Programming-Model Centric Debugging for OpenMP/OMPss

Kevin Pouget
Jean-François Méhaut, Miguel Santana

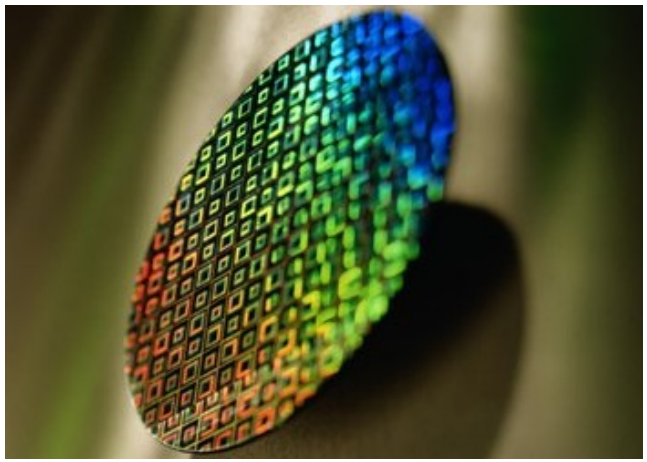
Université Grenoble Alpes / LIG, STMicroelectronics, France
Nano2017-DEMA project

4^{th} JLESC Workshop, Bonn, Germany
December 3^{rd}, 2015

Introduction: Embedded Systems and MPSoC

Convergence of Embedded and HPC

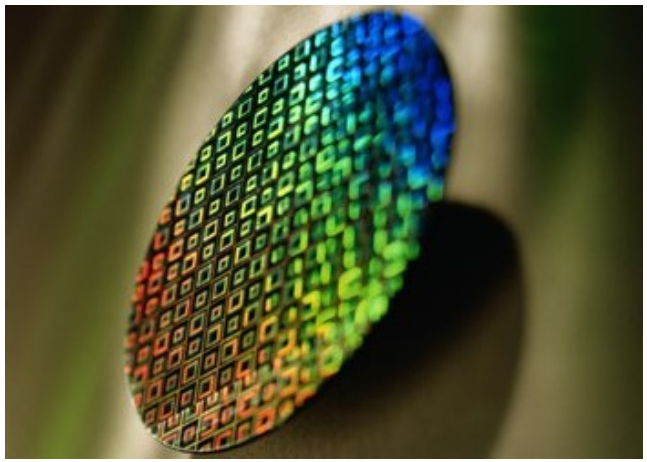
- HPC in embedded systems
  - high-def. multimedia
  - augmented reality
  - video games on smartphones
- Embedded systems in HPC
  - dedicated hardware accelerators
  - energy efficiency
    - e.g. Mont-Blanc projects



Introduction: Embedded Systems and MPSoC

⇒ important demand for:

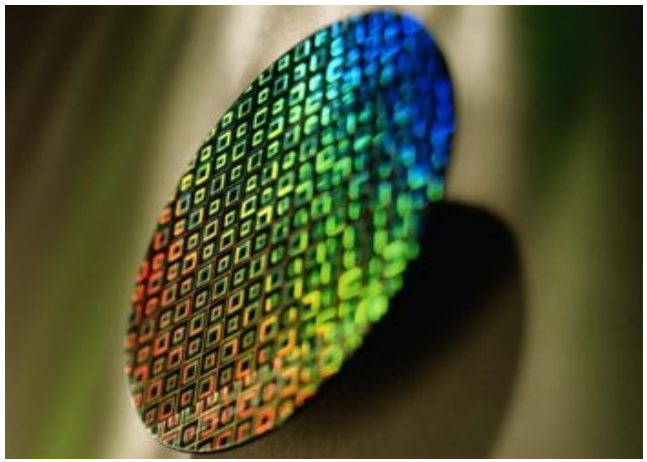
- Powerful parallel architectures
- High-level development methodologies
- Efficient verification & validation tools



Introduction: Embedded Systems and MPSoC

⇒ important demand for:

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

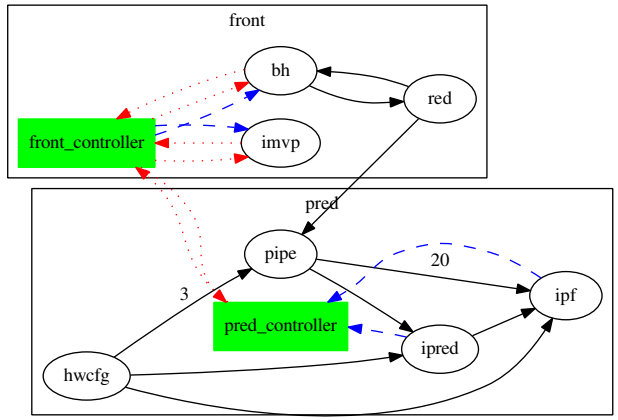


Introduction: Embedded Systems and MPSoC

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

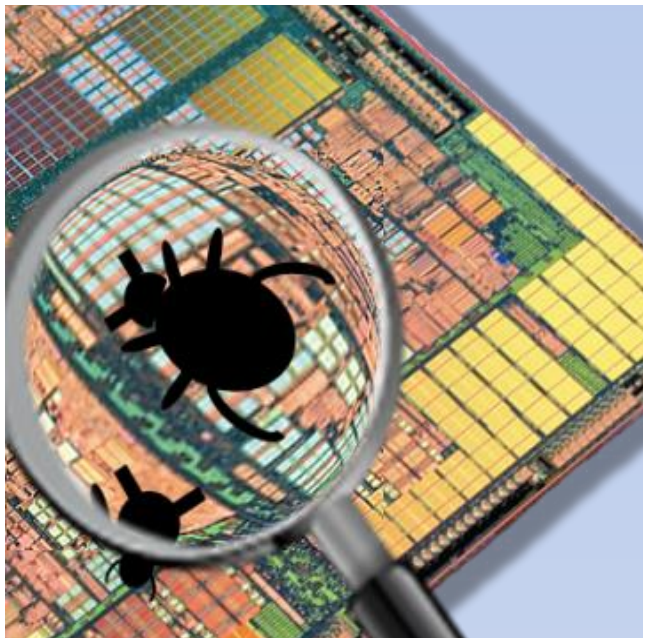


Introduction: Embedded Systems and MPSoC

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

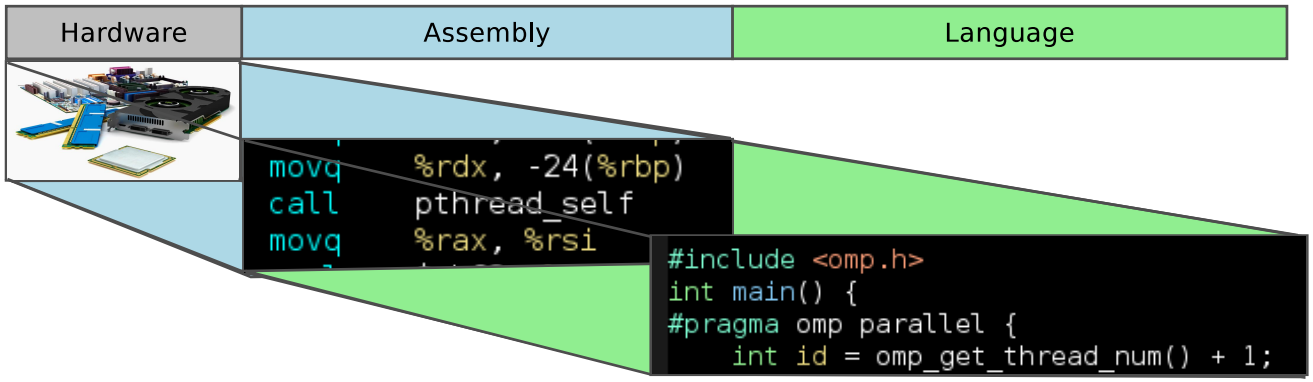


Agenda

Compiler Optimization and Runtime SystEms

Verification and Validation: Debugging

Compiler Optimization and Runtime SystEms

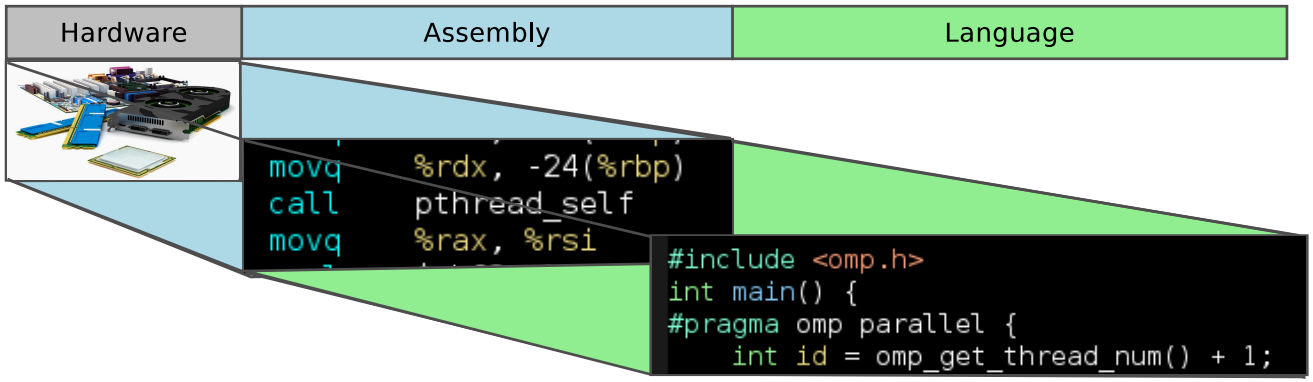


Source-Level Interactive Debugging (e.g. GDB)

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

Verification and Validation: Debugging

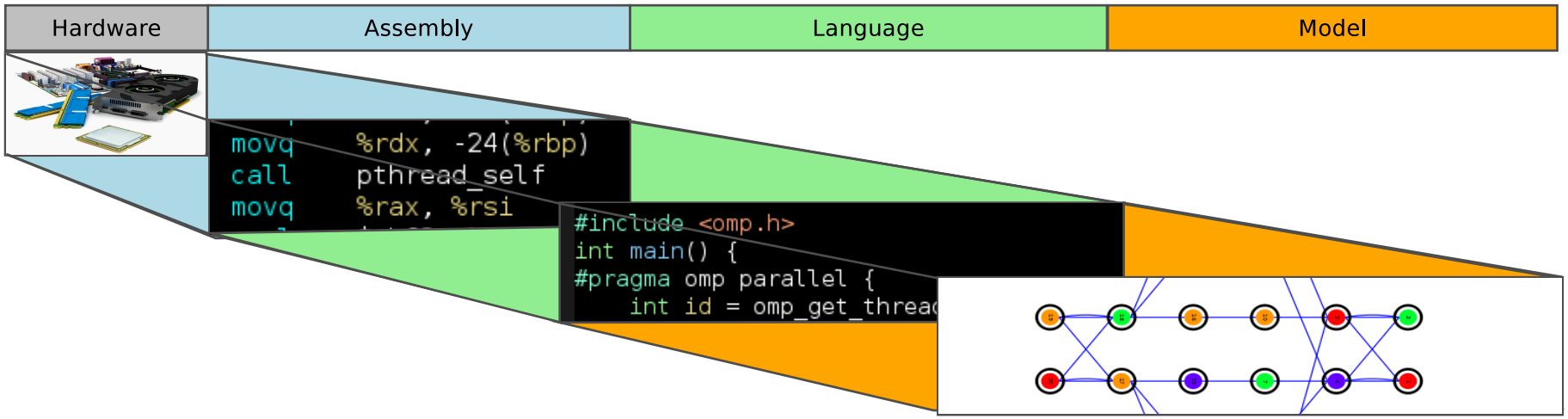
Compiler Optimization and Runtime SystEms



What about programming models?

Verification and Validation: Debugging

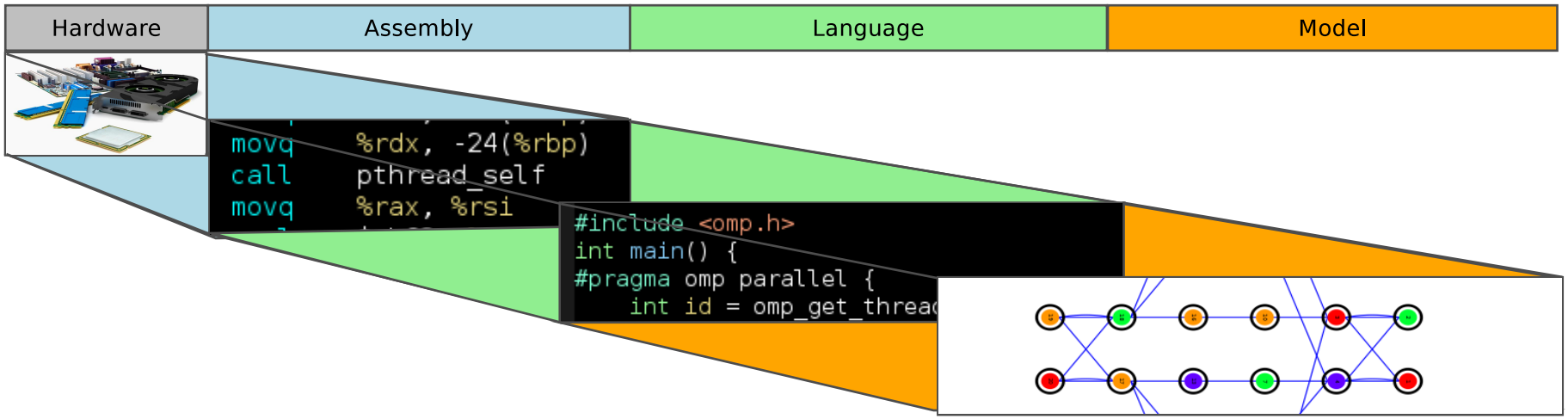
Compiler Optimization and Runtime SystEms



What about programming models?

