Projet de compilation avancée

Vérification statique des séquences d'appels aux fonctions collectives MPI

Hugo Lallemant, Mickaël Saes-Vincensini

5 Novembre 2024

Contexte

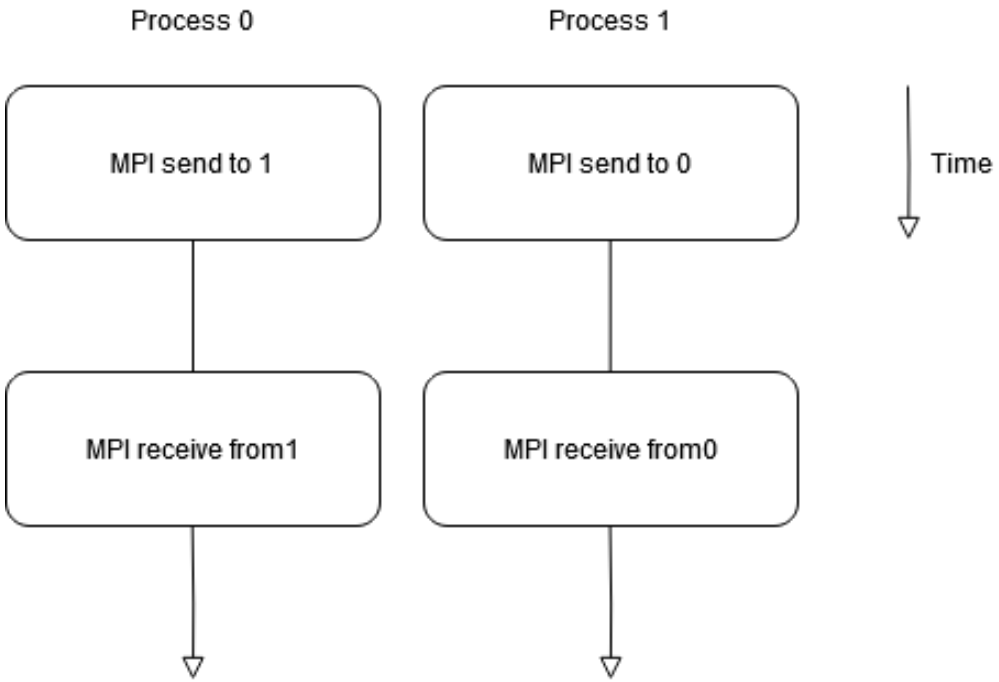
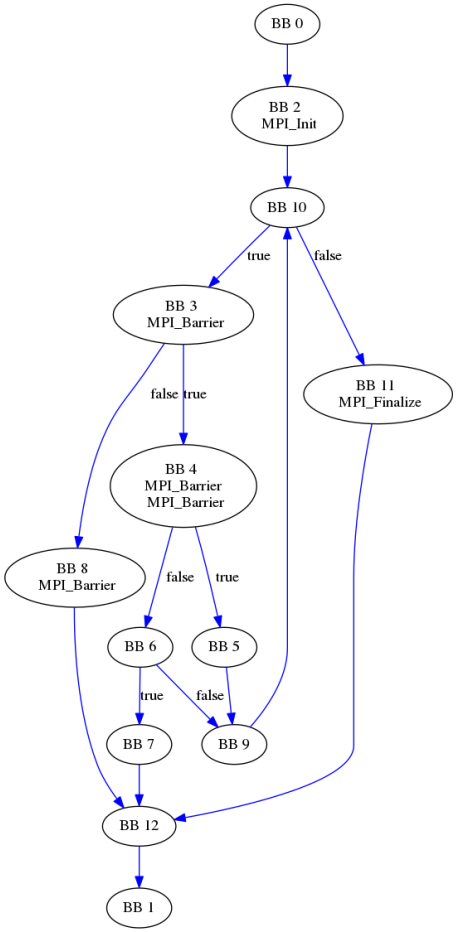


