



# WRF Computation

Dave Gill

## Parallelism in WRF

Why is it required  
What is "scaling"

## OpenMP and MPI

## Halos

## Domain Decomposition

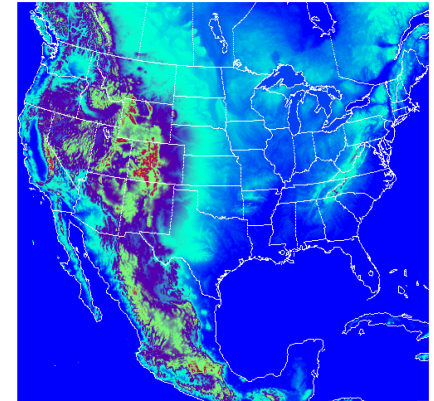
Reasonable, max, min, bad  
Dimensions and decomposition  
Coarse grid vs nest grid

## Where is all of this information

Using rsl files

## WRF v4, 1500x1500x50, 3 km, CONUS Suite

- 50 vertical levels
- No cumulus
- Hybrid vertical activated
- Moist theta
- 18 s dt
- 6 minute simulation
- 10 hour spin-up, then restart
- No I/O included in timing
- Single radiation time step
- 18 non-radiation time steps

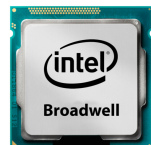


## Parallelism:

Execution of  
processes are  
carried out  
simultaneously

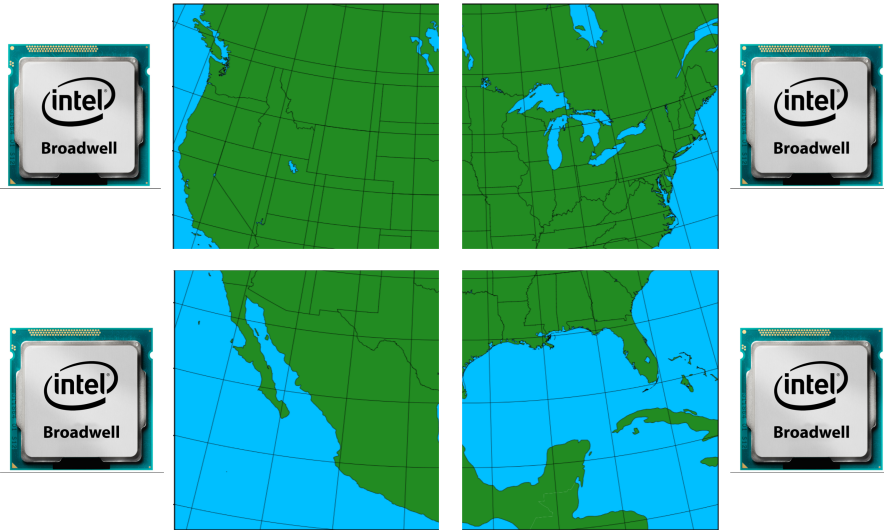


Without  
parallelism, all  
of the work is  
handled by a  
single  
processor



## Why is Parallelism Required

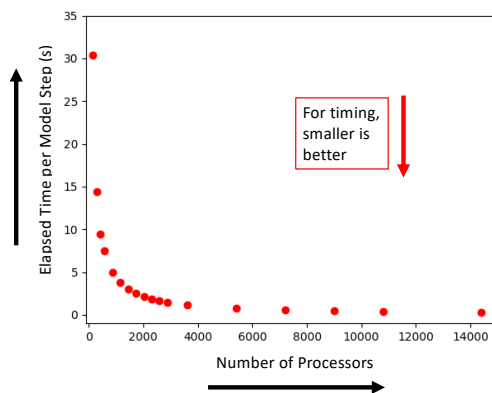
- When we **decompose** the WRF domain into smaller pieces, we distribute work to additional processors
- Since **less work** is performed by each individual processor, the elapsed time to complete the computational task for each processor is reduced
- The aggregated parallel job completes in **less elapsed time**