Verification and Validation: Debugging

Compiler Optimization and Runtime SystEms

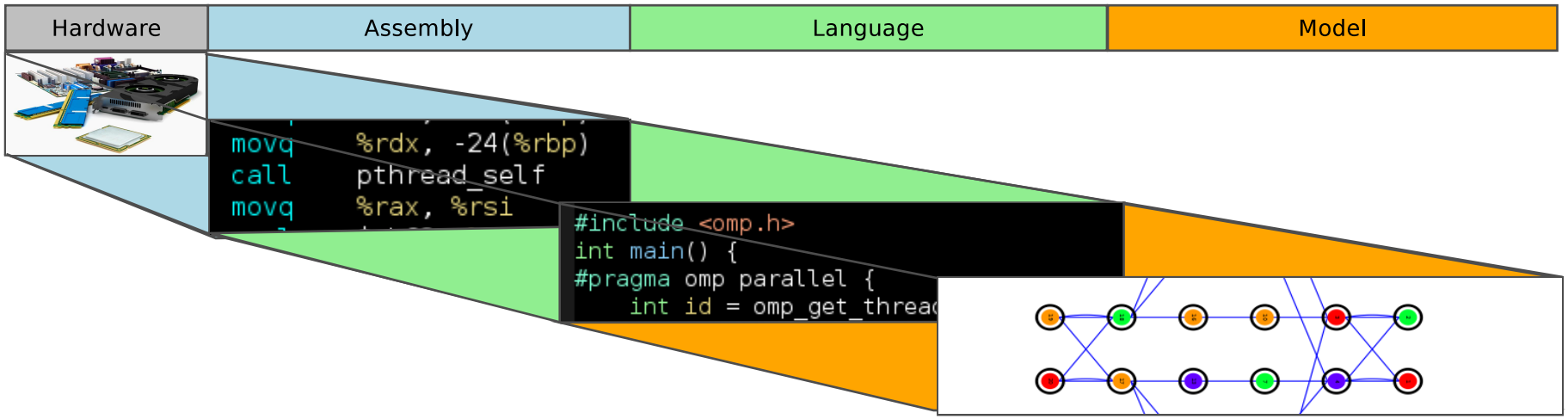


What about programming models?

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!

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.

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

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

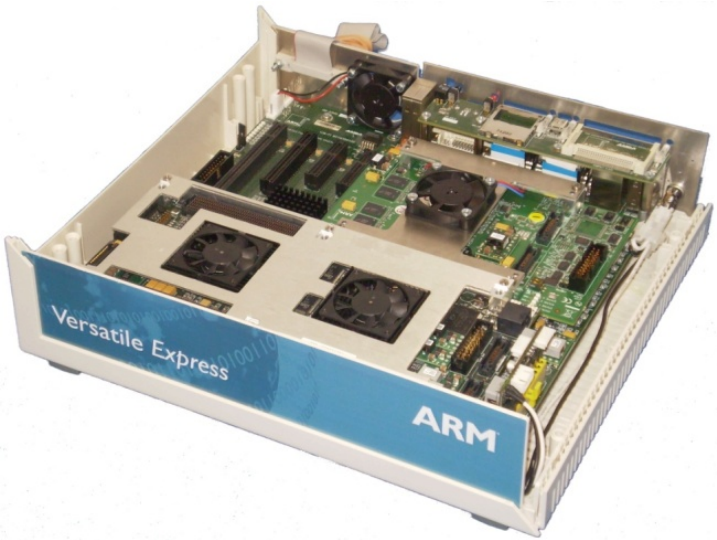
1. Provide a Structural Representation
   - Draw application architecture diagrams
   - Represent the relationship between the entities
2. Monitor Dynamic Behaviors
   - Monitor the collaboration between the tasks
   - Detect communication, synchronization events
3. Interact with the Abstract Machine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

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!

Debugging Embedded and Multicore Applications

ARM Juno



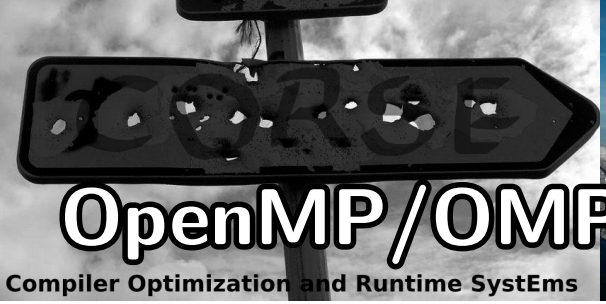
- asymmetric archi.
- ARM big.LITTLE + Mali GPU

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 ST



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

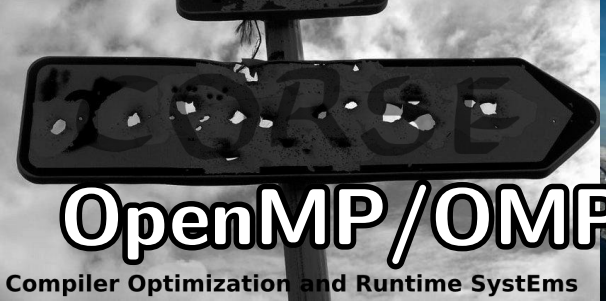
1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

OpenMP/OMPss: structural representation

Compiler Optimization and Runtime SystEms

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

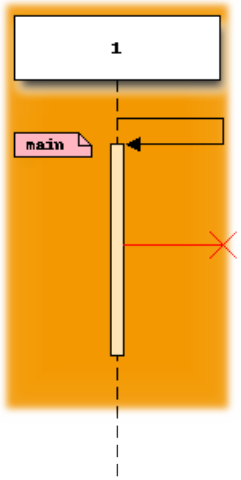
1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

