**Title:**

**Principal** **Investigator**: Scott A. Klasky

**Institution** Oak Ridge National Laboratory

**PI Phone Number, PI Email Address:** 865-274-7050, klasky@ornl.gov

Funding Opportunity Announcement Number: DE-FOA-0001338

**List of all co-PIs and Key/Senior Personnel, with their organizations.**

Hasan Abbasi, Kimmy Mu, Computer Science and Mathematics Division, ORNL

Post Doc, Sudharshan V. OLCF, ORNL

Jay Lofstead, Lee Ward: Sandia National Laboratory

Manish Parashar: Rutgers University

Main Ainsworth: Brown University

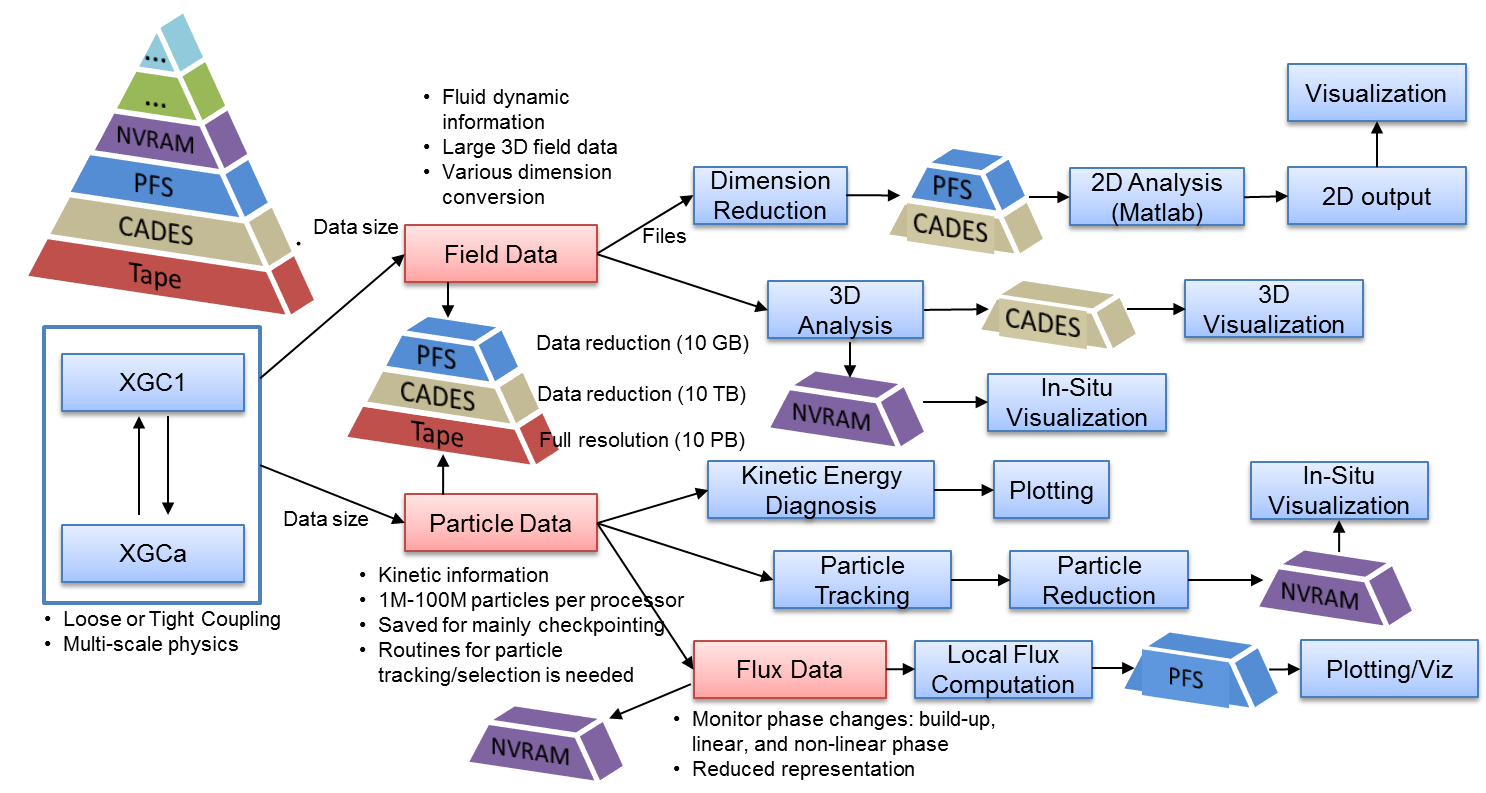
This proposal covers topic two: Scalable Storage Software Infrastructure:

**Executive Summary**

We propose to research and prototype a novel middleware and storage infrastructure system that can effectively manage data objects across the dramatically changing data storage hierarchy. We envision this system, taking user and system provided metadata and seamlessly mapping data objects across the storage hierarchy in an efficient and importance-driven manner. (Do we need to mention entropy-based lossy/lossless data reduction here? Others?)

