Enhanced airflow simulations around filling machines in clean rooms

by: Ralph Eisenschmid, process engineer
PRACE coach: Bärbel Große-Wöhrmann
Excellence in packaging

• 450 employees in pharma sector
• 1700 employees worldwide
• 240 million EUR revenues (consolidated, 2012)
• >Export share 80%

• Development and manufacture of filling and packaging machinery
• Family enterprise in the 3rd generation
• Global structures
• High employee satisfaction
• Excellent qualification level
• Process understanding
• Outstanding commitment to R & D
Expertise
Sterile and non-sterile filling / packaging and well-coating solutions for diagnostics and pharmaceutical products

Products
Fully automated machines for the processing of liquid, pasty and powdery products in bottles, vials, microvials, cartridges and cuvettes. Fully automatic well-coating lines for microtitre plates.

Focus / Speciality
Processing of a wide variety of closure and container shapes on a single machine.
Expertise
Sterilizable freeze drying plants and fully automatic loading and unloading systems for the pharmaceutical industry.

Products
Customized solutions for vials, ampoules, syringes.

Focus / Speciality
Upgrades from pilot plants to production plants with flexible systems and highest level of automation.
Isolation Technology

Expertise
Design, manufacture, installation and validation of customized isolators.

Products
Isolators for liquid / powder filling / lyo loading/unloading systems (aseptic and/or potent), ebeam tunnels for syringe tub sterilization, gastight doors, $\text{H}_2\text{O}_2$ rapid gassing transfer chambers.

Focus / Speciality
M + P isolators are used worldwide to protect high value products during their manufacture, preparation and dispensing from process generated and external contamination.
Benefits

Leading Quality and Variety in Fill/Finish Equipment

- One stop shop / Synergy effect
- Fast and educated decisions for a consistent concept
- Constant improvement of quality
- Customized Turn-Key Solutions

- Shorter time to market
- Better return on investment (ROI)
- Improved economic efficiency
- Global agreements
Our motivation:

- and.....we love SHAPE !!!
Enhanced airflow simulations around filling machines in clean rooms with OpenFOAM

- Setting up of convenient and vast automatic working batch scripts
- Prediction of resources and performance on OpenFOAM's meshing tool snappyHexMesh by varying refinement levels and surface layers
- Meshing scaling tests
- Karman-vortex benchmark by DNS simulation for calibrating stationary and transient turbulent model solvers (RANS and DDES, IDDES)
- Fixing bottle necks: serial decomposition and restriction tools
- Evaluating of best fitting params on turbulence model solvers
- Scaling tests on turbulence model solvers
• Not big science...... a diligent but routine piece of work has been waiting for us!
Materials and Methods

- **Hardware Hermit XE6**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak performance</td>
<td>1.045 PFlops</td>
</tr>
<tr>
<td>Cabinets</td>
<td>38 with 96 nodes each</td>
</tr>
<tr>
<td>Number of compute nodes</td>
<td>3552</td>
</tr>
<tr>
<td>Number of compute cores</td>
<td>per node 2 sockets with 16 cores each: 113 664</td>
</tr>
<tr>
<td>Number of service nodes</td>
<td>96</td>
</tr>
<tr>
<td>Processor compute nodes</td>
<td>Dual Socket AMD Interlagos @ 2.3GHz 16 cores each</td>
</tr>
<tr>
<td>Memory/node</td>
<td>32 GB and 64 GB</td>
</tr>
<tr>
<td>Disk capacity</td>
<td>2.7 PB</td>
</tr>
<tr>
<td>Node-node interconnect</td>
<td>CRAY Gemini</td>
</tr>
<tr>
<td>Special nodes</td>
<td>External Access Nodes, Pre- &amp; Postprocessing Nodes, Remote Visualization Nodes</td>
</tr>
<tr>
<td>Power consumption</td>
<td>2 MW maximal</td>
</tr>
</tbody>
</table>
Materials and Methods

- Simulation Software OpenFOAM 2.2 (http://www.openfoam.org) with Cray Profiler; Postprocessing with Paraview 3.9.8

