University of Victoria
Engineering & Computer Science Co-op
Work Term Report
Summer 2022

SmartSim Dataset Conversions for Simulation Visualization and Analysis

Hewlett Packard Enterprise
Seattle, WA, USA

Julia Putko
V00889506
Work Term #3
Computer Science
juliaputko@uvic.ca
August 24, 2022

In partial fulfillment of the academic requirements of this co-op term

This report is the review of work completed at Hewlett Packard Enterprise in Seattle, WA, USA, during my third work term at the University of Victoria, for the completion of my fourth year of a Computer Science degree. My work consisted of working on API design for a software package called SmartSim, working to complete data conversion methods for SmartSim datasets. This report will focus on the research and iteration on design requirements, and the final method design. This report will also cover the investigation into data formats, why Xarray was the primary data format being chosen for the purposes.This report will be divided into two major aspects required to reach the final implementation: data format research and prioritization, and method design. Data format research and prioritization involves researching several aspects of data formats, including which data formats are commonly used by the people who might be using SmartSim, and what conversion into the data format would allow. The method design required many iterations of a design document in order to write a method that will integrate seamlessly into the SmartSim and will perform the appropriate functions without disrupting the existing API and functionality of SmartSim. The result of the research and design was the decision to prioritize Xarray as the data format for conversion from a SmartSim dataset. The method was decided to be split into two steps in order to preserve the existing way that users are able to interact with a SmartSim dataset. This report will focus on SmartRedis rather than SmartSim, and will not look at the deeper workings of either other than general information required to understand the method design.

I would like to thank my manager, Matt Ellis, my work term supervisor, Andrew Shao, the other SmartSim team members: Alessandro Rigazzi, Amanda Richardson, Bill Scherer, Matt Drozt, as well as Alyssa Cote and Nikhil Nunna, members of the management team at HPE: Ben Robbins, Richard Korry, Sheryl Awe, and all other interns I was able to interact with within HPE for their patience, support, and willingness to provide feedback and review changes and additions made.

Sincerely,

Julia Putko

Executive Summary

The focus of this report is to determine how to simplify the ability to retrieve a SmartSim dataset from the database within SmartSim and convert it into a more commonly used data format that is accessible for analysis and visualization. This report includes: the investigation into several dataformats, the reasoning behind why Xarray was determined to be the primary data format being chosen for the purposes, and a description of the final method design. Currently SmartSim users can only retrieve SmartSim datasets using the existing API. Users must unpack tensors and metadata and must manually convert them. Providing convenient methods to convert these datasets into common formats would provide users with the functionality and software that the data format provides. NetCDF, and Xarray are determined as the priority for dataset conversion, and the description and prevalence of use are documented for the final justification for the decision to prioritize Xarray as the initial focus for the preliminary dataset conversion method. The final design of the dataset conversion method was decided as being a two-step process: 1) where additional metadata is added to an existing Dataset to add the possibility of conversion and 2) a transformation method which ingests a Dataset and returns an object in the native data format. The add_metadata_for_xarray() method performs this first task, allowing the transform_to_xarray() method to identify which fields should be used in the construction of the specific data format. The method design and the decision to prioritize Xarray took into consideration several requirements that had to be achieved before approval.

Professional Reflection

My role during my work term included researching data formats and assessing their priority for conversion from a SmartSim dataset into the format, writing a design document for method design, and writing the method and merging it into the existing SmartSim codebase. I was able to develop a general understanding of the code base for SmartSim, research and understanding the purpose, structure, and contents of past design documents for method design, accumulated research on various data formats and their relevance and limitations for the purpose within SmartSim, used my findings to decide on a dataformat to focus on, developed a design document for the conversion from a SmartSim/SmartRedis dataset into that data format, and then developed the method for the conversion. I wrote the methods in python, using a basic git workflow to submit pull requests, and receive reviews on my work. I had experience writing code that was up to a production standard for eventual merging into the SmartSim codebase for an upcoming release. I am actively working on writing clean code, and writing tests that cover expected failures and successes. I was able to move from having a general understanding of SmartSim, to doing research on data formats, making a decision for which data formats to prioritize and writing justification behind my decision. I was given the opportunity to iterate on the design of a method for data format conversion from SmartSim's native format into another data format, while developing an understanding of how the design will integrate into the SmartSim environment, and finally writing a method for one of the chosen data formats in python. I was able to put into practice technical documentation and design experience within software engineering. I was able to advance my skills in modeling a well rounded and well communicated understanding of