Our research will focus on techniques to ultimately reduce the “time to knowledge” in data-intensive scientific workflows. One particular focus will be to create and mange importance-driven “areas of interest” and ensure that the data objects are appropriately mapped throughout the middleware and storage hierarchy. These research outcomes will provide insights, as well as building blocks that will enable efficient and flexible of data storage and retrieval for data-intensive sciences across the DOE computational and experimental facilities. Our approach will be developed within the context of four concrete archetypal application drivers: 1) fusion (XGC1, GTC), 2) accelerator science (PIConGPU, Warp), 3) material science (QMCPack), and 4) computational seismology (SPECFM3DGLOBAL). The research approaches and prototypes will be guided by the requirements of high performance data facilities and be adaptable to evolving architectures, algorithms, and programming paradigms.

As an example of prototype system, an example workflow for our fusion driver is shown below.



Key pieces:

1. ADIOS, our research and production middleware platform, provides an efficient and scalable representation of data objects in the memory and storage layers. We will conduct research into extending this support of data across all layers of the emerging storage hierarchy. In particular, to provide support for the representation and interaction with a single variable in a data object over the various storage layers, this research will leverage Sirocco for storage management, and data object layering. Additionally, we will research and extend the metadata catalogs that are partially supported in ADIOS to manage data objects, and provide a consistent view to applications.
2. We will research a new system, and appropriate API that allows application users to express their intentions with respect to the data objects. These intentions can then be abstracted and used to optimize the organization, placement and retrieval of data across the storage hierarchy. These optimization will take the user intent, and the storage system constraints (lifetime, bandwidth, latency, size, etc) and ensure an optimal layout and usage of the required data objects.
3. We will investigate both lossy and lossless compression techniques that can reduce data effectively without compromising important physics within data objects. The will allow data to be flexibly placed onto capacity-limited storage devices, and migrated between adjacent storage layers for performance reasons.
4. The capture and storage of provenance information will be integrated alongside the user intent data and placed in a central metadata catalog. This information can then be used on a per-user basis to predict data access requirements, and thereby tune the performance of the system to enable much faster time to solution.

More rambling…

I can see from our example figure that data is categorized into multiple pieces:: 1) there is the mesh, which contains information from many variables in the code, along with Jacobians to move data from one coordinate system to another, and relationships to other variables which are defined on the mesh. We also see that there are particle information which are not aligned to the mesh but contain the bulk of the data. We know that data on the fields can be “re-generated” from the particle information, but we can use information from the fields to help classify which “particles” are “more important” and which particles are “less important”. This means that we have relationships from the different variables. Furthermore, we know that if a piece of the workflow request a mesh, then it is very likely to want other fields which are defnined on that mesh, and not for example, particle information. This means that there are ways to predict the information. Furthermore we can understand that field information (defined on a mesh) can be analyzed by meshing errors (Mark help), which can define various regions of intrest on the mesh which are viewed with steep gradeints, and that information can be saved on faster storage, and the other information (the builk of the data hopefully) can be placed on slower areas. This means that users need the ability to plugin their data processors, but the system should have default methodologies to bucket their data object into various sub objects.

During the data lifecycle, we want to save the provenance of what happened during the in situ processing (for example) which can allow playback and vitial information for people who are looking at the data later. We also want the ability to re-generate data, through checkpoint-restart capability. But it is here that data can be thought of as coming from solutions of equations which can be re-produced with a certain level of accuracy. This means that the storage system needs the ability to re-create data when necessary, rather than store and retrieve this. The system characteristics can help this to take place, by understanding what can easily be reproduced and which are much more difficult.

We also have the ability to help users write not just when they say write, but when the system is happy with the write. Since the future systems (Summit) have a limited about of writes that the OLCF/IBM will allow due to the nature of NVRAM, users need the ability to understand this through their intentions, and combing that with the system knowledge. This tradeoff becomes critical…. If a user wants to write all of their physics information (for example), they may see this as being reasonable (1 PB/day on today’s system, 10 PB on the next generation system. But if they find out that near the end of the run they will outputting too much data, they can understand the tradeoffs (output only later timesteps, output reduced resolution/accuracy, variables, output only certain variables, and not others, output. The problem is that today and in the future, users have no idea of how to do this properly We see that many users (material science codes for example) often see I/O as taking away from FLOPS, so they place all of their processing in the code, and if they are not interested in certain objects, they do NOT write it. But as the code grows into a community code, or grows in concurrency, then re-running the code because certain quantities were not saved, is NOT sufficient.

