CROCO – training 2024 PSF Barcelonette

Introduction to CROCO and Parallelization

CROCO
Coastal and Regional Ocean COmmunity model

Outline

1. Available Parallelization options in CROCO
2. MPI Concept and techniques
3. MPI Basic setup in Croco
4. Advanced MPI CPP options in CROCO
5. SUMMARY FOR MPI // in CROCO

Available paradigms in CROCO

CROCO
Coastal and Regional Ocean COmmunity model

1.MPI Parallelization

designed for distributed systems, such as clusters of supercomputers where each node has its own memory. Each process in MPI operates in its own memory space, which requires explicit message passing to communicate between them. This makes it ideal for distributed memory environments.

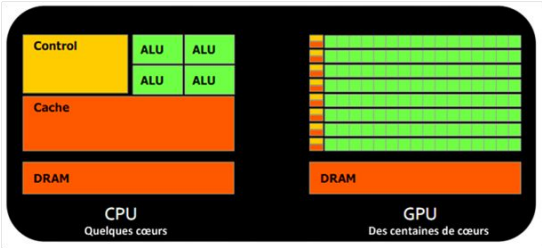
2. OPENMP Parallelization

designed for shared-memory systems, typically multiprocessor or multicore computers with shared memory. The threads created by OpenMP share the same memory and can communicate directly without message passing.

Coming soon…

CROCO
Coastal and Regional Ocean COmmunity model

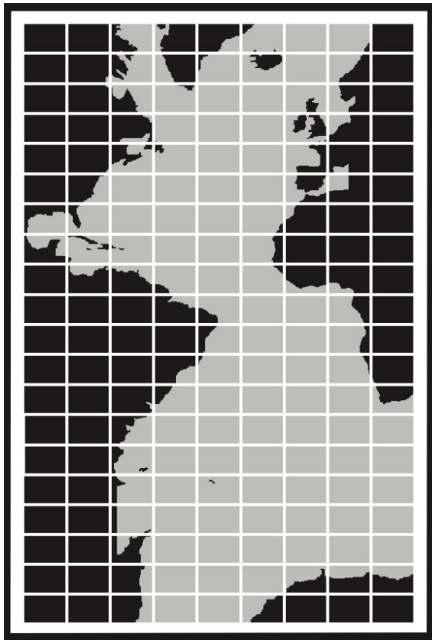
- no hybrid MPI/OpenMP version
- GPU version under development



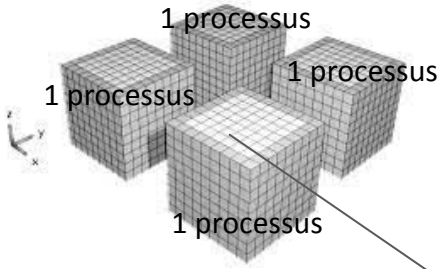
* MPI // is preferred!

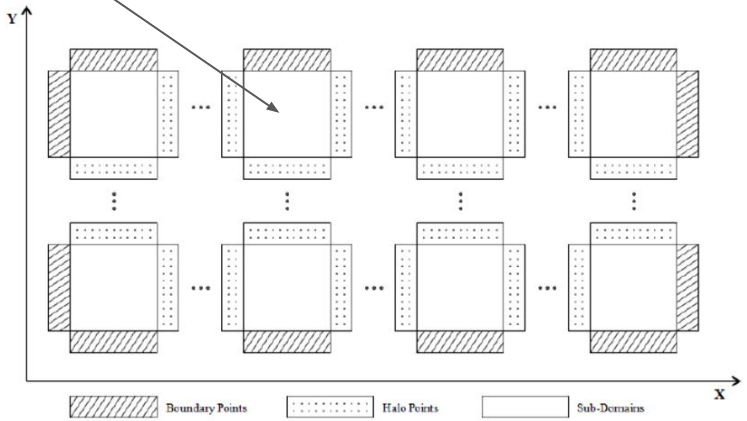
Concept and techniques : domain decomposition

