Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD

Beth Plale and Dennis Gannon
Indiana University

Dan Reed
University of North Carolina, Chapel Hill

Sara Graves
University of Alabama Huntsville

Kelvin Droegemeier
Oklahoma University

Bob Wilhelmson
NCSA

Mohan Ramamurthy
UCAR, Unidata

Abstract

LEAD is a large-scale effort to build infrastructure that allows atmospheric science researchers to dynamically and adaptively respond to weather patterns to produce better-than-real time predictions of tornadoes and other "mesoscale" weather events. In this paper we discuss an architectural framework that is forming our thinking about adaptabilty and give early solutions in workflow and monitoring. [1]

1 Introduction

LEAD is a large-scale effort to build infrastructure that allows atmospheric science researchers to dynamically and adaptively respond to weather patterns to produce better-than-real time predictions of tornadoes and other "mesoscale" weather events. This is accomplished by middleware that facilitates adaptive utilization of distributed resources, sensors and workflows, driven by an adaptive event architecture. LEAD is being constructed as a service-oriented architecture. As such, component functionality is encapsulated into individual web services that have well defined interfaces. These services represent both the atomic application tasks as well as the 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.

The meteorology goal of the project is improved prediction of mesoscale weather phenomena; that is, regional scale weather phenomena such as tornadoes, severe storms and flash floods. The June 1990 outbreak that spawned 64 tornadoes across the Midwest and Hurricane Ivan that touched land at Pensacola, FL September 2004 are two examples of mesoscale weather phenomena at the larger end. Improved forecasting of these storms will enable more targeted warnings with longer advance times. More targeted warnings can reduce the cost of evacuating a coastline. During Ivan, evacuation plans were made from St. Charles parish in Louisiana as far east as Walton county in Florida, a span of 300 miles.

LEAD was funded as a large ITR by the National Science Foundation to explore state-of-the-art software artifacts and approaches in the construction of cyberinfrastructure that can advance the goals of mesoscale meteorology. Forecasts today are static, that is, they are routinely generated on a regular schedule, independent of current weather conditions. But several factors are converging to make significant advancement in forecasting possible. Technology exists to detect regional weather conditions in real time. University of Alabama, Huntsville has developed a class of algorithms that use data mining classifying techniques to detect mesoscale phenomena in data streaming from the Weather Surveillance Radar - 1988 Doppler (WSR-88D) occurring in real time [11].

A sister project to LEAD, the CASA project [9] is developing small scale Doppler radars of a size that can be mounted on cell phone towers. These small scale radars have a 30 km radius and sense at a far higher resolution and frequency than the WSR-88D Doppler radar. A full volume scan (approximately 14- $360\deg$ sweeps) of a WSR-88D radar takes 5-7 minutes. A CASA radar, on the other hand, generates a volume scan every 30 seconds. Finally, meteorologists are extending the capa-

[1]LEAD is funded by the National Science Foundation under the following Cooperative Agreements: ATM-0331594 (Oklahoma), ATM-0331591 (Colorado State), ATM-0331574 (Millersville), ATM-0331480 (Indiana), ATM-0331579 (Alabama in Huntsville), ATM03-31586 (Howard), ATM-0331587 (UCAR), and ATM-0331578 (Illinois at Urbana-Champaign). CASA is funded by the NSF under Cooperative Agreement ECE-0313747 to the University of Massachusetts at Amherst

bilities of the forecast models for faster than real time simulation of the weather. In lay terms, the models are continuously primed with current weather conditions to reduce time to generate a forecast when mesoscale weather phenomena actually do arise.

The unique challenge in LEAD is to construct a distributed service framework that is highly responsive to external events. The most important events to meteorologists are those associated with current weather conditions. The service framework must be able to respond to weather conditions by directing and allocating resources to collect more information and generate forecasts. External event arise that affect the systems ability to meet the demands of forecast time lines. These are problems in ingesting data, in network failure, in the resource availability, and in inadequate model progress for instance.

