# Project Narrative

## 1 Introduction

**Motivation.** Exponential increases in data size and the need to distill enormous amounts of information into usable knowledge are pushing the limits of data processing in many disciplines. A popular model to address the data avalanche is to process (transform, analyze, visualize) data closer to the source by linking processing with computation in a stream, pipeline, or network. Today such coupling or linking is performed either through storage or through memory/interconnect. In this proposal we refer to the former as *loose* coupling and the latter as *tight* coupling. Both modes have pros and cons. Loose coupling has favorable data translation and fault tolerance properties: the file is a common data representation language, and persistent storage provides data replication and a firewall against cascading errors. Despite these properties, the long latency, I/O bandwidth pressure, and the energy cost of moving data into and out of storage make loose coupling inappropriate for extreme-scale scientific data analysis, encouraging the community to turn to tight coupling instead. Tight coupling, while avoiding the I/O bottleneck and data movement energy costs, juxtaposes dissimilar components with different programming, data, performance, and fault models. A fragile system that practicing scientists often avoid, tight coupling is limited in practice to a simple direct stream custom-tailored to the endpoints. In most cases this means very specific data models for the specific producer/consumer pair with rigid communication lacking flexible flow control to buffer data rate mismatches, no consistent fault tolerance strategy, and no reuse for other combinations of endpoints.

**Objective.** The crux of the problem is the dichotomy between loose and tight coupling when actually neither is optimal for extreme-scale science. We propose loosening the grip of tight coupling, in essence decoupling tightly coupled data flows (called *Decaf* in this proposal) while keeping their favorable high performance and low power characteristics. For example, Figure 1 shows a workflow consisting of tight parallel tasks. We will expand each of those links with a miniature dataflow graph made of reusable primitives. The presence of optional short- and long-term storage in the dataflow gives us the best of both worlds: tight coupling whenever possible but loose coupling when usage patterns require persistent data. Our dataflow nodes may be heterogeneous and reside anywhere in the system; we will optimize their number, location, and organization to satisfy objectives such as data movement or time to solution. We will augment the dataflow with essential data operators—pipelining, selection, and aggregation—in order to achieve space-time data permutations within the dataflow, and we will execute the dataflow over various transport layers in current and future HPC architectures.

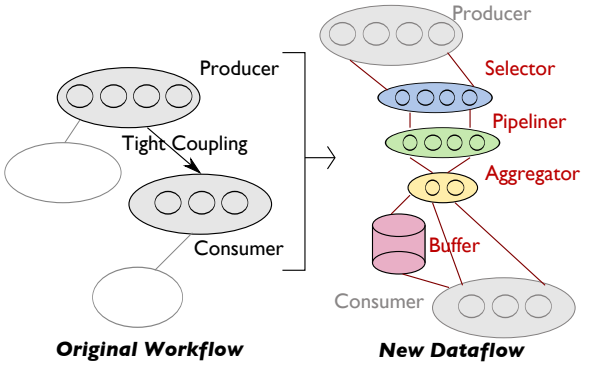


**Figure 1:** From a workflow to a dataflow, we will automatically expand links with a small set of primitives composed in an optimal way to satisfy user requirements.

**Novel contributions and deliverables.** Our proposed research addresses themes 2, 3, and 5 of the FOA: we target in situ methods and workflows, and we will evaluate our research in full and proxy applications developed in this project and by other groups in order to *improve performance, reduce power, add fault tolerance, and enhance usability*. Our proposal will result in (1) a library of dataflow primitives, akin to "BLAS for dataflow," (2) a method for automatically constructing broadly applicable dataflows from the same set of primitives, and (3) a generic and reusable solution that other workflow and coupling tools can use. We will not sacrifice performance or increase power: on the contrary, intelligently loosening rigid flows with a small set of data operators will enable connectivity of a simulation to analysis tools that would

otherwise default to expensive postprocessing. *Automatic pipelining* will support shallow copy adapters that transform data models as needed, a chunk of data at a time, rather than in its entirety, relieving pressure at all levels of memory. *Automatic aggregation* will reduce interconnect congestion. The ability to *select and permute* time-varying data in situ will reduce the need to read entire datasets back into memory for this single operation. Similarly, the ability to *handle faults* during coupling using optimal buffering of partial results, stateful operators, and criticality of data can avoid expensive entire restarts in case of errors. *Automatic buffering* allows bursty data production and consumption rates to be managed and faults to be contained. Because storage is just another producer or consumer in our design, we can augment it with the same data primitives and operate on data where they reside, further reducing postprocessing data round trips.

**Research areas and project organization.** We will integrate three levels of abstraction and software shown in Figure 2. A *high-level data description (led by Peterka)* (Section 3.1) will capture both the intrinsic data properties (sizes, types) and the extrinsic properties that depend on producer or consumer endpoints (data rates, chunk sizes). The *transport layer (led by Kimpe)* (Section 3.2) will move data, providing pipelining, and control flow with short- and long-term buffering using four coupling primitives (selector, pipeliner, aggregator, buffer). The data descriptions and coupling primitives will guide the synthesis of a *dataflow model (led by Lofstead)* (Section 3.3) that optimizes the performance of each link between producer and consumer. The data description layer will also define criticality of data that will be used to devise low-overhead *resilience strategies (led by Cappello)* (Section 3.4) combining appropriate detection and recovery techniques for fault tolerance at the two lower layers. The computing system is described by the other cross-cutting piece of our project, where we will tie into existing OS/R research to leverage capabilities such as global control backplanes and couple applications across enclaves or partitions (Section 4.2).

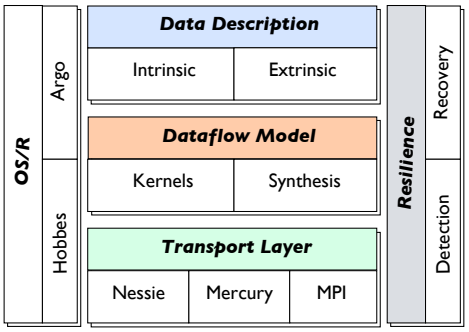


**Figure 2:** Overall project organization.

**Use cases and impact on DOE mission.** Three common use cases—data streams, data pipelines, and data networks—will drive our research. We will decouple data streaming in the in situ analysis of DOE simulations to trace particles in nuclear reactor Navier-Stokes computations. We will study data pipelining with several independent steps of in situ analysis to identify features in Ginzburg-Landau superconductivity simulations and then to track those features as they evolve over time. We will decouple dataflow networks in a workflow that transforms particles, meshes, and grids in Vlasov-Poisson N-body cosmology codes. Our proposal is designed to integrate with and contribute to four other efforts in the ASCR portfolio: Argo, Hobbes, SDAV, and CESAR. Argo and Hobbes are OS/R efforts in extreme-scale runtimes and operating systems. While Argo and Hobbes partition systems by enclaves and support information flow within and across enclaves, they do not provide any API or runtime support specifically for data processing and data coupling: our proposal addresses this gap. CESAR, one of three co-design centers, targets multiphysics coupling among other research foci. The coupling proxy app being developed in CESAR addresses accuracy and performance of simulation-simulation coupling but does not address simulation-analysis coupling, nor does it optimize coupling of complex workflows. Again, we target this gap. SDAV is the SciDAC-3 data management, analysis, and visualization effort to place SDMAV tools in the hands of computational scientists. Coupling of simulation and analysis in SDAV exists as custom and predefined point solutions for specific science use cases, but no general-purpose dataflow library is covered in SDAV or elsewhere. The solution that we propose will be able to be deployed in out-years through production-oriented funding mechanisms such as SciDAC.