Servicio Meteorológico Nacional

Bitácora de Optimización del WRF en Cluster Rayo

Nota Técnica SMN 2018-47

Maximiliano A. Sacco
Departamento de Investigación y Desarrollo
Gerencia de Investigación, Desarrollo y Capacitación. SMN
Marzo 2018

Dorrego 4019 (C1425GBE) Buenos Aires, Argentina
Tel: (+54 011) 5167-6767 - smn@smn.gov.ar - www.smn.gov.ar

Ministerio de Defensa
Presidencia de la Nación

Resumen

Se realizó este trabajo con la intención de evaluar distintas estrategias de compilación y ejecución del modelo WRFV3.7 de manera de lograr el mejor rendimiento posible en el cluster Rayo del Servicio Meteorológico Nacional. Se comparó el desempeño de los compiladores de GNU e Intel y el desempeño de varias librarías MPI (OpenMPI, Mpich e IMPI). También se evaluaron distintos modelos de memoria (distribuida y compartida) y finalmente se estudio la respuesta de distintas estrategias de asignación de CPU y caches para mejorar los tiempos de corrida del modelo y el desempeño de los procesos de E/S.

Abstract

This work was carried out with the intention of evaluating different strategies of compilation and execution of the WRFV3.7 model in order to achieve the best possible performance in the Rayo cluster of the National Meteorological Service. The performance of the GNU and Intel compilers and the performance of several MPI libraries (OpenMPI, Mpich and IMPI) were compared. Different memory models (distributed and shared) were also evaluated and finally the response of different CPU assignment and caching strategies was studied to improve the run times of the model and the performance of the I / O processes.

Palabras clave: WRF, optimización, cluster, MPI, OpenMP

Citar como:

Sacco M. , 2018: Bitácora de optimización de WRF en cluster Rayo. Nota Técnica SMN 2018-47.

Bitácora de Optimización de WRF en cluster Rayo

Maximiliano A. Sacco

Investigación y Desarrollo - Servicio Meteorológico Nacional
msacco@smn.gob.ar

Resumen

Se realizo este trabajo con la intención de evaluar distintas estrategias de compilación y ejecución del modelo WRFV3.7 de manera de lograr el mejor rendimiento posible en el cluster Rayo del Sevicio Meteorológico Nacional. Se comparó el desempeño de los compiladores de GNU e Intel y el desempeño de varias librarías MPI (OpenMPI, Mpich e IMPI). También se evaluaron distintos modelos de memoria (distribuida y compartida) y finalmente se estudio la respuesta de distintas estrategias de asignación de CPU y caches para mejorar los tiempos de corrida del modelo y el desempeño de los procesos de E/S.

1. Introducción

Con el objetivo de lograr mejoras en los tiempos de ejecución del Modelo WRF, se realizara en el presente informe un estudio de casos y se medirán y compararan los tiempos de ejecución. En la sección 2 se evaluara el desempeño de distintos compiladores y librerías para generar un binario eficiente. En la sección 3.2 se evaluaran las distintas librerías de MPI y su desempeño interactuando en escenarios de memoria compartida y distribuida. En la sección 3.1 se analizara la distribución de procesos y threads en los CPU y el uso de la memoria cache. Finalmente en la sección 4 se analizaran estrategias de E/S.

A continuación se muestras las herramientas de hardware y software utilizadas para generar los casos de prueba. En el caso del hardware se corresponde con la configuración actual (2016) del cluster Rayo.

1.1. Hardware

Nodos 2

Memoria 64GB DDR3 @1600Mhz

Red e1000e: peth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: Rx/Tx

Procesadores 2 por nodo

CPU Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz

Cache 15MB

Cores 6 por procesador

Threads 2 por core

Instruction Set AVX

1.2. Software

A continuación se listaran las herramientas de software y sus respectivas versiones utilizadas para compilar el WRF.

- WRF-3.7
  - netcdf-4.3.3

  - netcdff-4.4.2
  - jasper-1.900.1
  - hdf5-1.8.14
  - libpng-1.2.50
  - zlib-1.2.8
- Compiladores
  - GNU (GCC - Fortran) 4.4.7 20120313 (Red Hat 4.4.7-16)
  - INTEL icc-ifort-16.0.0
- Librerías MPI
  - Intel impi-5.1.1
  - mpich-3.0.4
  - openmpi-1.10.2

2. Compiladores y librerías

Los fuentes del modelo y las librerías necesarias fueron compiladas siguiendo las instrucciones detalladas en [**1**]. Las configuraciones de cada compilación se muestran en el Apéndice B En esta sección se compara el desempeño de las librerías de MPI mencionadas en 1 y los binarios generados por los compiladores de *Intel* y *GNU*.

