Procedia Computer Science

Volume 29, 2014, Pages 1246–1255

ICCS 2014. 14th International Conference on Computational Science

# An Open Framework for Dynamic Big-Data-Driven Application Systems (DBDDAS) Development

Craig C. Douglas[1,2]

[1] *University of Wyoming, School of Energy Resources, Laramie, WY, U.S.A.*

[2] *King Abdullah University of Science & Technology, SRI –Center NumPor, Thuwal, K.S.A.*

*craig.c.douglas@gmail.com*

**Abstract**

In this paper, we outline key features that dynamic data-driven application systems (DDDAS) have. A DDDAS is an application that has data assimilation that can change the models and/or scales of the computation and that the application controls the data collection based on the computational results. The term Big Data (BD) has come into being in recent years that is highly applicable to most DDDAS since most applications use networks of sensors that generate an overwhelming amount of data in the lifespan of the application runs. We describe what a dynamic big-data-driven application system (DBDDAS) toolkit must have in order to provide all of the essential building blocks that are necessary to easily create new DDDAS without re-inventing the building blocks.

*Keywords:* Big data, DDDAS, dynamic data driven applications, open source software, sensor networks, uncertainty quantification, anomaly detection

## 1 Introduction

Dynamic data-driven application systems (DDDAS) is an application that uses data assimilation that can change the models and/or scales of the computation and wherein the application controls the data collection based on the computational results. The result is more precise predictions and more useful visualizations or results.

Big Data is a paradigm for methods to handle enormous amounts of data that is either streamed, which is the DDDAS preferred method, or that is historically stored datasets that are possibly still growing for data mining. The field of computational sciences in recent years has merged with data intensive computing to become what is referenced as Big Data [1]. Big Data has become a superset of computational sciences with applications pervasive everywhere [2] leading to the interesting question, "What if you had all of the data?" [3].

Technology limits to some extent how big the Big Data can be, but over time the size has increased by 1,000 fold every few years since the 1950's. Table 1 presents what was considered Big Data in earlier eras. There is another aspect of Big Data when time is important: the smaller the time interval,

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.112

the smaller the data size that transforms a problem into a Big Data problem. What makes a problem Big Data is to a degree dependent on data throughput capabilities in a given situation.

Almost all interesting DDDAS cases overlap with Big Data. Solving one solves for the other one, so it makes sense to study both simultaneously. The need for DBDDAS has already emerged in analysis, design, and business, engineering and scientific processes. Resource management, manufacturing process controls, traffic management, systems engineering, civil engineering, weather and climate prediction, geological exploration, social and behavioral modeling, cognitive measurement, and biosensing are areas that already benefit from DDDAS.

| Unit | Approximately | $10^n$ | Related to |
|---|---|---|---|
| Kilobyte (KB) | 1,000 bytes | 3 | Circa 1952 computer memory |
| 32 KB | 32,000 bytes | 4 | NASA Apollo 11 computer memory |
| Megabyte (MB) | 1,000 KB | 6 | Circa 1976 supercomputer memory |
| Gigabyte (GB) | 1,000 MB | 9 | Mid 1980's disk controller memory attached to a mainframe (that had 128 MB memory) |
| 32 GB | 32,000 MB | 10 | 2013 typical smart phone memory |
| Terabyte (TB) | 1,000 GB | 12 | 2012 largest SSD in a laptop |
| Petabyte (PB) | 1,000 TB | 15 | 250,000 DVD's or the entire digital library of all known books written in all known languages |
| Exabyte | 1,000 PB | 18 | 175 EB copied to disk in 2010 (est.) |
| Zettabyte (ZB) | 1,000 EB | 21 | 2 ZB copied to disk in 2011 (est.) |
| 10 Zettabytes | 10,000 EB | 22 | Single NSA Big Dataset in 2013 (est.) |

**Table 1:** What constitutes Big Data has changed over time

Our research in DBDDAS is motivated from the following:

- Obtaining and communicating out of sequence, questionable quality, remote data to local and/or remote computers and using it to steer simulations and answer what if scenarios.
- Clear societal gains from more accurate, long term forecasts from nonlinear, time dependent, complex models.
- Updating the entire DDDAS paradigm using recent Big Data research concepts.

