Running simulation and visualization

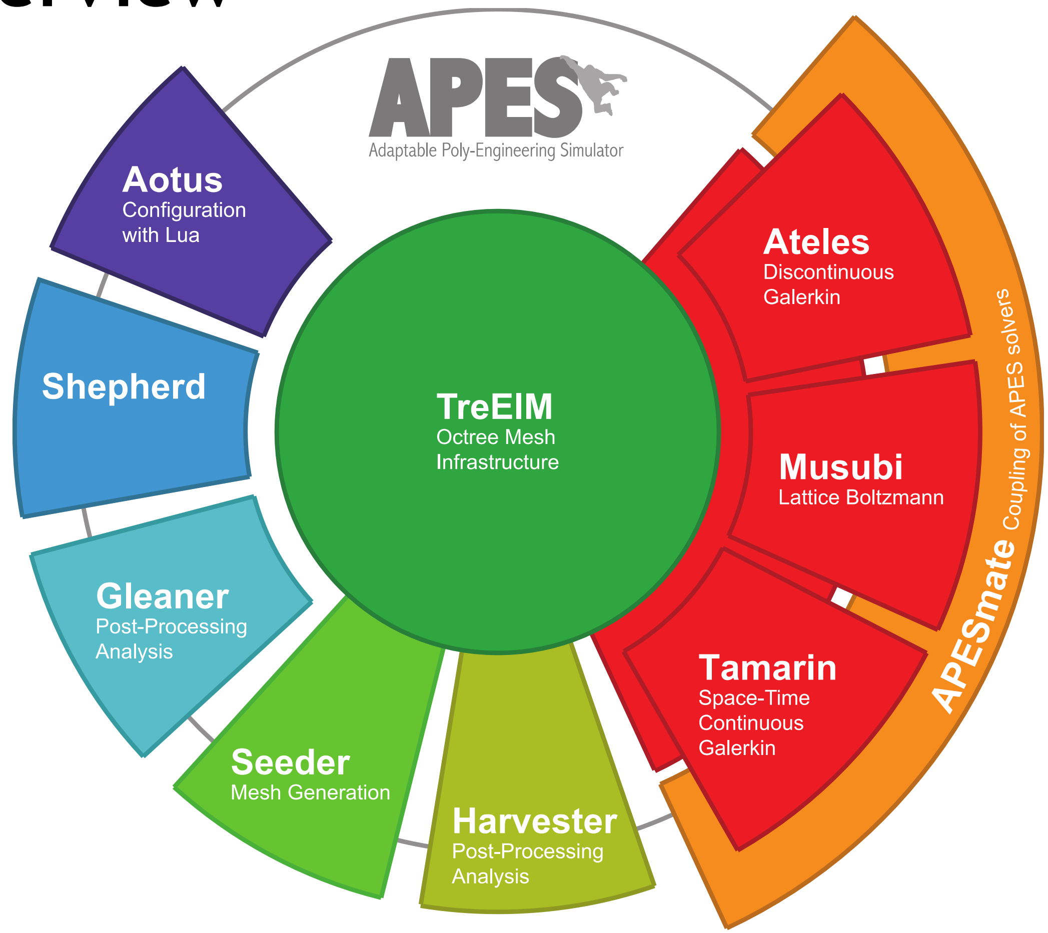
Plan

- Ateles
  - CFD software based on discontinuous Galerkin method
  - https://geb.inf.tu-dresden.de/doxy/ateles/page/index.html
- Test case: Flow over square geometry
- Exercise
  - Editing configuration file
  - Submitting job
  - Convert data into vtk files (tomorrow)
- Data visualization in CFD
- Exercise
  - Using Paraview (with provided vtk files)

