Data Management in Dynamic Environment-driven Computational Science

Yogesh L. Simmhan, Sangmi Lee Pallickara, Nithya N. Vijayakumar, and Beth Plale

Computer Science Department, Indiana University, Bloomington IN 47405 USA
{ysimmhan,leesangm,nvijayak,plale}@cs.indiana.edu

Abstract. Advances in numerical modeling, computational hardware, and problem solving environments have driven the growth of computational science over the past decades. Science gateways, based on service oriented architectures and scientific workflows, provide yet another step in democratizing access to advanced numerical and scientific tools, computational resource and massive data storage, and fostering collaborations. Dynamic, data-driven applications, such as those found in weather forecasting, present interesting challenges to Science Gateways, which are being addressed as part of the LEAD Cyberinfrastructure project. In this article, we discuss three important data related problems faced by such adaptive data-driven environments: managing a user's personal workspace and metadata on the Grid, tracking the provenance of scientific workflows and data products, and continuous data mining over observational weather data.

Key words: LEAD, science gateways, cyberinfrastructure, data & metadata management, provenance, data quality, data mining, streams

1 Introduction

Science has evolved over the past several decades, from an empirical and theoretical approach to one that includes simulations and modeling [4]. Additionally, scientific discoveries are increasingly propelled by large, inter-disciplinary groups working across geographical boundaries [40]. For instance, projects such as the Large Hadron Collider aim to solve grand-challenges in science through a collaboration of over 4000 scientists from 40 countries and having access to a central particle accelerator facility costing over US$2 billion [32].

Several advancements in scientific application and computer science have contributed to this evolution. Numerical techniques and algorithms have improved, allowing the real world to be modeled more accurately than ever before [16]. Weather forecasting models such as WRF, short for Weather Research and Forecasting, can now accurately predict regional mesoscale weather at resolutions of 1 Km grid spacing, with an accuracy of over 80%, 5 days in advance of

the weather, by integrating data streams across dozens of physical dimensions [20].

Similar advances in computational hardware can now be leveraged transparently through *Science Gateways* [12] that are built on top of standards such as the Common Component Architecture [3] and the *Open Grid Services Architecture (OGSA)* [10]. Science Gateways, also known as *Grids* or *Cyberinfrastructure*, have democratized access to advanced numerical tools, computational cycles, and data resources, that can be uniformly and conveniently accessed by the average scientist through online *Portal* interfaces [21].

However, environmental sciences such as mesoscale meteorology pose special challenges to these Science Gateways since they are largely triggered by events occurring in the external environment. A common requirement that exemplifies this is when a coarse-resolution regional weather forecasting simulation detects a precursor signature of a tornado in a certain region, it should spawn off another fine-resolution simulation in that specific geographical location to see if a tornado is indeed going to form. There are three key implications of such scientific applications that need to be addressed. One, scientific simulations have to be designed such that their structure is dynamic. Secondly, compute and data resources provisioned for the scientific experiments need to adapt to such external events. And lastly, there should be the ability to manage large volumes of data and associated metadata that are generated by various sensors and instruments deployed globally and from the experiments themselves.

In the subsequent section, we delve deeper into the challenges posed by the adaptive and dynamic needs of environmental sciences, and use mesoscale meteorology forecasting in the context of the *Linked Environments for Atmospheric Discovery (LEAD)* [9] project as an example to motivate the problems. In Section 3, we discuss the LEAD Cyberinfrastructure that we are building and the various enabling technologies in it. In Sections 4, 5, and 6, we will look more closely at the data management problems when dealing with terascale data, and successively look at the *myLEAD* personal metadata catalog to describe and manage user's data, the *Karma* provenance framework to track scientific data products and execution of experiments, and the *Calder* data mining tool used with streaming data. Finally, in Section 7, we summarize and present our conclusions.

2 Motivation: Mesoscale Weather Forecasting

Weather forecasting is a static process. Models ingest data generated from sensors like radars, mobile meso-nets, upper-air balloons, geostationary and polar orbiting satellites, commercial aircrafts, and surface observations, for a certain temporal and spatial range required by the forecast model. Then, analysis and assimilation of these data sources take place by performing quality control checks, extrapolating missing data points, and creating a 3D model grid of the forecast region at the given resolution. This is followed by running the actual

prediction algorithms using weather models configured by the scientists, and mining of the data to detect abnormal weather patterns. The final step generates 2D/3D images and animations of the forecast that can be disseminated to the scientists and end users. A typical regional prediction of this nature takes about 4 hours to complete, depending on the size of the region and resolution of the forecast.

The key problem with such a model is that it is completely static and the forecast is pre-scheduled to run at certain intervals. Even if a hurricane signature is detected during the data-mining part of the experiment, no action can be taken till the experiment completes, the weather researcher reviews the results, and manually configures and starts another forecast for that particular region. The LEAD project aims to take this well-oiled static computational science mesoscale meteorology forecasting process and tear it apart to be dynamic in response to the environment. There are several benefits to doing this and are supported by recent advancements in weather research.

