inventors for the digital world

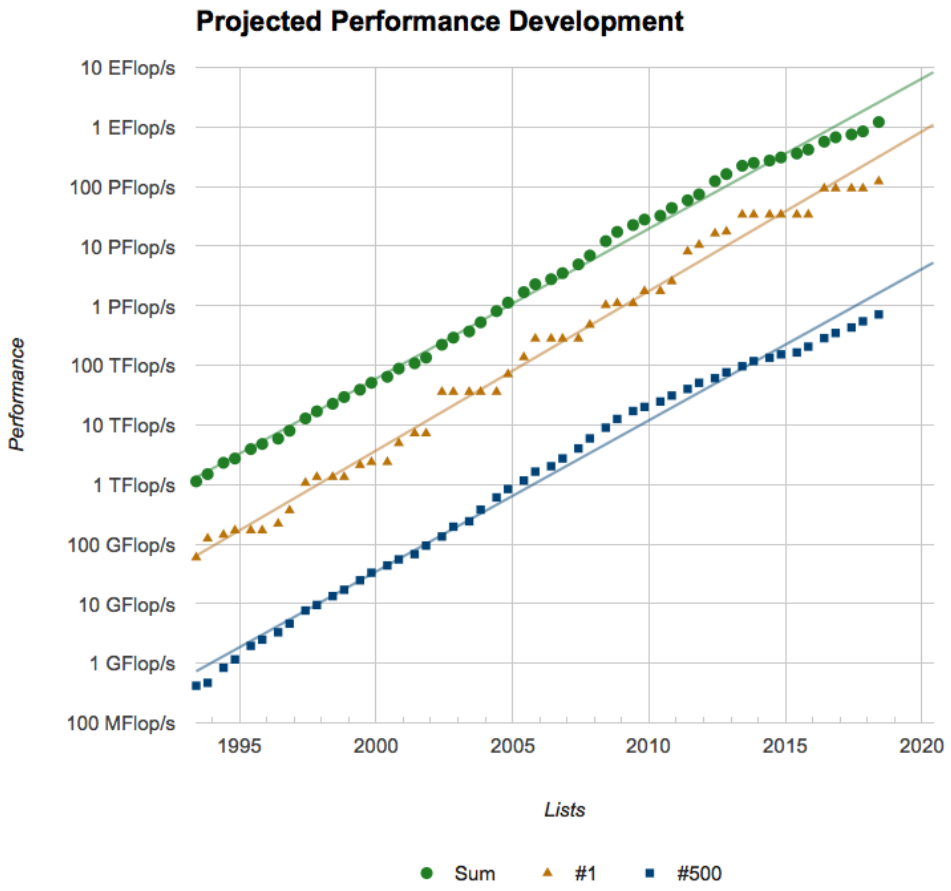
PARCOACH Extension for a Full-Interprocedural Collectives Verification

Pierre Huchant[1,2], **Emmanuelle Saillard**[2], Denis Barthou[1,2], Hugo Brunie[1,2,3] and Patrick Carribault[3]

1. Univ. of Bordeaux, Bordeaux INP - 2. Inria Bordeaux - 3. CEA, DAM, DIF, F-91297 Arpajon, France

Correctness 2018
Dallas, Texas

Applications/Supercomputers Evolution



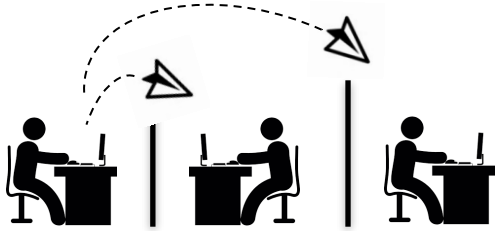
- More complex applications (i.e., combination of parallel programming models)
- Machines more complex, heterogeneous architectures
- Exascale systems targeted in 2020

How can we help developers having correct parallel applications?

➡ Precision ➡ Scalability ➡ Soundness ➡ Heterogeneity

Parallel Programming Models

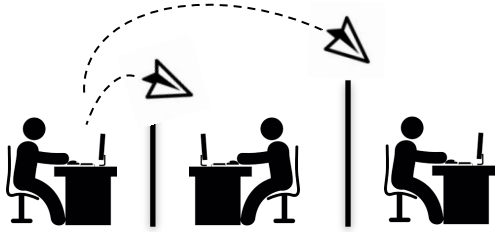
MPI



OpenMP

Collectives

MPI



Blocking or non-blocking communication involving all MPI processes of the same communicator

`MPI_Barrier, MPI_Ibarrier, MPI_Bcast,…`

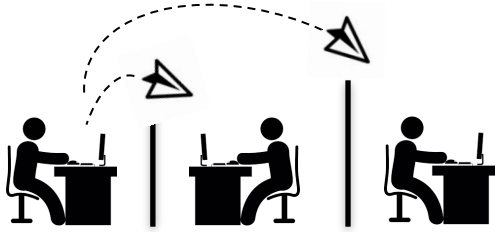
OpenMP

Barrier and worksharing construct

`#pragma omp barrier/single/for/ sections/workshare`

Collective Errors

MPI



Blocking or non-blocking communication involving all MPI processes of the same communicator

