Co-Array Fortran

Abdullah Celik

January 30, 2014

Contents

Basics

- belongs to family of PGAS model languages (like UPC, Titanium etc.)
- language extension to Fortran
- incures Single-Program-Multiple-Data (SPMD) model
- SPMD model means:
  a) one program is replicate n times
  b) every replication is called *image*
  c) every image has its own set of data
  $\Rightarrow$ multiple set of data
- additionally:
  images run asynchronously
  $\Rightarrow$ different execution path

Basics

Opening Question

"What is the smallest change required to convert Fortran into a robust, efficient parallel language?"

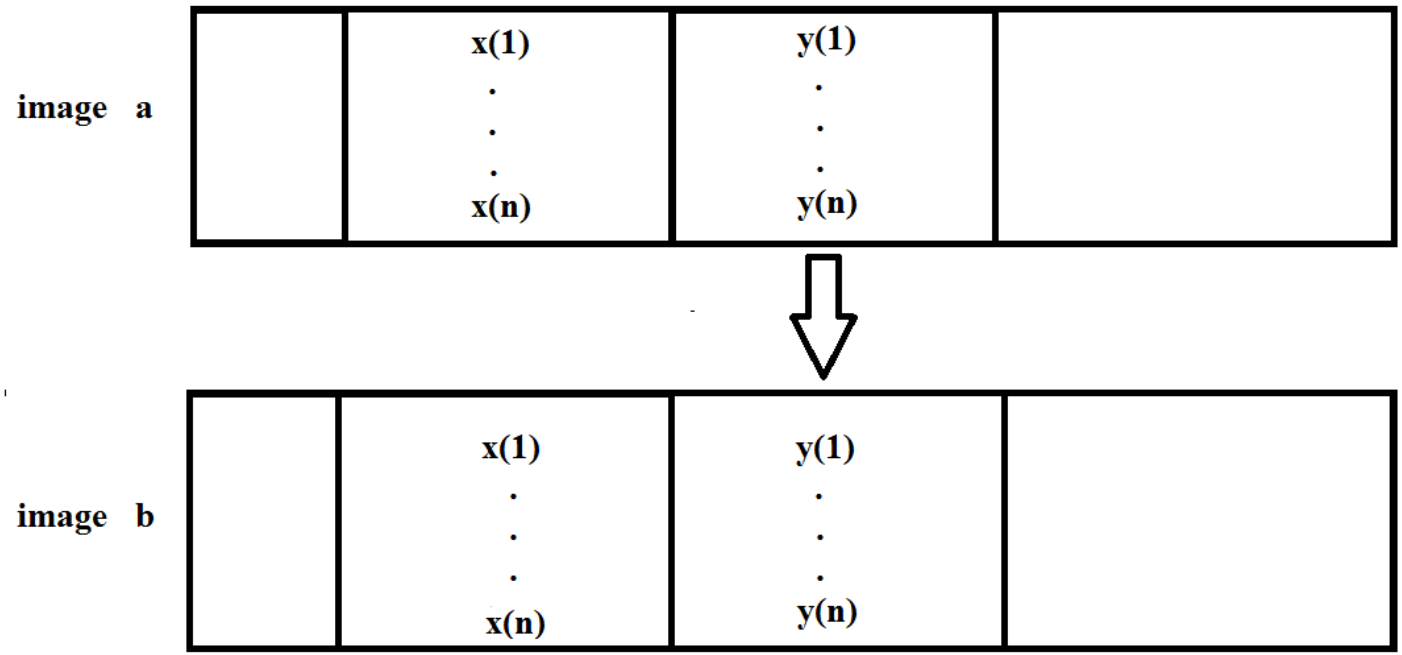
Principal Extension

- For computation: every image uses its own local set of data
- Question I: But what happens if one image needs a data from another image?
- Answer: co-array
- co-arrays are shared data
- clarify relationship between images
- Question II: How ?