En función a esto, se diseñaron los siguientes casos de prueba, con el fin de evaluar algunas alternativas que incluyeran opciones pagas y gratuitas.

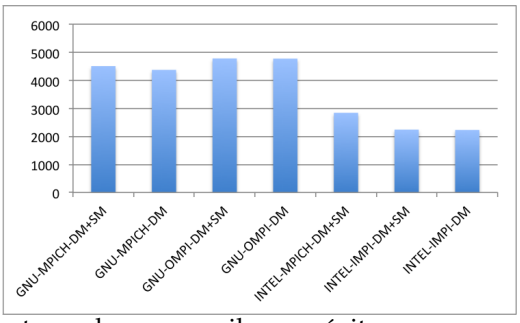
	caso 1	**caso 2**	**caso 3**	**caso 4**	**caso 5**	**caso 6**	**caso 7**	**caso 8**
Compilador	GNU	GNU	GNU	GNU	Intel	Intel	Intel	Intel
Librería MPI	Mpich	Mpich	OMPI	OMPI	OMPI	Mpich	Impi	Impi
Memória	DM+SM	DM	DM+SM	DM	DM+SM	DM+SM	DM+SM	DM

Cuadro 1: *Diseño de los casos de pruebas en función de las herramientas disponibles*

Se ejecutaron todos los casos utilizando un pronóstico modelo de complejidad media del 22 de diciembre del 2015 con una grilla de 420x500 puntos y un pronostico a 6 hs y se procesaron lanzando 24 procesos MPI en cada nodo dando un total de 48 procesos MPI. Si bien cada nodo posee 12 cores con HiperThreading, resulto en todos los casos en una mejora de performance la ejecución de 24 procesos por nodo que su equivalente de 12 procesos por nodo.

A continuación se muestra los resultados de dicha corridas. La primer columna de la tabla 2 muestra el media y la varianza del paso de tiempo utilizado dt . Este paso es variable y cambia según la complejidad encontrada, dependiendo la precision y la representación numérica de los compiladores, cada caso presenta variaciones sutiles. La segunda columna muestra la media y la varianza del tiempo que tardo en procesar un paso de tiempo T_{dt} . Las siguiente columnas son, la cantidad de pasos totales requeridas para procesar las 6hs; el tiempo de procesamiento utilizado para realizar operaciones de E/S; y el tiempo total que tardo en ejecutar la simulación completa. Todos los tiempos estan medidos en segundos.

Caso	dt $\bar{\mu}$ (σ)	T_{dt} $\bar{\mu}$ (σ)	Pasos	E/S $\bar{\mu}$	Tiempo Total
1	21,708 (4,45)	4,529 (1,87)	995	161	4506
2	21,708 (4,45)	4,395 (1,73)	995	145	4373
3	21.708 (4,45)	4,804(1,74)	995	144	4780
4	21,708 (4,45)	4,797 (1,68)	955	137	4773
5	-	-	-	-	-
6	21,906 (4,43)	2,882 (1,85)	986	159	2842
7	21,774(4,51)	2,261 (1,46)	992	124	2242
8	21,752(4,51)	2,246 (1,46)	993	124	2230



Cabe destacar el que Caso 5, luego de muchos intentos se logro compilar con éxito, pero no se pudo obtener una versión capaz de ejecutar el modelo exitosamente, por dicha razón no figuran datos en la tabla de resultados.

2.1. Otras Opciones

Una variante que no se probo y que la bibliografía recomienda evaluar es utilizar pNetCDF sobre un file system distribuido, por ejemplo, lustre FS. Esto parece mejorar la performance de las escrituras a disco por parte de los procesos MPI, permitiendo que estos se hagan en paralelo.

Otro aspecto que debería ser analizado en futuros test es utilizar la última versión disponible del GCC.

