NC STATE UNIVERSITY

Improved Efficiency for Wave and Surge Models via Adaptive Domain Decomposition

JC Dietrich[1], CN Dawson[2], A Thomas[1]

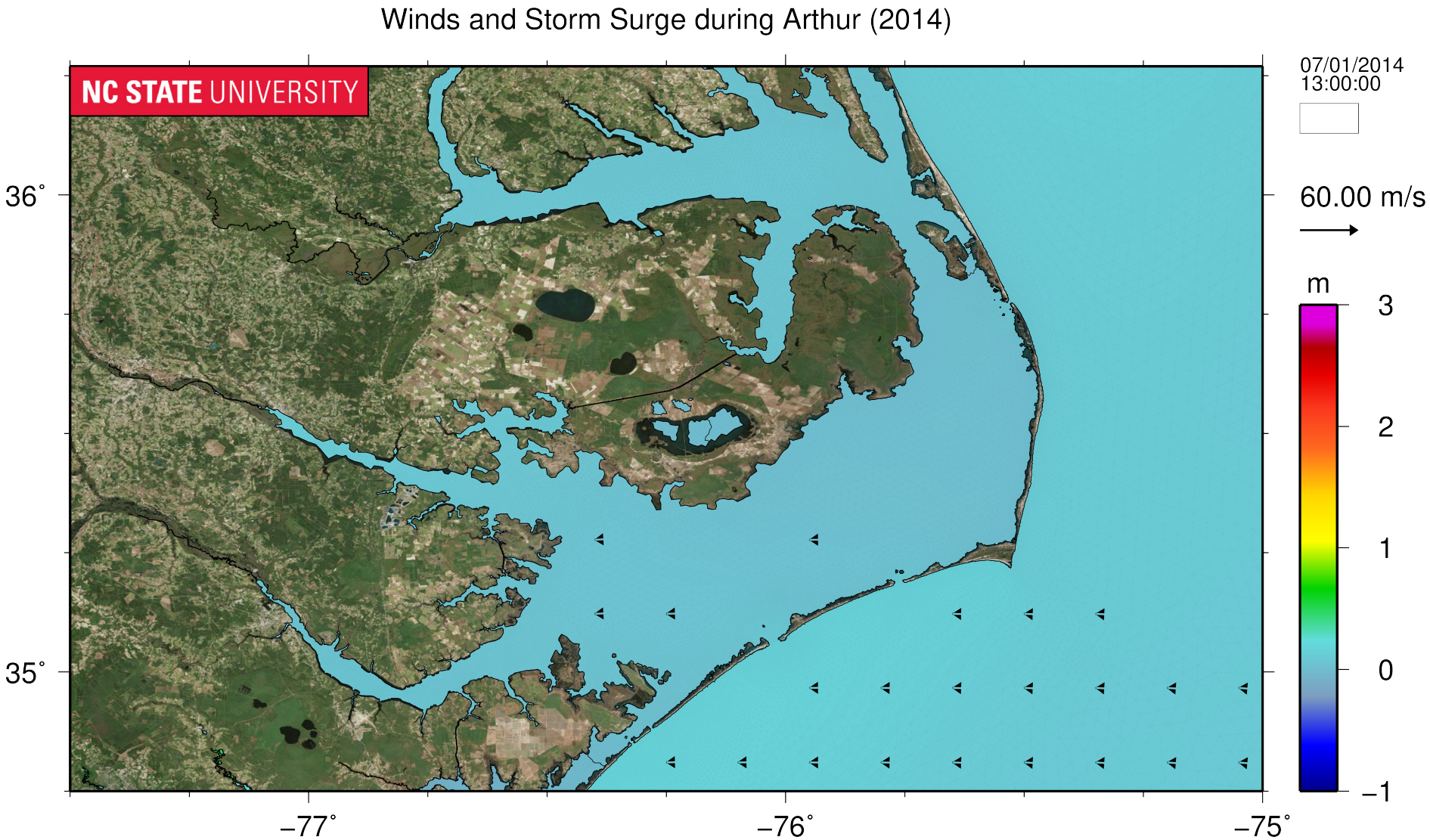
[1]Dep't of Civil, Construction, and Environmental Engineering, NC State University
[2]Inst. for Computational Engineering and Sciences, University of Texas at Austin

XXI Intl Conference on Computational Methods in Water Resources
Toronto ON, 23 June 2016

COASTAL RESILIENCE CENTER
A U.S. Department of Homeland Security Center of Excellence

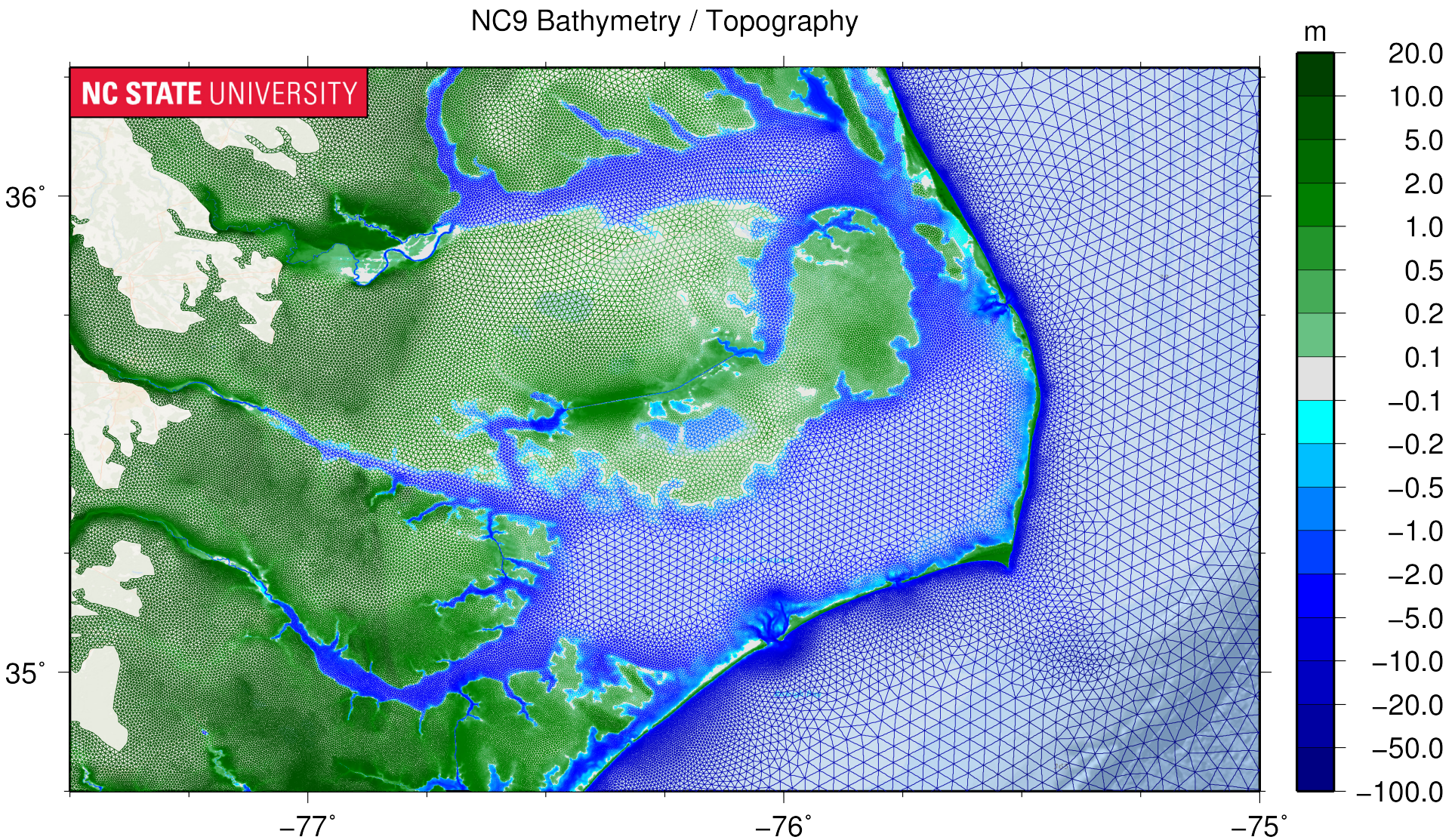
Predictive Models for Hurricane Waves and Storm Surge

