Collaborative **R**esearch into **E**xascale **S**ystemware, **T**ools and **A**pplications

Gregor Matura, German Aerospace Center (DLR)

Achim Basermann, Fang Chen, Markus Flatken, Andreas Gerndt (DLR)

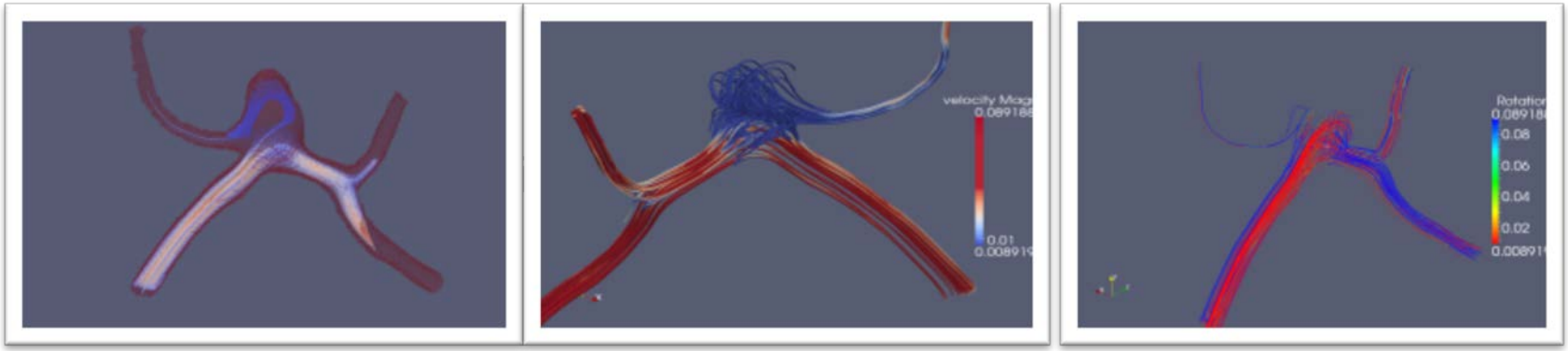
James Hetherington, Timm Krüger, Rupert Nash (UCL)

Challenges in exascale post processing

- Huge amount of data to be processed and visualized
- Not possible to store data on disk
- Moving data is costly
- Memory issue
- Efficiency of parallelization in respect to visualization techniques
- Latency

Post processing

- Blood flow simulation for aneurysm study
- Flow visualization.
  - Volumes
  - Lines
  - Particles



Approaches

- In-situ visualization
  - Visualize on-going simulation result, without pausing simulation
  - Visualize coarse data
- Interactive visualization
  - Interactive framerates (no latency to human eyes)
  - Finding suitable visualization techniques
  - Finding suitable parallelization for chosen visualization techniques
  - Exploring rendering with GPU
- Multi-resolution data visualization
  - Define level of details
  - Provide visualization on different level of details.
  - Enable first result on a much coarser mesh.

In-situ visualization

- Test result with HemeLB

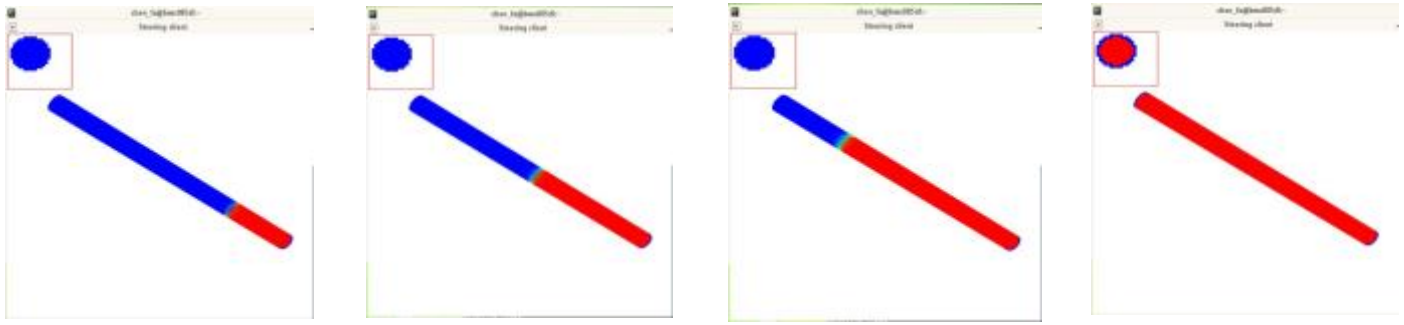


Fig: An example of cut plane in-situ visualization of a testing simulation. This visualization is provided at run time, i.e., while simulation is running.