Firstly, *regional observational models* have better forecast accuracy for a region than do continental models because the resolution of the latter has to be coarser in order to even run on today's computer systems. The solution is to selectively nest regional forecasts within a larger continental model. Secondly, *steerable radars*, notably the CASA Radars, are now being deployed. These allow the focus and collection of high-resolution data on narrow regions, instead of performing 360° swathes all the time. These dynamically steered instruments can be leveraged to increase the forecasting accuracy. And lastly, democratization of scientific resources is now possible through *community resources* such as Teragrid [6] and the availability of well established standards for accessing them. High-schools students can now get access to and learn about the same tools and resources used by weather researchers.

These advances require concomitant advances in fundamental ways in which computational science is done, before they can be leveraged to the fullest extent. These advancements include:

1. Adaptivity in computational models, allowing them to react to external events,
2. Adaptive detection and response to weather, through continuous data mining and instrument steering,
3. Adaptive use of available resources to respond to current computational and data load, and priorities of tasks, and
4. Ability for the underlying data subsystem to mine, record, track, and annotate data products in real time.

In the next section, we give an overview of the overall LEAD architecture. An appreciation of the portal interface to the system and the experiment execution tool is useful for the understanding of the remainder of the paper; so we provide that as well.

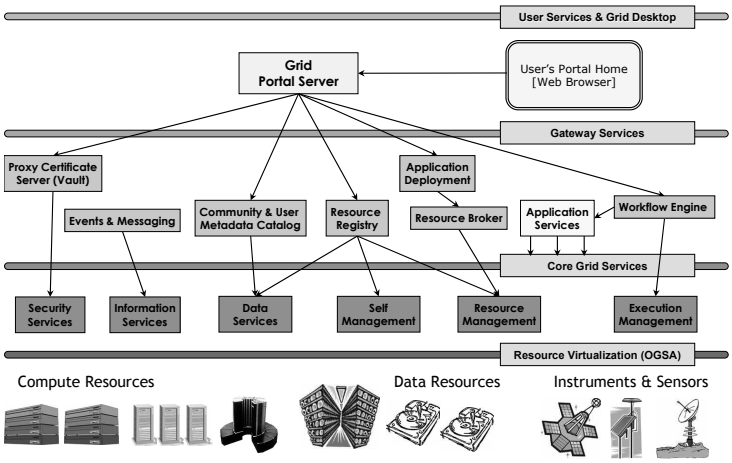

Fig. 1. The LEAD infrastructure is assembled as a set of services that interact with compute, data, and sensor resources, and accessed by a user through a portal.

3 The LEAD Cyberinfrastructure

The LEAD vision is to effect a paradigm shift in the way computation science in mesoscale meteorology is performed, brought about by a service framework for data search and model execution, for weather researchers, and students and teachers at K-12 levels and beyond. The LEAD Cyberinfrastructure builds upon a *Service Oriented Architecture (SOA)* to provide a uniform and secure interface to access resources of common interest to a distributed community of users [12]. Figure 1 illustrates this architecture. At the bottom are physical resources, such as computational clusters, mass-storage, instruments, and sensors. The service architecture virtualizes these resources so that they can be accessed using standard protocols and interfaces, without worrying about the underlying architecture or implementation. The OGSA standard [10] is commonly used as the foundation for resource virtualization in many Grid systems. These resources can be grouped as a set of core services that include security services for authentication and identity mapping, data services for moving, storing, replicating, searching, and accessing data, resource management services to schedule and monitor resources, and execution management services to plan, schedule, and manage the lifecycle of jobs run on the Grid.

On top of these core services are gateway services that provide value-added functionality and are directly exposed to the user community. These include certificate services for identity management and single sign-on capability, metadata catalogs, resource registries, notification services, workflow engines, and application services. The LEAD Portal acts as an online desktop for the users of the gateway, and provides visual interfaces to interact with the various gateway services.

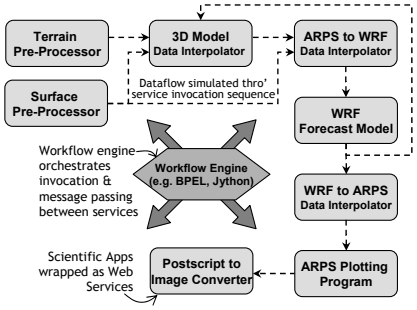

Fig. 2. Typical weather forecasting workflow in LEAD. It goes through stages of ingesting and preprocessing observational data, assimilating it into a 3D grid, running the prediction model on it, and disseminating the forecast as graphics and animations. Each box is a service and their execution is orchestrated by the workflow engine in the center.

Scientists compose experiments, such as complex data searches or model runs, as *workflows*, which consist of domain and middleware services connected as a graph. Domain applications are wrapped as web-services using the Application Factory Toolkit [19]. These application services can be graphically connected together to represent the dataflow between them using the XBaya workflow composer GUI [33], which then compiles the workflow into a Jython or BPEL [1] script that can be executed. A typical weather forecasting workflow is shown in Figure 2. The rectangular boxes represent scientific applications exposed as web-services using the Application Factory. The dotted lines signify dataflow, in the form of files consumed, transformed, and produced by the applications. A workflow engine [39] acts as a central service that orchestrates the invocation of each service in the workflow according to the execution logic.