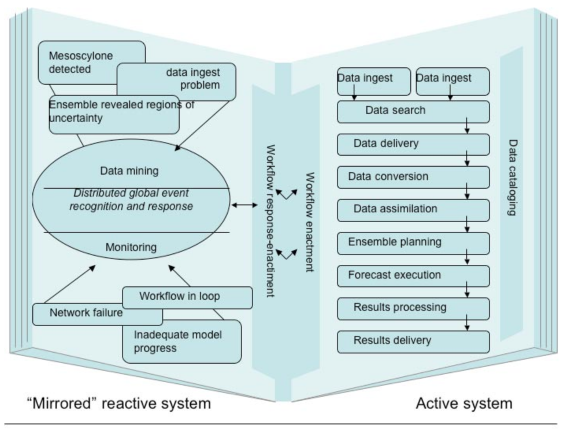


Figure 1. "Re-active" system: a mirrored, first-class system enforcing underlying re-active guarantees.

In this paper we discuss an architectural framework that is forming our thinking about adaptability and give early solutions in workflow and monitoring. We view an adaptive infrastructure as mirroring the traditional forecast flow, while largely hidden from view. This is illustrated in Figure 1. While the active system executes a very visible sequence of services in the generation of a forecast, the adaptive system is hidden, monitoring the behavior of the system, the application, and the external environment. Through generating events that can be understood in the larger context of global behavior, the active system influences the workflow system to enact appropriate changes to the system. We introduce a conceptual framework for the adaptive system based on the notion of the *software bus*, a network bus style component that integrates the adaptive pieces of the system into a interoperable whole. The actual implementation of the communication medium is still being defined.

2 LEAD architecture

LEAD is one of a handful of large scale cyberinfrastructure projects being undertaken in the world that are basing their architecture on a service-oriented model. In a service-oriented architecture, the individual distributed components are services, their interface is described by a WSDL definition [4], the service has a lifetime that can be controlled external to the service, and there is a single unifying security and communication model.

LEAD is a large system incorporating new and existing tools by numerous groups. LEAD, as is common in other large-scale projects, has legacy systems that provide key functionality for the software infrastructure. Some of these are consistent with the service-oriented architecture, some are not. The existing components fall into the following functional categories:

- forecast model – assimilation, forecast, ensemble, statistical analysis (ARPS [18], WRF [12])
- weather event detection – on-the-fly data mining techniques applied to data streams or post mortem applied to data sets. (ADAM [15])
- data dissemination – (IDD [3], Calder [14])
- data storage and cataloging – OPeNDAP [7], RLS [2]
- person experiment metadata catalog – myLEAD [13]
- monitoring – Autopilot [16]
- workflow – gBPEL [8], ogre [8]

Certain legacy pieces are being brought into the architecture by wrapping them in a service interface. We are currently wrapping pieces of the forecast model. Other components must be worked with as they exist. For instance, obtaining observational data from its sources for the LEAD testbed sites would be extremely difficult without data dissemination systems that operate on high speed networks. Fortunately IDD and IRaDS provide this. We treat these systems as existing fixtures and install a "point-of-presence" grid service in front of each. For the established OPeNDAP data server we integrate it into the architecture by sacrificing the dynamic binding benefit of a SOA. Essentially, we hard-code information about their location, behavior, and interface. On the downside, adding another instance of this kind of service cannot done easily because it requires substantial administrative overhead. These partially-integrated services also lack the ability to respond to global commands by the workflow as discussed in Section 4.

The LEAD architecture is a mix of persistent and transient services. Persistent services have and unlimited lifetime and service multiple users simultaneously. Transient services have a limited lifetime. Their lifetime is determined by the lifetime of the workflow that is carrying out a particular task. A workflow can be viewed as a global0-level thread of execution across the service-oriented architecture.

3 Forecast Example

Mesoscale meteorology forecasting is both computationally and data intense. As additional resources as the Teragrid [1] and CASA radars emerge, the opportunity to carry out more refined and responsive forecasts exists. The following example gives a feel for the kind of forecasting researchers hope to carry out in the near future.

