Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD

Beth Plale[1], Dennis Gannon[1], Dan Reed[2], Sara Graves[3], Kelvin Droegemeier[4], Bob Wilhelmson[5], and Mohan Ramamurthy

[1] Indiana University
[2] University of North Carolina, Chapel Hill
[3] University of Alabama Huntsville
[4] Oklahoma University
[5] University of Illinois Urbana Champaign
[6] UCAR, Unidata

Abstract. LEAD is a large-scale effort to build a service-oriented 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 adaptability and give early solutions in workflow and monitoring. [7]

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

[7] 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

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 624–631, 2005.

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. Forecasts today are static, that is, generated on a regular schedule, independent of current weather conditions. But important technology and science factors are converging to make significant advancement in forecasting possible. Fast algorithms exists to dynamically detect mesoscale phenomena in data streaming from the Weather Surveillance Radar - 1988 Doppler (WSR-88D) [6]. In a sister project to LEAD, the CASA project [4] is developing small scale regional Doppler radars (*i.e.* NetRad 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. Finally, large-scale computational grids, such as Teragrid [1], are sufficiently mature in their support infrastructure to where harnessing large numbers of compute resources on demand for application processing is now feasible.

The unique challenge in LEAD is to construct a distributed service framework that is highly responsive to real-time events. Key events are derived from current weather conditions. Specifically, the service framework must be able to respond to weather conditions by directing and allocating resources to collect more information and generate forecasts. Events also occur as execution or runtime phenomena: problems in ingesting data, in network failure, in the resource availability, and in inadequate model progress for instance.

We view real time responsiveness as one enforced by an adaptive infrastructure that mirrors the forecast control flow, while remains largely hidden from user view. While a workflow executes services in the generation of a forecast, the adaptive system is largely 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 adaptive system interacts with the workflow system to enact appropriate responses to events. In this paper we introduce a conceptual framework for an adaptive system based on the notion of the *software bus*, a network bus style component that unifies vastly heterogeneous detection components. The actual implementation of the communication medium is still being defined. We also discuss early success in workflow and monitoring.

2 Motivation

Mesoscale meteorology forecasting is both computationally intense and data intense. The following example gives a computer science view of the kind of forecasting that meteorology 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 binary files containing

temperature, wind, and microphysics. The files undergo statistical analysis to uncover *regions of uncertainty* corresponding to regions across the state where there are high levels of disagreement across the ensemble versions. Such regions may occur when there is too little data available of the region.

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 additional data. Suppose the radars collect data from 0900hr to 1100hr. The collected data is then converted and assimilated into the 3D input data grid (third and fourth boxes in Figure 1.) Another forecast is kicked off at 1100hr. This time the forecast is a 6 hour forecast, because the forecast need only cover the late afternoon hours when the majority of severe mesoscale weather occurs. The second run finishes at 1300hr. 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.

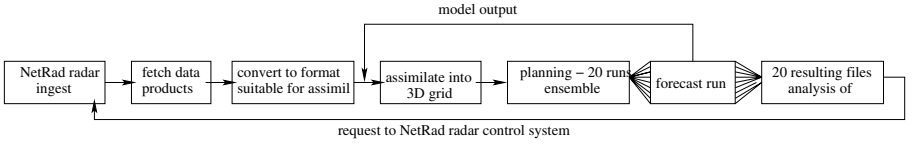

Fig. 1. Control flow for the 20-ensemble run example

This example exposes a number of challenges in provisioning for runs like these. In the flow graph of the example given in Figure 1, for run 1, data products used in the forecast are fetched from where they reside. The ADAS data assimilation system (labeled "assimilate into 3D grid") 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. The 3D grid is passed to an ensemble planner who knows the whereabouts of the 20 forecast 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.

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.

3 Adaptive System

Adaptive handling in LEAD is driven by three broad requirements as follows:

- **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,
- **Numerous simultaneous forecasts over limited resources** – 100 or more users may be running forecast models at any one time and concurrently request LEAD resources, and
- **Expecting the unexpected** – workflow can never anticipate every appropriate orchestration response.

