Performance Evaluation of MPI on Weather and Hydrological Models

Alessandro Fanfarillo
elfanfa@ucar.edu

August 8th 2018

Cheyenne - NCAR Supercomputer

- Cheyenne is a 5.34-petaflops, high-performance computer built for NCAR by SGI.
- 2.3-GHz Intel Xeon E5-2697V4 (Broadwell) processors. 36 cores on each processor.
- **Mellanox EDR InfiniBand:** Bandwidth: 25 GBps bidirectional per link. Latency: MPI ping-pong < 1 μs; hardware link 130 ns.
- Five MPI implementations installed:
    - MPT SGI
    - Intel MPI
    - Open-MPI
    - MVAPICH2-2.3
    - MPICH-3.2

- "Casper": heterogeneous system of specialized data analysis and visualization resources and large-memory, multi-GPU nodes (Nvidia V100).

Weather Research and Forecast Model (WRF)

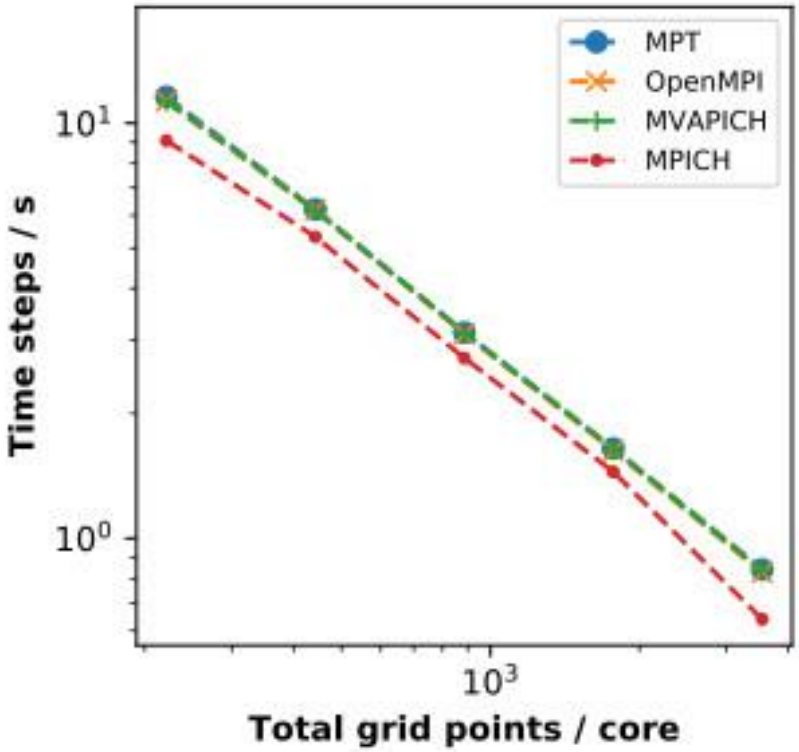
- Parallel numerical mesoscale weather forecasting application used for both research and operational environments.
- It is a supported "community model", i.e. a free and shared resource with distributed development and centralized support.
- Used for:
  - Numerical weather prediction
  - Meteorological case studies
  - Regional climate
  - Air quality, wind energy, hydrology, etc.
- Implemented mostly in Fortran90 with some parts in C, Perl. It used MPI for inter-process communication.

Summary of WRF benchmarks

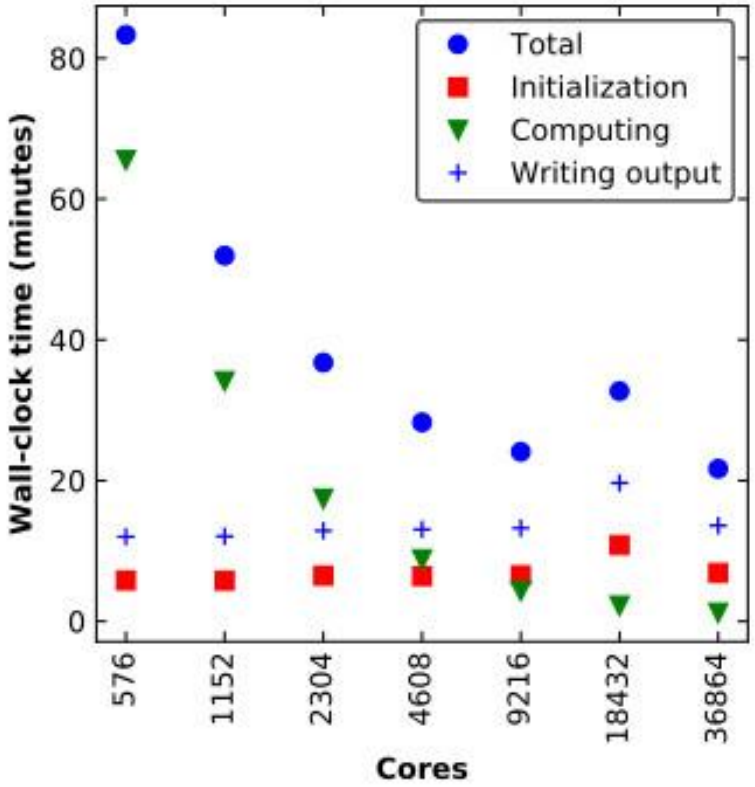
All the tests made by Akira “Aiko” Kyle (Carnegie Mellon University) and Dixit Patel (CU Boulder) interns at NCAR.

