Numerical weather prediction in high-performance computing (HPC) environments

Implementation of the WRF model on ARIS

Theodore M. Giannaros
Post-doc Researcher
National Observatory of Athens, IERSD
Email: thgian@noa.gr
Web: http://theodoregiannaros.eu
Parallelism in WRF

- Distributed memory (DM) - “MPI”
- Shared-memory (SM) - “OpenMP”
- Clusters of SM processors ("hybrid MPI+OpenMP")

50+ compilation options: Serial, DM, SM, Hybrid (DM+SM), numerous compilers and architectures
Domain decomposition

DM works in “patches”: MPI processes
SM works in “tiles”: Threads in each MPI process
Example
2 nodes on ARIS, each with 20 CPUs; 40 CPUs in total

So, what are my options?
40 MPI processes, 1 thread per each (pure DM)
OMP_NUM_THREADS=1; mpirun -np 40 ./wrf.exe
OR
20 MPI processes, 2 threads per each (hybrid DM+SM)
OMP_NUM_THREADS=2; mpirun -np 20 ./wrf.exe
OR
10 MPI processes, 4 threads per each (hybrid DM+SM)
OMP_NUM_THREADS=4; mpirun -np 10 ./wrf.exe
OR
and so on …

Experience with WRF has shown that hybrid DM+SM does not always have a positive effect on computational performance
Better to use “pure MPI”
Getting started: Read, think, design

Define your objectives
What are your **scientific** and/or **practical objectives**? Why do you need to run WRF? How will you know that your simulations are successful?

Get to know your problem
Review literature! What are the **atmospheric processes** involved? Which are the most **important** (clouds, radiation, convection, etc.)? What is known? Is anything **missing**? Judge the **efficacy** of your “simulations-to-do”.

Determine available observational datasets
What **observations** are available? Again, become familiar with the **processes** that you want to study. How will the observations be used for **verifying** and/or **complementing** your simulations? Judge the **adequacy** of your “simulations-to-do”.

Prepare your strategy
Are you going to focus on a **case study**? If yes, **which** one and **why**? Are there adequate **observations** for verifying your “simulations-to-do”? Will you set up an **operational** weather forecasting service? What are the practical **requirements**?
Model configuration: Domains (1)

Consider first
• Target horizontal grid spacing
• Resolution of initialization data
Most often, you will need to adopt a nesting strategy.

Hints
• Place domain boundaries away from each other, and away from steep topography
• Odd parent-child ratios are preferred (e.g. 3:1, 5:1)
• Higher horizontal resolution will also require higher vertical resolution
• Use at least 30-35 vertical levels; larger density closer to the ground and to the model top
• Lambert: mid-latitudes, Mercator: low-latitudes, Lat-Lon: global, Rotated lat-lon: regional
• Start inside-out (first the nest, move up)

Do remember!
Avoid the “grey zone” (4-10 km)
Static (input) data
Does land data represent your area adequately well? If not, consider using alternative datasets (land use, topography)
May have profound impact on your results!

Real-world example (200 m domain for Rio de Janeiro)

Default USGS (30 s)  SRTM (9s)
Dynamic (input) data
Ask yourself: how good are the data used for initializing WRF?

Real-world example
Wind forecasts for the Guanabara bay in Rio de Janeiro, Brazil, verified against observed data
**GFS**: Forecasts driven by 0.5deg, 6h NCEP/GFS
**ECMWF**: Forecasts driven by 0.5deg, 6h ECMWF/IFS

<table>
<thead>
<tr>
<th>Location</th>
<th>GFS</th>
<th>ECMWF</th>
<th>GFS</th>
<th>ECMWF</th>
<th>GFS</th>
<th>ECMWF</th>
<th>GFS</th>
<th>ECMWF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBRJ</td>
<td>-6.4</td>
<td>25.5</td>
<td>91.8</td>
<td>90.9</td>
<td>65.8</td>
<td>62.3</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>RJ1</td>
<td>-25.8</td>
<td>-8.4</td>
<td>82.3</td>
<td>70.5</td>
<td>66.4</td>
<td>56.2</td>
<td>16</td>
<td>21</td>
</tr>
<tr>
<td>RJ2</td>
<td>-3.3</td>
<td>-9.0</td>
<td>94.9</td>
<td>84.9</td>
<td>74.0</td>
<td>62.9</td>
<td>21</td>
<td>28</td>
</tr>
<tr>
<td>RJ3</td>
<td>-7.2</td>
<td>-8.0</td>
<td>83.1</td>
<td>74.8</td>
<td>72.8</td>
<td>60.9</td>
<td>2</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 5. Same as Table 4 but for wind direction verification statistics.
From a computational point of view

- Assuming a 3:1 parent-child ratio, the nest will require 3x as many time steps to keep pace with the parent grid.
- Rule of thumb: a nested WRF simulation costs ~4x the cost of a single parent domain simulation.
- Coarse domains are not a “headache”: doubling their grid points will result to ~25% increase in nested domain simulation time.

Estimating (roughly) the cost (3:1 ratio example)

1. If the fine and the coarse grid have the same dimensions (# of grid points), then the required CPU for integrating a single time step will be about the same for both domains.
2. Given that the fine grid time step is 1/3 of the coarse grid time step, it is deduced that the nest will require 3x the CPU to catch up with the coarse domain.
Too many options! Where to start from?