2.2. Análisis de los resultados

En general se observa que el compilador *Intel* supera en performance corriendo en un 63 % del tiempo de corrida del *GCC*, por ejemplo al comparar el caso 1 y 6 que utilizan la misma librería MPI y el mismo modelo de memoria. También se puede ver que los desempeños de las distintas librerías MPI no son iguales, pero estos datos no son concluyentes en principio porque cada librería puede asumir distintas configuraciones por defecto que, para cada cluster en particular podría generar mejoras en la performance. Lo que si se puede mencionar es que en el cluster Rayo, con configuraciones por defecto, la librería de Intel supera a las otras. Finalmente, el modelo de memoria resulta levemente mejor el de solo memoria distribuida, pero esto no es un parámetro a tener en cuenta es este momento ya que no se ha ejecutado el modelo utilizando memoria compartida (todos los procesos son MPI).

3. Cores, Threads, Pin & Placement

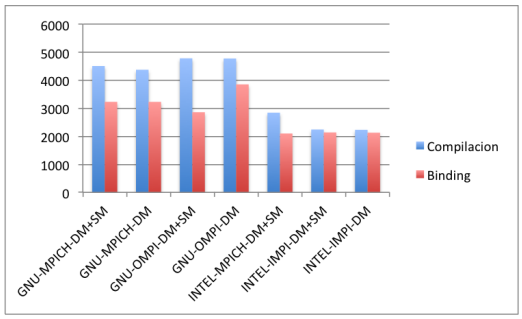
3.1. Asignación de procesos a procesadores

En esta sección estudiaremos como incide en la performance acomodar los procesos en los procesadores y fijarlos para que siempre se ejecuten los mismos procesos en los mismos procesadores. Los sistemas operativos convencionales tienen un administrador de procesos que se encarga de asignar un proceso a un procesador durante un tiempo determinado y luego encolarlo para dar el procesador a otro proceso que este esperando, esta metodología se denomina Time Sharing. En las computadores que varios procesadores ocurre lo mismo, pero en principio no hay garantías de que un procesos vuelva a ejecutar en el mismo procesador.

comando:

```
mpirun –map–by core –bind–to core ./wrf.exe
```

Caso	dt $\bar{\mu}$ (σ)	T_{dt} $\bar{\mu}$ (σ)	Pasos	E/S $\bar{\mu}$	Tiempo Total
1	21,708 (4,45)	3,244 (1,33)	995	115	3228
2	21,708 (4,45)	3,243 (1,24)	995	105	3227
3	21.708 (7,45)	3,877 (1,27)	995	102	3858
4	21.708 (7,45)	3,870 (1,27)	995	100	3851
6	21,906 (4,43)	2,130 (1,40)	986	123	2100
7	21,774 (4,51)	2,154 (1,39)	992	122	2137
8	21,752 (4,50)	2,143 (1,39)	993	122	2128

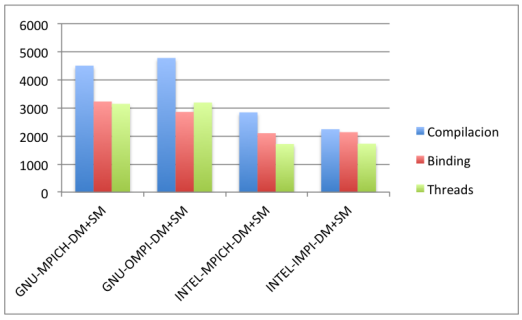


3.2. Overhead MPI

En esta sección analizaremos el impacto de intercomunicar los procesos usando mensajes MPI contra utilizar memoria compartida en caso de se posible. esto es 2 threads , 12 proc por nodo 2 nodos. comando:

```
export OMP_NUM_THREAD = 2
mpirun –map–by core –bind–to core ./wrf.exe
```

Caso	dt $\bar{\mu}$ (σ)	T_{dt} $\bar{\mu}$ (σ)	Pasos	E/S $\bar{\mu}$	Tiempo Total
1	21,708 (4,45)	3,161 (1,19)	995	99	3146
3	21.708 (7,45)	3,208(1,27)	995	104	3192
6	21,840 (4,56)	1,734 (1,25)	994	109	1715
7	21,730(4,46)	1,736 (1,24)	994	109	1726

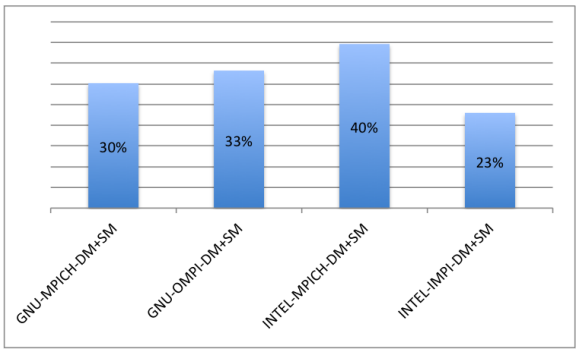


3.3. Otras Opciones

La opción de Hiperthreading esta activa en la BIOS de los nodos del cluster, la bibliografía recomienda desactivarla. Esta es una opción que no pudo probarse en esta versión del informe.

