DE LA RECHERCHE À L'INDUSTRIE

Correctness Analysis of MPI-3 Non-Blocking Communications in PARCOACH

Julien JAEGER[1], Emmanuelle SAILLARD[1], Patrick CARRIBAULT[1] and Denis BARTHOU[2]

[1]CEA, DAM, DIF, F-91297 Arpajon, France *[2]Bordeaux Institute of Technology /LaBRI INRIA, Bordeaux, France*

INTRO

1 GOAL: provide a lightweight static analysis for correct sequence of MPI 3 non-blocking communications

MPI standard v.3.0 provides non-blocking collectives
- But strict constraints hinder the mixed use of all flavor of collectives
- They add up to the rules to correctly use non-blocking communications

We improve the PARCOACH [1] tool with two new features:
- A **new analysis** checking the presence of completion calls for all non-blocking communications
- The **improvement of the original analysis** of PARCOACH to include non-blocking collectives

2 PARCOACH = PARallel COntrol flow Anomaly CHecker

PARCOACH is designed:
- to be compatible with existing dynamic tools like MUST[2]
- to found MPI collectives errors in MPI programs

PARCOACH = GCC plugin which gathers:
- a **static analysis** that identifies MPI collective operations that can deadlock
- a **static instrumentation** for exectution time verification
- **precise messages** with source location

