

Realization of Dynamically Adaptive Weather Analysis and Forecasting in LEAD: Four Years Down the Road

Lavanya Ramakrishnan, Yogesh Simmhan, and Beth Plale

School of Informatics, Indiana University, Bloomington, IN 47045,
[laramakr, ysimmhan, plale]@cs.indiana.edu

Abstract. Linked Environments for Atmospheric Discovery (LEAD) is a large-scale cyberinfrastructure effort in support of mesoscale meteorology. One of the primary goals of the infrastructure is support for real-time dynamic, adaptive response to severe weather. In this paper we revisit the conception of dynamic adaptivity as appeared in our 2005 DDDAS workshop paper, and discuss changes since the original conceptualization, and lessons learned in working with a complex service oriented architecture in support of data driven science.

1 Introduction

Linked Environments for Atmospheric Discovery (LEAD)[2]¹ is a large-scale cyberinfrastructure effort in support of mesoscale meteorology. This is accomplished through middleware that facilitates adaptive utilization of distributed resources, sensors and workflows, driven by an adaptive service-oriented architecture (SOA). As an SOA, LEAD encapsulates both application and middleware functionality into services. These services include both atomic application tasks as well as resource and instrument monitoring agents that drive the workflow. The project is broad, with significant effort expended on important efforts such as education and outreach.

LEAD was conceived in the early 2000's in response to the then state-of-the-art in meteorology forecasting. Forecasts were issued on a static, cyclic schedule, independent of current weather conditions. But important technology and science factors were converging to make it possible to transform weather forecasting by making forecast initiation automatic and responsive to the weather. The grid computing community was focused on web services as a scalable, interoperable architecture paradigm [12]. Research was occurring on one-pass data-mining algorithms for mesoscale phenomena [3]. The CASA Engineering Research Center

¹ Funded by National Science Foundation under Cooperative Agreements: ATM-0331594 (OU), ATM-0331591 (CO State), ATM-0331574 (Millersville), ATM-0331480 (IU), ATM-0331579 (UAH), ATM03-31586 (Howard), ATM-0331587 (UCAR), and ATM-0331578 (UIUC).

[4] was building small, inexpensive high resolution Doppler radars. Finally, large-scale computational grids, such as TeraGrid, began to emerge as a community resource for large-scale distributed computations.

In this paper we revisit the concept of dynamic adaptivity as was presented in our DDDAS workshop paper of 2005 [1], a conceptualization that has grown and matured. We discuss the facets of the model as they exist today, and touch on lessons learned in working with a complex SOA in support of data driven science.

2 System Model

Creating a cyberinfrastructure that supports dynamic, adaptive responses to current weather conditions requires several facets of dynamism. The service framework must be able to respond to weather conditions by detecting the condition then directing and allocating resources to collect more information about the weather and generate forecasts. Events also occur as execution or run-time phenomena: problems in ingesting data, in network failure, in the resource availability, and in inadequate model progress for instance. Adaptivity is driven by several key requirements:

User-initiated Workflows. A typical mode of usage of the LEAD system is user-initiated workflow through a portal (also known as “science gateway”) where a user composes a workflow or selects a pre-composed workflow and configures the computational components, and data selection criteria. In this scenario, the system needs mechanisms to procure resources and enable workflow execution, provide recovery mechanisms from persistent and transient service failures, adapt to resource availability, and recover from resource failures during workflow execution.

Priorities of Workflows. The LEAD cyberinfrastructure simultaneously supports science research and educational use, so workflow prioritization must be supported. Consider the case of an educational LEAD workshop where resources have been reserved through out-of-band mechanisms for advanced reservation. Resource allocation needs to be based on existing load on the machines, resource availability, the user priorities and workflow load. The bounded set of resources available to the workshop might need to be proportionally shared among the workflow users. If a severe weather event were to occur during the workshop, resources might need to be reallocated and conflicting events might need some arbitration.

