Options for High Performance Computing in CaMa-Flood

22 April, 2024  
Dai Yamazaki and CaMa-Flood development team  
IIS U-Tokyo

In this document, the options for high-performance computing (= computationally efficient simulations) are explained. This note includes explanations of following options:

* Adoptive Timestep
* Single-precision Mode
* OpenMP parallelization
* MPI parallelization, OpenMP/MPI hybrid parallelization
* Vector-processor code
* Bit-identical simulation

Some sample codes are available in etc/options\_HPC/

# Adaptive timestep

## Calculate maximum acceptable timestep

By using adaptive timestep option, the time step ΔT time is automatically optimized to maximize computational efficiency while avoiding numerical instability (Yamazaki et al. 2013). The adaptive time step   
ΔTmax can be activated by setting LADPSTP=.TRUE. in the namelist. It is activated as a default, and recommended to use.

The maximum acceptable timestep is estimated by the CFL condition as below:

where ΔX is the length to the next downstream grid (nxtdst), g is gravity acceleration, and h is water depth. The coefficient α is safety parameter (set to 0.7 in default setting). In general, ΔTmax becomes smaller at higher-resolution (small grid size) and deeper water depth. The ΔTmax is calculated for all grid cells, and minimum value is used as the timestep for model physics (i.e. constant ΔT for all grids).

Note: When MPI is used, the minimum ΔTmax among all MPI process is calculated by MPI\_AllReduce commands, so the timestep becomes similar for all MPI processes..

## How timestep is controlled in CaMa-Flood

When adaptive time step is activated, CaMa-Flood use two timesteps to advance simulations. The timestep DT specified by users in the namelist represents fixed timestep DT\_fix. This timestep is used in the main time advancement subroutine CMF\_DRV\_ADVANCE. At the start of each DT\_fix timestep, the physics subroutine CMF\_PHYSICS\_ADVANCE is called. The maximum acceptable timestep is calculated at the start of the physics scheme by CALC\_ADPSTP subroutine, and the adaptive time step DT\_adp is given by dividing the fixed timestep by an integer value (to satisfy CFL condition).



Figure: Timestep control when Adaptive Timestep option is used.

The fixed timestep DT in the namelist should be determined considering the Input/Output or coupling frequency. For example, if the model only use daily forcing and output is also daily, it is OK to use DT=86400 (e.g. 60sec \* 60min \* 24hour). If input is daily but output is hourly, the fixed timestep should be DT=3600.

In general, fixed timestep should be the greatest common divisor of all input/output/coupling frequency. For example, if input runoff forcing is 10 minutes temporal resolution, and 15minutes resolution tidal downstream boundary is used, the fixed timestep should be DT=300.

# Single-precision Mode

Since version 4.1 (released in 2023 March), Single-precision mode is activated in default setting. In single precision mode, variables which does not require very strict accuracy are treated as single-precision type (4-byte REAL, float32). Only several variables are treated as double-precision tipe (8-byte REAL, float64) primary for strict water budget conservation.

When Single-precision mode is not used, all variables are treated as double-precision.

Activation of single-precision mode is controlled in Mkinclude file in adm/ directory. Please specify the pre-processing directive DSINGLE=-DSinglePrec\_CMF in Mkinclude, and compile for CaMa-Flood source code in src/ directory.

The types of the variables are controlled by src/parking1.F90 code. The tag JPRD is used for the variables always treated as double precision. The tag JPRB is used for the variable whose precision is changed according to Single-precision mode setting as below.

```

#ifdef SinglePrec\_CMF

INTEGER, PARAMETER :: JPRB = SELECTED\_REAL\_KIND(6,37)

#else

INTEGER, PARAMETER :: JPRB = SELECTED\_REAL\_KIND(13,300)

#endif

INTEGER, PARAMETER :: JPRD = SELECTED\_REAL\_KIND(13,300)

```

CaMa-Flood variables are declared in variable control modules (files named yos\_cmf\_\*.F90 in src/ directory).

Always-double precision variables and precision-switchable variables are declared like the below examples.

```

!\*\*\* river & floodpain

! storage variables are always in double precision

REAL(KIND=JPRD),ALLOCATABLE,TARGET :: P2RIVSTO(:,:) !! river storage [m3]

REAL(KIND=JPRD),ALLOCATABLE,TARGET :: P2FLDSTO(:,:) !! floodplain storage [m3]

REAL(KIND=JPRB),ALLOCATABLE,TARGET :: D2RIVOUT(:,:) !! river outflow [m3/s]

REAL(KIND=JPRB),ALLOCATABLE,TARGET :: D2FLDOUT(:,:) !! floodplain outflow [m3/s]

```

Some critical variables (especially, variables related to the water budget conservations) were kept as Double Precision even when Single Precision Options were used. Based on trial and errors, prognostic variables related to water storage, such as RIVSTO, FLDSTO in default configulation, plus GDWSTO, DAMSTO, LEVSTO for advanced options). So that, the same level of water budget error compared to the default Double Precision mode, without any additional computational cost in Single Precision Mode.