`MPI_Barrier, MPI_Ibarrier, MPI_Bcast,…`

OpenMP

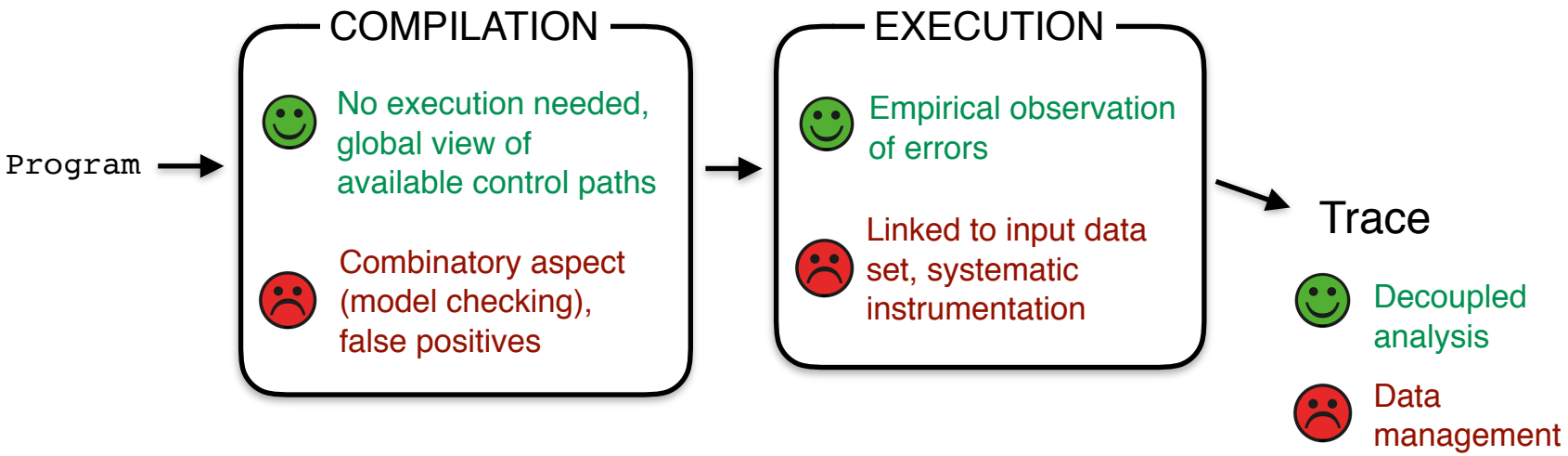
Barrier and worksharing construct

`#pragma omp barrier/single/for/ sections/workshare`

MPI/OpenMP specification : All MPI processes / OpenMP threads must have the same sequence of collectives

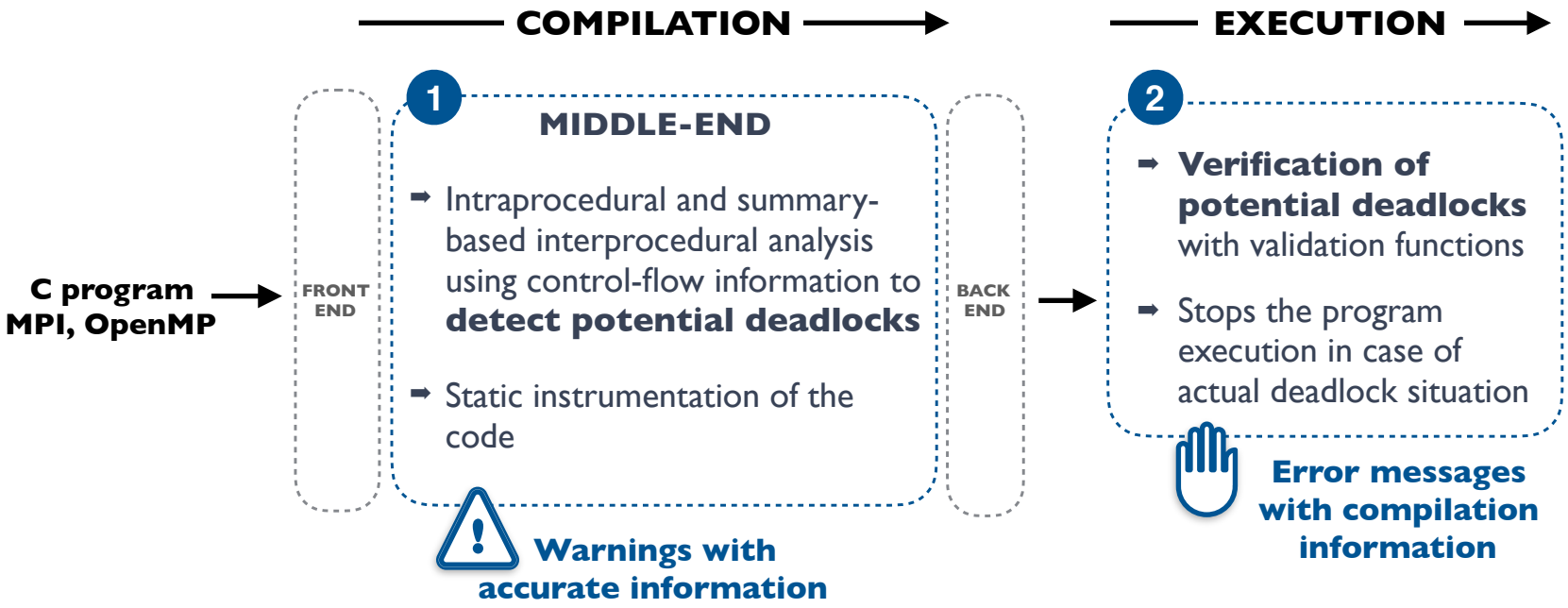
Goal : Detect collective errors (i.e., collective mismatch)