Region	Res	Horizontal Gridpoints	Vertical Gridpoints	Tot Gridpoints	Time Step	Run Time
CONUS	12 km	425	300	127,500	72 secs	6 hrs
CONUS	2.5 km	1901	1301	2,473,201	15 secs	6 hrs
Maria	3 km	1396	1384	1,932,064	9 secs	3 hrs
Maria	1 km	3665	2894	10,606,510	3 secs	1 hrs

Overall Scaling

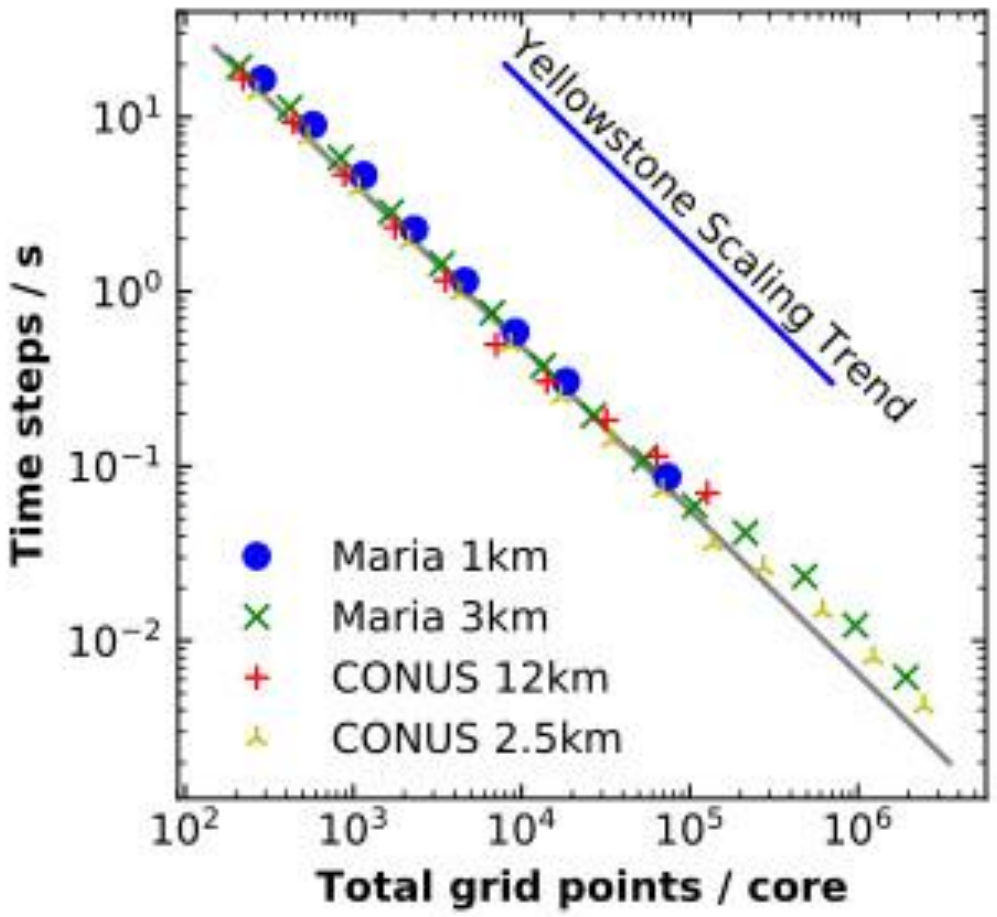


Conus 12 Km



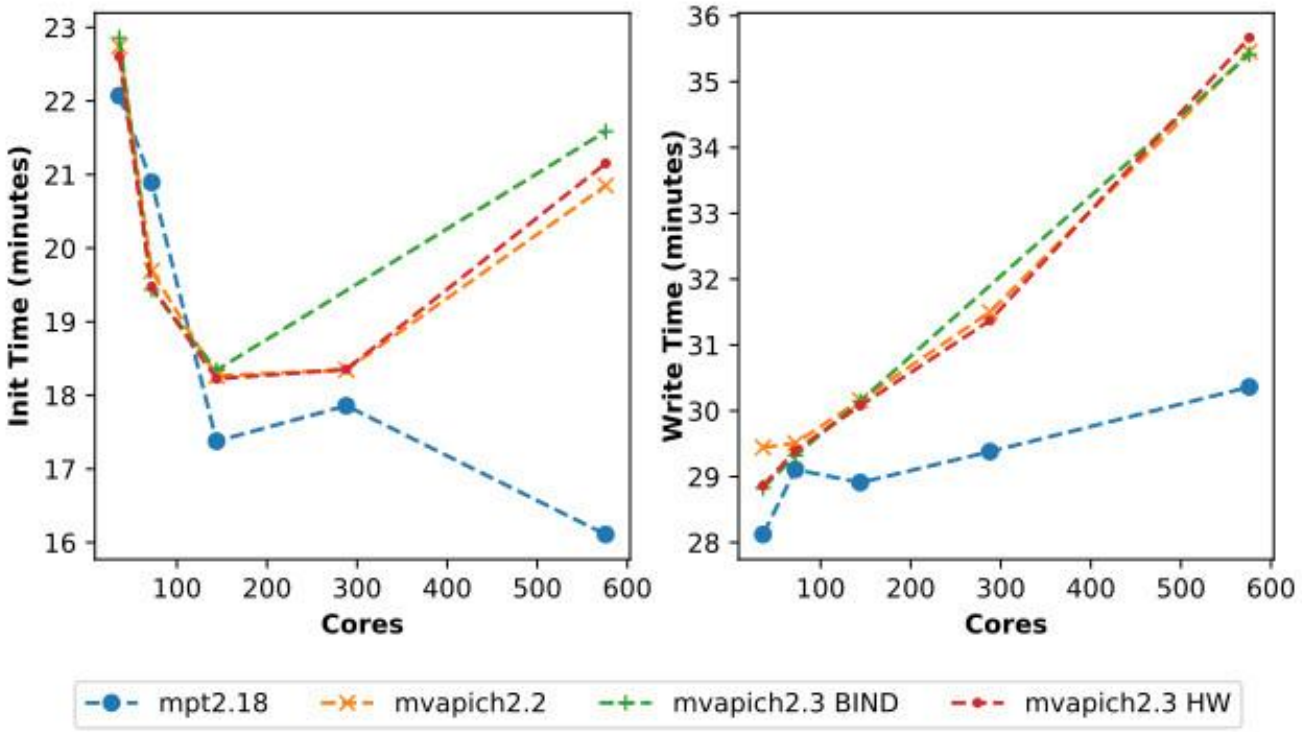
Maria 1 Km

Computation time scaling

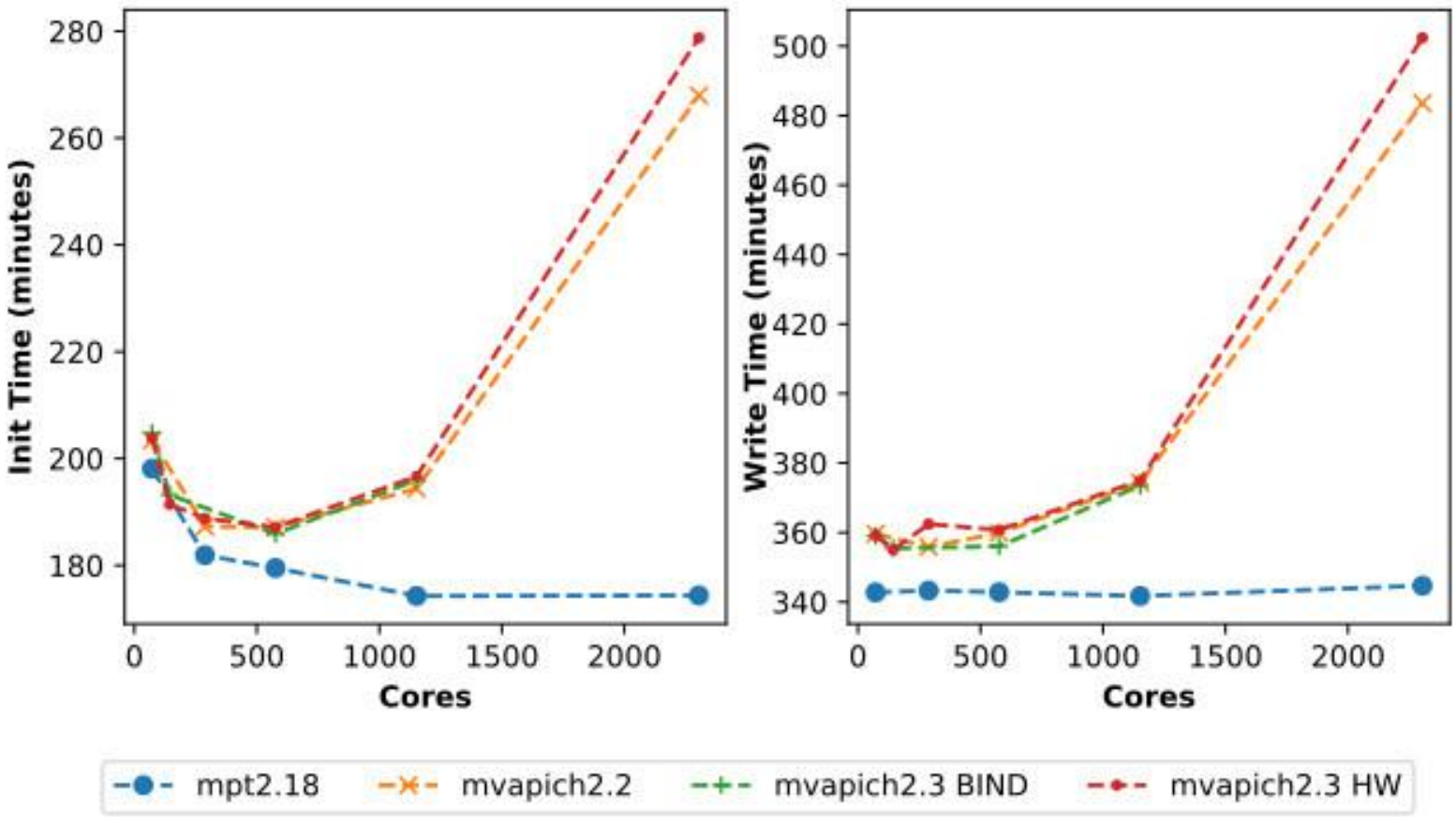