The adaptivity requirements posited in the previous section are addressed by the LEAD Cyberinfrastructure. The workflow engine is capable of receiving notifications about external weather events that take place, and dynamically alter the execution logic for experiments. This allows for the adaptation of the computational model at runtime. Data mining applications constantly monitor data streams from various sensors looking for abnormal weather signatures. Based on the type of weather activity, these applications can configure and launch an appropriate workflow for the specific geographical region. Resource brokers, self-management services, and monitoring services detect and adapt to failures in the hardware and service substrate, and also provision available services to the required tasks at hand. Provenance services recording workflow execution help workflows resume from points of failure. Personal catalogs tracking a user's experiment assist in reconfiguring and restarting workflows, as also in providing the current status of workflows to the user. These data management tools that enable adaptivity are described in the sections below.

4 myLEAD: Personal Metadata Catalog

The *myLEAD personal workspace* comprises of a metadata catalog service and a separate back end storage manager. The notion of a separate DBMS hosted catalog for metadata is gaining a foothold in computational science through tools such as myGrid [14], MCS [38], and SRB [31], in distributed computing through Lustre [15], and even in enterprise networks through the Acopia ARX [26].

The myLEAD metadata catalog accepts data product descriptions on a wide range of data products including text, binary, images, workflow scripts, and input parameters. Data product descriptions arrive at the catalog as XML documents coded according to the LEAD Metadata Schema (LMS) [30]. A thin service layer provides atomic inserts into the catalog and back end, and performs other duties in cooperation with the workflow system [13]. Early inspiration for the metadata catalog is the Globus MCS metadata catalog [38], and it utilizes the service interfaces provided by the UK e-Science OGSA-DAI tool [2]. It is a distributed service with an instance located at each site in the LEAD Cyberinfrastructure [9]. A user's personal workspace resides at one LEAD Grid site, and is backed up to a master site. Metadata descriptions reside in the metadata catalog, as do smaller data products. Larger products are stored to a storage service, currently the Distributed Replica Service (DRS) [7].

An early estimate of usage of myLEAD is 500 users, where, at any one moment, 25 users are executing a large-scale ensemble computational model. Such ensemble workflows are capable of having up to 1,200 functional applications, and consume and produce up to 10,000 data products [28].

4.1 Data Model

The logical data model for a personal workspace consists of *projects*, *experiments*, *collections*, *logical files*, and *attributes*. Users can store one or more projects in their workspace. Under the projects, one or more experiments, which can themselves contain several collections, can be included. Logical files can belong to one or more collections, experiments, or projects. The structure of a personal workspace can vary based on the user's preference or the design of the applications that cooperate with the metadata catalog. These logical data model objects can be described by attributes associated with them. Attributes can be keywords, or simple or complex name-value pairs that are added during the creation of the objects and enhanced during future accesses to it.

The relational schema we have developed for the data model is highly generalized. Figure 3 shows the UML diagram for a slightly simplified version of the database schema. The database maintains most of the components of the data model in a single table. For instance, experiments, projects, and collections for all user spaces are stored in a single table. Logical files are kept in a separate table. The term *Attribute* is used in the general sense to mean a descriptive feature of a data object. Hereafter, we capitalize it to distinguish it from the

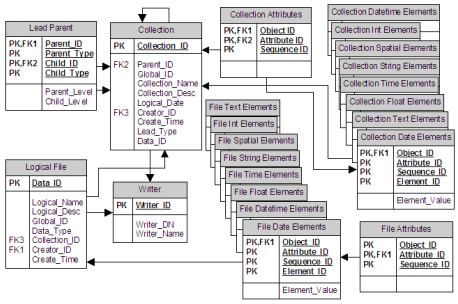

Fig. 3. Simplified relational schema of the myLEAD database. The organizational details (experiments, projects, etc.) and application attributes are coded in the data and not in the schema. This gives the catalog independence from the scientific domain.

database table of the same name. As depicted in Figure 3, an Attribute is implemented in the database schema as attribute and element tables. In the attribute table, the name and structure (i.e., data type) of a possibly complex Attribute are defined. The element table specifies a ⟨name, value⟩ pair belonging to one or more entries in the attribute table.

The attribute table defines attributes as containing one or more elements. Here, an attribute can be added on-the-fly by adding a new row to the attribute table. Although it is slightly more complicated because the attribute must be declared before an instance is created, this design decision reflects the balance we maintain between support for annotations after-the-fact and efficient querying. Additional details on the schema and database support can be found in [18].

4.2 Storing and Querying Metadata by a Hybrid Approach

In the myLEAD metadata catalog, the metadata of the data product is shredded into both Character Large Objects (CLOB) and relational tables. Due to the focus of the catalog on locating data objects that meet a specified criteria, the XML LMS document is stored using a hybrid technique employing both *inlining* and *shredding* [18]. Parts of the document received at the catalog are stored as CLOBs in the database for faster reconstruction. Key pieces of the schema are shredded (broken apart) for fast querying. This eliminates the need for achieving lossless shredding from XML since the shredded data is no longer needed to construct the XML documents returned in query responses.