Almost all DDDAS have dealt with Big Data issues throughout the history of DDDAS. Most projects have individually invented Big Data concepts with no field wide consensus on how to handle Big Data. Combining the two paradigms not only makes sense, but is essential to creating new DBDDAS quickly and efficiently. Re-inventing the wheel, so to speak, is bad science once the wheel has been identified.

The remainder of the paper is organized as follows. Section 2 is a review of the DDDAS abd Big Data paradigms. Section 3 is a proposal for an open source toolkit to implement DBDDAS that should make creating a new DDDAS far easier than is currently the case. Section 4 is an example based on the toolkit proposal. Section 5 has conclusions.

# 2 DBDDAS := DDDAS + Big Data Paradigms

DDDAS is related to a number of other concepts that all historically relate back to the original data assimilation concepts first used in the weather and climate modeling communities in the late 1960's.

Some of the concepts are called intelligent data assimilation, steered data assimilation, solution-based engineering science (SBES), cyber-physical sciences (CPS), etc. There are concepts in the sciences, engineering, space sciences, and even funding agency specific terms that are all very similar. Which term is used is irrelevant, what is relevant is the overall concept.

DDDAS is much more than simply adding data assimilation to some application that originally takes static input data and provides an approximation to the solution at some forward time, which is then iterated with more runs to produce a sequence (or movie visually) of predictions in future time. DDDAS extends conventional applications by augmenting them with most or all of the following key points: (1) Model state save, modification, and restoration (through cold or warm restarts). (2) Ensemble data assimilation algorithms to modify ensemble member states by comparing the data with synthetic data of the same kind created from the simulation state (ensemble Kalman filtering or stochastic filtering). (3) Multiscale algorithms with model dependent interpolation methods. (4) Retrieve, filter, and ingest data from intelligent sensors with potentially significant processing capabilities (possibly through a data warehouse mechanism). (5) Visualize computed results on a wide variety of devices including mobile ones such as a tablet or smart phone, laptop, workstation, on up to a 3D immersive room.

Big Data extends conventional applications by adding the following to produce a corresponding DDDAS: (a) Robust, highly distributed file systems capable of managing Big Data applications. (b) Flexible methods for handling large quantities of data in highly parallel computing environments, e.g., Map-Reduce, which is a parallel merge-sort algorithm. (c) Fast, parallel read-write capabilities, e.g., NetCDF or HDF5. (d) Time dependent data retrieval based on dynamically chosen windows in time. Hence, automatic weighting of data based on how recent it was acquired can be easily determined and used. (e) Structured query language (SQL) and Not only SQL (NoSQL) capabilities for storing and retrieving data from (dynamically growing) databases or data streams.

DDDAS requires sensors capable of dynamically supplying data to a simulation. An ideal sensor is sensitive, selective, and communicates high level spatial and other relevant information to the simulation rapidly using negligible bandwidth. Integrated Sensing and Processing (ISP) replaces current sensor designs with DDDAS optimized sensor system architectures that are comprised of interdependent networks of functional elements, each of which spans the roles and functions of multiple separate subsystems in present generation sensor systems. ISP simplifies sensing in DDDAS through spanning required multiple roles and functions. ISP research is developing mathematical tools that facilitate the design and global optimization of systems that interactively unite usually independent functions of sensing, signal processing, communication, and exploitation. ISP reduces crucial degrees of freedom in sensing system design and operation without regard to traditional subsystem limits and interconnect structures. This reduction is realized by applying modern systematic methods from application based computational modeling and fast data adaptive representations to discover and take advantage of any structure present in the data across every stage of the sensor system. In many instances, ISP enables an instantaneous dimensionality reduction to a tractable optimization problem that is far more appropriate to the structure of the problem than the traditional sensing approach.

# 3 OpenDBDDAS Toolkit Components

In this section, we describe a set of components that can be combined and used to construct a DBDDAS conveniently. Similar projects that use this toolkit will have the advantage of being able to easily intermix pieces of their applications to construct better and unified ones. The cost of learning how to make an efficient DDDAS will also decrease. The toolkit must be usable by non-computer scientists.