Example of Coastal Flooding



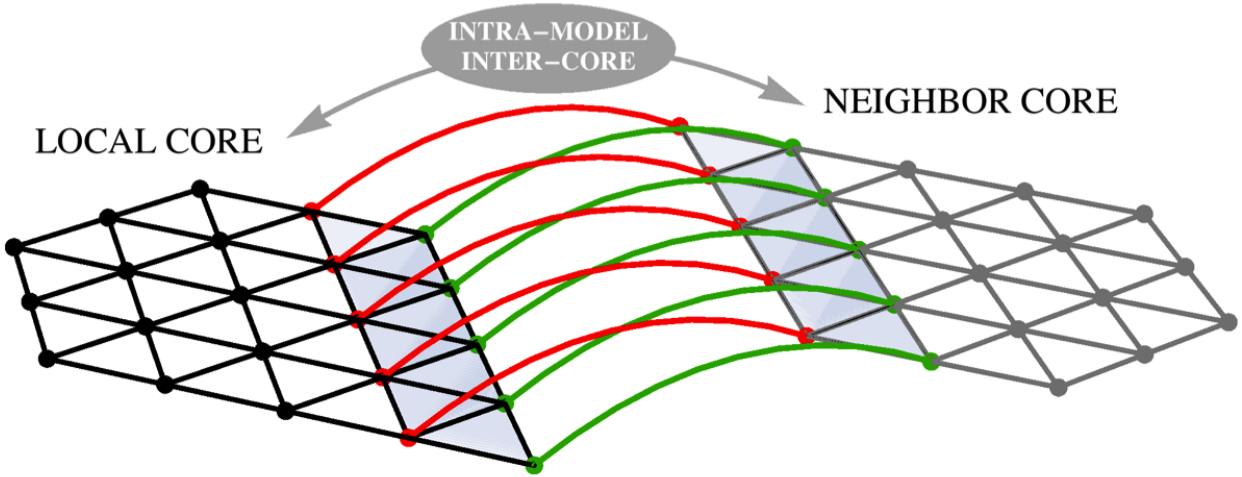
Predictive Models for Hurricane Waves and Storm Surge

Finite-Element Mesh for NC Coast



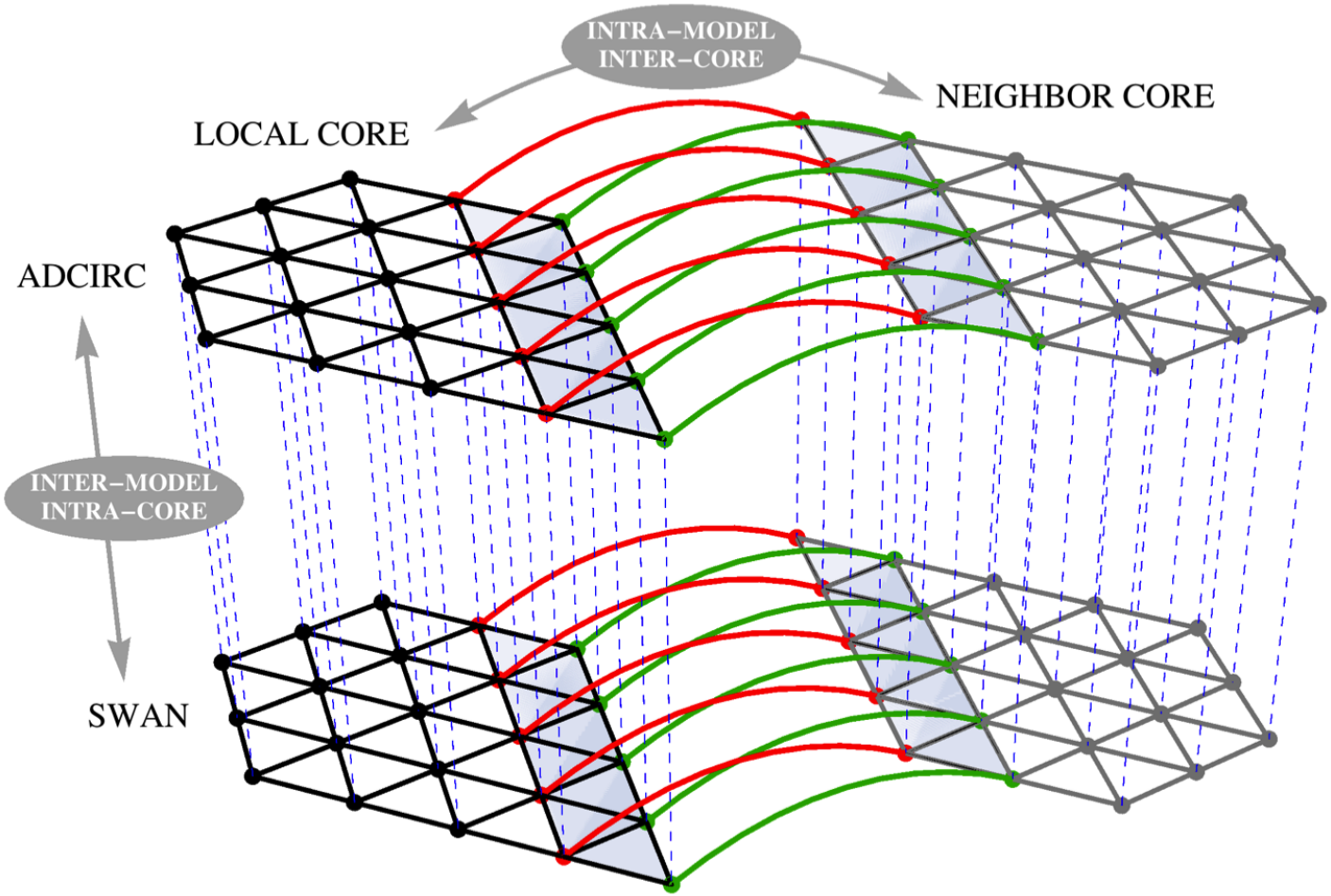
Load Balancing via Static Domain Decomposition

Schematic of Parallel Communication



Load Balancing via Static Domain Decomposition

Schematic of Parallel Communication



Load Balancing via Static Domain Decomposition

Existing Preprocessing in ADCIRC

We use METIS to decompose our domain

- Separate library, written in C, developed at Univ. Minnesota
- Weights based on the number of edges connected to each vertex
- METIS tries to equalize the weights across the subdomains