## What is “scaling”

- The ability of the WRF model to handle these additional processors is a **measure of how well** the WRF model scales
- On a supercomputer, the **most efficient** usage of additional processors tends to be by using **full or nearly full nodes**
- With perfect scaling
  - Twice as many nodes = 2x faster = 1/2 of the original time
  - 3x as many nodes = 3x faster = 1/3 of the original time
  - $nx$  as many nodes =  $nx$  faster =  $1/n$  of the original time
- Raw scaling plots look like hyperbolas

## What is “scaling”

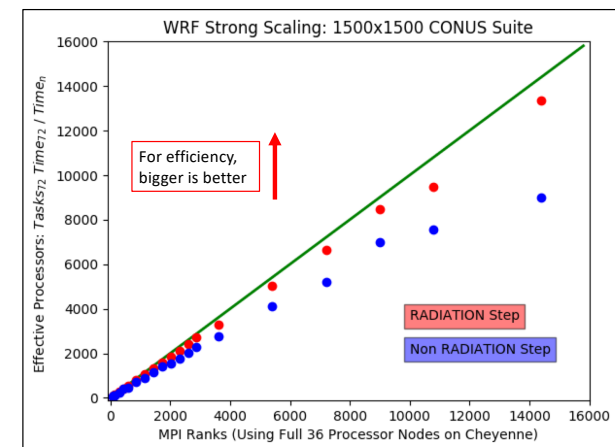


- While the information is accurate, this presentation of the timing data is not easy to interpret
- Scaling plots are typically recast with an easy comparison against a known value (100% efficient) or a fixed quantity (such as Days of Simulation per Elapsed Day of Computation)

## WRF v4, 1500x1500x50, 3 km, CONUS Suite

Base scaling  
using 72  
processors

2 full nodes  
on cheyenne





## WRF v4, 1500x1500x50, 3 km, CONUS Suite

MPI Ranks	I-Dim	J-Dim	Radiation (s)	Non Rad (s)	Days / Day	Scaling Rad	Scaling NonRad
72	188	167	55.30796	8.63828	1.35284964	1	1
108	167	125	37.42179	5.93252	1.98206299	0.98530758	0.970726324
144	125	125	30.35208	6.99176	1.92971713	0.91110659	0.617747177
288	94	84	14.4129	2.56148	4.80432774	0.95934822	0.843094617
432	84	63	9.47383	1.53414	7.73159676	0.97299543	0.938449772
576	63	63	7.45682	1.33647	9.23785158	0.92713717	0.807938076
864	56	47	4.9866	0.886827	13.8802748	0.92427639	0.81172164
1152	47	42	3.75109	0.686984	18.1196878	0.92153147	0.785888027
1440	42	38	2.98979	0.53509	23.0603669	0.92494724	0.807180101
1728	42	32	2.5313	0.442302	27.6412012	0.91040111	0.813761487
2016	36	32	2.15486	0.398824	31.3355417	0.91666479	0.773549235
2304	32	32	1.87563	0.352528	35.6549881	0.92148971	0.765744139
2592	32	28	1.65853	0.307171	40.6957251	0.92632164	0.781168216
2880	32	25	1.46797	0.269931	46.1852403	0.9419123	0.800045197
3600	25	25	1.20596	0.225149	55.687883	0.91724369	0.767338962
5400	21	20	0.79517	0.150944	83.5784193	0.9273985	0.763045014
7200	19	17	0.59816	0.119898	107.319039	0.92463488	0.720469065
9000	17	15	0.47138	0.0891089	141.358285	0.93865603	0.775525677
10800	15	14	0.41968	0.0825039	154.876666	0.87857352	0.698009831
14400	13	13	0.29857	0.06929	195.189659	0.92621429	0.623342474

## What is “scaling”

- We have a **fixed-sized problem** that we want to solve: WRF model with a particular set of parameters
- Look at the measure of the effectiveness of **increasing the number of processors** on this fixed-size problem
- Referred to as **strong scaling**
- Basically, if I use more processors, does this model finish in less time

