Enabling End-to-End Data-Driven Sensor-Based Scientific and Engineering Applications [⋆]

Nanyan Jiang and Manish Parashar

Center for Autonomic Computing (CAC) &
The Applied Software Systems Laboratory (TASSL)
Department of Electrical and Computer Engineering
Rutgers University, Piscataway NJ 08855, USA
{nanyanj,parashar}@rutgers.edu

Abstract. Technical advances are leading to a pervasive computational infrastructure that integrates computational processes with embedded sensors and actuators, and giving rise to a new paradigm for monitoring, understanding, and managing natural and engineered systems – one that is information/data-driven. However, developing and deploying these applications remains a challenge, primarily due to the lack of programming and runtime support. This paper addresses these challenges and presents a programming system for end-to-end sensor/actuator-based scientific and engineering applications. Specifically, the programming system provides semantically meaningful abstractions and runtime mechanisms for integrating sensor systems with computational models for scientific processes, and for in-network data processing such as aggregation, adaptive interpolation and assimilations. The overall architecture of the programming system and the design of its key components, as well as its prototype implementation are described. An end-to-end dynamic data-driven oil reservoir application that combines reservoir simulation models with sensors/actuators in an instrumented oilfield is used as a case study to demonstrate the operation of the programming system, as well as to experimentally demonstrate its effectiveness and performance.

1 Introduction

Technical advances are rapidly leading to a revolution in the type and level of instrumentation of natural and engineered systems, and are resulting in a pervasive computation ecosystem that integrates computers, networks, data archives, instruments, observatories, experiments, and embedded sensors and actuators. This in turn is enabling a new paradigm for monitoring, understanding, and managing natural and engineered systems – one that is information/data-driven and

[⋆] The research presented in this paper is supported in part by National Science Foundation via grants numbers CNS 0723594, IIP 0758566, IIP 0733988, CNS 0305495, CNS 0426354, IIS 0430826 and ANI 0335244, and by Department of Energy via the grant number DE-FG02-06ER54857, and was conducted as part of the NSF Center for Autonomic Computing at Rutgers University.

that symbiotically and opportunistically combines computations, experiments, observations, and real-time information to model, manage, control, adapt, and optimize.

Several application domains, such as waste management [1], underwater ocean phenomenon monitoring [2], city-wide structural monitoring [3] and end-to-end soil monitoring system [4], are already experiencing this revolution in instrumentation, and can potentially allow new quantitative synthesis and hypothesis testing in near real time as data streams in from distributed instruments. However, these application present many new and challenging requirements due to (1) the data volume and rates, (2) the uncertainty in this data and the need to characterize and manage this uncertainty and (3) the need to assimilate and transport required data (often from remote sites over low bandwidth wide area networks) in near real-time so that it can be effectively integrated with (running) computational models and analysis systems. A key challenge is the lack of effective programming and system software supports that can support these applications in an end-to-end manner. As a result, in most existing instrumented systems data acquisition is a separate offline process and this data is typically used in a post-processing manner [5].

The overall goal of the research effort presented in this paper is to investigate sensor system middleware and programming support that will enable distributed networks of sensors to function, not only as passive measurement devices, but as intelligent data processing instruments, capable of data quality assurance, statistical synthesis and hypotheses testing as they stream data from the physical environment to the computational world [5]. Further, application should be able to interact with the sensor system to control sensing and data processing behaviors. The programming systems enables sensor-driven applications at two levels. First, it provides programming abstractions for integrating sensor systems with computational models for scientific processes (e.g. biophysical, geophysical processes) and with other application components in an end-to-end experiment. Second, it supports programming models and systems for developing in-network data processing mechanisms. The former supports complex querying of the sensor system, while the latter enables development of in-network data processing mechanisms such as aggregation, adaptive interpolation and assimilations, both via semantically meaningful abstractions. The research is driven by the management and control of subsurface geosystems, such as managing subsurface contaminants at the Ruby Gulch waste repository [1] and management and optimization of oil reservoirs [6]. Crosscutting requirements of these applications include multi-scale, multi-resolution data access, data quality and uncertainty estimation, and predictable temporal response to varying application characteristics.