MPI 9x20 processus

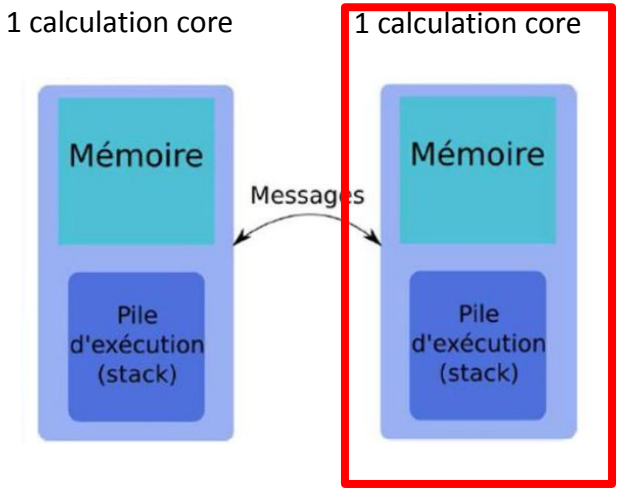


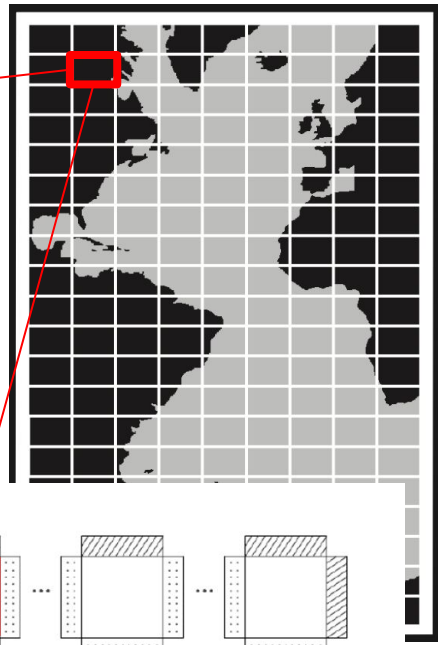
MPI 4 processus



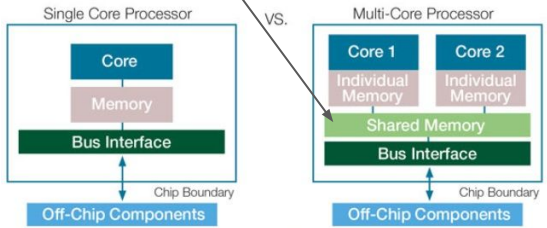


MPI (Message Passing Interface) : distributed memory

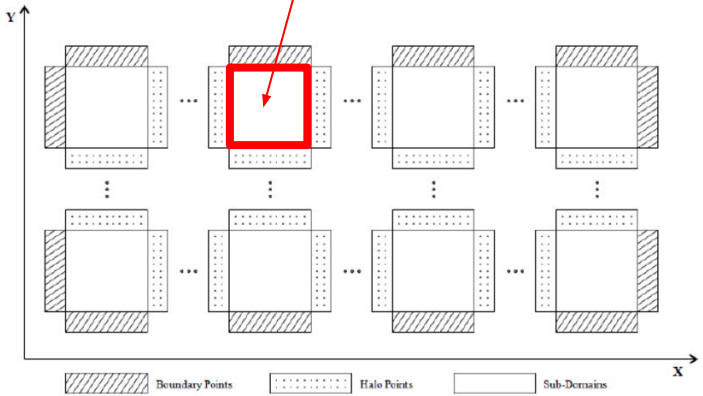




=> compute cores do not have access to a common memory
=> exchanges via network message
=> in practice, MPI also manages shared memory, but this is transparent to the user.



Comparaison entre les processeurs uni-core et multi-core



How to configure MPI in CROCO

Coastal and Regional Ocean COmmunity model

STEP1 : edit files

1) param.h

Specify tiles number in xi and eta directions => NP_XI, NP_ETA,

⚠ maximum of tiles number in ETA direction is preferred

Variables in param.h:

- `NP_XI`: decompostion in XI direction
- `NP_ETA`: decomposition in ETA direction
- `NNODES`: number of cores (=``NP_XI*NP_ETA``, except with `MPI_NOLAND`)
- `NPP = 1`
- `NSUB_X` and `NSUB_ETA`, number of sub-tiles (almost always =1)

```
!
! Domain subdivision parameters
! ====== =========== ==========
!
! NPP             Maximum allowed number of parallel threads;
! NSUB_X,NSUB_E   Number of SHARED memory subdomains in XI- and
!                                                  ETA-directions;
! NNODES         Total number of MPI processes (nodes);
! NP_XI,NP_ETA   Number of MPI subdomains in XI- and ETA-directions;
!
      integer NSUB_X, NSUB_E, NPP
#ifdef MPI
      integer NP_XI, NP_ETA, NNODES
      parameter (NP_XI=1,  NP_ETA=4,  NNODES=NP_XI*NP_ETA)
      parameter (NPP=1)
      parameter (NSUB_X=1, NSUB_E=1)
#elif defined OPENMP
      parameter (NPP=4)
# ifdef AUTOTILING
      common/distrib/NSUB_X, NSUB_E
# else
      parameter (NSUB_X=1, NSUB_E=NPP)
# endif
#else
      parameter (NPP=1)
```