PARCOACH ANALYSIS

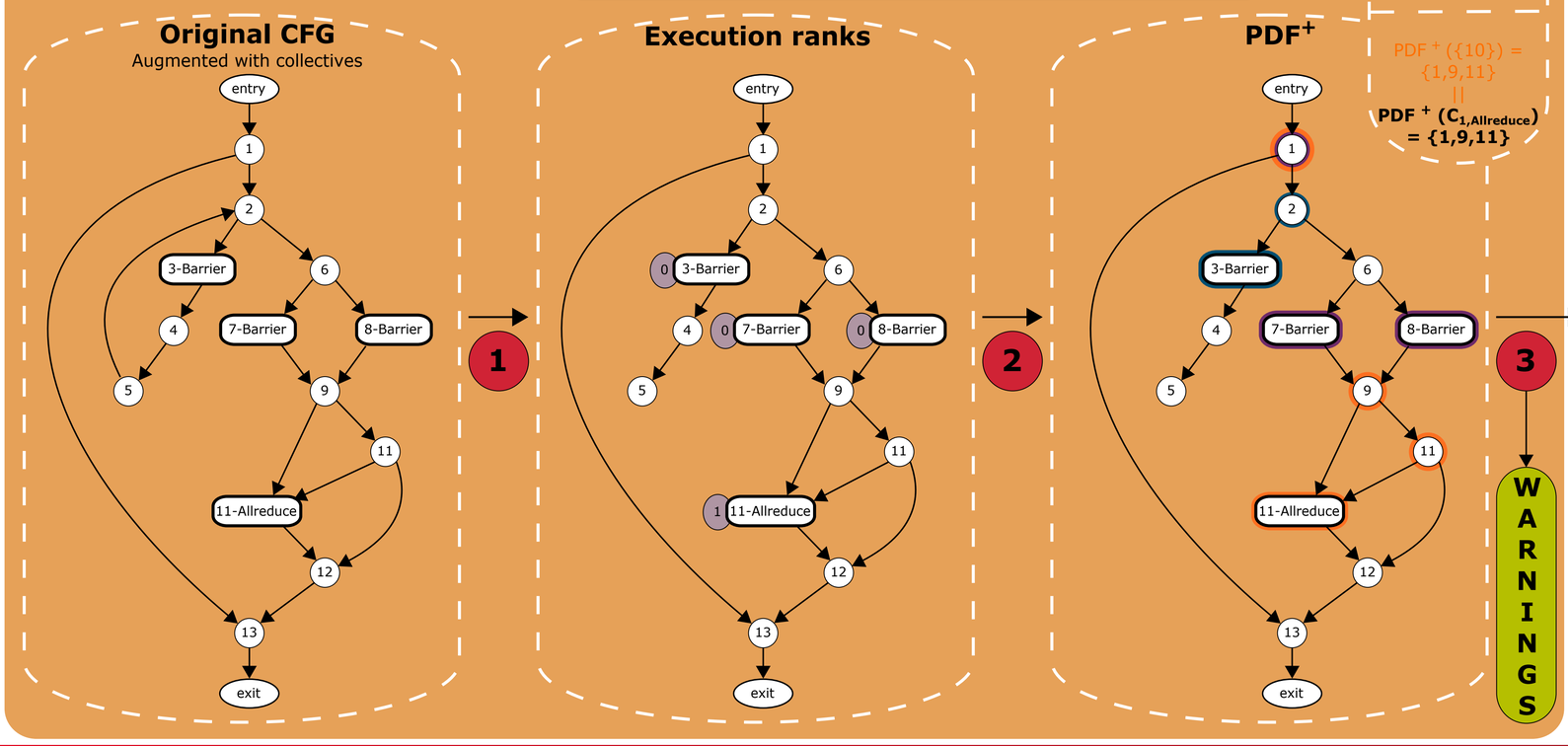
3 Static analysis

- **Postdominance:** A node *u* postdominates a node *v* if all paths from *v* to exit go through *u*.
- **PDF:** The set of nodes *v* such that *u* postdominates a successor of *v*, but not the node *v*.
- **PDF$^+$:** The iterated postdominance frontier is defined as the transitive closure of PDF [3].

Static analysis based on the CFG
- Intermediate representation
- Programming language independant
- Target machine independant
- Model all program exectuions
- Right level to study instruction flow

Static analysis steps

1. **Compute execution rank for nodes with collectives**
   - Minimum number of collectives encountered so far from entry to exit
2. **Check the PDF$^+$ of each collectives names c, in each rank k**
   - If the PDF$^+$ is empty, all paths from entry to exit execute the same k^{th} collective *c*. If not, there might be an issue with the sequence of collectives.
3-1. **Issue warnings for problematic nodes**
   - The tested nodes are stored in a list *O* for static instrumentation
   - The set of nodes given by the PDF$^+$ are used to issue compilation warnings



4 Static instrumentation

Static instrumentation steps

3-2. **Instrument problematic collectives**
   - Using the list of nodes *O* providing by analysis
   - Insert the function *CC* before the collectives
   - Function *CC* takes as inputs the collective ids and the communicator
   - Function *CC* verifies that ids are the same, then calls MPI_Abort if not

4. **Execute instrumented code**
   - In case of incorrect code, calls MPI_Abort with an error message instead of crashing.