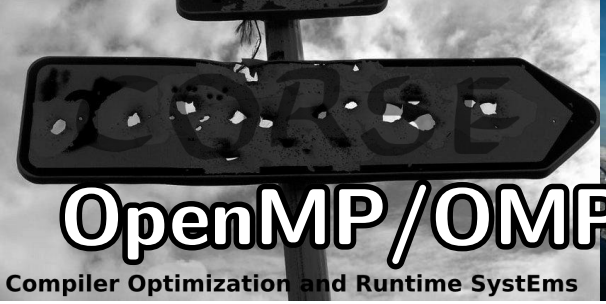


OpenMP/OMPss: mcGDB sequence diagram

Compiler Optimization and Runtime SystEms

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

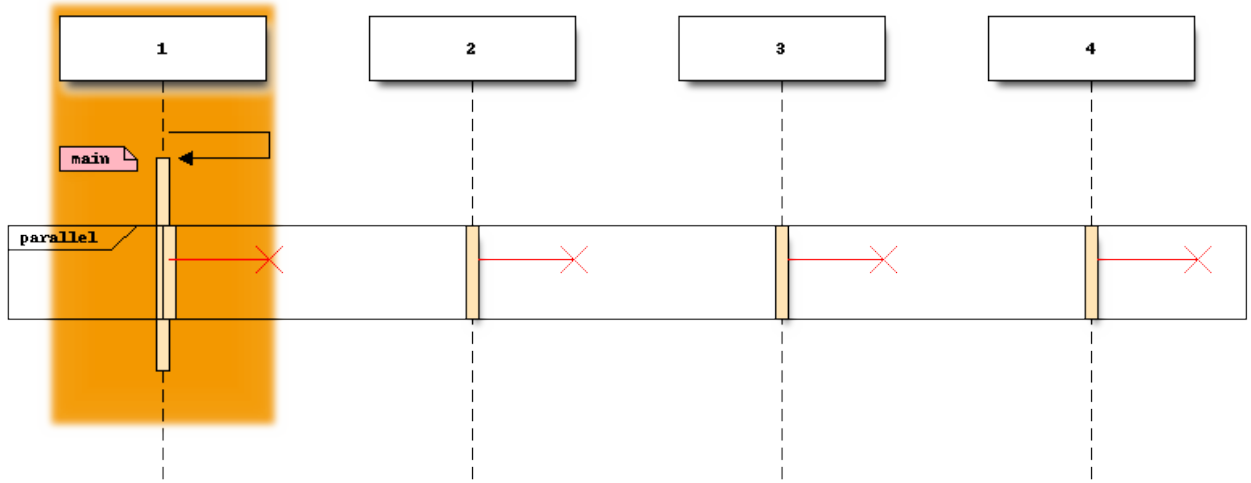




OpenMP/OMPss: mcGDB sequence diagram

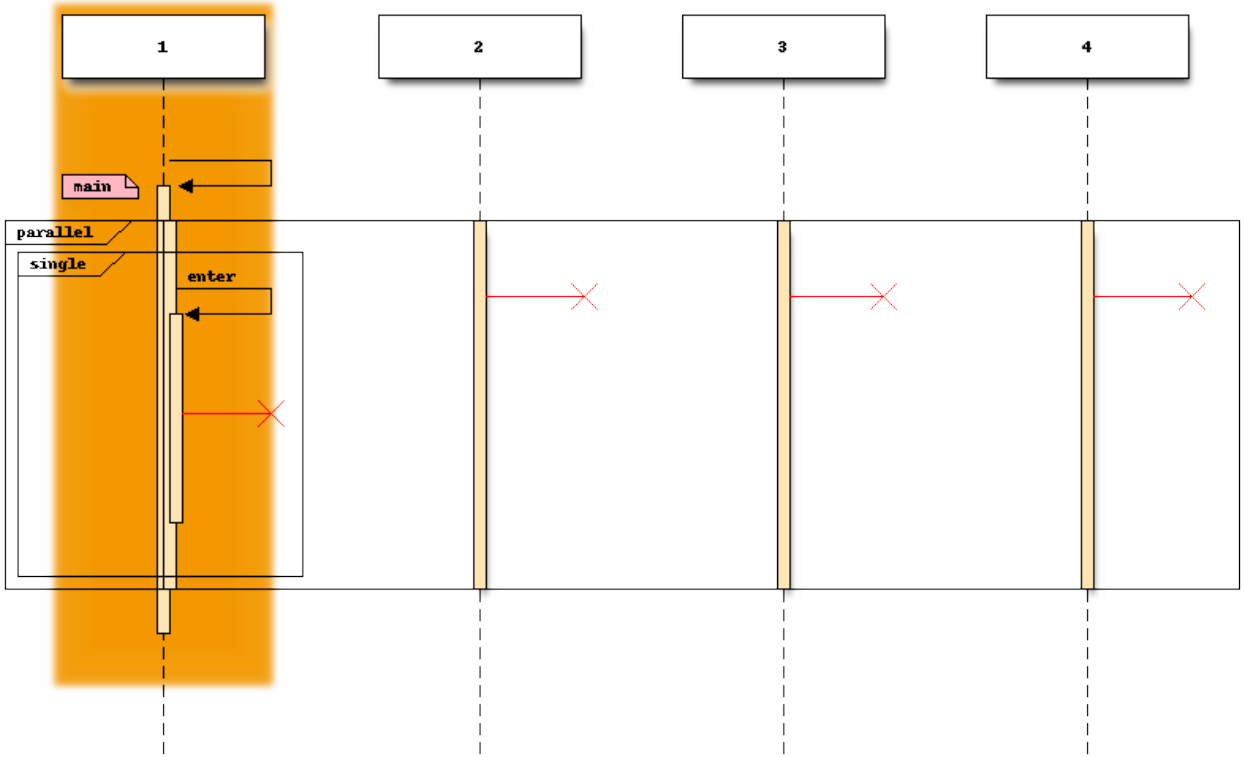
Compiler Optimization and Runtime SystEms