MVAPICH2 - Collective and IO

- BIND
  - MV2_CPU_BINDING_POLICY=hybrid
  - MV2_HYBRID_BINDING_POLICY=bunch
- HW
  - MV2_USE_MCAST=1
  - MV2_ENABLE_SHARP=1



MVAPICH2 - Collectives and IO



Maria 3 Km

WRF-Hydro and National Weather Model

Colorado Flood of 11-15 Sept. 2013

WRF-Hydro was originally designed as a model coupling framework to facilitate coupling between WRF and components of terrestrial hydrological models.

WRF-Hydro is both a stand-alone hydrological modeling architecture as well as a coupling architecture for coupling of hydrological models with atmospheric models.

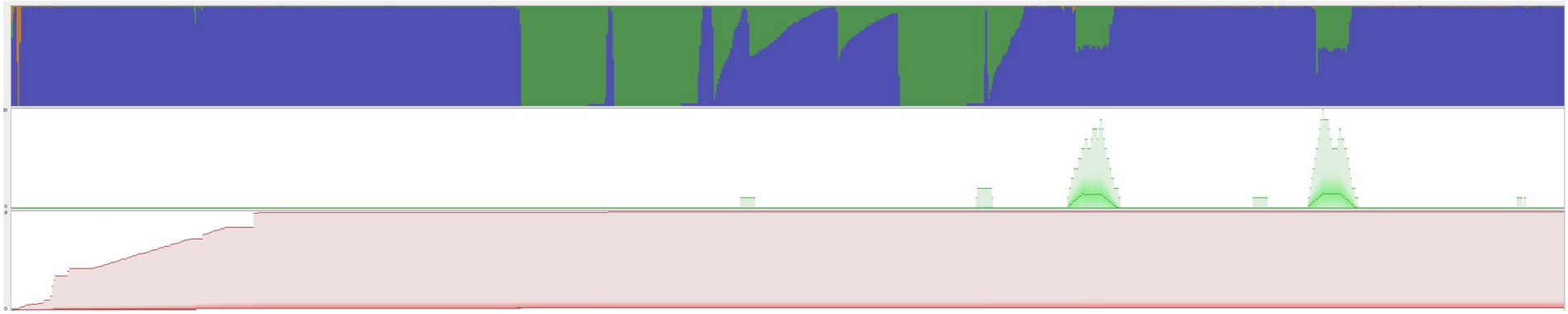
Although it was originally designed to be used within the WRF model, it has evolved over time to possess many additional attributes.

WRF-Hydro has adopted a 'community-based' development processes.

An important aspect of WRF-Hydro has been serving as the core model for the new **National Water Model**.