4.3 Service Architecture

The myLEAD personal workspace is a distributed tool as depicted in Figure 4. At the lowest layer, there are set of distributed storage services such as the Distributed Replica Service (DRS) [7] for the personal data products and Unidata's

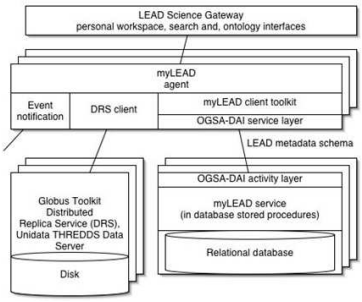

Fig. 4. Architecture of myLEAD personal workspace. The agent is a single service layer on top of the storage repository and the metadata catalog. It intercepts workflow activities by listening on event notification channels.

THREDDS Data Server (TDS) [8] for the public data products. Metadata on personal data products is managed by the myLEAD service and stored to a relational database. Much of the logic is implemented as database stored procedures. The data products themselves are either co-located with the metadata in the database (as in the case of workflow scripts), or passed to a replica manager, such as DRS. We envision providing support for TDS in the future, which provides features specific to the meteorology domain, such as sub-setting and aggregating files, and extracting fields from binary netCDF file. The server is a long-lived grid service built on top of a relational database. It is built on the OGSA-DAI service [2] interface layer.

The myLEAD agent provides client-side services for human users working interactively through the portal, and to other services in the LEAD system, particularly the workflow execution engine. The myLEAD agent responds to activities being carried out on the LEAD Cyberinfrastructure by listening on an event notification channel [17] on which status events about workflow progress are published. The agent uses the notification messages to determine the current state of a particular workflow, and actively manages the user space by, for instance, creating a new collection when a major mode transition has taken place [29]. Users interact with the tools primarily through the LEAD Cyberinfrastructure [11]. However, we are building user interactive features beyond the portal, such as to download and visualize archived experiments on their laptops.

5 Karma: Provenance Framework

Provenance [34, 5] is a form of metadata that describes the causality of an event, and the context within which to interpret it. Provenance about workflow executions is vital for scientific workflows and experiments to ensure that the exact sequence of tasks executed in the experiment is recorded [34]. This log, called the *workflow provenance*, describes the events that took place during

the course of a workflow's execution and tries to address the questions of what services were invoked, where the workflow and services ran, what their inputs and outputs were (including data products used and produced), who invoked the workflow, and so on. This type of provenance is necessary to verify and validate the experiments at a later time, and brings in accountability. It is a useful debugging tool post-execution and a monitoring tool while the experiment is running. It can also be used to track resource usage and help with scheduling future runs of the workflow.

Provenance about data generated by and used in workflows is termed *data provenance*. It attempts to answer questions about the origin of the data (in the form of workflows or services that created it), the input data that were transformed to create this data, and the usage trail for the data product. This form of provenance is necessary to discover the creator of the data, to provide insight on its quality, and may also help determine its copyright. The data usage trail also comes in handy when users of the data need to be notified of potential errors in the creation of the data.

The *Karma provenance framework* [35] collects both these forms of provenance for the scientific workflows running in the LEAD Cyberinfrastructure. The provenance model is designed around an abstract workflow model and activities defined at different parts of the workflow help collect provenance. A key application of the collected provenance is in estimating the quality of workflow derived data products in the LEAD system.

5.1 Provenance Model

The *Karma provenance framework* [35] used in the LEAD project uses an abstract model of a workflow, which it considers as a directed graph, with nodes representing services and edges representing the dataflow between them. Services are used as a layer of abstraction on top of scientific tasks to enable their use in a SOA [19]. As a workflow executes, different services are invoked in sequence by a workflow engine interpreting the workflow logic. Data products and other parameters are passed as inputs to the service, which subsequently emits the generated data products. Invoking a service consists of staging the input files to the service location, launching the scientific task that the service wraps (usually as a command-line application), monitoring the execution of the scientific application, registering the output result from the application with the myLEAD catalog, and staging the data to a central data repository. These files may be used by subsequent services invoked by that workflow or other workflows.

As can be seen, the workflow execution takes place at 3 levels: at the workflow engine level, at the service level, and at the scientific application level. The Karma provenance framework tracks provenance as *activities* that take place at different stages of the workflow. These activities in a workflow are distributed across various dimensions, one of them being the level. The activities, such as `WorkflowStarts/Finishes`, `ServiceStarts/Finishes`, and

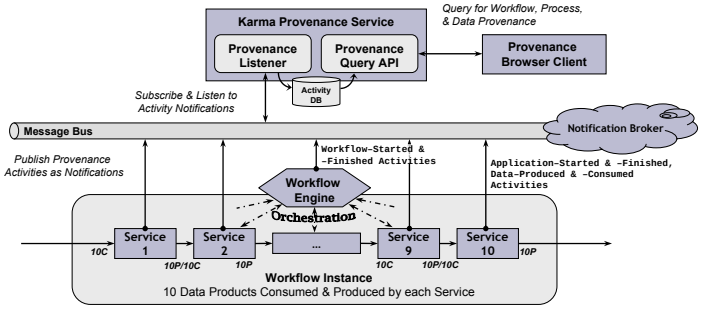

Fig. 5. The Karma Provenance Architecture. Workflows executing at the bottom publish provenance activities to a notification broker in the middle. These XML activities are subscribed to by the Karma provenance service at the top and recorded in a database. The provenance graph is reconstructed just-in-time and disseminated when queried for by clients.