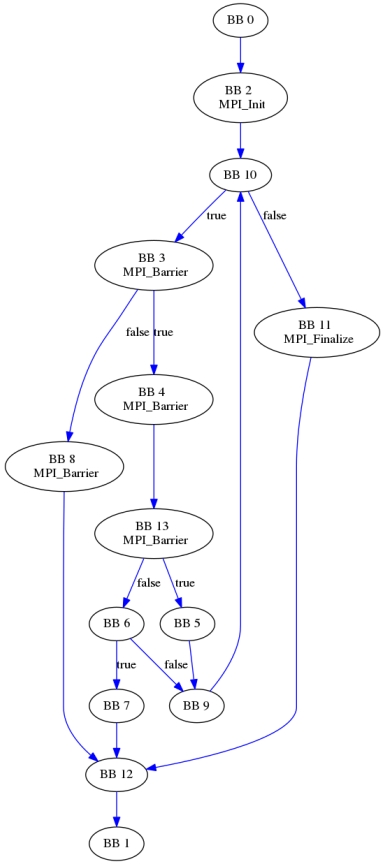
Schéma d'un deadlock dans une exécution de code

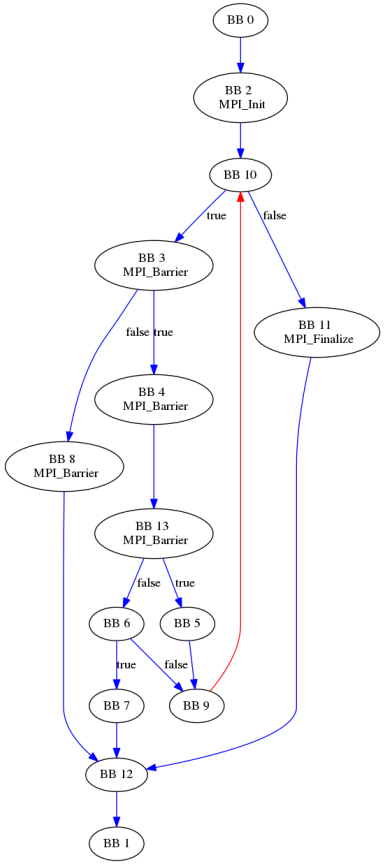
Préparation du CFG




```
typedef struct mpi_ranks
{
        int code;
        int * ranks;
} mpi_ranks;
```

Modification du CFG



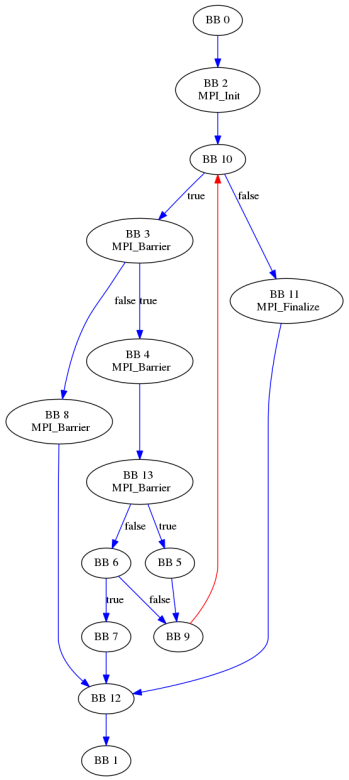


Rang des collectives

$$\text{Rang}(n) = \max_{x \in \text{preds}(n)} \text{Rang}(x)$$

Équation que nous résolvons pour calculer le rang des collectives

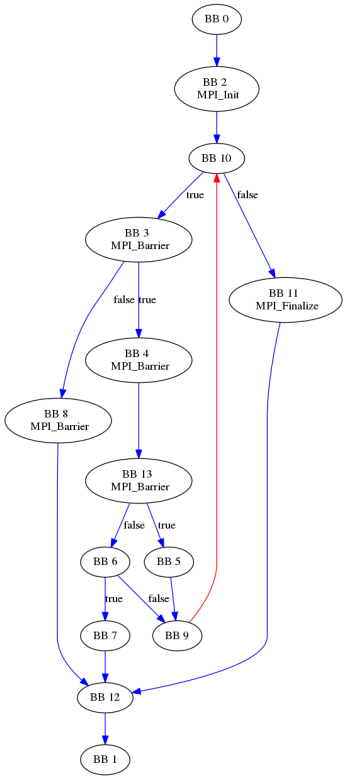
```
index:  0 - collective: 5 - [0, 0, 0, 0, 0, ]
index:  2 - collective: 0 - [1, 0, 0, 0, 0, ]
index:  3 - collective: 4 - [1, 0, 0, 0, 1, ]
index:  4 - collective: 4 - [1, 0, 0, 0, 2, ]
index: 13 - collective: 4 - [1, 0, 0, 0, 3, ]
index:  5 - collective: 5 - [1, 0, 0, 0, 3, ]
index:  6 - collective: 5 - [1, 0, 0, 0, 3, ]
index:  7 - collective: 5 - [1, 0, 0, 0, 3, ]
index:  8 - collective: 4 - [1, 0, 0, 0, 2, ]
index:  9 - collective: 5 - [1, 0, 0, 0, 3, ]
index: 10 - collective: 5 - [1, 0, 0, 0, 0, ]
index: 11 - collective: 1 - [1, 1, 0, 0, 0, ]
index: 12 - collective: 5 - [1, 1, 0, 0, 3, ]
index:  1 - collective: 5 - [1, 1, 0, 0, 3, ]
code: 0, rank: 1 - 2
code: 1, rank: 1 - 11
code: 4, rank: 1 - 3
code: 4, rank: 2 - 4, 8
code: 4, rank: 3 - 13
```