NWM Analysis Cycle: Before cache opts

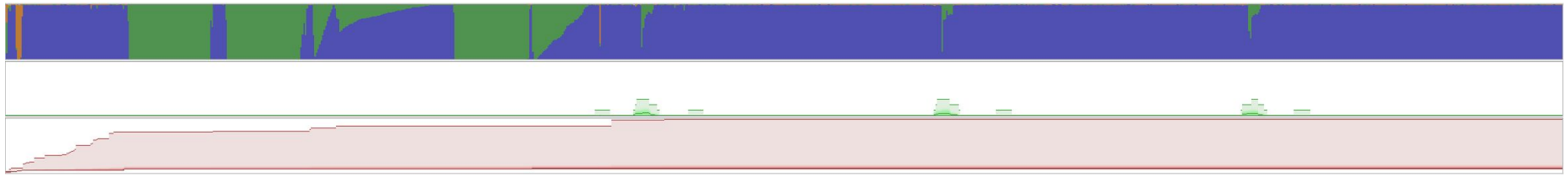
500 processes over 14 nodes on Cheyenne - 560 sec



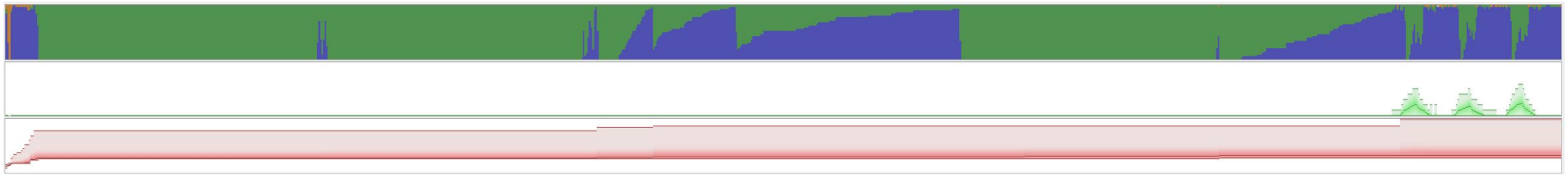
32 processes over 2 nodes on Cheyenne - 1 h 10 m