`ApplicationStarts/Finishes`, take place at the boundaries between different levels. In addition, the activities contain a logical timestamp that helps order them and tracks the causality of the events. A third parameter present in the activity is the host where the event took place, which captures the distributed nature of the execution across organizational boundaries. Finally, two activities, `DataProduced/Consumed`, generated by the applications, help to track the dataflow between the applications.

Based on these activities generated by the various components of the workflow, a dynamic model of the workflow execution and the dataflow between the services can be constructed at workflow runtime. These activities are collected by the central provenance service, and used to build the workflow and data provenance graphs, and query upon them. Figure 5 shows the architecture of Karma. As the workflow executes at the bottom, its components produce activities, represented as XML notifications published to a pub-sub notification system [17]. The Karma provenance service subscribes to these activities and records them in a relational database. When a client queries the Karma service for workflow or data provenance through its web-service interface, the activities are reconstructed and composed together to form the workflow or dataflow graph, and returned to the client as an XML provenance document. Client libraries and the Application Service Toolkit [19] automate and ease the user's burden of generating these activities. Empirical performance measures have shown the overhead for collecting provenance activities to be under 1% of the application run time [36].

5.2 Provenance and Data Quality

One novel use that provenance is being applied to in LEAD is in predicting the quality of data products generated from the workflows. It is intuitive that the way a data is created has a bearing on its quality. Since provenance describes the

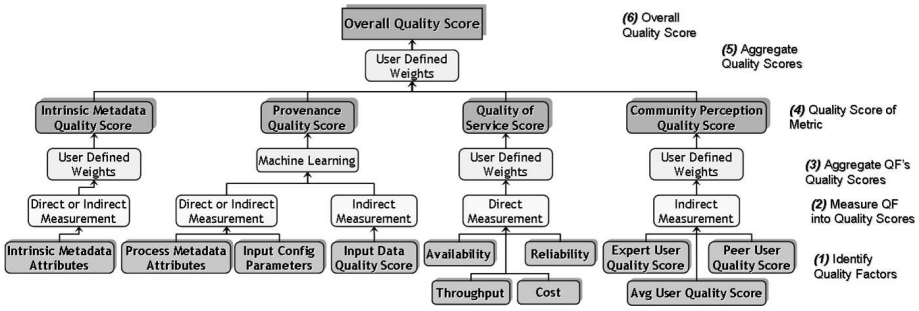

Fig. 6. Model to evaluate quality score of data product using various metrics applied to metadata attributes based on user-defined quality constraints. At the bottom are the attributes that affect the data's quality, and they are aggregated, using various metrics and user-defined constraints, to the overall quality score for the data at the top.

derivation history of data created by a workflow, it can be leveraged to estimate the data quality. Provenance forms a key attribute in the generic quality model [37] being used to quantify the quality of data products in LEAD. Such a quantification is necessary to allow comparison of data products when a user is trying to locate data for input to a workflow or for visualization and analysis. Typical search techniques for scientific data depend on the user to provide values for attributes that are then matched. These usually end up being too broad resulting in a large number of results with little to distinguish between them. Our quality model brings in not just the *intrinsic metadata* available for the data product, but also hidden (or indirect) metadata such as the *quality of service* of accessing the data product, the *community's perception* of the data, and *data provenance*.

Recognizing that quality is a very subjective matter, our quality model allows the users to define *quality constraints* on the intrinsic and hidden attributes of data products at a fine granularity [37]. The constraints are rules that define the relative importance of each attribute, which can then be evaluated for each matching data product and aggregated into a numerical *quality score* for the data product. This score then forms the basis for filtering query results and presenting only the most relevant data products in a sorted order to the user.

The quality model [37] used to evaluate the user's quality constraint is shown in Figure 6. Starting at the bottom, we have various intrinsic metadata attributes and indirect attributes available for a data product, including the provenance, represented as process metadata and input parameters. Based on the type of the attribute, different techniques are used to measured and converted them into a numerical estimate. For example, provenance is used to construct a quality model for the process deriving that data, and this model is used to generate a provenance quality score for the data. *Quality metrics* modeled as weighting functions are applied to the quality scores for attributes, guided by the user's constraints. These result in an aggregate quality score for

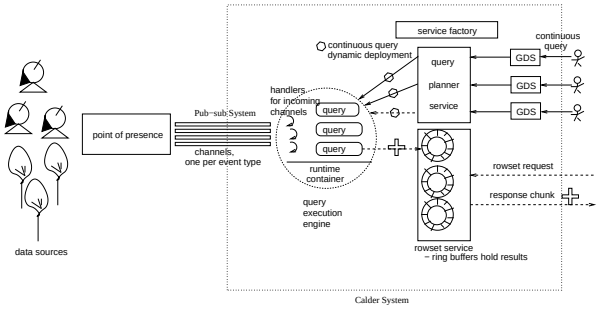

Fig. 7. Calder Architecture.

each metric, that are further combined into a single *overall quality score* for the data product. This score can then be used to rank the data product and help in the data search process.

