Porting of ALADIN 25T1 (second export) on LINUX PC

Andrey Bogatchev, NIMH

e-mail: *Andrey.Bogatchev@meteo.bg*

1. Introduction

The experience from experiments for running ALADIN on a two-processors LINUX PC was used for porting ALADIN 25T1.

2. System description

- two Intel Xeon processors of 2.8 Ghz clock-rate each
- 2 GB shared memory
- Two 150 GB disks, with RAID-5 system for mirroring the main file systems
- Operating system: Linux 2.4.20-30.9smp
- FORTRAN compiler – Intel 32 bit FORTRAN compiler v 8.0 for LINUX
- C compiler (if any) – GCC
- Message passing interface - MPICH2 Release 0.97
- compilation tool e-make.0.4 (Eric Sevault) – *Thank you Eric!*
- LAPACK library source code version 3.0 and corresponding BLAS library, compiled with the Intel compiler

3. Software tuning

- MPICH2 configuration options:
  --with-device=ch3:sshm --enable-f77 --enable-f90 –enable-fast

  ch3:sshm is the MPI2 device with scalable shared-memory communication for shared-memory machine.

  -enable-f77 -enable-f90 are options for supporting FORTRAN 77 and 90

  -enable-fast is option for highest performance of MPICH2.
- necessary environment variables to be exported before configuring and making MPICH2

  export FC=ifort

  export F90=ifort

  (*ifort* is the name of Intel FORTRAN compiler)
- compiling the code : the compilation should be done, using the drivers of MPICH, *mpif77* and *mpif90* for FORTRAN and *mpcc* for C.
- FORTRAN 90 flags:
  -O3 -xN -free -noauto -std90 -DLX86P -DMPI2 -convert big_endian -pc 64 -traceback -assume byterecl.

  F77 flags are the same with exception of -nofree and -DBLAS.

  -DLX86P is preprocessor definition for LINUX

  -DMPI2 is necessary for compiling the MPL routines

  -convert big_endian keeps the big endian presentation of unformatted files

  -xN is specific optimisation flag.

4. Code modifications

XRD

- introducing proper timing routines in timef.F, cptime.F
- introducing in facomp.h and lficom0.h LX86P at the proper place
- in directory grib_mf – only FORTRAN routines are used and their modifications for LINUX (*thanks to Jean-Daniel Grill and his PALADIN*)

ARPEGE and ALADIN

- namnasa.h – taken form Clear Case

- removing of double entities in USE statement in number of routines – *see Olda's report:*
- correcting misplaced declarations in some routines – usually the shape of array is declared after array declaration (see for example canari.F90, extrapad.F90, extrap.F90 and so on.
- correcting some formats, on which compiler complains – canali.F90, evcost.F90 et caetera
- eggpack.F90 – some vector functions are not working properly (may be due to compiler) and were replaced by their scalar versions:

```
! Compute XY grid points under CENTER origin
DO i=_IONE_,NB_PTS%ONX
  DO j=_IONE_,NB_PTS%ONY
    GRID_XY_C(i,j)%X = (FLOAT(i)-(FLOAT(NB_PTS%ONX+_IONE_)/_TWO_))* PDEL%ONX
    GRID_XY_C(i,j)%Y = (FLOAT(j)-(FLOAT(NB_PTS%ONY+_IONE_)/_TWO_))* PDEL%ONY
!ab>>
  GRID_XY_P(i,j)=XY_NEW_TO_STD_ORIGIN(GRID_INFO%CT_COORD,GRID_XY_C(i,j),P_P,TPI)
!ab<<
  END DO
END DO

! Change XY coordinates in CENTER Origin in STD Origin
!ab GRID_XY_P=UNPACK&
&(XY_NEW_TO_STD_ORIGIN(GRID_INFO%CT_COORD,PACK(GRID_XY_C,M),P_P,TPI),M,DUMMY_XY)
and after the tests
! Compute ouputs datas depending projection type
!ab>>
DO i=_IONE_,NB_PTS%ONX
  DO j=_IONE_,NB_PTS%ONY

!ab GRID_COORD=UNPACK(XY_TO_LATLON(PACK(GRID_XY_P,M),P_P,TPI),M, DUMMY_COORD)
    GRID_COORD(i,j)=XY_TO_LATLON(GRID_XY_P(i,j),P_P,TPI)

!ab GRID_MF=UNPACK(MAP_FACTOR(PACK(GRID_COORD,M), P_P,TPI,RT), M,_ZERO_)
    GRID_MF(i,j)=MAP_FACTOR(GRID_COORD(i,j),P_P,TPI,RT)

!ab GRID_PGN=UNPACK(GN(PACK(GRID_COORD,M),P_P),M,DUMMY_PGN)
    GRID_PGN(i,j)=GN(GRID_COORD(i,j),P_P)

!ab GRID_COORD=UNPACK(ANGLE_DOMAIN(PACK(GRID_COORD,M),TPI,'0+','R'),M,DUMMY_COORD)
    GRID_COORD(i,j)=ANGLE_DOMAIN(GRID_COORD(i,j),TPI,'0+','R')

  ENDDO
ENDDO
!ab<<
```

- corresponding modification was introduced in PALADIN package also.

CONTENTS