At 0600hr a research meteorologist kicks off a 2km resolution, 12 hour forecast over the state of Oklahoma. The run is an *ensemble* run in that it consists of 20 copies of the model each with the physics tweaked slightly differently. The runs complete by 0800hr. The final output of the run is 20 files in a binary format containing temperature, wind, and microphysics. The files undergo statistical analysis for the purpose of uncovering *regions of uncertainty* corresponding to regions in the state where there are high levels of disagreement across the ensemble versions. These regions can occur when there is too little data available of the region. See Figure 3.

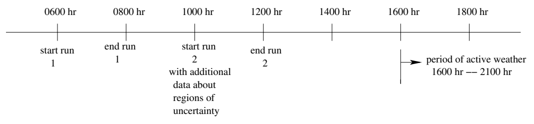

0600 hr
0800 hr
1000 hr
1200 hr
1400 hr
1600 hr
1800 hr
start run 1
end run 1
start run 2 with additional data about regions of uncertainty
end run 2
period of active weather 1600 hr — 2100 hr

Figure 2. Forecast timeline for the 20-version ensemble run.

To reduce the level of uncertainty, meteorologists want to selectively gather more information about the identified regions. An appropriate outcome of the statistical analysis, then, would be to focus the small NetRad radars in the regions of uncertainty and have them gather data over a limited time period to augment the forecast. Suppose the radars collect data from 0800hr to 1000hr. The collected data is then converted and assimilated into the 3D input data grid (third and fourth boxes in Figure 3.) Another forecast is kicked off at 1000hr. This time the forecast is a 6 hour forecast, not a 12 hour forecast. As depicted in Figure 2, run 2 finishes at 1200hr. The ensemble results are analyzed again. This time the levels of uncertainty have been reduced so as to give the meteorologist a sufficiently high degree of trust in the forecast. The cycle has completed by 1200hr, meaning a viable forecast is generated well in advance of the 1500 - 2100hr window when the majority of severe weather occurs.

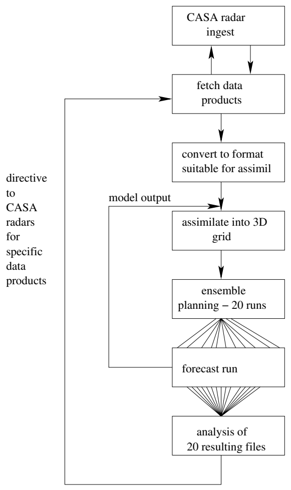

CASA radar ingest
fetch data products
convert to format suitable for assimil
model output
assimilate into 3D grid
ensemble planning – 20 runs
forecast run
analysis of 20 resulting files
directive to CASA radars for specific data products

Figure 3. Control flow for the 20-ensemble run example.

This example exposes a number of challenges in provisioning for runs like these. In the flow graph of the example given in Figure 3, for run 1, data products used in the forecast are fetched from where they reside. The ADAS data assimilation system assumes data products are located in a named directory on the local file system. A forecast might use a dozen or so different data products drawn from Doppler radar, CASA radar, satellite, buoys, balloons, etc. The converted products are assimilated into a single 3D grid in box 4. The 3D grid is passed to an ensemble planner who knows the whereabouts of the 20 ARPS models with their tweaked physics.

The forecasts are scheduled on computationally intense resources. A single 27 km resolution, 84 hour forecast over the US (a CONUS forecast), consumes 60 processors (30 nodes) of a dual processor Pentium IV Xeon 2.0 GHz cluster at Oklahoma University to run, and takes 6 hours to complete. A regional forecast running an ensemble will presumably require more resources, but due to its smaller area, the scale-up is expected to be less than a factor of 20.