In Section 3.1, we describe a six point data warehouse system that is key to processing Big Data. In Section 3.2, we describe sensor description languages. In Section 3.3, we describe high performance disk input/output. In Section 3.4, we describe cache aware hashing. In Section 3.5, we describe secure, scalable map-reduce systems. In Section 3.2, we describe a DDDAS toolkit.

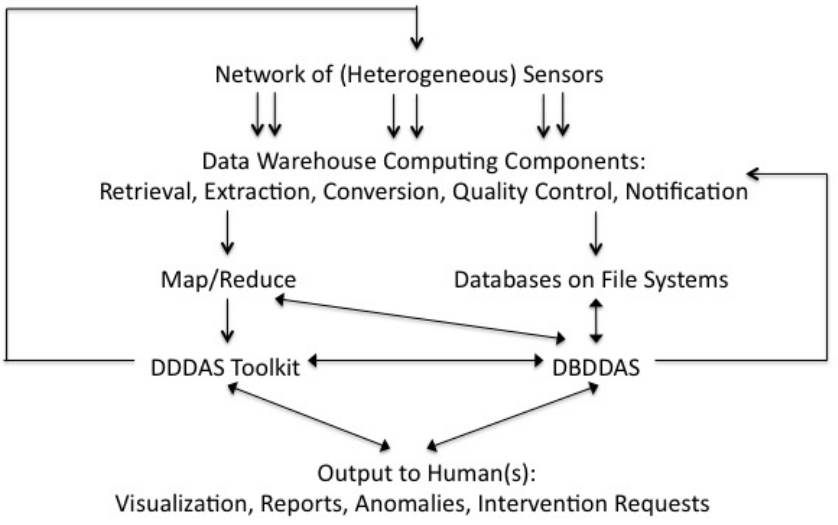


**Figure 1:** Typical use of OpenDBDDAS with a DDDAS

## 3.1 Data Warehouse

Data is stored, possibly temporarily, in a data warehouse. This is simply a comprehensive and very flexible set of computer databases. Data entering goes through up to six steps:

1. *Retrieval*: Get the data from sensors or the growing databases. This may mean receiving data directly from a sensor or database or indirectly through another computer or storage device. There can be network side effects that cause data loss or damage.
2. *Extraction*: The data may be quite messy in its raw form. Thus, the relevant data may have to be extracted from the transmitted information.
3. *Conversion*: The units of the data may not be appropriate for all of the applications using the data warehouse. A consistent set of units is necessary.
4. *Quality control*: If possible, bad data should be removed or repaired and missing or incomplete data should be repaired.
5. *Store*: If the data needs to be archived, it must go to the right medium, which might be permanent or semi-permanent storage. If the data will not be archived, then it must exist only briefly and then be discarded if it is not used during a known and well defined time period.
6. *Notification*: Any simulation using the data must be informed as new data enters the data warehouse, which could necessitate either a cold or warm restart or a new start up.

All of these steps are common in both Big Data and DDDAS. Typically the quantity of data is extremely large and there is a timeliness that must be addressed by the data center so the applications run in (faster than) real-time.

ISP simplifies the data center entry process by performing data extraction and data conversion at the detector inside of the sensor, which eliminates steps 2 and 3 above. Data is delivered from the ISP

as high level information tokens that require almost no extra communication bandwidth. Bad data may be edited, completed, or removed as data are tokenized, potentially eliminating step 4 in the data center, too.

## 3.2 Sensor Description Language

Having a standard sensor description language is an obvious component for a DBDDAS Toolkit. SensorML [4] provides such a standard and provides a framework within which the geometric, dynamic, and observational characteristics of sensors and sensor systems are defined. There are many different sensor types, from simple thermometers to complex electron microscopes and Earth orbiting satellites. SensorML supports the definition of atomic process models and process chains to encode any of these types of sensors. SensorML is part of the Geospatial standards [5]. Unfortunately, while SensorML is well defined, its implementation comes from a commercial source and is anything, but free.

A open source implementation of SensorML (or equivalent) is essential to the DDDAS community agreeing on a single description of essentially all known sensors.

## 3.3 High Performance Disk I/O

What is different between paralle disk and distributed file system?