6 Calder: Stream Mining of Environmental Data

In LEAD, the forecasting application is a special kind of data driven application that is triggered when an abnormal weather event is detected. Dynamic execution of forecast models is specified using a *rule-action* paradigm. A *rule*, defined by a user and run for a specific period of time, is a combination of filtering tasks and data mining actions triggered by the occurrence of certain events. The *action* portion of the rule-action paradigm is an invocation that kicks off a latent forecast model. Continuous data mining is performed in LEAD using the *Calder system*. A brief description of the dynamic adaptation scenario in LEAD is provided in [43].

Calder [41, 25] is a web-service that performs continuous query execution over real time streams of observational data (Figure 7). The query submission, modeled on OGSA-DAI [2], is through a data service that implements an extended realization of the GGF DAIS specification to stream systems [24]. Queries are expressed as *event-action* SQL queries deployed into the system dynamically at runtime with an associated lifetime. Calder is responsive to asynchronous stream rates and has sophisticated event scheduling mechanisms that improve the service time compared to conventional stream processing [27]. The Calder system contains a distributed query service and a stream provenance service. The distributed query planner optimizes the queries and distributes them among computational resources using a cost-efficient model [23]. The provenance service uses different models [42] to track the provenance of streams and queries in the system.

In LEAD, Calder is invoked from within a workflow as shown in Figure 8, which shows the execution of the data mining algorithm on the observational NexRad Level II data. The request to the Calder stream query service returns

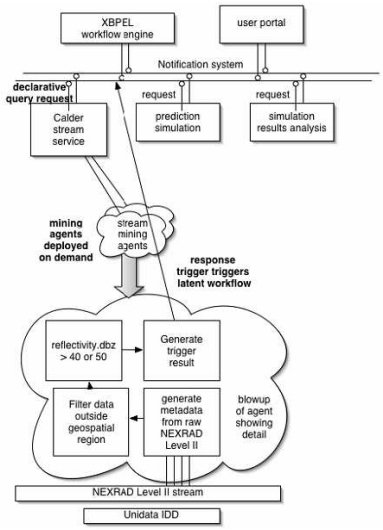

Fig. 8. Stream processing to detect vortices in Doppler radar data (below) as part of a larger workflow (above).

when the data mining results in a response trigger, such as "bad weather found". The Calder system communicates with external LEAD components using a common notification bus that transports WS-Eventing messages [17], and a separate, internal event channel for the transfer of data streams. Calder's query execution engine subscribes to channels that stream in observational data as events. These events are either described by XML metadata or arrive as *bzipped* binary data chunks.

When the query is instantiated at the computational node, it executes the filtering/data mining loop depicted at the bottom of Figure 8 for every incoming NexRad Level II Doppler radar volume scan. If the classification algorithm detects a vortex pattern whose intensity exceeds a pre-defined threshold (MDA algorithm [22]) or detects possible storm centers where the reflectivity exceeds a pre-defined threshold (SDA algorithm), a response trigger is issued to the WS-Eventing notification channel. The workflow engine subscribes to the notification channel, and acts on the message by waking up the dormant prediction simulation. We are currently expanding the continuous mining scenario in LEAD to include mining over multiple radars and aggregation of mining outputs that could be used to monitor the movement of storms over a spatial region, thereby providing more meaningful information to the forecast system.

7 Conclusions

The LEAD Cyberinfrastructure is providing essential data management tools that enable computational weather scientists to carry out investigations that

are dynamically adaptive to weather. These tools, described in this article, allow scientists to manage experimental data in a grid-embedded workspace by automatically cataloging all pieces in the e-Science experiment; precisely track the execution of the workflow and creation of the data products as provenance to help scientists verify their results at a later time and to help the community in evaluating and reusing the data through quality metrics; and in mining observational weather data in real time to automatically respond to weather events by configuring and running computational models, the results of which could potentially save lives.

LEAD has a major commitment to providing facilities to educational users. The same tools being developed for weather researchers can also be configured for students and teachers to run simple weather forecasting models through the LEAD portal as part of class assignments. Glossaries and ontological dictionaries are being developed to assist beginners in learning key concepts about weather, and techniques such as community quality perception indices used by the quality model can possibly serve as means for knowledge transfer between researchers and students.

The data management tools developed for LEAD are also being applied to other data driven computational science domains with similar dynamic and adaptive properties. Our contributions, highlighted in this article, make it easier for the scientist to locate, understand, and use their data, allowing them to advance the frontiers of science more rapidly.

