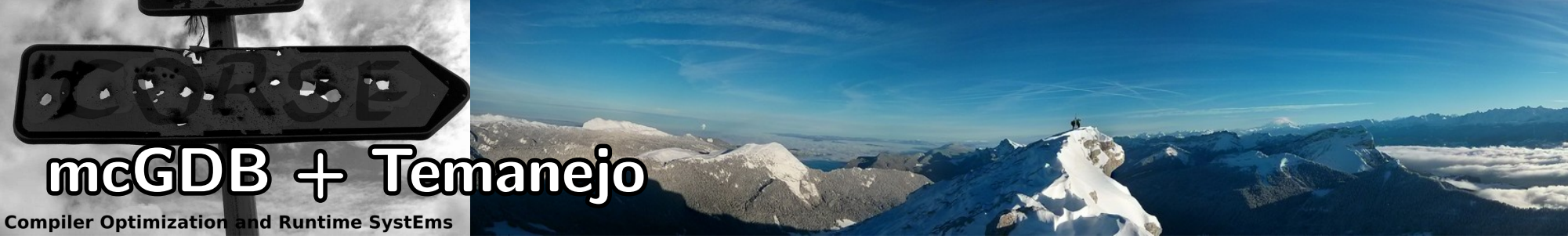
Node shape means label

Values	#Nodes	Color
1	20	ange Co

Dep. color means varname

Values	#Dep's	Color
i	5	ange C
y	5	ange C
z	5	ange C





More on OMP tasks

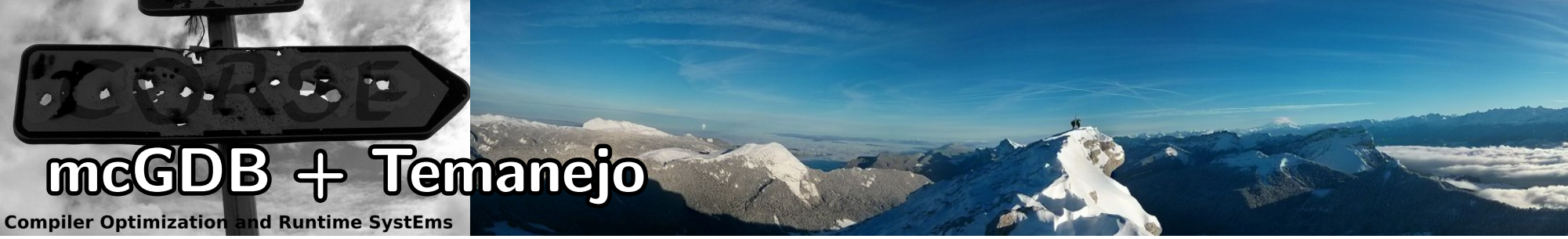
```
#pragma omp task depend(in: i,j)
```

Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:11][j:12])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears



More on OMP tasks

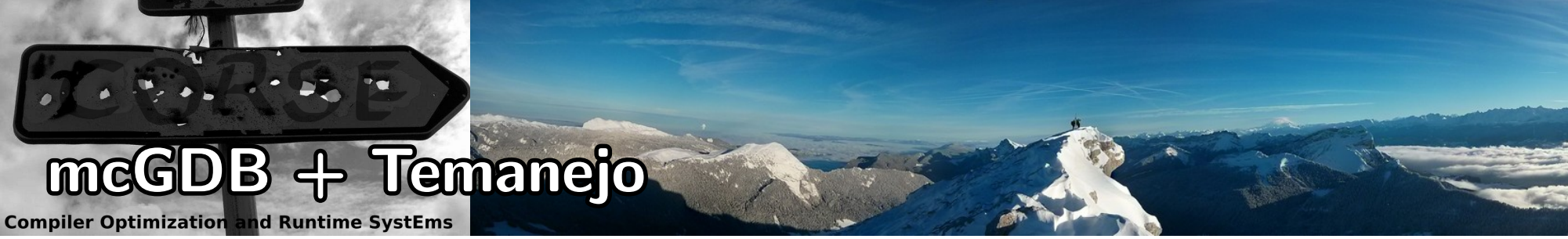
```
#pragma omp task depend(in: i,j)
```

Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:11][j:12])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears



More on OMP tasks

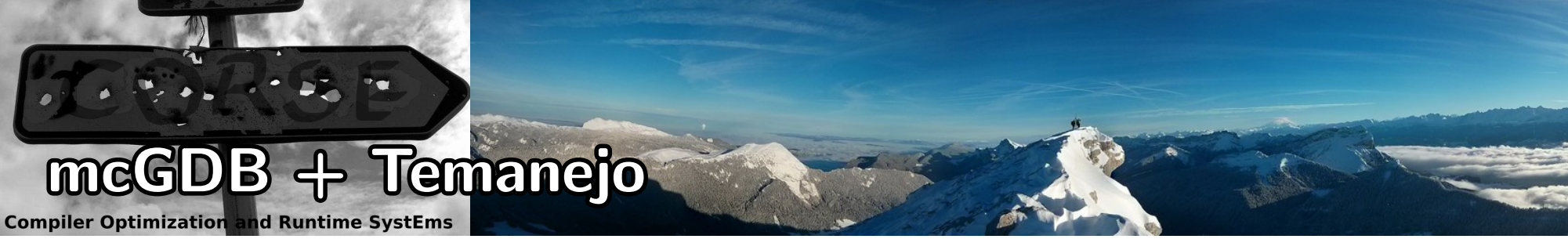
```
#pragma omp task depend(in: i,j)
```

Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:l1][j:l2])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears

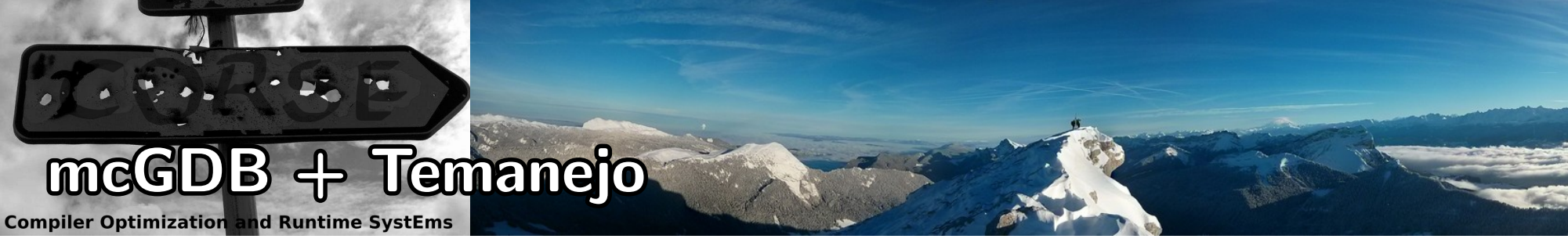


Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:l1][j:l2])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears
  - symbol mangling, array types stripped, ...
- tricky for me to implement!

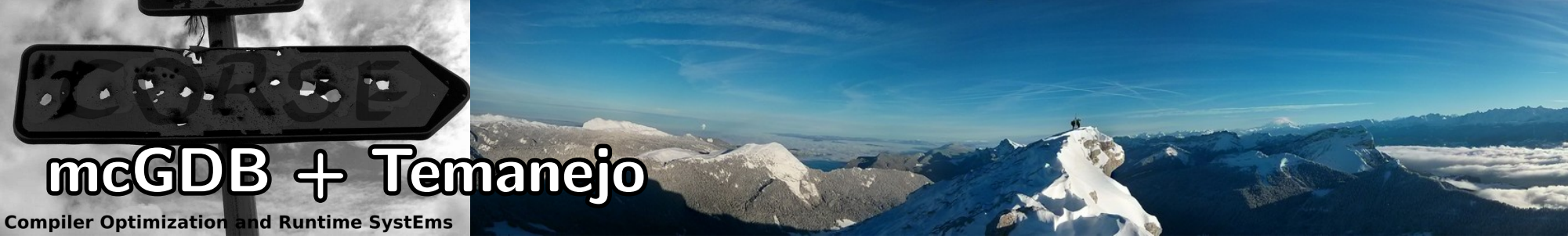


Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:l1][j:l2])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears
  - symbol mangling, array types stripped, ...
- tricky for me to implement!