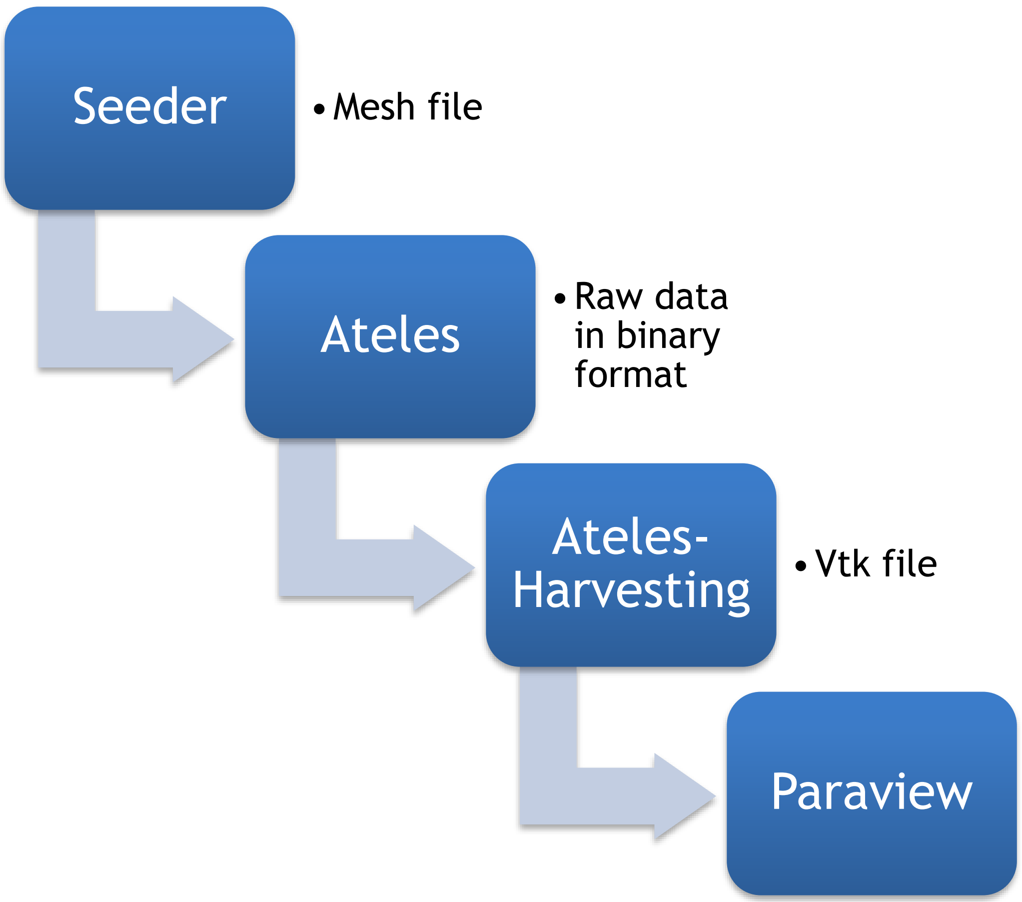
Ateles, Part of APES

- APES:
  - Simulation framework for large scale parallel computations
  - Based on Octree meshes
  - Includes tools for pre- and post-processing
- Ateles:
  - High order discontinuous Galerkin solver
  - Distributed memory parallel

APES Overview



CFD simulation pipeline



Today:
1. Run simulation
2. Visualization using Paraview

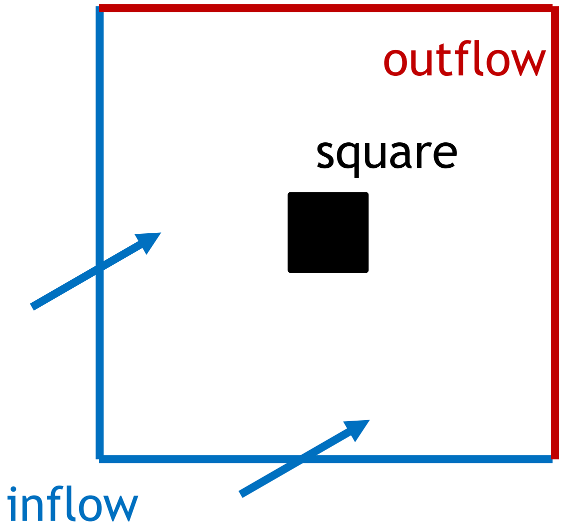
Configuration files and submission of batch scripts

FLOW AROUND SQUARE GEOMETRY

Description of the Simulation

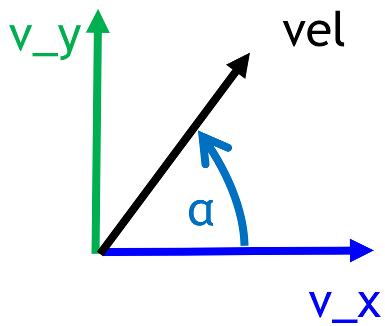
- A square obstacle is located in the middle of the domain and the flow is coming from the lower left
- 3 meshes with different resolution are provided for you.