8 Acknowledgments

The authors would like to thank our collaborators in the LEAD project, notably Dennis Gannon, Kelvin Droegemeier, Dan Reed, Mohan Ramamurthy, Bob Wilhelmson, Sara Graves, and Rich Clark; students who contributed to this work, Scott Jensen, Ying Liu, and Yiming Sun; and the National Science Foundation (ATM-0331480, CDA-0116050, EIA-0202048) and the Department of Energy (DE-FG02-04ER25600) whose grants enabled this work.

References

1. Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *Business Process Execution Language for Web Services Version 1.1.* BEA Systems and International Business Machines Corporation andMicrosoft Corporation and SAP AG and Siebel Systems, 2003.
2. Mario Antonioletti, Malcolm Atkinson, Rob Baxter, Andrew Borley, Neil P. Chue Hong, Brian Collins, Neil Hardman, Alastair C. Hume, Alan Knox, Mike Jackson, Amy Krause, Simon Laws, James Magowan, Norman W. Paton, Dave Pearson, Tom Sugden, Paul Watson, and Martin Westhead. The design and implementation of grid database services in ogsa-dai: Research articles. *Concurrency and Computation: Practice and Experience*, 17(2-4):357–376, 2005.

3. Rob Armstrong, Dennis Gannon, Al Geist, Katarzyna Keahey, Scott Kohn, Lois McInnes, Steve Parker, and Brent Smolinski. Toward a common component architecture for high-performance scientific computing. In *High Performance Distributed Computing Conference*, 1999.
4. Gordon Bell, Jim Gray, and Alex Szalay. Petascale computational systems. *Computer*, 39(1):110–112, 2006.
5. Rajendra Bose and James Frew. Lineage Retrieval for Scientific Data Processing: A Survey. *ACM Computing Surveys*, 37(1):1–28, 2005.
6. Charlie Catlett. *The TeraGrid: A Primer*. TeraGrid, 2002.
7. Ann Chervenak, Robert Schuler, Carl Kesselman, Scott Koranda, and Brian Moe. Wide area data replication for scientific collaborations. In *Workshop on Grid Computing*, 2005.
8. Ben Domenico, John Caron, Ethan Davis, Robb Kambic, and Stefano Nativi. Thematic real-time environmental distributed data services (thredds): Incorporating interactive analysis tools into nsdl. *Digital Information*, 2(4), 2002.
9. Kelvin K. Droegemeier, Dennis Gannon, Daniel Reed, Beth Plale, Jay Alameda, Tom Baltzer, Keith Brewster, Richard Clark, Ben Domenico, Sara Graves, Everette Joseph, Donald Murray, Rahul Ramachandran, Mohan Ramamurthy, Lavanya Ramakrishnan, John A. Rushing, Daniel Weber, Robert Wilhelmson, Anne Wilson, Ming Xue, and Sepideh Yalda. Service-oriented environments for dynamically interacting with mesoscale weather. *Computing in Science and Engineering*, 7(6):12–29, 2005.
10. Ian Foster, Hiro Kishimoto, Andreas Savva, Dave Berry, Andrew Grimshaw, Bill Horn, Fred Maciel, Frank Siebenlist, Ravi Subramaniam, Jem Treadwell, and Jeffrin Von Reich. *The Open Grid Services Architecture, Version 1.5*. Global Grid Forum, 2006.
11. Dennis Gannon, Jay Alameda, Octav Chipara, Marcus Christie, Vinayak Dukle, Liang Fang, Matthew Farellee, Geoffrey Fox, Shawn Hampton, Gopi Kandaswamy, Deepti Kodeboyina, Charlie Moad, Marlon Pierce, Beth Plale, Albert Rossi, Yogesh Simmhan, Anuraag Sarangi, Aleksander Slominski, Satoshi Shirasauna, and Thomas Thomas. Building grid portal applications from a web-service component architecture. *Proceedings of the IEEE*, 93(3):551–563, 2005.
12. Dennis Gannon, Beth Plale, Marcus Christie, Liang Fang, Yi Huang, Scott Jensen, Gopi Kandaswamy, Suresh Marru, Sangmi Lee Pallickara, Satoshi Shirasuna, Yogesh Simmhan, Aleksander Slominski, and Yiming Sun. Service oriented architectures for science gateways on grid systems. In *International Conference on Service Oriented Computing*, 2005.
13. Dennis Gannon, Beth Plale, Suresh Marru, Gopi Kandaswamy, Yogesh Simmhan, and Satoshi Shirasuna. *Workflows for eScience: Scientific Workflows for Grids*, chapter Dynamic, Adaptive Workflows for Mesoscale Meteorology. Springer-Verlag, 2006.
14. Carole Goble, Chris Wroe, Robert Stevens, and the myGrid consortium. The mygrid project: services, architecture and demonstrator. In *UK e-Science programme All Hands Meeting*, 2003.
15. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, 1991.
16. Elias N. Houstis, John R. Rice, Efstratios Gallopoulos, and Randall Bramley, editors. *Enabling Technologies for Computational Science: Frameworks, Middleware and Environments*, chapter 1, pages 7–17. Kluwer Academic, 2000.