4

param.h

2) cppdefs.h :

activate MPI => #define MPI

STEP2 : compile

./jobcomp

STEP3 : execute

- mpirun -n 4 ./croco

(or mpiexec or other)

Processor Numbers= NNODES

Online documentation:

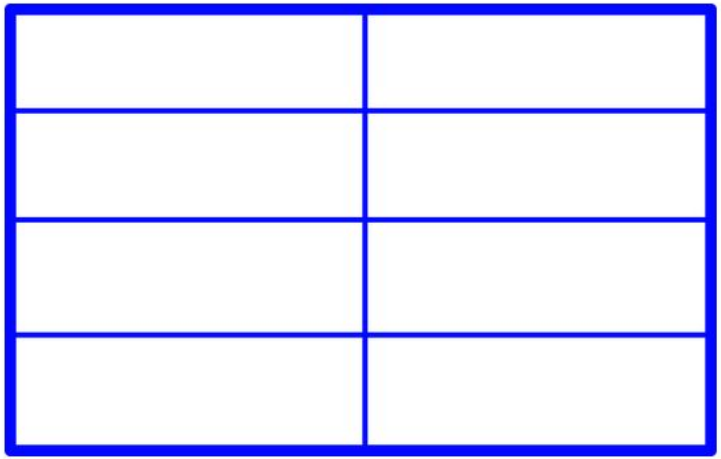
https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.parallel.html

Example of MPI domains

Example 1:

8 cores:

- NP_XI=2, NP_ETA=4, NNODES=8
- NPP=1, NSUB_X=1, NSUB_E=1

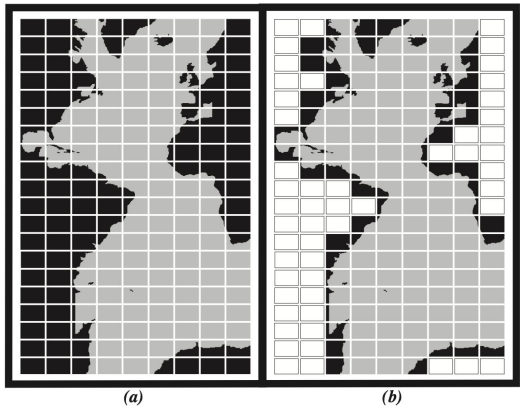


MPI CPP OPTIONS in CROCO
For writing output files

AVOID CALCULATION IN LAND AREAS (1)

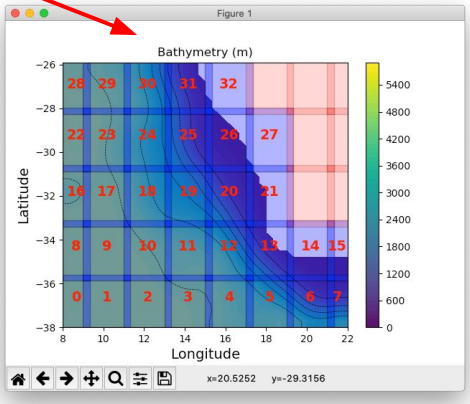
CROCO
Coastal and Regional Ocean COmmunity model

1. Preprocessing
In directory CROCO/MPI_NOLAND :
- read README
- compile: edit makefile + make
- edit namelist : grid file name, max number of procs
- execute: ./mpp_optimize
- view : ./mpp_plot.py croco_grd.nc benguela-008x005_033
- re-read README ...