Collective Errors Detection

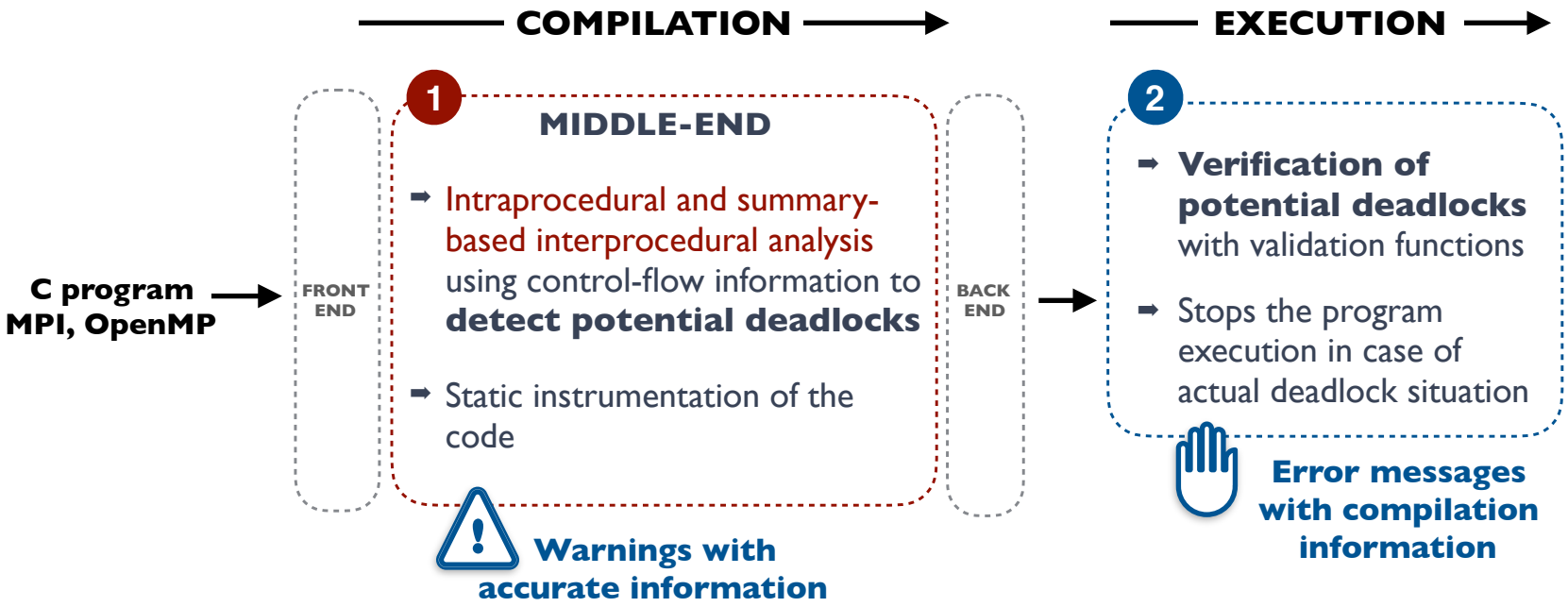


- **Static tools**
  - > MPI-SPIN, TASS, OAT, Zhang *et al.*, GCC, ICC, LLVM
- **Dynamic tools**
  - > MUST, DAMPI, MPI profiling library (MPICH2, MPI/SX)
- **Trace-based dynamic tools**
  - > IMC, Intel Inspector XE, Sun thread analyzer

PARallel COntrol flow Anomaly CHecker



PARallel COntrol flow Anomaly CHecker



Contributions

- ➡ More accurate interprocedural analysis
- ➡ Full integration into the LLVM compiler

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

MPI Motivating Example

```
void c( ) {

  if( )
     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

Deadlock

The machine state does not help to detect the source of the deadlock

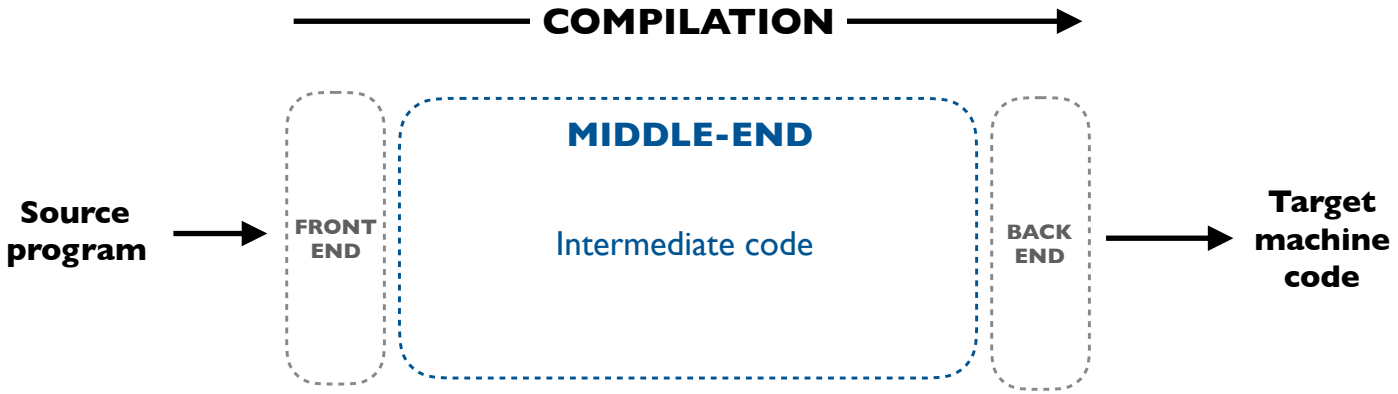
MPI Motivating Example

```
void c( ) {

  if( )
      MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/
  MPI_Finalize();
}
```

Deadlock

The machine state does not help to detect the source of the deadlock

Structure of a compiler



Intermediate representation

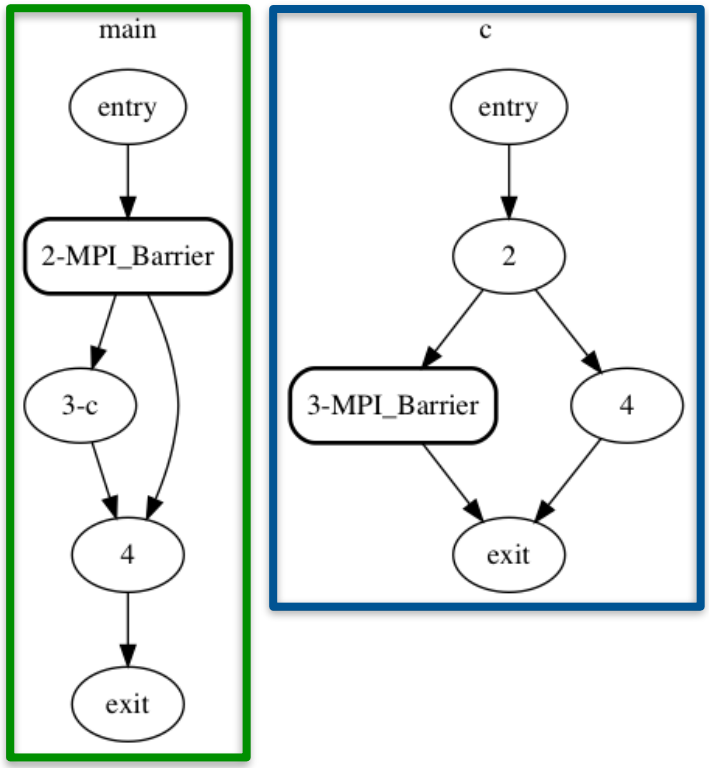
- Programming language independent
- Target machine independent

Control Flow Graph (CFG)

- Models all program executions
- Right representation to study instruction flow

Control Flow Graph