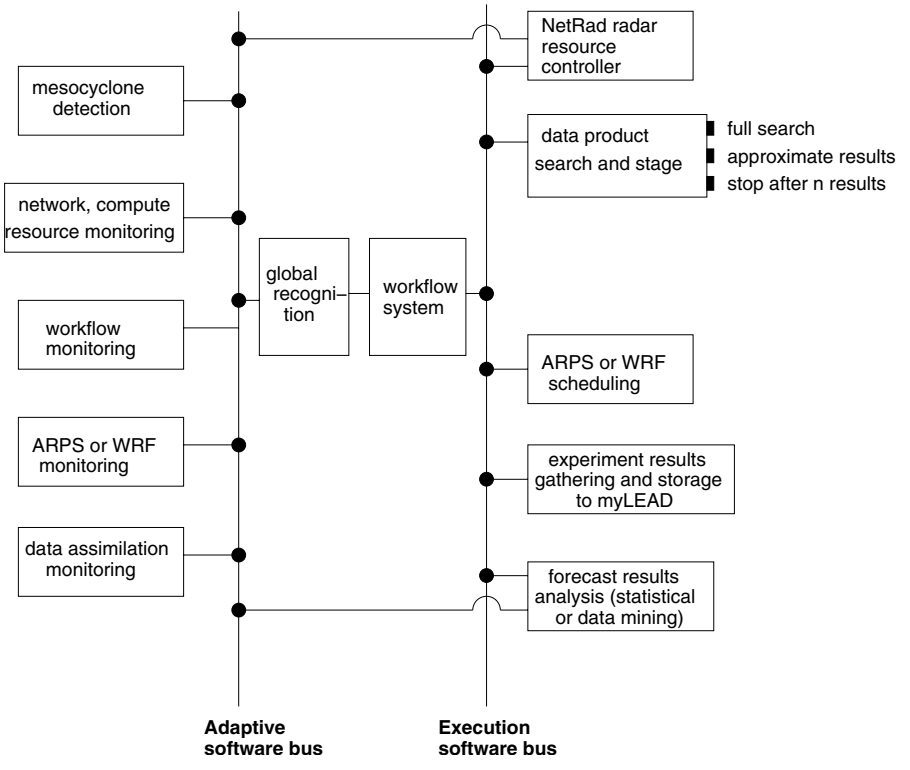

Fig. 2. Active and adaptive software busses. The latter carries events that effect adaptation in the system

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, at the service layer in response to inef-

ficiencies in an ongoing workflow execution, and at the hardware layer in terms of computational and network loads.

For the system to be responsive to simultaneously occurring high priority events, it employs services to detect, sense, and monitor the environment. These detection services are distributed and gather specialized information close to the source. The mesocyclone detection service (UAH-MDA [6]) might sit on a radar data stream (Level II data) at one of the test bed sites. 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.

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. Heterogeneous components generating reactive events, necessitate an entity that can arbitrate among the competing needs. We can think about the adaptive functionality as a separate system. This is depicted in Figure 2. Where the forecast components to the right are controlled by a workflow system, the adaptive components publish events onto an *adaptive software bus*.

Localized detection and reaction such as might be done by each active component is insufficient because it precludes detection and response to global behavior. For instance, suppose the forecast analysis of Figure 1 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 have higher priority. Mesocyclone detection may have detected a weather phenomena elsewhere that needs to be investigated with urgency. For instance, 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 compute resources must be balanced against the needs of other users and dynamically occurring events.

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 with recognition of urgency and importance.

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. These are depicted in Figure 2 as service interfaces to the service "data product search and stage."

4 Workflow in a Reactive System

Basic workflow enactment. Traditional workflow systems operate by executing a fixed schedule of tasks, based on a directed acyclic graph of dependencies. When a task completes, it enables another set of tasks to 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.

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.

Application services are programmed to publish status notifications by means 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. "

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 yet have the benefit of low level events about the resource performance. 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.

5 Dynamic Detection of System-Level Behavior

The monitoring infrastructure is based on Autopilot [7], 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 2) 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 [2] 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 [8], performance modeling tools to enable dynamic steering of the workflow [5].

6 Dynamic Detection of Mesoscale Phenomena

As part of a larger suite of algorithms to detect mesoscale weather conditions, University of Alabama Huntsville developed the UAH Mesocyclone Detection Algorithm [6] referred to earlier. Mesocyclones contain a velocity signature known as a Rankine Vortex [3], representing incoming and outgoing radial velocity couplets in the WSR-88D data. UAH-MDA identifies shear segments based on this

signature and applies fast classification techniques to isolate mesocyclones in large data sets. The identification accuracy compares favorably to existing MDA algorithms.

7 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.

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.

References

1. Teragrid. http://www.teragrid.org, 2005.
2. 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.
3. R. J. Donaldson. Vortex signature recognition by a doppler radar. *Journal Applied Meteorology*, 9:661–670, 1970.
4. 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.
5. 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.
6. Xiang Li, Rahul Ramachandran, John Rushing, and Sara Graves. Mining nexrad radar data: An investigative study. In *American Meteorology Society annual meeting*, 2004.
7. 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.
8. Rich Wolski. Dynamically forecasting network performance using the network weather service. *Journal of Cluster Computing*, 1:119–132, January 1998.