the problem and possible solution, and then being able to translate my design into code. I had some exposure to high performance computing systems, being walked through the steps of running a simulation model on a supercomputer. I had the opportunity to see the process of setting up and running simulations on high performance computing systems, and was given opportunities to understand how the SmartSim library enables online analysis and machine learning in HPC systems. I am interested in the different applications of supercomputers and the large amounts of data processing that is required for simulating processes, and would be interested to explore similar work in the future. I participated in agile software development, including sprint planning, standup meetings, and demonstrations and updates of work. My work during my work term involved collaboration within a multi-person team, and required being able to participate and work effectively with others. The work I completed required constant iteration on design, which required me to remain flexible and open to different methods of problem solving. I was ultimately able to participate in conversation on method design and method creation. I was able to observe what software development looks like for industry within a large company, and the level of code design and quality that goes into the ultimate product. I enjoyed working within a smaller team within a large company, which adjusted my expectations to what I might look for in a company I work for in the future. The course material in my classes, especially on software development methods, software design, operating systems, and my classes in python prepared me for certain aspects of this work term. The skills I have observed and have been able to practice during this work term have given me a better understanding of software development as a whole, from research and design to final implementation and iteration of code.

Table of Contents

List of Figures

Glossary

Conversion	Refers to a larger multi-step workflow where a user can create a SmartRedis dataset which can then be interpreted later (on the Python side) and transformed into a {Dataformat} object.
Data Format	A method of data storage, sometimes with associated metadata
SmartSim	A software library consisting of SmartSim (for the deployment of HPC workloads) and SmartRedis (a collection of Redis clients that include features of high performance computing)
SmartRedis	A collection of Redis clients that include features of high performance computing
SmartSim/SmartRedis dataset	A multidimensional array or tensor in fortran, C, C++, python.
Tensor	A multidimensional array
Xarray	a data format and a python package that allows for analysis and visualization
Xarray DataArray	N-dimensional array with labeled coordinates and dimensions.

1.0 Introduction

The process of research and design is necessary for the creation of methods to convert a SmartSim Dataset into a data format convenient for visualization, analysis, and manipulation. SmartSim is divided into two parts, SmartSim, the infrastructure library, is used for the automation of the process of deploying high performance computing workloads. [1][2]. SmartRedis is what is being used to push/pull data to/from the database (referred to as the 'orchestrator') by the simulation and use visualization and analysis tools like jupyter or matplotlib to plot the results. Relevant to this report is that SmartSim provides scientists a flexible, easy to use method for interacting with the data generated by numerical simulation models while they are running. Simulation models use SmartRedis to send data from simulation to the orchestrator database to be stored, and then the python SmartRedis client can be used to retrieve data from the database. The ability to visualize and interpret simulation data in real time is valuable in understanding the behavior of what is going on in a simulation model, for example the behavior of a physical system. A SmartSim Dataset is a multidimensional array, or tensor in Fortran, C, C++, python with associated metadata). Currently SmartSim users can only retrieve SmartSim datasets and must unpack tensors and metadata using the dataset API to incorporate SmartSim into their current workflows. The purpose of creating a data conversion method is to alleviate this requirement and lower technical barriers to integrating data visualization and analysis. The capabilities exist within Smartsim, but including a method for this conversion would make the process more compact and usable. This report will focus on one popular data format that has been selected as the

priority, Xarray. However the intention is to expand these methods for several other data formats.

2.0 Data Format Research

Data formats enhance the usability of data, giving users the methods to analyze, visualize and transform the data. Data formats overcome the limitations of a SmartSim dataset, allowing data management with methods that are familiar and well documented. The dataformats prioritized were determined by looking at which data formats support [3], the data formats used by the current SmartSim user base in the SmartSim zoo, and popularity within HPC systems. Converting SmartSim datasets into common file formats would give SmartSim users the tools and experience to perform data analysis and visualization. Based on these criteria, we will prioritize implementing support for Xarray, NetCDF as they are commonly used and convenient for data analysis and visualization

Other formats are being considered, however have been omitted from this report as it is proprietary information.

2.1 NetCDF

NetCDF is a file format and a set of software libraries. It includes the support of a wide variety of software and analysis tools for manipulation, display, and the ability to perform operations on subsets of data [4,5]. NetCDF is a common data format for analysis and visualization of climate, weather, forecast, atmosphere, surface and ocean data. It is the preferred data format for science analysis, storage, 3D visualization (map projections,

animations, time series and plots). It contains variables, dimensions and attributes, and is easily portable across various computer platforms, which promotes the processing and sharing of files. NetCDF files need dedicated software or libraries to be able to read or visualize data stored.