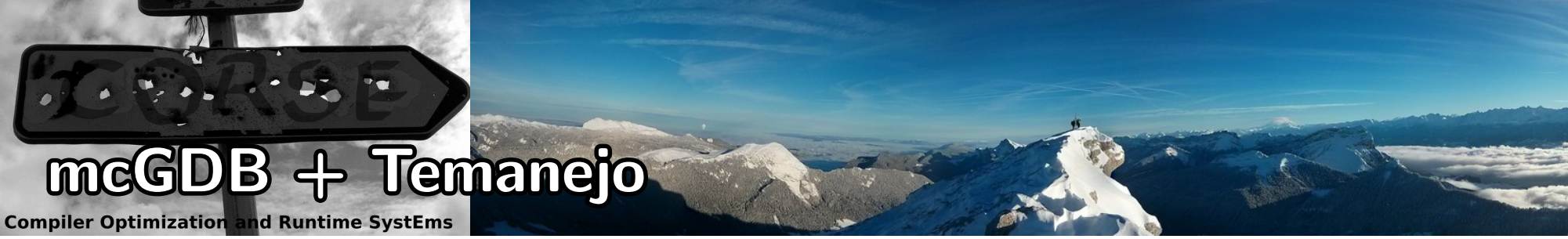


Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:11][j:12])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears
  - symbol mangling, array types stripped, ...
- tricky for me to implement!



Support on-going:

```
//arrays and matrix
#pragma omp task depend(in: array[i])
#pragma omp task depend(in: matrix[i][j])

//ranges (*not* overlapping)
#pragma omp task depend(in: array[i:j])
#pragma omp task depend(in: matrix[i:11][j:12])
```

- not verified by compiler, likely source of bug
- dynamic, so good match for interactive debugging
- but debugger unfriendly compilation:
  - array/index merged, `len` disappears
  - symbol mangling, array types stripped, ...
- tricky for me to implement!

OpenMP callstack

bad callstack of the threads `where`

```
Thread 3
#0 gomp_mutex_lock (mutex=0x7ffff7bd6748 <default_lock>)
#1 GOMP_critical_start ()
#2 in main._omp_fn.0 () at parallel-demo.c:17
#3 in GOMP_parallel_trampoline (arg=0x602060)
#4 in gomp_thread_start (xdata=<optimized out>)
#5 in pthread_create_trampoline (arg=0x602b10)
#6 in start_thread (
#7 in clone ()
Thread 2
#0  main._omp_fn.0 () at parallel-demo.c:17
#1  in gomp_thread_start (xdata=<optimized out>)
```

OpenMP callstack

bad callstack of the threads where

```
Thread 3
#0 gomp_mutex_lock (mutex=0x7ffff7bd6748 <default_lock>)
#1 GOMP_critical_start ()
#2 in main._omp_fn.0 () at parallel-demo.c:17
#3 in GOMP_parallel_trampoline (arg=0x602060)
#4 in gomp_thread_start (xdata=<optimized out>)
#5 in pthread_create_trampoline (arg=0x602b10)
#6 in start_thread (
#7 in clone ()
Thread 2
#0  main._omp_fn.0 () at parallel-demo.c:17
#1  in gomp_thread_start (xdata=<optimized out>)
```

OpenMP/OMPss: OpenMP callstack

Compiler Optimization and Runtime SystEms

OpenMP callstack

bad	callstack of the threads	`where`
better	OpenMP aware thread callstack	`where no-filter`

```
Thread 3
#0  gomp_mutex_lock (...)
#1  #pragma omp critical_start ()
#2  #parallel zone 1 of main () at parallel-demo.c:17
Thread 2
#0  #parallel zone 1 of main () at parallel-demo.c:17
Thread 1
#0  #parallel zone 1 of main () at parallel-demo.c:19
#2  #pragma omp parallel ()
#4  main () at parallel-demo.c:5
```

OpenMP callstack

bad	callstack of the threads	`where`
better	OpenMP aware thread callstack	`where no-filter`

```
Thread 3
#0  gomp_mutex_lock (...)
#1  #pragma omp critical_start ()
#2  #parallel zone 1 of main () at parallel-demo.c:17
Thread 2
#0  #parallel zone 1 of main () at parallel-demo.c:17
Thread 1
#0  #parallel zone 1 of main () at parallel-demo.c:19
#2  #pragma omp parallel ()
#4  main () at parallel-demo.c:5
```