- Visualization is done on the same node where simulation is distributed.

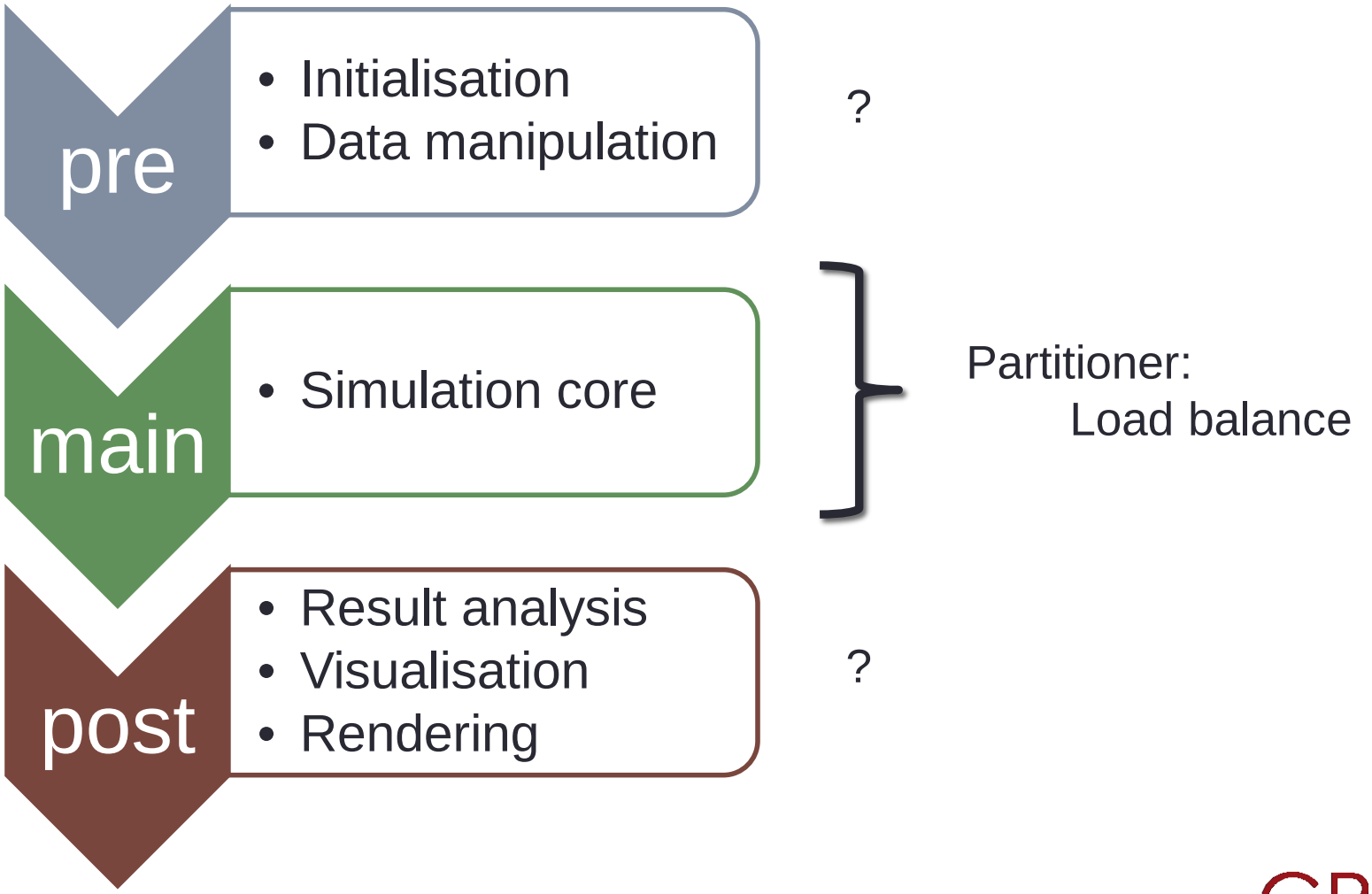
Using Virtual Reality

- Power-wall, display-wall systems
- Immersive visualization
- Provide great details
- Enhanced depth perception in VR
- Enable user to explore their data in a natural way