```
void c( ) {

  if( )                          2
     MPI_Barrier(com2);          3
  else
    /*…*/                        4
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);             2

  if( )
    c( );                        3
  /*…*/
  MPI_Finalize();                4
}
```

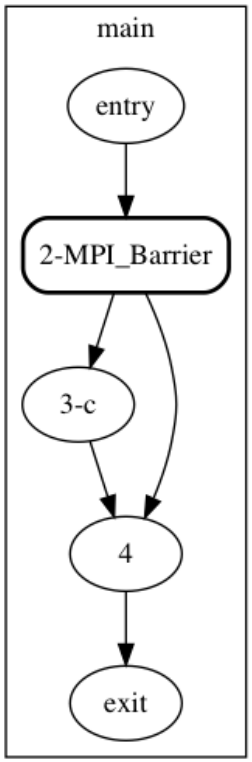


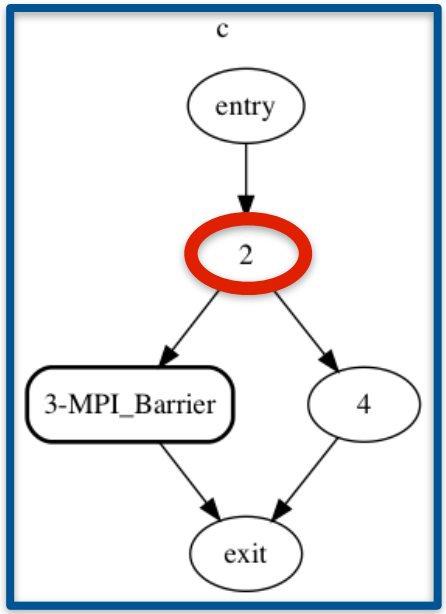
Iterated post dominance frontier (PDF+)

U : Set of nodes in PPCFG

PDF+(U) : Set of nodes that can lead both to a node in U or not

PDF+({3_c}) = {2_c}





Summary of function c: ∅

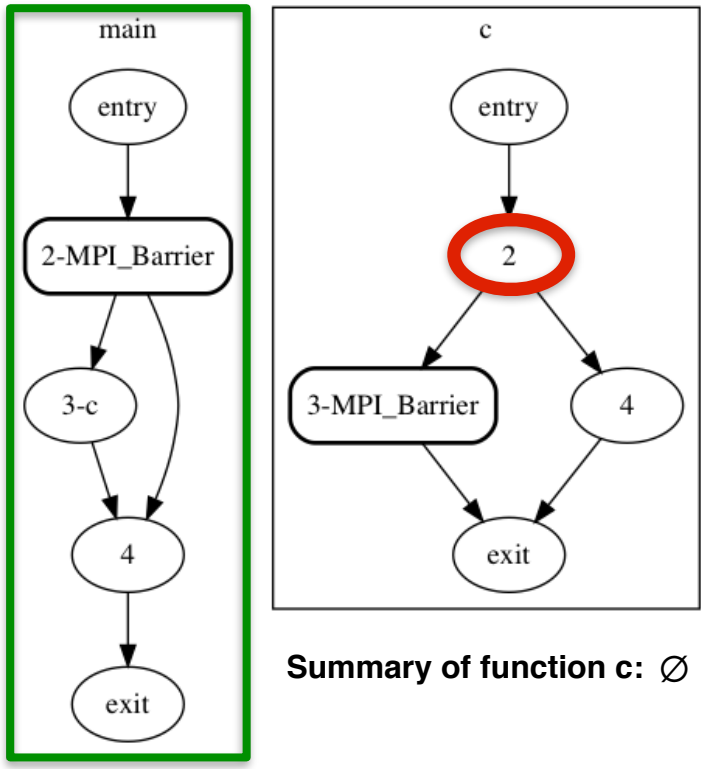
Iterated post dominance frontier (PDF+)

U : Set of nodes in PPCFG

PDF+(U) : Set of nodes that can lead both to a node in U or not

PDF+({2_{main}}) = ∅

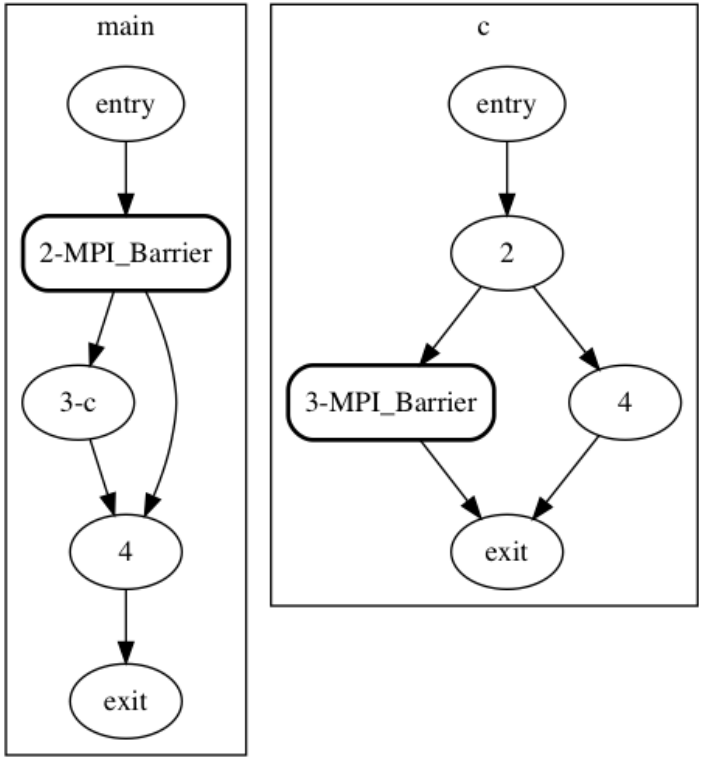
PDF+({3_c}) = {2_c}



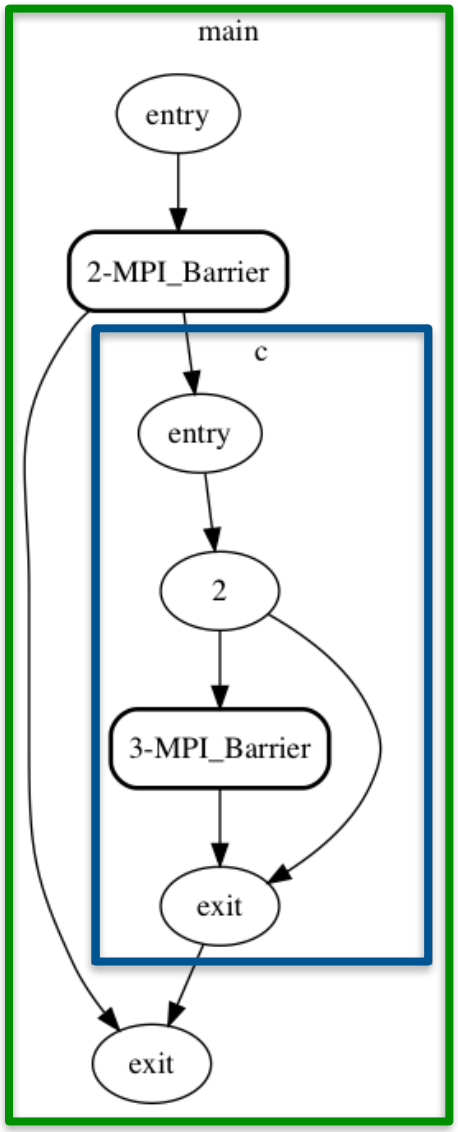
Summary of function c: ∅