17. Yi Huang, Alek Slominski, Chatura Herath, and Dennis Gannon. WS-Messenger: A Web Services based Messaging System for Service-Oriented Grid Computing. In *Cluster Computing and Grid Conference*, 2006.
18. Scott Jensen, Beth Plale, Sangmi Lee Pallickara, and Yiming Sun. A hybrid xml-relational grid metadata catalog. In *International Conference Workshops on Parallel Processing*, 2006.
19. Gopi Kandaswamy, Liang Fang, Yi Huang, Satoshi Shirasuna, Suresh Marru, and Dennis Gannon. Building Web Services for Scientific Grid Applications. *IBM Journal of Research and Development*, 50(2/3):249–260, 2006.
20. Richard A. Kerr. Storm-in-a-box forecasting. *Science*, 304(5673):946–468, 2004.
21. Sriram Krishnan, Randall Bramley, Dennis Gannon, Rachana Ananthakrishnan, Madhusudhan Govindaraju, Aleksander Slominski, Yogesh Simmhan, Jay Alameda, Richard Alkire, Timothy Drews, and Eric Webb. The xcat science portal. *Journal of Scientific Programming*, 10(4):303–317, 2002.
22. Xiang Li, Rahul Ramachandran, John Rushing, Sara Graves, Kevin Kelleher, S. Lakshmivarahan, Douglas Kennedy, and Jason Levit. Mining nexrad radar data: An investigative study. In *Interactive Information and Processing Systems*. American Meteorological Society, 2004.
23. Ying Liu and Beth Plale. Query optimization for distributed data streams. In *Software Engineering and Data Engineering Conference*, 2006.
24. Ying Liu, Beth Plale, and Nithya Vijayakumar. Realization of ggf dais data service interface for grid access to data streams. Technical Report 613, Indiana University, Computer Science Department, 2005.
25. Ying Liu, Nithya N. Vijayakumar, and Beth Plale. Stream processing in data-driven computational science. In *Grid Conference*, 2006.
26. Acopia Networks. File virtualization with the acopia arx. Technical report, Acopia Networks, 2005.
27. Beth Plale. Leveraging run time knowledge about event rates to improve memory utilization in wide area data stream filtering. In *High Performance Distributed Computing Conference*, 2002.
28. Beth Plale. Usage study for data storage repository in lead. Technical Report 001, LEAD, 2005.
29. Beth Plale, Dennis Gannon, Yi Huang, Gopi Kandaswamy, Sangmi Lee Pallickara, and Aleksander Slominski. Cooperating services for data-driven computational experimentation. *Computing in Science and Engineering*, 07(5):34–43, 2005.
30. Beth Plale, Rahul Ramachandran, and Steve Tanner. Data management support for adaptive analysis and prediction of the atmosphere in lead. In *Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*, 2006.
31. Arcot Rajasekar, Michael Wan, and Reagan Moore. Mysrb & srb: Components of a data grid. In *High Performance Distributed Computing Conference*, 2002.
32. Kurt Riesselmann. 600 US scientists + 3500 scientists from other countries = The New High-Energy Frontier. *Symmetry*, 2(3):18–21, 2005.
33. Satoshi Shirasuna and Dennis Gannon. Xbaya: A graphical workflow composer for the web services architecture. Technical Report 004, LEAD, 2006.
34. Yogesh Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.
35. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A Framework for Collecting Provenance in Data-Centric Scientific Workflows. In *International Conference on Web Services*, 2006.

36. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. Performance evaluation of the karma provenance framework for scientific workflows. *LNCS*, 4145, 2006.
37. Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. Towards a Quality Model for Effective Data Selection in Collaboratories. In *IEEE Workshop on Scientific Workflows and Dataflows*, 2006.
38. Gurmeet Singh, Shishir Bharathi, Ann Chervenak, Ewa Deelman, Carl Kesselman, Mary Manohar, Sonal Patil, and Laura Pearlman. A metadata catalog service for data intensive applications. In *ACM Supercomputing Conference*, 2003.
39. Alek Slominski. *Workflows for e-Science*, chapter Adapting BPEL to Scientific Workflows. Springer-Verlag, 2006. In Press.
40. Dennis E. Stevenson. Science, computational science, and computer science: at a crossroads. In *Conference on Computer Science*. ACM Press, 1993.
41. Nithya N. Vijayakumar, Ying Liu, and Beth Plale. Calder query grid service: Insights and experimental evaluation. In *Cluster Computing and Grid Conference*, 2006.
42. Nithya N. Vijayakumar and Beth Plale. Towards low overhead provenance tracking in near real-time stream filtering. *LNCS*, 4145, 2006.
43. Nithya N. Vijayakumar, Beth Plale, Rahul Ramachandran, and Xiang Li. Dynamic filtering and mining triggers in mesoscale meteorology forecasting. In *International Geoscience and Remote Sensing Symposium*, 2006.