Introduction to co-array syntax

1. real, dimension(n)[*] : x, y
2. y(:)= y(:)[a]

Figure 1



Design Shortcomings - Weaknesses

- co-arrays allocated over all images
- co-arrays = global variables
  $\Rightarrow$ no allocation into a local variable possible
- no global pointers available

Co-Array Fortran 2.0

- CAF 2.0 contains the following major renewals:
  a) processor subsets
  b) topologies
  c) enhanced synchronization
- two aims:
  1. eliminate original design drawback
  2. improve the language

Processor Subsets

- possibility to divide work in in components
- Question: How?
- Answer : clustering of images in **teams**
- Proceeding:
  1. all images in a default team
  2. creation of a certain number of teams
  3. assignment of images to created teams
  4. additionally every image has a rank

Figure 1

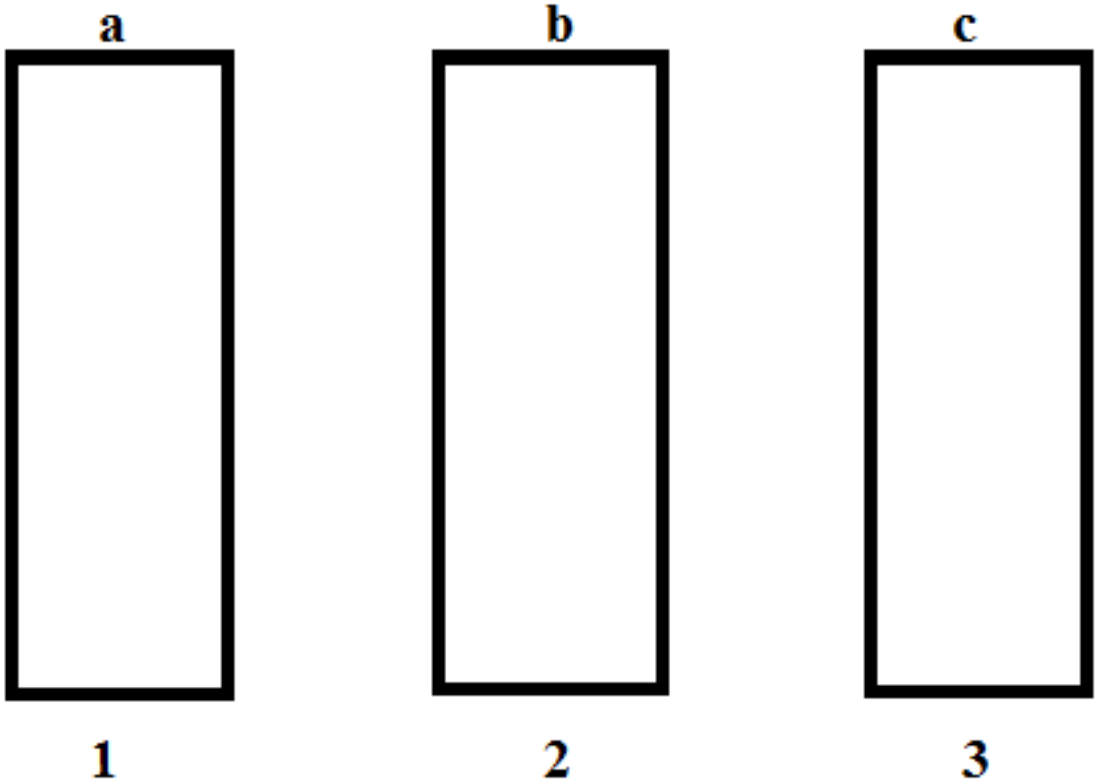


