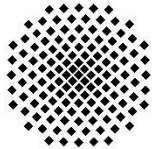


Universität Stuttgart

Challenges and Opportunities in using Software-Defined Visualization in MegaMol

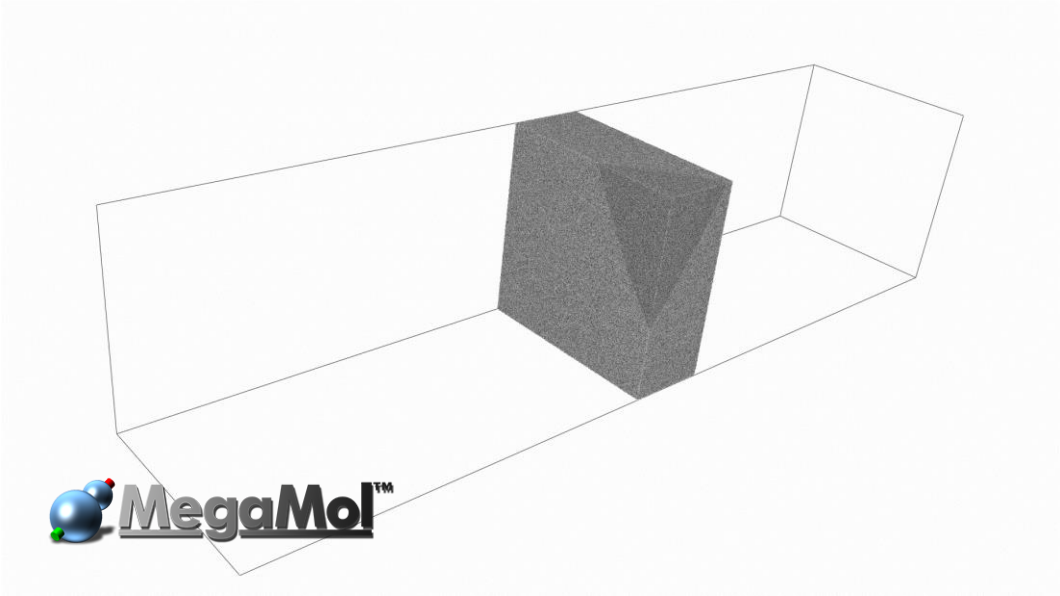
Tobias Rau, Patrick Gralka, Michael Krone, Guido Reina, Thomas Ertl

IXPUG | Bologna | 2018-03-06

Visualisierungsinstitut der Universität Stuttgart

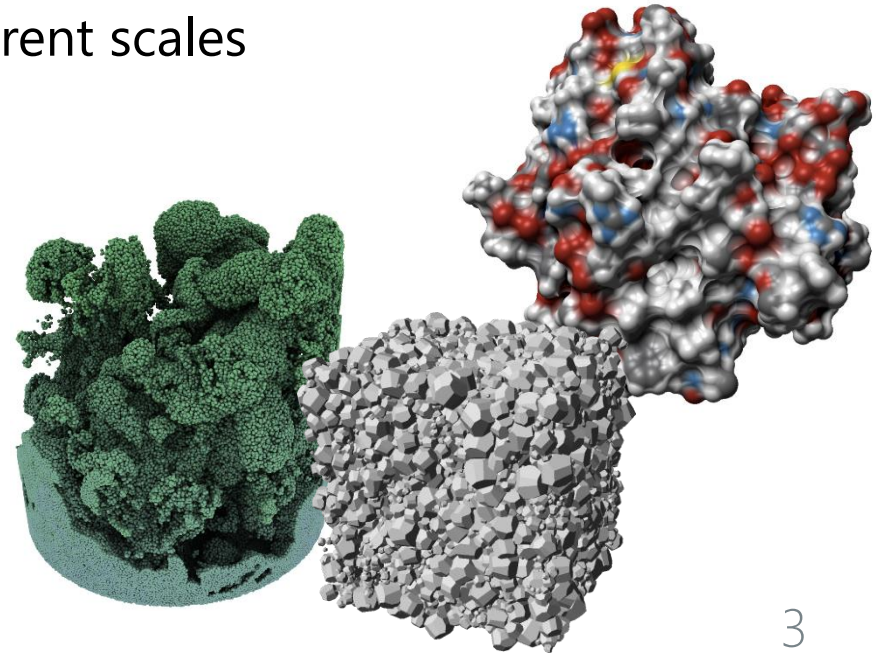
The MegaMol Visualization Framework

MegaMol is not a visualization tool.
MegaMol is a platform for visualization research.