Dynamic Weather Events. We consider the case of dynamic weather event detection with data mining. Users have the freedom to specify dynamic mining criteria from the portal, and use the triggers from detected weather phenomena as the basis for automated forecast initiation. This freedom creates resource arbitration issues. Multiple weather events and their severity might factor into assigning priorities between users for appropriate allocation of limited available resources, for instance.

Advanced User Workflow Alternatives. An advanced user has a workflow and provides a set of constraints (e.g., a time deadline) and the work to be done. Tradeoffs may have to be made to arbitrate resources. For instance, the user might be willing to sacrifice forecast resolution to get early results which might then define the rest of the workflow.

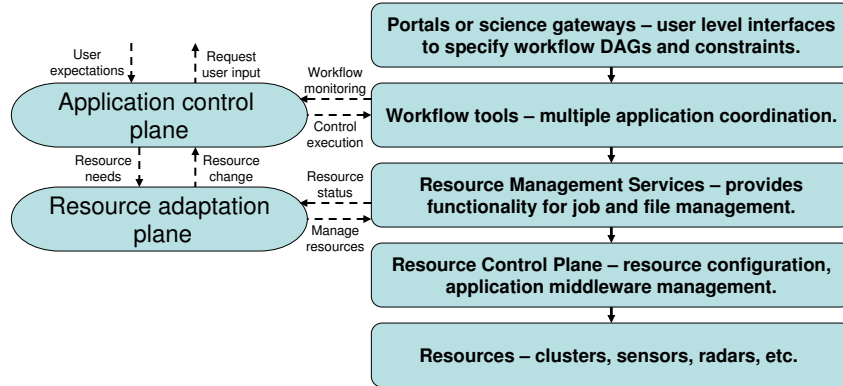


Fig. 1. Service and resource stack is controlled by application control plane interacting at workflow level; resource adaptation plane effects changes to underlying layers. Stream mining is a user-level abstraction, so executes as a node in a workflow.

The conceptualization of the system as reported in the 2005 DDDAS workshop paper [1] casts an adaptive infrastructure as an adaptation system that mirrors the forecast control flow. While a workflow executes services in the generation of a forecast, the adaptive system is busy monitoring the behavior of the system, the application, and the external environment. Through pushing events to a single software bus, the adaptive system interacts with the workflow system to enact appropriate responses to events.

The model eventually adopted is somewhat more sophisticated. As shown in Figure 1, the external-facing execution is one of users interacting through the portal to run workflows. The workflows consume resources, and access to the resources is mediated by a resource control plane [11]. The adaptive components are mostly hidden from the user. A critical component of the application control plane is monitoring workflow execution (Section 4). The resource adaptation plane manages changes in resource allocation, in consultation with the application control plane, and in response to a number of external stimuli (Section 5).

At the core of the LEAD architecture is a pair of scalable publish-subscribe event notification systems. One is a high-bandwidth event streaming bus designed to handle large amounts of distributed data traffic from instruments and other remote sources [5]. The second bus handles communication between the

service components of the system. While not as fast as a specialized bus, it does not need to be. Its role is to be the conduit for the notifications related to the response triggers and all events associated with the workflow enactment as well as the overall state of the system [7]. This event bus is based on the WS-Eventing standard endorsed by Microsoft, IBM and others and is a very simple XML message channel.

3 Dynamic Data Mining

Dynamic weather event responsiveness is achieved by means of the Calder stream processing engine (SPE) developed at Indiana University [5] to provide on-the-wire continuous query processing access, filtering, and transforming of data in data streams. Functionality includes access to a large suite of clustering data mining algorithms for detecting mesoscale weather conditions developed at the University of Alabama Huntsville [3]. The SPE model is a view of data streams as a single coherent data repository (a “stream store”) of indefinite streams, with provisioning for issuing SQL-like, continuous queries to access streams. The layer that transports stream data is a binary publish-subscribe system. Sensors and instruments are currently added to the stream network by a manual process of installing a point-of-presence in front of the instrument that converts events from the native format to the system’s XML format and pub-sub communication protocol.