Our preprocessor was written about 15-18 years ago

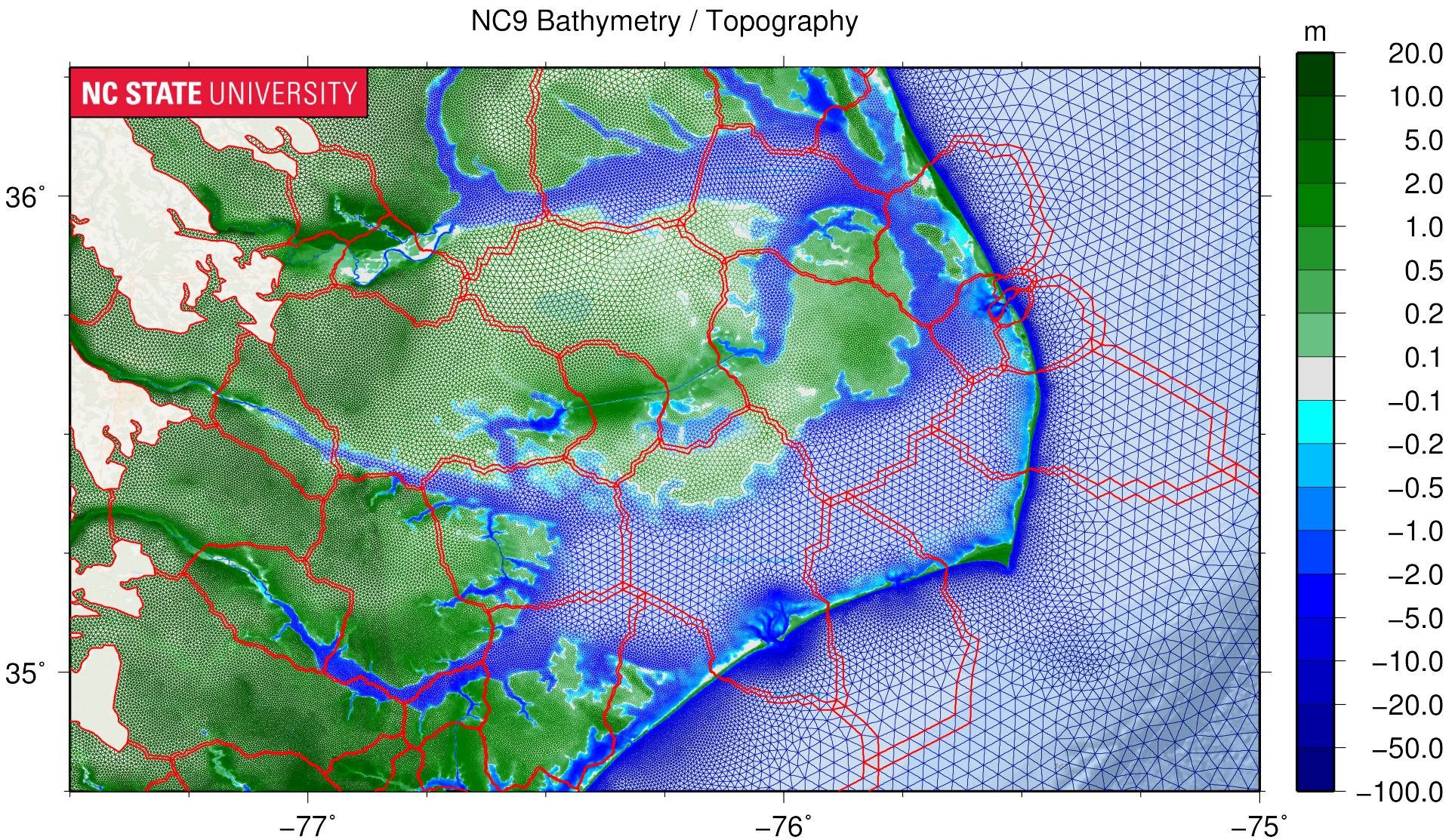
- Our domains were entirely wet – oceans and coastal regions
- No floodplains or dry regions to consider in the decomposition
- Initial decomposition was static

Need to revise the domain decomposition

- Equal amounts of wet and dry to each core
- Skip computations on dry vertices

Load Balancing via Static Domain Decomposition

Example on 95 Cores



Load Balancing via Static Domain Decomposition

Changes to Preprocessing in ADCIRC

Now METIS is called twice

- Called first to decompose only the wet regions
- Called again to decompose only the dry regions

Each core is assigned one wet region, and one dry region

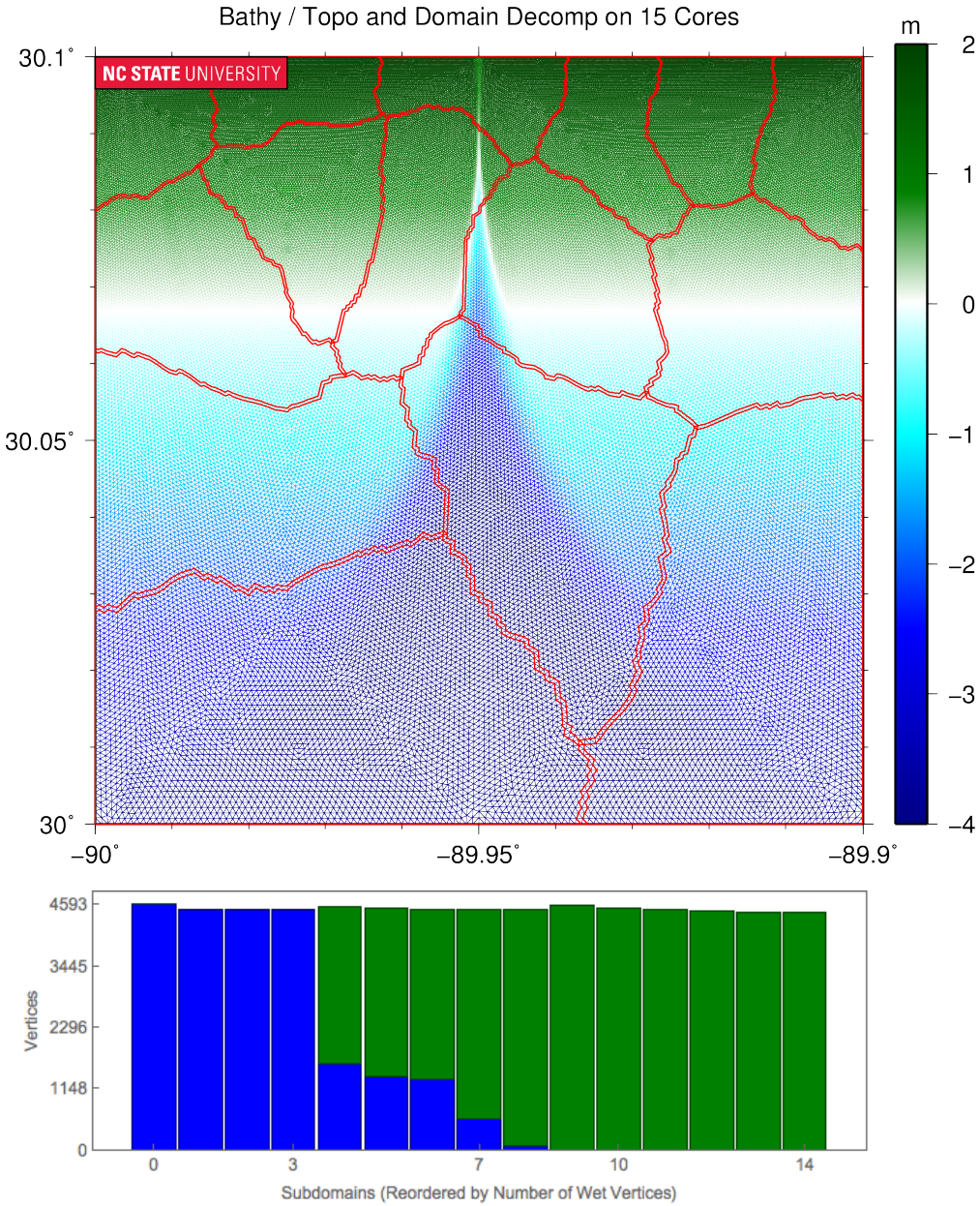
- This way, each core will be contributing to the workload
- Work is balanced at start of simulation
- May become imbalanced due to wetting / drying during simulation