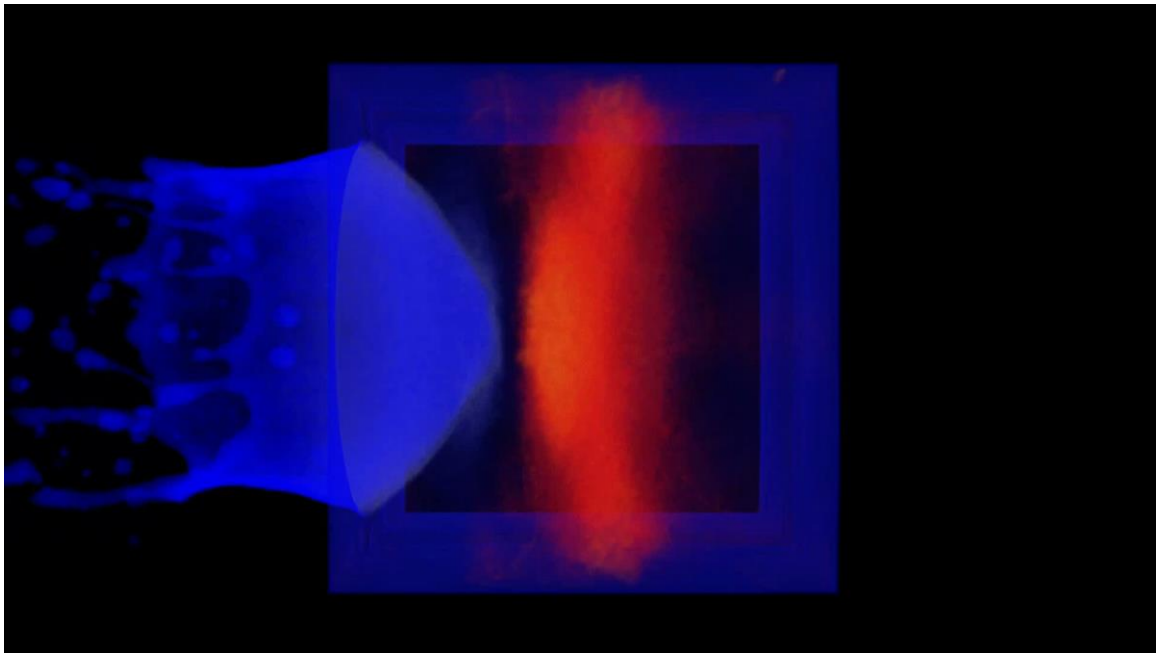
A Platform for Visualization Research

- Started at Visualization Research Center of the University of Stuttgart
- Researchers across different disciplines
- Particle-based simulations across different scales
  - Atoms, Molecules, Proteins
  - Crystallites, Granular Media
- Development of new visualization techniques
  - 5 full-time researcher positions



User Scenarios for Visualization

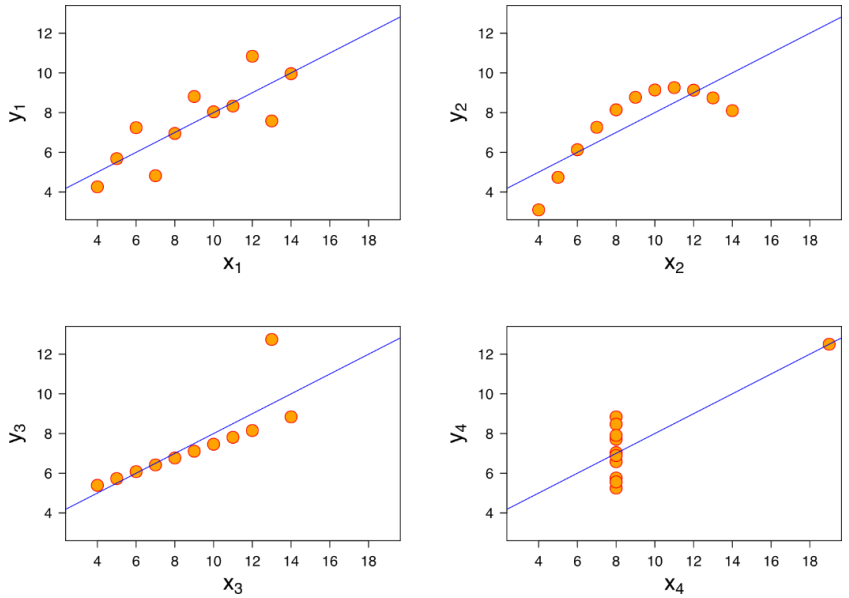
- Communicate results
  - Example: Shockwave in Aluminum



Explore your findings.