<table>
<thead>
<tr>
<th>Toolbox</th>
<th>Description</th>
<th>Parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>blockMesh</td>
<td>i.e. 200x100*60 mesh</td>
<td>single</td>
</tr>
<tr>
<td>decomposePar</td>
<td>i.e. 64 processors</td>
<td>single</td>
</tr>
<tr>
<td>snappyHexMesh</td>
<td>i.e. refinement level 4; 2 surface layers</td>
<td>parallel</td>
</tr>
<tr>
<td>reconstructParMesh</td>
<td>reconstructing refined mesh</td>
<td>single</td>
</tr>
<tr>
<td>decomposePar</td>
<td>i.e. 256 processors for solver</td>
<td>single</td>
</tr>
<tr>
<td>CFD solvers</td>
<td>i.e. SpalartAllmarasDDES</td>
<td>parallel</td>
</tr>
<tr>
<td>reconstructPar</td>
<td>reconstructing mesh and fields for postprocessing</td>
<td>single</td>
</tr>
<tr>
<td>foamToVTK</td>
<td>universal use postprocessing format</td>
<td>single</td>
</tr>
</tbody>
</table>
our geometries:

- cube: 1 x 1 x 1 m
our geometries:

- box with Karman cylinder
our geometries:

- inert gas flushing of vials
our geometries:

- Clean Room: Mock-Up model of a RABS (Restricted Area Barrier System = clean room) with aseptic liquid filling machines inside
Meshing cell number prediction for resources and walltime estimation

- **blockMesh**: very simple cell number prediction

n = 10 x 10 x 10 cells = 1000 cells
Meshing cell number prediction for resources and walltime estimation

- snappyHexMesh after \( r=1 \) refinement levels

\[ n = 6 \times 20 \times 20 \times 20 \] surface cells, but inside?

Cells snapped to the STL surface
Meshing cell number prediction for resources and walltime estimation

- snappyHexMesh after $r=2$ refinement levels and $c=3$ adapting layers
Meshing cell number prediction for resources and walltime estimation

- snappyHexMesh after \( r=2 \) refinement levels and \( c=1 \) adapting layer
Meshing cell number prediction for resources and walltime estimation

- snappyHexMesh after $r=2$ refinement levels and $c=4$ adapting layers
Meshing cell number prediction for resources and walltime estimation

- **snappyHexMesh** after \( r = 2 \) refinement levels and \( s = 4 \) surface layers
As one can see, snappyHexMesh refines or splits surface-intersecting cells (i.e. STL CAD geometry) into $2\times2\times2=8$ cells. Here an example in 2D; $2\times2=4$ cells.

http://www.openfoam.org/docs/user/snappyHexMesh.php
• So, every refinement level adds roughly $8^r$ cells to the surface, where $r$ is the actual refinement level.
• For refinement = 1 to $r$ one will expect an expansion in series like..
  $n = 8^1 + 8^2 + 8^3 + 8^4 + \ldots + 8^r$ (...minus original cells)
• Every surface layer $s$ will add ...
  $n = s \cdot (n_s)^r$ (...minus original cells).....
• ....surface layer cells ; where $n_s$ is the number of the final surface cells after last refinement step, but before adding surface layers
• Estimating the number of adapted cells (in snappyHexMeshDict they are called "cells between layers") is not as easy, because these cells have irregular shape like deformed tetraeders and general polyeders
Meshing cell number prediction for resources and walltime estimation

• Introduction of analytical approximations on cell-number equations (detailed derivation omitted):

\[ n(r) = (4^0 + 4^1 + 4^2 + \ldots + 4^r) A/(d_0)^2 \]

A is geometry surface;
d_0 is blockmesh mean cell size

• ...and the sum of this series lead to:

\[ n(r) = (4^{(r+1)} - 1)/(4-1) \times A / (d_0)^2 \]

• valid for meshes with neglected adaption layers and coarse blockmeshes with big specific surfaces:

\[ d_0 A / V > 1 \]

V is geometry volume
Meshing cell number prediction for resources and walltime estimation

- Introduction of empirical approximations on cell-number equations; respecting irregular shaped adapted cells:

\[ n(r) = \frac{(z^0 + z^1 + z^2 + \ldots + z^r)}{d_0^2} \]

where \( z \) is a real number between 4 and 8, that fits a power law regression best on tests performed from \( r=0 \) up to \( r=6 \); \( A \) is geometry surface; \( d_0 \) is blockmesh mean cell size

- ...and after expansion in series:

\[ n(r) = \frac{(z^{r+1} - 1)}{(z-1)} \frac{A}{d_0^2} \]