No guarantee that each core's wet and dry regions will be connected

- One core may have two subdomains that are far from each other
- Potential problem for large domains – we are increasing the communication

Test Case 1 – Ideal Channel and Floodplain

Initial Domain Decomposition



First test case is channel and floodplain:
- Ideal mesh: 64,415 vertices
- Shallow depths from −4m to +2m
- Tidal range from −1m to +1m

So we expect a lot of wetting/drying:
- Roughly 1/3 of the domain by size
- More of the domain by resolution

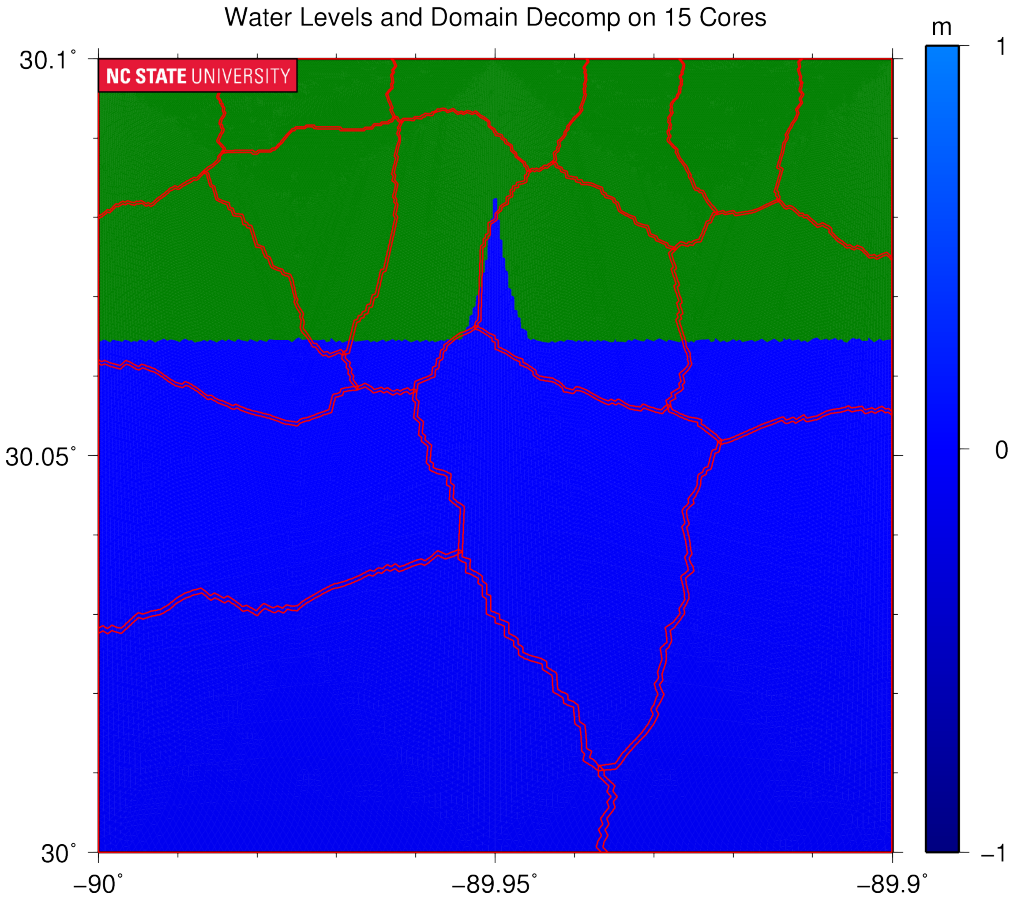
Initial decomposition is sub-optimal:
- 4 cores start fully wet
- 5 cores start partly wet/dry
- 6 cores start fully dry

Simulation of 4 days:
- Eight tidal cycles
- Extensive wetting and drying
- Some cores are always dry

Wall-clock time of $\sim$ 17.8 min

Test Case 1 – Ideal Channel and Floodplain

Initial Domain Decomposition



First test case is channel and floodplain:
- Ideal mesh: 64,415 vertices
- Shallow depths from –4m to +2m
- Tidal range from –1m to +1m

So we expect a lot of wetting/drying:
- Roughly 1/3 of the domain by size
- More of the domain by resolution

Initial decomposition is sub-optimal:
- 4 cores start fully wet
- 5 cores start partly wet/dry
- 6 cores start fully dry

Simulation of 4 days:
- Eight tidal cycles
- Extensive wetting and drying
- Some cores are always dry

Wall-clock time of $\sim$ 17.8 min

Test Case 1 – Ideal Channel and Floodplain

Optimizing Initial Decomposition for Wet Vertices

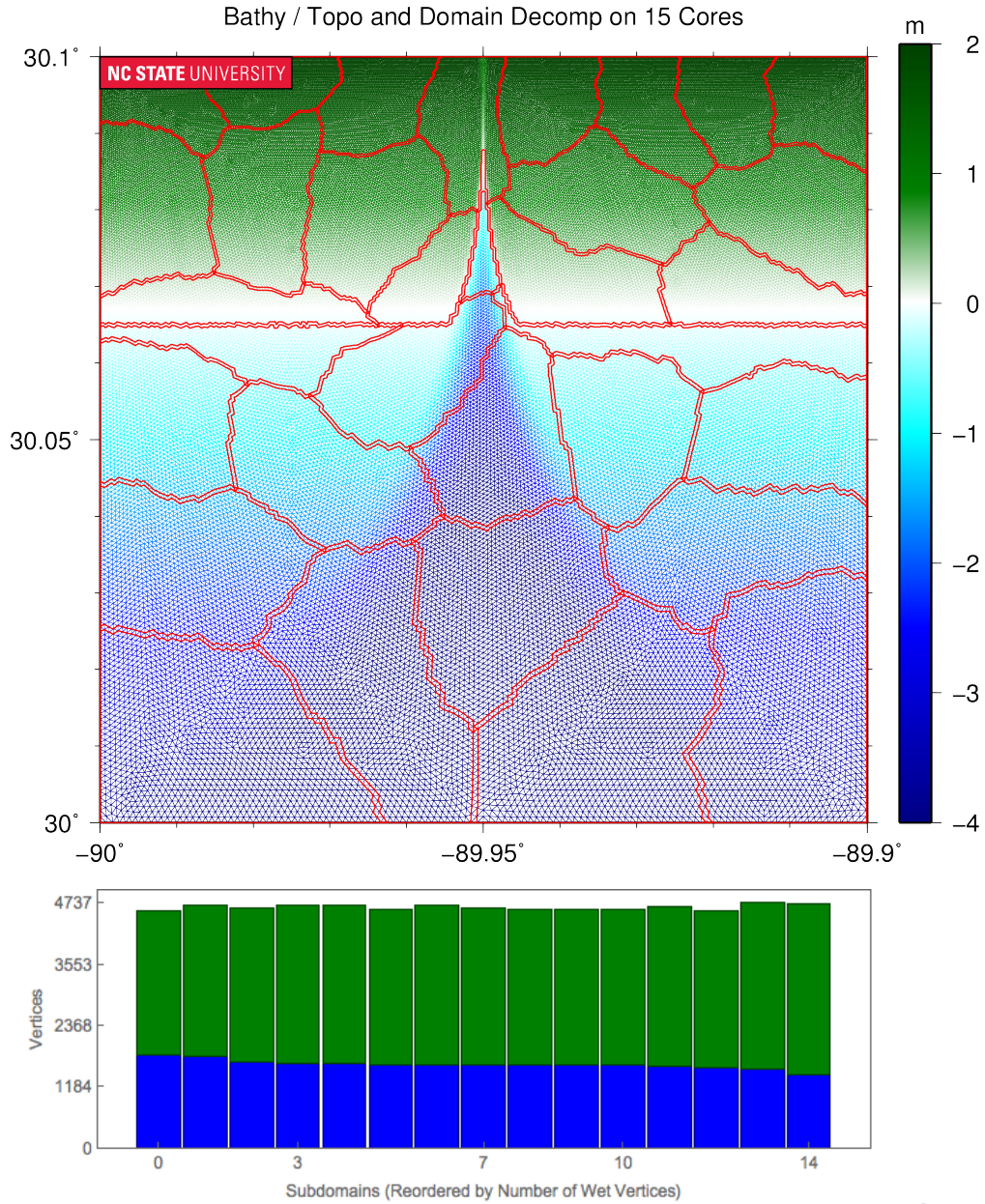
We optimized the decomposition:
- All cores are both wet and dry