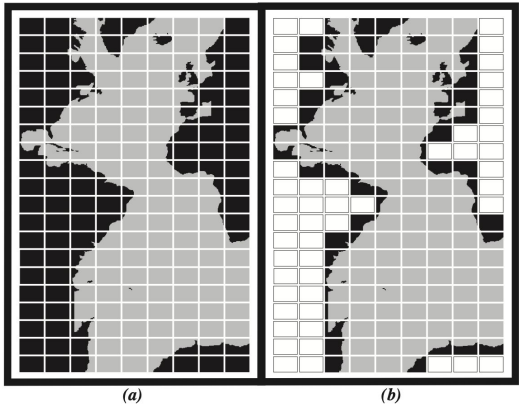



```
!------------------------------------
!      namproc
!------------------------------------
!   jprocx = maximum number of proc
!
&NAMPROC
      jprocx=220
!
!-----------------------------------------
!      namfile  of filename
!-----------------------------------------
! NAMELIST /namfile/ cbathy
! cbathy = name of the  bathy/mask file(nc)
! covdta = Root for the overdata file name .
! Complete name will be {covdta}.{NP_XI}x{NP_ETA}_{NPP}
&NAMFILE
      cbathy='croco_grd.nc'
      covdta = 'benguela'
/
```

```
-bash
(base) sdb-benshila:MPP_PREP rblod$ ./mpp_optimiz
 Number of pts     :        1936
 Number of sea pts :        1411

  optimum choice
  ==============

 --> Number of CPUs : NNODES =           33

  NP_XI =           8  NP_ETA =           5
  Lm =            6  Mm =            9

  number of sea CPUs                33
  number of land CPUs                7
  average overhead          0.69463869463869454
  minimum overhead          0.138461545
  maximum overhead           1.00000000
  nb of overhead p. < 10 %                    0
  nb of overhead p. 10 < nb < 30 %            3
  nb de overhead p  30 < nb < 50 %            5
  number of integration points             4290
  number of additionnal pts                2398
  % sup                              2.26744175

(base) sdb-benshila:MPP_PREP rblod$
```



AVOID CALCULATION IN LAND AREAS (2)



2. Three files to edit in CROCO

-cppdefs.h : CPP OPTION: #define MPI_NOLAND

- param.h: insert values for NP_XI, NP_ETA and NPP given by the preprocessing

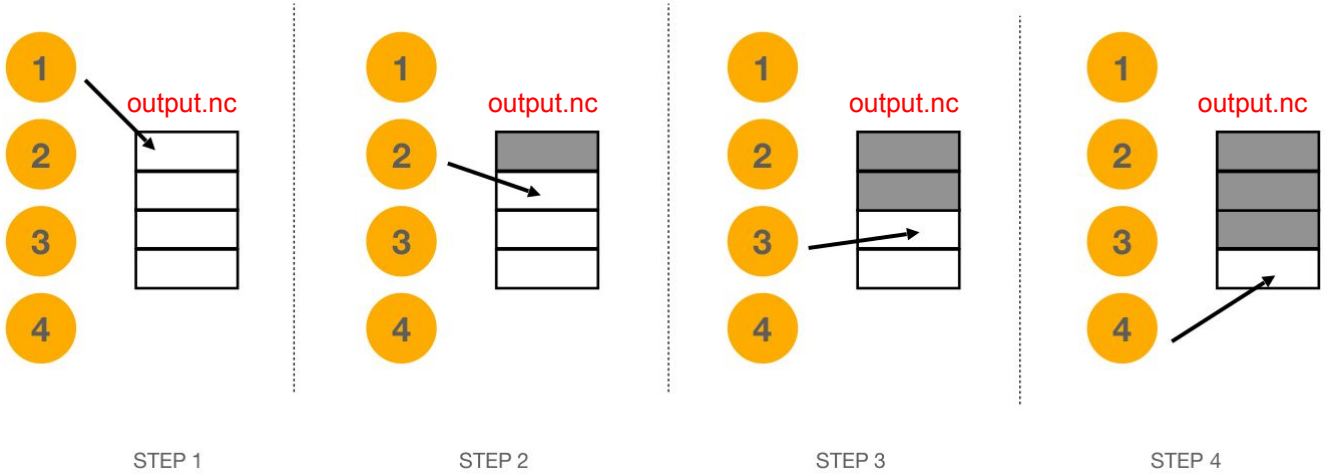
- MPI_Setup.F: be carefull to the name of grid file
(NPP ou NNODES <= NP_XI x NP_ETA)

3. Compile and execute

WARNING : grid file as to be called croco_grd.nc (or to be changed in MPI_Setup.F)

Writing MPI 1/4 files: by default

mpirun -np 4 ./croco. (NP_ETA=4)

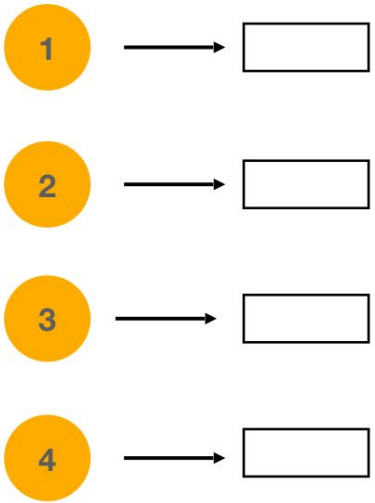


sequential writing, waiting for each proc to finish before the next one writes
Very inefficient !!!!!!!