3.4. Análisis de los resultados

En todos los casos se logro mejorar los tiempos de corrida del modelo.



El gráfico de la izquierda muestra una tabla con el porcentaje de mejora en tiempo para cada version de compilador y librería MPI. Se puede ver que en todos los casos hubo una mejora muy importante en los tiempos de ejecución. Como este tipo de optimización es muy dependiente no solo de la topología del cluster sino también del dominio que se esta simulando, debemos interpretar estos números solo como una medida

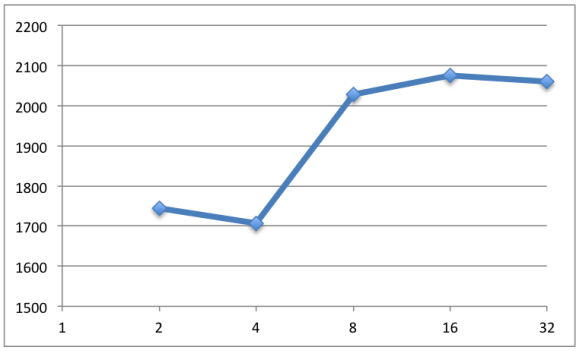
de la ganancia del esfuerzo y tiempo que vale la pena invertir en función de mejorar el tiempo de corrida del modelo.

4. Domino, Entrada y Salida

En esta seccion analizaremos dos parámetros del namelist.input que modifican ciertos comportamientos del modelo que pueden resultar en una mejora del tiempo de ejecución.

4.1. Numtiles

El modelo divide el domino espacial según la cantidad de procesos disponibles y cada porción del dominio se denomina **patch**.

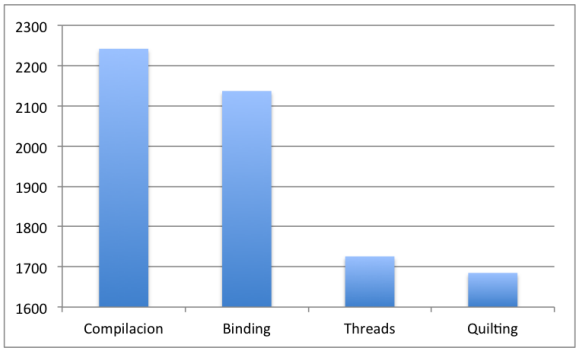


El parámetro **numtiles** del *namelist.input* nos permite subdividir cada patch en tiles con el fin de lograr ajustar la porción de datos que serán procesados a una cantidad que pueda ser contenida en la cache.

Para evaluar la incidencia en los tiempos de ejecución de este parámetro, se realizaron sucesivas corridas de un único caso (el caso 7 - Intel) con distintos valores para el parámetro **numtiles** como se ve en la figura. En este caso el mejor tiempo de corrida se logró para un $numtiles = 4$ logrando un tiempo de corrida de $1707seg$ representando una mejora en la performance de un 2 % aprox.

4.2. Quilting

En esta sección evaluaremos las mejoras en performance de liberar a los procesos de las tareas de E/S para que dediquen su tiempo a procesamiento numérico y asignar unos pocos procesos exclusivos a E/S. Por supuesto, cuantos mas procesos tengamos para realizar E/S menos tendremos para realizar computo, razón por lo cual debemos encontrar la proporción óptima. La configuración óptima dependerá de la configuración del cluster y del domino que se desea modelar.



En este caso el mejor tiempo de corrida se logró para un $nio_tasks_per_group = 24; nio_groups = 2$, logrando un tiempo de corrida de $1685seg$ representando una mejora en la performance de un 3 % aprox.

5. Conclusiones

La conclusión final que se obtiene del presente trabajo es que vale la pena dedicar horas de esfuerzo a optimizar la corrida del WRF en un cluster sobre todo en condiciones operativos donde el domino de simulación permanece fijo. Ya que el ajuste fino especializado para cada configuración de cluster impacta de manera notoria en los tiempos de ejecución del modelo.