## OpenMP (Shared) and MPI (Distributed)

- Modern supercomputers have large processor counts

### • Cheyenne:

- 145,152 processor cores
- 4,032 nodes
- 36 cores/node

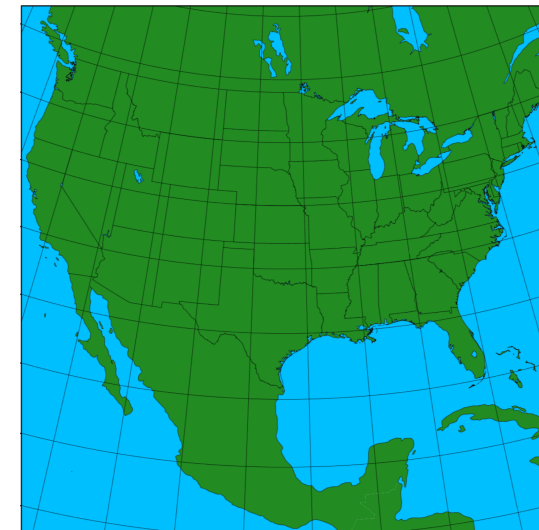
An SGI ICE XA Cluster, the Cheyenne supercomputer features 145,152 Intel Xeon processor cores in 4,032 dual-socket nodes (36 cores/node) and 313 TB of total memory.

Cheyenne's login nodes give users access to the [GLADE](#) shared-disk resource and the [High Performance Storage System](#) (HPSS).



### • Two types of parallelism

- Between nodes (MPI) – distributed memory
- Within nodes (OpenMP) – shared memory



With **shared memory** processing, the domain is split among the run-time available OpenMP threads

For WRF, for **small processor counts**, this is effective

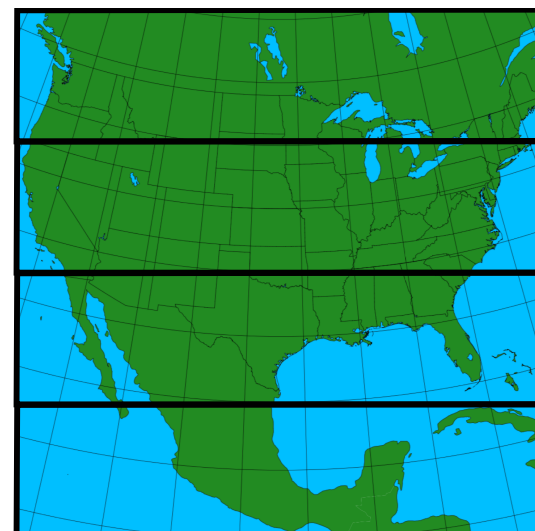
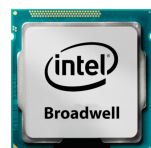
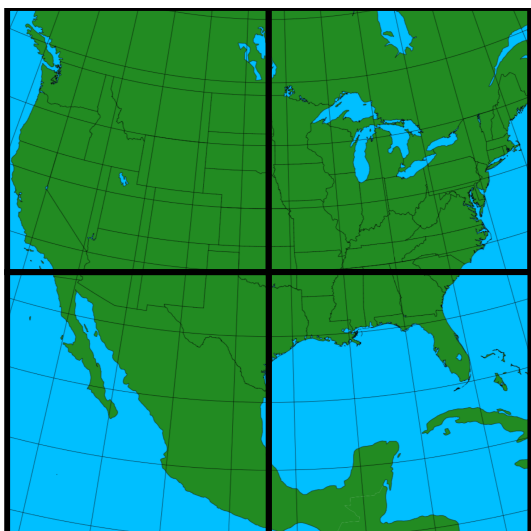
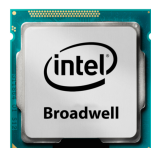


With larger WRF jobs, the number of OpenMP threads is not sufficient to run the model