Changes to adcprep:
- METIS called twice
- Weights only on vertices (not edges)
- Separate subdomains for wet and dry

For example, core 0:
- Wet region with 1591 vertices
- Dry region with 3013 vertices
- Not guaranteed to connect

Now every core is contributing
- Still imbalances during tidal cycle

Wall-clock time of $\sim$ 13.1 min
- Speedup of **26 percent**



Test Case 1 – Ideal Channel and Floodplain

Optimizing Initial Decomposition for Wet Vertices

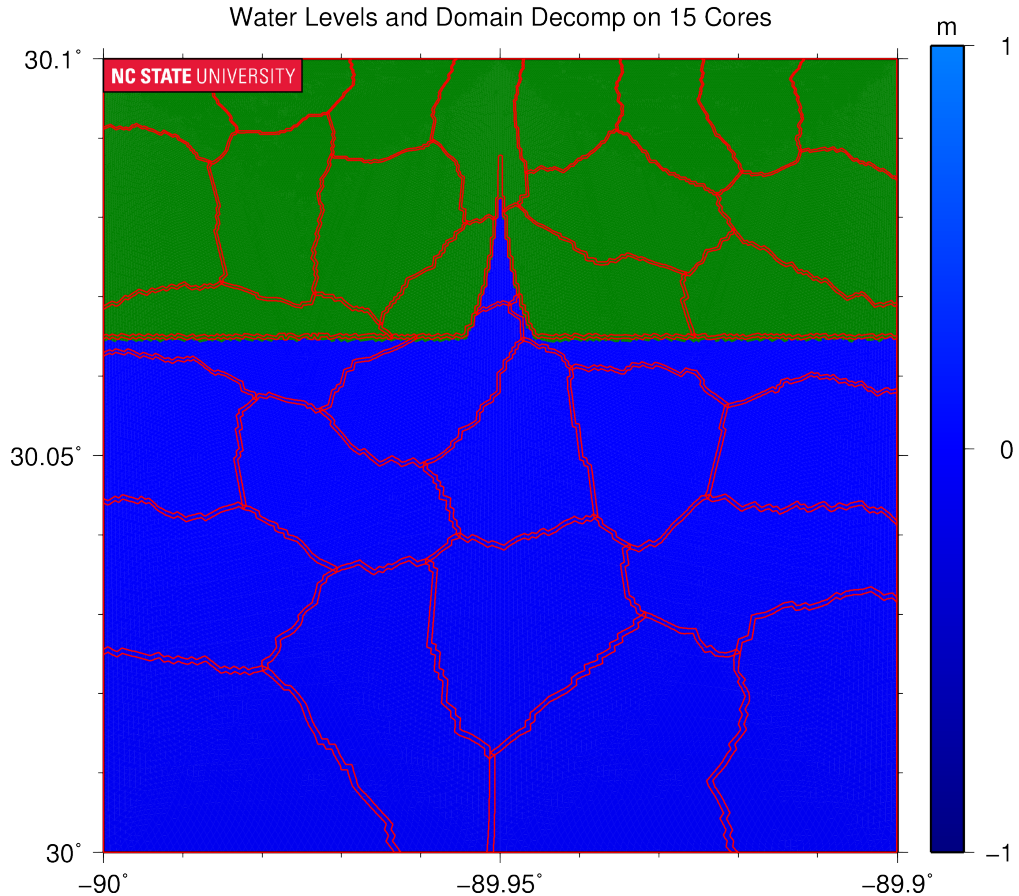
We optimized the decomposition:
- All cores are both wet and dry

Changes to `adcprep`:
- METIS called twice
- Weights only on vertices (not edges)
- Separate subdomains for wet and dry

For example, core 0:
- Wet region with 1591 vertices
- Dry region with 3013 vertices
- Not guaranteed to connect

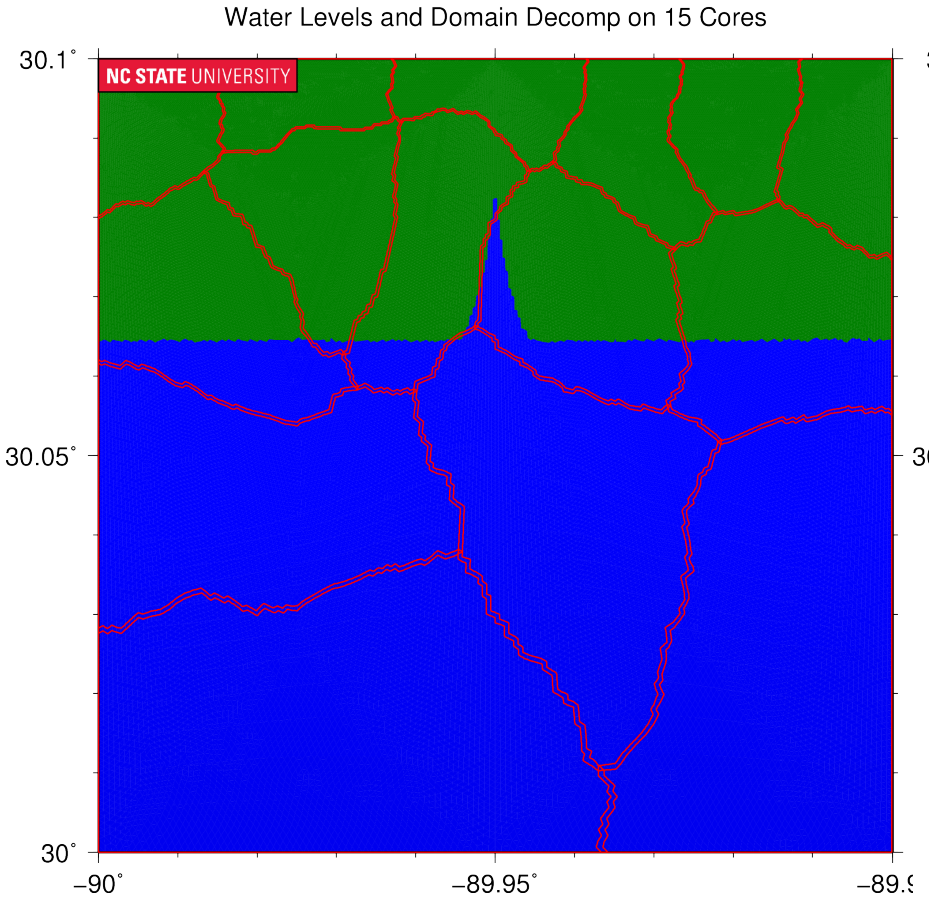
Now every core is contributing
- Still imbalances during tidal cycle

