Scalabilization of Fullpos/927

A Report on the scalability issues of F*ull*pos/927 in the Arpege / Aladin

By Tayfun DALKILIC
(TSMS)

Supervisor Ryad EL KHATIB
(METOE-FRANCE - CNRM/GMAP)

25 September 2010

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

F*ull*Pos is a Post-Processing package embedded inside the Arpege/IFS/Aladin software. It is fully compatible with the model itself. Mainly, it is composed of two pieces, vertical interpolations and horizontal interpolations.

F*ull*Pos needs the auxiliary library for the I/Os and for the spectral transforms it uses TFL and TAL externals. It can be run with respect to the proper namelist of Arpege/IFS/Aladin cycle using in the experiment. While some of the namelist variables are only private to the F*ull*Pos, model variables in the namelist are also used.

When making historical files in F*ull*Pos, it needs to invoke two parts which are required to start the control cascade from almost the beginning. By removing the code below the condition LFPART2 and making more straightforward mechanism instead is planed in this study.

2. AN OVERVIEW OF THE F*ull*POS SOURCE CODE TREE

F*ull*Pos is included in the configuration 001 of Arpege/IFS/Aladin and it is enable to run in-line or off-line mode. The general mechanism, in terms of calling sequence from the beginning is described below.

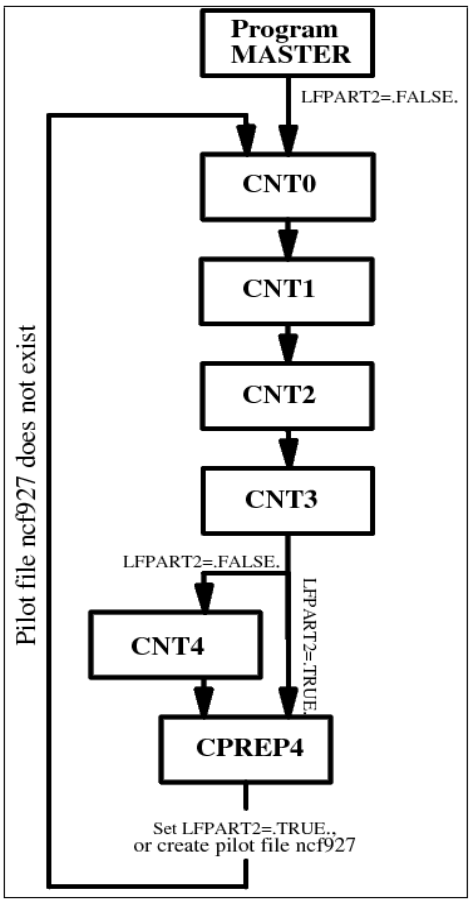

Figure 1. *General mechanism of the PP inside the model. (From FullPos tech. guide)*

Master is the main program that initiates calling sequence at the top of the source tree. There are some control levels which are named CNT0, CNT1, CNT2, CNT3, CNT4. Control level 0 initializes the constants and the namelist variables. Control level 1 and 2 are useless for F*ull*Pos. Control level 3 read the initial conditions (if needed, climatology data as well) and compute the working arrays. Control level 4 is a temporal loop which initializes the time-dependant post-processing variables and perform the post-processing inside the model temporal loop.

When F*ull*Pos is configured to make historical files, control cascade should be invoked two times in order to change the setup of the spectral transforms. So that, the mechanism control by logical LFPART2 and CPREP4 calls once more CNTO. After the second control levels completed, spectral transform setup again and the intermediate file is read previously written by external part. One of the main reason effects the scalability is this I/O part. F*ull*POS organised like that because spectral transform was not modular at the time F*ull*POS was conceived.

These sequence is controlled by logical LFPART2 which is the internal key telling whether the running part is the internal or the external one. If it is false then it is a external, and if it is true then it is an internal part(part2).

3. SCALABILITY

In order to avoid to start almost all the process from the beginning and to take the advantage of the modularity of the spectral transform, we are planing to handle two different type of geometry (Global to LAM or vice versa) at the same time which are required some modifications with respect to the spectral transform package.