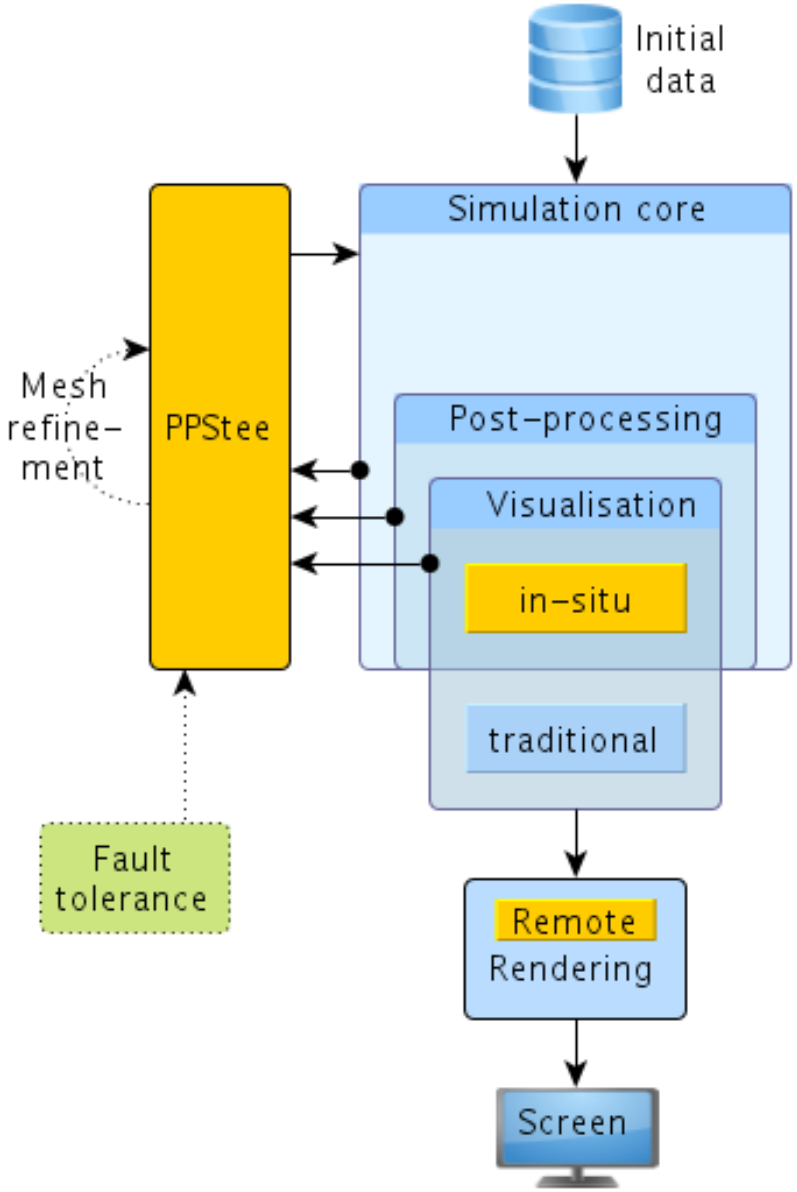
PPSTEE

A Pre-Processing Interface for Steering Exascale Simulations by Intermediate Result Analysis through In-Situ Post-Processing

Motivation


pre
• Initialisation
• Data manipulation
?
main
• Simulation core
Partitioner:
Load balance
post
• Result analysis
• Visualisation
• Rendering
?