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



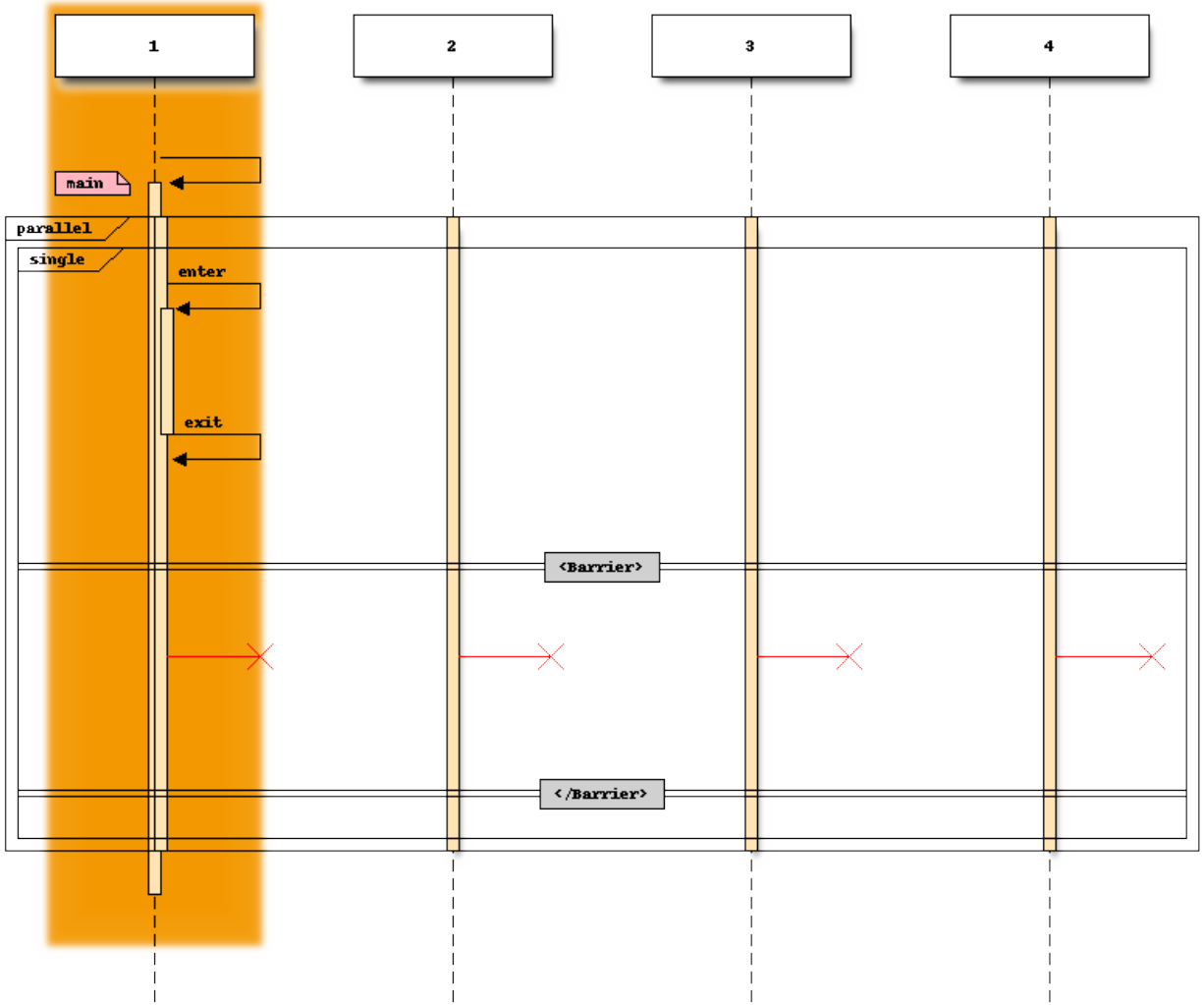
OpenMP/OMPss: mcGDB sequence diagram

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



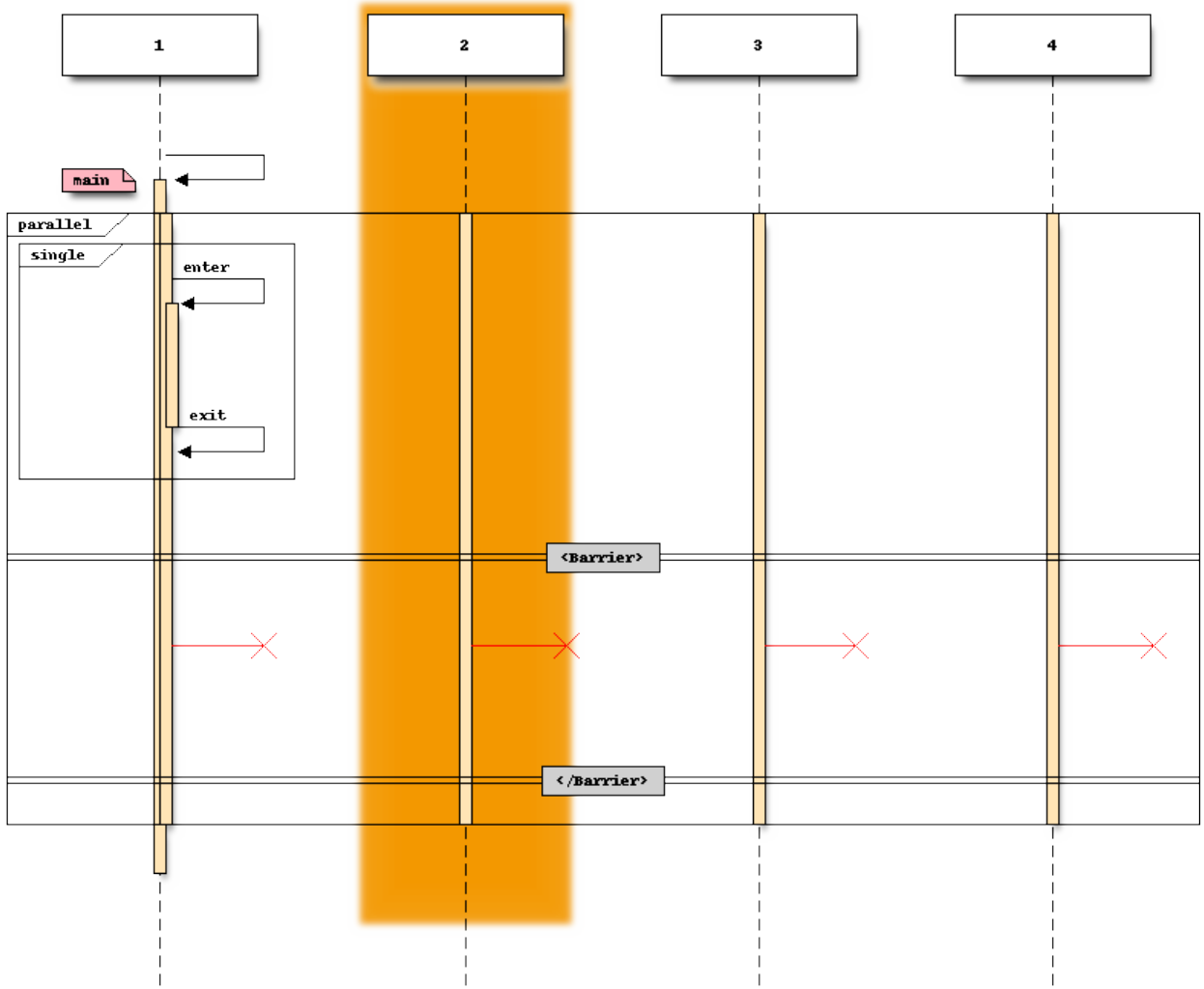
OpenMP/OMPss: mcGDB sequence diagram

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



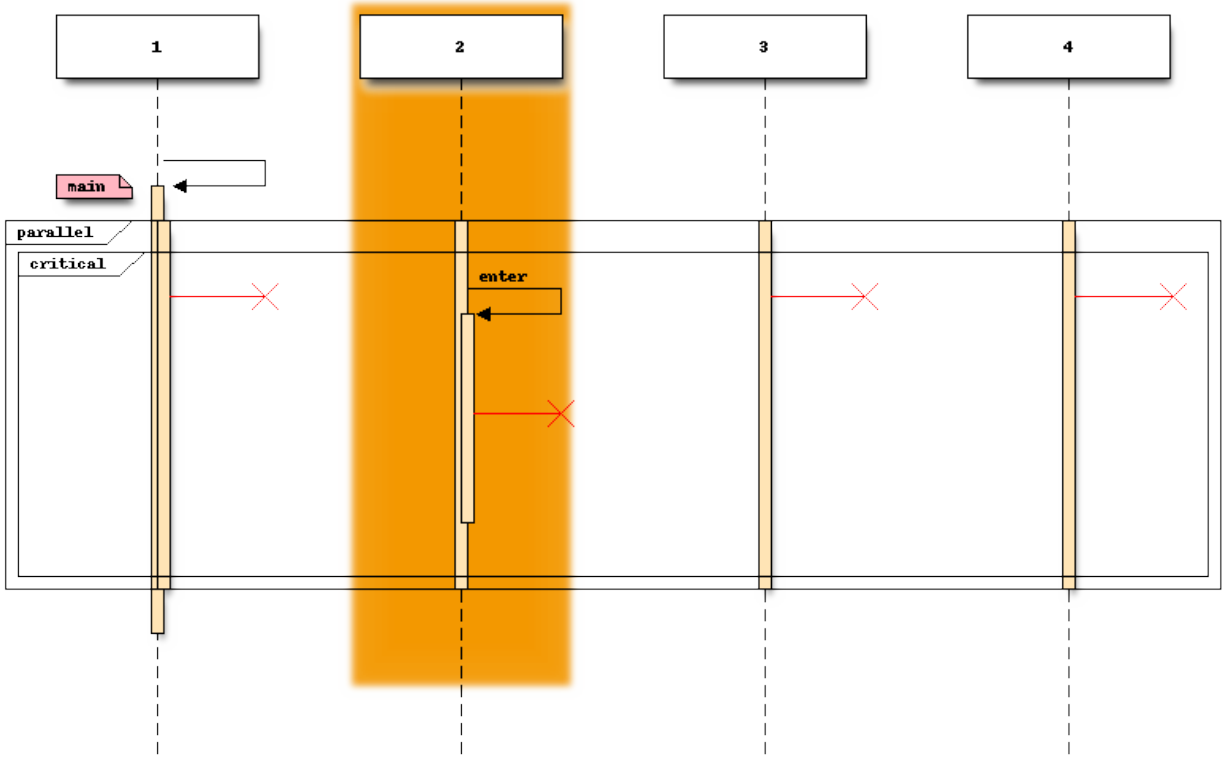
OpenMP/OMPss: mcGDB sequence diagram

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



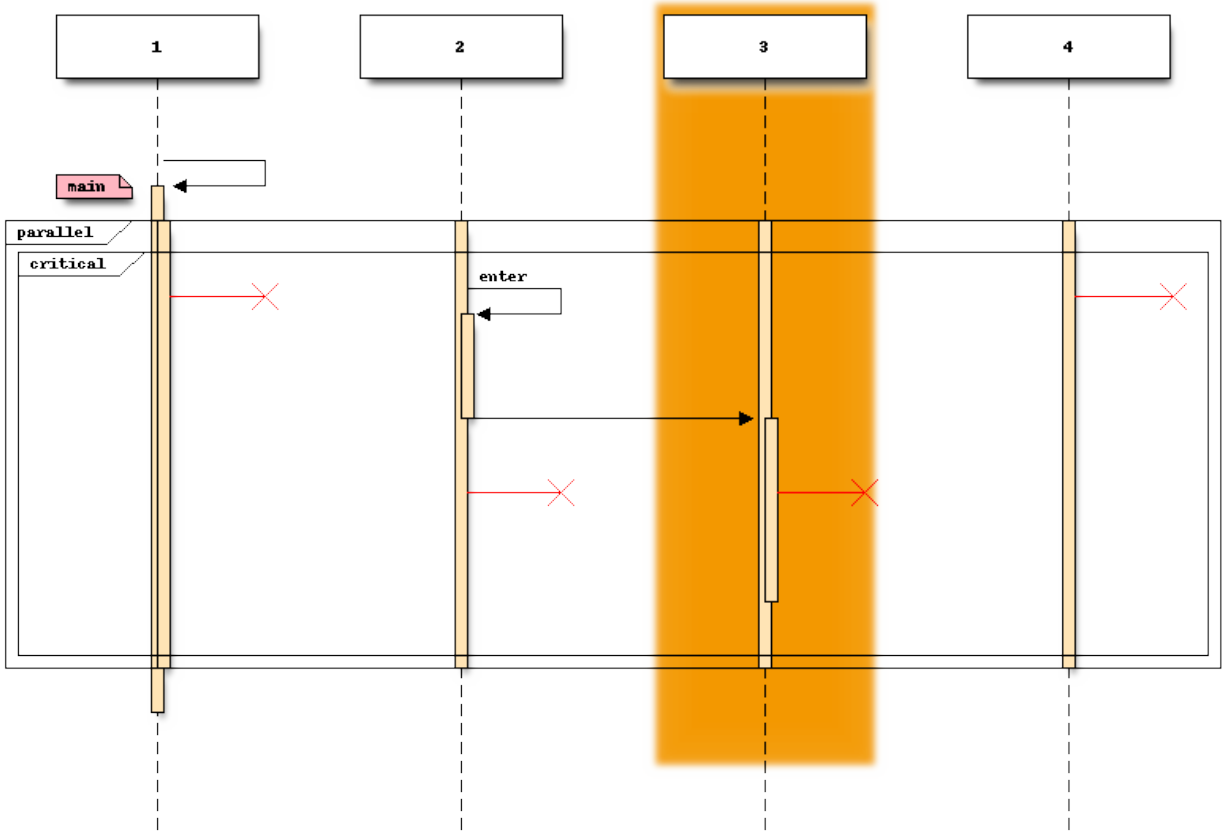
OpenMP/OMPss: mcGDB sequence diagram

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