Referencias

[1] Manual de compilación WRF
`http://www2.mmm.ucar.edu/wrf/OnLineTutorial/compilation_tutorial.php`

[2] Optimización WRF con INTEL
`https://software.intel.com/en-us/articles/performance-hints-for-wrf-on-intel-architecture`

[3] Optimización WRF con INTEL 2
`https://software.intel.com/en-us/articles/performance-tools-for-software-developers-building-wrf`

[4] Optimización WRF con INTEL MKL
`http://www.dtcenter.org/HurrWRF/users/tutorial/2014_tutorial/tutorialpractical2014/compile_exercise.htm`

[5] T. Langkamp and J. Böhner
Influence of the compiler on multi-CPU performance of WRFv3- Institute of Geography, University of Hamburg, Germany Geosci. Model Dev., 4, 611?623, 2011 www.geosci-model-dev.net/4/611/2011/ doi:10.5194/gmd-4-611-2011

Apéndice A namelist.input

```
&time_control
run_days                            = 00,
run_hours                           = 06,
run_minutes                         = 00,
run_seconds                         = 00,
start_year                          = 2015,    2016,    2016,
start_month                         = 12,    04,    04,
start_day                           = 22,      07,      07,
start_hour                          = 12,      00,      00,
start_minute                        = 00,    00,    00,
start_second                        = 00,  00,  00,
end_year                            = 2015,    2016,    2016,
end_month                           = 12,    04,    04,
end_day                             = 22,      07,      07,
end_hour                            = 18,      12,      12,
end_minute                          = 00,    00,    00,
end_second                          = 00,  00,  00,
interval_seconds                    = 10800
input_from_file                     = .true.,.true.,.true.,
history_interval                    = 60,  10,   60,
frames_per_outfile                  = 1, 1000, 1000,
restart                             = .false.,
restart_interval                    = 5000,
io_form_history                     = 2
io_form_restart                     = 2
io_form_input                       = 2
io_form_boundary                    = 2
debug_level                         = 50
/
&domains
numtiles                            = 1
use_adaptive_time_step              = .true.
step_to_output_time                 = .true.
target_cfl                          = 1.2
max_step_increase_pct               = 5
starting_time_step                  = 20
max_time_step                       = 50
min_time_step                       = 10
smooth_cg_topo                      = .false.
time_step                           = 24,
time_step_fract_num                 = 0,
time_step_fract_den                 = 1,
max_dom                             = 1,
e_we                                = 420,    250,   94,
e_sn                                = 500,    271,    91,
e_vert                              = 38,    38,    28,
p_top_requested                     = 5000,
num_metgrid_levels                  = 27,
num_metgrid_soil_levels             = 4,
dx                                  = 4000, 8000,  3333.33,
dy                                  = 4000, 8000,  3333.33,
grid_id                             = 1,     2,     3,
parent_id                           = 0,     1,     2,
i_parent_start                      = 1,     55,    30,
j_parent_start                      = 1,     154,    30,
parent_grid_ratio                   = 1,     3,     3,
parent_time_step_ratio              = 1,     3,     3,
feedback                            = 0,
smooth_option                       = 0
/

&physics
mp_physics                          = 6,     3,     3,
ra_lw_physics                       = 1,     1,     1,
ra_sw_physics                       = 1,     1,     1,
radt                                = 10,    30,    30,
sf_sfclay_physics                   = 1,     1,     1,
sf_surface_physics                  = 2,     2,     2,
bl_pbl_physics                      = 1,     1,     1,
bldt                                = 0,     0,     0,
```

```
cu_physics                          = 0,     1,     0,
cudt                                = 5,     5,     5,
isfflx                              = 1,
ifsnow                              = 1,
icloud                              = 1,
surface_input_source                = 1,
num_soil_layers                     = 4,
sf_urban_physics                    = 0,     0,     0,
/

&fdda
/

&dynamics
w_damping                           = 0,
diff_opt                            = 1,
km_opt                              = 4,
diff_6th_opt                        = 0,      0,      0,
diff_6th_factor                     = 0.12,   0.12,   0.12,
base_temp                           = 290.
damp_opt                            = 3,
zdamp                               = 5000.,  5000.,  5000.,
dampcoef                            = 0.1,    0.2,    0.2
khdif                               = 0,      0,      0,
kvdif                               = 0,      0,      0,
non_hydrostatic                     = .true., .true., .true.,
moist_adv_opt                       = 1,      1,      1,
scalar_adv_opt                      = 1,      1,      1,
/

&bdy_control
spec_bdy_width                      = 5,
spec_zone                           = 1,
relax_zone                          = 4,
specified                           = .true., .false.,.false.,
nested                              = .false., .true., .true.,
/

&grib2
/

&namelist_quilt
nio_tasks_per_group = 0,
nio_groups = 1,
/
```