User Scenarios for Visualization

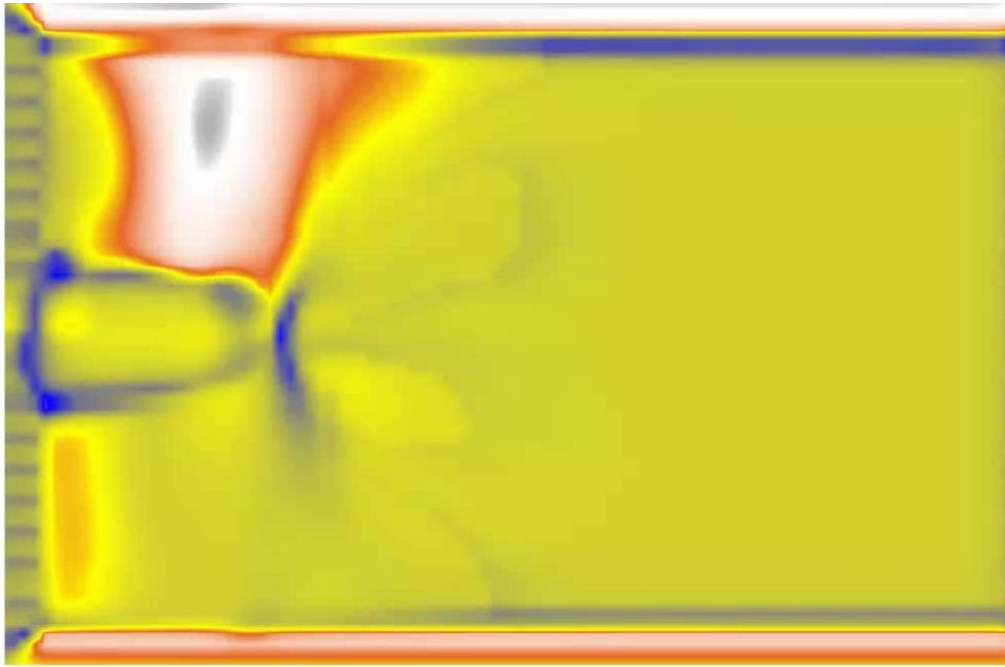
- Communicate results
  - Example: Shockwave in Aluminum
- Verify, check plausibility
  - Example: Anscombe's quartet



Debug your Simulation.

User Scenarios for Visualization

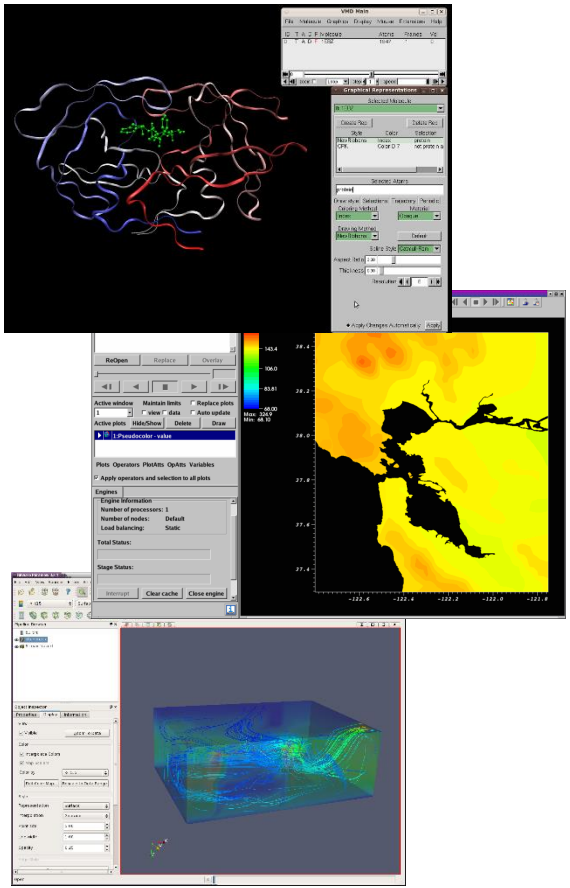
- Communicate results
  - Example: Shockwave in Aluminum
- Verify, check plausibility
  - Example: Anscombe's quartet
- Facilitate discovery
  - Example: Spherical waves during crack propagation



Discover the unexpected.