Choosing one (or more) existing high performance parallel disk input/output systems that does not include a distributed file system is realistic. For example, HDF5 and NetCDF are two candidates that have large followings. In most computing environments that support Big Data, there already is a distributed file system installed and maintained by the computer center staff. For example, Lustre, IBM's GPFS, Google's GFS, and Apache's HDFS are all common.

## 3.4 Cache Aware Hash Table System

The user part of a Map-Reduce typically uses hash tables and hashing. Textbook style implementations (e.g., [6]) are simple codes and have no bugs. However, both Google and Yahoo! developed ideas that incorporate hardware acceleration through clever use of CPU memory caches. That the codes do the same thing as the textbook style codes is not obvious nor are the codes clearly correct. However, the codes are fast and can run up to 150 times faster than the textbook style codes for the right type of hash functions and data.

HIPAA ( Health Insurance Portability and Accountability Act)

## 3.5 ~~HPPA~~ Compliant Map-Reduce

An open source Map-Reduce [7] suitable for medical records that must (a) be HPPA (the U.S. standard for medical record privacy) compliant, (b) scale well for very large databases, (c) have individual user access capabilities, (d) not have complexity O(disk access) on a distributed file system, and (e) detect and report inappropriate uses of the system. (d) means that it should use OpenMP and MPI. Such a Map-Reduce system will be useful well beyond just medical records.

## 3.6 DDDAS Toolkit

The DDDAS Toolkit will consist of at a minimum the following components:

1. Define DDDAS and its runtime requirements abstractly.
2. Uncertainty quantification and statistical analysis.
3. Numerical solvers for standard nonlinear time dependent coupled PDEs and optimization.
4. Fuzzy mathematics.
5. Data assimilation using multiscale interpolation methods and ensemble methods.
6. Item identification, tracking, and steering.

7. Anomaly tracking and verification. [8]
8. Human notification methods.

Many of these components already exist in some form or another. Collecting ones that the DDDAS community will agree to use is a challenge that must be met.

In [9] is a definition of a DDDAS in terms of a 5-tuple: $W$=($S$, $D$, $M$, $A$, $R$), where

- $S = \{s_i\}$ is the set of static inputs and corresponds to what an application that does not have data assimilation capabilities.
- $D = \{d_i\}$ is the set of dynamic inputs that pass through the data warehouse to the DDDAS. The magnitude of $D$ measures how dynamic the application actually is.
- $M = \{m_i\}$ is the set of models and cmayan contain one or more mathematical, multiscale, and scientific specific models.
- $A = \{a_i\}$ is the set of applications, which can be just one application or multiple ones that are typical in multi-physics applications.
- $R = \{r_i\}$ is the set of runtime requirements. In [10] is a detailed description of possible computing, networking, scheduling, and security possibilities that need to be defined for a general purpose DDDAS.

Many DBDDAS have to solve an inverse problem in order to estimate parameters that mathematically define the models used in the application. A small error in estimating the parameters can lead to large errors in the predictions from the models. Quantifying the uncertainties and errors is an emerging field in science and very important to estimating the errors in the predictions [11]. A significant advantage of the DBDDAS paradigm is that the additional dynamic data from sensors and data sources during runtime leads to self correcting processes unavailable to traditional applications just based on static input datasets and vast numbers of samples to measure and quantify uncertainties that in the traditional case requires vast numbers of example runs to acquire far less data.

For any scientific or engineering based DBDDAS, there will be solvers need for the mathematical models. Linear, nonlinear, time stepping, and optimization method solvers must be available in a general purpose toolkit. Additionally, meshing and pre- and post-processing using graphical methods are also necessary in order to prepare initial inputs and to visualize predictions. None of these requirements need to be created anew since software libraries have been developed in this area since the early 1970's.

Many DBDDAS have one or aspects that cannot be represented by single, precise values. For example, a range of values (e.g., the temperature is -3 C to 0 C) might all be that is available and is still useful. Alternately, multivalued logic (versus *true-false* logic) may be necessary. Fuzzy mathematics and logic, respectively, has been developed to deal with both requirements [12].