PPStee data flow: overview



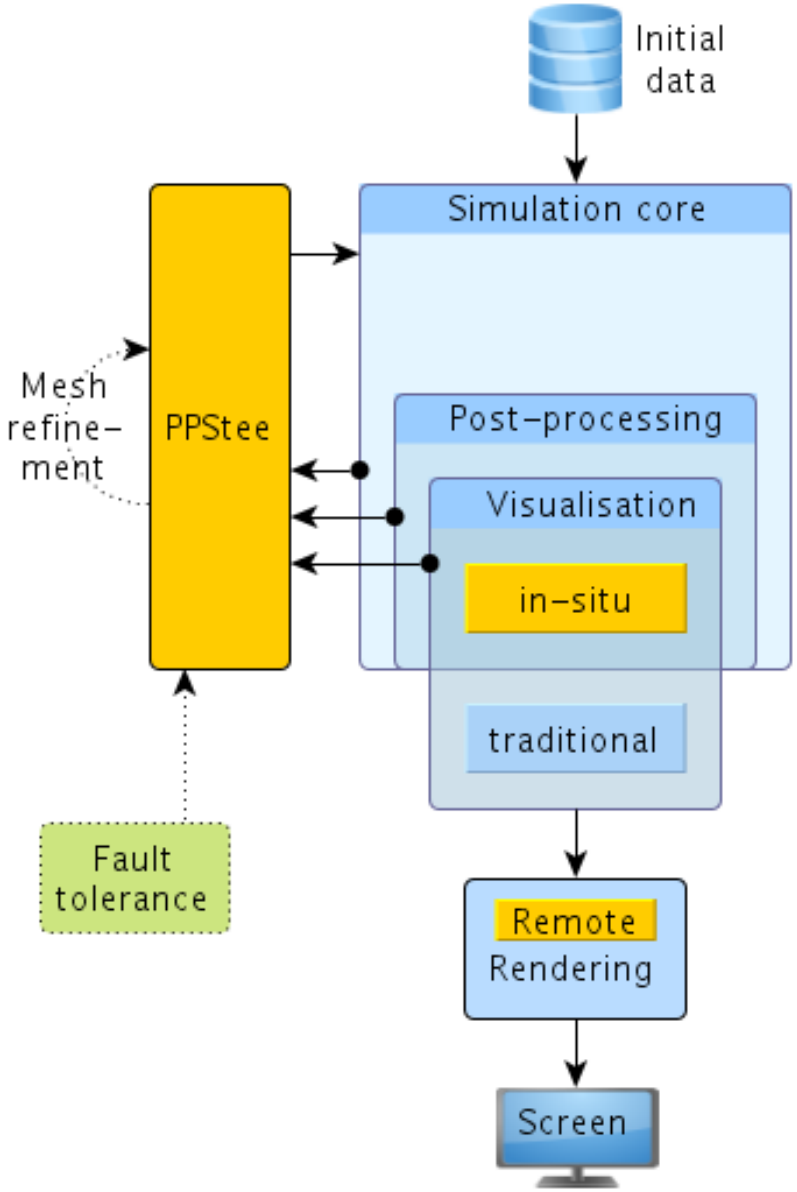
Assumptions

- The simulation uses a partitioner
  - There is some sort of mesh
  - The mesh is there (= in memory): an initial (=bad) read-in happens
- The simulation has more than one stage
  - E.g.: an in-situ data analysis or visualization is integrated

Properties

- Swappable external partitioning tool
- Flexible data format
- Incorporates different simulation stages like computation and visualization
- Easily adjustable to
  - new partitioning tools
  - different kinds of stages
  - fault tolerance
  - (more/any) mesh refinement

PPStee data flow: revisited



Advantages

- Offers standardised partitioner access
- Relies on established external partitioning tools
  (however own ones can be integrated as well)
- Little overhead: if partitioning is already implemented required interface input is present in some (similar) form
- Small memory requirements

Disadvantages

- Individual routines of external partitioning tools covering special functionalities have to be implemented separately (yet this is possible)
- Another software layer

Basic usage: example

Old call to ParMETIS:

```
ParMETIS_V3_PartKway(
        vtxdist, xadj, adjncy,
        vwgt, adjwgt,
        wgtflag, numflag, ncon, nparts,
        tpwgts, ubvec, options, edgecut,
        part,
        comm);
```

Basic usage: example

Call to PPStee:

```
// get interface
PPStee ppstee;
// submit graph
ppstee.submitGraph(pgraph);
// submit weights
ppstee.submitNewStage(wgtCmp, PPSTEE_STAGE_COMPUTATION);
ppstee.submitNewStage(wgtVis, PPSTEE_STAGE_VISUALISATION);
// calculate partitioning
PPSteePart* ppart;
ppstee.getPartitioning(&ppart);
```

Basic usage: example

Build graph:

```
// get graph (as ParMETIS type)
PPSteeGraph pgraph = PPSteeGraphParmetis(MPI_COMM_WORLD,
                                         vtxdist, xadj, adjncy);
```