The focus of this paper is on the end-to-end abstractions provided by the programming system, and on how they can be used to enable scientific/engineering applications to discover, query, interact with, and control instrumented physical systems in a semantically meaningful way. Specifically, this papers describes the design and operation of the *GridMap* abstraction, and demonstrates its usage and effectiveness using an *Instrumented Oilfield* application as a case study. A prototype programming system has been implemented. An experimental evaluation using this prototype is also presented.

The rest of the paper is organized as follows. Section 2 gives an overview of the overall programming system and the *GridMap/iZone* abstractions it provides. Section 3 presents instrumented oilfield application case study. Section 4 presents an experimental evaluation of the programming system and the abstraction provided. Section 5 discusses related work. Section 6 concludes the paper and outlines current and future work.

2 The *GridMap* & *iZone* Programming Abstractions

Scientific applications often require measurements at pre-defined grid points, which are often different from the locations of the raw data provided directly by the sensor network. As a result, the sensor-driven scientific and engineering applications require a virtual layer, where the logical representation of the state of the environment provided to the applications may be different from the physical representation of raw measurements from sensor network. The abstractions described in this section enable applications to specify such a virtual layer and the models (e.g. regression models, interpolation functions, etc.) that should be used to estimate data on the virtual layer from sensor readings, as well to develop in-network implementations of the data estimation mechanisms.

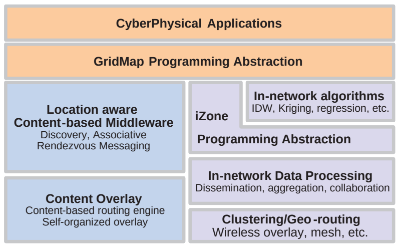
CyberPhysical Applications
GridMap Programming Abstraction
Location aware Content-based Middleware
Discovery, Associative Rendezvous Messaging
iZone
In-network algorithms
IDW, Kriging, regression, etc.
Programming Abstraction
In-network Data Processing
Dissemination, aggregation, collaboration
Content Overlay
Content-based routing engine
Self-organized overlay
Clustering/Geo-routing
Wireless overlay, mesh, etc.

Fig. 1. A schematic overview of the programming system.

A schematic overview of the overall programming system architecture is presented in Fig. 1. It consists of a two-level programming abstraction, the end-to-end *GridMap* programming abstraction that enables computational applications to access and integrate sensor data into their models, and the in-network *iZone* programming abstraction to enable the development of scalable in-network data processing mechanisms. The underlying middleware provides an in-network data processing engine, which supports efficient data dissemination, aggregation and collaboration in dynamics, resource-constraint heterogeneous sensor networks.

The *GridMap* Abstraction: The *GridMap* abstraction consists of two operators. The first operator allows the application to construct a virtual grid (a *GridMap*), corresponding to the computational grid used by the computational models, on the instrumented domain. Once this virtual grid has been overlayed on the sensor system, the application can use the second operator to query data corresponding to a region of interest on this virtual grid. The interface of this second operator includes a specification of the method (e.g., interpolator) that should be used to estimate data at a grid point in the region of interest using physical data from sensors that are in the neighborhood of the point. The operator also includes parameters such as the size of neighborhood that should be used in the estimation, and what are the constraints on the accuracy and cost of the estimation.

The *GridMap* operators include end-to-end query operations, i.e., *query*, *notify*, *retrieve* as well as operators to construct, modify and delete the *GridMap* , i.e., *init*, *delete*, *refine* and *coarsen*. The parameters of the *query* operator include a specification of the region of interest within the *GridMap* and the interpolation function that should be used to compute values at the grid points of interest form the sensor values. The execution of this operator results in a query message being routed to relevant nodes (i.e., cluster heads) in the sensor network. The query specification is then matched against existing profiles, and if required, appropriate in-network operators are invoked. The *notify* operator is used to register notification requests, for example, and application may be interested in be notified if the maximum sensor reading in a region of the *GridMap* exceeds a certain threshold. The application can retrieve previously queried values using the *retrieve* operator. The *init* operator is used to initialize the grid points associated with *GridMap*. The *refine* operator modifies an existing *GridMap* by adding more grid points to increase the resolution of the *GridMap*. The *coarsen* operator modifies an existing *GridMap* by suspending some of the virtual grid points to effectively reduce the resolution of the *GridMap*. Note that, both *refine* and *coarsen* operators do not re-construct of the entire *GridMap* which makes them more efficient in dynamically changing physical environments.

