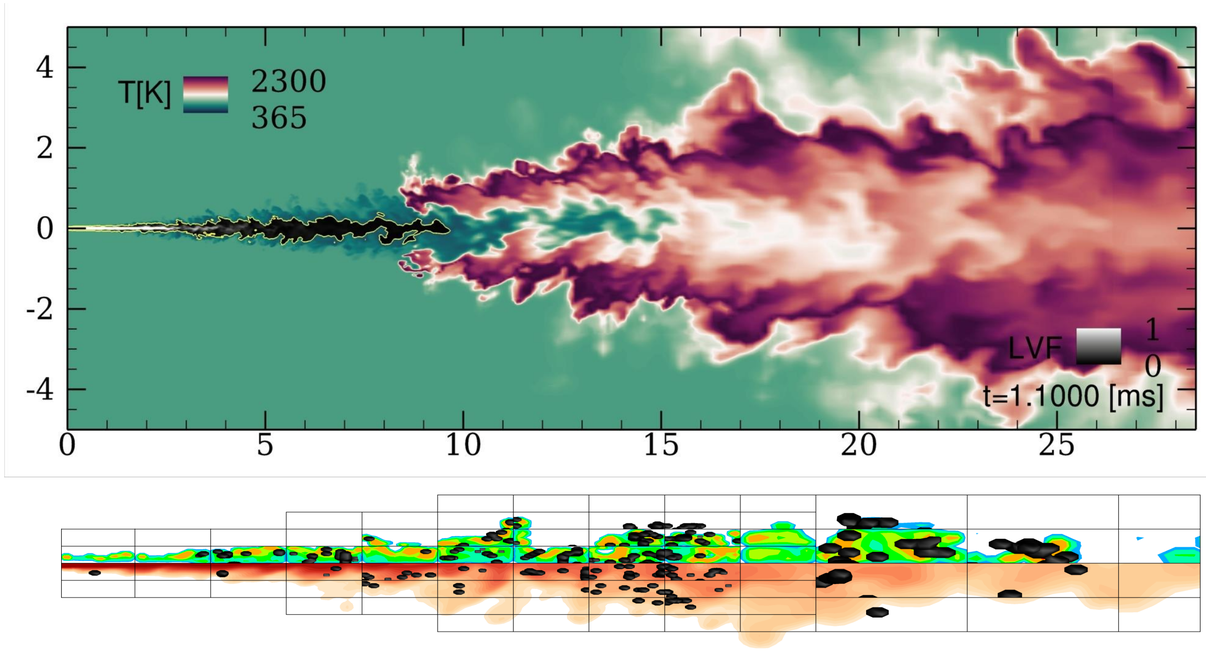
EuroHPC Summit Week 2022
#PRACEdays

Dynamic Load-Balancing of Reacting Multiphase Flows

V. Azizi[1], G. van den Oord[1], M. Fathi Azarkhavarani[2], S. Hickel[2]

[1] netherlands eScience center [2] TU Delft



Reacting Spray-A case

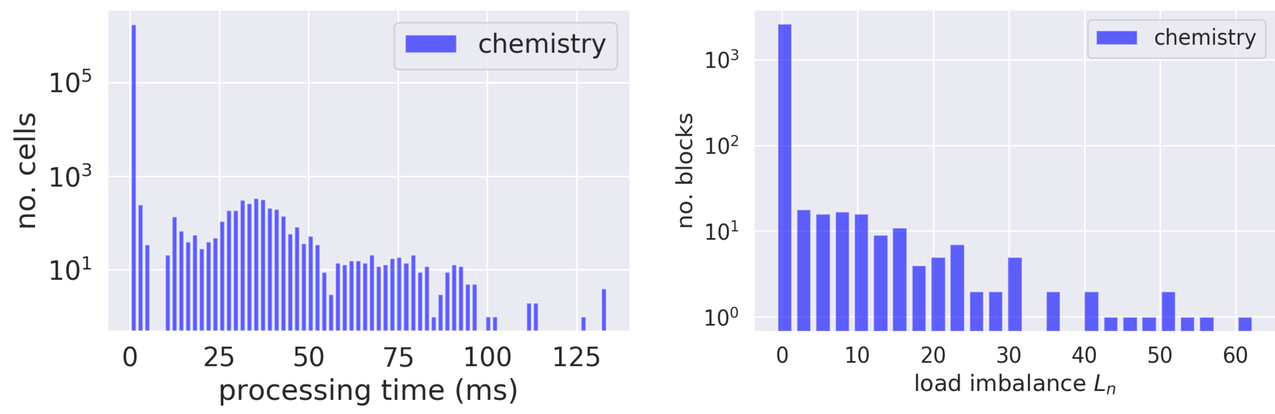
The ECN Spray-A is a benchmark case for the turbulent mixing of cold fuel jets with a hot high-pressure environment, and includes condensation, evaporations, and chemical reactions. Balancing the computational load caused by the spatially and temporally scattered multi-phase, chemically reacting regions is intractable with traditional mesh partitioning. To mitigate this problem in INCA, an adaptive-mesh multi-physics code, we use dynamic load-balancing to redistribute compute-intensive mesh points for both chemical reactions and thermodynamic equation of state solutions independently.

On the left, we visualize a snapshot of the ECN Spray-A benchmark in which a fuel jet is injected into a gas-filled cylinder, leading to scattered ignition cores with high compute intensity. Below we display the computational cost; the square domains denote the traditional mesh partitioning, the lower red-to-white background displays the concentration of N-Dodecane, while the upper half displays the calculation time of the chemical reactions. The black spots represent regions with chemical reactions that take more then 10ms to evaluate.