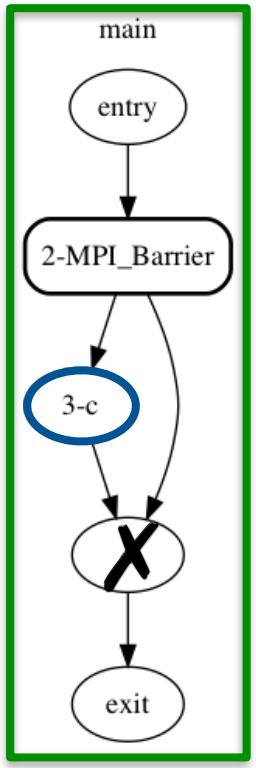
Summary-based interprocedural analysis = Intraprocedural analysis

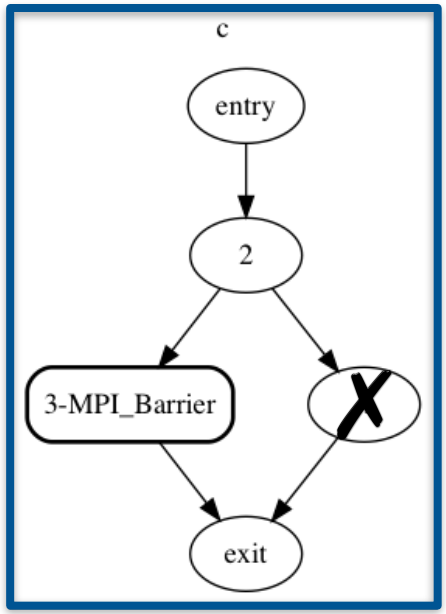
Parallel Program Control Flow Graph

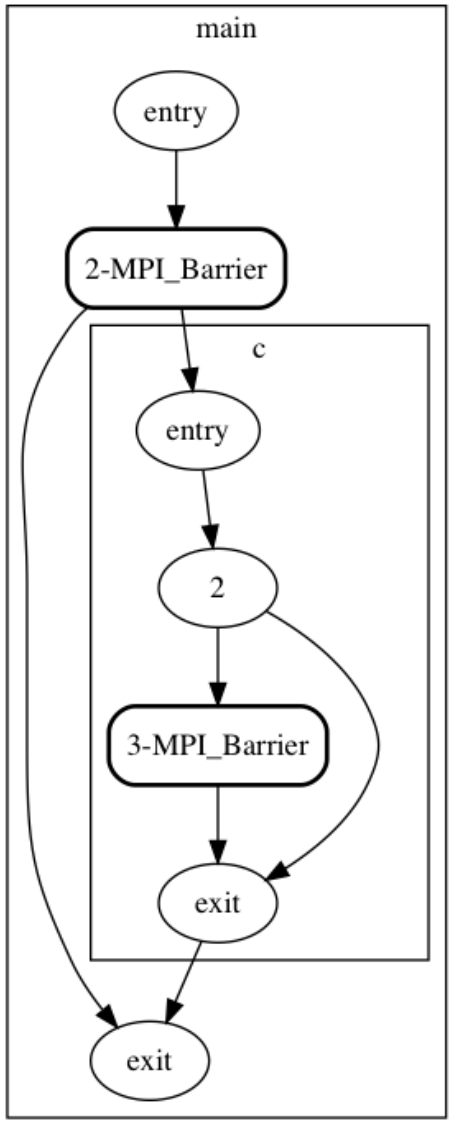


Parallel Program Control Flow Graph





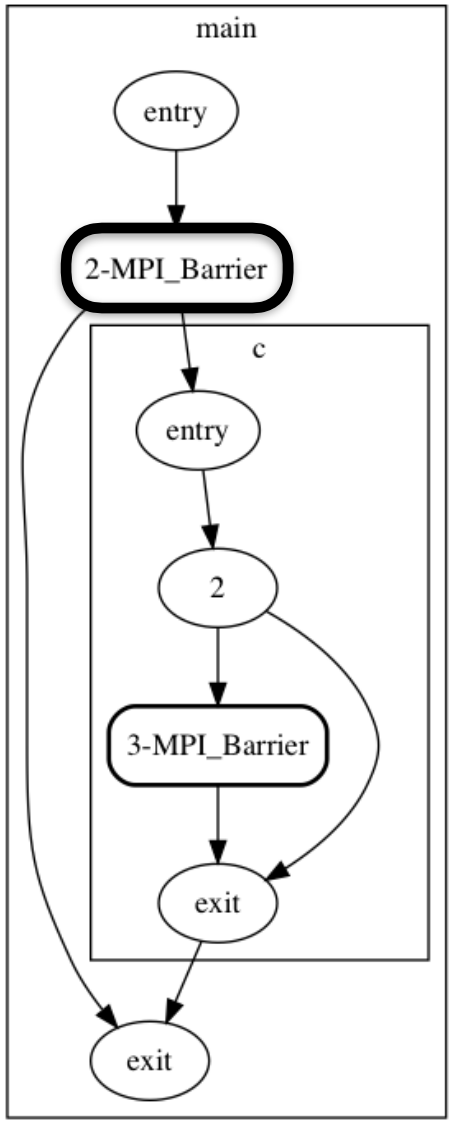




Iterated post dominance frontier (PDF+)

U : Set of nodes in PPCFG

PDF+(U) : Set of nodes that can lead both to a node in U or not



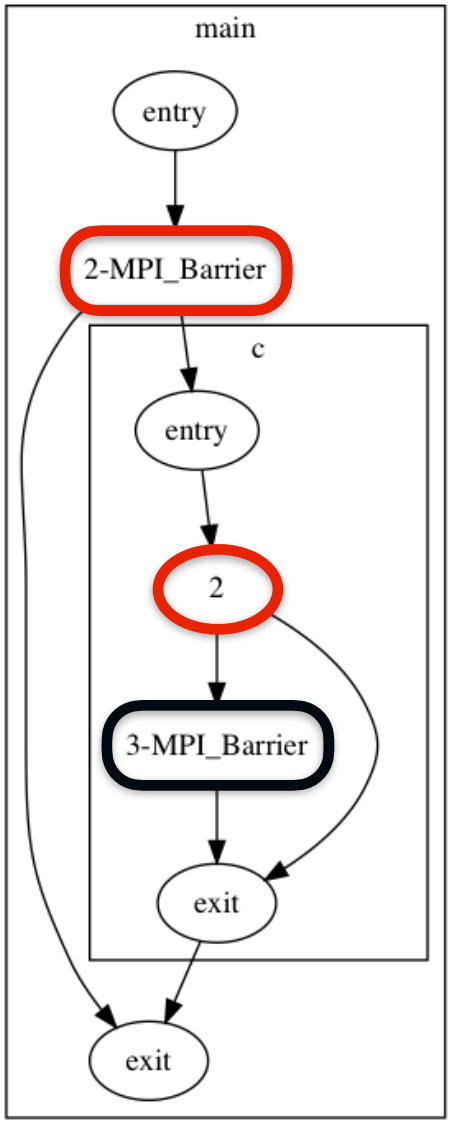
Iterated post dominance frontier (PDF+)

U : Set of nodes in PPCFG

PDF+(U) : Set of nodes that can lead both to a node in U or not

$$PDF^{+}(\{2_{main}\}) = \varnothing$$

Collective Error Detection



Iterated post dominance frontier (PDF^+)

U : Set of nodes in PPCFG

$PDF^+(U)$: Set of nodes that can lead both to a node in U or not

$PDF^+(\{2_{main}\}) = \varnothing$

$PDF^+(\{3_c\}) = \{2_c, 2_{main}\}$

Feedback to users