Apéndice B Compilación

B.1 Caso de prueba 1 (GNU-MPICH-DM+SM)

```
DESCRIPTION     =       GNU ($SFC/$SCC)
DMPARALLEL      =        1
OMPCPP          =        -D_OPENMP
OMP             =        -fopenmp
OMPCC           =        -fopenmp
SFC             =       gfortran
SCC             =       gcc
CCOMP           =       gcc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT -DMPI2_THREAD_SUPPORT
FC              =       time $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR -DWRF_USE_CLM -DNO_IEEE_MODULE
CFLAGS_LOCAL    =       -w -O3 -c
LDFLAGS_LOCAL   =
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O2 -ftree-vectorize -funroll-loops -ftree-loop-linear
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       # -g $(FCNOOPT) # -ggdb -fbacktrace -fbounds-check -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-none
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.1.1 Optimizacion

Según recomendaciones encontradas en [5] se intentaron los siguientes cambios:

Nivel de optimizacion O3: No se pudo lograr compilar el código.

Opcion -ftree-loop-linear: Compilo correctamente, no se observo mejora alguna en el tiempo de ejecución.

B.2 Caso de prueba 2 (GNU-MPICH-DM)

```
DESCRIPTION     =       GNU ($SFC/$SCC)
DMPARALLEL      =        1
OMPCPP          =       # -D_OPENMP
OMP             =       # -fopenmp
OMPCC           =       # -fopenmp
SFC             =       gfortran
SCC             =       gcc
CCOMP           =       gcc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC              =       time $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR -DWRF_USE_CLM -DNO_IEEE_MODULE
CFLAGS_LOCAL    =       -w -O3 -c
LDFLAGS_LOCAL   =
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O2 -ftree-vectorize -funroll-loops
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       # -g $(FCNOOPT) # -ggdb -fbacktrace -fbounds-check -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-none
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.3 Caso de prueba 3 (GNU-OMPI-DM+SM)

La simulación se corrió completa (4570 pasos), y se obtuvieron los siguientes valores:

```
DESCRIPTION     =       GNU ($SFC/$SCC)
DMPARALLEL      =        1
OMPCPP          =       -D_OPENMP
OMP             =       -fopenmp
OMPCC           =       -fopenmp
SFC             =       gfortran
SCC             =       gcc
CCOMP           =       gcc
DM_FC           =       mpif90
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT -DMPI2_THREAD_SUPPORT
FC              =       time $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR -DWRF_USE_CLM -DNO_IEEE_MODULE
CFLAGS_LOCAL    =       -w -O3 -c
LDFLAGS_LOCAL   =
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O2 -ftree-vectorize -funroll-loops
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       # -g $(FCNOOPT) # -ggdb -fbacktrace -fbounds-check -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-none
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.4 Caso de prueba 4 (GNU-OMPI-DM)

