Towards the new online visualization approach

M.O. Bakhterev, S.V. Porshnev, P.A. Vasev

N.N. Krasovskii Institute, Ekaterinburg, Russia

Insitu visualization is a technique, when intermediate data of numeric program running on a supercomputer is visualized for analysis during execution. Insitu techniques are used nowadays even for writing data to storage in order to overcome I/O bottlenecks of exascale computations [1]. There is also *online visualization* technique, which allows not only to visualize intermediate data during execution, but to control program flow (online visualization is also called computational steering). This also might be used in automated environments, when super-computation is steered by external processes.

Several frameworks help to implement some elements of these techniques: Sensei, ADIOS, and others (see our review of modern approaches [2]). Our group also develops online visualization frameworks. We think that in general, to implement insitu and online visualizations, a framework programmer needs to do the following.

N1) Provide an API for intermediate data output from a main numeric program.

N2) Provide tools to program nontrivial processing pipelines of the given data.

N3) Provide an API for main numeric program to read steering commands.

N4) Provide tools to program data paths from external sources into a numeric program.

N5) Develop a visual frontend, if required.

In our current project, we suggest the following approach.

1) Use POSIX file API with FUSE backend as the basic API for N1 and N3. Thus, a numeric program may receive data requests and steering commands with file reads and output requested data with file writes.

2) Use a specialized language to define processing graphs which may be started and attached to a running numeric program, thus, addressing N2. This might be used for N4 also.

3) Use the same language for defining views for N5.

Consider the following example of code written in the developed language, figure 1.

```
input1: input_slot path="/data/grid";   // read input data from numeric program
iso: @input1 | isosurface value=0.7;    // process input using isosurface algorithm
@iso | write_slot path="/processed";    // write isosurface to disk
view name="view1" {                     // definition of the view; there may be many views
  render3d {
    @iso | triangles;                   // present isosurface to a user
  }
}
```

Figure 1. The code is constructed as a set of nodes with parameters. Here "name:" term denotes new names, "@name" term denotes link to node, "node1 | node2" denotes a pipeline, "{}" defines nesting.

We suppose that defining in-situ/on-line visualization pipeline in this way is an effective tool for developers. The problems of effective automatic pipeline node assignment to the network hosts and the effective communication of pipeline nodes are under research.

References

1. Pugmire D., et al. Visualization as a Service for Scientific Data // Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI – 17th Smoky Mountains Computational Sciences and Engineering Conference, SMC 2020, Oak Ridge, TN, USA, August 26–28, 2020, Revised Selected Papers. CCIS, vol. 1315. Springer, 2021. P. 157–174. DOI: 10.1007/978-3-030-63393-6_11.
2. Vasev D. Analyzing an Ideas Used in Modern HPC Computation Steering // 2020 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT). DOI: 10.1109/USBEREIT48449.2020.9117685.