Figure: Phase 1 - Before dividing into teams

Figure 2

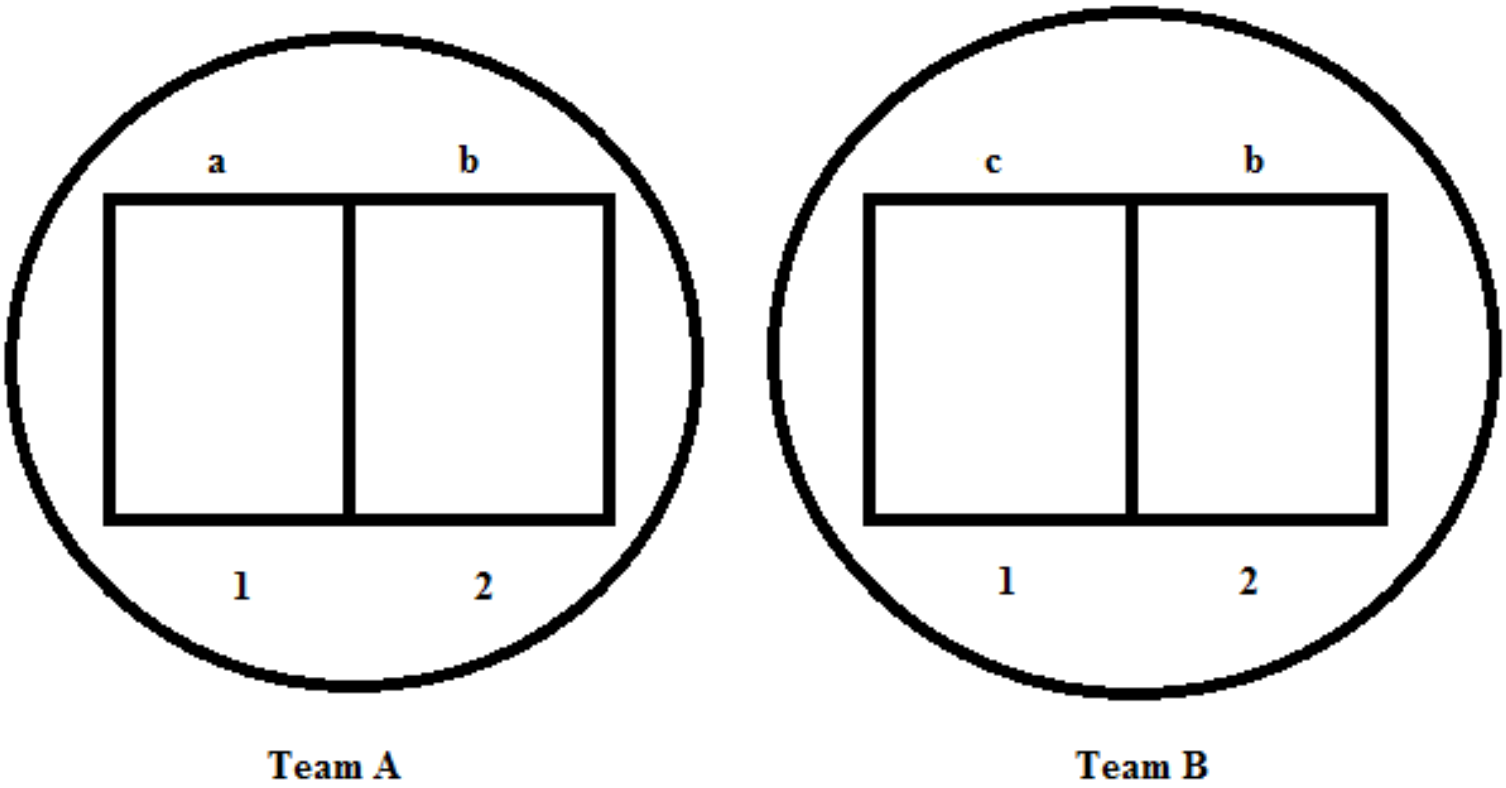


Figure: Phase 2 - After dividing into two teams

Implementing

- idea: splitting and merging
- Use of the following functions:

1. For splitting

team_split(**existing_team, color, key, new_team, result_colors, result_teams**)

2. For merging

team_merge(**existing_teams, new_team**)

Implementing

- Consider the two VIPs (**V**ery **I**mportant **P**arameters):
  1) color
  2) key
- color: provides the information to which team the image is assigned
- key : the rank of the image in the team

Implementing

- Important :
  Images can be distributed over several teams
  $\Rightarrow$ not disjoint
- Parameters for distribution:
  a) **result_colors**: vector of integers
  b) **result_teams**: vector of team variables