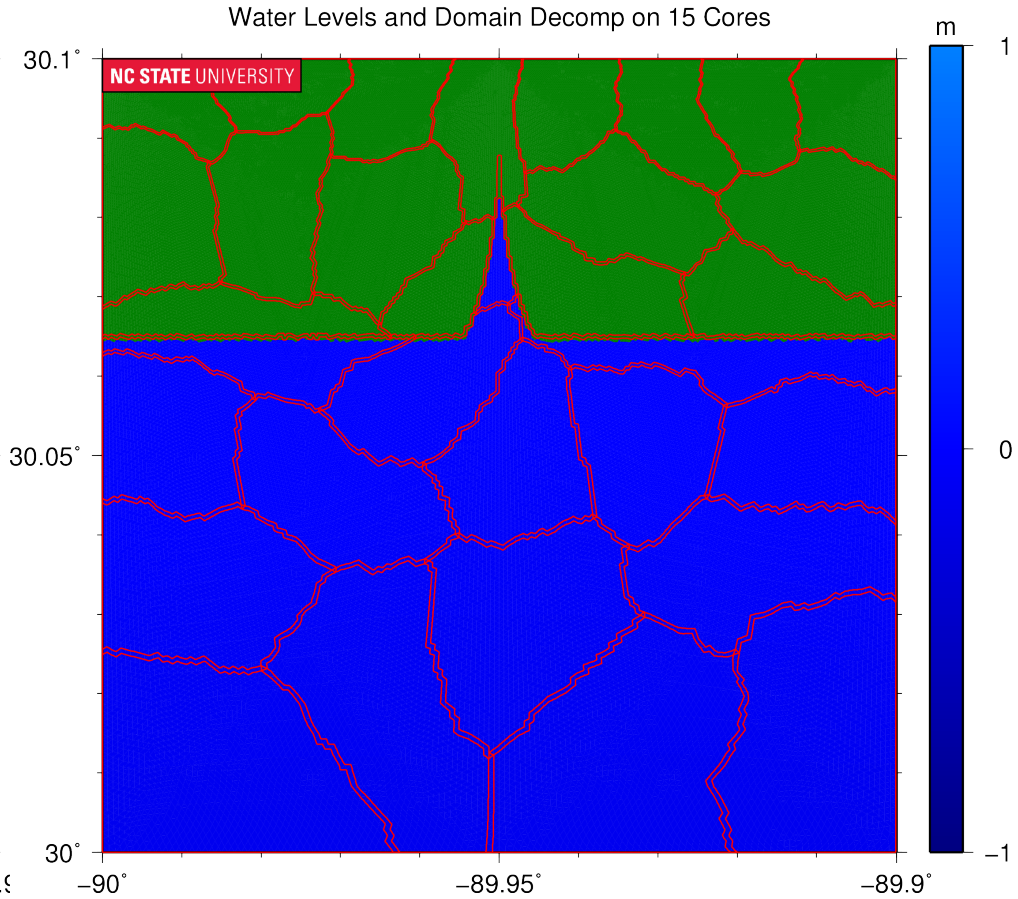
Wall-clock time of $\sim$ 13.1 min
- Speedup of **26 percent**



Test Case 1 – Ideal Channel and Floodplain

Water Level Comparison and Timings

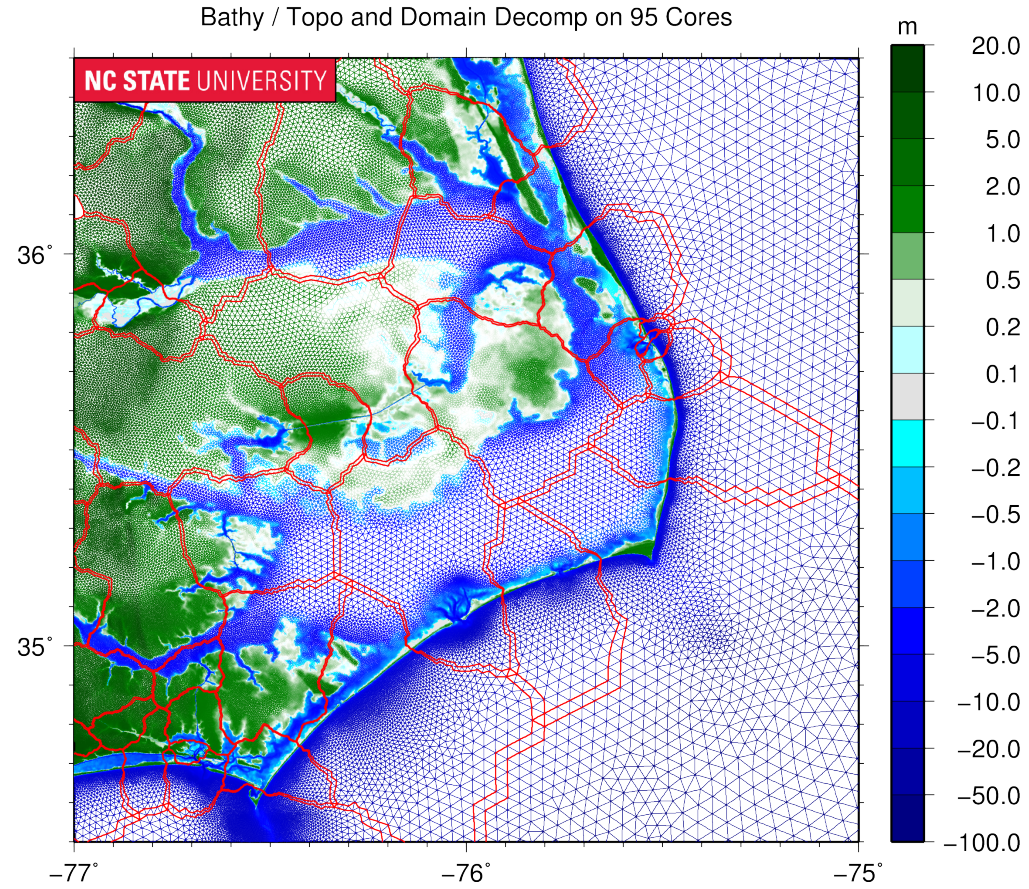


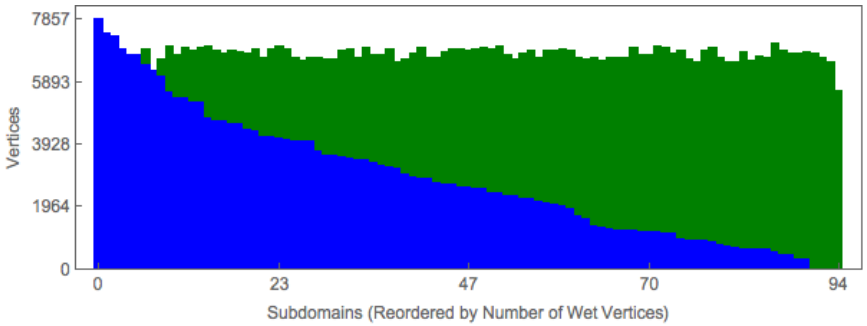


Test	Code Version	Cores	CPU-hr	% Change
Wet/Dry	ADCIRC v52.22	15	4.46	
	+ load balancing		3.29	**– 26.3**

Test Case 2 – Hurricane Irene (2011) on NC9

Initial Domain Decomposition





Second test case is Irene (2011):
- NC9 mesh: 622,946 vertices

Should be a lot of wetting/drying:
- Roughly 1/2 of domain starts dry
- Flooding of NC coastal regions