Post dominance des ensembles

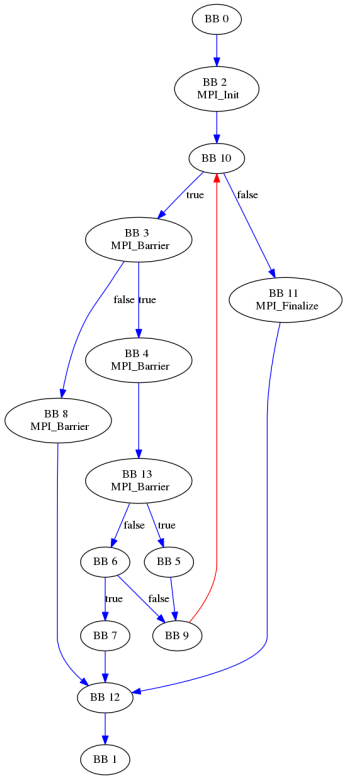
```
---- set postdominated ----
code: 0, rank: 1 - 2
code: 1, rank: 1 - 11
code: 4, rank: 1 - 3
code: 4, rank: 2 - 3, 4, 8
code: 4, rank: 3 - 4, 13
```

```
code: 0, rank: 1 - 2
code: 1, rank: 1 - 11
code: 4, rank: 1 - 3
code: 4, rank: 2 - 4, 8
code: 4, rank: 3 - 13
```



Frontière de post dominance des noeud

```
----------------- post dominance frontier -------------------------
Basic Block: 0 :
Basic Block: 2 :
Basic Block: 3 : 10
Basic Block: 4 : 3
Basic Block: 13 : 3
Basic Block: 5 : 13
Basic Block: 6 : 13
Basic Block: 7 : 6
Basic Block: 8 : 3
Basic Block: 9 : 6, 13
Basic Block: 10 : 6, 13
Basic Block: 11 : 10
Basic Block: 12 :
Basic Block: 1 :
-------------------------------------------------------------------
```



Frontière de post dominance des ensembles

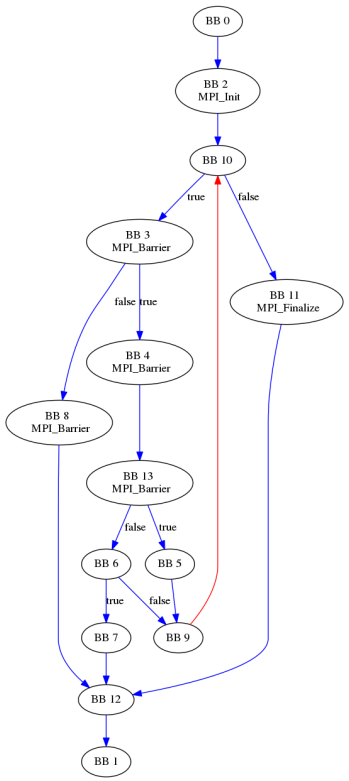
Soit PDF(E) = { X | $\exists S \in \text{Succs}(X)$ tq $E \gg_{sp} S$, $E \not\gg_p X$ }

Soit F = { X | $\exists n \in E$ tq $E \gg_{sp} n$, $X \in \text{PDF}(n)$, $E \not\gg_p X$ }