Writing MPI files 2/4: parallel files

#define PARALLEL_FILES

mpirun -np 4 ./croco. (NP_ETA=4)



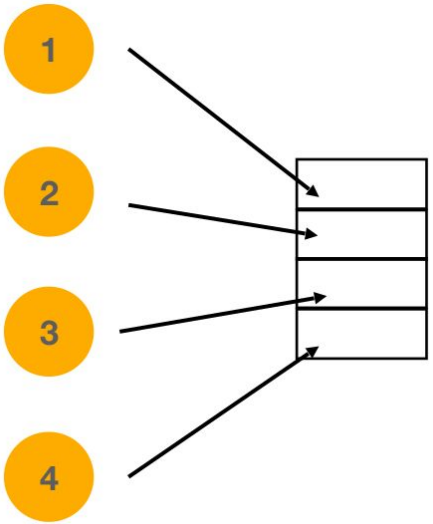
Fast, but many output files

Need to recombine them
(cf ncjoin utility)

Writing MPI 3/4 files: parallel writing

#define KEY NC4PAR

mpirun -np 4 ./croco. (NP_ETA=4)



Fast with a single output file

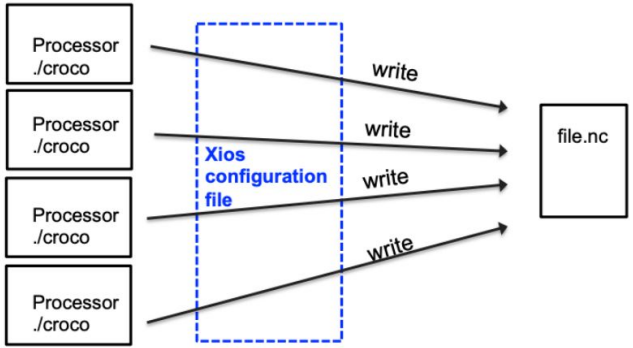
Requires NetCDF4 library
installed with parallel support

Writing MPI 4/4 files: XIOS

External server developed at IPSL http://forge.ipsl.jussieu.fr/ioserver

XIOS : attached mode

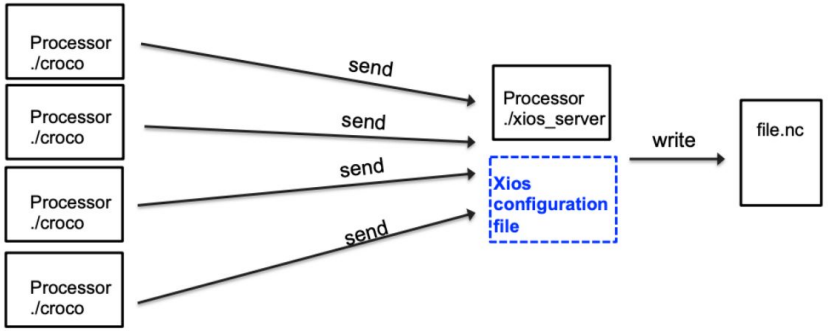
each croco executable compute and **write (like a classical library)**



Ergonomy AND efficient parallel writing BUT writing overhead

XIOS : detached mode (server mode)

each croco executable compute and **send field to the server**



- croco executables for computing only
- only xios server writes output
- Flexibility AND efficient parallel writing AND (almost) no overhead

To go further https://croco-ocean.gitlabpages.inria.fr/croco_doc/tutos/tutos.21.xios.html

Writing MPI - SUMMARY -

CROCO
Coastal and Regional Ocean COmmunity model

		Complexity to implement	Disadvantages	Advantages
1	By default	By default	By default	• Simple to implement, nothing to do
2	Parallel Files	#define PARALLEL_FILES	many output files, need to recombine them	• Simple to implement
3	Parallel writing	#define KEY_NC4_PAR library NETCDF 4 Parallel	installation of NETCDF4 can be tricky	• fast with a single output file
4	XIOS attached mode	#define XIOS, XML file	hard to install XIOS and to use it	• very ergonomic and efficient • useful with a big domain
5	XIOS detached mode	#define XIOS, XML file	hard to install XIOS and to use it	• very ergonomic and efficient • the best tool with a big domain

TP MPI

CROCO
Coastal and Regional Ocean COmmunity model

⇨ Online documentation:
https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.parallel.html

⇨ Use MPI on your regional configuration with 4 processus (NP_ETA=4)