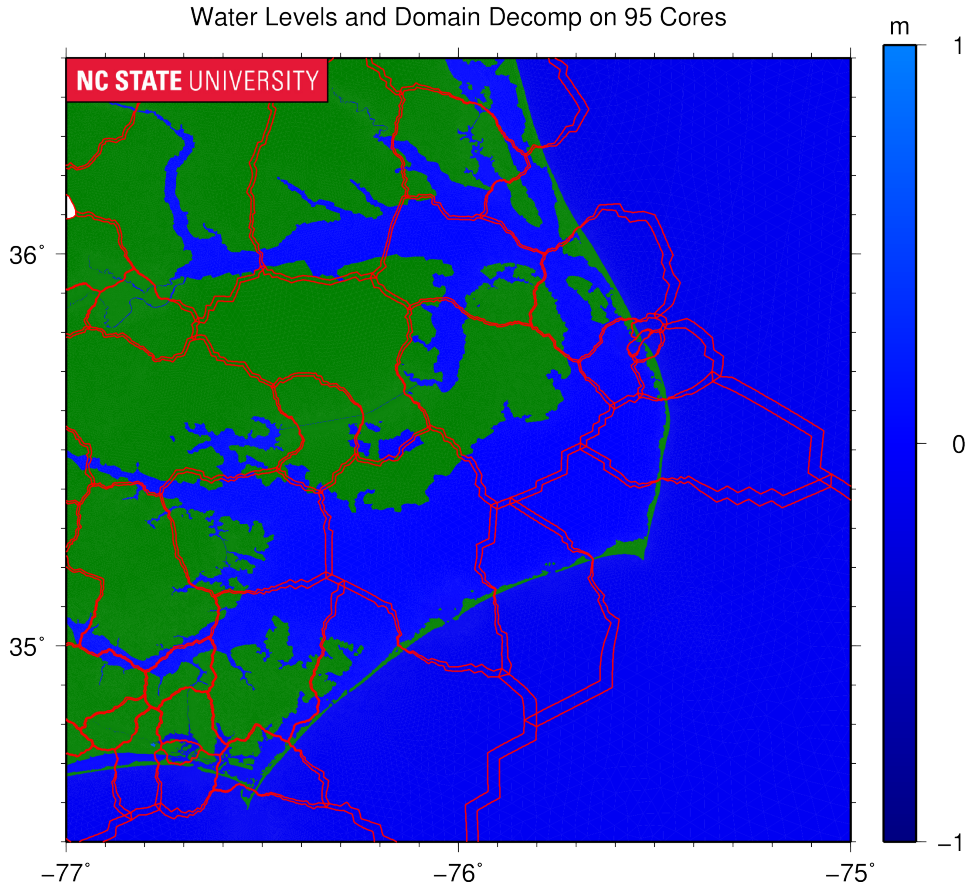
Initial decomposition is sub-optimal:
- Only 7 cores start fully wet
- Most cores are mostly dry

Simulation of 8 days:
- Initial flooding in SW Pamlico Sound
- Sound-side flooding of Hatteras Island

Wall-clock time of 3.52 hr

Test Case 2 – Hurricane Irene (2011) on NC9

Initial Domain Decomposition



Second test case is Irene (2011):
- NC9 mesh: 622,946 vertices

Should be a lot of wetting/drying:
- Roughly 1/2 of domain starts dry
- Flooding of NC coastal regions

Initial decomposition is sub-optimal:
- Only 7 cores start fully wet
- Most cores are mostly dry

Simulation of 8 days:
- Initial flooding in SW Pamlico Sound
- Sound-side flooding of Hatteras Island

Wall-clock time of 3.52 hr

Test Case 2 – Hurricane Irene (2011) on NC9

Optimizing Initial Decomposition for Wet Vertices

We optimized the decomposition:
- All cores are both wet and dry
- Domain boundaries follow shoreline
- Channels separate from floodplains

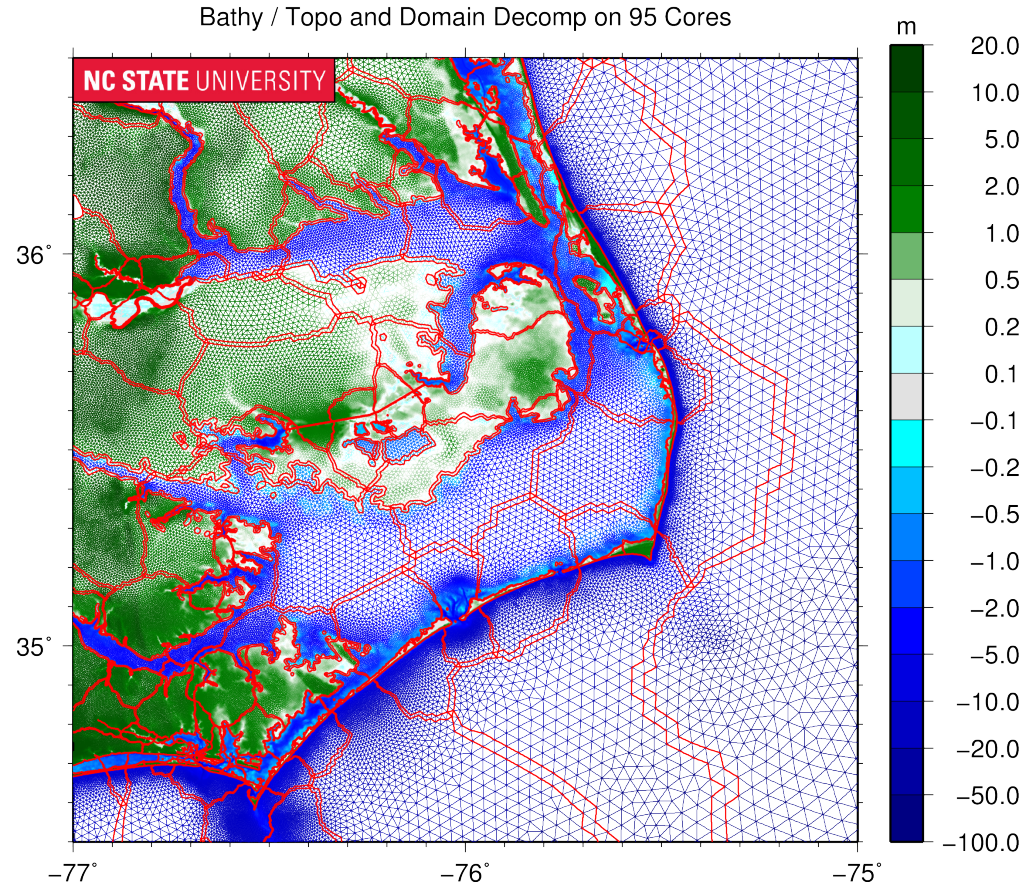
Domain not decomposed equally:
- Some cores have more wet / dry
- Reflects complexity of mesh

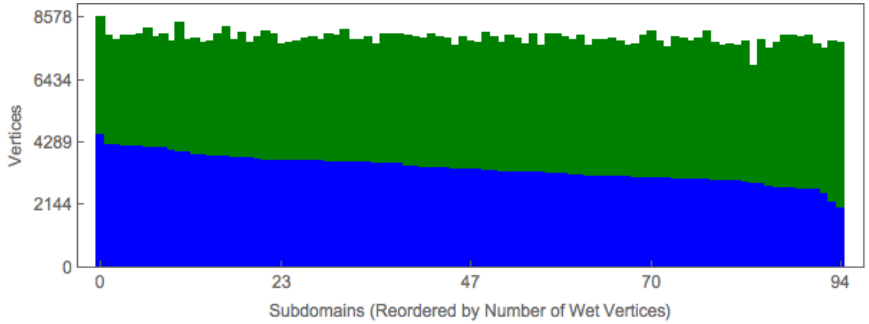
For example, core 0000:
- Wet region with 2274 vertices
- Dry region with 5479 vertices

For example, core 0001:
- Wet region with 3066 vertices
- Dry region with 4766 vertices

Now every core is contributing
- Still imbalances during storm

Wall-clock time of $\sim$ 2.77 hr
- Speedup of **21 percent**





Test Case 2 – Hurricane Irene (2011) on NC9

Optimizing Initial Decomposition for Wet Vertices

We optimized the decomposition:
- All cores are both wet and dry
- Domain boundaries follow shoreline
- Channels separate from floodplains

