Fermilab LDRD Proposal

Project Title: Preparing HEP reconstruction and analysis software for exascale-era computing

Principal Investigator: Marc Paterno

Lead Division/Sector/Section: SCD/CS/SSI

Co-Investigators (w/Institutions): n/a

Proposed FY and Total Budget: (summary of budget page in K dollars

	SWF	SWF OH	M & S	M&S OH	Contingency	Total
FY15						
FY16	$132.9	$113.5	$51.6	$12.0	$0.0	$310.0
FY17	$136.9	$116.9	$51.6	$12.0	$0.0	$317.4
Total	$269.8	$230.4	$103.2	$24.0	$0.0	$627.4

1 Project Description

All HEP experiments rely on event processing software systems to manage algorithms and data from detectors and simulations. The current generation is designed to run well on commodity compute clusters such as FermiGRID. The Office of Science is investing heavily in very large-scale computing centers and has been pushing hard for all branches of science to make use of these systems in the future. They are asking us to consider how software must change, assuming that CPU cycles are free and file I/O is costly. The future will be dominated by cores with little high-speed memory and with several tiers of slower memory, by ultra-high-speed networking, and by specialized resources for file I/O. Our software systems are not ready for this shift. The purpose of this research project is to produce a prototype software system for HEP event processing that demonstrates an architecture and a design that will be capable of scaling to greater than 100K cores and efficiently moving event data through these new high-performance computing platforms in forms necessary for algorithmic work. Fermilab's HEP framework developers do not have a record of research in this DOE advanced computing community. Success in this project will give us the foundation we need to participate in future DOE grant calls.

While porting exercises are underway for performing specialized subsets of HEP processing on current systems, no overall re-architecting has been started to address the imminent larger structural changes. We propose moving to a distributed memory, multi-process design to gain back memory space across the application, and to eliminate all I/O to physical storage, except at specialized aggregation points. We will leverage state-of-the-art R&D efforts focused on exascale computing, such as Legion [1] , Charm++ [2] , or MPI for distributed programming technology, and HDF5 [3] for parallel storage.

A successful demonstration will enable a path towards using the enormous compute power of the DOE-funded exascale facilities. Access to these facilities will be predicated on their efficient use and our current software system do not fit their constraints. The neutrino and muon programs could directly benefit from these available cycles, seeing significant reductions in turn-around time for large-scale processing tasks.

2 Significance

The P5 report notes a major computing challenge facing our field: "The present practice is to handle much of the computing within individual projects. Rapidly evolving computer architectures and increasing data volumes require effective crosscutting solutions that are being developed in other science disciplines and in industry."

The computing model for HEP experiments has been based around high-throughput computing (HTC) rather than HPC, as the computing problem is naturally parallel. However, the need to make efficient use of HPC resources drives the need to modify the computing model. In a discussion of the LHC experiments in particular, Bloom and Gutsche note:

> If we were to only take advantage of the expected growth in CPU power for fixed cost at 25% per year and in storage at 20% per year, we would expect deficits of a factor of four or twelve in CPU and a factor of three in storage under the assumption of fixed budgets for computing resources, even after accounting for potential algorithmic improvements. Clearly changes to the current paradigm must be considered. [4]

Our current computing model is a barrier to efficient use of the kinds of machines which will provide most of the available CPU cycles in the exascale era. If HEP experiments use just 5% of the compute capability of the next generation of ASCR HPC machines, it is approximately 25 times what grid resources will provide. [5]

The traditional HEP processing model for how data are moved between processing steps is from disk storage, to memory, back to disk storage: files on disk are the transfer mechanism between processes, and every computing node is expected to have significant I/O capability. This model is not consistent with the design of leadership-scale machines now in use. Such machines typically have dedicated, specialized I/O resources, rather than having I/O capability on each node. For example, ANL's Mira has 49,152 compute nodes but only 384 I/O nodes, and future machines will have an even greater imbalance between compute capability and I/O capability. [6]

Upcoming machines will have enormous data transfer bandwidth between compute nodes and I/O nodes (e.g. ANL's Aurora has a planned interconnect aggregate bandwidth of more than 2.5 PB/s, compared to a filesystem throughput of 1 TB/s. [7] To make efficient use of such machines requires that our frameworks support communication of the data through the interconnect networks.