The *iZone* Abstraction: The *iZone* abstraction in turn, enables the implementation of the estimation functions. The *iZone* itself is a representation of the neighborhood that is used to compute a grid point, and may be specified using a range of coordinates, a function, etc. The *iZone* abstraction also provides operators, such as *discover*, *expand*, and *shrink* for obtaining sensors corresponding to the region of interest as well as for defining in-network processing operators, such as *get*, *put*, and *aggregate*, to compute a desired grid point from sensor values from this region as summarized in [7].

Note that, with *GridMap/iZone* programming system, user-defined function can be implemented with *iZone* operators, which can then be applied in a straightforward fashion as a function operator on the actual running environment with *GridMap* operators.

3 An End-to-End Oil Reservoir Application Using *GridMap/iZone* Abstractions

In this section, we demonstrate how the *GridMap*/*iZone* abstractions can be used to implement an end-to-end oil reservoir application that combines reservoir simulation models with sensors/actuators in an instrumented oilfield.

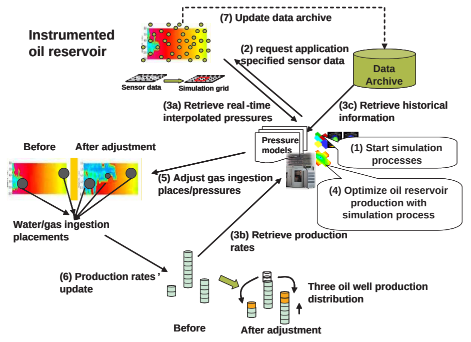
(7) Update data archive
Instrumented oil reservoir
(2) request application specified sensor data
Data Archive
Sensor data
Simulation grid
(3a) Retrieve real-time interpolated pressures
(3c) Retrieve historical information
Before
After adjustment
Pressure models
(1) Start simulation processes
(5) Adjust gas ingestion places/pressures
(4) Optimize oil reservoir production with simulation process
Water/gas ingestion placements
(3b) Retrieve production rates
(6) Production rates update
Three oil well production distribution
Before
After adjustment

Fig. 2. Overview of an end-to-end oil reservoir application.

Subsurface behavioral surveillance and sensing is becoming available in an increasing number of environmental and energy reservoir applications. The deployment of sensors is offering unlimited possibilities to monitor and obtain a dynamic understanding of the different processes taking place at different spatial and temporal scales. The proposed programming abstractions and systems software solutions can enable the end-to-end management process for detecting and tracking reservoir changes, assimilating and inverting data for determining reservoir properties, and providing feedback to enhance temporal and spatial resolutions and track other specific processes in the subsurface, so as to ensure new optimal operation (in terms of profitability, safety or environmental impact).

An overview of this application scenario is shown in Fig. 2. The figure illustrates the steps involved in constructing a closed control loop for optimal reservoir management, including computational processes for issuing queries to instrumented oil reservoir, retrieving the relevant data and integrating it with the simulation processes, and making appropriate decisions for updating oil production policy.

The evolution of pressure and concentration in the oil field during production are simulated using comprehensive mathematical models of the subsurface. As the simulations evolve, these models periodically update pressure and concentration distributions in particular regions (e.g., regions requiring adaptive refinement due to high errors) from the oil field. Sensors deployed in the oil field monitor and retrieve current pressures and/or concentrations in regions of interest. The results of the simulations are then used by the production optimization process to generate optimal configuration (i.e. production rate, gas, water pressures, etc.). The realization of these steps using the *GridMap/iZone* abstractions are briefly described below.

The application first uses the *GridMap* "init" operator to construct a virtual sensor grid overlayed on the oilfield, to match the grid used by the simulations models. For example, a *GridMap* instance constructed using the specification $<x_{lb}{:}\delta_x{:}x_{ub},\ y_{lb}{:}\delta_y{:}y_{ub}>$ represents a two dimensional rectangular virtual grid with its left bottom corner at (x_{lb}, y_{lb}) and top right corner at (x_{ub}, y_{ub}), and with a spacing of δ_x and δ_y along the x and y dimension respectively. The application can also specify the interpolation method, for example, "closest neighbor", "IDW" or "Kriging", to be used to compute data values at the virtual grid points using sensor values.