Data assimilation using multiscale interpolation methods [13] and ensemble methods [14] is essential to creating a DDDAS or DBDDAS. Typically a coarse mesh of sensors is independent of the much finer mesh used to solve mathematical problems inside the application. Interpolation from the coarse mesh onto the much finer mesh is most accurately achieved using a mathematically operator induced multiscale interpolation method. For example, in a typical oil reservoir, there may be a few hundred locations for the sensors and a few million finer mesh points. Standard interpolation is not accurate enough.

Item identification, tracking, and steering is a large field of research with evolving results. The item(s) can be ground, aerospace, or water based vehicles with or without people on board to help with steering.

Trying to identify people with particular psychological properties when passing through airport security is a commercial product that has been deployed by the U.S. Transportation Security Agency that started as a NSF funded DDDAS project. On the other hand, tracking a vehicle in a complex urban environment is still open research. When a vehicle passes through an obscured intersection with other similar moving vehicles, it is not easy to determine which is the vehicle to continue tracking. Vehicle steering is also an active research field.

Finding data anomalies is challenging [8]. An anomaly is defined as any data that is outside of the domain of expected data. Expected data is both application and time dependent within an application. Getting real data for training is nearly impossible for many applications. Noisy data may appear to be an anomaly, but is not. A general purpose anomaly detection library is essential to the Toolkit and is still an active research area.

Automatic human notification methods to provide a human in the loop requires being able to communicate through a number of channels: email, FAX, instant messaging, phone calls using automatically generated voice messages, and visual interruptions on computers monitoring an application. An ordered set of methods must be simple to organize that allows for time constraints for how long each method should be relied on before escalating and which ones should be used in parallel.

The Toolkit consists of quite complicated components that are either already existing or still being researched and constructed. However, to be effective and improve the likelihood of general adoption by application specialists, the Toolkit must be usable by non-computer scientists.

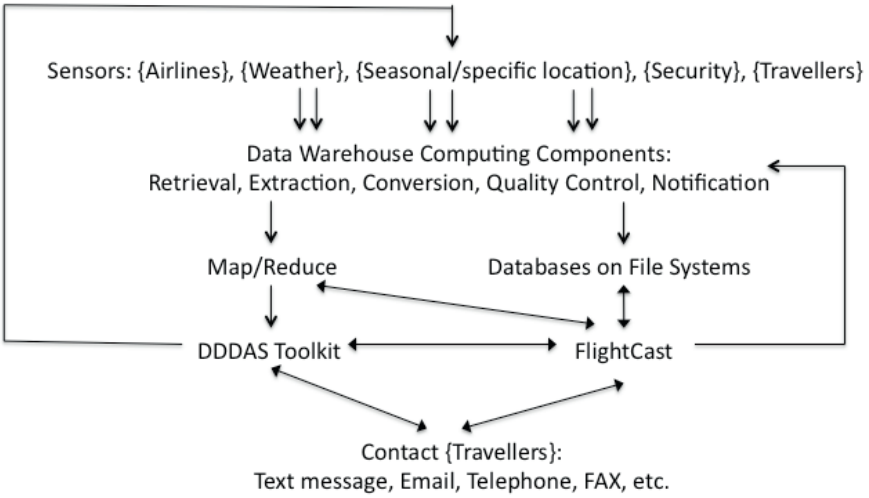


**Figure 2**: FlightCast as it would be implemented using OpenDBDDAS framework

# 4 An Example

FlightCast [15] (see Figure 2) was developed to accurately predict when flight connections would be missed far enough in advance to seamlessly allow a frequent flyer time to get one or more backup itineraries. Approximately 20% of all flights in the U.S. are more than 15 minutes late (the industry standard for determining if a flight is late) or cancelled each year. This statistic has remained remarkably constant over the last four decades for the major airlines that serve more than a small region.

FlightCast tracks the results of airline flights over time and uses this data to accurately predict the probability of either a significant delay or cancellation of a flight. Factors taken into account include

flight information provided by airlines, weather information, terrorism threat level as reported by the U.S. Federal Aviation Administration (FAA), and seasonal and location specific information.

