HDF5 for Parallel I/O in High Resolution 3-D Global FDTD Earth-Ionosphere Waveguide Models

Santosh Pokhrel [*] [(1)], Miguel Rodriquez [(1)], Alireza Samimi [(2)], Gerd Heber [(3)] and Jamesina Simpson [(1)]
(1) The University of Utah, Salt Lake City, UT 84102
(2) Nanometrics Inc., San Francisco Bay, California
(3) HDF group, the University of Illinois at Urbana-Champaign, IL, 61820

Three-dimensional finite-difference time-domain (FDTD) models of the global Earth-ionosphere waveguide were initially used only at extremely low and ultra-low frequencies, for applications ranging from Schumann resonances, space weather hazards to electric power grids, and hypothesized earthquake precursors. In recent years, the grid resolution of these models has increased to one kilometer and higher so that very low frequencies on a global scale using FDTD (A. Samimi and J. J. Simpson, *Antennas and Wireless Propag*., 15, 1959-1962, 2016). At these high resolutions, efficient parallel input/output operations from all of the processors of the supercomputer are required to keep the total simulation time to as low as possible. However, generating optimal parallel I/O schemes is challenging due to the spherical-coordinate global mesh that includes merging of cells in the East-West direction as either Pole is approached.

In this work, the Hierarchical Data Format 5 (HDF5) is developed as a parallel I/O scheme for a high-resolution global latitude-longitude FDTD model. HDF5 is chosen because of its wide availability, its effectiveness in storing and retrieving large quantities of scientific data, and its compatibility with high performance supercomputing modules.

First the application of HDF to read/write data from/to a spherical-coordinate global FDTD model is discussed. HDF5 is used to input the Earth's topography, the ionospheric composition, and the geomagnetic field. HDF5 is also used to output data such as sampled electric and magnetic fields over time at specific locations of interest, and spatial snapshots of the grid at different instances in time. Next, the performance of HDF is compared with two other approaches: (1) using MPI commands such as MPI_BROADCAST (for input) and MPI_REDUCE (for output); and (2) having each processor read / write its own data in turn. After running various performance tests, it is found that HDF greatly outperforms the other methods at high resolutions.