The I/O systems on current and future machines use specialized parallel filesystems, such as Lustre, as well as specialized I/O libraries that can take advantage of the parallel filesystem. One of the premier software tools in this domain is HDF5, which is in use at many current HPC facilities. The HDF Group is actively pursuing its continued development.

Efficient use of even current commodity clusters could benefit from a reorganization of our software; the storage I/O capacity of affordable worker nodes is significantly less than that of larger capacity disk servers ($\sim$ 100 MB/s vs. $\sim$ 300 MB/s or more), while 10–40 Gb/s network interconnects are affordable. Processing chains that require writing to disk between steps use a significant fraction of their CPU resources performing I/O: a recent MicroBooNE performance review showed that about 50% of the program's time was spent in I/O. [8] Avoidance of unnecessary file writing could benefit multi-step workflows.

Through multiple trajectories, we are trying (and are being pushed) to approach HPC resources. Initiatives include (a) the Fermilab's "HEP Cloud" project, designed to seamlessly integrate HPC resources into existing job submission infrastructure, (b) an ANL leadership center, Computational Physics-funded, project to port the Muon g-2 simulation to run on current and near-future HPC resources, and (c) invitations to write proposals for developing an understanding of how to leverage extreme-scale (low-memory, high-bandwidth, limited storage I/O capacity) machines for HEP use. [9]

Different parts of the HEP community are moving forward to address efficiency problems with the traditional computing model in HEP. For example, CMS has re-organized their simulation and reconstruction workflow so that the files on disks local to a given node are re-used, on that node, for multiple steps of the workflow. This avoids superfluous copying of files, but it means that many steps of the workflow have to be carried out on the same node, and it still requires the I/O work. This limits how flexible job scheduling can be, and can lead to inefficiencies in grid resource usage. MicroBooNE uses a more traditional model, in which different phases of the workflow are handled as separate grid jobs; the requires moving data into and out of shared global/distributed storage between jobs. This allows the grid job scheduling to be flexible, but much time is then used for file I/O. The use of *xrootd* is a step toward making data transfer network-oriented, rather than filesystem-oriented. However, it does not move away from the existing organization of data on disk, and it does not at all address the writing of output.

While these projects all address, in some way, the HPC challenge, none of them fundamentally alter the structure of our framework programs, to bring them into alignment with the demands of exascale-era machines.

3 Research Plan

3.1 Overview

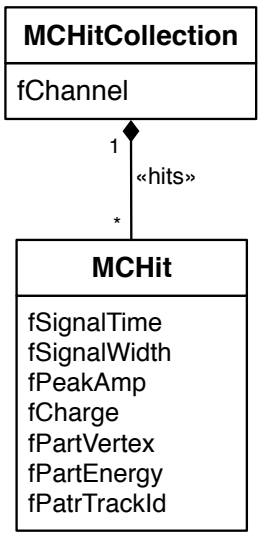
The primary objective of this project is to produce a prototype software system suitable for moving HEP experiment event data through multiple processing stages in an exascale-class computing facility. In particular, work will concentrate on demonstration of two critical components of a complete event-processing system: (a) high-performance I/O to a parallel filesystem, and (b) communication of event data through high-performance node interconnects (rather than through the filesystem), between processes of one application running across hundreds or thousands of nodes—a *distributed program*.

Thus the research plan is organized into three parts:

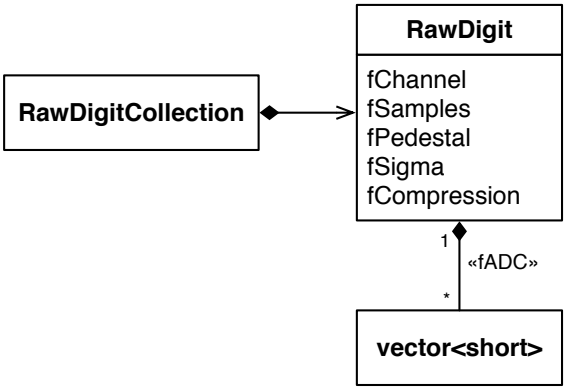
1. Production of a proof-of-concept I/O implementation for a selection of realistic HEP simulation and experimental data.
2. Production of a proof-of-concept implementation for a distributed event data product store, suitable for handling these same data on high-core-count, low-memory-per-core computing hardware.
3. Studying the scaling performance of the resulting systems, iterating on the architecture and design choices to optimize the system.