2.2 Xarray

Xarray is a python package for working with multidimensional arrays, not a data format. A conversion from the SmartSim dataset to the Xarray would still be required in order to use this python package. Manipulations and analysis can be made as Xarray datasets, or they can be easily converted into other formats for additional data analysis or storage. NetCDF is the recommended method to store Xarray data structures, however Xarray also supports reading and writing from Zarr, pickle, OPeNDAP, Rasterio, GRIB format via cfgrib, PyNIO, PesudoNetCDF, CSV + pandas, and MongoDB. Xarray provides methods for data access, manipulation and visualization, as well as supporting an extensive list of projects that build functionality upon Xarray[6]. Xarray is popular in the geosciences, machine learning, biomechanics, thermodynamics, and visualization for large data.

Xarray provides the capabilities for data manipulations, analysis and visualizations, as well as convenient conversions into other formats for additional data analysis and storage. Creating a method to convert to Xarray provides users with the capabilities that Xarray provides.

3.0 Method Design

Dataset conversion refers to a multi-step workflow where a user can create a SmartRedis dataset. which can then be retrieved on the Python side, and then transformed into a {Dataformat} object. An Xarray Dataset conversion, in this case, performs the conversion into the Xarray Dataarray object. The first step to accomplish our goal of converting a SmartSim dataset to an Xarray DataArray was to create a design that assured that these methods did not interfere with the existing dataset API to extract and manipulate data. Using these methods, the intention is to allow the user to retain the ability to access all the metadata and tensors by their original field names to avoid mangling under the hood, retaining accessibility for multiple levels of user. We want the methods support attaching data and metadata to a tensor within the SmartRedis dataset, support the ability to define coordinates of each dimension of a tensor added to a dataset, support adding multiple tensors into a SmartRedis dataset and storing their common metadata.

Several alternative method designs have been omitted as it is proprietary information.

3.1 Final Method Design

The result of the iterations on the interface design resulted in the decision to create a two-step data conversion process: The first is an add metadata method where additional metadata is added to an existing dataset to allow for the possibility of conversion. The second step is the transformation method. The add_metadata_for_xarray() method

performs this first task, allowing the transform_to_xarray() method to identify which fields should be used in the construction of the specific data format.

```
// add meta_data_for_xarray interface
add_metadata_for_xarray(dataset, data_names, dim_names,
coord_names=None, attr_names=None)
```

Figure 1. Final design add_metadata_for_xarray function signature.

```
// transform_to_xarray interface
transform_to_xarray(dataset)
```

Figure 2. Final design transform_to_xarray function signature.

We expect users to construct the datasets themselves using the Dataset API before calling the add_metadata_for_xarray() method. Only the field names will be passed into add_metadata_for_xarray(), so the actual structure of the dataset and any of the metadata will not be affected after calling the method. Being explicit about field names and how the SmartRedis dataset is being built while allowing for a general method to construct the Xarray DataArray provides transparency for the user while maintaining a simple API design that can be extended to further data format. With this approach, the user's ability to extract data (metadata, etc.), remains intact after any conversion.

Separating the adding of the metadata and the transformation into the appropriate data format minimizes the SmartRedis interference with the existing dataset.The ability to

extract data (metadata,tensors, etc.) by their original field names remains intact after any call to ``add_metadata_for_xarray()``. Adding metadata for compatibility means that the same dataset can be converted into multiple different data formats and so will allow us to support additional format types in the future.

We expect users to construct the datasets themselves using the Dataset API before calling the ``add_metadata_for_xarray()`` method.
Only the field names will be being passed into ``add_metadata_for_xarray()``, so the actual structure of the dataset and any of the metadata will
not be affected after calling the method.

```
ds1 = Dataset("ds-1d")
dataset.add_tensor("1ddata",data1d)
dataset.add_tensor("x",longitude_1d)
dataset.add_meta_string("x_coord_units",'degrees E')
dataset.add_meta_string("x_coord_longname",'Longitude')
dataset.add_meta_string("units",'m/s')
dataset.add_meta_string("longname",'velocity')
dataset.add_meta_string("convention",'CF1.5')
dataset.add_meta_string("dim_data_x","x")
```

Figure 3. An example of the creation of a SmartRedis Dataset and addition of tensor data and metadata done by the user.

```
    # Calling method add_metadata_for_xarray on the created
dataset
  DatasetConverter.add_metadata_for_xarray(
        ds1,
        data_names=["1ddata"],
        dim_names=["dim_data_x"],
        coord_names=["x"],
        attr_names=["units","longname","convention"]
    )
    # Calling method add_metadata_for_xarray for longitude
coordinate
    DatasetConverter.add_metadata_for_xarray(
        ds1,
        data_names=["x"],
        dim_names=["dim_data_x"],
        attr_names=["x_coord_units","x_coord_longname"]
    )
```