As an example of the SPE in LEAD, suppose a user wishes to keep an eye on the storm front moving into Chicago later that evening. He/she logs into the portal, and configures an agent to observe NEXRAD Level II radar data streams for severe weather developing over the Chicago region. The request is in the form of a continuous query. The query executes the mining algorithm repeatedly. Data mining will result in a response trigger, such as “concentration of high reflectivity found centered at lat=x, lon=y”. The Calder service communicates with other LEAD components using the WS-Eventing notification system. It uses an internal event channel for the transfer of data streams. The query execution engine subscribes to channels that stream observational data as events that arrive as bziped binary data chunks and are broken open to extract metadata that is then stored as an XML event.

4 System Monitoring

The dynamic nature of the workflows and data products in LEAD necessitates runtime monitoring to capture the workflow execution trace including invocation of services, creation of data products, and use of computational, storage, and network resources. There are two key drivers to our monitoring: to gather a near real-time view of the system to detect and pinpoint problems, and for building an archive of process and data provenance to assist in resource usage prediction and intelligent allocation for future workflow runs. Orthogonally, there are three types of monitoring that are done: application resource usage

monitoring, application fault monitoring, and quality of service monitoring. In this section we describe the monitoring requirements and detail the use of the Karma provenance system [6] in monitoring the LEAD system.

Monitoring resource usage. Resource usage monitoring provides information on resource behavior, predicted time for workflow completion, and helps guide the creation of new “soft” resources as a side-effect of workflow execution. Taking a top-down view, resource usage starts with workflows that are launched through the workflow engine. The workflows, which behave as services consume resources from the workflow engine, itself a service. Various services that form part of the workflow represent the next level of resources used. Services launch application instances that run on computational nodes consuming compute resources. Data transfer between applications consumes network bandwidth while staging files consumes storage resources. In addition to knowledge about the available resource set present in the system that might be available through a network monitoring tool and prior resource reservations, real-time resource usage information will give an estimate of the resources available for allocation less those that might be prone to faults. In case of data products, creation of replicas as part of the workflow run makes a new resource (data replica) available to the system. Similar “soft” resources are transient services that are created for one workflow but may be reused by others. Resource usage information also allows us to extrapolate into the future behavior aiding resource allocation decisions.

Monitoring application faults. Dynamic systems have faults that take place in the applications plane and need appropriate action such as restarting the application from the last checkpoint, rerunning or redeploying applications. Faults may take place at different levels and it is possible that a service failure was related to a hardware resource failure. Hence sufficient correlation has to be present to link resources used across levels. Faults may take place in service creation because of insufficient hardware resources or permissions, during staging because of missing external data files or network failure, or during application execution due to missing software libraries.

Monitoring Quality of Service. Monitoring also aids in ensuring a minimum quality of service guarantee for applications. Workflow execution progress is tracked and used to estimate completion time. If the job cannot finish within the window stated, it may be necessary to preempt a lower priority workflow. The quality of data is determined through real-time monitoring and the maintenance of a quality model [8]. Data quality is a function of, among other attributes, the objective quality of service for accessing or transferring the data as well as the subjective quality of the data from a user’s perspective, which may be configured for specific application needs.

LEAD uses the Karma provenance system for workflow instrumentation and services to generate real-time monitoring events and for storing them for future mining. Karma defines an information model [6] built upon a process-oriented view of workflows, and a data-oriented view of product generation and consumption. Activities describe the creation and termination of workflows and services, invocation of services and their responses (or faults), data transferred, consumed

and produced by applications, and computational resources used by applications. The activities help identify the level at which the activity took place (workflow, service, application) and the time through causal ordering, along with attributes that describe specific activities. For example, the data produced activity generated by a service would describe the application that generated the data, the workflow it was part of and the stage in the workflow, the unique ID for the data along with the specific URL for that replica, and the timestamp of creation.