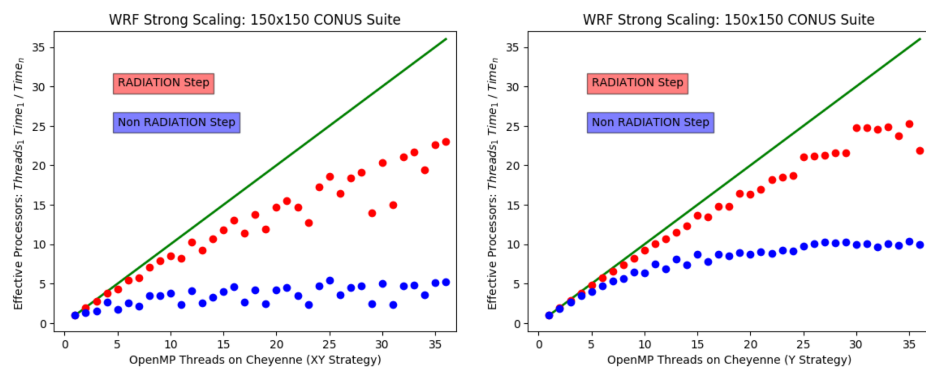
Distributed memory processing is required for larger jobs

## Using OpenMP Threading in WRF – Single Node

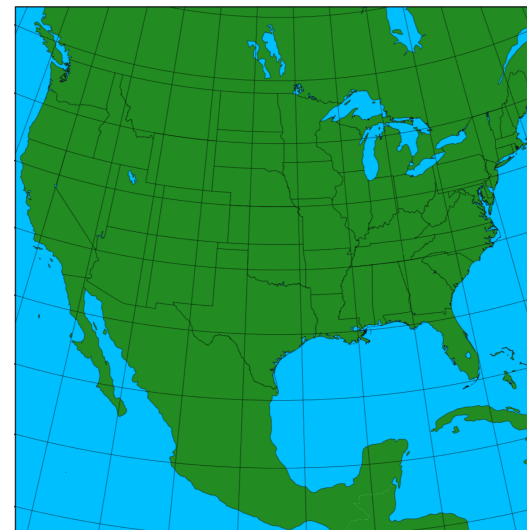
- The **WRF model supports OpenMP** threading
- Following is a 150x150 benchmark case (same geophysical size as 1500x1500, just using 30-km resolution instead of 3-km resolution)
- Has CU activated
- Running on a **single node of Cheyenne**, from 1 – 36 cores
- **Two tile strategies**: XY and Y



## Using OpenMP Threading in WRF – Single Node

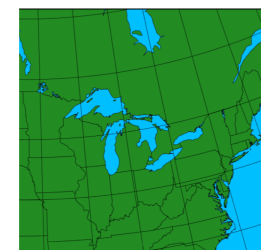
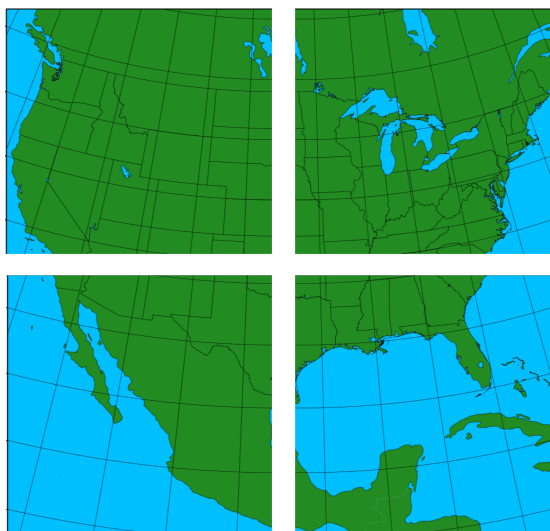


For MPI distributed memory jobs, we no longer can assume that neighboring processors can access the same memory



Each processor works independently, on its own patch.

Later the information is sent to other patches.



