Collaborative Research into Exascale Systemware, Tools and Applications

Exascale Pre and Post Processing

Achim Basermann, Gregor Matura, Fang Chen, Markus Flatken, Andreas Gerndt (DLR)
James Hetherington, Timm Krüger, Jiri Jaros, Rupert Nash (UCL)
Martin Aumüller (USTUTT)
CRESTA Work Package 5: User Tools

- Objectives: development of pre and post processing as well as remote hybrid rendering tools to support exascale applications and users including:
  
  - Mesh manipulation and partitioning tools
  - Visualisation tools
  - Data management tools
Survey

• Post Processing

• Remote Hybrid Rendering

• Pre processing interface PPSTee
Challenges in Exascale Post Processing

- Huge amount of data to be processed and visualised
- Not possible to store data on disk
- Moving data is costly
- Memory issue
- Efficiency of parallelisation with respect to visualisation techniques
- Latency
Post Processing

- Blood flow simulation for aneurysm study
- Flow visualisation
  - Volumes
  - Lines
  - Particles
Approaches

• In-situ visualisation
  • Visualise on-going simulation result, without pausing simulation
  • Visualise coarse data

• Interactive visualisation
  • Interactive frame rates (no latency to human eyes)
  • Finding suitable visualisation techniques
  • Finding suitable parallelisation for chosen visualisation techniques
  • Exploring rendering with GPU

• Multi-resolution data visualisation
  • Define level of details.
  • Provide visualisation on different levels of details.
  • Enable first result on a much coarser mesh.
In-Situ Visualisation

- Test result with HemeLB

![Image](image.png)

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

- Visualisation is done on the same node where simulation is performed.
Using Virtual Reality

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

- Steer mesh or simulation parameters
- Based on provided in-situ visualisation
- Carry out steering during simulation is running
- Prevent failure
- Achieve better convergence for the solver
- Saving time
Remote Hybrid Rendering

- Goal: Access simulations on exascale resources from desktops and immersive virtual environments.

Remote Desktops (Zuffenhausen, RWTH Aachen)

Remote Tiled Displays / Powerwalls (HLRS, IHS)

Remote Virtual Reality (Stuttgart, Sindelfingen)
Remote Hybrid Rendering: Advantage

- Simulation data is rendered on or near the exascale system using scalable methods
- Static context information is overlayed locally
- Better interactivity because of some decoupling of interaction from remote rendering

Remote + Local = Display
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

post
- Result analysis
- Visualisation
- Rendering

Partitioner:
Load balance
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 visualisation is integrated
Properties

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

• Pre-processing data structures and algorithms suitable for exascale systems identified

• Prototype pre-processing interface for exascale systems defined and tested with HemeLB

• Concepts for data structures and algorithms suitable for exascale interactive post-processing including remote hybrid rendering developed

• Tests of prototype post-processing tools with HemeLB and OpenFOAM data sets successful
Future work

• Integration of PPStee into HemeLB production version
• Performance measurements with HemeLB (also using Zoltan)
• Further tests with other applications (OpenFOAM)
• Revision of architecture
• Comparison with other frameworks
  (some of them cover features of PPStee, e.g. ITAPS)

• Further development of the exascale post processing and remote hybrid rendering techniques

• Further co-design with CRESTA applications
Thank you for your attention!
Basic usage: example

Old call to ParMETIS:

```
ParMETIS_V3_PartKway(
    vtxdist, xadj, adjncy,
    vwgt, adjwgt,
    wgtflag, numflag, ncon, nparts,
    tpwghts, 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);