The application can now "query" the sensor system using regions of the virtual grid. The application query specifies the data types of interest (e.g., pressure, concentration), error thresholds, etc., using the syntax described in Section 2. The queries are forwarded by the runtime system to appropriate sensors nodes. For each virtual grid point, *iZones* are constructed by discovering all relevant sensors. The data retrieved from these sensors is then interpolated onto the required virtual grid points using the specified interpolation method and the in-networks mechanisms provide by an *iZone*. The resulting data values on the virtual grid are then returned to the application.

The application can now continue the simulations using the updated data, and along with current production rate, historical data, etc., to, for example, predict changes in oil reservoir, evaluate different configurations, and optimize desired objective functions such as maximizing production rate and/or minimizing cost (e.g. gas ingestion cost).

4 Implementation and Experimental Evaluation

4.1 Implementation Overview

The current prototype implementation of *GridMap/iZone* programming system consists of two key parts. The sensor network component is implemented using the 802.11 protocol and standard location based clustering to construct a two level self-organizing overlay of sensor. This component implements the mechanisms for sensor discovery, query dissemination, data gathering and aggregation and in-network data processing. It has been prototyped using sensors emulated on the Orbit wireless testbed [8]. The wide-area component is built using Java and on top of the JXTA peer-to-peer substrate [9] and deployed on the Rutgers campus Grid. This component integrates computations processes (i.e. simulations), data archives and user subsystem to the sensor system through gateway nodes. Queries issued by the computational process are routed to the appropriate sensor nodes (and aggregated and interpolated data values routed) via the gateway and cluster heads. The rest of this section focuses on an experimental evaluation using end-to-end application scenarios.

4.2 Experimental Evaluation

The parameter estimation and oil reservoir optimization scenarios in an instrumented oilfield (described in Section 3) is implemented using the prototype of the *GridMap/iZone* programming system described above and is used for the experiments presented in this section. The objectives of the experiments are not only to demonstrate the ability of *GridMap/iZone* to support the integration of computational processes with run-time in-network processing of sensor data, but also to evaluate the effectiveness and efficiency of the prototype implementation for realistic scenarios. The scenarios in the experiments are driven by a real-world sensor dataset obtained from an instrumented oil field with 2000 sensors, and consisting of pressure measurements sampled 96 time per day. A two-tiered sensor overlay with 40 clusters was emulated on 40 nodes of the Orbit wireless testbed so that each cluster ran on a single node of the testbed. The sensors are assumed to be randomly distributed in the sensors field. The computational processes ran on compute cluster located at a different campus at Rutgers.

The end-to-end cost of an interaction between a computational process and the sensor system consists of 5 parts: (1) the cost of issuing the *GridMap* query by a computational process, and routing it through the network to the sensor network gateway; (2) the cost of forwarding the query from the gateway to all relevant sensors associated with the *GridMap*; (3) cost of in-sensor-network computations (e.g., interpolation) associated with a query; (4) the cost of aggregating the data and forwarding it to the gateway; and (5) the cost returning the results back to the computational process. The experimental evaluation in this paper focuses on the in-sensor-network costs, i.e., (2), (3) and (4). These costs are measured in terms of the number of messages, the number of hops per message and the volume of data transferred, and the impact of overall size of the *GridMap*. The tradeoffs between the accuracy of data estimation and associated costs are also evaluated.

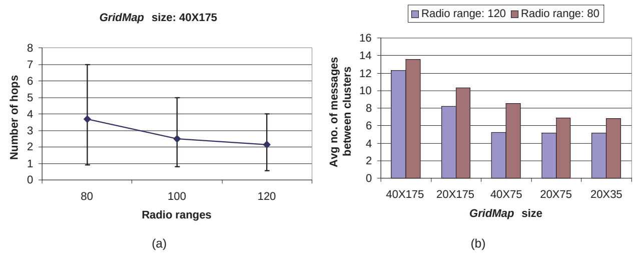

Fig. 3. Communication costs of querying all the sensors associated with a *GridMap*.

