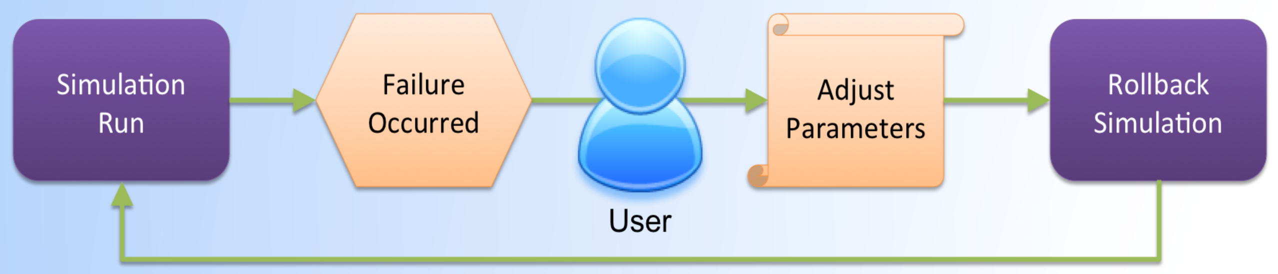
Improving Simulation Data Processing Pipeline using Conduit via HDF5

Gavin Sonne[1], Ming Jiang[2], Brian Gallagher[2], Daniel Laney[2], Cyrus Harrison[2]
[1]Hartnell College, UC Santa Cruz; [2]Lawrence Livermore National Laboratory

Background:

Arbitrary Lagrangian-Eulerian (ALE) Codes

- Simulation workflows are complex.
- Involves tuning which requires significant user time and effort.
- Process that relies heavily on the knowledge of the user.

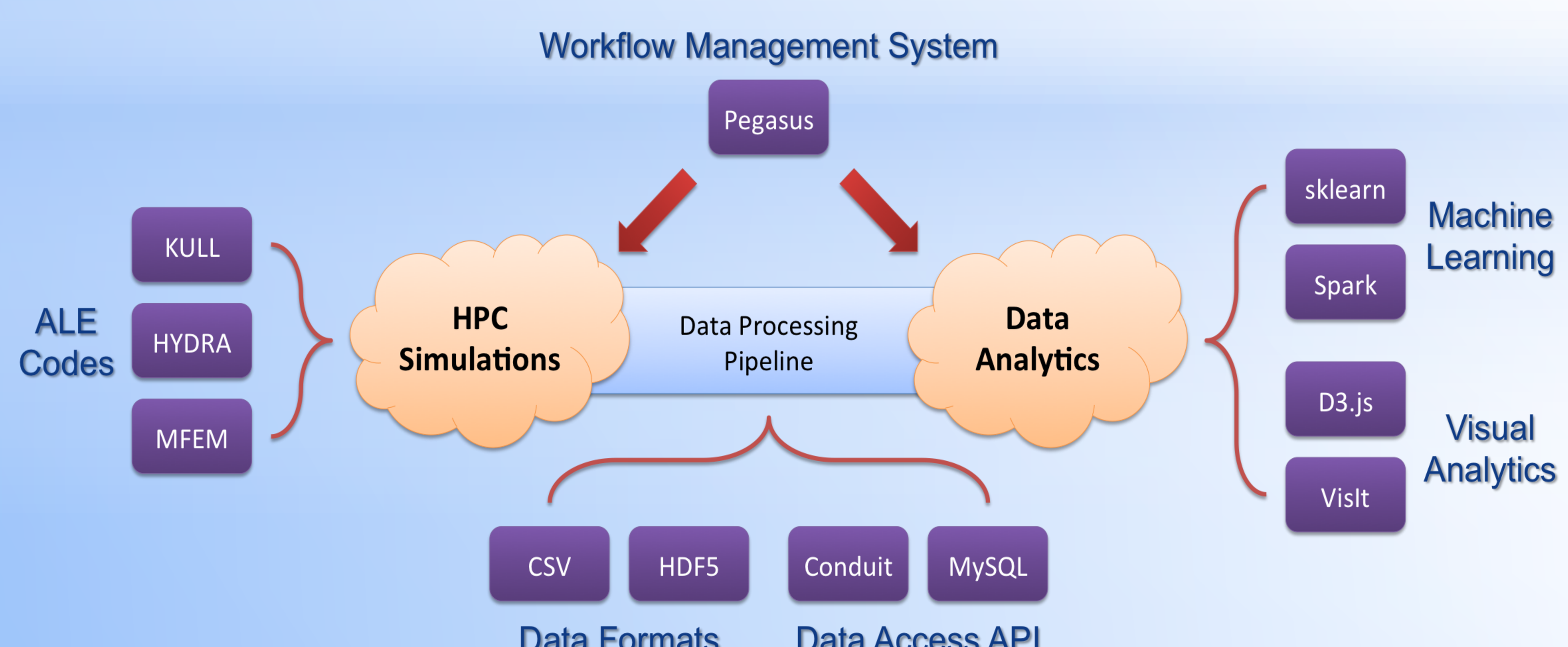


Learning Algorithm-Generated Empirical Relaxer (LAGER)

- Semi-automate this process by using a data analytics approach to reduce reliance on the user.
- Exploit machine learning and develop novel data visualization techniques.
- Develop an *in situ* infrastructure which runs predictive analytics alongside the simulation so that failures can be avoided.

