Extremely Large, Wide-Area Power-Line Models

Ross Adelman
US Army Research Laboratory
ross.n.adelman.civ@mail.mil

Introduction

- The United States spent $381 billion in 2016 for electricity according to the U.S. Energy Info. Admin.; that's more than a billion dollars a day!
- Stand-off, non-contact electric- and magnetic-field sensors can be used to estimate line geometry, voltages, and currents on power lines
- These values carry an immense amount of information about the power grid and power use, which can be used to:
  - improve the stability, conserve power, and reduce outages on national grids and tactical microgrids
  - detect sagging lines and encroaching trees ready for trimming
  - map downed or damaged lines, transformers, and other electrical equipment during disaster recovery or nation building efforts

Objectives

- Construct extremely large, wide-area power-line models to explore these application areas and design algorithms for them
- Develop **multi-node, boundary element method (BEM)** software that runs on the **ARL Excalibur and Centennial** supercomputers for solving such large models

Sample Terrain Modeled
49.5 km^2 northeast of Wash., DC near the ARL campus (2 elements/m^2)

Power Lines Modeled
9.9 km of four 230-kV, three-phase transmission circuits running SW-NE (119 k elements)

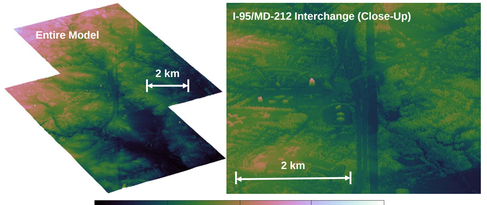
~100 M elements combined

Challenges

- The magnetic field can be computed directly from the line currents
- Computing the electric field requires solving the Laplace equation with the line voltages and grounded terrain as boundary conditions
- Single-node BEM software restricted to 10 million elements or less on standard memory compute nodes with 128 GB of memory
- Extremely large, wide-area power-line models will require hundreds of millions or billions of elements, so software must be parallelized across 10–1000+ compute nodes

Power-Line Models

- Terrain was modeled at 1-m resolution using LIDAR data courtesy of the Army Geospatial Center
- Locations of power poles were extracted using Google Maps
- Power lines were estimated between the power poles

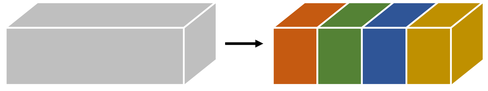


Approach: Domain Decomposition Method

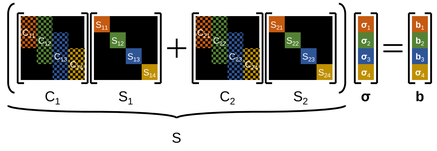
- The domain is decomposed into multiple subdomains, and each subdomain is assigned to a different compute node
- The BEM is used to solve each subdomain, and the solutions are coupled together to enforce continuity along the shared boundary
- Coupling is accomplished using the **coupling matrices**, C_1 and C_2, which yields a large system of equations, solvable through one global solution process via GMRES
- All MVPs involving the dense matrices, S_1 and S_2, arising in the BEM are accelerated using the **fast multipole method** to achieve linear scaling
- Software is also accelerated using the **GPU**

Parallel Matrix-Vector Product (MVP)

- Consider the following example problem, which has been decomposed into four subdomains:

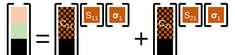


- The linear system for this example problem takes the form:

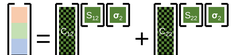


- C_1 and C_2 are highly sparse (one or two nonzero entries per row, shown using checkboard colors); and any blocks away from the diagonal are all zero unless the two corresponding subdomains share a boundary
- The computation of the MVP, S**σ**, is distributed across P nodes by taking advantage of the block nature of the system matrix:

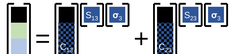
Performed on Node 1:



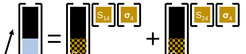
Performed on Node 2:



Performed on Node 3:



Performed on Node 4:

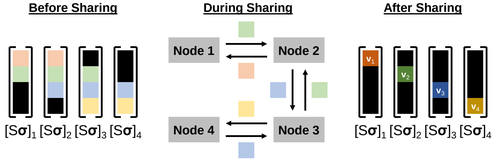


Domain 4 does not share a boundary with domains 1 and 2, and so does not affect these parts of the MVP. This happens for any two subdomains that do not share a boundary

- The total MVP, **v** = S**σ**, is the sum of these partial MVPs (shown using pastel colors):

Parallel MVP, Cont'd

- The entire MVP is not collected and stored in any one location, but rather distributed across the nodes
- The nodes share parts of their partial MVPs with each other:



- Each node only communicates with their immediate neighbors, so communication is done entirely in parallel, minimizing I/O overhead and giving good scaling performance
- In this example problem, each node has only one or two neighbors

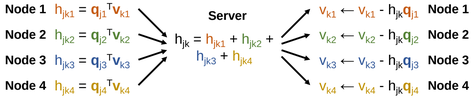
Parallel GMRES

Previous basis vectors in the Krylov subspace

Newly computed vector that must be orthogonalized

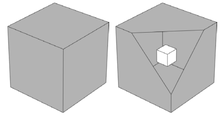
- The Arnoldi process is parallelized across all of the nodes
- For example, $\mathbf{v}_k$ is orthogonalized against $\mathbf{q}_j$ in the following manner:

Step 1: Each node computes a partial dot product
Step 2: The server computes the full dot product
Step 3: Each node removes their part of $\mathbf{q}_j$ from $\mathbf{v}_k$



- A diagonal preconditioner was used to improve scaling of system matrix entries and speed up convergence
- The relative tolerance was set to 10^{-5}; all problems converged
- GMRES was restarted after every 100 iterations with unlimited restarts

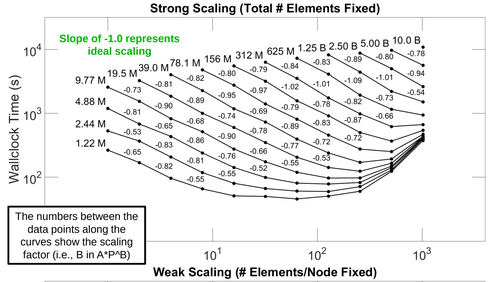
Scalability Testing

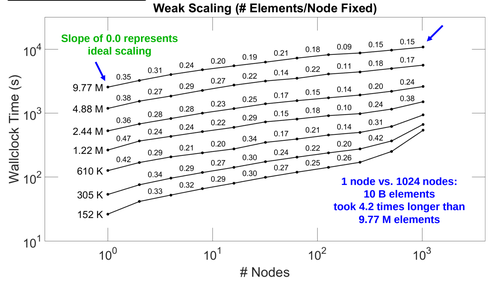


- Characterized the scalability of the method using a standard problem: energized cube(s) inside a grounded box (one pair per compute node)
- Investigated two types of scaling:
  - **strong scaling**: the total number of elements is held constant
  - **weak scaling**: the number of elements per node is held constant
- **Largest test problem had 10 billion elements, and was solved in 3.0 hours across 1024 nodes**
- All results are from ARL Centennial, an SGI ICE XA system, which has:
  - 1,783 standard memory nodes (128 GB each)
  - 32 GPU nodes (256 GB and one NVIDIA K40 each)
  - 32 big memory nodes (512 GB each)
  - Enhanced Data Rate InfiniBand network (100 Gbps) with a fat tree topology

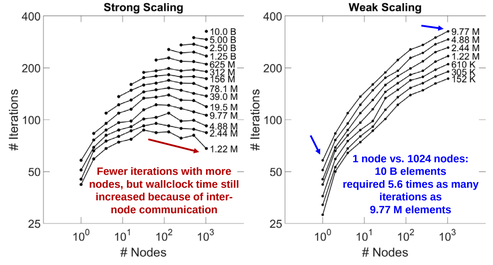
UNCLASSIFIED // APPROVED FOR PUBLIC RELEASE

Results of Scalability Testing





- Wallclock times include setup, solution via GMRES, and evaluation
- Scaling is very good, but not ideal
- Attributable to the behavior of GMRES: in general, larger problem sizes and node counts require more iterations to converge

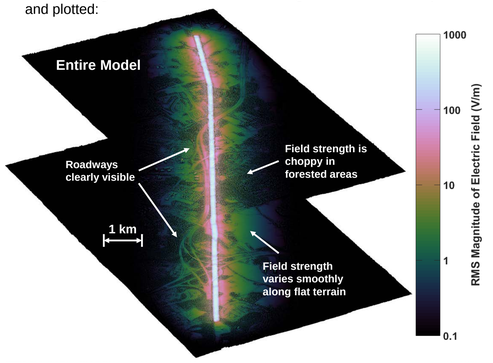


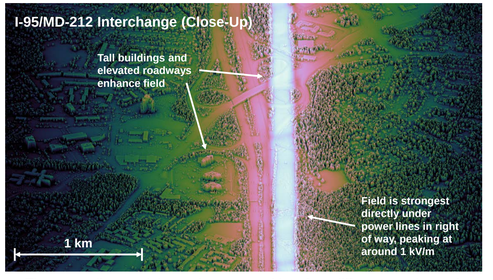
Discussion of Scalability Results

- Very good scaling over large range of problem sizes and node counts
- Scaling performance is best when the number of elements per node is high, i.e., memory use is maximized on each node
- Appears to be scalable well beyond 1024 nodes
- Better scaling may be possible with more than one subdomain per node
- Results shown are from running on ARL Centennial, but similar results were observed when running on ARL Excalibur, a Cray XC40 system

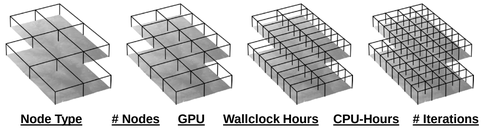
Results of Power-Line Models

- The rms magnitude of the electric field along the terrain was computed and plotted:





- The power-line model was decomposed into four different numbers of subdomains to study scaling performance on other types of nodes:



Node Type	# Nodes	GPU	Wallclock Hours	CPU-Hours	# Iterations
Big Memory	7	--	8.70	2436	263
GPU	14	Off	4.91	2750	267
GPU	14	On	3.82	2139 **(Best)**	267
Standard Memory	28	--	2.91	3259	278
Standard Memory	56	--	1.68 **(Best)**	3763	270

Discussion of Power-Line Results

- A 100-million-element model of four 230-kV transmission circuits over 49.5 km^2 of terrain was solved in 1.68 hours
- Solved on three different types of nodes (standard, GPU, and big memory) with and without the GPU enabled
- Future work will include adding more transmission and distribution circuits, and extending the model to include more land area (e.g., the entire Washington, DC metro area)