Logical Tiling in An AMR Framework: Implementation and Performance

Weiqun Zhang, Ann Almgren (BoxLib); Anshu Dubey, Daniel Graves (Chombo)*
LBNL

To address performance challenges that will accompany next generation node architectures based on many-core processors with NUMA domains, we have introduced tiling into BoxLib and Chombo. Tiling is a well-known loop transformation that can improve both serial and parallel performance of structured grid codes.

BoxLib and Chombo

- BoxLib and Chombo are mature, publicly available software frameworks for building massively parallel block-structured AMR applications.
- Refinement in time and space
- Implemented as layered C++ / Fortran.
- High-performance implementations using hybrid parallelism: MPI + OpenMP

Serial Speed-up on a Single Core of Edison

<table>
<thead>
<tr>
<th>Tile Size</th>
<th>GNU compiler Time(s)</th>
<th>Intel compiler Time(s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 x 4 x 4</td>
<td>8.5</td>
<td>8.7</td>
<td>1.8</td>
</tr>
<tr>
<td>128 x 8 x 8</td>
<td>9.0</td>
<td>9.6</td>
<td>1.6</td>
</tr>
<tr>
<td>128 x 16 x 16</td>
<td>9.6</td>
<td>10.5</td>
<td>1.5</td>
</tr>
<tr>
<td>128 x 32 x 32</td>
<td>23.7</td>
<td>10.4</td>
<td>1.5</td>
</tr>
<tr>
<td>128 x 64 x 64</td>
<td>24.4</td>
<td>10.9</td>
<td>1.4</td>
</tr>
<tr>
<td>no tiling</td>
<td>28.6</td>
<td>15.5</td>
<td>–</td>
</tr>
</tbody>
</table>

Implementation in BoxLib

```cpp
tool tiling = true;
// Loop over tiles rather than grids
for (MFIter mfi(mf, tiling); mfi.isValid(), ++mfi) {
    // Define the tile of this iteration. This tile, rather than the grid that the tile
    // is a part of, will be used to define the extent of the data that the subroutine will
    // operate on.
    const Box& tbox = mfi.tilebox();
    // Get a reference to the FArrayBox so that we can access both the data and the size
    // of the FArrayBox. The FArrayBox itself is unchanged by using tiling.
    FArrayBox& fab = mf[mfi];
    // Define double* pointer to the data of this FArrayBox.
    double* a = fab.dataPtr();
    // Define abox as the Box on which the data in the FArrayBox is defined.
    // This is also unchanged by using tiling.
    const Box& abox = fab.box();
    // We can now pass the information to a Fortran routine, using the
    // index information from tbox rather than abox to specify the work region.
    f(tbox.loVect(), tbox.hiVect(), a, abox.loVect(), abox.hiVect());
}
```

Tiling

- Logical tiling decreases working set size → reduces cache misses → improves single-thread performance
- Logical tiling enables more effective use of threads on many-core architectures
- Regional tiling will manage data locality to address NUMA issues

AMR + Tiling

The fact that a tiling strategy must work in the context of complex multiphysics applications on adaptive grid hierarchies dictates three necessary features. The tiling strategy must
- work for a union of grids that are not necessarily of equal size and shape, and that do not necessarily span the entire rectangular domain
- be such that the tile size can be modified depending on the nature of the loop, as different parts of the algorithm may have very different computational and communication demands.
- be sufficiently lightweight to adapt to the frequently changing grid structure at all levels but the coarsest as the simulation evolves.

Future Plans

- Performance testing of logical tiling framework for large-scale Nyx calculations on Edison and Cori
- Optimal tile size for different algorithmic components?
- Explore different execution models with details hidden in tile iterator.
- Further development includes incorporation of regional tiling through integration of TiDA library
- *Joint work with Tan Nguyen, John Shalf (LBNL) and Didem Unat (Koc Univ.)*