On montre que E=F

```
---- set frontiers ----
code: 0, rank: 1 -
code: 1, rank: 1 - 10
Potential MPI Deadlock
code: 4, rank: 1 - 10
Potential MPI Deadlock
code: 4, rank: 2 - 10
Potential MPI Deadlock
code: 4, rank: 3 - 3
Potential MPI Deadlock
```

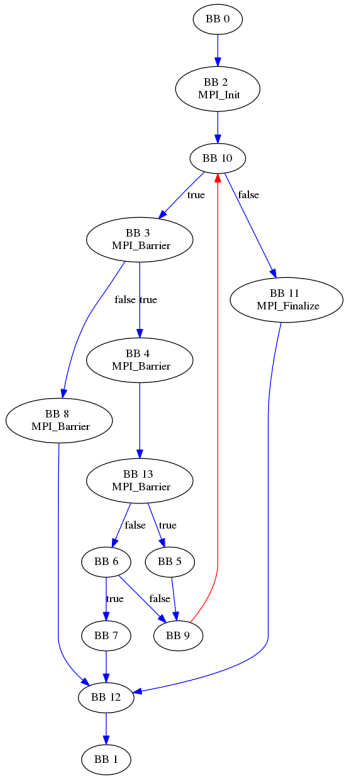
```
code: 0, rank: 1 - 2
code: 1, rank: 1 - 11
code: 4, rank: 1 - 3
code: 4, rank: 2 - 4, 8
code: 4, rank: 3 - 13
```



Frontière itérée

```
---- set iterated frontiers ----
code: 0, rank: 1 -
code: 1, rank: 1 - 3, 6, 10, 13
Potential MPI Deadlock
code: 4, rank: 1 - 3, 6, 10, 13
Potential MPI Deadlock
code: 4, rank: 2 - 3, 6, 10, 13
Potential MPI Deadlock
code: 4, rank: 3 - 3, 6, 10, 13
Potential MPI Deadlock
```

```
code: 0, rank: 1 - 2
code: 1, rank: 1 - 11
code: 4, rank: 1 - 3
code: 4, rank: 2 - 4, 8
code: 4, rank: 3 - 13
```



Affichage d'un warning

```
[mickael.saes-vincensini@hpc01 coav-2024]$ make test6
mkdir -p bin
g++_1220 -I`gcc_1220 -print-file-name=plugin`/include -g -Wall -fno-rtti -shared -fPIC -o bin/libplugin.so src/mpi_plugin.cpp
mpicc tests/test6.c -g -O3 -o bin/test6 -fplugin=./bin/libplugin.so
plugin_init: Entering...
plugin_init: Check ok...
plugin_init: Pass added...
Now starting to examine function mpi_invalid
tests/test6.c: In function 'mpi_invalid':
tests/test6.c:18:9: warning: Potential issue: MPI collective MPI_Barrier in block 6
   18 |         MPI_Barrier(MPI_COMM_WORLD);
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~
tests/test6.c:13:12: warning: Potential issue caused by the following fork in block 2
   13 |         if (c > 5) {
      |            ^
Now starting to examine function mpi_valid
No potential deadlock found.
Now starting to examine function main
tests/test6.c: In function 'main':
tests/test6.c:38:7: warning: Potential issue: MPI collective MPI_Barrier in block 4
   38 |       MPI_Barrier(MPI_COMM_WORLD);
      |       ^~~~~~~~~~~~~~~~~~~~~~~~~~~
tests/test6.c:35:6: warning: Potential issue caused by the following fork in block 2
   35 |   if (c < 20) {
      |      ^
tests/test6.c:36:8: warning: Potential issue caused by the following fork in block 3
   36 |     if (c < 10)
      |        ^
tests/test6.c:40:5: warning: Potential issue: MPI collective MPI_Barrier in block 5
   40 |     MPI_Barrier(MPI_COMM_WORLD);
      |     ^~~~~~~~~~~~~~~~~~~~~~~~~~~
tests/test6.c:35:6: warning: Potential issue caused by the following fork in block 2
   35 |   if (c < 20) {
      |      ^
```

Figure – Affichage généré par le warning

Gestion des pragma

```
#pragma ProjetCA mpicoll_check f
#pragma ProjetCA mpicoll_check (f1, f2, ..., fn)
```