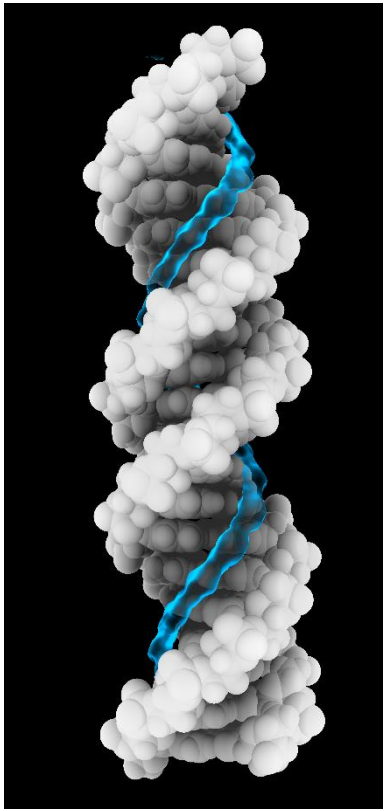
Related Software

- VMD, VisIt, Paraview, PyMol, Amira, Ovito, …
  - Either very general…
  - … or with powerful analysis functionality
- Common drawback
  - Interactivity on workstation
  - Quality of visualization for non-trivial datasets
- Why MegaMol?
  - Focus on scalability and high-quality rendering



SDVis Extending MegaMol

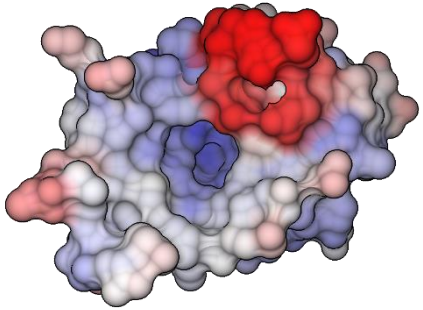
- GPU render path is limited in scalability
  - Requires sophisticated LOD- or compression techniques
- SDVis allows to circumvent memory-bound issues
- SDVis enables MegaMol to be used on a larger variety of hardware
- But: OSPRay scales at overview scenarios while GPUs can only scale at detail scenarios

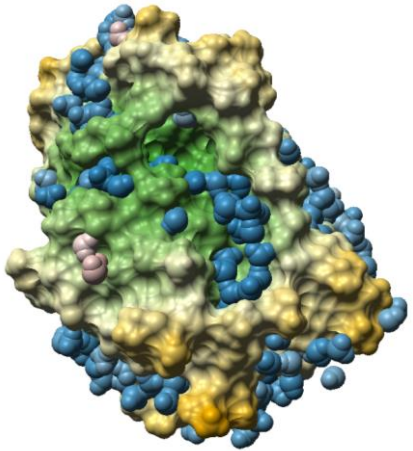


Use Case: Molecular Surface Rendering

Molecular Surface Visualization

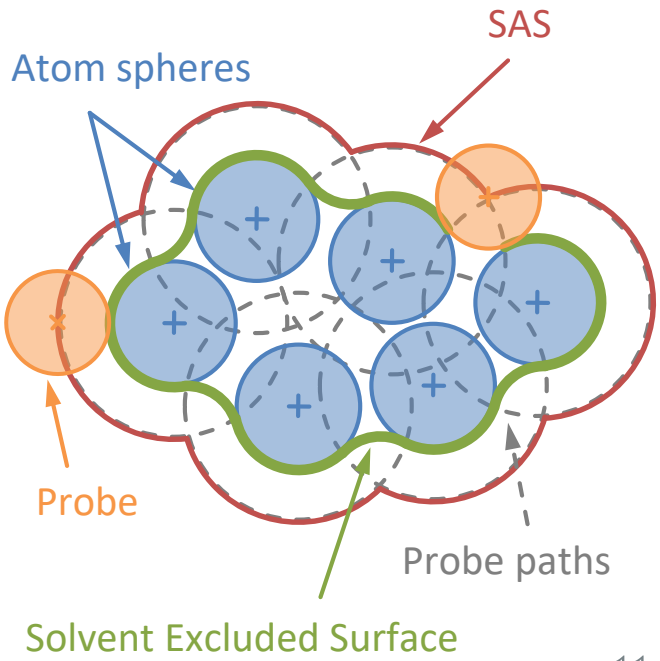
- Important visualization for analysis of molecular data
  - Show interface between a molecule and the environment
- Popular choice: Solvent Excluded Surfaces (SES)
  - Molecular surface with respect to a certain substrate molecule
  - Pro: useful for binding & docking
    - Shows what parts of the molecule are accessible to this substrate
  - Con: involved computation and rendering of the surface