Build weights:

```
// construct and set weights for computation
PPSteeWeights wgtCmp(&pgraph);
wgtCmp.setWeightsData(vwgt_c, adjwgt_c);
// construct and set weights for visualisation
PPSteeWeights wgtVis(&pgraph);
wgtVis.setWeightsData(vwgt_v, adjwgt_v);
```

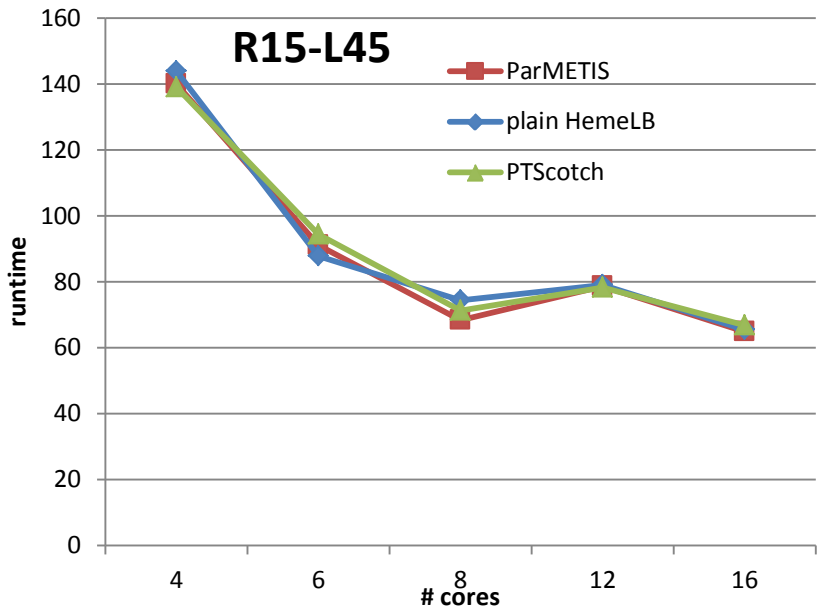
Future work

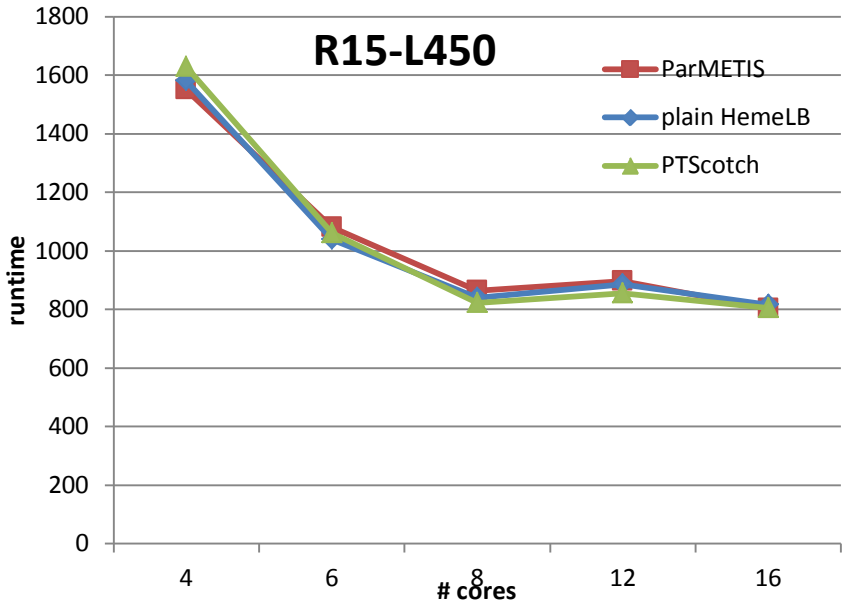
- Integration into HemeLB
- Performance measurements with HemeLB
- Further tests with other applications
- Revision of architecture
- Comparison with other frameworks
  (some of them cover features of PPStee, e.g. ITAPS)

Preliminary results

HemeLB test runs on HemeLB test data sets (R15-L45 and R15-L450)

PPStee using ParMETIS vs. PPStee using PTScotch
vs. plain HemeLB code (ParMETIS)





CREST

Thank you

for your attention!