The water budget error at each time step in the default mode is ~1.E-10 [km3], and the error stays at the same range ~1.E-10 [km3] in Single Precision Mode. If all variables (including the above critical variables) are switched to Single Precision, the water budget error order increased to ~1.E05 [km3] for each calculation step, which is not negligible for climate-scale simulations.

# OpenMP parallelization

OpenMP multi-thread parallelization is implemented in CaMa-Flood and activated in default setting.

In order to activate OpenMP parallelization, please edit adm/Mkiclude file. It can be activated by compiling Fortran90 source code with OpenMP option (Intel Fortran: FCMP = ifort -qopenmp, gfortran: FCMP = gfortran -fopenmp).

Then, please set OpenMP thread number in the gosh script (e.g. export OMP\_NUM\_THREADS=16).

In Fortran90 source code, the DO loops which is tagged with OpenMP directive (!$ OMP PARALLEL DO / !$OMP END PARALLEL DO) are parallelized in the simulation.

Note that water budget calculation along river network needs to be parallelized with !$OMP ATOMIC directive. This is needed because one grid can have multiple upstream grids, and access to the storage variable memory should be done with exclusion control.

# MPI parallelization

A simple MPI parallelization is implemented since v4.1.

When MPI is used, global land domain is divided into several parts and calculated independently as different MPI processes. In order to maximize the efficiency, grids in a same river basin are allocated to the same MPI process, so that between-MPI-node communications can be minimized.

MPI parallelization in CaMa-Flood v4.1 is assumes below points:

* MPI regionalization is defined by a pre-processed map (mpireg.bin).
* Global map and global runoff forcing is used as input (no option for regionally distributed map and runoff input, as of now)
* Results of MPI nodes are gathered to the master node, and output is saved as a global map.
* Time step ΔT is shared with all MPI nodes.
* MPI + OpenMP parallel computation is supported.

## How to use MPI parallelization

Activation of MPI parallelization requires (1) preparation of the MPI region map, and (2) compiling Fortran90 code with MPI option. Then, (3) Edit gosh script and execute simulations

### (1) MPI region map preparation

Basically, grids in one river basin are assigned to a same MPI region to reduce communication cost. However, as some river basins are inter-connected by bifurcation channel, river basin ID should be updated, and these inter-connected basins should be merged when MPI regionalization data is prepared.

Codes for MPI regionalization (considering inter-basin connectivity) are prepared in map/src\_param/ directory.

All MPI related codes are aggregated in the shell script s03-mpi\_map\_setup.sh. So users can prepare the maps required for MPI by compiling Fortran90 source code and executing s03-mpi\_map\_setup.sh with some edits on the script.

Please go to your simulation map directory (e.g. map/glb\_06min), copy the src\_map directory, compile, and execute scripts.

```

cd map/$(your\_map)/

cp -r ../src/src\_map .

cd src\_map

make all

vi s03-mpi\_map\_setup.sh

./s03-mpi\_map\_setup.sh

```

The s03-mpi\_map\_setup.sh script contains below codes:

#### set\_bifparam.F90

Code to set river bifurcation parameters. (Note: this code is also used in s01-channel\_params.sh). Bifurcation parameter file is needed to calculate basin mask map considering bifurcation connectivity.

Bifurcation channel parameters (bifprm.txt) is generated, by modifying the original bifurcation parameter source file (bifori.txt).

#### set\_bif\_basin.F90

Code to analyze bifurcation channel parameter file bifprm.txt, and integrate river basins inter-connected by bifurcation channels.

Each bifurcation pathway connects one grid with another grids. If connected grids belongs to difference river basins, the ID of the smaller river basin is changed to the ID of the larger basin. Thus, these basins are treated as “basins connected by bifurcation channel” and allocated to the same MPI node.

##### <Output files>

* bifbsn.bin = basin ID file, updated by considering inter-basin bifurcation channel connectivity
* bifcol.bin = basin color file for visualization, corresponding to bifbsn.bin
* bifmod.bin = supplemental map. 1= Basins absorbed by merging. 2= Basins expanded by merging
* bpoint.bin = point of inter-basin connectivity. 3= River bifurcation connectivity, 2=overland bifurcation connectivity, 1=not connected in bifbsn.bin

地図が表示されたゲーム画面

低い精度で自動的に生成された説明

Figure: River basin ID map (left) before and (right) after considering bifurcation channel connectivity. We can observe some basins are merged through bifurcation channels (e.g. Mekong delta, Amazon and Orinoco)

##### <Note / Options>

To avoid the generation of very big integrated basins, some inter-basin bifurcation channels are excluded when merging basins.