OpenMP/OMPss: OpenMP callstack

Compiler Optimization and Runtime SystEms

OpenMP callstack

bad	callstack of the threads	`where`
better	OpenMP aware thread callstack	`where no-filter`
best	OpenMP tree callstack	`omp where`

```
   main () at parallel-demo.c:5
      #pragma omp parallel
         #parallel zone #1 of main
-->      + at parallel-demo.c:17                [Thread 1, 4]
            #pragma omp critical_start ()
-->            gomp_mutex_lock (...)            [Thread 3]
-->      + at parallel-demo.c:19                [Thread 2]
```

OpenMP/OMPss: runtime support

Compiler Optimization and Runtime SystEms

- Intel OpenMP and GOMP:
  - breakpoints
  - `libmcgdb-omp` preloaded helper
    - ⋆ full support, main targets

- OMPss tasks:
  - Some breakpoints, `libmcgdb-omp` not implemented
    - ⋆ only task support
  - Initial support of Ayudame helper
    - ⋆ HLRS Ayudame/Temanejo targets for free

- OMP Trace/Debugging Interfaces
  - No support :(task dependencies missing) but it looks promising!!
    - ⋆ runtime agnostic mcGDB support

OpenMP/OMPss: runtime support

Compiler Optimization and Runtime SystEms

- Intel OpenMP and GOMP:
  - breakpoints
  - `libmcgdb-omp` preloaded helper
    - ★ full support, main targets

- OMPss tasks:
  - Some breakpoints, `libmcgdb-omp` not implemented
    - ★ only task support
  - Initial support of Ayudame helper
    - ★ HLRS Ayudame/Temanejo targets for free

- OMP Trace/Debugging Interfaces
  - No support :(task dependencies missing) but it looks promising!!
    - ★ runtime agnostic mcGDB support

OpenMP/OMPss: runtime support

Compiler Optimization and Runtime SystEms

- Intel OpenMP and GOMP:
  - breakpoints
  - `libmcgdb-omp` preloaded helper
    - ★ full support, main targets
- OMPss tasks:
  - Some breakpoints, `libmcgdb-omp` not implemented
    - ★ only task support
  - Initial support of Ayudame helper
    - ★ HLRS Ayudame/Temanejo targets for free
- OMP Trace/Debugging Interfaces
  - No support :(task dependencies missing) but it looks promising!!
    - ★ runtime agnostic mcGDB support

OpenMP/OMPss: runtime support

Compiler Optimization and Runtime SystEms

- Intel OpenMP and GOMP:
  - breakpoints
  - `libmcgdb-omp` preloaded helper
    - ★ full support, main targets
- OMPss tasks:
  - Some breakpoints, `libmcgdb-omp` not implemented
    - ★ only task support
  - Initial support of Ayudame helper
    - ★ HLRS Ayudame/Temanejo targets for free
- OMP Trace/Debugging Interfaces
  - No support :(task dependencies missing) but it looks promising!!
    - ★ runtime agnostic mcGDB support

Agenda

Compiler Optimization and Runtime SystEms

Conclusion

- OpenMP is a good target for model-centric debugging:
  - fork-join aware execution control and sequence diagram visualization
  - tree-based callstack
  - task graph visualization
  - task dependency verification and visualization
- OMP Trace API support would be great, but not yet possible

Conclusion

- OpenMP is a good target for model-centric debugging:
  - fork-join aware execution control and sequence diagram visualization
  - tree-based callstack
  - task graph visualization
  - task dependency verification and visualization
- OMP Trace API support would be great, but not yet possible

Conclusion

- OpenMP is a good target for model-centric debugging:
  - fork-join aware execution control and sequence diagram visualization
  - tree-based callstack
  - task graph visualization
  - task dependency verification and visualization
- OMP Trace API support would be great, but not yet possible

Future work

- Finish and polish array task dependency support
- Validate with capture of KASTORS benchmark task graph
  - ... and benchmark mcGDB slowdown
- OpenMP performance debugging/profiling: based on tasks?

Future work

- Finish and polish array task dependency support
- Validate with capture of KASTORS benchmark task graph
  - ... and benchmark mcGDB slowdown
- OpenMP performance debugging/profiling: based on tasks?