Communication Costs: This experiment measures the communication costs, in terms of average number of hops, of querying all the sensors associated with a *GridMap*. The first set of experiments keep the size of the *GridMap* fixed at 40ft x 175ft, and varies the radio range of the sensors. The plots in Fig. 3(a) show that, as the radio range increases, the average number of hops decrease. The next set of experiments measured the impact of increasing the size of the *GridMap* on the average number of messages between clusters in the sensor network for two different radio ranges. The plots in Fig. 3(b) shows that, as the size of the *GridMap* increases, so does the average number of messages required per update. For example, when the size of *GridMap* is doubled,e.g., from 20ft x 175ft to 40ft x 175ft in the experiment, the average number of messages increase by about 1.5 times. Additionally, as the size of the *GridMap* becomes smaller (e.g. 20ft x 75ft and 20ft x 35ft), the number of messages does not change much, partially because almost all of the involved clusters of the given *GridMap* are within a single hop of each other.

Tradeoff between Accuracy and Communication Costs: In this experiment, we examine the tradeoff between accuracy and communication costs for cases where the data processing (i.e., interpolation) is done within the sensors system and external to the sensor system (for example, at a remote server). The latter case represents the conventional approach where raw data is collected from the sensor network and transferred to a server where required processing is performed offline. In this case the costs measured in terms of the data volume are transferred to the gateway. The accuracy of interpolation is determined in terms of the interpolation error and depends on the interpolation mechanism used. To ensure a fair comparison, the same data processing is used in both cases.

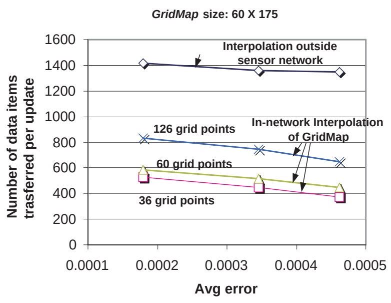

Fig. 4. Tradeoff between accuracy and communication costs.

As seen in Fig. 4, increasing the required accuracy increase the volume of communication. For in-network interpolation, this is because more data is required for each *iZone*. Furthermore, as the size of the *GridMap* increases, the volume of communication also increases. Note that the communication cost does not grow linearly with the number of grid points over the same *GridMap*. This is partially because part of the data is shared between neighboring *iZones*, therefore eliminating the need for some of the communications when computing the grid points.

For the case where the interpolation is performed outside the sensor network, the data volume does not change significantly with increasing accuracy since only a small additional amount of data is needed. A key observation is that for the same level of accuracy, the communication volume is significantly large when the interpolation is performed outside the sensor network, which also implies increased bandwidth consumption, latencies as well as energy costs. This demonstrate a key benefit of in-network data processing, specially in the case of application requiring near real-time analysis of and reaction to sensed data.

5 Related Work

There has been a significant body of research focused on programming support for sensor networks. Several systems [10, 11] provide abstractions for specifying the local behaviors of sensors, e.g., state programming [10], and CLB [12] and abstract region [11]. In the *macroprogramming* approach, global behaviors are specified and the programming system generates the local behaviors and necessary interactions, e.g., Kairos [13] and ATaG [14]. *Database-oriented* approaches provide abstractions that view the sensor network as a virtual database system and provide SQL-like interfaces for querying the networks, e.g., TinyDB [15]. The related *data streaming* approach, supported by TelegraphCQ [16], views data as information data streams, and applications monitor and react to them as they pass through the network. Other related systems include the soil ecology monitoring system [4] and the ring buffer network bus (RBNB) DataTurbine [17], which are data collection or management prototypes that uses wireless sensor systems as the component of an end-to-end system.

The programming systems discussed above have to be extended to support end-to-end sensor-driven applications and the interactions between computational models and the sensor system, and address requirements discussed in Section 1. Ideally, a scientist should only have to specify application data requirements using high-level abstractions, and the system should transform these requirements into appropriate operations, interactions and coordination within the sensor system to transform the sensed data so as to match these requirements. The *GridMap/iZone* is such a programming system to provide semantically meaningful abstractions and runtime mechanisms for integrating sensor systems with computational models for scientific processes by essentially virtualizing the sensor field to match its representation used by the computational model. The in-network data processing supports aggregation, adaptive interpolation and assimilations.