- We will
  - Set parameters (today)
  - Submit job (today)
  - Convert results into visualization files(tomorrow)



Setting of the inflow speed

- The inflow speed is the velocity with which the fluid enters the domain at the boundaries.
- This velocity is also used as the initial condition for the entire domain at t=0.
- We specify the velocity via the Mach number
  - Mach = flow velocity / speed of sound
  - Flow velocity = Mach * speed of sound
- Angle of attack
  - V_x = vel * cos(alpha)
  - V_y = vel * sin(alpha)



Configuration file for ateles

- It contains simulation parameters which are to be read by ateles:
  - Discretization scheme in space and time
  - Equation system
  - Simulation time
  - Location of mesh files
  - Initial and boundary condition (IC, BC)
  - Tracking (output results)
  - Restart (data of the whole simulation domain)
    - Restart simulation or output results

Example of configuration file

- Text file with Lua script
  - Arithmetic operations
  - User defined functions
  - Fits well with batch jobs
- Solver reads in values from certain variables
  - scheme, equation, ...
- Lines starting with -- are comments
- Be careful:
  - Variables are case sensitive
  - Commas according to Lua syntax may be required

```
-- Scheme definitions --
scheme = {
  -- the spatial discretization scheme
  spatial =  {
    name = 'modg_2d',
    m = degree
  },
  temporal = {
    name = 'explicitSSPRungeKutta',
    steps = 2,
    -- how to control the timestep
    control = {
      name  = 'cfl',   -- adaptive timesteps
      cfl   = 0.8      -- Courant factor
    }
  }
}

-- Equation definitions --
equation = {
  name   = 'euler_2d',
  …
}
```

Output from the simulation

- There are three types of output:
  - Output logs `slurm-JOBID.out`
  - Restart files (`*.lua` and `*.lsb`)
  - Tracking files (not used this time)
  - `timing.res` (ignore it)

- The restart-files are of special interest since they are the basis for later visualization.

Job script

- In your job-script you need to start the application (here: Ateles)

- You might also do some preparation tasks beforehand (loading modules, creating paths, define variables...)

square_dd.job

```
#!/bin/bash
#SBATCH --time=02:00:00
#SBATCH --nodes=1

## And here goes the number of tasks per node,
## which usually is the number of
## cores per cpu die times the number of cpu dies ## per node.
## On Barnard this is 52. We want to use 8 here.
#SBATCH --ntasks-per-node=8

## Provide the amount of memory to use in MB on Barnard the available memory
## on each node is 515 000 MB. The default limitation is 300 MB per core.
#SBATCH –mem-per-cpu=8000

srun $KURS/bin/ateles ateles.lua
```

Exercise:

- Vary the following parameters and submit the appropriate jobs
  - Alpha: 0°, 30°,45° (maybe anything between 0° and 90°)
  - Mach number: 0.3; 0.8; 1.3
  - Mesh-resolution: 40, 100, 200 (elements in each direction, only these 3 are available for this task)

- Examples:
  - *Vary resolution*: Alpha: 45°, Mach: 0.8, Mesh: 40, 100, 200
  - *Vary Ma*: Alpha: 45°, Mach: 0.3, 0.8, 1.3, Mesh: 100
  - *Vary angle of attack*: Alpha: 0°, 30°, 45°, Mach: 0.8 Mesh: 100

Workflow

- Copy the exercise data:
  `cp -r $KURS/exercises/hpcfdx4 $MYWS`
- Change into the directory:
  `cd $MYWS/hpcfdx4`
- Create a directory for your set of parameters:
  `mkdir n40_m0.8_a45`
- Copy the necessary files into that directory:
  `cp ateles.lua square.job n40_m0.8_a45`
- Change into the directory:
  `cd n40_m0.8_a45`
- Edit ateles.lua (alpha, m, nelems):
  `vim ateles.lua`
- Create directories for restart:
  `mkdir restart`
- Submit job:
  `sbatch --reservation=YOUR-RESERVATION square.job`
- Change back to the parent directory, set up further simulations as you like

Directories containing meshes are named:

```
mesh_40
mesh_100
mesh_200
```

To check the job status:

```
squeue --me
```

Editor *gedit*

- For viewing and editing the configuration file *ateles.lua* we need a text editor

- If you are not familiar with textbased editors like vim or emacs we suggest to use **gedit**

- The file can be opened using
  `vim ateles.lua`

Note on compute time

- Use 10 cores for 40 elements,
- 20 cores for 100 elements
- and 40 cores for 200 elements