Processor 2  
rsl.out.0002  
rsl.error.0002



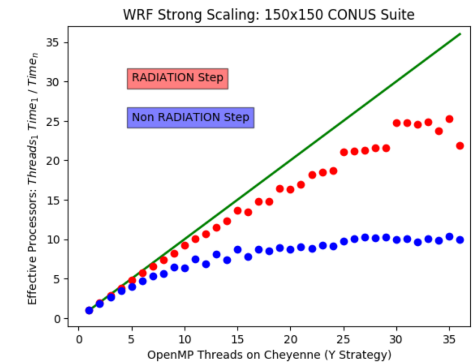
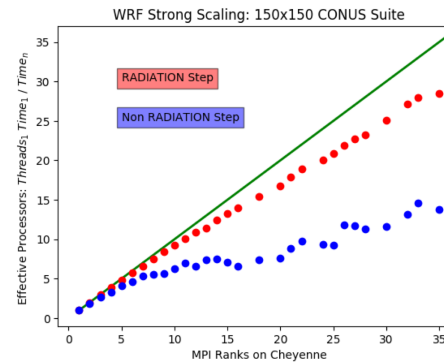
Processor 3  
rsl.out.0003  
rsl.error.0003

Processor 0  
rsl.out.0000  
rsl.error.0000



Processor 1  
rsl.out.0001  
rsl.error.0001

## Using MPI vs OpenMP in WRF – Single Node



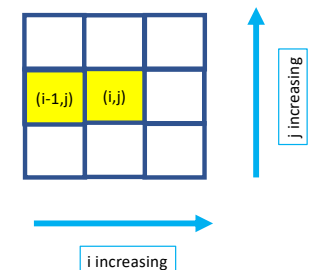
## OpenMP and MPI in WRF

- For a **sufficiently large WRF domain**, MPI scales to about 70% efficiency over a processor range of about **150x nodes**
  - Smallest node count is determined by memory requirements of WRF
  - Efficiency tends to drop before reaching a maximum allowable node count
- NOTE: **within node scaling is not effective** and is not typically used on a supercomputer
- OpenMP is ALWAYS within a single share-memory processing unit
- May have a clue as to why OpenMP parallelism is about 50% less than single-node MPI performance – **first touch**
- With MPI patches need to sometimes **send and receive information** from each other, referred to as messages and **message passing**
- WRF uses **HALO regions** to assist with message passing

## HALO

$$porig(i,k,j) = ( po(i,k,j) + po(i-1,k,j) ) * 0.5$$

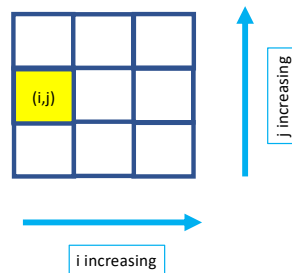
- Solve a simple **2-point stencil**, used for averaging a mass-point pressure to a momentum cell face location
- The assumption is that for each  $(i,j)$ , the  $(i-1,j)$  location is a **neighboring point**



## HALO

$$porig(i,k,j) = ( po(i,k,j) + po(i-1,k,j) ) * 0.5$$

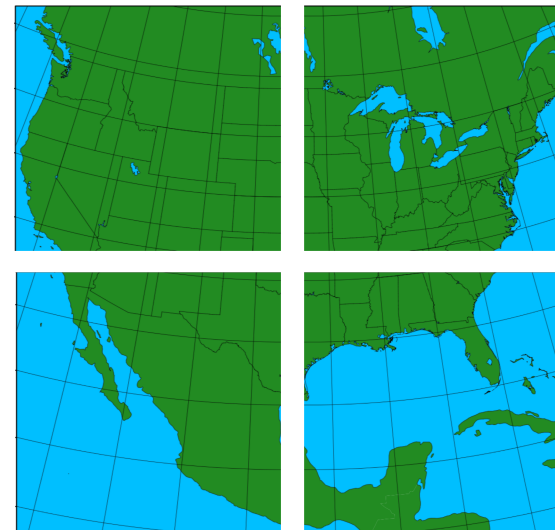
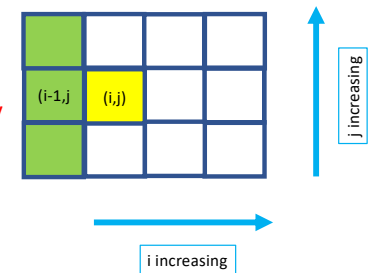
- For this stencil, if our grid cell lies on a **western boundary**, there is no neighboring point
- To get the information, we could **communicate with the next patch** and request the data
- However, **communication is much slower** than local memory access