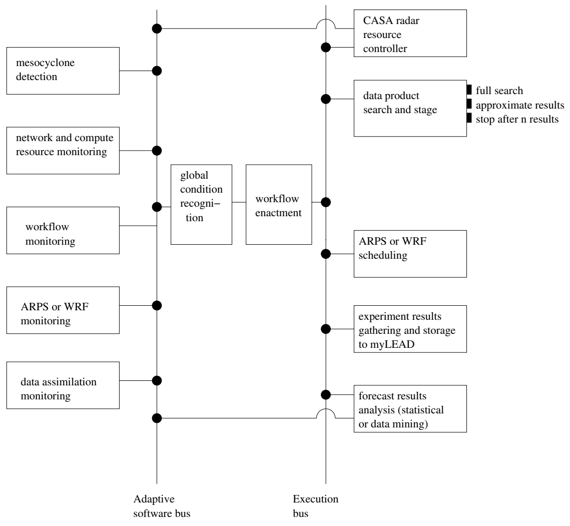
If the analysis step in the first run detects high levels of uncertainty, it will issue a directive to focus the regional radars on the areas of uncertainty. The gathered data products are delivered to a directory where they are converted and assimilated. The assimilation system is optimized to ingest model results from a prior run, reducing the assimilation time as only the new CASA data must be processed. The ensemble forecast and analysis process is repeated, this time to generate a 6 hour forecast. The process may be repeated multiple times with shorter and shorter forecast windows.

It is important to point out that subsequent loops through the graph in this example are triggered by the outcome of the analysis results of the prior iteration. More pointedly, a second loop iteration is initiated based on meteorological analysis, it is this component that is the source of a directive to the local radars in the areas where uncertainty exists.

4 Adaptive System

Adaptive handling in LEAD is driven by three broad requirements as listed below and discussed in detail in this section.

- **Numerous simultaneous forecasts over limited resources** – it is not inconceivable to have 100 or more users simultaneously running forecast models that concurrently request LEAD resources
- **Highly heterogeneous and numerous sources of adaptive events** – events that have the potential to trigger adaptive behavior occur at the environment level, application level, service level, and at the hardware level, and
- **Expecting the unexpected** – workflow can never anticipate every appropriate orchestration response.



CASA radar resource controller
mesocyclone detection
data product search and stage
full search
approximate results
stop after n results
network and compute resource monitoring
global condition recogni-tion
workflow enactment
workflow monitoring
ARPS or WRF scheduling
ARPS or WRF monitoring
experiment results gathering and storage to myLEAD
data assimilation monitoring
forecast results analysis (statistical or data mining)
Adaptive software bus
Execution bus

Figure 4. Active and adaptive software busses. The latter carries events that effect adaptation in the system.

To frame our thinking, we view the components of the traditional forecast system as sitting on a software "execution bus" where they receive commands from the workflow engine (see Figure 4.) The components are web services that are responsive to messages addressed directly to them and are also responsive to a broadcast message. A workflow script carries out the ordered execution. But a large part of the important behavior in LEAD is reactive, or adaptive. For the system to be responsive to simultaneously occurring high priority events, the system employs hidden services that detect, sense, and monitor the environment. These services sit on a separate software bus that we call the *adaptive bus*. The system requirements discussed below drive the need for a separate bus, and for a single bus that unifies the diversity of components capable of generating these urgent events. These detection services are distributed and gather specialized information close to the source. The mesocyclone detection service might sit on a radar data stream (Level II data) at one of the test bed sites, watching for suspicious behavior. An instance might also be deployed to examine the 3D assimilated data set generated by a particular experiment. Monitoring detection components can monitor network and compute resources, forecasting components, data assimilation, or the workflow itself.

Simultaneous forecasts over limited resources. Given the projected scale of LEAD, it is not inconceivable to have 100 or more users simultaneously running forecast models that concurrently request resources. Some of these models may be investigations of a serious weather phenomena. Others might be users tinkering with a model. How does the workflow engine mediate access to the distributed resources? In a data driven setting, workflow must arbitrate access to resources based on some notion of urgency and importance.

Highly heterogeneous and numerous sources of adaptive events. Events that cause adaptive behavior can occur at any level in the system: in the environment when a weather condition arises, in a forecast model analysis that results in directives to a regional radar for focused data gathering, at the service layer in response to inefficiencies in an ongoing workflow execution, and at the hardware layer in terms of computational and network loads.

To achieve optimal adaptation solutions, entities that generate adaptive events must have a mechanism to push those events to a place where they can be considered and acted upon. Similarly, with heterogeneous components generating reactive events, there must be some entity that can arbitrate amongst the competing needs.