- respecting small specific surfaces and fine blockmesh:

\[ n(r) = \frac{(z^{r+1} - 1)}{(z-1)} \frac{A}{d_0^2} + \frac{V}{d_0^3} - \frac{A}{d_0^2} \]
simplest approximation of measured cell numbers:

\[ n(r) = \frac{(z^{(r+1)} - 1)}{(z-1)} \frac{A}{(d_0)^2} + \frac{V}{(d_0)^3} - \frac{A}{(d_0)^2} \]
Meshing cell number prediction for resources and walltime estimation

- snappyHexMesh after $r=6$ refinement levels and $c=3$ adapting layers: cell number fits very well to power law equations above

10 Mio cells
Meshing cell number prediction for resources and walltime estimation

• Details on r=6 mesh
Meshing cell number prediction for resources and walltime estimation

- Karman mesh example with r=2
Meshing cell number prediction for resources and walltime estimation

- clean room mesh example with r=2
Meshing cell number prediction for resources and walltime estimation

- clean room mesh example with r=2 - detail
Meshing cell number prediction for resources and walltime estimation

- clean room mesh example with r=2 and s=4
Meshing cell number prediction for resources and walltime estimation

- clean room mesh example with $r=2$ and $s=4$ - detail
Meshing cell number prediction for resources and walltime estimation

- clean room mesh example with $r=2$ and $s=4$ - detail

![Meshing cell number prediction for resources and walltime estimation](image)

not all surfaces built with surface layers!!!
Simulation jobs and their evaluation

- Cray Profiler's output based analysis of resources, walltime and performance of OpenFOAM's toolboxes
- Typical serial tool's profiler output summary:
Simulation jobs and their evaluation

- **typical parallel tool's profiler output summary:**

```plaintext
./profile-O-icoFoam-2153915.sdb.txt

Profile of icoFoam

<table>
<thead>
<tr>
<th>Number of processes</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>System summary</td>
<td>min,</td>
</tr>
<tr>
<td>Wall clock time</td>
<td>3.610,</td>
</tr>
<tr>
<td>Maximum memory usage (MB/proc)</td>
<td>27.117,</td>
</tr>
<tr>
<td>Memory touched (MB/proc)</td>
<td>26.980,</td>
</tr>
<tr>
<td>User processor time</td>
<td>1.628,</td>
</tr>
<tr>
<td>System processor time</td>
<td>0.160,</td>
</tr>
<tr>
<td>Node memory size (MB/node)</td>
<td>32768.000,</td>
</tr>
<tr>
<td>Total I/O time</td>
<td>0.004,</td>
</tr>
<tr>
<td>Total I/O bytes</td>
<td>0.053M,</td>
</tr>
</tbody>
</table>
```

---

Normal text file length: 6707 lines: 159

Ln: 61 Col: 1 Sel: 0

UNIX ANSI INS
Simulation jobs and their evaluation

- **typical snappyHexMesh log file output:**

```plaintext
Snapped mesh: cells:10413080  faces:33997644  points:13132311
Cells per refinement level:

<table>
<thead>
<tr>
<th>Level</th>
<th>Cells</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>1</td>
<td>2232</td>
</tr>
<tr>
<td>2</td>
<td>17352</td>
</tr>
<tr>
<td>3</td>
<td>90792</td>
</tr>
<tr>
<td>4</td>
<td>270016</td>
</tr>
<tr>
<td>5</td>
<td>1840688</td>
</tr>
<tr>
<td>6</td>
<td>8191936</td>
</tr>
</tbody>
</table>

Writing mesh to time constant
Wrote mesh in = 16.78 s.
Mesh snapped in = 45.41 s.
Finished meshing in = 151.07 s.
End

Finalising parallel run
Application 3984236 resources: utime ~9623s, stime ~214s
```
Simulation jobs and their evaluation

• **Lessons learned by this analysis?**

• parallel jobs take only minutes
• serial jobs take hours or days!!!
• so, serial jobs are definitely the hardest bottleneck in total simulation process
• `reconstructParMesh` just can handle meshes up to 30 - 40 Mio cells, before ordinary 64 GB nodes run out of memory
• `decomposePar` has the same out of memory problem after decomposing snapped meshes >40 M cells
• `reconstructPar` and `foamToVTK` run out of memory >30 M cells and, more fatal, they run out of walltime (24h) on even smaller meshes when having many time steps (write-out times)
• parallel scaling tests are needless for this reason, because serial jobs run 100x longer and block one node (= 31 processors idle) anyway
Simulation jobs and their evaluation

- solution for this serial performance problem: extinguish most of them...(don't know how - helpful advice welcome!)