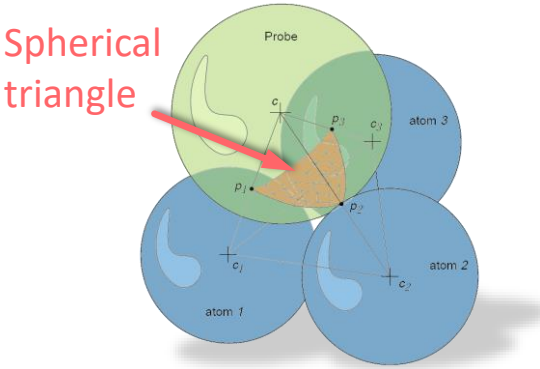
Solvent Excluded Surface – Definition

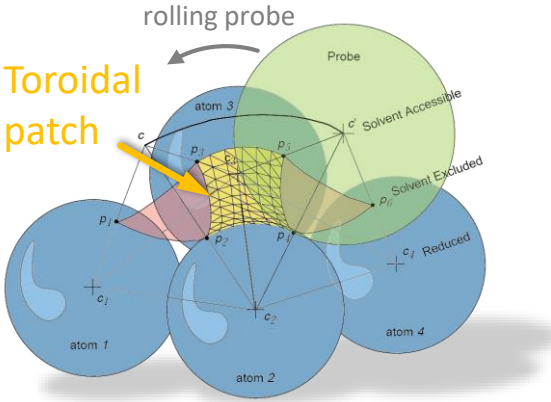
- Prerequisite: *hard sphere model* (each atom is represented by a sphere)
- Spherical probe rolling over atom spheres
  - Probe center traces out Solvent Accessible Surface (SAS)
    - Enlarged atom spheres
  - Probe surface traces out SES
    - Smooth molecular surface
    - Maintains atom spheres, closes small gaps
    - Probe cannot reach parts under the surface

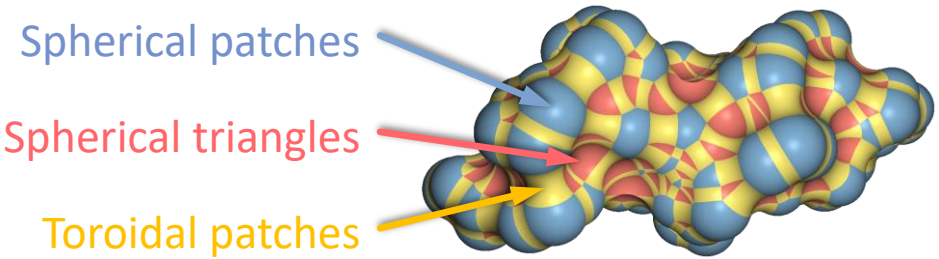


Solvent Excluded Surface – Rendering

- SES is composed of three geometric primitives
  - Spherical patches → Probe in contact with one atom
  - Spherical triangles → Probe in contact with three atoms
  - Toroidal patch → Probe in contact with two atoms
- **Requires new geometric primitives in OSPRay**
  - Analytical root finding & additional intersection tests
  - Clipping planes / spheres (spherical triangles / tori)