NWM Analysis Cycle: After Cache Optimization

500 processes over 14 nodes on Cheyenne - 238 sec

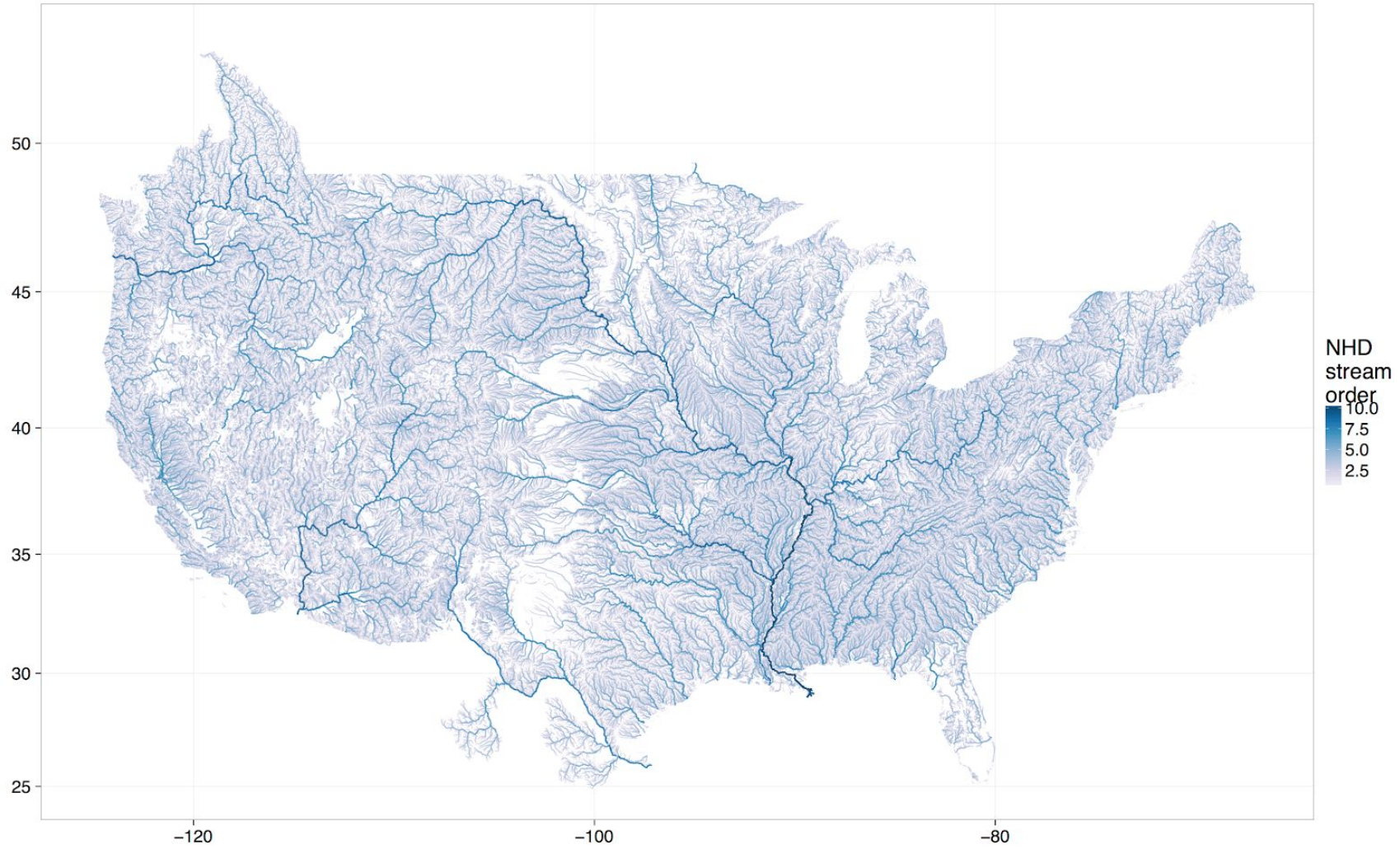


32 processes over 4 nodes (8 procs per node) on Cheyenne - 18 min



Speedup 500 vs. 32 processes ~ 4.5X
Linear speedup supposed to be ~ 15.6X

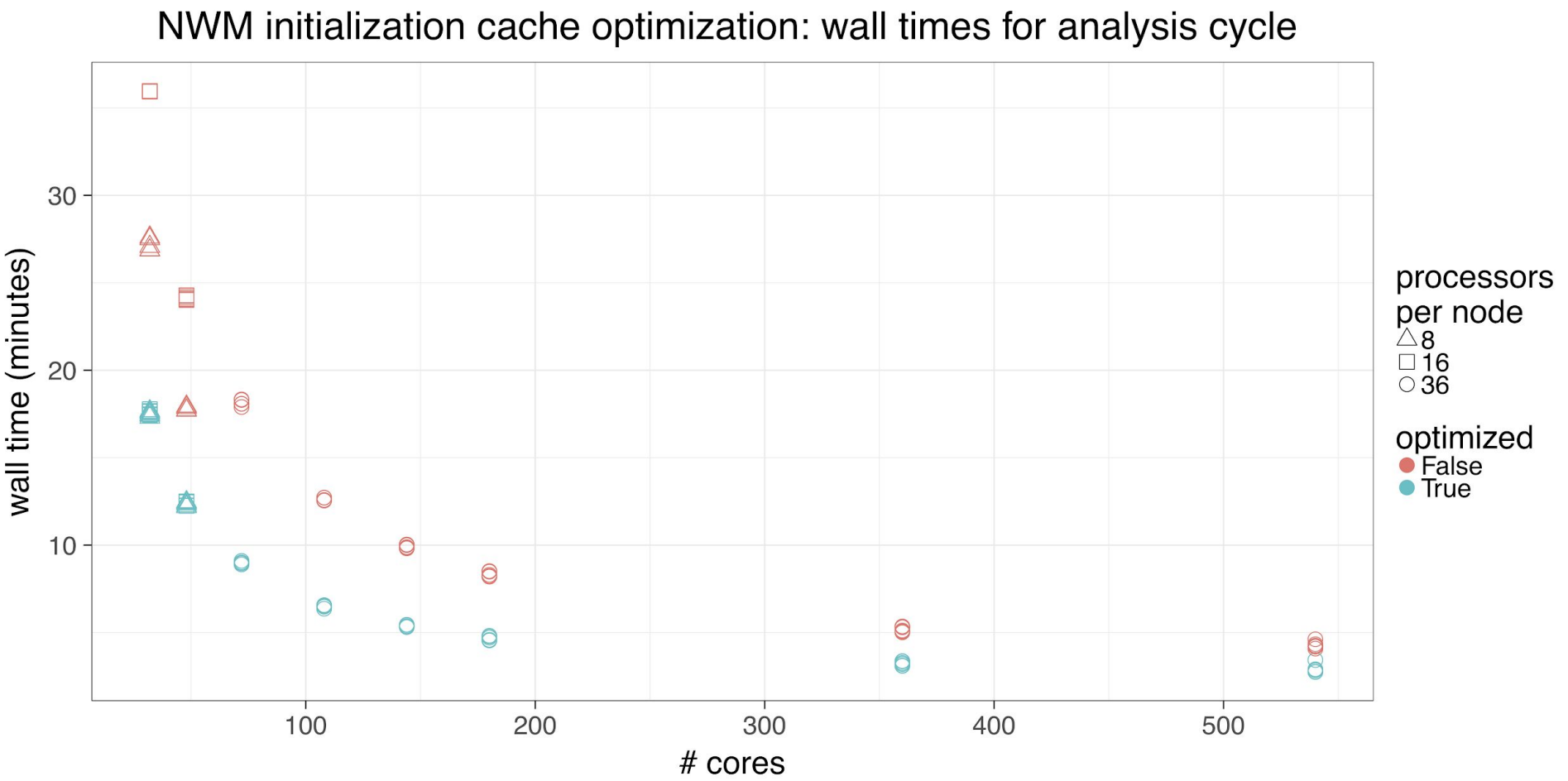
USA River Map



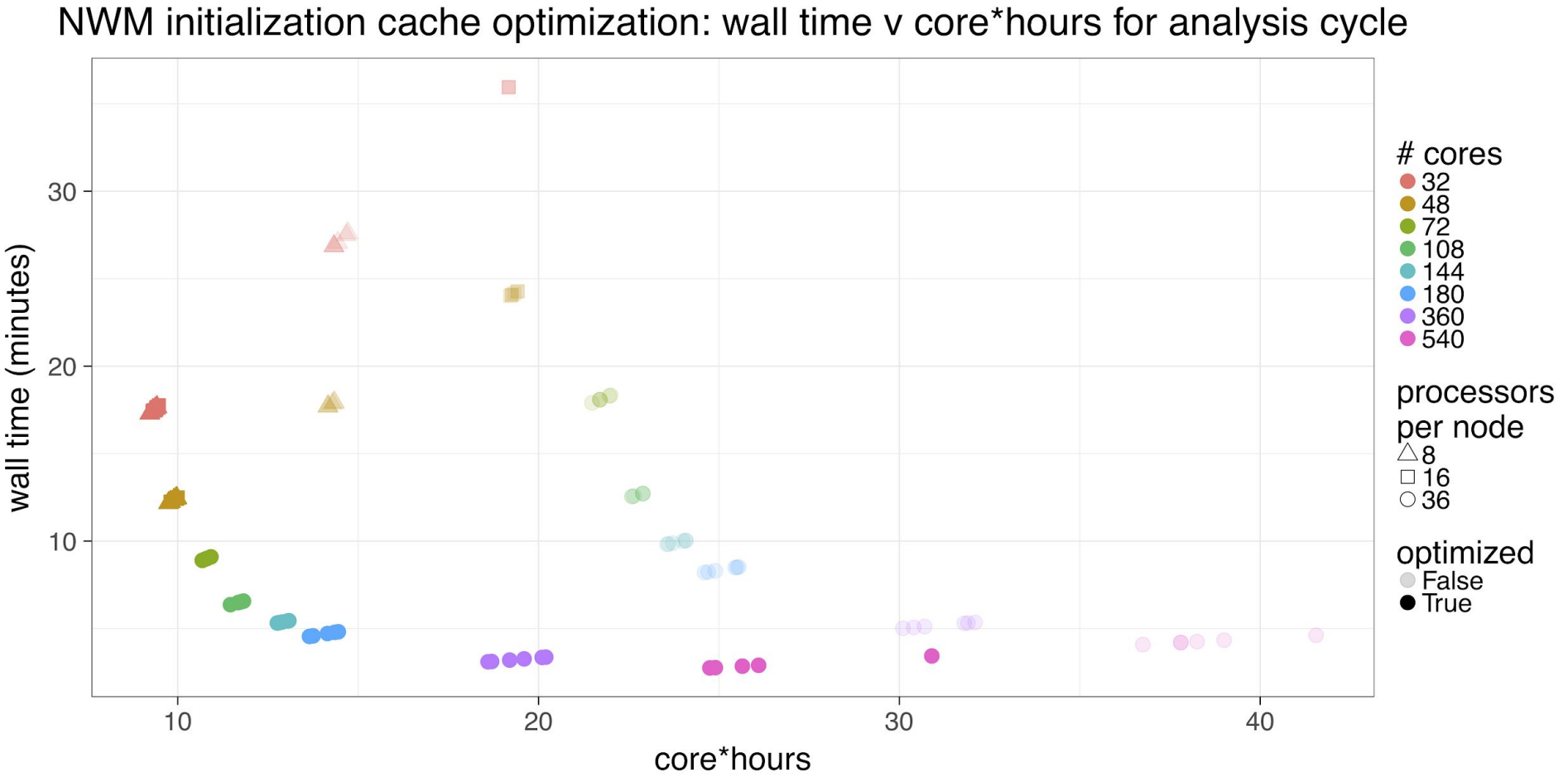
Improving the "Ramps"

- The lack of a better data structure to represent the channel topology forces the code to scan the global links list several times.
- Unbalanced situations penalize the performance (a lot!).
- All the "Ramps" represent unbalanced situations.
- There are two possible solutions:
  a. using a multi-threaded solution (OpenMP)
  b. changing the data structures

Walltime vs. Core count



Walltime - core*hours



Improving the IO idle time

- Process 0 is in charge of computing AND dealing with IO.
- This double duty on process 0 slows down the entire application.