Localized detection and reaction such as might be done by each component is insufficient because it precludes detection and response to global behavior. For instance, suppose the forecast analysis of Figure 3 identifies two areas of uncertainty over Oklahoma that it wants to resolve, directing the radars to that region for the next

2 hours. But competing interests may need higher priority. Mesocyclone detection may have detected a weather phenomena elsewhere that needs to be investigated with urgency. As another example, the compute resource monitor may note that a particular forecast run is not achieving the level of performance that it needs. But initiating a request for additional resources has to be balanced against the needs of the system. Again, MDA may have just detected a weather phenomena that needs urgent investigation. In this case, the existing resource-starved run may not get its additional resources.

This necessitates a general mechanism whereby an entity can push an event to a location where it becomes part of the reactive behavior of the system. These events are markedly different from notifications in terms of the urgency with which they must be handled. A separate bus for adaptive events enables a reactive system that supports rapid recognition of global conditions and competing conditions. It is critical information can then be used to effect future workflow enactment as quickly as possible.

Expecting the unexpected. A workflow engine can never anticipate every appropriate orchestration response, but in the absence of a specific plan, the workflow may be able to turn to high-level goals that the individual service-level components can help enforce. For instance, one high-level goal may be to *Minimize end-to-end latency*, that is, minimize the total time from when an mesoscale event occurs to when a forecast is available. *Improve accuracy of prediction* is another. If the adaptive capability of the overall system makes it easier to incorporate new data into a forecast, for instance, and that new data improves the accuracy of the prediction, then the reactive system has contributed to a more accurate forecast. Finally, a suitable goal may be to maximize *Most recent weather data*. If the forecast system requires an infusion of the most recent data obtainable, the reactive system must minimize the time to respond to this need.

Service level components can contribute to enforcing high level goals by implementing and exposing different levels of service. The data search and stage service for instance could implement three levels of response: *full search* where all data resources are searched for relevant products, *approximate results* in which case a fast search returns representative results, or *first results* where the search is conducted at the first location only. Faster searches may result in older or smaller result sets and consequently less accurate forecasts, but in the face of a more urgent need, or the need for a forecast completed as soon as possible, this level of service may be sufficient.

5 Workflow in a reactive system.

Basic workflow enactment. Traditional workflow systems operate by executing a fixed schedule of tasks. It is based on a directed acyclic graph of dependencies. When a task completes, it enables another set of task to continue start operating. Our basic system operates in this manner but with a more flexible architecture than most. A workflow is described in terms of a script described in the industry standard "Business Processing Execution Language" (BPEL) language. A workflow is nothing more than a template for execution of a set of task given a set of parameter bindings. In our case, these parameter bindings correspond to the location, protocol and content of a particular input stream or data file for the workflow template. BPEL allows us to define the workflow in terms of a sequence of well-planed tasks. However, it also allows us to define the task in response to anticipated trigger events. For example, a workflow may be programmed to understand a several possible scenarios of weather conditions. The workflow script can say, "in case we receive notification 'X' then execute scenario X. When we see notification 'Y' then execute scenario Y. When instantiated with various initializing parameter (such as which instrument and event streams it must listen to) the workflow engine begins its execution.

The life cycle of a simple LEAD BPEL workflow is simple. It starts by executing any requests to services that are available. For example, it may request a resolver service to locate the current data streams of current weather conditions. A request to a service is based on a document-request oriented mode that is unique to our approach. Rather than using the "please-do-this-while-I-wait-for-your-conclusion" remote procedure call (RPC) mechanism, we use the more modern "literal document request" model. The workflow sends a request to an application service. For example, "Please execute the WRF simulation code with these parameters". Rather than wait for a response from the WRF engine giving the results of the computation, the workflow only receives a "request acknowledged" response from the WRF engine. However, to proceed to the next task, the workflow may need to wait for the results.

Each of our application services is programmed to publish notifications about its status. This uses an implementation of the WS-Eventing web service "publish/subscribe" proposed standard. When the WRF application service completes execution (or fails in its task) it publishes a "notification" of the result. These notifications are small xml documents that say things like "simulation complete. Results are at this URL .", or "insufficient resource to complete execution." The workflow template must be programmed to respond to any of these possible scenarios.