```
DESCRIPTION     =       GNU ($SFC/$SCC)
DMPARALLEL      =        1
OMPCPP          =       # -D_OPENMP
OMP             =       # -fopenmp
OMPCC           =       # -fopenmp
SFC             =       gfortran
SCC             =       gcc
CCOMP           =       gcc
DM_FC           =       mpif90
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC              =       time $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       #-fdefault-real-8
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_SUBR -DWRF_USE_CLM -DNO_IEEE_MODULE
CFLAGS_LOCAL    =       -w -O3 -c
LDFLAGS_LOCAL   =
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O2 -ftree-vectorize -funroll-loops
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0
FCDEBUG         =       # -g $(FCNOOPT) # -ggdb -fbacktrace -fbounds-check -ffpe-trap=invalid,zero,overflow
FORMAT_FIXED    =       -ffixed-form
FORMAT_FREE     =       -ffree-form -ffree-line-length-none
FCSUFFIX        =
BYTESWAPIO      =       -fconvert=big-endian -frecord-marker=4
FCBASEOPTS_NO_G =       -w $(FORMAT_FREE) $(BYTESWAPIO)
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.5 Caso de prueba 5 (INTEL-OMPI-DM+SM)

```
DESCRIPTION     =       INTEL ($SFC/$SCC): Xeon (SNB with AVX mods)
DMPARALLEL      =        1
OMPCPP          =       -D_OPENMP
OMP             =       -openmp -fpp -auto
OMPCC           =       -openmp -fpp -auto
SFC             =       ifort
SCC             =       icc
CCOMP           =       icc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT -DMPI2_THREAD_SUPPORT
FC              =       $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       -real-size `expr 8 \* $(RWORDSIZE)` -i4
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_FUNC -DCHUNK=64 -DXEON_OPTIMIZED_WSM5 -DOPTIMIZE_CFL_TEST -DWRF_USE_CLM
OPTNOSIMD       =
OPTAVX          =       -xAVX
CFLAGS_LOCAL    =       -w -O3 $(OPTAVX)
LDFLAGS_LOCAL   =       $(OPTAVX)
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 $(OPTAVX)
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0 -fno-inline -no-ip
FCDEBUG         =       # -g $(FCNOOPT) -traceback # -fpe0 -check all -ftrapuv -unroll0 -u
FORMAT_FIXED    =       -FI
FORMAT_FREE     =       -FR
FCSUFFIX        =
BYTESWAPIO      =       -convert big_endian
FCBASEOPTS_NO_G =       -w $(OMP) -auto -ftz -fno-alias -fp-model fast=1 -no-prec-div -no-prec-sqrt $(FORMAT_FREE) $(BYTESWA
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.6 Caso de prueba 6 (INTEL-MPICH-DM+SM)

```
DESCRIPTION     =       INTEL ($SFC/$SCC): Xeon (SNB with AVX mods)
DMPARALLEL      =        1
```

```
OMPCPP          =       -D_OPENMP
OMP             =       -openmp -fpp -auto
OMPCC           =       -openmp -fpp -auto
SFC             =       ifort
SCC             =       icc
CCOMP           =       icc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT -DMPI2_THREAD_SUPPORT
FC              =       $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       -real-size `expr 8 \* $(RWORDSIZE)` -i4
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_FUNC -DCHUNK=64 -DXEON_OPTIMIZED_WSM5 -DOPTIMIZE_CFL_TEST -DWRF_USE_CLM
OPTNOSIMD       =
OPTAVX          =       -xAVX
CFLAGS_LOCAL    =       -w -O3 $(OPTAVX)
LDFLAGS_LOCAL   =       $(OPTAVX)
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 $(OPTAVX)
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0 -fno-inline -no-ip
FCDEBUG         =       # -g $(FCNOOPT) -traceback # -fpe0 -check all -ftrapuv -unroll0 -u
FORMAT_FIXED    =       -FI
FORMAT_FREE     =       -FR
FCSUFFIX        =
BYTESWAPIO      =       -convert big_endian
FCBASEOPTS_NO_G =       -w $(OMP) -auto -ftz -fno-alias -fp-model fast=1 -no-prec-div -no-prec-sqrt $(FORMAT_FREE) $(BYTESWA
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.7 Caso de prueba 7 (INTEL-IMPI-DM+SM)

```
DESCRIPTION     =       INTEL ($SFC/$SCC): Xeon (SNB with AVX mods)
DMPARALLEL      =       1
OMPCPP          =       -D_OPENMP
OMP             =       -openmp -fpp -auto
OMPCC           =       -openmp -fpp -auto
SFC             =       ifort
SCC             =       icc
CCOMP           =       icc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT -DMPI2_THREAD_SUPPORT
FC              =       $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       -real-size `expr 8 \* $(RWORDSIZE)` -i4
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_FUNC -DCHUNK=64 -DXEON_OPTIMIZED_WSM5 -DOPTIMIZE_CFL_TEST -DWRF_USE_CLM
OPTNOSIMD       =
OPTAVX          =       -xAVX
CFLAGS_LOCAL    =       -w -O3 $(OPTAVX)
LDFLAGS_LOCAL   =       $(OPTAVX)
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 $(OPTAVX)
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0 -fno-inline -no-ip
FCDEBUG         =       # -g $(FCNOOPT) -traceback # -fpe0 -check all -ftrapuv -unroll0 -u
FORMAT_FIXED    =       -FI
FORMAT_FREE     =       -FR
FCSUFFIX        =
BYTESWAPIO      =       -convert big_endian
FCBASEOPTS_NO_G =       -w $(OMP) -auto -ftz -fno-alias -fp-model fast=1 -no-prec-div -no-prec-sqrt $(FORMAT_FREE) $(BYTESWA
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

B.8 Caso de prueba 8 (INTEL-IMPI-DM)