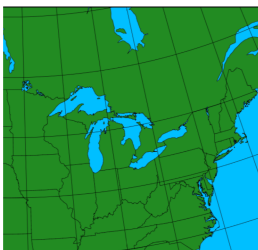
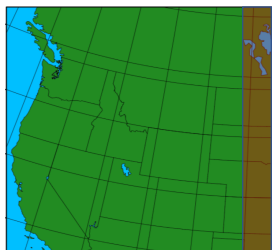
## HALO

$$porig(i,k,j) = ( po(i,k,j) + po(i-1,k,j) ) * 0.5$$

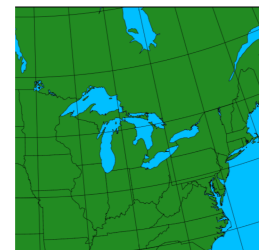
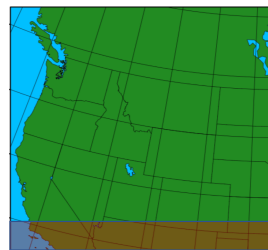
- Instead of communicating with a distributed memory processor for each computation, a **surrounding** group of cells along the boundary holds **read-only** information
- This **halo region** is kept updated periodically from the neighboring distributed memory processor



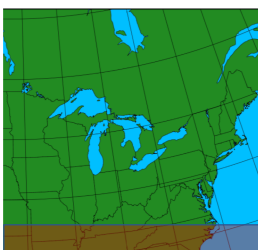




Halos to the  
left and  
below



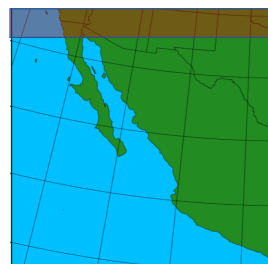
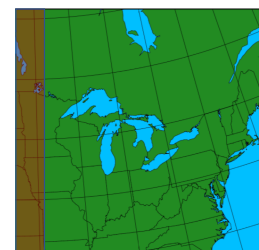
Halos to the  
right and  
above

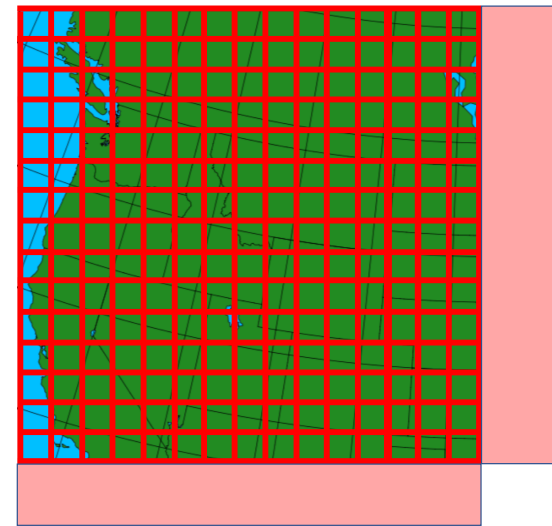
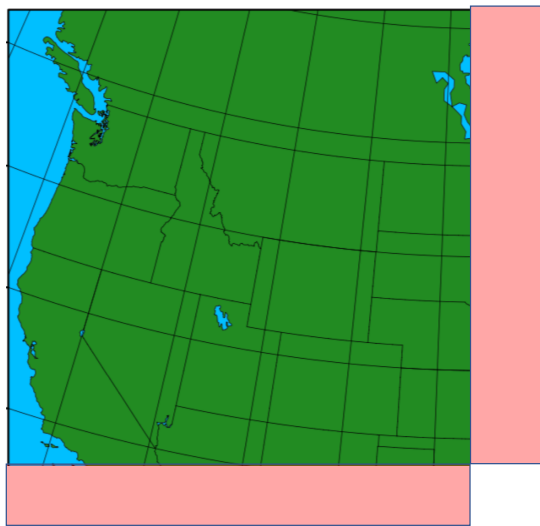