Once the workflow is enacted, each task is a request to a service to complete a task. These services are web services that respond to the workflow engine that the task has been executed. However our BPEL engine saves the state of the workflow in a database. Though the workflow instance is itself a service, it becomes a ”virtual service” that can reside in the database until a response is received. This may take minutes or hours. Depending upon the nature of the result, the workflow can respond to different scenarios. If so designed, the workflow can be programmed to wait for months to respond to a specific notification, or series of notifications, that represent a signal that a particular configuration of weather conditions has occurred. In this case, if it knows what to do, it can respond accordingly.

Model for a truly adaptive workflow system. The scenario as described above is inadequate for addressing the adaptive needs of the system. It fails to respond to global conditions and does not have the benefit of low level events about the resource performance. For this reason, we are integrating the workflow with the monitoring framework discussed in Section 6. The monitoring system can be aware of the traffic on the instrument channels, of the status of the computational resources, and of the cost of a computation based on instrumented applications and a database that can relate patterns of requests to expected resource needs.

The most fundamental limitation of the workflow architecture is that it can only execute workflows that represent programmed responses to anticipated weather scenarios. This is ongoing work and includes considering a balance of the intelligence of the workflow with the intelligence of each component as discussed earlier.

The notification system that drives LEAD workflow is a core component of the LEAD Grid middleware. Any client application or service may listen for notifications, act on them and publish new notifications. We can think of the instrument channel as the event bus that is running in parallel with the workflow service notification bus.

6 Dynamic Detection of System-level Behavior

The monitoring infrastructure is based on Autopilot [16], a toolkit for real-time application and resource monitoring and steering. Autopilot sensor services running on the resources collect performance information at periodic intervals. At each stage the workflow engine publishes events about the progress of the workflow - e.g: Workflow Started, Task A started, Task A ended, etc. The global Control/Monitor service (named “global condition recognition” in Figure 4) acts as a sensor client receiving real-time performance information. The Control/Monitor service also subscribes to events to monitor the progress of the workflow allowing for correlation between collected performance information and the progress of the workflow. It may act as just a simple monitor writing collected data to a file or a more sophisticated scheduling/steering service orchestrating the workflow. The control service may be an actuator client steering the workflow based on the performance monitoring and adaptation or fault monitoring and recovery.

Codes such as WRF instrumented to capture dynamic performance data from hardware performance counter using SvPablo [5] can be streamed to send information through the Autopilot sensor information. The Control/Monitor service may be able to additionally use information from other sensors [17], performance modeling tools to enable dynamic steering of the workflow [10].

7 Dynamic Detection of Mesoscale Phenomena

Mesocyclones contain a velocity signature known as a Rankine Vortex [6], representing incoming and outgoing radial velocity couplets in the WSR-88D data. For consecutive gate measurements with the same range and increasing scan angle, this signature is known to have continuously increasing velocity values followed by gates with similar values, but opposite signs, that decrease in magnitude with increasing scan angle. Identifying shear segments based on this signature is the approach. As part of a larger suite of algorithms that detect mesoscale weather conditions, University of Alabama Huntsville developed the UAH Mesosyclone Detection Algorithm [11]. It differs from other implementation in that it uses fast classification techniques to isolate mesocylones in large data sets. The identification accuracy compares favorably to existing MDA algorithms, but UAH-MDA has the major benefit that the algorithm executes fast enough to efficiently and expeditiously process WSR-88D data on the fly.

8 Acknowledgments

The first author would like to thank Jerry Brotzge, Director of NetRad Operations, University of Oklahoma for extensive discussion in identifying interoperability between LEAD and CASA.

9 Conclusion

Adaptability is a key goal to LEAD success. The science goals will not be achievable without an underlying software infrastructure that manages the unexpected events arising in the system with the same attention to detail that it gives to executing a forecast. It is the synergy between the 'active' and 'adaptive' systems that will bring about the flexibility in the system that is critical to advancing meteorology research.