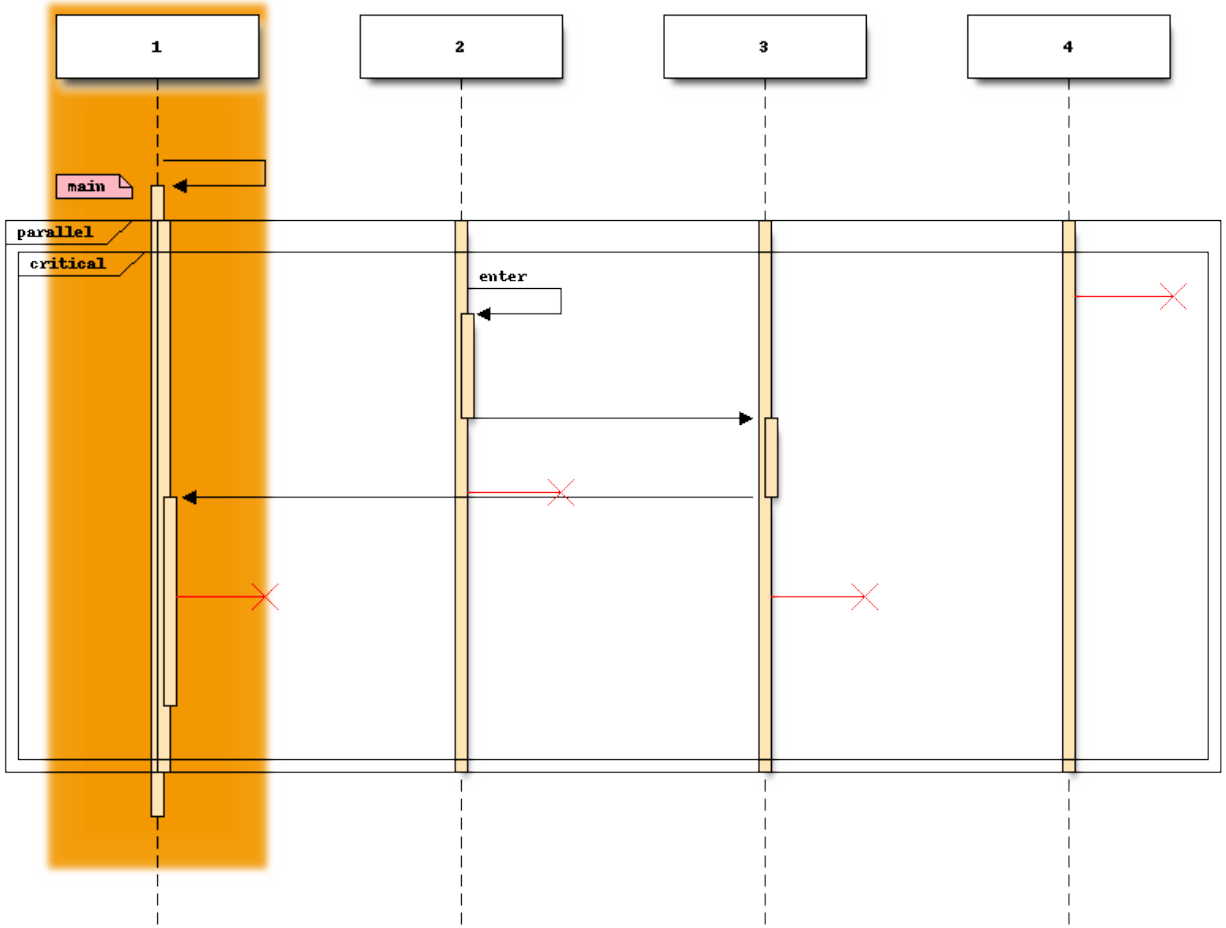
OpenMP/OMPss: mcGDB sequence diagram

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

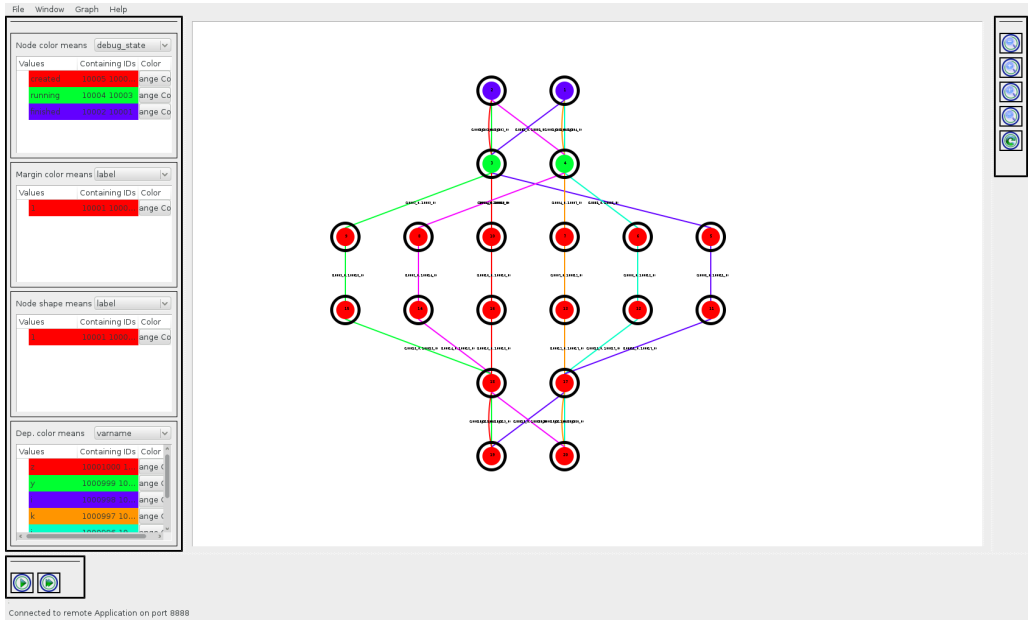
... provide a structural representation
... provide details about entity state

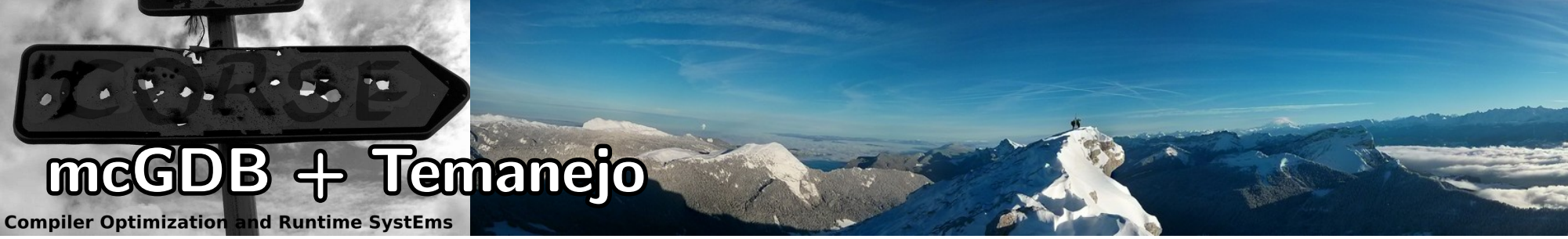
1. **fork-join** $\Longrightarrow$ OpenMP sequence diagrams
2. **task-based** $\Longrightarrow$ mcGDB+Temanejo cooperation

(HLRS Stuttgart) Temanejo ...

✓ is a great visualization tool for task debugging,

✗ and does not support source-level debugging.