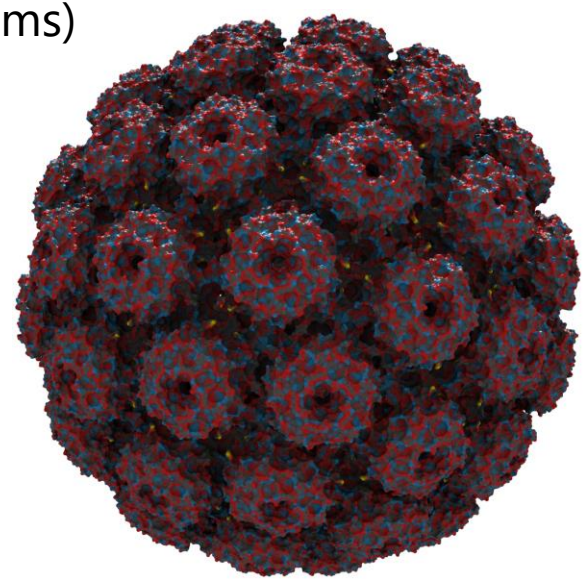






Solvent Excluded Surface – Results

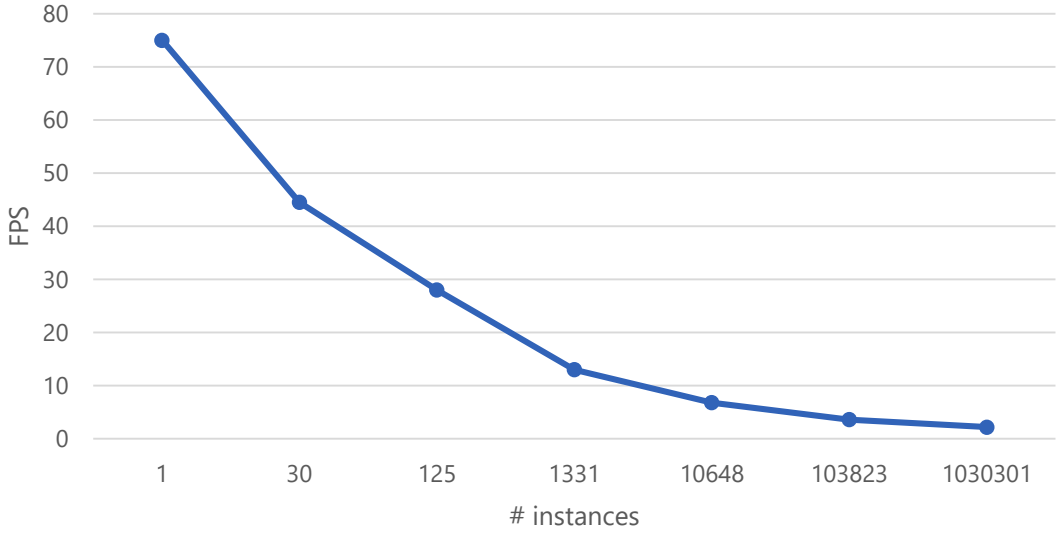
- Performance
  - Interactive rendering even for large data (>1M atoms)
  - Preliminary results *(work in progress)*
- Advantages compared to GPU-based rendering
  - Increased scalability for rendering
  - Transparency
    - Important to see inner structures
  - Improved shape perception (global illumination)
  - Cavity detection via Ambient Occlusion

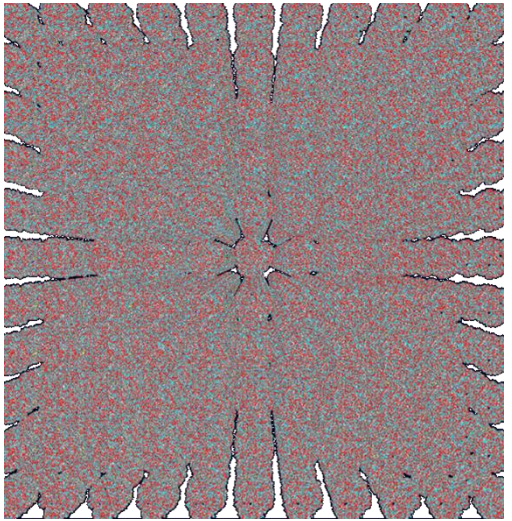


Simian virus capsid (~1M atoms)

Solvent Excluded Surface – Results

- Synthetic test case: Molecule instanced on regular grid
  - Molecule: 1AF6 → 10k atoms
  - SES: 38954 primitives