3.2 HDF5 R&D

Central to any event-processing program is the event data that is processed. In addition to algorithms for use in simulation and reconstruction of LArTPC data, the LArSoft toolkit also defines a set of *data products* used by them. The data products are defined as C++ classes; relevant for this work



(a) Simulated hits in the LArTPC detectors are represented by MCHit objects, each of which contains a variety of data. A single detector channel readout can be associated with many such hits.



(b) A RawDigit represents a single channel readout from a LArTPC detector, and consists of a collection of summary data, and the individual samples from the associated ADC. One event contains a collection of (typically) many such RawDigits.

Figure 1: Some of the basic event data classes in LArSoft.

are the data contained within those classes. Figures 1a and 1b show the data of some of the basic classes in LArSoft.

The first part of the project involves determining how to best encode such data products for storage in the HDF5 format, enabling the use of high-performance parallel filesystems, such as Lustre. The HDF5 format supports hierarchical structures, which is critical for describing the highly-structured HEP data. A challenging aspect of this task involves dealing with the *irregular* nature of much of HEP data: not all events have the same number of hits, not all tracks have the same number of hits, *etc.* HDF5 is traditionally used for storing hierarchical structures of *regular* data.

Users of modern HEP frameworks are accustomed to automatic generation of the I/O functions to support the classes they define. In the *art* framework, which underlies LArSoft, the primary I/O system is ROOT. Authors of data classes rely upon ROOT's facilities to take information from the C++ header file, describing the class, as well as a *selection* file that specifies additional information needed for ROOT's tools to generate the necessary I/O routines. A fragment of one of LArSoft's selection files is shown in figure 2; only the part associated with the RawDigit class from figure 1b is included.

In order for HDF5 to be a feasible addition to the family of I/O facilities, we will need a mechanism by which the necessary HDF5 I/O code will be generated. While ROOT does this in part by parsing the C++ header, a possibly more promising route is to devise a *data definition language* (DDL) for HDF5. Using a DDL, one describes the data layout of the class, and both the class header and the related I/O code would be generated. This has the advantage of making support for additional programming languages (such as the very popular Python) easier to add into the suite of available tools.

We will establish a "special project" with the HDF group, and work in collaboration with the HDF group on developing the organization of data and the software necessary to demonstrate the use of HDF5 for the storage of HEP data, including both the demonstration and performance

```
<lcgdict>
 <class name="raw::RawDigit     " ClassVersion="13"           >
  <version ClassVersion="12" checksum="160151695"/>
  <version ClassVersion="13" checksum="412021819"/>
 </class>
 <class name="std::vector<raw::RawDigit>       "                 />
 <class name="art::Wrapper< std::vector<raw::RawDigit>        >"/>
 <ioread
    version="[-11]"
    sourceClass="raw::RawDigit"
    source="unsigned short fChannel"
    targetClass="raw::RawDigit"
    target="fChannel"
    include="RawDigit.h">
    <![CDATA[fChannel = onfile.fChannel;]]>
</ioread>
</lcgdict>
```

Figure 2: The selection file used by ROOT's tools to control generation of I/O code related to the RawDigit class.

measurement of the prototyped I/O facilities, and prototyping of a DDL. We will obtain feedback from the LArSoft and *art* user community to guide this development.

The current HDF5 interfaces are implemented in C. This makes them usable from C++, which is what we require. But the C interfaces are not the most convenient to use from C++: they do not provide the efficiency, robustness, or ease of use possible with modern C++. If time and funding allows, we would also pursue the development of a native modern C++ interface for use of the HDF5 I/O facilities.

Three quarters are allocated for this work.

3.3 Distributed data model R&D

The I/O facilities for current and future machines will be concentrated in specialized I/O nodes, and the compute nodes will have no direct access to significant persistent storage. There will be many compute nodes for each I/O node; for example, Mira at ANL has 128 compute nodes for every I/O node. Thus it is necessary to have an efficient means to transfer the data using the high-performance interconnects between the compute and I/O nodes.

The second part of the project involves investigating state-of-the-art R&D efforts in exascale and distributed computing, and the production of a proof-of-concept distributed data store suitable for use with modern HEP event-processing frameworks.

