Dynamic Load-Balancing of Reacting Multiphase Flows

V. Azizi¹, G. van den Oord¹, M. Fathi Azarkhavarani², S. Hickel²

Imbalance of computational cost

The bulk of the grid cells take less than a millisecond to progress the reaction stage of the time stepping, but for a few mesh points, solving the stiff system of equations for chemical reactions is orders of magnitude more expensive. This histogram tail is represented by the black regions in the jet cross section view above. The same problem occurs for the solution of the equation of state for vapor-liquid mixtures. Furthermore, the imbalance is highly dynamical for transient setups like Spray-A, and grid cell compute costs are often unpredictable.

Balancing partitioning strategies

To achieve maximal efficiency and allow frequent re-partitioning we follow these guiding principles:

- Minimize global communication during the re-partitioning construction.
- Trade partition quality for partition speed to certain degree.

We have created the following algorithms for quickly finding sufficient partitions:

- **Greedy**: Calculate average compute time per processor (global); calculate load imbalance for each processor (local). As it proceeds, we allow to processor (global) to understand, request work from processor (local) if requested, and process (local) with no request. Only if requested (global), send (local) compute time to (global).

- **Sort**: Calculate average compute time per processor (global); sort load imbalance of each processor (local). If processor (local) on node (global) is underloaded, request work from processor (local) if available. If node (global) is overloaded but processor (local) is underloaded, offload work to node (global) if available. Only if requested (global), send (local) compute time to (global).

- **Sort2**: Sort 200 processors on the local cluster node. If node (global) requests work, all processors send (local) compute time to the remaining node (global).

Load imbalance improvement

We have compared the load imbalance (LI) per processor for the various strategies. Without load balancing, we see a maximum LI of 28.

The violin shapes in the figure left represent various distributions of the LI per processor after dynamically balancing the computation of the chemistry in Spray-A. The maximum LI is reduced to about 1.13 with our load-balancing strategy.

We use the SORT method slightly outperforming GREEDY, whereas the intra-node communication stimulating algorithm SORT2 performing sub-par.

Conclusion

The strong scaling plot left shows a consistent speedup for the LB strategies, and an overall speedup of around 2x. The SORT method, which restricts node-crossing communication, is overall slower than the simpler algorithms, indicating that networking bandwidth has no significant impact for this case. This result can be further improved if more chemically reacting species are added to the simulation.

We are currently generalizing our methods into a new library, QUICKCLUB, to be used within any application that suffers from large computational dynamical imbalances w.r.t. the mesh partitioning.

Feel free to contact us: Victor Azizi (v.azizi@esciencecenter.nl), Gijs van den Oord (g.vandenoord@esciencecenter.nl)