Project Goals:

- Preliminary work to establish data processing pipeline between simulation and analytics.
- Convert simulation data into the Hierarchical Data Format 5 (HDF5).
- Implement HDF5 reader with NumPy array output.
- Implement Conduit with HDF5 Output.
- Evaluate Conduit's potential as a data exchange method at simulation run time.



High Level Overview of the Workflow Management System

HDF5:

Currently, the data that is extracted from the simulation code is stored in comma separated value (CSV) text files. HDF5 is a file format used to organize and store large amounts of numerical scientific data. A python program was designed that parses the CSV data, and writes it into an HDF5 format using the h5py library. In this implementation, each item of floating point data is reduced from double to single precision, which decreases the output file size by half. Header metadata and information about the HDF5 file are stored as attributes.

```
def write_tbl(outputFile, headerList, dataArray, stepCount, stepList):
    dset = outputFile.create_dataset(("Timestep%05d" % (stepCount-len(stepList)) + "_%05d" %
(stepCount - 1)), data=dataArray, dtype='<f4')
```

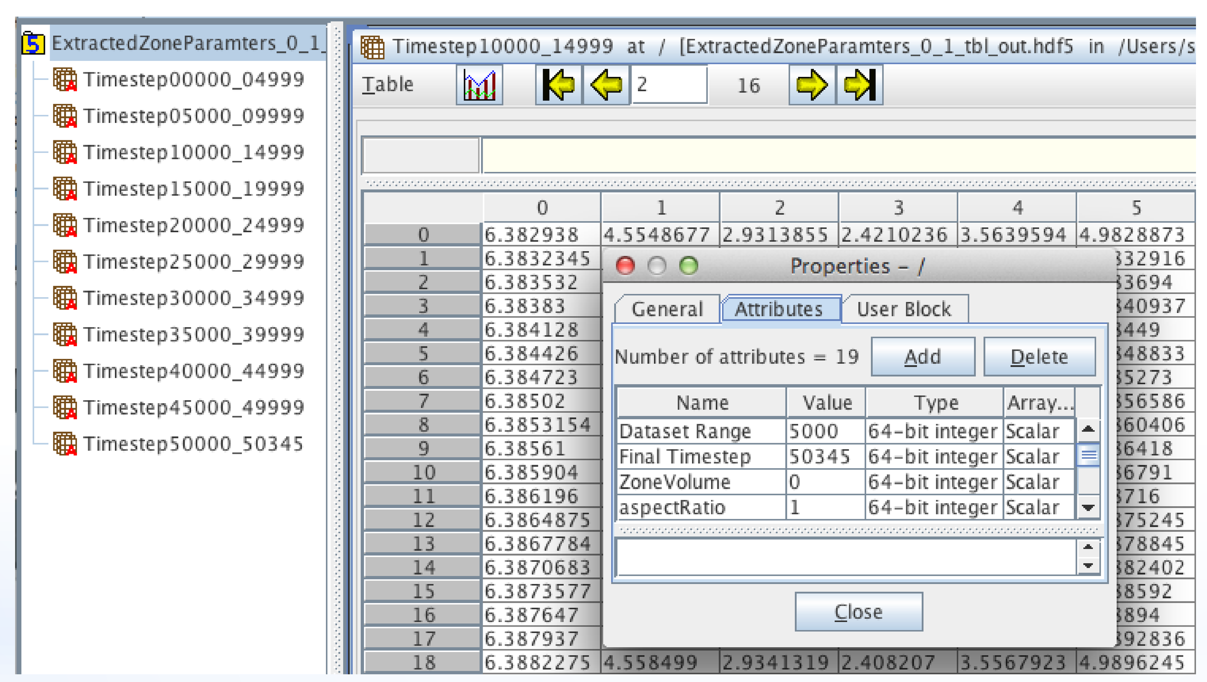
Function in the parser that writes HDF5 Datasets

```
flax:lager sonne1$ ls -l ExtractedZoneParamters_0_1_tbl_out.hdf5 ExtractedZoneParamte
-rw-r--r--  1 sonne1  55445  1256648544 Jul 28 15:29 ExtractedZoneParamters_0_1_DOUBL
-rw-r--r--  1 sonne1  55445   628330464 Jul 28 15:13 ExtractedZoneParamters_0_1_tbl_
```

