2.1.2 ECP/VTK-m

Overview The ECP/VTK-m project is providing the core capabilities to perform scientific visualization on exascale architectures. The ECP/VTK-m project fills the critical feature gap of performing visualization and analysis on processors like graphics-based processors and many integrated core. The results of this project will be delivered in tools like ParaView, VisIt, and Ascent as well as in stand-alone form. Moreover, these projects are depending on this ECP effort to be able to make effective use of ECP architectures.

One of the biggest recent changes in high-performance computing is the increasing use of accelerators. Accelerators contain processing cores that independently are inferior to a core in a typical CPU, but these cores are replicated and grouped such that their aggregate execution provides a very high computation rate at a much lower power.

Current and future CPU processors also require much more explicit parallelism. Each successive version of the hardware packs more cores into each processor, and technologies like hyperthreading and vector operations require even more parallel processing to leverage each core's full potential.

VTK-m is a toolkit of scientific visualization algorithms for emerging processor architectures. VTK-m supports the fine-grained concurrency for data analysis and visualization algorithms required to drive extreme scale computing by providing abstract models for data and execution that can be applied to a variety of algorithms across many different processor architectures.

The ECP/VTK-m project is building up the VTK-m codebase with the necessary visualization algorithm implementations that run across the varied hardware platforms to be leveraged at the exascale. We will be working with other ECP projects, such as ALPINE, to integrate the new VTK-m code into production software to enable visualization on our HPC systems.

Key Challenges The scientific visualization research community has been building scalable HPC algorithms for over 15 years, and today there are multiple production tools that provide excellent scalability. However, our current visualization tools are based on a message-passing programming model. More to the point, they rely on a coarse decomposition with ghost regions to isolate parallel execution [7, 8]. However, this decomposition works best when each processing element has on the order of a hundered thousand to a million data cells [9] and is known to break down as we approach the level of concurrency needed on modern accelerators [10, 11].

DOE has made significant investments in HPC visualization capabilities. For us to feasibly update this software for the upcoming exascale machines, we need to be selective on what needs to be updated, and we need to maximize the code we can continue to use. Regardless, there is a significant amount of software to be engineered and implemented, so we need to extend our development resources by simplifying algorithm implementation and providing performance portability across current and future devices.

Solution Strategy The ECP/VTK-m project leverages VTK-m [12] to overcome these key challenges. VTK-m has a software framework that provides the following critical features.

1. **Visualization building blocks:** VTK-m contains the common data structures and operations required for scientific visualization. This base framework simplifies the development of visualization algorithms [13].

2. **Device portability:** VTK-m uses the notion of an abstract device adapter, which allows algorithms written once in VTK-m to run well on many computing architectures. The device adapter is constructed from a small but versatile set of data parallel primitives, which can be optimized for each platform [14]. It has been shown that this approach not only simplifies parallel implementations, but also allows them to work well across many platforms [15, 16, 17].

3. **Flexible integration:** VTK-m is designed to integrate well with other software. This is achieved with flexible data models to capture the structure of applications' data [18] and array wrappers that can adapt to target memory layouts [19].

Even with these features provided by VTK-m, we have a lot of work ahead of us to be ready for exascale. Our approach is to incrementally add features to VTK-m and expose them in tools like ParaView and VisIt.

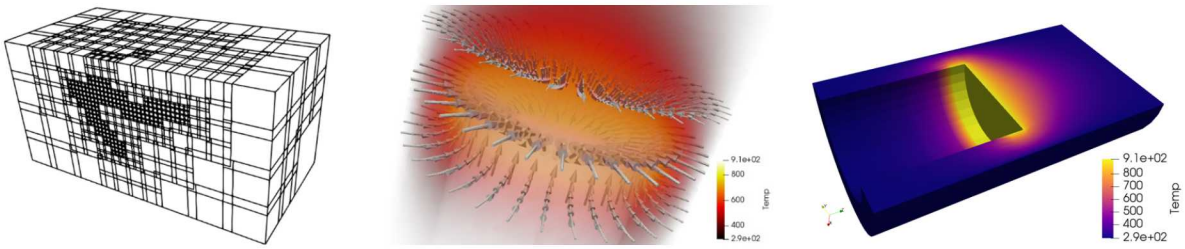

Figure 2: Examples of recent progress in VTK-m include (from left to right) multiblock data structures, gradient estimation, and mapping of fields to colors.

Recent Progress The VTK-m project is organized into many implementation activities. The following features have been completed in the past 12 months.

- **Key Reduce Worklet:** This adds a basic building block to VTK-m that is very useful in constructing algorithms that manipulate or generate topology [20].
- **Spatial Division:** Introductory algorithms to divide space based on the distribution of geometry within it. This is an important step in building spatial lookup structures.
- **Basic Particle Advection:** Particle advection traces the path of particles in a vector field. This tracing is fundamental for many flow visualization techniques. Our initial implementation works on simple structures
- **Surface Normals:** Normals, unit vectors that point perpendicular to a surface, are important to provide shading of 3D surfaces while rendering. These often need to be derived from the geometry itself.
- **Multiblock Data:** Treat multiple blocks of data, such as those depicted in Figure 2 at left, as first-class data sets. Direct support of multiblock data not only provides programming convenience but also allows us to improve scheduling tasks for smaller groups of data.
- **Gradients:** Gradients, depicted in Figure 2 at center, are an important metric of fields and must often be derived using topologic data. Gradients are also fundamental in finding important vector field qualities like divergence, vorticity, and q-criterion.
- **Field to Colors:** Pseudocoloring, demonstrated in Figure 2 at right, is a fundamental feature of scientific visualization, and it depends on a good mechanism of converting field data to colors.
- **VTK-m 1.1 Release:** VTK-m 1.1 was released in December 2017.