(HLRS Stuttgart) Temanejo ...

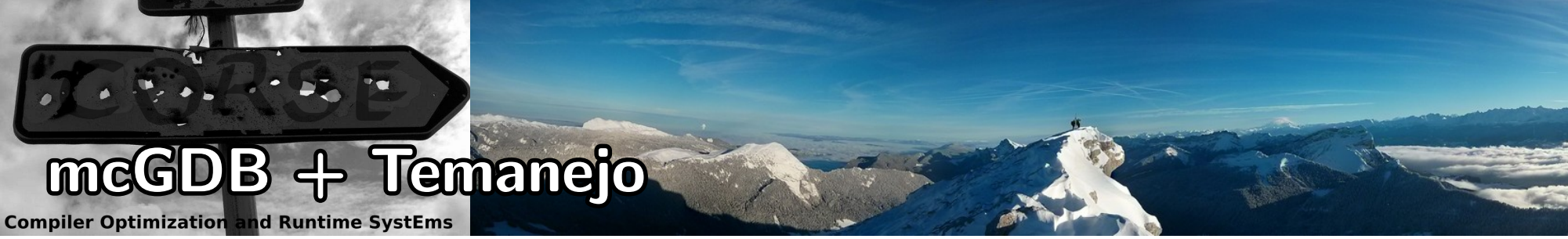
✓ is a great visualization tool for task debugging,

✗ and does not support source-level debugging.

mcGDB ...

✗ has no visualization engine,

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



(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

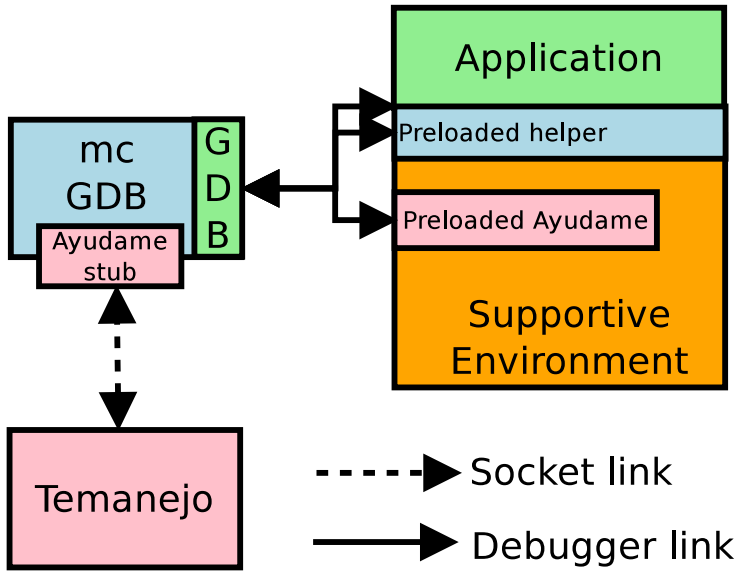
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

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

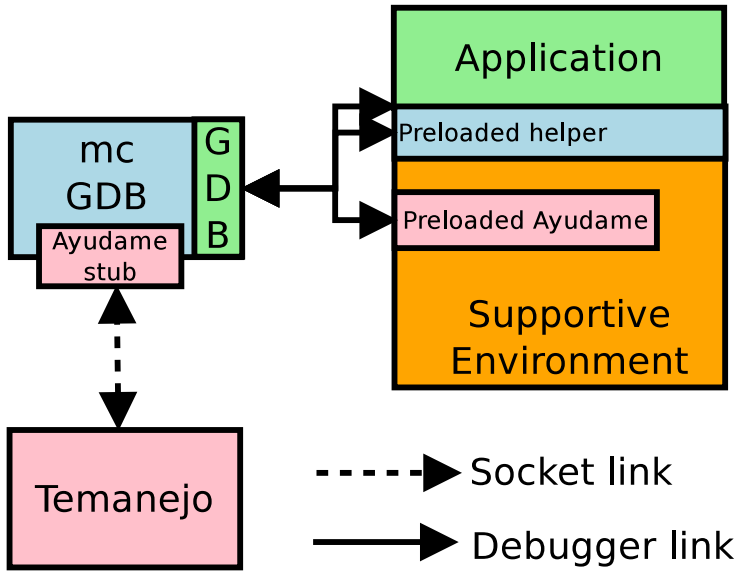
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