3.1 Namelist

NFPSPEC is the integer number that activate the kind of new algorithm used in F*ull*POS. It is located in the NAMCT0 namelist block. If it is 0, then it may be called as ordinary post-processing which means it just performs the interpolations without changing the spectral geometry. In the case of 1, it indicates that it is one of the configuration 927 family (927 arp2arp ,e927 arp2ald ,ee927 ald2ald). If it is equal to 2 then it is a new algorithm plugged into the source code.

NFPSPEC controls the LFPSPEC at the same time. Therefore LFPSPEC is removed from the namelist and the routines SUMPINI and NAMCT0.

NFPSPEC = 0 => ordinary F*ull*POS (LFPSPEC=.F.)
NFPSPEC = 1 => 927, e927, ee927 (LFPSPEC=.T.)
NFPSPEC = 2 => new mechanism (LFPSPEC=.F.)

LFPSPEC .T. = Post-processed dynamic fields are written out as spectral coefficients
.F. = Post-processed dynamic fields are written out as grid point values.

Modified routines are;

arp/namelist/namct0.h => namelist block

arp/module/yomct0td.F90 => temporal data module to be dispatched
arp/setup/suct0.F90 => setup level 0 control commons

NFPSPEC considered as a integer number because of obtaining the consistency with the current situation (NFPSPEC 0 and 1). After the validation of new algorithm, it is planing to depart the part2.

3.2 Setup

In the case of NFPSPEC is equal to 2, there will be two different spectral geometry defined simultaneously. However, spectral transform package is bugged to handle these two different type of geometry . Therefore, some of the routines in spectral package made more flexible by doing some modifications.

3.2.1 Initializing the Transform Package

SUETRANS - Initialize the aladin transform package
SUTRANS - Initialize the transform package

New logical array LTYPE_TRANS is added to be able to store all the type of the transformations. While arguments of the array store the resolution number, logical values store if it is in global or LAM.

LTYPE_TRANS(1) = .F. means that first defined resolution is calling ESETUP_TRANS
LTYPE_TRANS(2) = .T. means that second defined resolution is calling SETUP_TRANS

SETUP_TRANS0 - called only once to have a basic initialization. According to type of the resolution it is called by SUTRANS or SUETRANS only once.

SETUP_TRANS - resolution dependent initialization in the case of arpege
SETUP_ETRANS - resolution dependent initialization in the case of aladin. Control of array allocations for aladin specific variables added. KRESOL switched to intent out argument like in setup_trans.

Each call to these routines (SETUP_TRANS and SETUP_ETRANS) creates a new resolution up to a maximum of NMAX_RESOL setup in SETUP_TRANS0 and updates the values for LTYPE_TRANS. Therefore, these routines must be called after SETUP_TRANS0.

3.2.2 Data Transposition

SUTRANS or SUETRANS is called in the second part of the setup which is SU0YOMB. Consequently, first resolution has been defined. For the setup of the second resolution SUFPTRANS is considered as a new routine which is called by SUBFPOS. After completing setup of the second geometry, global address of each grid point on each processor saved in an array both for the C+I and E zone (in the case of LELAM). Some of the informations are inquired from the transform package.

These global addresses which are computed by using the local address of the starting geometry tried to match with the global address of the target geometry. If these two global addresses

are equal to each other, local addresses in the target geometry and the number of the grid points are saved.

Thus, the number of grid point each processor will send to receiver are computed. Local address in the send buffer of the grid points will be send by each processor and the address of the receivers are also saved. Local address in the target array received by MYPROC kept as well. These couple of arrays hold the essential distribution informations of the grid points.

send-info(:,i,j) : local address in the source array (send part) of the grid point send by proc j to proc i

recv-info(j,:) : local address in the target array(recv part) received by MYPROC from proc. j

Number of grid points will be send to the other processor and their addresses together with the receiver part will be used as a input to a new routine called as TRFP2TRANS. It is mainly extract the fields both for C+I and E zone. Extracted fields are send or received according to target distribution already setup in SUFPTRANS.