In a default setting, inter-basin channels are excluded when the size (grid number) of the two basins to be merged is larger than a threshold (6% of global land grids).

As a result, **recommended MPI regions are up to 16** in default setting.

If you want to increase the MPI regions, please use 'MaxMPI' option and add MaxMPI as argument.

```

% ./set\_bif\_basin MaxMPI

```

Then, the threshold is changed to "3% of global land grids", and **recommended MPI regions increased to 30**.

##### set\_mpi\_region.F90

Allocate grids to MPI regions, following the updated basin ID file bifbsn.bin. MPI region file (mpireg-##.bin) is generated. Grids are allocated in order to equalize the grid numbers in each MPI region.

Please use the MPI region number (MPI\_NP) as an argument, and then the map for that MPI\_NP is generated. For example, if you use 16 MPI nodes, please execute codes with argument “16”

```

./set\_mpi\_region 16

```

Then, MPI region map mpireg-16.bin is generated.

In the sample s03-mpi\_map\_setup.sh script, MPI region files for MPI\_NP=4,8,16 are generated.

屋内, 地図, テーブル, 座る が含まれている画像

自動的に生成された説明

Figure: MPI region map. (let) MPI\_NP=16, (right) MPI\_NP=30 with MaxMPI option.

### (2) Compile Fortran90 codes with MPI option

For activating MPI parallelization in CaMa-Flood, the source code should be compiled with MPI options.

Please edit adm/Mkinclude, and specify DMPI=-DUseMPI\_CMF, and also specify the compiler with MPI functions (for example, use FCMP = mpif90 -fopenmp, depending on your environment) Then, the Frotran90 codes are compiles with MPI functions.

Here, how MPI is used in CaMa-Flood Fortran90 code is briefly explained.

#### cmf\_ctrl\_mpi\_mod.F90

All subroutines including MPI functions are included in the module file: cmf\_ctrl\_mpi\_mod.F90.

It contains initialization, finalize, time step control, and data gathering for output.

#### MPI region calculation in the main code

MPI process number is stored as REGIONTHIS, while number of MPI process is sorted as REGIONALL.

CaMa-Flood converts river network data (nextxy.bin) from 2D map to 1D vector in initialization. Using MPI region file (mpireg.bin), only grids corresponding to REGINTHIS are converted to 1D vector in each MPI node. Thus calculation is done almost same as the regionalized simulations, except for few points.

* Adaptive time step ΔT is shared with all MPI nodes by MPI\_AllReduce function (subroutine CMF\_MPI\_ADPSTP)
* The output data is gathered to the primary node (REGIONTHIS=1, using MPI\_AllReduce function (subroutine CMF\_MPI\_AllReduceR2MAP, CMF\_MPI\_AllReduceR1PTH)

### (3) Edit shell script and execute simulations.

The sample script to execute CaMa-Flood simulation with MPI is prepared in gosh/test5-mpi.sh. In order to activate MPI, MPI region map (prepared in the previous section) should be specified. For example, if you are using 16 MPI nodes, please prepare mpireg-16.bin, and specify the map by CMPIREG=”mpireg-16.bin” in the namelist.

Finally, please check your computer environment and modify the MPI-related parts of the script (e.g. mpirun).

Basically, using MPI/OpenMP hybrid parallelization is firster than only using OpenMP.



Figure: Calculation time (results on UTokyo IIS server)

# Vector-processor Optimization

The subroutine to calculate flood stage (CMF\_CALC\_FLDSTG in cmf\_calc\_fldstg\_mod.F90 ) needs a special care when using in a vector-processor machine (such as Earth Simulator).

This subroutine contains a loop to decide the current flood stage, which is difficult to parallelize for vector-processors. Thus, another subroutine with optimization for vector-processor is prepared (CMF\_OPT\_FLDSTG\_ES in cmf\_calc\_fldstg\_mod.F90 ).

This code can be activated by speficing option: "LSTG\_ES = .TRUE." in the namelist (shell script)

NOTE: The subroutine (CMF\_OPT\_FLDSTG\_ES) is significantly slower when used for scaler-processor machine.

# Bit-identical simulation

If you need to get bit-identical results (i.e. exactly same value, even considering rounding error in double precision), please specify activate DATM=-DNoAtom option in adm/MkInclude.

When this option is turned on, OMP\_ATOMIC parallelization is avoided (no parallelization for loops with OMP\_ATOMIC function). This makes simulations slower, but bit-identical results (which is not depending on the calculation order issue when using OpenMP) can be achieved.

Note: OMP\_ATMIC used to be controlled by LBITSAFE option, but it was switched to #ifdef DNoAtom in order to avoid complex way of coding