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 process  (global)
calculate load imbalance for each process   (local)
do n = 1..nprocs
    if overloaded, try to offload to processor (self+n)
    if underloaded, accept work from processor (self-n)
    Recalculate load imbalance for each process (local)
    Stop if max(Load Imbalance) < Stop Value
```

Sort

```
calculate average compute time per process  (global)
calculate load imbalance for each process   (local)
do n = 1..nprocs
    proc_sorted = processors sorted on load imbalance
    m = processor counting backwards in proc_sorted
    if overloaded, try to offload to processor (self+m)
    if underloaded, accept work from processor (self-m)
    Recalculate load imbalance for each process (local)
    Stop if max(Load Imbalance) < Stop Value
```

Sort2

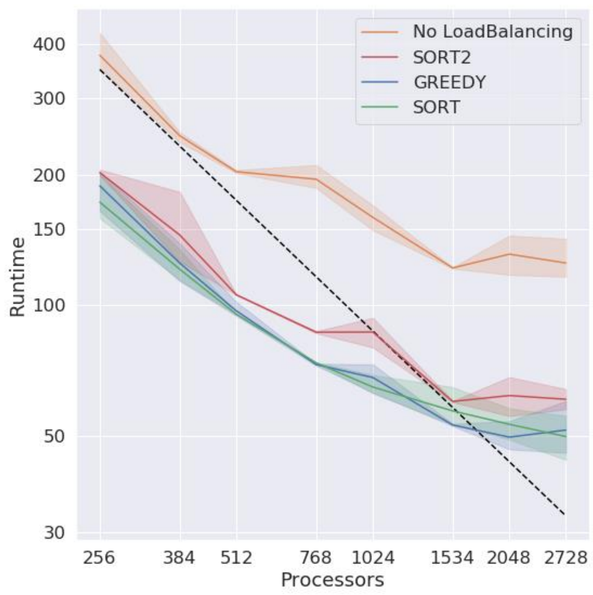
```
Execute SORT algorithm on the local cluster node
If needed, execute SORT again over all processors using the remaining
load imbalance
```

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 loadbalancing strategy.

We see 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 SORT2 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, QUICKLB, 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)

The EuroHPC Summit Week conference series receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreements 823767 (PRACE-6IP) and 824151 (HPC-GIG).

#EHPCSW