Next Steps Our next efforts include:

- **External Surface:** Extracting the external faces of solid geometry is important for efficient solid rendering.
- **Location Structures:** Many scientific visualization algorithms require finding points or cells based on a world location.
- **Dynamic Types:** The initial implementation of VTK-m used templating to adjust to different data structures. However, when data types are not known at compile time, which is common in applications like ParaView and VisIt, templating for all possible combinations becomes infeasible. Provide mechanisms to enable runtime polymorphism.
- **OpenMP:** Our current multicore implementation uses TBB [21] for its multicore support. However, much of the code we wish to integrate with uses OpenMP [22], and the two threading implementations can conflict with each other. Thus, add a device adapter to VTK-m that uses OpenMP so this conflict will not happen.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] B. Smith, R. Bartlett, and xSDK developers. xSDK community package policies, 2016. version 0.3, December 2, 2016, https://dx.doi.org/10.6084/m9.figshare.4495136.

[2] R. Bartlett, J. Sarich, B. Smith, T. Gamblin, and xSDK developers. xSDK community installation policies: GNU Autoconf and CMake options, 2016. version 0.1, December 19, 2016, https://dx.doi.org/10.6084/m9.figshare.4495133.

[3] Todd Gamblin, Matthew P. LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and W. Scott Futral. The Spack Package Manager: Bringing order to HPC software chaos. In *Supercomputing 2015 (SC'15)*, Austin, Texas, November 15-20 2015. LLNL-CONF-669890.

[4] Alquimia Project Team. Alquimia Web page. https://bitbucket.org/berkeleylab/alquimia.

[5] PFLOTRAN Project Team. PFLOTRAN Web page. http://www.pflotran.org.

[6] Alicia Marie Klinvex. xSDKTrilinos user manual. Technical Report SAND2016-3396 O, Sandia, 2016.

[7] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C. Charles Law, and Michael Papka. Large-scale data visualization using parallel data streaming. *IEEE Computer Graphics and Applications*, 21(4):34–41, July/August 2001.

[8] Hank Childs, David Pugmire, Sean Ahern, Brad Whitlock, Mark Howison, Prabhat, Gunther H. Weber, and E. Wes Bethel. Extreme scaling of production visualization software on diverse architectures. *IEEE Computer Graphics and Applications*, 30(3):22–31, May/June 2010. DOI 10.1109/MCG.2010.51.

[9] Kenneth Moreland. The ParaView tutorial, version 4.4. Technical Report SAND2015-7813 TR, Sandia National Laboratories, 2015.

[10] Kenneth Moreland. Oh, $#*@! Exascale! The effect of emerging architectures on scientific discovery. In *2012 SC Companion (Proceedings of the Ultrascale Visualization Workshop)*, pages 224–231, November 2012. DOI 10.1109/SC.Companion.2012.38.

[11] Kenneth Moreland, Berk Geveci, Kwan-Liu Ma, and Robert Maynard. A classification of scientific visualization algorithms for massive threading. In *Proceedings of Ultrascale Visualization Workshop*, November 2013.

[12] Kenneth Moreland, Christopher Sewell, William Usher, Li ta Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, Matthew Larsen, Chun-Ming Chen, Robert Maynard, and Berk Geveci. Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications*, 36(3):48–58, May/June 2016. DOI 10.1109/MCG.2016.48.

[13] Kenneth Moreland. The vtk-m user's guide. techreport SAND 2018-0475 B, Sandia National Laboratories, 2018. http://m.vtk.org/images/c/c8/VTKmUsersGuide.pdf.

[14] Guy E. Blelloch. *Vector Models for Data-Parallel Computing*. MIT Press, 1990. ISBN 0-262-02313-X.

[15] Li-ta Lo, Chris Sewell, and James Ahrens. PISTON: A portable cross-platform framework for data-parallel visualization operators. In *Eurographics Symposium on Parallel Graphics and Visualization*, 2012. DOI 10.2312/EGPGV/EGPGV12/011-020.

[16] Matthew Larsen, Jeremy S. Meredith, Paul A. Navrátil, and Hank Childs. Ray tracing within a data parallel framework. In *IEEE Pacific Visualization Symposium (PacificVis)*, April 2015. DOI 10.1109/PACIFICVIS.2015.7156388.

[17] Kenneth Moreland, Matthew Larsen, and Hank Childs. Visualization for exascale: Portable performance is critical. In *Supercomputing Frontiers and Innovations*, volume 2, 2015. DOI 10.2312/pgv.20141083.

[18] Jeremy S. Meredith, Sean Ahern, Dave Pugmire, and Robert Sisneros. EAVL: The extreme-scale analysis and visualization library. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pages 21–30, 2012. DOI 10.2312/EGPGV/EGPGV12/021-030.

[19] Kenneth Moreland, Brad King, Robert Maynard, and Kwan-Liu Ma. Flexible analysis software for emerging architectures. In *2012 SC Companion (Petascale Data Analytics: Challenges and Opportunities)*, pages 821–826, November 2012. DOI 10.1109/SC.Companion.2012.115.

[20] Robert Miller, Kenneth Moreland, and Kwan-Liu Ma. Finely-threaded history-based topology computation. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 41–48, 2014. DOI 10.2312/pgv.20141083.

[21] James Reinders. *Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism*. O'Reilly, July 2007. ISBN 978-0-596-51480-8.

[22] Barbara Chapman, Gabriele Jost, and Ruud van der Pas. *Using OpenMP*. MIT Press, 2007. ISBN 978-0-262-53302-7.