Multi-Threaded Hybridization of the OpenFOAM Linear Algebra Libraries
(A story of prototype implementations and future intentions)

Massimiliano Culpo\textsuperscript{1,2}

\textsuperscript{1}CINECA - SuperComputing Applications and Innovation Department
\textsuperscript{2}PRACE1IP - Partnership for Advanced Computing in Europe
Motivation to the work

PRACE 1IP - Task 7.2

... will provide petascaling expertise to ensure that key application codes can effectively exploit Tier-0 systems. Will identify opportunities to enable applications through engagement with selected scientific communities, industrial users and specific application projects ...

PRACE 1IP - Task 7.5

... will work with users to implement new programming techniques, paradigms and algorithms for Tier-0 systems, which have the potential to facilitate significant improvements in the performance of their applications. This task will work in close collaboration with tasks 7.1 and 7.2 ...
## icoFoam: Cavity Flow Benchmark

<table>
<thead>
<tr>
<th>Task #</th>
<th>Time</th>
<th>% MPI</th>
<th>Time</th>
<th>% MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19259</td>
<td>-</td>
<td>18565</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>4005</td>
<td>2.56%</td>
<td>4020</td>
<td>2.43%</td>
</tr>
<tr>
<td>24</td>
<td>2608</td>
<td>4.10%</td>
<td>2616</td>
<td>4.36%</td>
</tr>
<tr>
<td>48</td>
<td>1326</td>
<td>6.88%</td>
<td>1311</td>
<td>6.05%</td>
</tr>
<tr>
<td>96</td>
<td>482</td>
<td>10.24%</td>
<td>469</td>
<td>9.19%</td>
</tr>
<tr>
<td>192</td>
<td>218</td>
<td>20.68%</td>
<td>210</td>
<td>18.71%</td>
</tr>
<tr>
<td>384</td>
<td>123</td>
<td>37.17%</td>
<td>107</td>
<td>31.20%</td>
</tr>
<tr>
<td>768</td>
<td>92</td>
<td>68.67%</td>
<td>109</td>
<td>71.09%</td>
</tr>
</tbody>
</table>

**Table:** Scaling results on the $200 \times 200 \times 200$ cells cavity flow test case
icoFoam: Cavity Flow Benchmark

Nothing but facts!

- On-core performance
  - around 500 MFlops
- Lot of time spent in
  - MPI_Allreduce
  - 8 bytes buffer size
- Fit very well to Tier-1
- ... but not to Tier-0
  ↓
- Tuning needed
  - scaling behavior
  - exploit core architecture

**Figure:** Intra-node scaling behavior of the 200 × 200 × 200 cells cavity flow test case
A Glimpse on OpenFOAM Structure

Finite Volume Primer

\[ \int_{Cell} \text{div}(A \nabla p) \, dV := \sum_{Faces} \int_{f} (A \nabla p) \cdot S_f \, dS \]

Data Representation

- Variables may be associated with:
  1. cells
  2. faces
  3. points
- Time discretized independently
A Glimpse on OpenFOAM Structure

Solution Process: Overview

1. Different solvers, same kernel
   - sparse linear systems
   - iterative Krylov methods
   - SpMV multiplication

Task Decomposition

- Zero-Halo layer approach
- Internal Edges
  - treated as BCs
- Usual pattern
  1. Interface initialization
  2. Local computation
  3. Interface update
- Local: Diagonal Block
- Interface: Off-Diagonal Blocks
Preconditioned Conjugate Gradient

Base Algorithm
1: \( r_0 \leftarrow b - Ax_0 \)
2: \( z_0 \leftarrow M^{-1}r_0 \)
3: \( p_0 \leftarrow z_0 \)
4: \( k \leftarrow 0 \)
5: repeat
6: \( \alpha_k \leftarrow \frac{r_k^T z_k}{p_k^T Ap_k} \)
7: \( x_{k+1} \leftarrow x_k + \alpha_k p_k \)
8: \( r_{k+1} \leftarrow r_k - \alpha_k Ap_k \)
9: \( z_{k+1} \leftarrow M^{-1}r_{k+1} \)
10: \( \beta_k \leftarrow \frac{z_{k+1}^T r_{k+1}}{z_k^T r_k} \)
11: \( p_{k+1} \leftarrow z_{k+1} + \beta_k p_k \)
12: \( k \leftarrow k + 1 \)
13: until \( \|r_{k+1}\| < tol \)