```
1  void c( ) {
2
3     if( )
4        MPI_Barrier(com2);
5     else
6        /*…*/
7  }
8
9  int main( ) {
10
11    /*…*/
12    MPI_Barrier(com1);
13
14    if( )
15       c( );
16    /*…*/
17    MPI_Finalize();
18 }
```

What a user can read on stderr

```
PARCOACH: warning: MPI_Barrier line 4
possibly not called by all processes
because of conditional(s) line(s) 3, 14
```

Code Instrumentation

```
void c( ) {

  if( )

     MPI_Barrier(com2);
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);

  if( )
    c( );
  /*…*/


  MPI_Finalize();
}
```

- **No warning = no instrumentation**

Code Instrumentation

```
void c( ) {

  if( )
     CC(MPI,com2,i_barrier,O)
     MPI_Barrier(com2)✗
  else
    /*…*/
}

int main( ) {

  /*…*/
  MPI_Barrier(com1);✓

  if( )
    c( );
  /*…*/
  CC(MPI,com2,0∅ )
  MPI_Finalize();
}
```

- **No warning = no instrumentation**
- If at least one warning:
  - > Insert a Check Collective (CC) function **CC(i_{model},com_c,i_c,O)** before each collective c, **starting with the 1st collective that may deadlock**
  - > Insert **CC(i_{model},com,0,∅)** before exit statements and some MPI functions (`MPI_Abort, MPI_Finalize`)

  - ✳ i_{model} = Parallel programming model used
  - ✳ com_c = communicator related to c (0 for OpenMP)
  - ✳ i_c = collective ID
  - ✳ O = collectives that may deadlock (set generated at compile-time)

Experimental Results

- GCC plugin 

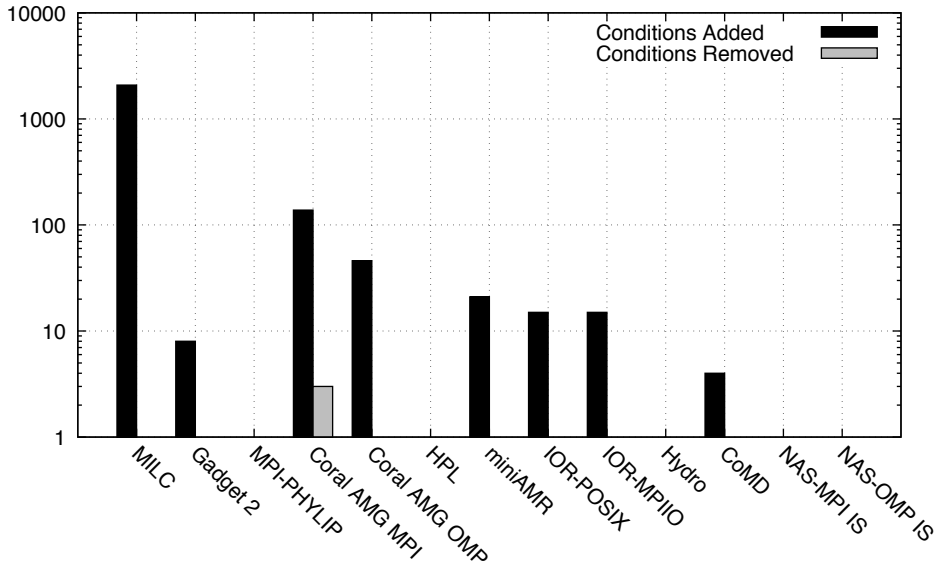
→ LLVM Pass 

- Intraprocedural Analysis VS Full-interprocedural Analysis

- Applications and Benchmarks Statistics

Application	Parallelism	# func.	# coll.	# com.
MILC*	MPI	24,242	635	253
Gadget-2	MPI	193	70	1
MPI-PHYLIP*	MPI	4,000	128	12