```
DESCRIPTION     =       INTEL ($SFC/$SCC): Xeon (SNB with AVX mods)
DMPARALLEL      =       1
```

```
OMPCPP          =       # -D_OPENMP
OMP             =       # -openmp -fpp -auto
OMPCC           =       # -openmp -fpp -auto
SFC             =       ifort
SCC             =       icc
CCOMP           =       icc
DM_FC           =       mpif90 -f90=$(SFC)
DM_CC           =       mpicc -cc=$(SCC) -DMPI2_SUPPORT
FC              =       $(DM_FC)
CC              =       $(DM_CC) -DFSEEKO64_OK
LD              =       $(FC)
RWORDSIZE       =       $(NATIVE_RWORDSIZE)
PROMOTION       =       -real-size 'expr 8 \* $(RWORDSIZE)' -i4
ARCH_LOCAL      =       -DNONSTANDARD_SYSTEM_FUNC -DCHUNK=64 -DXEON_OPTIMIZED_WSM5 -DOPTIMIZE_CFL_TEST -DWRF_USE_CLM
OPTNOSIMD       =
OPTAVX          =       -xAVX
CFLAGS_LOCAL    =       -w -O3 $(OPTAVX)
LDFLAGS_LOCAL   =       $(OPTAVX)
CPLUSPLUSLIB    =
ESMF_LDFLAG     =       $(CPLUSPLUSLIB)
FCOPTIM         =       -O3 $(OPTAVX)
FCREDUCEDOPT    =       $(FCOPTIM)
FCNOOPT         =       -O0 -fno-inline -no-ip
FCDEBUG         =       # -g $(FCNOOPT) -traceback # -fpe0 -check all -ftrapuv -unroll0 -u
FORMAT_FIXED    =       -FI
FORMAT_FREE     =       -FR
FCSUFFIX        =
BYTESWAPIO      =       -convert big_endian
FCBASEOPTS_NO_G =       -w $(OMP) -auto -ftz -fno-alias -fp-model fast=1 -no-prec-div -no-prec-sqrt $(FORMAT_FREE) $(BYTESWA
FCBASEOPTS      =       $(FCBASEOPTS_NO_G) $(FCDEBUG)
```

Instrucciones para publicar Notas Técnicas

En el SMN existieron y existen una importante cantidad de publicaciones periódicas dedicadas a informar a usuarios distintos aspectos de las actividades del servicio, en general asociados con observaciones o pronósticos meteorológicos.

Existe no obstante abundante material escrito de carácter técnico que no tiene un vehículo de comunicación adecuado ya que no se acomoda a las publicaciones arriba mencionadas ni es apropiado para revistas científicas. Este material, sin embargo, es fundamental para plasmar las actividades y desarrollos de la institución y que esta dé cuenta de su producción técnica. Es importante que las actividades de la institución puedan ser comprendidas con solo acercarse a sus diferentes publicaciones y la longitud de los documentos no debe ser un limitante.

Los interesados en transformar sus trabajos en Notas Técnicas pueden comunicarse con Ramón de Elía (rdelia@smn.gov.ar), Luciano Vidal (lvidal@smn.gov.ar) o Martin Rugna (mrugna@smn.gov.ar) de la Gerencia de Investigación, Desarrollo y Capacitación, para obtener la plantilla WORD que sirve de modelo para la escritura de la Nota Técnica. Una vez armado el documento deben enviarlo en formato PDF a los correos antes mencionados. Antes del envío final los autores deben informarse del número de serie que le corresponde a su trabajo e incluirlo en la portada.

La versión digital de la Nota Técnica quedará publicada en el Repositorio Digital del Servicio Meteorológico Nacional. Cualquier consulta o duda al respecto, comunicarse con Melisa Acevedo (macevedo@smn.gov.ar).