Key Operations at Each Step
1. Sparse Matrix-Vector multiplication
2. Preconditioning
3. Scalar products

Zero-Halo Layer Scalar Product

```c
scalar SumProd = 0;
scalar partialSum = 0;
// Local part of the product
for (label ii=0; ii<max; ii++)
    partialSum += f1p[ii] * f2p[ii];
// Gather other tasks contribution
MPI_Allreduce(&SumProd, &partialSum, 1, MPI_SCALAR, MPI_SUM, MPI_COMM_WORLD);
```
Lessons learned so far...

1. The scalar products in the PCG algorithm act as barriers.
2. The whole bunch of MPI_Allreduce:
   - stems from an algorithmic constraint
   - is unavoidable, unless we venture on an algorithmic rewrite
3. How can we reduce communication and preserve the algorithm?
   - add multi-threading capabilities to sparse linear-algebra
4. To do that we have to dig into sparse matrix representation!
LDU Sparse Matrix Format

Storage Format Quick Guide

1. Matrix is considered
   - square
   - structurally symmetric
   - sum of three parts \( A = L + D + U \)

2. Off-diagonal positions mapping
   - lPtr (globally ordered)
   - uPtr (locally ordered)

3. Values stored in three double vectors

4. No fill-in introduced

\[
\begin{align*}
lPtr &= [0 \ 0 \ 1 \ 1 \ 2 \ 3 \ 3 \ 4 \ 4 \ 5 \ 6 \ 7] \\
uPtr &= [1 \ 3 \ 2 \ 4 \ 5 \ 4 \ 6 \ 5 \ 7 \ 8 \ 7 \ 8]
\end{align*}
\]
Sparse Matrix-Vector Multiplication

1 // Diagonal contributions
2 for (register label cell = 0; cell < nCells; cell++)
3 {
4    ApsiPtr[cell] = diagPtr[cell] * psiPtr[cell];
5 }
6 // Off-diagonal contributions
7 for (register label face = 0; face < nFaces; face++)
8 {
9    ApsiPtr[uPtr[face]] += lowerPtr[face] * psiPtr[lPtr[face]];
10   ApsiPtr[lPtr[face]] += upperPtr[face] * psiPtr[uPtr[face]];
11 }
Modifications to LDU Format

What prevents multi-threading?

1. Off-diagonal contributions
   - Concurrent write-access
   - Access by cells needed

2. Owner sort
   - owPtr

3. Losort
   - reshape
   - loPtr

4. Introduce doubly indirect access of rvalues

\[
\begin{align*}
\text{owPtr} & = [0, 2, 4, 5, 7, 9, 10, 11, 12, 12] \\
\text{loPtr} & = [0, 0, 1, 2, 3, 5, 7, 8, 10, 12] \\
\text{reshape} & = [0, 2, 1, 3, 5, 4, 7, 6, 8, 10, 9, 11]
\end{align*}
\]
Sparse Matrix-Vector Multiplication

// Off-diagonal contributions
#pragma omp for
for (label cell = 0; cell < nCells; ++cell)
{
    for (label fidx = owPtr[cell]; fidx < owPtr[cell + 1]; ++fidx)
    {
        AxPtr[cell] += upperPtr[fidx] * xPtr[uPtr[fidx]];
    }
    for (label fidx = loPtr[cell]; fidx < loPtr[cell + 1]; ++fidx)
    {
        AxPtr[cell] += lowerPtr[reshape[fidx]] * xPtr[lPtr[reshape[fidx]]];
    }
}
Can we go further?

1. Cache-Blocking Techniques
   - improvement of on-core performance
   - ...and thus loss on scalability!

2. Change of Matrix Format to
   - hide latency to memory (S-CSR)
   - reduce size of data to be transferred (\(\delta\)-CSR)

3. Ideas to be Developed in PRACE 2IP

Thanks, and have fun in the evening!