A Data Sharing Infrastructure for Collaborative Science

Dan Gunter (dkgunter@lbl.gov) and Lavanya Ramkrishnan (lramakrishnan@lbl.gov), LBNL

Summary. In this short statement, we describe our vision for a data sharing infrastructure. We motivate and focus the vision with two observations: (1) Scientific research, and therefore interdisciplinary collaboration, increasingly revolves around data; (2) We are currently faced with a diverse representations for data, metadata, and tools, and freeing ourselves from this is essential for collaboration. In our vision, the software will guide scientists into natural modes of sharing, introducing minimal friction into the inevitable reconciliation of semantics across disciplines, and providing easily extensible frameworks to unify and simplify analysis tools. The two basic components we see as necessary to realize this vision are a higher common data abstraction layer, and a core set of software capabilities that leverage it. Specifically, we envision the former as a type of schema-less data store and the latter to include universal search, a scalable analysis framework, and user sandboxes.

Scientific research, and therefore interdisciplinary collaboration, increasingly revolves around data. Scientific discoveries that address important challenges facing our society require federation or integration of data from multiple research groups and disciplines. For example, the Carbon Capture Simulation Initiative (CCSI) (a partnership among national laboratories, industry and academic institutions) integrates modeling from the atomic level up to an entire power plant to achieve its goal of reducing the cost and risk associated with innovative new carbon capture solutions. The Materials Project exposes a database of *ab initio* computations of basic properties for thousands of known materials as a query-enabled store of data objects. Common data models are central to both CCSI and the Materials Project. This work is part of a much larger trend: the emergence of a data-centric *fourth paradigm* of science, distinct from the existing three paradigms of theory, experiment, and simulation. The fourth paradigm is about performing scientific research by building on, and adding to, the data that others have produced from instruments and analyses. This concept is inseparable from scientific collaboration.

We are currently faced with diverse representations for data and metadata. Today, scientific data and meta-data is primarily stored in file systems, usually in ad-hoc text formats, but occasionally in relational databases and hierarchical databases. Though file systems are a necessary software layer, they are not sufficient for user-level data management storage -- nor do they offer an adequate interface for data analysis. Relational databases are rarely used because they are inherently difficult to construct from the internal data structures. Normalized relational designs must also, in practice, be optimized for (known) access use-cases, making them inefficient for collaborative open-ended data analysis. For data containing large arrays (images, meshes, etc.), hierarchical file-databases such as HDF5 and NetCDF are often used; these have some wonderful scalability properties on parallel file systems, but have limitations for data sharing because there is minimal support for exposing data semantics or locating data across files. The end result of the current arrangement is that integration of data from different sources must be largely written from scratch for each new collaboration.

We need a better common-denominator data abstraction layer. This abstraction layer must be able easily accommodate diverse data-types and formats from diverse models and sources, while retaining a unified high-level model for locating and extracting desired subsets. A promising set of technologies to apply to this problem are schema-less, or "nosql" databases; these have emerged in recent years as an answer to global Internet companies' problems of extreme scalability and/or extreme diversity of data types. A wide variety of implementations exist, based on the ideas proposed by Amazon (Dynamo), Google (Bigtable), and others. Our preliminary work with the Materials Project has shown that they can also be a viable approach for managing complex semi-structured scientific data. Whatever the solution, our vision is that users be able to easily access this higher-level data abstraction layer from within code, CLI, or remotely, and that it will be a system service just like a global file system. Arrays, images, time-series, meshes, etc. would be transparently linked to descriptive and summary metadata so that they could be discovered by searches across datasets, runs, projects, and domains.

An orthogonal set of software components can leverage this data abstraction layer. Assuming for the moment that the data abstraction layer exists, what are the *classes* of capability that should be built on top of it so that users can seamlessly bring their own analysis codes and operate on shared data and share the results back with the community? Here, we suggest a few that we consider most important.

Search is a basic requirement -- you can't analyze something you can't find; this also includes the ability to embed search with *in situ* or streaming analyses (or vice-versa). Users should be able to rely on a common search language that can find data by arbitrary combinations of attributes. There are many existing possibilities for this language. For example, many NoSQL databases use a record-at-a-time syntax (which simplifies scalability), with explicit MapReduce programs to combine data from different "tables"; some layer SQL-like languages over this base functionality; still others like Microsoft's LINQ embed the query in the code. The choice of database technology will influence, but not necessarily determine, the search language.

An **analysis framework** allows a user to apply their analysis code to the data without re-inventing the inglorious details of keeping track of progress, pushing the results back, and keeping provenance for the whole process. One useful technology in this context is the MapReduce programming model. MapReduce, introduced by Google to handle its need to analyze terabytes of data, adapts a basic functional programming operation so that it can be scaled linearly on a distributed machine. In so doing, the MapReduce model addresses a number of requirements of exploratory collaborative data analysis including data locality, fault tolerance, and the ability to compose user-defined functions to operate on the data. Implementations are reasonably mature: the major open-source implementation, Apache Hadoop, has thousands of users and an entire ecosystem of third-party tools.

U**ser sandboxes** are necessary to both preserve and protect data, from idea to publication, in a world where data stores span many users and discovery is as easy as a Google search. In a shared-data environment, there need to be rigid controls guaranteeing that data operations, or even the side-effects of these operations, do not "leak" into other people's spaces, or indeed

into the historical space of the same user. Also, scientists must be able to control who can find and retrieve their data. To make sure that protecting data does not become an onerous task, there should be concise and declarative ways to specify which kinds of data are visible and by whom. In our vision, scientists can easily "publish" datasets containing arbitrary combinations of attributes. Access controls on data must exist throughout its entire lifecycle, and thus be included on the interfaces from the start.

Conclusion. The need for an integrated approach for sharing data and analysis tools has grown urgent, as DOE's Open-Access Science User Facilities generate more data than we can store and analyze. To enable multi-disciplinary collaborations, we need a data sharing infrastructure that addresses the challenges of diversity of data types within a *common data abstraction layer*, which in turn enables implementations of a relatively small set of orthogonal software capabilities. We believe that these capabilities should include *universal search, a scalable analysis framework, and user sandboxes.*