Most modern event-processing frameworks (including both *art* and CMSSW, the two frameworks primarily developed at Fermilab) have, as a central data abstraction, the *Event*. The fundamental algorithmic building block of an event-processing program is the event-processing module, called in *art* a *producer*; a simulation or reconstruction program consists of a (possibly large) number of such producers. The *Event* is, as far as the framework is concerned, the atomic unit of data passed to a producer for processing. The *Event* provides access to the instances of the data products (described in the previous section). Simulation and reconstruction algorithms written by experimenters do

not obtain access to data products by directly reading an input file. Instead, the *Event* provides the interface through which access to data products is obtained. Data products may be "gotten from" or "put into" the *Event* through a small handful of API functions it provides. Algorithms encapsulated in modules do not call each other directly; instead, they work in collaboration by each algorithm getting from the *Event* the inputs it requires, and putting into the *Event* the products it generates.

In order to allow the construction of *distributed* event-processing programs without disrupting the natural and successful model of simulation and reconstruction algorithms as modules that interact with the *Event*, we need to provide an equivalent to the *Event* that provides access to a distributed data store—one suitable for communicating event data between the processes of a distributed program.

The first goal of this work will be to produce a prototype implementation of such a data store that can efficiently communicate event-product data from where they would be generated, on compute nodes, to where the final products of simulation or reconstruction would be written to storage, on the I/O nodes, making efficient use of the high-performance communication facilities available on the machine, using advanced features like Remote Direct Memory Access (RDMA).

In addition to the development of an event API suitable for a distributed environment, it is necessary to have an efficient encoding of the data products that minimizes communication overhead. For example, current data models make extensive use of pointers (bare memory locations), which are not easily communicated between processes which do not share the same address space. The I/O facilities we currently rely upon make heavy use of buffering, serialization, and compression to reduce the size of the resulting files, and to obtain platform independence, but these techniques are compute-intensive. On a homogeneous exascale machine, much of this is not needed, and precludes low-latency communication.

Three quarters are allocated for this work.

3.4 Scaling Studies

Because our concentration is on communication and storage of the data, we do not need to port real HEP simulation or reconstruction algorithms to the new platforms. Instead we will simulate the workloads, using realistic distributions of processing times and data sizes.

We will develop the infrastructure necessary to perform the scaling studies using Fermilab HPC resources. This includes verifying that the interaction between multiple processes on compute nodes and the I/O system on the I/O nodes works correctly, as well as some efficiency tuning in small-scale applications. It is critical to do this locally, because access to national supercomputing resources is limited.

When we have a sufficiently completed prototype for large-scale performance studies to be feasible, we will apply for startup awards as appropriate at NERSC, ANL, and ORNL. Each of these has a program for providing small awards for new projects: (a) *Startup* allocations at NERSC, (b) *Director's Discretionary Preflight or Startup Allocations of Time* program at ALCF, and (c) *Director's Discretion* grants at OLCF.

We will then deploy the prototype software on the available machines. We first target familiar x86 HPC platforms, so that we may more quickly begin performance and scaling studies. Primary questions these studies will address include: (a) How does the efficiency of data transfer between nodes vary with the number of nodes being used? (b) Can we predict the effective limit on the number of cores that can be used effectively? (c) What is the appropriate mix of "computing" and I/O tasks for efficient use of a given type of machine? (d) How many processing steps or algorithms must be combined in one program for reasonable resource use? (e) What should be the structure

of the "file"? How should event data be aggregated for efficient access? A valuable result of this testing will be the development of infrastructure that can be re-used to evaluate machines as they become available.

A final part of this work is to iterate upon the architecture and design, in order to optimize the prototype system. The most important figure of merit in this case is the throughput of the complete system: for a given computational load (reflecting the number and complexity of the algorithms being executed), this is the number of events per second can be processed by a given fraction of a given exascale-class machine.