The FlightCast DBDDAS consists of five major types of components:

1. *Sensors*: collect data for the DBDDAS from numerous sources.
2. *Transport*: transfer data between the sensors and the other parts of the DBDDAS through the data warehouse.
3. *Data store*: in the data warehouse for collected data.
4. *Predictor*: predict flight delay and cancellation based on historical data and current conditions.
5. *Corrector*: analyze the accuracy of predictions and make corrections to weights used by the predictor to improve accuracy over time.

Note that component 1 is the top layer in Figure 2, components 2 and 3 are part of the data warehouse described in Section 3.1. Components 4 and 5 are the heart of the DBDDAS and extensively use the DDDAS toolkit functions.

The sensors are software agents. There are agents for travellers, airlines, weather, threat levels, and seasonal trends. The agents were written typically in Python and some required dynamic modifications to keep up with changes in querying methods to online databases maintained by airlines, weather forecasters, and the FAA.

We used extremely low level web requests for very specific information without receiving the standard, extensive, graphics laden, and ad infested web pages in return. We then parsed the returned information using simple state based tools that compilers are frequently constructed from. Cookies played a part in further reducing the amount of data returned by the web sites.

The traveller sensor tracks where the passenger should be before and during the flight(s). It also monitors personal preferences such as the order of choice of airlines, nonstop flights versus connecting ones, maximize or minimize airline frequent flyer qualifying miles on rebooking, and whether or not to optimize arriving at the destination in a timely fashion or significantly late (e.g., in order to reap future travel benefits).

The airline and weather sensors were similar to each other and had to be tailored to specific web sites. The airline sensors were specific to each monitored airline and collected information about flight departure and arrival times, cancellations, and could also collect terminal and gate information, which would be useful in determining the ability to get between flights. The weather sensor collected information about wind speed, wind direction, visibility, temperature, and the type of precipitation.

The seasonal and location specific sensors have a required learning period of at least one year. We used a number of year's worth of weather data specific to a set of airports to preload a database and correlated flight delays and cancellations to it while letting it grow dynamically.

The transport component [16] was Java based software that linked the sensor, predictor, and corrector components. The original transport component worked on serial computers only, but was modified to run on many computers as a GRID computing entity and stored data in SQL based databases.

The predictor combined data from the relevant sensors (airline, weather, threat level, seasonal, and a specific traveller), weighted the data, and produced a probability (from 0 to 1) of either a significant delay or cancellation. Each factor in the probability was produced using fuzzy logic. The factors used in the predictions were stored in the database for use by the corrector component later.

The corrector component reviewed the results of the predictor on a regular basis. It is inexpensive computationally due to the information saved in the prediction component. The main purpose of the corrector is to adjust weights used in the predictor component. Both the predictor and corrector components usually ran multiple times over time before the actual flight(s) and the corrector ran once more after the flight(s).

FlightCast resembles the combined DDDAS and Big Data that we believe is the future of DBDDAS. Building up a database of flight information for just Delta and United airlines over one

year generated enormous amounts of data, even with relatively low sampling rates. After several months of operating FlightCast tracking one particular frequent flyer with a weekly roundtrip commute between New York's Laguardia and Lexington's airports, FlightCast had a better than 95% correct prediction rate. The two airlines tracked had nothing similar to FlightCast and could only react to aircraft delays and maintenance issues on the day of flights plus day before weather related cancellations. To date there is no similar tool to FlightCast available to the general public, much less deployed by any airline. It is still the state of the art DBDDAS in its field.

# 5 Conclusions and Future Work

Most DDDAS have a number of common components:

- Correct sensors need to be chosen to provide good data from specific locations.
- Sensors need to provide data with a known error bound within known time constraints.
- Data is not guaranteed to be accurate, complete, nor timely, but will arrive anyway and must be sensibly processed, including discarding.
- Intelligent sensing and processing (ISP) greatly simplifies data warehouse ingestion.
- Hardware sensors, like software ones, should be dynamically reprogrammable during simulations.
- Rapid error growth in new data when steering is unavailable frequently occurs, causing simulations to be restarted. Automatic new methods to recognize and curtail rapid error growth are essential. New data intensive methods are helpful.
- An application's models, numerical methods, and most significant items that are tracked may need to be changed as a result of the incoming sensor data or a human in the loop.
- Extremely large quantities of input and/or output data that is difficult to analyze.
- Inadequate visualization of output data.