TRFP2TRANS is called by STEPO that checks the already changed CDCONF (configuration of STEPO) by DYNFPOS. CDCONF(7:8)='FF' is the new configuration which is controlled by NFPSPEC=2 in the routine DYNFPOS. New configuration is activated in the horizontal and lagged vertical post-processing part of the DYNFPOS.

3.2.3 Deallocation of Transform Package Related Arrays

There are two routines which are using to release the space allocated by transform package. Trans_end under the TFL externals terminate the transform package in the case of arpege and etrans_end under the TAL externals release all the allocated arrays for aladin. However, these two routines are designed to terminate the transform package if all the defined geometries are in the same type. In the event of NFPSPEC=2, different types of geometries setup.

For instance, there is no problem to terminate all the transform package one of the trans_end routines for the first defined geometry. Terminating the second geometry by using the related routine(trans_end or etrans_end), it try to deallocate the arrays already released by previously called routine. Because of this problem, these routines are divided into two part one include the resolution dependant part, while the other part for the deallocation of the resolution independent part.

tal/external/etrans_end0.F90 => new routine for the resolution independent part

tfl/external/trans_end0.F90 => new routine for the resolution independent part

tal/external/etrans_end.F90 => modified routine for the resolution dependant part

tfl/external/trans_end.F90 => modified routine for the resolution dependant part

Combination of terminating routines are controlled by LTYPE_TRANS which are calling in arp/utility/freemem.F90. Freemem is a routine to free memory used by pointer allocations.

3.3 List of Modifications and New Routines

Name of the Routine	Modifications
sumpini.F90	LFPSPEC replaced by NFPSPEC
namct0.h	LFPSPEC replaced by NFPSPEC
suct0.F90	initialize, control and printout NFPSPEC
subfpos.F90	call sufptrans
suetrans.F90	NFPSPEC, allocation LTYPE_TRANS
sutrans.F90	NFPSPEC, allocation LTYPE_TRANS
dealfpos.F90	NIFPEL* and NFPSPEC related deallocs. added
freemem.f90	deallocations for defined geometry
sufpezo.F90	global indexing E zone
esetup_trans.F90	KRESOL and control of allocation for ALD specific variables
etrans_end.F90	JRESOL, move resol specific deallocations to etrans_end0
trans_end.F90	JRESOL, move resol specific deallocations to trans_end0
dynfpos.F90	CDCONF(7:8)='FF' new configuration
stepo.F90	call trfp2trans according to new CDCONF

Table 3.1: *List of modified source codes*

Name of the New Routine	Purpose
sufptrans.F90	routine to setup spectral transform for FPOS and scatter global addresses among the procs.
trans_end0.F90	deallocate temporary used arrays for FPOS in the case of ARP.
trans_end0.h	interface
etrans_end0.F90	deallocate temporary used arrays for FPOS in the case of ALD.
etrans_end0.h	interface
trfp2trans.F90	extract the fields and distribute to procs.

Table 3.2: *List of new source codes*

4. CONCLUSION

F*ull*POS is a post-processing package containing many features such as making ARPEGE or ALADIN history files, whether starting from a file ARPEGE or a file ALADIN. However, these sequence subdivided into two parts (internal and external part) which are required I/O operations and starting control cascade from the beginning in order to change the setup of the spectral transform . Scalability of F*ull*POS can be increased avoiding from these limitations.

In this study, spectral package debugged if it has ability to setup two different type of geometry. Some of the routines in the package are modified to have a convenient way of keeping different type of geometries. Grid point transposition from FPOS distribution to the spectral distribution is computed, however it still needs validation.

Target distribution of the grid points which are already computed will use to call the direct spectral transform (from grid-point to spectral) as a next step. Also, filter on the fields should be applied to make a selection if the fields are in grid-point or spectral before calling direct spectral transform. By the way, some of the fields will be kept in grid-point. Vector fields and orography spectral fit (if not provided by climatology) aspects should be considered as well.

Coding modular spectral norms wrapper for FPOS in terms of generalization will be useful for validation purposes at the beginning and then it is essential anyhow.

I/O part will write out spectrally fitted fields and grid-point fields into a file.