Conclusion et perspectives

```
unsigned int execute (function *fun)
{
    cfgviz_dump(fun, "initial");
    prepare_cfg(fun);
    cfgviz_dump(fun, "split");
    calculate_dominance_info(CDI_POST_DOMINATORS);
    bitmap_head *frontiers = post_dominance_frontiers(fun);
    bitmap_head *invalid_edges = cfg_prime(fun);
    cfgviz_dump(fun, "invalid_edges", invalid_edges);
    calculate_rank(fun, invalid_edges);
    bitmap_head **sets = collective_rank_set(fun);
    bitmap_head **set_postdominated = set_post_dominance(fun, sets);
    bitmap_head **set_frontiers = set_post_dominance_frontiers(fun, sets,
                                                set_postdominated, frontiers);
    bitmap_head **it_frontier = iterated_post_dominance_frontiers(fun,
                                                set_frontiers, frontiers);
    bool warnings = print_warnings(fun, it_frontier, sets);
    if (!warnings) printf("No potential deadlock found.\n");
    clean_aux_field(fun, 0);
    free_dominance_info(CDI_POST_DOMINATORS);
    return 0;
}
```

Projet d'analyse statique des fonctions pour détecter les deadlocks dans les collectives `MPI`

Preuve de la slide 7 partie 1

Soit E un ensemble de nœuds

Soit PDF(E) $= \{ X \mid \exists S \in \text{Succs}(X) \text{ tq } E \gg_{sp} S,\ E \not\gg_p X \}$

Soit F $= \{ X \mid \exists n \in E \text{ tq } E \gg_{sp} n,\ X \in \text{PDF}(n),\ E \not\gg_p X \}$

On va montrer que PDF(E) = F

Soit $x_1 \in F$

$\Leftrightarrow \exists n \in E,\ x_1 \in \text{PDF}(n),\ E \gg_p n,\ E \not\gg_p x_1$

$\Leftrightarrow \exists n \in E,\ \exists S \in \text{Succ}(x_1),\ n \gg_{sp} S,\ n \not\gg_p x,\ E \gg_p n,\ E \not\gg_p x_1$

$\Rightarrow \exists S \in \text{Succ}(x_1),\ E \gg_p n \gg_p S,\ E \not\gg_p x_1$

$\Rightarrow x_1 \in \text{PDF}(E) \Rightarrow F \subseteq \text{PDF}(E)$

Preuve de la slide 7 partie 2

Soit E un ensemble de nœuds

Soit PDF(E) $= \{ X \mid \exists S \in \text{Succs}(X) \text{ tq } E \gg_{sp} S,\ E \not\gg_{p} X\}$

Soit F $= \{ X \mid \exists n \in E \text{ tq } E \gg_{sp} n,\ X \in \text{PDF}(n),\ E \not\gg_{p} X\}$

On va montrer que PDF(E) = F

Soit $x_2 \in \text{PDF}(E)$

$\Leftrightarrow \exists S \in \text{Succ}(x_2),\ E \gg_{sp} S,\ E \not\gg_{p} x_2$

$\Rightarrow S \not\gg_{p} x_2$, car $E \gg_{sp} S$ et $E \not\gg_{p} x_2$

donc $x \in \text{PDF}(S)$, car $S \in \text{Succ}(x_2), S \gg_{sp} S$, et $S \not\gg_{p} x_2$

donc $x_2 \in F \Longrightarrow \text{PDF}(E) \subseteq F$

Code lié au warning

```c
#pragma Projet_CA mpicoll_check (main, mpi_invalid, mpi_valid)
void mpi_not_checked() {
        MPI_Barrier(MPI_COMM_WORLD);
}
void mpi_invalid(int c) {
        if (c > 5) {
                MPI_Barrier(MPI_COMM_WORLD);
                return;
        } else MPI_Barrier(MPI_COMM_WORLD);

        MPI_Barrier(MPI_COMM_WORLD);
        return;
}
```

```c
void mpi_valid(int c) {
        if (c > 5) {
                MPI_Barrier(MPI_COMM_WORLD);
        } else MPI_Barrier(MPI_COMM_WORLD);

        MPI_Barrier(MPI_COMM_WORLD);
        return;
}
int main(int argc, char * argv[])
{
  MPI_Init(&argc, &argv);
  int c = 5;
  if (c < 20) {
    if (c < 10)
    {
      MPI_Barrier(MPI_COMM_WORLD);
    }
    MPI_Barrier(MPI_COMM_WORLD);
  }
  MPI_Finalize();
  return 1;
}
```