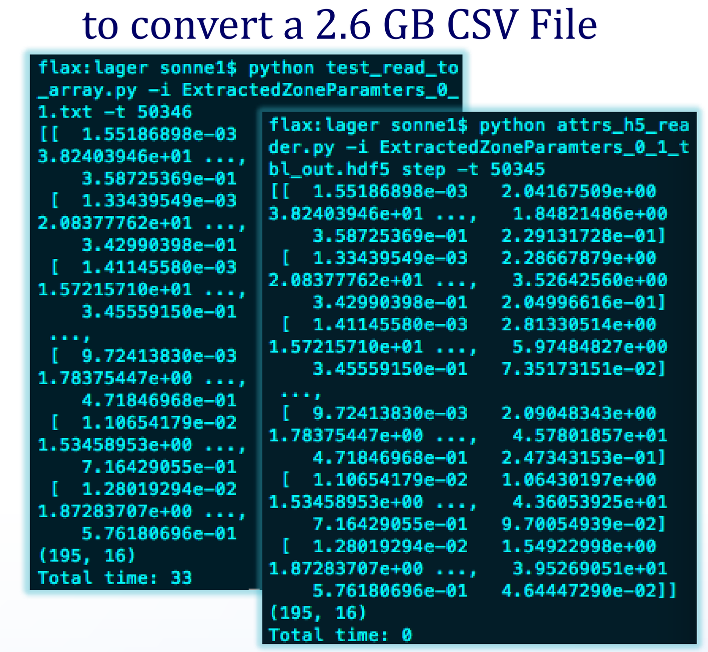
File size comparison of single vs. double float precision

```
Converting CSV file ExtractedZoneParamters_0_1.txt
Writing steps 0-4999 to file.
Writing steps 5000-9999 to file.
Writing steps 10000-14999 to file.
Writing steps 15000-19999 to file.
Writing steps 20000-24999 to file.
Writing steps 25000-29999 to file.
Writing steps 30000-34999 to file.
Writing steps 35000-39999 to file.
Writing steps 40000-44999 to file.
Writing steps 45000-49999 to file.
Writing steps 50000-50345 to file.
Total time: 113
flax:lager sonne1$
```

Parser takes 113 seconds to convert a 2.6 GB CSV File



Generated HDF5 File shown in HDFView, with attributes



Returning array from final time step:
- From CSV takes 33 seconds
- From HDF5 takes 0.1 seconds

With the data converted to an HDF5 format, it is easier to work with and manipulate. For a user who wants access to the data, a program was designed that returns a two dimensional dimensional array for a particular time-step, zone, or feature of the simulation code. Extracting these arrays is much less costly when working with HDF5 as opposed to the original CSV file. This is because HDF5 essentially stores data as NumPy arrays that can be indexed. On the contrary, when working with a CSV file, in the worst case (seeking to the last line of the file), the program must traverse the entire file to get at the data requested by the user. This is illustrated in the lower right figure above.

Conduit:

- A flexible way to describe complex in-core data.
- A C++ API for accessing that data.
- Designed to be adaptable to other languages, such as Python.
- Designed for in-core data exchange and sharing, thus has the potential to be very useful in the future as a component of the data processing pipeline between simulations and data analytics.

Discussion:

It has been demonstrated that HDF5 is a very useful format for storing very large amounts of data produced by simulation codes. More interestingly, it is very useful to have data in this format when a user needs to be able to extract specific sections of the data in order to work with it. Additionally, once the data is in this format, accessing it can be done much more efficiently than doing so when the data is stored as a CSV.

Continuing Work:

In the time remaining, the plan is to take an existing output from Conduit used with the Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics mini-app (LULESH) and output the data in HDF5 instead of using the SILO library.

Both this research group and the developers of Conduit are interested to know whether Conduit data can be directly and easily exported into HDF5 and back again with little to no additional manipulation of the data. This avenue will be explored very soon.

Another future goal is to insert Conduit into the simulation codes for our data analytics. Another is to output directly to HDF5 and skip the costly and inefficient step of writing the data out to cumbersome CSV files.

Acknowledgements:

Many thanks to Joshua Kallman, Jay Salmonson, Joe Koning, Luc Peterson, Judy Thomas, and Amy Huang. Most of all, thank you to LLNL for providing the opportunity to do this research.

This work performed under the auspices of the U.S. Department of Energy and an appointment to the Office of Science, Community College Internship (CCI) Program at the Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LLNL-POST-675649