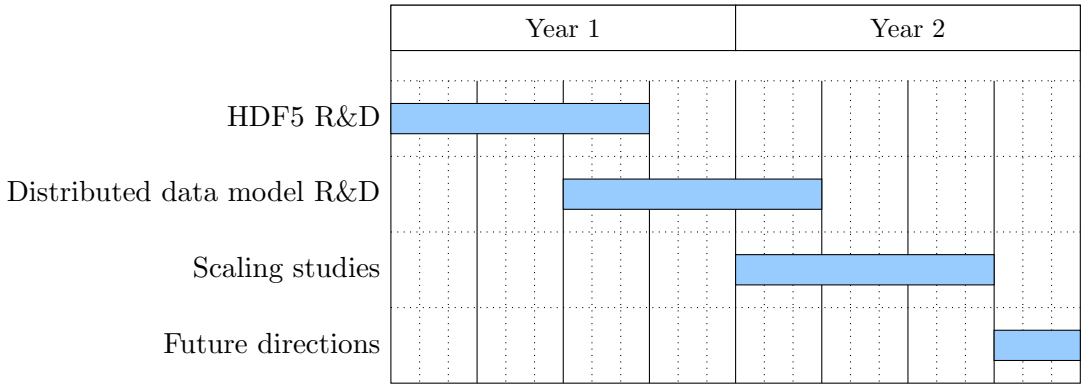
Three quarters are devoted to these tasks.

3.5 Report and future directions

In the final phase of the project, we will write up our results for publication. We will also develop a plan describing how HEP event processing frameworks (especially the *art* and CMSSW frameworks, developed mainly at Fermilab) can move forward to operate efficiently in the exascale era, based on what we have learned in this project.

One quarter is allocated for this work.

3.6 Schedule and Milestones



4 Dissemination of Results and Future Funding

We will pursue publication of our results in the fora that are important to the DOE supercomputing community. Foremost is the annual ACM/IEEE Supercomputing conference, and especially the workshops associated with it, for example DISC, the International Workshop on Data Intensive Scalable Computing. In addition we will pursue journal publication in an appropriate IEEE or ACM journal, such as *ACM Transactions on Parallel Computing*.

The DOE regularly opens calls for proposals for research into the use of extreme-scale computing technologies in the support of DOE science. One of the goals of this project is to gain some visibility in the DOE supercomputing community for this kind of work, thus increasing the likelihood of success for our responses to future calls.

5 Research Facilities

Development work will be done on the SSI department's development cluster, which consists of the following machines, connected via a 40G InfiniBand network:

- Five 4x8-core AMD 6128 (2.2GHz), 64GiB 1333MHz DRAM
- One 4x8-core AMD 6136 (2.2GHz), 64GiB 1333MHz DRAM
- One 2x10-core Intel E5-2680 (2.8GHz), 64GiB 1866MHz DRAM

These servers will be used to develop and verify correct behavior of the prototype software.

Access to a Luster filesystem and a modest number of computer nodes managed and maintained by the Fermilab HPC group for the LQCD and CC projects is available for initial performance testing.

Large scale performance testing, and scalability testing, will be done on ALCF, ORLF, or NERSC resources. We will apply for those resources in time for their use in the second year of this project.

References

[1] The Legion programming system, online at `http://legion.stanford.edu`.

[2] The Charm++ home page, online at `http://charm.cs.uiuc.edu/`.

[3] The HDF Group. Hierarchical Data Format, version 5, 1997-2015. `http://www.hdfgroup.org/HDF5/`.

[4] K. Bloom and O. Gutsche, *Non-Traditional HPC Use Case: Energy Frontier Experiment*,

[5] "HEP Forum for Computational Excellence (HEP-FCE)", S Habib and R Roser; `http://science.energy.gov/~/media/hep/hepap/pdf/20150406/day2/hep_fce_HEPAP_v5_Roser_Habib.pdf`.

[6] DOE ASCAC Data Subcommitte, textitSynergistic Challenges in Data-Intensive Science and Exascale Computing, tech. report, US Dept. of Energy, Mar. 2013; `http://science.energy.gov/~/media/ascr/ascac/pdf/reports/2013/ASCAC_Data_Intensive_Computing_report_final.pdf`.

[7] Aurora Fact Sheet, `http://www.intel.com/newsroom/assets/Intel_Aurora_factsheet.pdf`.

[8] "Profile report of the reco-2D and reco-3D stages of the MicroBooNE reconstruction software", K. Knoepfel, J. Kowalkowski and P. Russo, Fermilab CS-doc-5579-v1, `https://cd-docdb.fnal.gov:440/cgi-bin/RetrieveFile?docid=5579&filename=MicroBooNE_profile_report.pdf&version=1`.

[9] Advanced Scientific Computing Research (ASCR) "Storage Systems and Input/Output (SSIO) for Extreme Scale Science", DOE National Laboratory Announcement, LAB 15-1338, `http://science.energy.gov/~/media/grants/pdf/lab-announcements/2015/LAB_15-1338.pdf`.