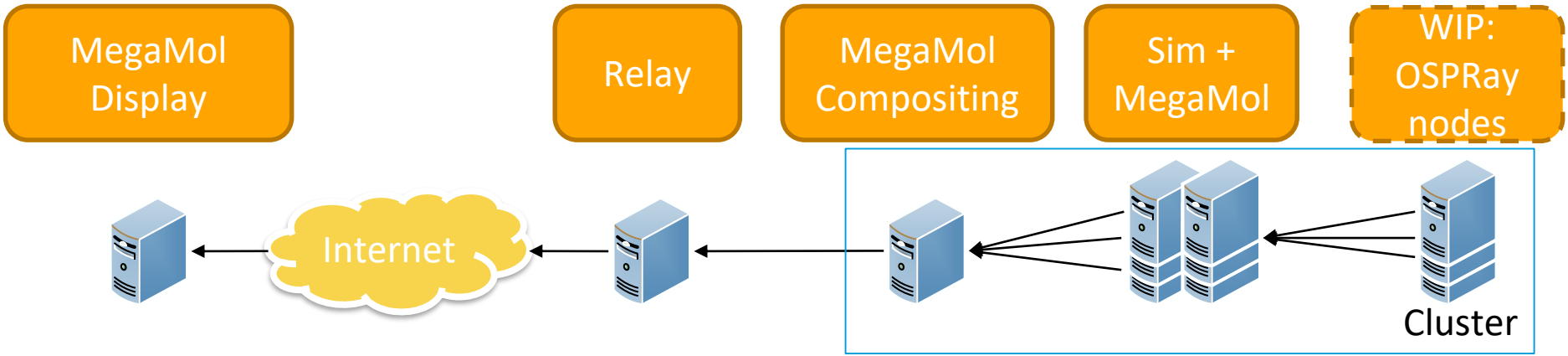


Measurements on:
i9-7900X, 64 GB RAM

Use Case: In-Situ Visualization

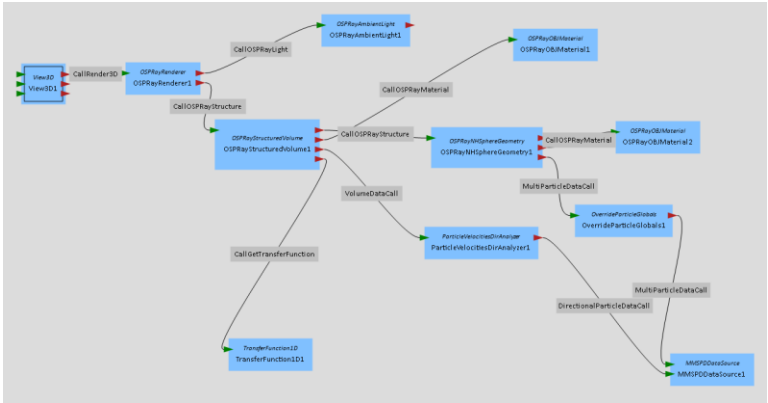
Asymmetric MegaMol Execution

- EGL frontend
- Data-distributed rendering
- Depth-based compositing



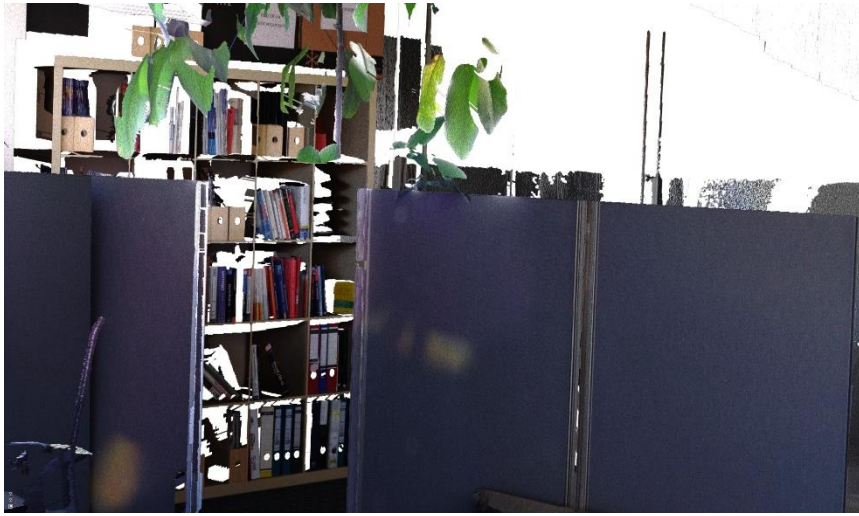
Advantages of Distributed MegaMol

- **Distributed MegaMol is fully featured on all nodes**
  - Each node has a complete, individual MegaMol graph
    - Processing, Filtering, Slicing, Highlighting, Rendering, …
  - Local operations for each node
- OSPRay offloading has no ‘semantic operations’
  - Operations happen on compositing node
  - Operations require re-distribution of data