Bench. / mini app.	**Parallelism**	**# func.**	**# coll.**	**# com.**
Coral AMG	MPI	1,207	79	19
	OpenMP	1,207	11	-
HPL	MPI	193	3	1
miniAMR	MPI	103	43	2
IOR-POSIX	MPI	175	82	5
IOR-MPIIO	MPI	197	88	5
Hydro	MPI	99	13	1
CoMD	MPI	124	8	1
NAS-MPI IS	MPI	36	9	1
NAS-OMP IS	OpenMP	51	3	-

Experimental Results

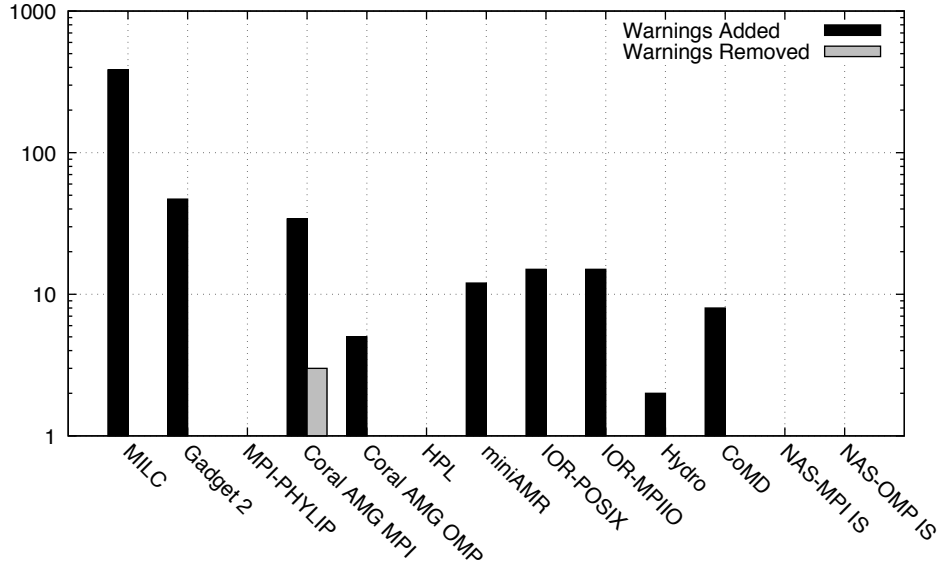


Conditionals

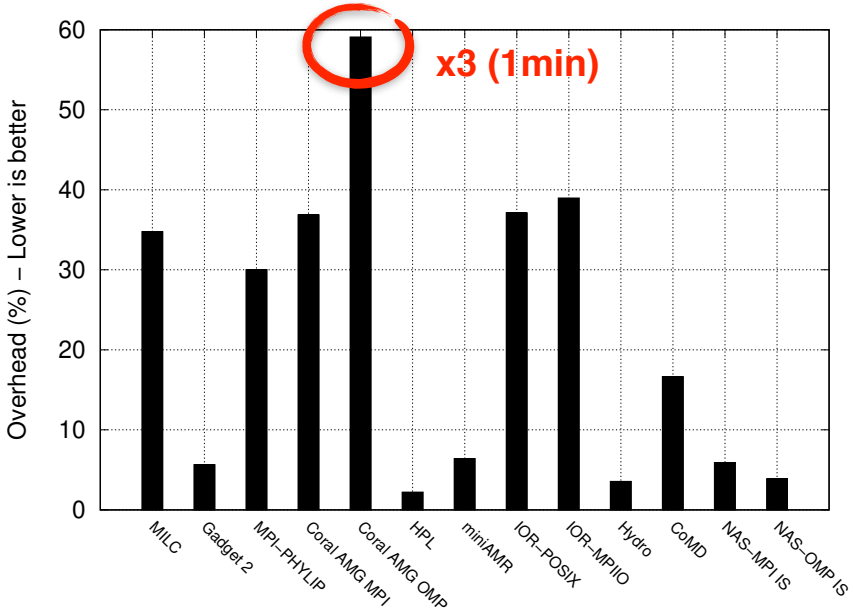
Intra VS Full-inter

Warnings

Intra VS Full-inter



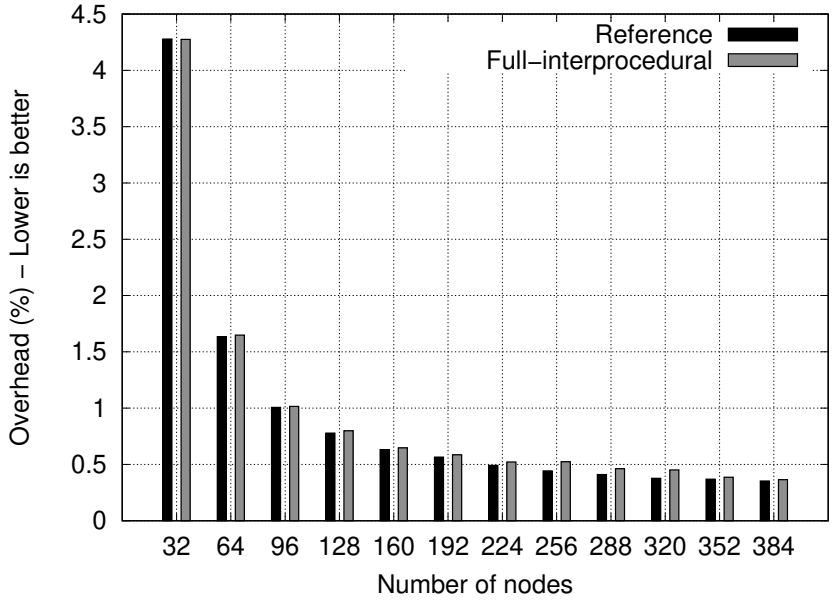
Overhead



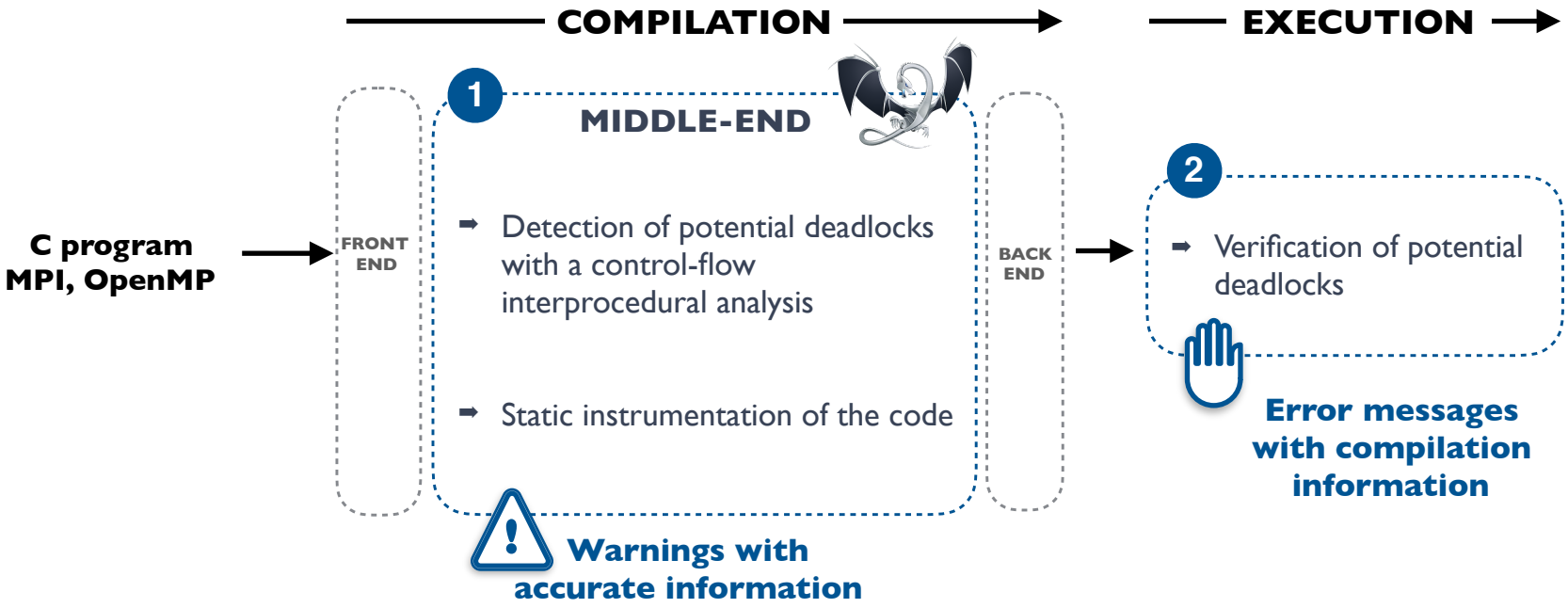
Compile-time overhead

Execution-time overhead

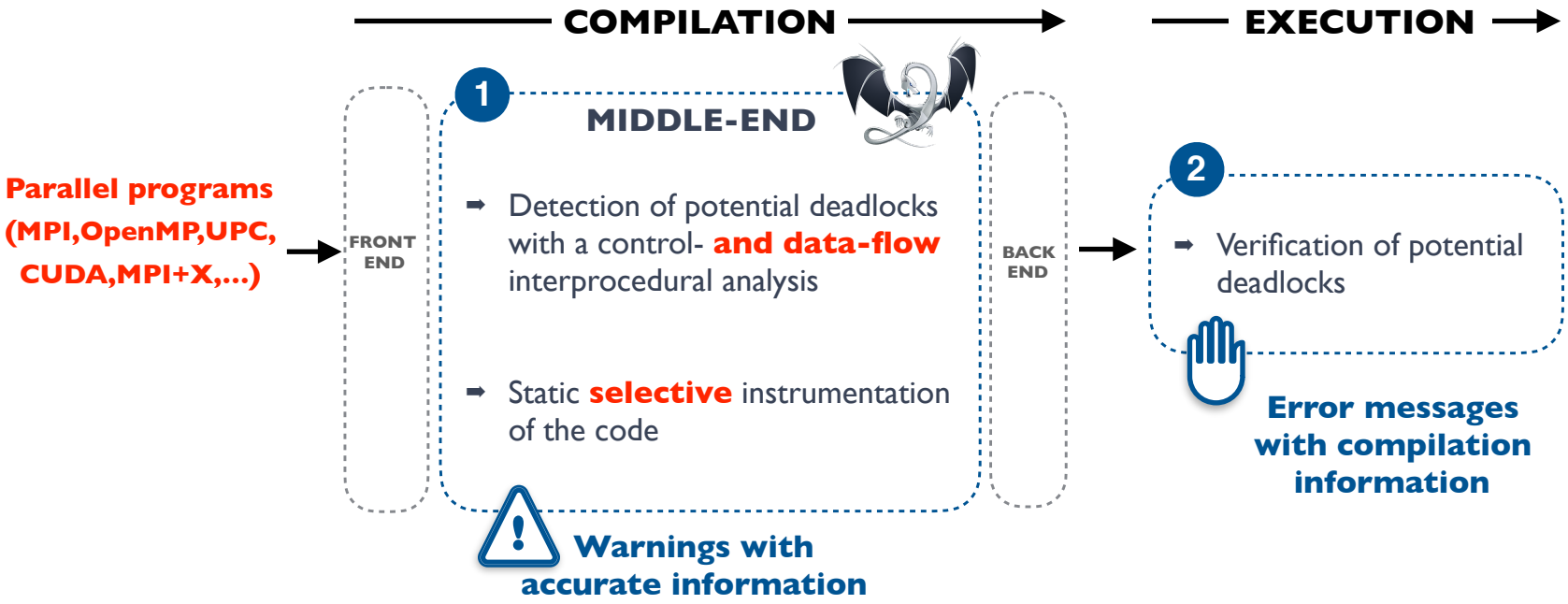
Hydro benchmark



PARallel COntrol flow Anomaly CHecker (PARCOACH)



PARallel COntrol flow Anomaly CHecker (PARCOACH)



➡ Precision ➡ Scalability ➡ Soundness ➡ Heterogeneity

Thank you!

Follow us on www.inria.fr

inventors for the digital world