Halos to the  
left and  
above

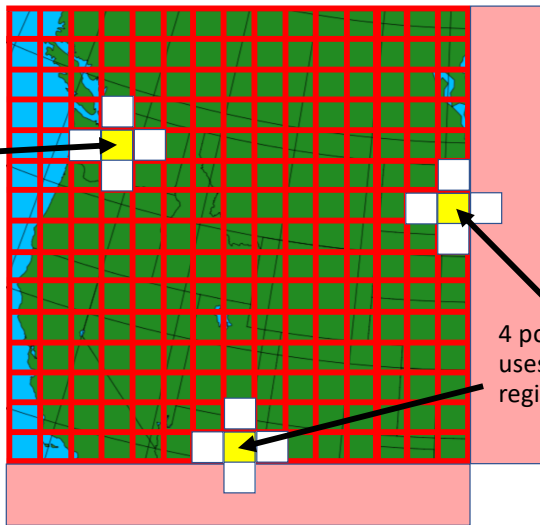


Halos to the  
right and  
below





4 point stencil  
requires no halo  
access



4 point stencil  
uses the halo  
region

## Domain Decomposition

- When we **decompose** the WRF domain into smaller pieces, we distribute work to **additional processors**  

```
> mpirun -np ??? wrf.exe
```
- QUESTIONS:
  - How do we determine **how many** processors to use for a job
  - What is the **shape** of the resultant decomposed grids
  - What are **performance consequences** of these decisions

## Domain Decomposition

- How many processors to use
- The **minimum** number of processors is based on the amount of **memory** that is available
- In the 1500x1500 benchmark case, a minimum of 4 nodes were required (4x36 processor cores) for the regular nodes, but only 2 nodes were required when using the large-memory nodes
- The WRF model would not run with a single node, it required too much memory

## Domain Decomposition

- How many processors to use
- The **maximum** number of processors is based on the underlying **stencil** communications inside of the WRF model
- The model gracefully halts if you try to make a resultant distributed memory patch with < 10 grid cells on either side
- For the 1500x1500 benchmark case, we could have 150 units of patches that are 10 grid cells across (in the i- and j- directions)
- Therefore a maximum of  $150 \times 150 = 22,500$  MPI processor cores

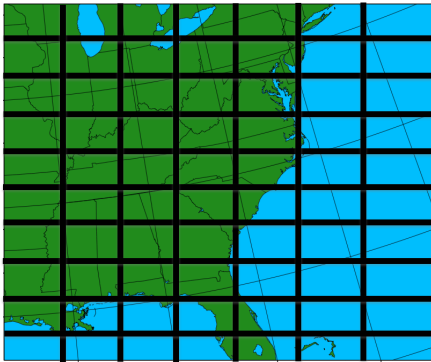
## Domain Decomposition

- How many processors to use
- The **recommended number** of processors is based on the timing performance that the WRF model is able to provide weighed against the timeliness of the required solution
- Usually, a **fewer** number of processors tends to more **efficiently utilize** the machine
- As long as **there are enough** processors (for memory) and **not too many** (for stencil sizes), the WRF model **solution is correct**

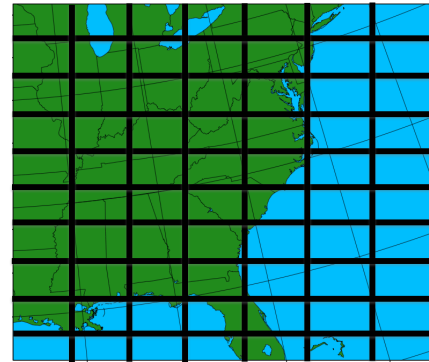
## Domain Decomposition

- How are the processor decompositions chosen
- By default, the decomposition of MPI tasks is computed as the **two closest multiplicative factors**
  - For example: 32 MPI tasks = 4x8 decomposition, NOT 2x16
  - For example: 144 MPI tasks = 12x12 decomposition, NOT 4x36
- The **larger** of the two factors decomposes the **j-direction**
- What to **avoid**: **primes** or large prime factors