The activities are published as notifications using the publish-subscribe system that implements the WS-Eventing specification [7]. The Karma provenance service subscribes to all workflow related notifications and builds a global view of the workflow execution by stitching the activities together. One of the views exported by the provenance service through its querying API is the current state of a workflow in the form of an execution trace. This provides information about the various services and applications used by the workflow, the data and compute resources used, and the progress of the workflow execution. More fine-grained views can also be extracted that details the trace of a single service or application invocation. The information collected can be mined to address the needs postulated earlier. In addition, the activity notifications can also be used directly to monitor the system in real-time.

5 Adaptation for Performability

In today's grid and workflow systems, where nature of the applications or workflow is known apriori, resource management, workflow planning and adaptation techniques are based on performance characteristics of the application [9]. The LEAD workflows, in addition to having dynamic characteristics, also have very tight constraints in terms of time deadlines, etc. In these types of workflows, it is important to consider the reliability and timely availability of the underlying resources in conjunction with the performance of the workflow. Our goal is to adapt for performability, a term originally defined by J. Meyer [10]. Performability is used as a composite measure of performance and dependability, which is the measure of the system's performance in the event of failures and availability. The bottom-up performability evaluation of grid resources and the top-down user expectations, workflow constraints or needs of the application guides the adaptation in the application control plane and the resource adaptation planes. Adaptation might include procuring additional resources than originally anticipated, changing resources and/or services for scheduled workflows, reaction to failures, fault-tolerance strategies, and so on.

To meet the performability guarantees of the workflow, we propose a two-way communication in our adaptation framework, between the resource adaptation plane and the application control plane (see Figure 1). The application control plane interacts with the resource control plane to inquire about resource status and availability, select resources for workflow execution, and guide resource recruitment decisions. In turn, the application control plane needs information

from the resource layer about resource status and availability and, during execution, about failures or changes in performance and reliability.

The adaptive system has workflow planner and controller components. The workflow planner applies user constraints and choices in conjunction with resource information to develop an “online” execution and adaptation plan. The workflow controller is the global monitoring agent that controls the run-time execution. The workflow planner comes up with an annotated plan to the original user-specified DAG that is then used by the workflow controller to monitor and orchestrate the progress of the workflow. The LEAD workflows have a unique set of requirements that drive different kinds of interaction between the workflow planner and the resource control plane. As discussed in section 2, the LEAD system is used for educational workshops. In this scenario, the workflow planner might need to distribute the bounded set of available resources among the workshop participants. It is possible during the course of the execution, additional resources become available which could be used by the existing workflows. When notified of such availability the workflow planning step can reconfigure the workflows to take advantage of the additional resources. In this scenario, the workflow adaptation will be completely transparent to the end user.

The goal of the workflow controller is to control the schedule and the adaptation tasks of the workflow engine. The workflow controller can potentially receive millions of adaptation events sometimes requiring conflicting actions and hence it uses an arbitration policy. For example if a weather event occurs during an educational workshop, resources will need to be reallocated and the other workflows paused till the higher priority workflows are serviced. The workflow controller uses pre-determined policies to determine the level of adaptation and dynamism it can respond to without additional intervention. Some adaptation decisions might require external human intervention, for example, if all TeraGrid machines go down at the same time. This multi-level adaptation framework enhances existing grid middleware allowing resource and user workflows to interact to enable a flexible, adaptive, resilient environment that can change to resource variability in conjunction with changing user requirements.