6 Conclusion

This paper presented a programming system for end-to-end dynamic data-driven sensor/actuator-based scientific and engineering applications. Specifically, the programming system provides the end-to-end *GridMap* abstraction that enables computational applications to access and integrate sensor data into their models, and the in-network *iZone* abstraction to enable the development of scalable in-network data processing mechanisms. The paper described the overall architecture of the programming system and the design of its key components, as well as its prototype implementation. An end-to-end oil reservoir application that combines reservoir simulation models with sensors/actuators in an instrumented oilfield was used as a case study to demonstrate the operation of the programming system, as well as to experimentally demonstrate its effectiveness and performance. We are currently working on using the programming system to enable other sensor-driven applications in other domains.

References

1. Parashar, M., Matossian, V., Klie, H., Thomas, S.G., Wheeler, M.F., Kurc, T., Saltz, J., Versteeg, R.: Towards dynamic data-driven management of the ruby gulch waste repository. Proceedings of the Workshop on DDDAS, International Conference on Computational Science (ICCS) (2006)
2. Johnson, K.S., Needoba, J.A., Riser, S.C., Showers, W.J.: Chemical sensor networks for the aquatic environment. Chemical Review (2007)
3. Kottapalli, V.A., Kiremidjiana, A.S., Lyncha, J.P., Carryerb, E., Kennyb, T.W., Lawa, K.H., Lei, Y.: Two-tiered wireless sensor network architecture for structural health monitoring. SPIEs 10th Annual International Symposium on Smart Structures and Materials (2003)
4. Szlavecz, K., Terzis, A., Musaloiu-E., R., Cogan, J., Small, S., Ozer, S., Burns, R., Gray, J., Szalay, A.S.: Life under your feet: An end-to-end soil ecology sensor network, database, web server, and analysis service. MSR-TR-2006-90 (2006)
5. Jiang, N., Parashar, M.: Programming support for sensor-based scientific applications. Proceedings of the Next Generation Software (NGS) Workshop in conjunction with the 22nd IPDPS (2008)
6. Klie, H., Bangerth, W., Gai, X., Wheeler, M.F., Stoffa, P., Sen, M., Parashar, M., Catalyurek, U., Saltz, J., Kurc, T.: Models, methods and middleware for grid-enabled multiphysics oil reservoir management. Engineering with Computers, Springer-Verlag (2006)
7. Jiang, N., Parashar, M.: In-network data estimation mechanisms for sensor-driven scientific applications. Proceedings of the 15th International Conference on High Performance Computing (HiPC) (2008)
8. ORBIT: ORBIT testbed. Internet: http://www.orbit-lab.org/
9. JXTA: Project JXTA. Internet: http://www.jxta.org
10. Liu, J., Chu, M., Liu, J., Reich, J., Zhao, F.: State-centric programming for sensor-actuator network systems. IEEE Pervasive Computing **2**(4) (2003) 50–62
11. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. in Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04) (2004)
12. Jiang, N., Schmidt, C., Matossian, V., Parashar, M.: Enabling applications in sensor-based pervasive environments. In: Proceedings of the 1st Workshop on Broadband Advanced Sensor Networks (BaseNets). (2004)
13. Gummadi, R., Gnawali, O., Govindan, R.: Macro-programming wireless sensor networks using *Kairos*. Proceedings of the DCOSS Conference (2005)
14. Bakshi, A., Prasanna, V.K., Reich, J., Larner, D.: The abstract task graph: A methodology for architecture-independent programming of networked sensor systems. Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services, EESR 05 (2005)
15. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TinyDB: an acquisitional query processing system for sensor networks. ACM Transactions on Database System **30**(1) (2005) 122–173
16. Chandrasekaran, S., Cooper, O., Deshpande, A., Fanklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., Shah, M.: TelegraphCQ: Continuous dataflow processing for an uncertain world. In Proceedings of CIDR Conference (2003)
17. Tilak, S., Hubbard, P., Miller, M., Fountain, T.: The ring buffer network bus (RBNB) dataturbine streaming data middleware for environmental observing systems. The 3rd IEEE International Conference on e-Science (2007)