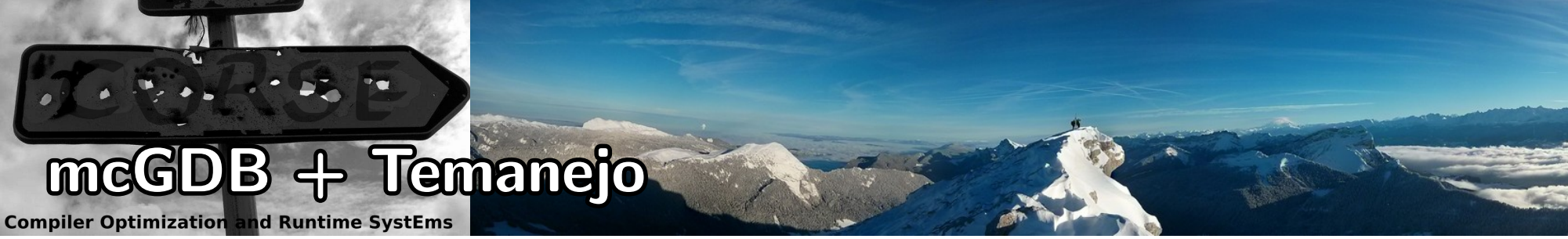
mcGDB

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

GDB

- language and assembly level execution control, memory inspection.





mcGDB – Temanejo cooperation:

Temanejo

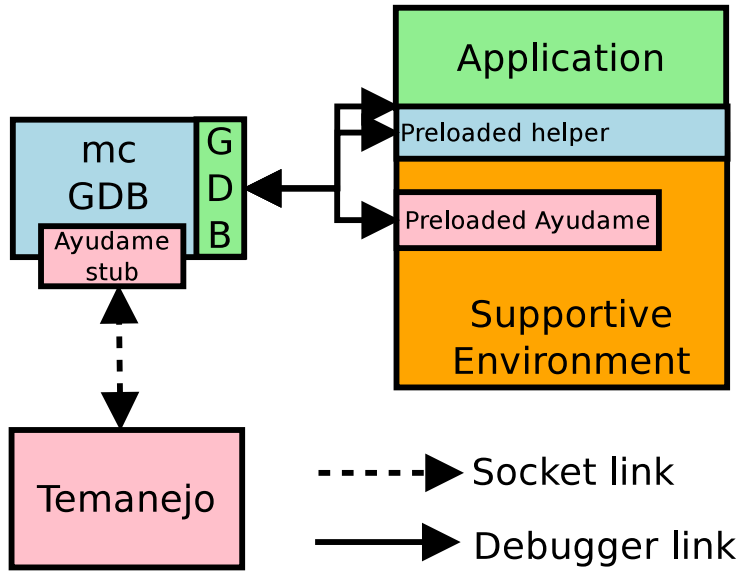
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

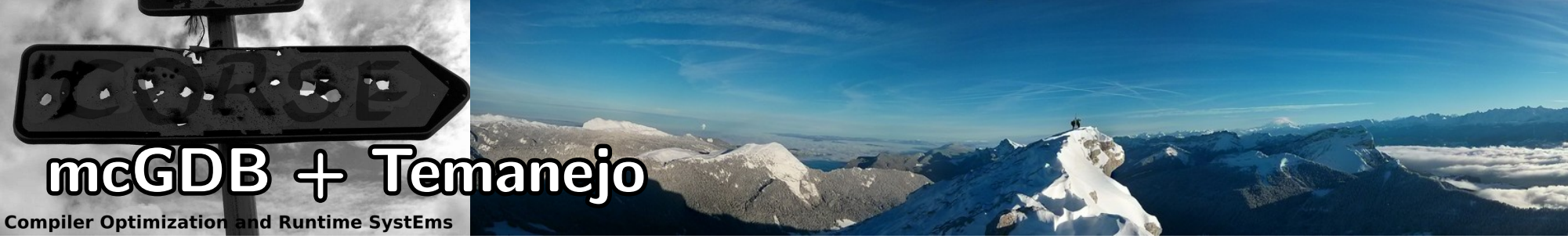
mcGDB

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

GDB

- language and assembly level execution control, memory inspection.





mcGDB – Temanejo cooperation:

Temanejo

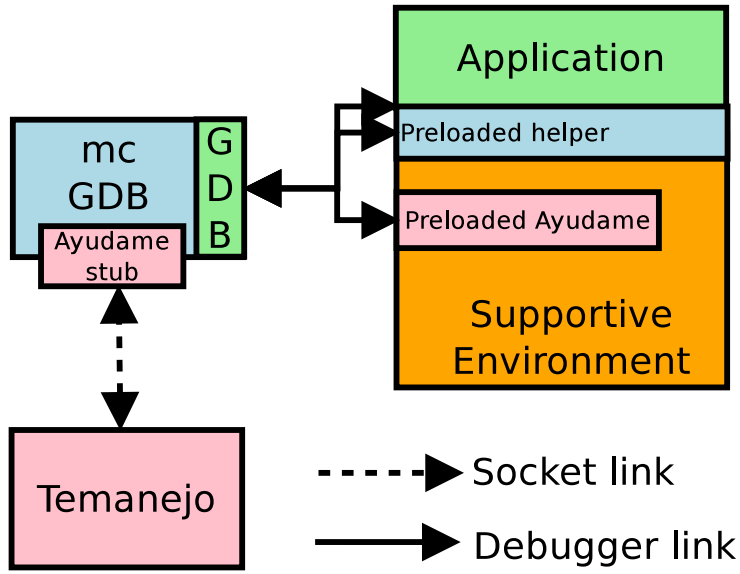
- task graph visualization
- simple model-level execution control.
- sequence diagram visualization.

mcGDB

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

GDB

- language and assembly level execution control, memory inspection.



Conclusion

Compiler Optimization and Runtime SystEms

- Debugging high-level applications is challenging
- Lack of information about programming models and frameworks

Our contribution: model-centric interactive debugging

- mcGDB extends GDB through its Python interface:
  - Framework for model-centric debugging
  - Interface patches contributed to the community
  - Source code soon-to-be open source (Apache Licence)
- mcGDB OpenMP support:
  - Developed for GNU GOMP and Intel OpenMP, OMPss partially
  - Better control of fork-join and task-based execution
  - Better current-state understanding
    - ★ OpenMP sequence diagrams
    - ★ Temanejo graph visualization

Conclusion

Compiler Optimization and Runtime SystEms

- Debugging high-level applications is challenging
- Lack of information about programming models and frameworks

Our contribution: model-centric interactive debugging

- mcGDB extends GDB through its Python interface:
  - Framework for model-centric debugging
  - Interface patches contributed to the community
  - Source code soon-to-be open source (Apache Licence)
- mcGDB OpenMP support:
  - Developed for GNU GOMP and Intel OpenMP, OMPss partially
  - Better control of fork-join and task-based execution
  - Better current-state understanding
    - ⋆ OpenMP sequence diagrams
    - ⋆ Temanejo graph visualization

Conclusion

Compiler Optimization and Runtime SystEms

- Debugging high-level applications is challenging
- Lack of information about programming models and frameworks

Our contribution: model-centric interactive debugging

- mcGDB extends GDB through its Python interface:
  - Framework for model-centric debugging
  - Interface patches contributed to the community
  - Source code soon-to-be open source (Apache Licence)
- mcGDB OpenMP support:
  - Developed for GNU GOMP and Intel OpenMP, OMPss partially
  - Better control of fork-join and task-based execution
  - Better current-state understanding
    - ★ OpenMP sequence diagrams
    - ★ Temanejo graph visualization

Programming-Model Centric Debugging for OpenMP/OMPss

Kevin Pouget
Jean-François Méhaut, Miguel Santana

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

4^{th} JLESC Workshop, Bonn, Germany
December 3^{rd}, 2015