6 Conclusion

The LEAD adaptation framework provides a strong foundation for exploring the effect of complex next-generation workflow characteristics, such as hierarchical workflows, uncertainties in execution path, on resource coordination. Four years into the LEAD project we are considerably closer to realizing the goal of a fully adaptive system. Architecting a system in a multidisciplinary, collaborative academic setting is benefited by the modular nature of a service-oriented architecture. The loosely coupled solutions need only minimally interact. And as the infrastructure begins to take on large numbers of external and educational users, most notably to serve the Nationally Collegiate Forecasting contest issues of reliability and long queue delays become the most immediate and pressing of issues, issues for which the adaptation framework described here is well suited.

Acknowledgements. The authors thank the remaining LEAD team, including PIs Kelvin Droegemeier (Oklahoma Univ.), Mohan Ramamurthy (Unidata), Dennis Gannon (Indiana Univ.), Sara Graves (Univ. of Alabama Huntsville), Daniel A. Reed (UNC Chapel Hill), and Bob Wilhelmson (NCSA). Author Lavanya Ramakrishnan conducted some of the work while a researcher at UNC Chapel Hill.

References

1. B. Plale, D. Gannon, D. Reed, S. Graves, K. Droegemeier, B. Wilhelmson, M. Ramamurthy. Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD. In *ICCS workshop on Dynamic Data Driven Applications and LNCS*, 3515, pp. 624-631, 2005.
2. K. K. Droegemeier, D. Gannon, D. Reed, B. Plale, J. Alameda, T. Baltzer, K. Brewster, R. Clark, B. Domenico, S. Graves, E. Joseph, D. Murray, R. Ramachandran, M. Ramamurthy, L. Ramakrishnan, J. A. Rushing, D. Weber, R. Wilhelmson, A. Wilson, M. Xue and S. Yalda. Service-Oriented Environments for Dynamically Interacting with Mesoscale Weather. *Computing in Science and Engineering*, 7(6), pp. 12-29, 2005.
3. X. Li, R. Ramachandran, J. Rushing, S. Graves, Kevin Kelleher, S. Lakshmiarahan, and Jason Levit. Mining NEXRAD Radar Data: An investigative study. In *American Meteorology Society annual meeting*, 2004.
4. K.K. Droegemeier, J. Kurose, D. McLaughlin, B. Philips, M. Preston, S. Sekelsky, J. Brotzge, V. Chandresakar. Distributed collaborative adaptive sensing for hazardous weather detection, tracking, and predicting. In *International Conference on Computational Science (ICCS)*, 2004.
5. Y. Liu, N. Vijayakumar, B. Plale. Stream Processing in Data-driven Computational Science. In *7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, 2006.
6. Y. L. Simmhan, B. Plale, and D. Gannon. A Framework for Collecting Provenance in Data-Centric Scientific Workflows. In *International Conference on Web Services (ICWS)*, 2006.
7. Y. Huang, A. Slominski, C. Herath and D. Gannon. WS-Messenger: A Web Services-based Messaging System for Service-Oriented Grid Computing. In *Cluster Computing and the Grid (CCGrid)*, 2006
8. Y. L. Simmhan, B. Plale, and D. Gannon. Towards a Quality Model for Effective Data Selection in Collaboratories. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, 2006.
9. A. Mandal, K. Kennedy, C. Koelbel, G. Marin, G. J. Mellor-Crummey, B. Liu and L. Johnsson. "Scheduling Strategies for Mapping Application Workflows onto the Grid". In *IEEE International Symposium on High Performance Distributed Computing*, 2005.
10. J. Meyer, On Evaluating the Performability of Degradable Computing Systems. In *IEEE Transactions Computers*, 1980.
11. L. Ramakrishnan, L. Grit, A. Iamnitchi, D. Irwin, A. Yumerefendi and J. Chase, "Toward a Doctrine of Containment: Grid Hosting with Adaptive Resource Control," In *ACM/IEEE SC 2006 Conference (SC'06)*, 2006.
12. I. Foster, C. Kesselman (eds), "The Grid 2: Blueprint for a New Computing Infrastructure," *Morgan Kaufmann Publishers Inc*, 2003.