Figure 4 . An example of the ``add_metadata_for_xarray()`` method calls to pass in field names of data and metadata of the created SmartRedis Dataset under the appropriate parameter names for the creation of the tensor data variable for the Xarray object and the coordinate data variable for the Xarray object.

```
    xarray_ret = DatasetConverter.transform_to_xarray(ds1)
```

Figure 5. An example of the returned dictionary of the ``transform_to_xarray()`` method.

```
{'1ddata': <xarray.DataArray '1ddata' (x: 10)>
    array([0.75239102, 0.87698733, 0.57916855, 0.59621001,
0.22552972,
        0.17998833, 0.27133364, 0.3092101 , 0.82813876,
0.00731646])
    Coordinates:
    * x         (x) float64 0.0 40.0 80.0 120.0 160.0 200.0
240.0 280.0 320.0 360.0
    Attributes:
        units:       m/s
        longname:    velocity
        convention:  CF1.5}
```

Figure 6. The return of transform_to_xarray method: a dictionary with the keys being the name of the dataarray, and the value being the Xarray DataArray.

Using these methods we can significantly shorten the amount of coding that the user must do to get to the stage where they can reconstruct and plot their simulation data. The design and philosophy chosen here to support Dataset conversions is format generic, and so will allow us to support additional format types in the future.

Conclusion and Recommendations

Through the investigations done in this report, a data format was prioritized, and a method was designed that does not interrupt the current user workflow and capabilities within SmartSim to allow users to convert into a data format converter for visualization

and analysis. Gathering appropriate information and condensing research into a form that is understandable to team members that may not have deep knowledge of data formats is crucial to be able to communicate information so that decisions can be made on which data format to prioritize. There are hundreds of different data formats, so having criteria for how to condense it is crucial. The importance of prioritization was based on several factors and it was important to choose data formats to prioritize that provided capabilities for visualization and analysis and were used in relevant domains. Understanding the requirements and the scope for method design is important for coming to final decisions on method design. Understanding the existing SmartSim codebase, how users currently interact with a smart redis dataset and how they should interact with the method before, during and after using the conversion methods is crucial to understanding requirements and the final design. The final method takes into consideration the user flow and code base and existing way that users interact with a SmartSim dataset. The design of the method took into consideration the intention to perform conversions to several other data formats. Further work should be done to extend the method to the other dataformats mentioned, netCDF, HDF5, for further access to the visualization and analysis tools that these data formats provide or provide access to.

Acknowledgements

I would like to thank my manager, Matt Ellis, my work term supervisor, Andrew Shao, the other SmartSim team members: Alessandro Rigazzi, Amanda Richardson, Bill Scherer, Matt Drozt, as well as Alyssa Cote and Nikhil Nunna, members of the management team at HPE: Ben Robbins, Richard Korry, Sheryl Awe, as well as all other

interns I was able to interact with within HPE for their patience, support, and willingness to provide feedback and review changes and additions made.

References

[1] ”SmartSim,” *Cray Labs, Hewlett Packard Enterprise* [Online]. Available: https://www.craylabs.org/docs/overview.html. [Accessed 25 August 2020]

[2] “SmartRedis,” *Cray Labs, Hewlett Packard Enterprise,* [Online]. Available: https://www.craylabs.org/docs/smartredis.html. [Accessed 25 August 2020]

[3] Thomas, R., Stephey, L., Greiner, A., Cook, B. “Monitoring Scientific Python Usage on a Supercomputer,” *Proc. of the 20th Python in Science Conf.* SCIPY 2021. [Online]. Available: https://conference.scipy.org/proceedings/scipy2021/pdfs/rollin_thomas.pdf. [Accessed 23 August 2020]

[4] “Software for Manipulating or Displaying NetCDF Data,” *UCAR, unidata,* [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/software.html. [Accessed 23 August 2020]

[5] “Data Resources: Display and Analysis tools that read PSL’s netCDF files,” *Physical Sciences Laboratory,* [Online]. Available: https://psl.noaa.gov/data/gridded_help/tools.html. [Accessed 23 August 2020]

[6] “Xarray related projects” July 2022. [Online]. Available: https://docs.xarray.dev/en/stable/ecosystem.html. [Accessed 23 August 2020]

General References

Folk, M., Heber, G., Koziol, Q., Pournmal, E. ”An overview of the HDF5 technology suite and its applications,” *The HDF Project* ,March 2011 [Online]. Available: [https://www.researchgate.net/publication/221103412_An_overview_of_the_HDF5_technology_suite_and_its_applications. [Accessed 23 August 2020]

“File formats for VTK version 4.2,” *VTK User’s Guide,* [Online]. Available: https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf [Accessed 23 August 2020]