Structure of used OpenFOAM toolboxes

- blockMesh
  - i.e. 200x100*600 mesh
  - single

- decomposePar
  - i.e. 64 processors, short runtime
  - single

- snappyHexMesh
  - i.e. refinement level 4; 2 surface layers
  - parallel

- reconstructParMesh
  - reconstructing refined mesh
  - single

- decomposePar
  - i.e. 256 processors for solver
  - single

- CFD solvers
  - i.e. SpalartAllmarasDDES
  - parallel

- reconstructPar
  - reconstructing mesh and fields for postprocessing
  - single

- foamToVTK
  - universal use postprocessing format
  - single
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh

Karman Vortex Street

- coarse mesh: 60000 cells
- fine mesh: 630000 cells

- Karman geometry was used for this simulations. Karman Vortex Street seems the most suitable case for transient flow simulation benchmarks, because it is well researched and probed, and a simple formulation for vortex frequency is also well-known. Direct Navier Stokes DNS is the nearest approximation to the reality.
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

• icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh
Postprocessing Results

- icoFoam DNS benchmark performed on coarse mesh

Animations

- ico-grob-2
- ico-grob-5
- karman-grob-10
- karman-grob-11
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh
Postprocessing Results

- icoFoam DNS benchmark performed on fine mesh

These Karman DNS Simulations show good compliance with theory of Karman Vortex Street, where \( f \) is frequency, \( Sr \) is Strouhal Number, \( U \) is velocity, \( d \) is diameter.

\[
f = \frac{Sr}{d} \quad \therefore \quad f = \frac{0.2}{0.45/0.2} = 0.45 \text{ Hz}
\]

Postprocessing shows a wavelength as expected of around 1.0m.

This Karman-DNS case can definitely be used as benchmark for transient turbulence modelings (like DDES and LES).
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

• simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh
Postprocessing Results

- simpleFoam stationary benchmark performed on fine mesh

simpleFoam does not well perform transient flows, as expected. But(!) two maxima trails of turbulence $k$ can be detected above and below the cylinder. That are definitely these trails of the alternating upper and lower vortex sequence, but even "blurred"

Another issue are the cellsizes of the nearest cells to the surface (surface contact). As shown in RANS theory, center of minimum cellsize should be $y^+ > 30$. For this CFD case, wall-function surface cells should have 30mm thickness! So it is impossible to resolve CAD structures below 60mm. Reynolds Number $Re$ is 9400, after all. This seems to be unreal. Maybe minimum $Re>1e5$ would help to get a case, which runs for RANS
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh

If running simpleFoam with short time steps, a stable pseudo-transient behavior was detected on the results.

- simpleFoam, as it is a stationary solver, uses implicit algorithms for timestepping. If we set time steps to proper (explicit) CFL numbers, simpleFoam shows a good approximation to start-up transient behavior. Advantages in comparison to transient solvers is its stability and robustness. Please evaluate yourself.
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

• simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam pseudo-transient benchmark performed on fine mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

mock-up model of a "laminar" air flow clean-room for aseptic liquid filling of pharma products

- We used a k-omega-SST turbulence model for this case. Sensitivity of initial conditions of turbulence variables $k$ and omega were evident, as it will be shown below. There is a problem with low Reynolds numbers (below 100000), as expected.

- Mesh size is 2.3 M cells
Postprocessing Results

• simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

bad initial conditions: \( k_0 = 0.2 \)
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

bad initial conditions: \( k_0 = 0.2 \)
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

good initial conditions: $k_0 = 0.01$
Postprocessing Results

• simpleFoam clean-room simulation performed on coarse mesh

good initial conditions: \( k_0 = 0.01 \)
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: k=0.012
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: k=0.016
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: $k=0.018$
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: $k=0.02$
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: $k=0.022$
Postprocessing Results

- simpleFoam clean-room simulation performed on coarse mesh

lowest shown k-level: $k=0.026$
Thank you!

We acknowledge PRACE for awarding us access to resource Hermit based in Germany at HLRS. The support of Bärbel Große-Wöhrmann from HLRS, Germany to the technical work is gratefully acknowledged.