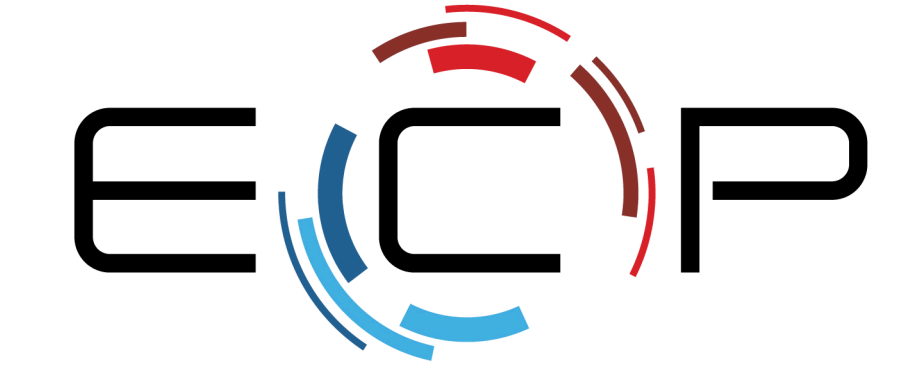
SAND2017-4309R



Milestone Completion Report

WBS 1.3.5.05 ECP/VTK-m
FY17Q2 [MS-17/01] Better Dynamic Types Design, SDA05-1

Kenneth Moreland
Sandia National Laboratories

April 24, 2017

EXECUTIVE SUMMARY

The FY17Q2 milestone of the ECP/VTK-m project, which is the first milestone, includes the completion of design documents for the introduction of virtual methods into the VTK-m framework. Specifically, the ability from within the code of a device (e.g. GPU or Xeon Phi) to jump to a virtual method specified at run time. This change will enable us to drastically reduce the compile time and the executable code size for the VTK-m library.

Our first design introduced the idea of adding virtual functions to classes that are used during algorithm execution. (Virtual methods were previously banned from the so called execution environment.) The design was straightforward. VTK-m already has the generic concepts of an "array handle" that provides a uniform interface to memory of different structures and an "array portal" that provides generic access to said memory. These array handles and portals use C++ templating to adjust them to different memory structures. This composition provides a powerful ability to adapt to data sources, but requires knowing static types. The proposed design creates a template specialization of an array portal that decorates another array handle while hiding its type. In this way we can wrap any type of static array handle and then feed it to a single compiled instance of a function.

The second design focused on the mechanics of implementing virtual methods on parallel devices with a focus on CUDA. Our initial experiments on CUDA showed a very large overhead for using virtual C++ classes with virtual methods, the standard approach. Instead, we are using an alternate method provided by C that uses function pointers.

With the completion of this milestone, we are able to move to the implementation of objects with virtual (like) methods. The upshot will be much faster compile times and much smaller library/executable sizes.

1. INTRODUCTION

This report documents the completion of the milestone "FY17Q2 [MS-17/01] Better Dynamic Types Design" (listed as epic STDA05-1 in JIRA) for the ECP/VTK-m project. The overarching goal of the ECP/VTK-m project is to enable scientific visualization on the emerging processors required for the latest generation of petascale computers and the extreme scale computers of the future.

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.

Although there will be some time spent in the VTK-m project building up the infrastructure, the majority of the work is in redeveloping, implementing, and supporting necessary visualization algorithms in the new system. We plan to leverage a significant amount of visualization software for the exascale, but there is still a large base of complex, computationally intensive algorithms built over the last two decades that need to be redesigned for advanced architectures. Although VTK-m simplifies the design, development, and implementation of such algorithms, updating the many critical scientific visualization algorithms in use today requires significant investment. And, of course, all this new software needs to be hardened for production, which adds a significant overhead to development.

Our proposed effort will in turn impact key scientific visualization tools. Up until now, these tools — ParaView, VisIt, and their in situ forms — have been underpinned by the Visualization ToolKit (VTK) library. VTK-m builds on the VTK effort, with the "-m" referring to many-core capability. The VTK-m name was selected to evoke what VTK has delivered: a high-quality library with rich functionality and production software engineering practices, enabling impact for many diverse user communities. Further, VTK-m is being developed by some of the same people who built VTK, including Kitware, Inc., which is the home to VTK (and other product lines). Developers of ParaView and VisIt are in the process of integrating VTK-m, using funding coming from SciDAC and ASC. However, while VTK-m has made great strides in recent years, it is missing myriad algorithms needed to be successful within the ECP. Developing those algorithms is the focus of this ECP proposal.

2. MILESTONE OVERVIEW

The "FY17Q2 [MS-17/01] Better Dynamic Types Design" milestone produces the design of a new set of classes that introduces runtime polymorphic behavior in the parallel code of VTK-m.

For the best efficiency across all platforms, VTK-m algorithms currently use static typing with C++ templates. However, many libraries like VTK, ParaView, and VisIt use dynamic types with virtual functions because data types often cannot be determined at compile time. We have an interface in VTK-m to merge these two typing mechanisms by generating all possible combinations of static types when faced with a dynamic type. Although this mechanism works, it generates very large executables and takes a very long time to compile. As we move forward, it is clear that these problems will get worse and become infeasible at exascale. We will rectify the problem by introducing some level of virtual methods, which require only a single code path, within VTK-m algorithms.

This first milestone produces a design document to propose an approach to the new system.

3. TECHNICAL WORK SCOPE, APPROACH, RESULTS

The approach for this milestone is straightforward. We build a pair of design documents to outline the technical approach for the dynamic types implementation.

We started by creating an initial pass of the design document to introduce virtual methods in VTK-m array portals. Some preliminary work under experimental ASCR funding has explored some of the viability and options of the proposal. This first design introduced the idea of adding virtual functions to classes that are used during algorithm execution. (Virtual methods were previously banned from the so called execution environment.) The design was straightforward. VTK-m already has the generic concepts of an “array handle” that provides a uniform interface to memory of different structures and an “array portal” that provides generic access to said memory. These array handles and portals use C++ templating to adjust them to different memory structures. This composition provides a powerful ability to adapt to data sources, but requires knowing static types. The proposed design creates a template specialization of an array portal that decorates another array handle while hiding its type. In this way we can wrap any type of static array handle and then feed it to a single compiled instance of a function.

After some initial experimenting with this first design, we found that there was little overhead on x86 architecture (including Xeon Phi), but a marked increase in time on CUDA using virtual methods. Closer inspection revealed that most of the added time is spent in allocating memory from within a device, which was required to establish the virtual methods needed by our design. Thus, we created a second design to solve these performance issues. The second design focused on the mechanics of implementing virtual methods on parallel devices with a focus on CUDA. Rather than directly use the virtual methods implemented by the C++ language, we are using an alternate method provided by C that uses function pointers.

4. CONCLUSIONS AND FUTURE WORK

We consider the design documents as a good direction for our implementation. We will be using this design to guide our future work in implementing dynamic types in VTK-m.

ACKNOWLEDGMENTS

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of

the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.