- Back to basics: Which processes are important? **Review literature.** What others did?
- Consider first well documented (**tried**) schemes

**Hints**

- Convective schemes are generally not required at \( dx < 4 \text{ km} \)
- Sophisticated microphysics schemes (double-moment, detailed species) may not be necessary at \( dx >> 10 \text{ km} \)
- Try to have **consistent physics** between the domains or use 1-way nesting
- If your simulation spans more than **5 days**, you could start thinking to adopt the **SST update** option
Garbage in, garbage out
NWP is a problem of initial conditions! Common “problematic” variables:
• Soil moisture and temperature
• Sea-surface temperature
• Bad representation of land/sea mask

Double-check your initial conditions (wrfinput_d0*)!

Let the model warm-up
• Allow for a reasonable spin-up period to avoid “noise” in certain fields (e.g. pressure).
• Spin-up is of great importance for convection, particularly deep convection.
• No rules of thumb; Trial and error process to identify the “ideal” spin-up period
• Computationally costly, but desired!
“Stability” versus “efficiency”

Recommended (maximum) integration time step (s) equals $6 \times dx$ (km)

Most often, this needs to be **downscaled** to avoid numerical instability (CFL violation)

**Example**

1-way nested, **15 km** coarse grid (**CG**) and **5 km** fine grid (**FG**)

- Ideally: CG dt=$6 \times 15=90$ s, FG dt=$90/3=30$ s (parent dt divided by 3:1 ratio)

Result: Model “blows up” quickly after the beginning of the simulation

- Reduce time step: CG dt=60 s, FG=60/3=20 s

Result: Model becomes numerically **steady**; but also $90/60=1.5 \times$ more expensive

- Reduce time step only for CG: CG dt=60 s, FG=60/2 (parent dt divided by 2:1 time step ratio)

Result: Model becomes numerically **steady**; **save** computational time

**Remember**

*You can reduce the CG time step without reducing model performance, as long as you are able to tweak the FG time step (adjust parent-child time step ratio; trial and error)*
Model configuration: Tackling CFL errors

Model “blows up” with CFL errors

Troubleshooting:
Check “where” the model becomes unstable: (a) which vertical level, (b) which i,j in model domain

A. If CFL violation occurs at the **first few vertical levels**, then it’s probably due to steep **orography**: (i) check i,j to verify (even approximately) whether the instability is over complex terrain; if that is the case, consider smoothing orography (GEOGRID.TBL; **smooth option**: 1-2-1)

B. If CFV violation occurs at **upper vertical levels**, then the available options you have are: (i) use the damping option for vertical velocities (**w_damping=1**), (ii) use a different damping option (**damp_opt=1,2,3**), (iii) **reduce** your integration time step, (iv) consider restructuring your **eta_levels** (if you defined them explicitly)
I/O optimization
I/O optimization can be a “bottleneck” for improving WRF performance. On some occasions, I/O takes more time compared to integration!

Good to remember
Output data quickly
Output small data
Output less data

Hints
• Use runtime i/o to reduce output variables (iofields_filename=“varsout.txt”). This will even allow you to cut your file sizes down to half!
• Consider your experiment. Do you need to output data every 1 h or less?
• Use parallel netCDF during compilation (not tested on ARIS)
• Use option to output 1 file per MPI process (io_form_history=102). Reported to save a lot time, but you need to manually join files at the end. Officially unsupported.
Definitions

**Performance**: Model speed ignoring I/O and initialization costs, measured directly as the average cost per model time step over a representative integration period. Can be expressed as either simulation speed or floating-point rate.

**Simulation speed**: Measure of the actual time-to-solution. Expresses the ratio of model time simulated to the actual time, and is computed as the ratio of model time step to the average time per time step, over a representative integration period.

**Scaling**: The ratio of increase in simulation speed to the increase in parallel processes.

**ARIS benchmarking tests**

- **Case A**: Single domain, 235x175x40, 24 km (Europe)
- **Case B**: Case A & 685x235x40, 6 km (Mediterranean)
- **Case C**: Case B & 538x499, 2 km (Greece)
  - 60 h numerical simulations
  - Benchmark period: T0+13 - T0+60 (48 hours)
  - Same physics for all cases and domains
Case C

<table>
<thead>
<tr>
<th>Number of cores</th>
<th>Time-to-solution (hh:mm:ss)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>26:34:53</td>
</tr>
<tr>
<td>24</td>
<td>15:59:33</td>
</tr>
<tr>
<td>48</td>
<td>08:47:03</td>
</tr>
<tr>
<td>96</td>
<td>04:50:50</td>
</tr>
<tr>
<td>192</td>
<td>02:49:45</td>
</tr>
<tr>
<td>384</td>
<td>01:42:29</td>
</tr>
<tr>
<td>768</td>
<td>01:08:09</td>
</tr>
<tr>
<td>1536</td>
<td>00:54:57</td>
</tr>
</tbody>
</table>

![Graph showing simulation speed and scaling for Case C]
Definitions

nproc_x: number of processors to use for decomposition in x-direction
nproc_y: number of processors to use for decomposition in y-direction

By default, WRF will use the square root of processors for deriving values for nproc_x and nproc_y. If this is not possible, some close values will be used.

Hint

WRF responds better to a more rectangular decomposition, i.e. nproc_x<<nproc_y:
• Longer inner loops for better vector and registry reuse
• Better cache blocking
• More efficient halo exchange communication pattern

Best combination defined by trial and error!

Take-away for MPI

• As the number of MPI tasks increases, the amount of work inside each MPI task decreases
• More MPI tasks, more contention for due to communications is likely
• As the computation time gets smaller compared to the communications time, parallel efficiency suffers
Thank you for your attention!
Questions? Comments?

Theodore M. Giannaros
Post-doc Researcher
National Observatory of Athens, IERSD
Email: thgian@noa.gr
Web: http://theodoregiannaros.eu