References

[1] TeraGrid, url = http://www.teragrid.org, year = 2005.

[2] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *IEEE Mass Storage Conference*, 2001.

[3] Mitchell S. Baltuch. Unidata's internet data distribution (IDD) system: Two years of data delivery. In *Proceedings of the Thirteenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, Anaheim California, February 1997.

[4] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web services description language (WSDL) 1.1. www.w3c.org/TR/wsdl, 2001.

[5] Luiz DeRose, Ying Zhang, and Daniel A. Reed. Svpablo: A multi-language performance analysis system. In *10th International Conference on Computer Performance Evaluation - Modeling Techniques and Tools - Performance Tools'98*, pages 352–355, 1998.

[6] R. J. Donaldson. Vortex signature recognition by a doppler radar. *Journal Applied Meteorology*, 9:661–670, 1970.

[7] James Gallagher and George Milkowski. Data transport within the distributed oceanographic data system. In *Fourth International World Wide Web Conference*, December 1995.

[8] D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Dukle, L. Fang, M. Farrellee, G. Fox, S. Hampton, G. Kandaswamy, D. Kodeboyina, C. Moad, M. Pierce, B. Plale, A. Rossi, Y. Simmhan, A. Sarangi, A. Slominski, S. Shirasuna, and T. Thomas. Building grid portal applications from a web-service component architecture. *To appear in Proceedings of the IEEE*, 2004.

[9] 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 *Computational Science - ICCS 2004: 4th International Conference*. June.

[10] K. Kennedy, M. Mazina, J. Mellor-Crummey, K. Cooper, L. Torczon, F. Berman, A. Chien, Holly Dail, O. Sievert, D. Angulo, I. Foster, D.Gannon, L. Johnsson, C. Kesselman, J. Dongarra, S. Vadhiyar, R. Wolski, R. Aydt, and D. Reed. Toward a framework for preparing and executing adaptive grid programs. In *Proceedings of the International Parallel and Distributed Processing Symposium Workshop (IPDPS NGS)*. IEEE Computer Society Press, April 2002.

[11] Xiang Li, Rahul Ramachandran, John Rushing, and Sara Graves. Mining nexrad radar data: An investigative study. In *American Meteorology Society annual meeting*, 2004.

[12] J. Michalakes, S. Chen, J. Dudhia, L. Hart, J. Klemp, J. Middlecoff, and W. Skamarock. Development of a next generation regional weather research and forecast model. In Walter Zwieflhofer and Norbert Kreitz, editors, *Developments in Teracomputing: Proceedings of the Ninth ECMWF Workshop on the Use of High Performance Computing in Meteorology*, pages 269–276. World Scientific, 2001.

[13] Beth Plale, Dennis Gannon, Jay Alameda, Bob Wilhelmson, Shawn Hampton, Al Rossi, and Kelvin Droegemeier. Active management of scientific data. In *IEEE Internet Computing special issue on Internet Access to Scientific Data*, volume 9, pages 27–34, January/February 2005.

[14] Beth Plale and Nithya Vijayakumar. Evaluation of rate-based adaptivity in joining asynchronous data streams. In *To appear ACM/IEEE 19th International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society Press, April 2005.

[15] Rahul Ramachandran, Helen Conover, Sara Graves, and Ken Keiser. Algorithm development

and mining (ADaM) system for earth science applicaitons. In *Second conference on artificial intelligence, 80th AMS meeting*, January 2000.

[16] Randy Ribler, Jeffrey Vetter, Huseyin Simitci, and Daniel Reed. Autopilot: Adaptive control of distributed applications. In *7th IEEE Symposium on High- Performance Distributed Computing*, July 1998.

[17] Rich Wolski. Dynamically forecasting network performance using the network weather service. *Journal of Cluster Computing*, 1:119–132, January 1998.

[18] M. Xue, K.K. Droegemeier, and V. Wong. Advanced regional prediction system (ARPS) - a multiscale nonhydrostatic atmospheric simulation and prediciton tool. part i. model dynamics and verification. *Meter. and Atmos. Physics*, 75:161–193.