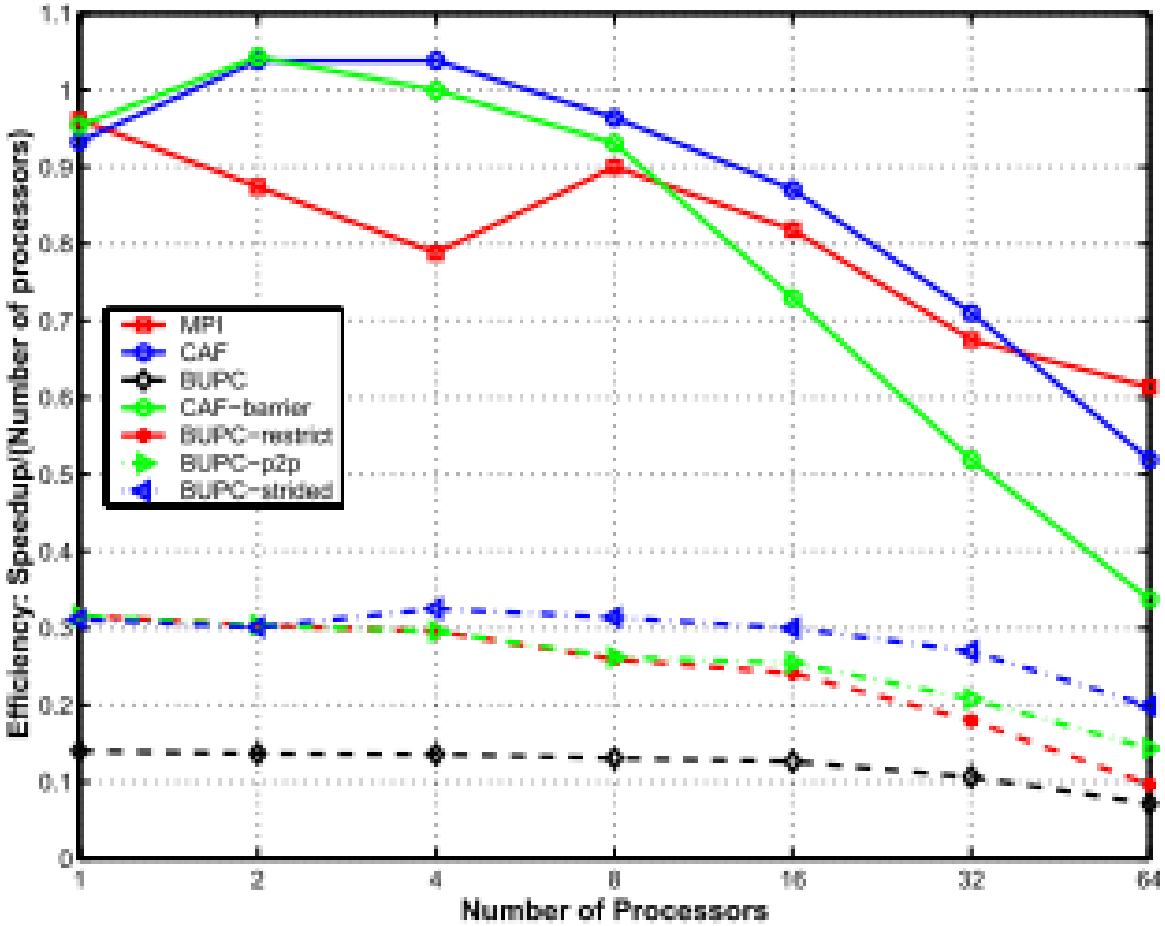
Advantages

- Major reasons for using processor subsets:
  1. minimization of communication latencies
  2. deficiency of collective can be remedy by programmer
  3. readable code

Information about Experiment

- NAS MG
- in NAS MG: a solution to a discrete Poisson problem is created approximately
- for creating a solution : V-cycle multigrid algorithm
- Scientific Computing
- for Co-Array Fortran code : cafc compiler
- for Unified Parallel C code: BUPC compiler

Comparison


Efficiency: Speedup/(Number of processors)
Number of Processors
MPI
CAF
BUPC
CAF−barrier
BUPC−restrict
BUPC−p2p
BUPC−strided
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1
2
4
8
16
32
64

Evaluation I

- CAF uses **vectorization of communication**
  $\rightarrow$ with the help of the bracket notation
- CAF also contains **synchronization strength reduction**
  means: transform the barrier synchronization into simply point-to-point notify/wait
- Note:
  CAF-barrier applies communication vectorization **AND** barrier synchronization
  $\Rightarrow$ performance of CAF-barrier decreases with a high number of processors

Evaluation II

- Question:
  Why there is such a big difference between CAF and UPC ?
- Answer:
  in all UPC versions: local pointers to access data
- **BUT**: Note that BUPC-strided and BUPC-p2p uses point-to-point synchronization
  $\Rightarrow$ increases performance in contrast to BUPC

What you should keep in mind...

1. bracket notation [ ]
2. SPMD model: 1 Program $\Longleftrightarrow$ multiple set of data
3. suitable for problems from Scientific Computing
4. provides better features than UPC

References

John Mellor-Crummey, Laksono Adhitanto, William N. Sherer III.,Guohua Jin

A new vision for CoArray Fortran

Robert W. Numrich, John Reid

Co-Array Fortran for parallel programming

Robert W. Numrich, John Reid

Co-arrays in the next Fortran Standard

Christian Coarfa, Yuri Dotsenko, John Mellor-Crummey, Francois Cantonnet, Tarek El-Ghazawi, Ashrujit Mohanty, Yiyi Yao, Daniel Chavarria-Miranda

An Evaluation of Global Address Space Languages: Co-Array Fortran and Unified Parallel C

The End