Checkpoint restarts are critical for successful runs, but quite often simulations (e.g. S3D, XGC) use the C/R capability to store the analysis/visualization data, and this dual use of data makes it difficult for middleware and storage systems to optimize. For example, when we have NVRAM and the data is only used for C/R, then we write once, and hopefully read-never form most nodes. But for analysis, it is write once and read N times. The question is how large is N, and is it worthwhile to re-organize the data for later post processing, or optimize the layout for the storage layer where the scientist will do the post processing. This often leads us to understand that data can be indexed in a very rough-inexpensive way, so that when it is reterivied we can see that not all data needs to be read in if it is not going to contribute to the future parts of the workflow. How can we quickly index data (low cost in storage), and use some of this information for variables perhaps not saved, but can be derived, so that we can retrieve which pieces of the data needs to compute this derived quantities and then re-compute in only the areas of interest.

**Hasan’s notes from the phone call on 5/21/2015**

* Prototype middleware and storage infrastructure
* manage data objects across the dramatically changing data storage hierarchy
* mapping data objects across the storage hierarchy in an importance-driven, efficient manner
* ultimately improve the “time to knowledge” in data-intensive scientific workflows
* “areas of interest”
* data objects are appropriately mapped
* ADIOS as an object storage middleware system
* map a single variable to the various layers of the storage hierarchy
* We will create a metadata catalog
* design into new APIs [[JM1]](file:///E:\Dropbox%20(ORNL)\SDG\hasan\LOI-sssio...%20(kimmycomputer's%20conflicted%20copy%202015-05-21).docx%23_msocom_1)
* express user data intentions
* capture the provenance information along the way
* “predict” when and where to move information
* have relationships from the different variables
* We also want the ability to re-generate data
* storage system needs the ability to re-create data when necessary
* then re-running the code because certain quantities were not saved, is NOT sufficient.
  + Isn't this exactly what we are saying?
* dual use of data makes it difficult for middleware and storage systems to optimize.
* is it worthwhile to re-organize the data for later post processing, or optimize the layout for the storage layer where the scientist will do the post processing

[[JM1]](file:///E:\Dropbox%20(ORNL)\SDG\hasan\LOI-sssio...%20(kimmycomputer's%20conflicted%20copy%202015-05-21).docx%23_msoanchor_1)Is it better to say we are going to research something new or to better support user requirement of data access, which will be enabled by new APIs.

Sirraco is already doing a lot of the key points of data migration today

Leverage that work today

Vertical distribution of data not horizontal in the architecture

Tradeoff between ingesting vs indexing vs streaming (nice line)

Basic tradeoff when we manage data

Ingesting tradeoff in data management heavy scenarios  (databases)

Streaming data cheaply without access optimization

Optimizations possible but eventually has a tradeoff

Streaming, indexing, mixture of indexing and streaming

Highly reused data is indexed more

Highly connected with the metadata

Can we make this decision from the system's perspective

How the application can provide the metadata

Make decision in the system/middleware

Keep additional metadata to aid the decision

We need to think about when we move data between different layers

Not just move data

But also process it

Always process during movement (in transit)

Cleanup, turn into multi-resolution

Represent the data on different layers

Different layers different access modes

Strengths and weaknesses

Representation should change

How is the data represented in one layer will connect to data in another layer

Michael Bender - cache oblivious data structures

* Like b-trees with great streaming capabilities

Relationship between different layers

Guided by the metadata

1. How do we treat the data in different forms
2. What happens to the data
3. What kind of things can we do on demand

Scott:

Byte reproducibility is what the storage system gives you

But results from computation is only accurate to a given level

Reading back lower precision could have no impact on results

Kimmy

? - Multiple copies with different resolution or disjoint sets?

Layout the data in a way that the more time you spend accessing the data the more accurate the data is

Scott

? Can we get a time based read in place?

Lee: not really, shared system + deadline = no go

Sarp: FS can't do it but could be done at a higher level

Carlos: Do relationship between resolution/time

Nice word: Time to Resolution

Connect a resolution with a sequential read time

Sarp:

? Redundancy based on interest levels?

Scott: Trying to keep redundancy to a minimum because there is trouble with large scale storage

Lee:

Sirraco is a work in progress

Sudharsan :

Active storage?

* Out of scope
* Intransit does sound like it and we should bring this to the project

Balanced placement of io libraries

Bring scheduling/placement information to the table

Expose the storage system layout to the applications

Theme 2

1 - improve ability of C/R?

We don't really do much here

2 - SSIO reliability

No idea what to do there?

How would a lightweight or node local distributed file system could help n-1 checkpoint scenarios

On demand lightweight relaxed posix distributed file system

? Can be a distributed object store?

Yes.

Pre-proposal due

* Git repo will have it
* Scott will take first pass
* Jay second

Keep working after the pre-application

Weekly call (1 hour)

Outline for full proposal

1 person leads each section (and writing size)