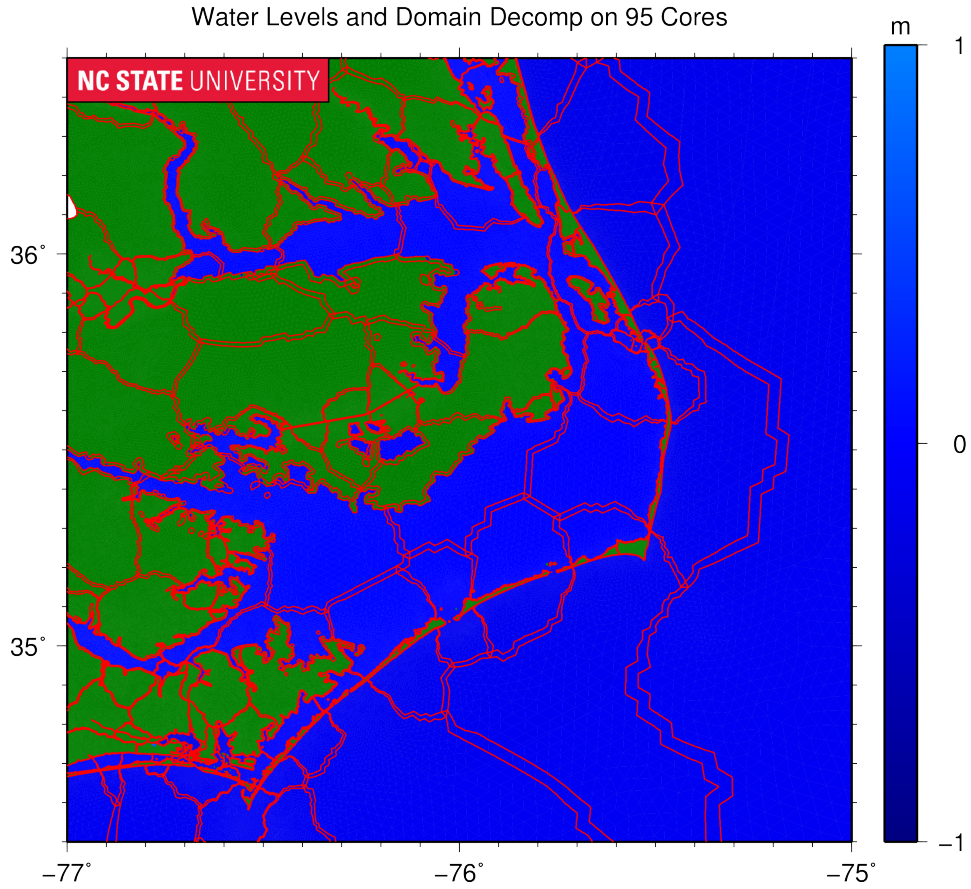
Domain not decomposed equally:
- Some cores have more wet / dry
- Reflects complexity of mesh

For example, core 0000:
- Wet region with 2274 vertices
- Dry region with 5479 vertices

For example, core 0001:
- Wet region with 3066 vertices
- Dry region with 4766 vertices

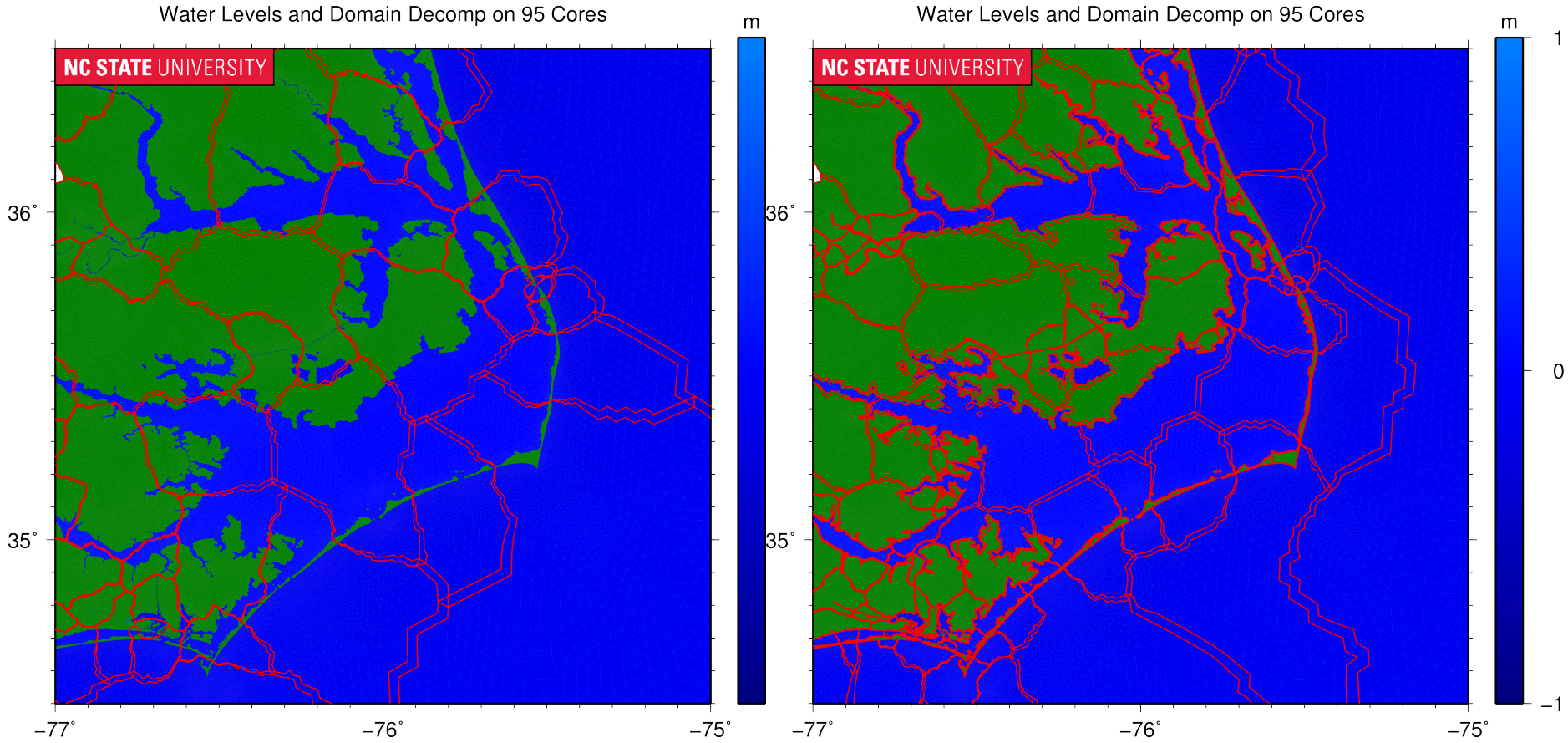
Now every core is contributing
- Still imbalances during storm

Wall-clock time of $\sim$ 2.77 hr
- Speedup of **21 percent**



Test Case 2 – Hurricane Irene (2011) on NC9

Water Level Comparison and Timings



Test	Code Version	Cores	CPU-hr	% Change
Tides	ADCIRC v52.22	95	259.2	
	+ load balancing		210.7	**– 18.7**
Irene	ADCIRC v52.22	95	334.3	
	+ load balancing		263.3	**– 21.2**

Summary and Future Work

Load Balancing via Domain Decomposition

Static load balancing

- Initial speed-up of 20 percent
- Faster forecasts for decision support for emergency managers
- Better utilize resources for larger studies for engineering design

Need to revise our preprocessor so it can be called as a subroutine

- Use ParMETIS – Each core will develop its own subdomain
- Won't need to run the prepreoccesor as a separate step
- Start with global input files and kick off a simulation
- Rebalance the workload during the simulation