There are two (and a half) possible solutions...:

1. Implementing parallel IO.
2. Creating two groups of processes, one group for IO and one for computing (asynchronous).

2 ½. **Overlap communication with IO**

Strategies for Asynchronous Progress

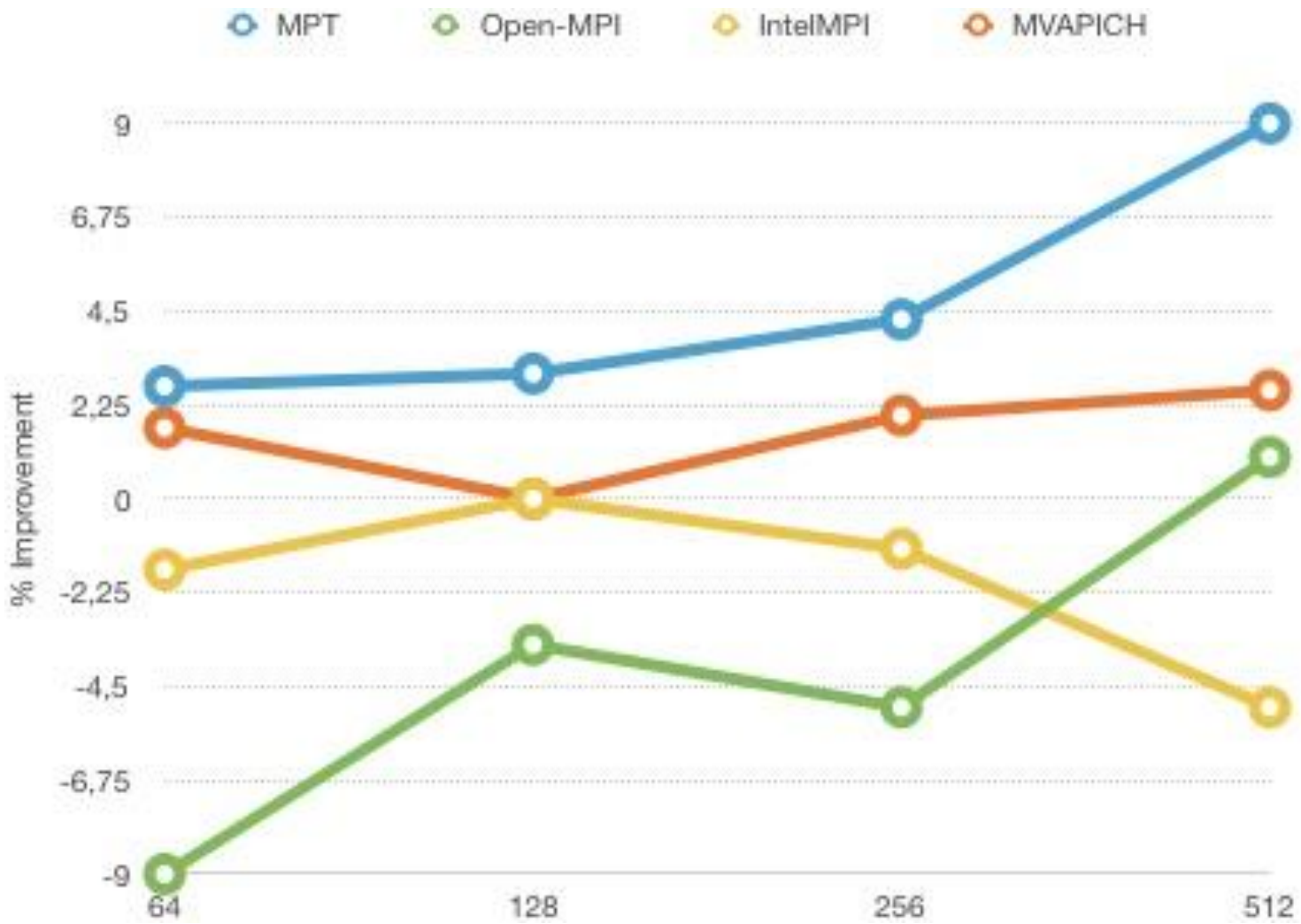
1. Manual Progress
2. Thread-based Progress (polling-based vs. interrupt-based)
3. Communication offload

Manual progress is the most portable but requires the user to call MPI functions (e.g. MPI Test).

Using a dedicated thread to poke MPI requires a thread-safe implementation and implies lots of resource contention and context switching.

Offloading does not look promising and may become a bottleneck because of the low performance of embedded processor on NIC.

"Improvements" - 4 Advances



Improvements - 12 Advances

	MVAPICH	MPT (intel)
% Improvement	1,5	5

Conclusions and Future Development

- MVAPICH2 looks promising (needs some tuning on Cheyenne).
- Overlapping communication and IO represents a good **non**-solution for WRF-Hydro.
- Need to implement some sort of parallel IO.
- Looking forward to using MVAPICH2-X-2.3 (better async progress).

Thanks!

Please, ask your questions!