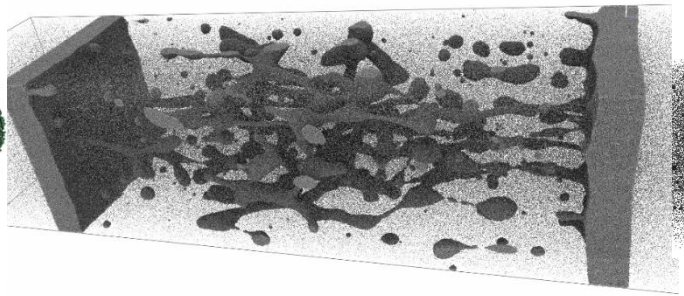
General In-Situ Setup

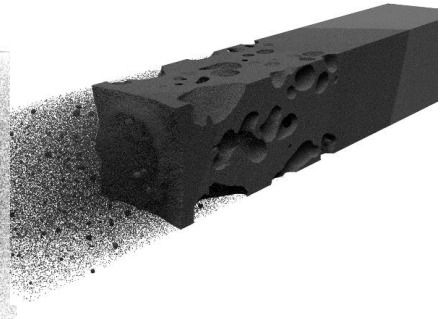
- Two paradigms available:
  - Image transfer
    - Low bandwidth requirements
    - Share compute load with simulation or use dedicated compute nodes
  - Data transfer
    - High bandwidth requirements
    - Allows for steering
- MegaMol provides NAMD interface (for ESPResSo, and older Gromacs versions)
  - WIP: Interface with other simulation software

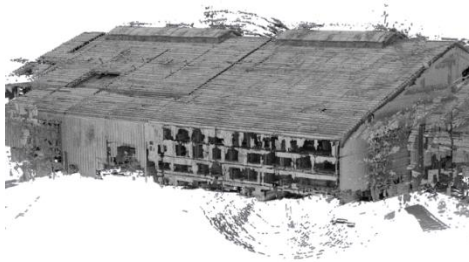


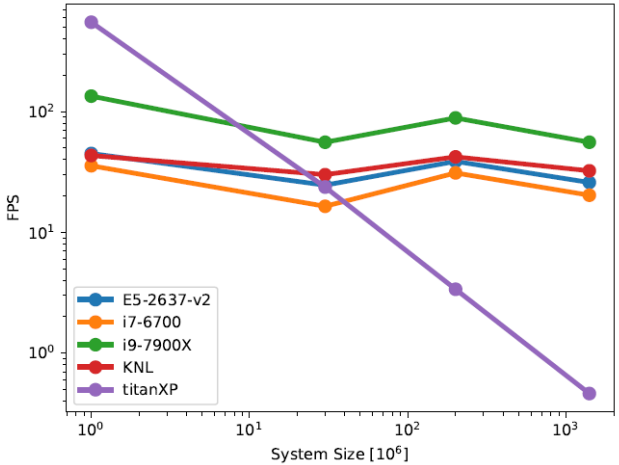
FARO
THE MEASURE OF SUCCESS®

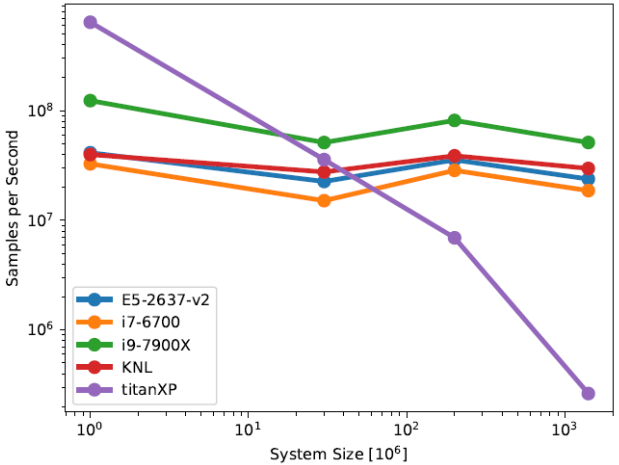
SDVis Benefits











Thank You

MegaMol™

megamol.org

- Stuttgart now part of the IPCC family
- Check out MegaMol on GitHub
- Interesting problems wanted! ☺

Universität Stuttgart

intel®

Baden-Württemberg Stiftung
WIR STIFTEN ZUKUNFT

SFB 716