Improving Overset Performance within STAR-CCM+

Björn Gmeiner, Eberhard Schreck, Daniel Uhlmann
Siemens PLM Software
Content

- Motivation
- Core-level and Node-level Optimization: Exact Volume Integration
- Parallel Performance Improvements
- Overset Prism Layer Shrinkage
- Examples
Improve Performance on Different Levels

- Core-level and node-level optimization (Exact volume integration)
  Effects of serial optimization techniques and node-level load balancing

- Parallel algorithm
  Redesign for parts of the parallel Overset algorithm

- Reducing overall work (Overset Prism Layer Shrinkage)
  Move cells to where they are needed: lightweight local morphing
Improving Overset Performance within STAR-CCM+

Core-level and Node-level Optimization:
Exact Volume Integration
Performance Enhancements for Exact Volume Integration (Core and Node level)

Avoid double counting
Motivation: Run-time for an Exact Volume Integration

Method:
Cells are chopped up into tetrahedral which are then being intersected against each other (optional).

For some evaluated cases (serial) before optimization:

Polyhedral mesh:
Overset Volume Correction is 60-80% of the total Overset intersection time

Trimmed mesh:
Overset Volume Correction is ~30% of the total Overset intersection time
Why is Exact Volume Calculation Expensive for Polyhedral Cells?

- **Average number of tetrahedra per polyhedral cell**: 30-40
- **Tetrahedron / tetrahedron intersection leading to zero-volume intersections**: ~70%
- **Average number of tetrahedral intersections per polyhedral cell**: ~1000
- **Load imbalance**: Depends much on the partitioning but can be a large factor
Optimization opportunities

- **Average number of tetrahedra per polyhedral cell**: ~35
  Reduce them by choosing another decomposition

- **Tetrahedron / tetrahedron intersection leading to zero-volume intersections**: ~70%
  Implement a fast tetrahedron-tetrahedron collision detection

- **Average number of tetrahedral intersections per polyhedral cell**: ~1000
  Optimize the computational intersections kernels

- **Load Balancing**: Node-wise re-distribution of work
Different Simple Element Decompositions: Polyhedral / Trimmed Meshes

- Good element shapes
  - #Elements: 10

- Worse element shapes
  - #Elements: 8

- Good element shapes
  - #Elements: 4

- Good element shapes
  - #Elements: 2
Optimization of Intersection Kernels

**Problem:**
Hotspot analysis shows the time is spent in many different kernels!

- ~ 25% of the time is spent within **10** hotspots
- ~ 50% of the time is spent within **40** hotspots

**Consequence:**
We have to optimize several hotspots and cannot easily expect an improvement by e.g. an order of magnitude. Basically to get an improvement you are trying to get one percent after another.
Optimization of Intersection Kernels

Anyhow, we tried and optimized by:

- Changing the tetrahedron volume calculation formula
- Avoiding square roots
- Removing small loops
- Removing branches
- Optimizing sorting routines
- Reordering executions
- ...

...
Serial optimizations improvements *(version 12.02)*

- **Improvement for polyhedral meshes:** ~25%
  - ~5% tetrahedra decomposition
  - ~10% fast tetrahedron-tetrahedron collision detection
  - ~10% kernel optimization

- **Improvement for trimmed meshes:** ~60%

Node-wise optimization improvements *(result based on serial optimization)*

- **Improvement (in parallel, 20 cores, version 13.04):** ~50%
  - and ~60% reduction in main memory consumption (using a blocking technique)
Improving Overset Performance within STAR-CCM+
Parallel Performance Improvements (v13.06)
Underlying task: For hole-cutting large surfaces have to be collected (in parallel) in order to perform inside/outside tests.

Previous approach: Collect the (global) surfaces for each process and then perform tests locally.

Drawback: High memory consumption (not scaling for this task) and high amount of communication required

New approach (version 13.06):
- Identify and collect only the surface parts that are locally required.
- Make the test itself “more local”
Parallel Performance Improvement (Planetary Gear, half populated)

![Graph showing parallel performance improvement.](image-url)
Parallel Performance Improvement for Gearbox (motion only !)
Improving Overset Performance within STAR-CCM+

Overset Prism Layer Shrinkage (v13.02)

“Honey I shrunk the Prisms”
Why Prism Layer Shrinkage for Overset?

Object in overset region

Not shrunk

last prism edge

boundary edge

Object in overset region

Shrunk

x: Active cell (not covered by object in the overset region)
Why Prism Layer Shrinkage for Overset?

- Boundary edge and the last prism edge are not getting changed by the shrinker.
- The shrinking algorithm changes the control edge by proposing a new ratio (prism layer stretching) between the two adjacent layers.
- Relaxation edges above or below the control edge are always relaxed towards the original ratio.
Algorithmic procedure

Setup connectivity (once per remeshing/repartitioning):

1. Identify prismatic cells
2. Mark vertices only attached to prismatic cells
3. Number prismatic vertex layers
4. Create vertex connectivities
5. Compute original ratios

Adapt prism layers (once per time-step):

1. Exchange adjacent vertex coordinates (for parallel only)
2. Calculate boundary wall distances to determine new shrinkage ratios
3. Move control vertex and relax other prism vertices
Algorithmic procedure

Setup connectivity (once per remeshing/repartitioning):

1. Identify prismatic cells
2. Mark vertices only attached to prismatic cells
3. Number prismatic vertex layers
4. Create vertex connectivities
5. Compute original ratios

Adapt prism layers (once per time-step):

1. Exchange adjacent vertex coordinates (for parallel only)
2. Calculate boundary wall distances to determine new shrinkage ratios
3. Move control vertex and relax other prism vertices
First Attempts

Use the Overset status (i.e. location of the acceptor cells)!

+ **Cheap!**

- Requires at (least a partial) overset intersection
- Non-accurate distances in space and time resulting in oscillations

Tried to smooth out, e.g. by using:
- nearest neighbour relaxation in space
- moving averages in time (simple, exponential, median)

but were not really happy with the results… *We need something more accurate!*  
=> Full distance based treatment
User Interface

Prism Layer Shrinkage

- Enables the shrinkage of prism layers if the background and overset regions are close to each other.
Examples

Solution Time 0.0001 (s)

Unrestricted © Siemens AG 2018
Page 24 / 35 04.10.2018
Test-geometry (3D): without shrinkage (left), with shrinkage (right)
Test-geometry (3D): without shrinkage (left), with shrinkage (right)
Without shrinkage (top), with shrinkage (bottom). Target: Three cell layers
ZeroGap without shrinkage (Gear pump, 2D)

Volume representation

Overset representation

0.0000 1.4000 2.8000 4.2000 5.6000 7.0000
ZeroGap with shrinkage, minimal gap distance: 0.000125
ZeroGap with shrinkage, minimal gap distance: 0

Volume representation

Overset representation
Lobe Blower (3D) (Mesh motion)
Lobe Blower (3D) (Velocity)
Gearbox VOF (2D)

Solution Time 0.0008 (s)

Solution Time 0.0008 (s)
Gerotor (3D)
Conclusion and Outlook

- Core-level and node-level optimizations significantly helped to improve the performance of the exact volume integration.
- Parallel performance improvements are going to help large Overset cases with respect to run-time and memory requirements.
- Overset Prism Layer Shrinkage now allows quickly setting up geometries involving very small gaps.