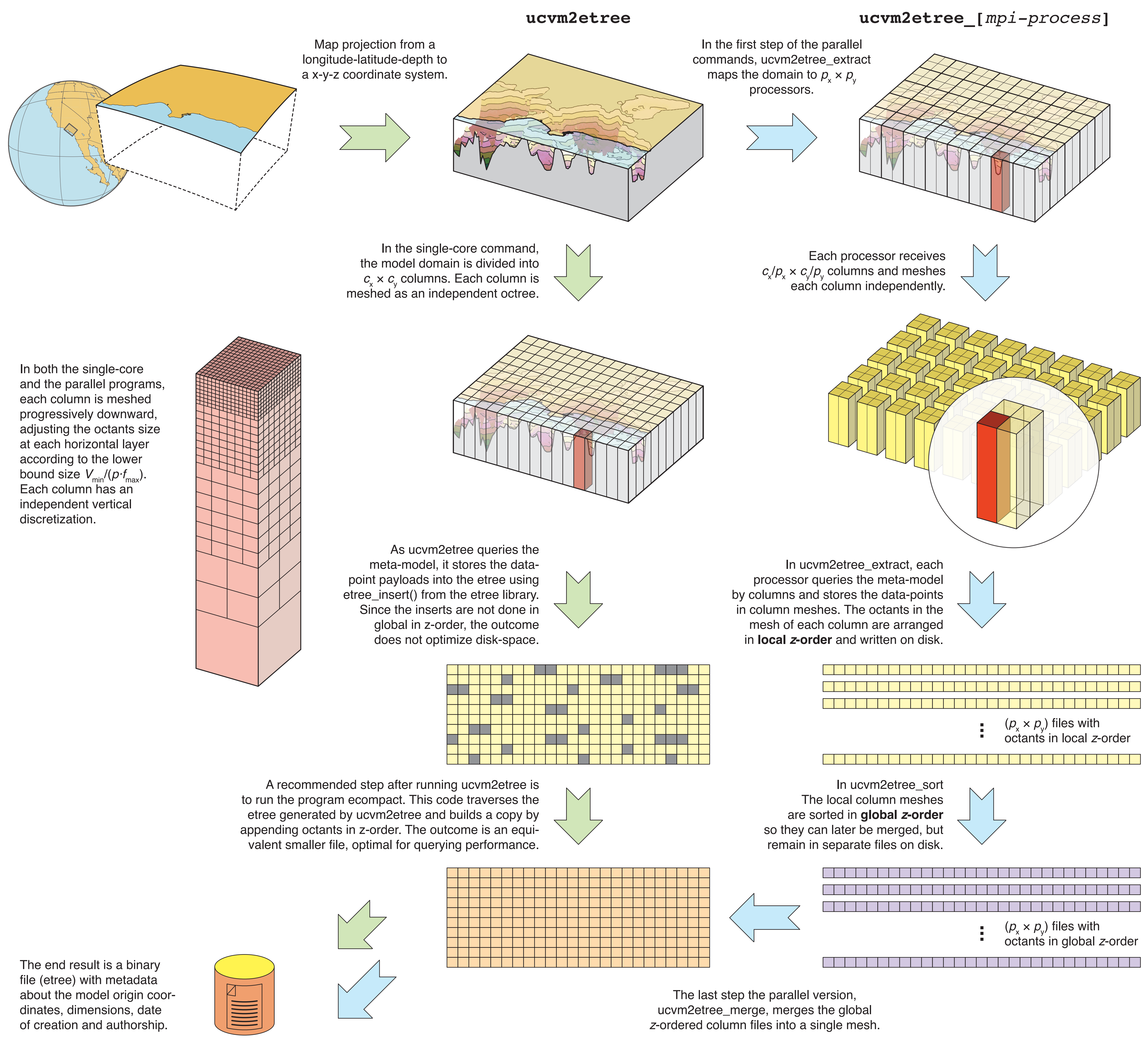

ucvm2etree
ucvm2etree_[mpi-process]
Map projection from a longitude-latitude-depth to a x-y-z coordinate system.
In the first step of the parallel commands, ucvm2etree_extract maps the domain to $p_x \times p_y$ processors.
In the single-core command, the model domain is divided into $c_x \times c_y$ columns. Each column is meshed as an independent octree.
Each processor receives $c_x/p_x \times c_y/p_y$ columns and meshes each column independently.
In both the single-core and the parallel programs, each column is meshed progressively downward, adjusting the octants size at each horizontal layer according to the lower bound size $V_{min}/(p \cdot f_{max})$. Each column has an independent vertical discretization.
As ucvm2etree queries the meta-model, it stores the data-point payloads into the etree using etree_insert() from the etree library. Since the inserts are not done in global in z-order, the outcome does not optimize disk-space.
In ucvm2etree_extract, each processor queries the meta-model by columns and stores the data-points in column meshes. The octants in the mesh of each column are arranged in **local *z*-order** and written on disk.
$(p_x \times p_y)$ files with octants in local *z*-order
A recommended step after running ucvm2etree is to run the program ecompact. This code traverses the etree generated by ucvm2etree and builds a copy by appending octants in z-order. The outcome is an equivalent smaller file, optimal for querying performance.
In ucvm2etree_sort The local column meshes are sorted in **global *z*-order** so they can later be merged, but remain in separate files on disk.
$(p_x \times p_y)$ files with octants in global *z*-order
The end result is a binary file (etree) with metadata about the model origin coordinates, dimensions, date of creation and authorship.
The last step the parallel version, ucvm2etree_merge, merges the global *z*-ordered column files into a single mesh.