The OpenDBDDAS Toolkit should have at a minimum, the following components: a general definition scheme for a DDDAS and its runtime requirements, uncertainty quantification and statistical analysis, numerical solvers for standard (nonlinear time dependent coupled) PDEs, optimization solvers, data assimilation using multiscale interpolation methods, item identification, tracking, and steering, anomaly tracking and verification, and human notification methods.

Using Big Data analytic techniques, much longer running simulations are possible instead of running a large sequence of short running simulations and then combining each result into a video of predictions. Additionally, data can be better analyzed, visualized, managed, and accessed faster using the Big Data paradigm than ad hoc data management methods created by nonexperts who excel otherwise in application simulators.

The future lies in first getting a consensus among applications specialists who create DDDAS-like or similar intelligent data assimilation application systems with large data on what should be in the initial release of the OpenDBDDAS Toolkit. Cooperation between many research groups who have existing DDDAS is essential. The second key point is to choose existing packages that meet these requirements and to start writing new packages where no acceptable one already exists. The third key point is getting a working example that is both useful and can be a model for creating new DBDDAS is essential.

# Acknowledgments

This research was supported in part by National Science Foundation grant EPS-1135483 and Award No. KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST).

# References


1. Bryant, RE, Katz, RH, Lazowska, ED, Big-Data Computing: Creating revolutionary breakthroughs in commerce, science, and society, Ver. 8, (2008), Computing Research Association, Computing Community Consortium.
2. Hey, T, Tansley, S, Tolle, K, The Fourth Paradigm: Data-Intensive Scientific Discovery, (2009), Microsoft Research.
3. Manyika, J, Chui, M, Brown, B, Bughin, J, Dobbs, R, Roxburgh, C, Byers, AH, Big data: The next frontier for innovation, competition, and productivity, James (2011), McKinsey Global Institute.
4. http://www.opengeospatial.org/standards/sensorml.
5. http://www.opengeospatial.org/ogc/markets-technologies/swe.
6. Aho, AV, Hopcrift, JE, Ullman, JD, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, 1974.
7. Dean, J and Ghemawat, S, MapReduce: Simplified Data Processing on Large Clusters, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004, pp. 137-150.
8. Chandola, V, Banerjee, A, and Kumar, V, Anomaly detection: a survey, ACM Computing Surveys, 41/3 (2009), article 15, 58 pages.
9. Darema, F, *DDDAS Computational Model and Environments*, Journal of Algorithms & Computational Technology, 5 (2011), pp. 545-560.
10. Douglas, CC, Allen, G, Efendiev, Y, and Qin, G, *High performance computing issues for grid based dynamic data-driven applications*, Proceedings of DCABES 2006, Xu, X. and Wang, G. (eds.), Shanghai University Press, Shanghai, (2006), pp. 175-178.
11. Biegler, L, et al, Large-Scale Inverse Problems and Quantification of Uncertainty, John Wiley & Sons, Chichester, 2011.
12. Kerre, EE and Mordeson, JN, A historical overview of fuzzy mathematics, *New Mathematics and Natural Computation*, 1 (2005), pp. 1-26.
13. Efendiev, Y and Hou, TY, Multiscale Finite Element Methods: Theory and Applications, Surveys and Tutorials in the Applied Mathematical Sciences series, 4 (2009), New York.
14. Evenson, G, Data Assimilation: The Ensemble Kalman Filter, Springer, Berlin, 2nd ed., 2009.
15. Hyatt, Jr., R, Bansal, D, Chakraborty, S, Hatcher, J, Lim, C-L, Maynard, CM, Presgrave, T, Douglas, CC, and Haase, G, Flight cast – an airline flight delay predicting DDDAS, Proceedings of DCABES 2007, Guo, Q and Guo, Y (eds.), Hubei Science and Technology Press, Wuhan, 1 (2007), pp. 85-88.
16. Li, W, A Dynamic Data-Driven Application System (DDDAS) Tool for Dynamic Reconfigurable Point-to-Point Data Communication, Master's Thesis, University of Kentucky Computer Science Department, Lexington, KY, (2006).