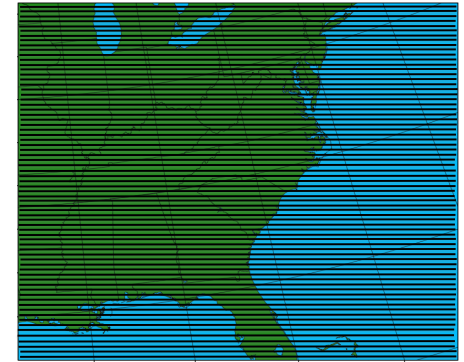
- 70 tasks
- 10 (j) x 7 (i)



- 70 tasks
- 10 (j) x 7 (i)



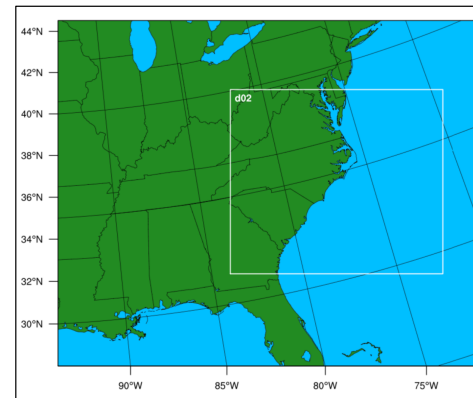
- 71 tasks
- 71 (j) x 1 (i)



## Domain Decomposition

- Domain decomposition choices are important with **nesting**
- All domains within the WRF model use the exact **same number** of processors, with the **same processor decomposition**
- Choosing coarse grid domains and fine grid domains that have a large grid cell difference negatively impacts timing performance
- A **small coarse grid** (compared to the nest) **restricts** the number of processors that are able to be utilized by the fine grid domain (the expensive domain)
- The **fine grid is the expensive** domain

## Domain Decomposition



```
&geogrid
  parent_id      = 1, 1,
  parent_grid_ratio = 1, 3,
  i_parent_start  = 1, 31,
  j_parent_start  = 1, 17,
  e_we           = 74, 112,
  e_sn           = 61, 97,
/
```

- Default max cores on CG: 36
- Default max cores on FG: 81
- With namelist options: 42 vs 99
- Do not make CG smaller than FG

## Where is all of this information

- When running the WRF model with MPI, **two text files** are generated by default for each MPI task
  - `rsl.out.nnnn`
  - `rsl.error.nnnn`
- Where `nnnn` is the processor number of the WRF job, 0000 through n-1 for the "mpirun -np n wrf.exe" job submission
- For a successfully completed job, important information is inside the `rsl.out.0000` file

## Where is all of this information

- How many MPI tasks are used

```
Ntasks in X          9 , ntasks in Y          12
```

- What is the decomposed domain size

Parent domain				
<code>ids,ide,jds,jde</code>	1	1500	1	1500
<code>ims,ime,jms,jme</code>	-4	174	-4	132
<code>ips,ipe,jps,jpe</code>	1	167	1	125

- How much memory is allocated on the heap

```
alloc_space_field: domain 1 ,          792646428 bytes allocated
```

## Where is all of this information

- How much time to write out data

```
Timing for Writing wrfout_d01_2019-05-05_22:00:00 for domain 1: 139.06874 elapsed seconds
```

- How much time to do a time step (first includes I/O time)

```
Timing for main: time 2019-05-05_22:00:18 on domain 1: 171.42862 elapsed seconds
Timing for main: time 2019-05-05_22:00:36 on domain 1: 6.98657 elapsed seconds
Timing for main: time 2019-05-05_22:00:54 on domain 1: 6.99922 elapsed seconds
Timing for main: time 2019-05-05_22:01:12 on domain 1: 6.99057 elapsed seconds
```

- Parallelism in WRF
  - Why is it required
  - What is "scaling"

- OpenMP and MPI

- Halos

- Domain Decomposition

- Reasonable, max, min, bad
- Dimensions and decomposition
- Coarse grid vs nest grid

- Where is all of this information

- Using `rsl` files