CONTRIBUTIONS

5 Correct number of completion calls

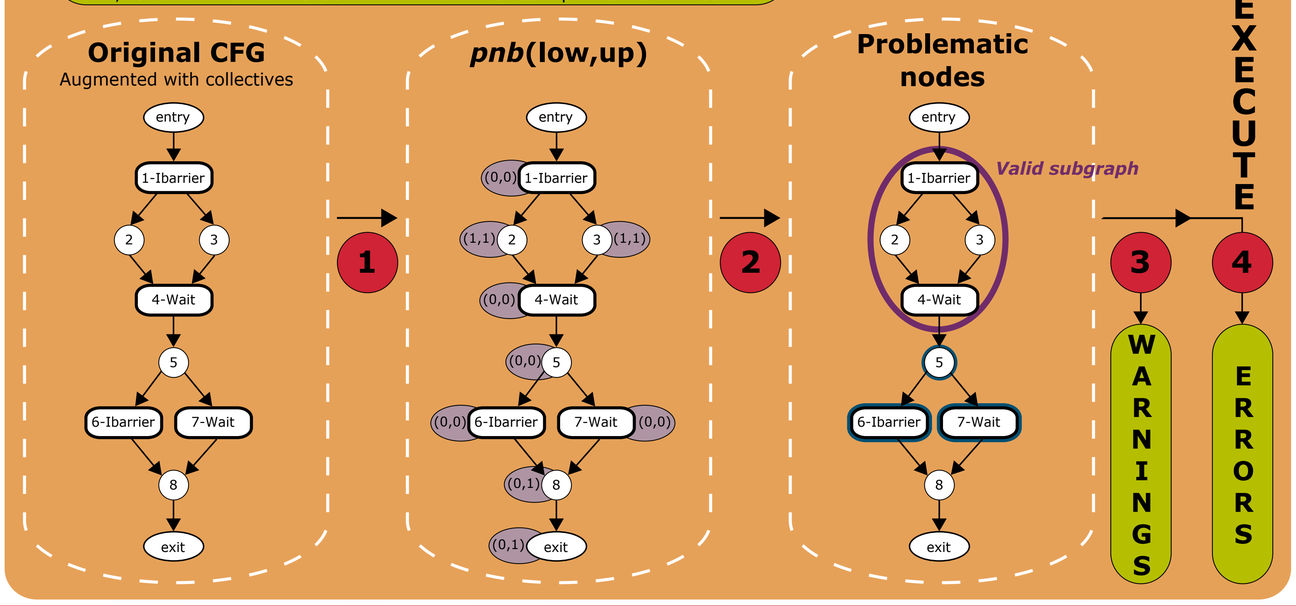
New static analysis steps

1-2. **Compute number of pending non-blocking communications**
   - lower bound = best case: a multiple completion call closes all pending non-blocking communications in the path (*pnb*=0); best path is kept in a join.
   - upper bound = worst case: a multiple completion call closes only one pending non-blocking communications in the path (*pnb*--); worst path is kept in a join.

2-2. **Check if non-blocking communications are not finished**
   - If upper bound is not null, a communication ***may*** be pending at exit
   - If lower bound is not null, a communication ***will*** be pending at exit
   - If a bound is not null, all interesting nodes are stored in problematic list

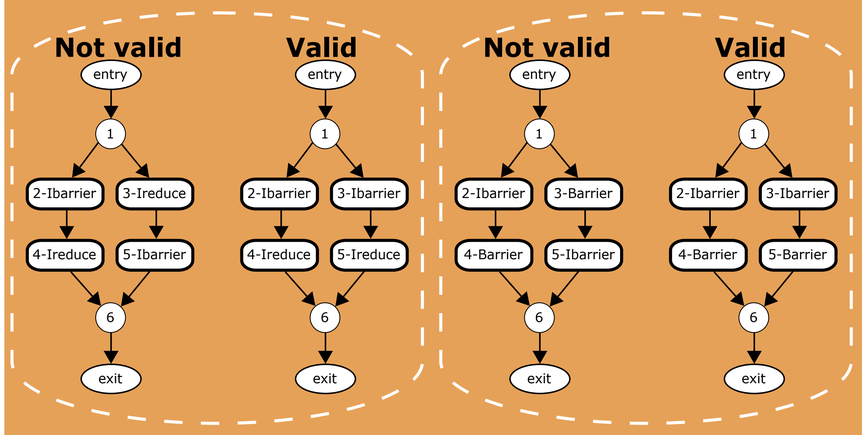
2-3. **Remove nodes in valid subgraph from problematic list**
   - If a completion call node *v* postdominates a communication node *u*, and the upper number of pending non-blocking communication for both nodes are null, remove all nodes between *u* and *v* from the list of problematic nodes.

Static matching of completion calls
- Not possible without data flow analysis
- To simplify we made one asumption:
  ~ completion calls close at least one pending non-blocking call (nbcom).
- Multiple completion call = completion call that may close several nbcom.
- Interesting node: fork, nbcom, completion.



6 NBC in original static analysis

Constraints on sequence of NBC
- Same constraints between NBC as blocking collectives
- Same constraints between blocking and non-blocking collectives.

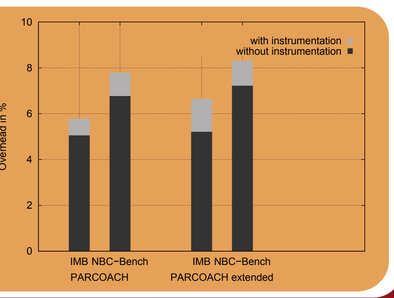


Modified static analysis steps

Perform the same steps
- Add the non-blocking collectives names to the list of collectives tested in the analysis.

7 Results

Compilation time overhead
- Tested on IMB 4.0 and our own benchmarks
- Less than 2% addition from original PARCOACH analysis overhead
- Overhead remains under 10% compared to initial compilation time



[1] E. Saillard, P. Carribault, and D. Barthou, PARCOACH: combining static and dynamic validation of MPI collective communications, Intl. Journal on High Performance Computing Applications (IJHPCA), 2014
[2] T. Hilbrich, B. R. de Supinski, F. Hänsel, M. S. Müller, M. Schulz, and W. E. Nagel, Runtime MPI collective checking with tree-based overlay networksr errors in the usage of MPI collective operations, In European MPI Users' Group Meeting, pages 129-134, 2013
[3] R. Cytron, J. Ferrante, B. Rosen, M. Wegman, and F. Zadeck, Efficiently computing static single assignment form and the control dependence graph, In ACM TOPLAS, pages 13(4):451-490, 1991

Contact: julien.jaeger@cea.fr